

Pet showroom

by

Rakshanda Reddy Parwath

B. Tech., Sri Indu College of Engineering and Technology, 2017

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computer Science  
College of Engineering

KANSAS STATE UNIVERSITY  
Manhattan, Kansas

2019

Approved by:

Major Professor  
Dr. Daniel Andresen

# **Copyright**

© Rakshanda Reddy Parwath 2019.

## **Abstract**

Many people in the United States own one or more pets. Pets reduce our stress levels. One's desired breed could be difficult to find in a nearby store. The main aim of this web development project is to make a pet store online where there will be a huge number of pets of various breeds and pet accessories which can be bought online and get them delivered at the doorstep. This web application gives a lot of options to choose from. It also contains information about the different types of breeds and suggests names for each breed. Having pets is a fun thing but taking care of their health is also important. This website provides different nutrition and care products required by pets based on a particular breed. This website makes it easy to buy all the items for a pet and maintain them at one place. This website would be a good stop for pet lovers.

# Table of Contents

List of Figures	vi
List of Tables	vii
Acknowledgements	viii
Chapter 1 - Introduction	1
Description	1
Motivation	1
Chapter 2 - Technologies Used	2
Django	2
Python	2
HTML	3
CSS	3
Chapter 3 - System Analysis and Requirements	4
Scope of the Project	4
Requirements	4
Hardware Requirements for developing the project	4
Software Requirements for developing the project	4
Requirements for clients	4
Functional Requirements	38
Chapter 4 - System Design	6
System Architecture	6
Data Flow Diagram	7
Sequence Diagram	8
Use Case Diagram	10
Database in django	11
Database structure	11
Chapter 5 - System Implementation	13
Modules	13
Admin	13
Users	13
Products	15

Cart	17
Chapter 6 - Results	18
Home Page	18
Login	19
Registration	19
Profile	21
After uploading an image	22
Modules	23
Pets	23
Searching in Pets	24
Accessories	25
Nutrition	25
Cart	27
Orders	27
Chapter 7 - Security and Exception Handling	28
Security	28
Exception Handling	29
Chapter 8 - Testing and Evaluation	31
Unit Testing	31
Correctness Testing	31
Performance Testing and scalability	33
What did i learn?	34
Chapter 9 - Accessibility of the Application	35
Color blind people accessibility	35
Elder people accessibility	35
Chapter 10 - Conclusion	36
Chapter 11 - Future Work	37
Chapter 12 - Bibliography	38

## List of Figures

Figure 1 System Architecture	6
Figure 2 Data flow diagram of the system	6
Figure 3 Sequence diagram of the system	6
Figure 4 Use Case diagram of the system	6

## List of Tables

Table 1 Test cases for correctness testing	6
Table 2 Scalability of the products	6

## **Acknowledgements**

I express my gratitude to my Advisor and Major Professor, Dr. Daniel Andresen for providing valuable guidance, support and trust at every stage of this project.

I am thankful to the professors in my committee, Dr. Torben Amtoft and Dr. Mitchell Nielsen for taking the time to serve on my committee and for continuous support.

With great pleasure, i want to take this opportunity to express my gratitude towards my family and friends who helped on testing my project as remote users.



# Chapter 1 - Introduction

## Description

This is an e-commerce website providing pets, their accessories and nutrition products. The website has different features for authenticated user and a guest user. Any user can register on this website. Once registered, user can login to the website. Any user can browse through the products on the website. But only authenticated users have access to the cart and profile. Authenticated users can add picture to their profile and add their desired products to the cart. After confirming the order, they can see the orders in the order page.

The purpose of this website is to provide user friendly pet store online. The target audience are pet lovers.

## Motivation

There are people who have pets and those pets give birth to more. Most of the pet owners do not have a place to sell to and they always compromise for lowest prices or sometimes free as they give them to their friends or relatives. It is difficult to do marketing of unexpected baby pets. I wanted to help those pet owners to be able sell their pets at the market price.

## Chapter 2 - Technologies Used

**Django** (<https://docs.djangoproject.com/en/2.1/>)

Django is a python-based free and open-source web framework. This framework follows MVT(model view template) architectural pattern. It consists of ORM(object relational mapper) that mediates between data models and a relational database. It also has a system for processing HTTP requests with a web templating system, can be called as views and a regular expression based URL dispatcher which is the controller. When creating superuser in Django, it tells us if the password is too common and asks if we want to bypass validation and create user anyway. To change any column or its datatype we have to change it in the models of the particular app. After modifying the models, we have to do migrations to view the changes. The following commands need to be run in order to do the migrations.

```
`python manage.py makemigrations`
```

```
`python manage.py migrate`
```

Django automatically encrypts the passwords in the database. In Django, there are templates which are written using html but are in a different format compared to html used in other frameworks.

### **Python**

Python is a high-level, interpreted programming language used for general purposes. It features dynamic type system and automatic memory management. It supports multiple programming paradigms, namely object-oriented, imperative, functional and procedural.

## **HTML**

HTML(Hypertext Markup Language) is the standard markup language for creating web pages mostly on the front end. All the html elements are delineated by tags, written as angular brackets. It is one of the World Wide Web Technologies.

## **CSS**

CSS is a stylesheet language used for describing the presentation of a document which is written with HTML. Normally, a website is designed with CSS.

## Chapter 3 - System Analysis and Requirements

### Scope of the Project

The scope of this project is to let the pet lovers enjoy the online pet store where the user can find pets, their accessories and nutrition products. The motive of this project is to make the user experience lot easier to interact.

### Requirements

#### Hardware Requirements for developing the project

- Hard Disk : 40 GB
- RAM : 1 GB

#### Software Requirements for developing the project

- Operating System : Mac OS/ Windows
- Web Technologies : Python, JavaScript, HTML, CSS
- Database : MySQL
- Database browser : SQLite
- Framework : Django
- IDE : Pycharm

#### Requirements for clients

- Browser : Chrome/Mozilla Firefox etc.

#### Functional Requirements

1. All users have access to browse the products through out the website.
2. Any user can register in the website.
3. Any registered user can login into the website.
4. If a password set by user is not strong, the system does not accept it.

5. When a user logs in with a wrong username or password it gives a flash message to enter correct username or password.
6. Every authenticated user has a profile with a default picture.
7. The authenticated user can change their profile picture and personal details on the profile page.
8. All the product modules should have search field. This search field should give results even when there is only some part of the name.
9. Authenticated users can add any desired product to the cart, remove the products from the cart and confirm the order.
10. Only authenticated users have access to the cart, profile and orders pages.
11. When a user manipulates the url and changes the product id. If the product id is not in the database it redirects the user to the page not found error.
12. Admin can add and delete the products from the database.
13. Admin can view the orders confirmed by the user.

## Chapter 4 - System Design

### System Architecture:

System architecture is the conceptual model that defines the structure, behavior and views of the system. It can be viewed as architectural description/representation of the system.

The following system architecture of the project shows that the first step for the user is to register, the user can browse the products with and without logging in. The next step, would be to login in order to select the desired products. The users can have access to the profile only when they are logged in. The desired products are added to cart by the authenticated user. After adding the items to the cart, the user can confirm the order and logout of the website.

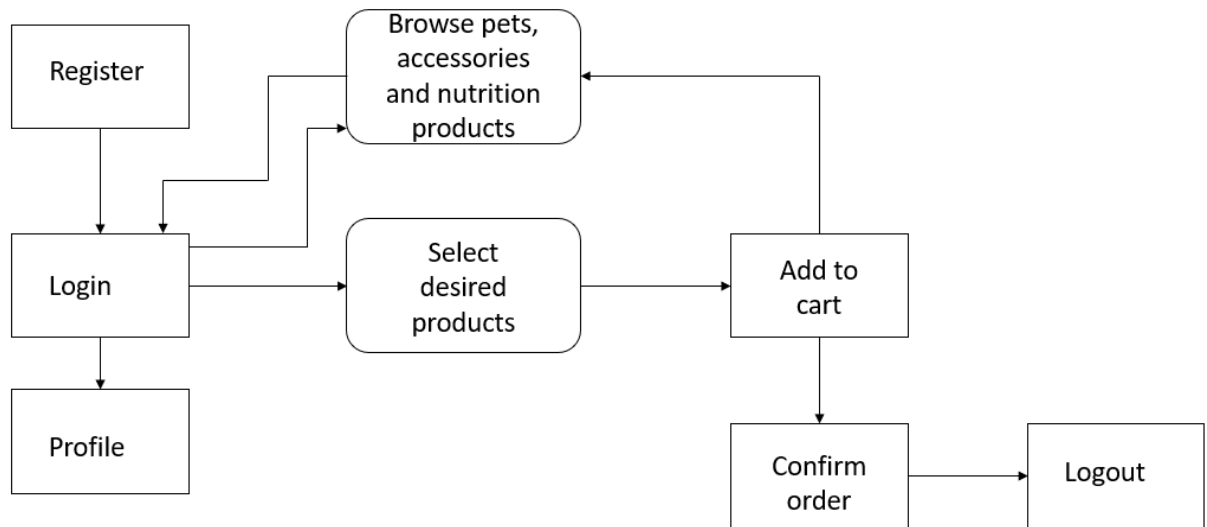


Figure 1 System Architecture

## Data Flow diagram:

Data flow diagram represents the flow of data of the system. It provides the information of the outputs and inputs of each entity and their processes.

The below is the data flow diagram of the project which shows how the data flow is between user, admin and the system. Admin has the authority to add and delete the products and view the orders that are confirmed by the user. These changes take place in the system. A remote user when registered in the system, all the details will be stored in the system. The remote user requests for the product details from the system and as a response, system displays the details of the system.

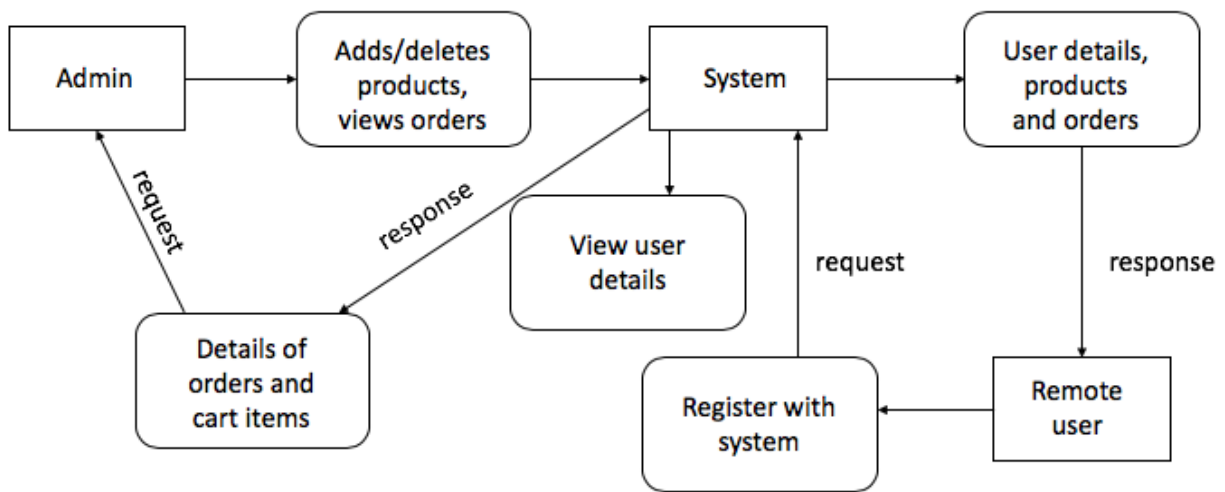


Figure 2 Data Flow diagram of the system

## **Sequence Diagram:**

A sequence diagram depicts interaction between objects in sequential order. It describes how and in what order the objects in the system function.

The following is the sequence diagram of the project which shows the order of the functions done in the project. A user registers with the system and creates a profile. All the user details provided are stored in the database which are viewable by the admin. The admin can add or delete the products on the website. The user browses through the products on the website and selects the desired products. The admin checks all the availability of the products, if a product is not available, then the admin can temporarily delete that particular item. All the desired products will be added to the cart by the user. The user can confirm the order once all the products are added to the cart. These orders will be viewed by the admin.



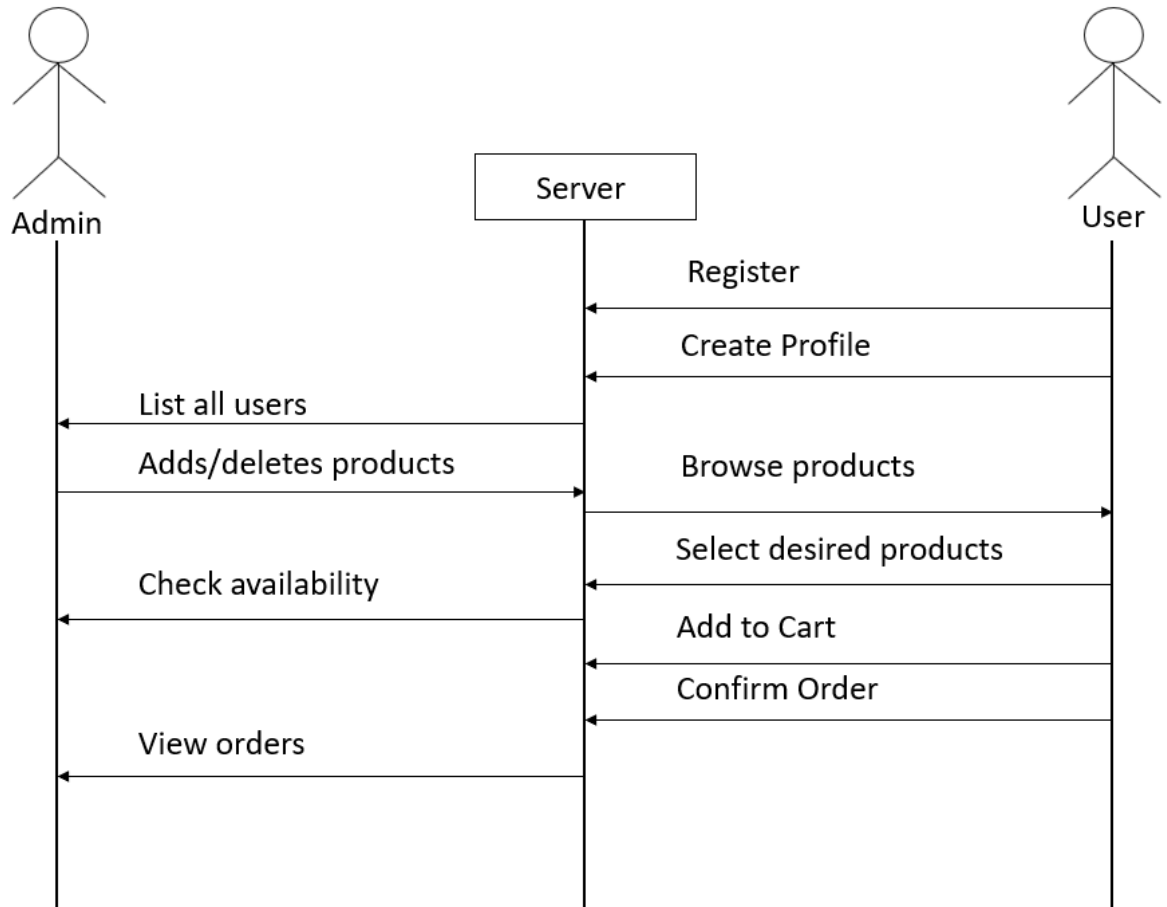


Figure 3 Sequence diagram of the system

### Use Case diagram:

Use case diagram models the functionality of a system using actors and set of actions, services and functions the system performs.

The following figure is the use case diagram of the system. It depicts the actions of user and admin and the system functions. The actions that can be done by both user and admin are login, change his/her profile, browse data throughout the website, select desired products and add to cart, confirm the order. The actions that can only be done by the admin are, Adding and deleting the products from the database and view the orders that are confirmed by the users. The user can perform one action that cannot be performed by the admin which is registering in the website.

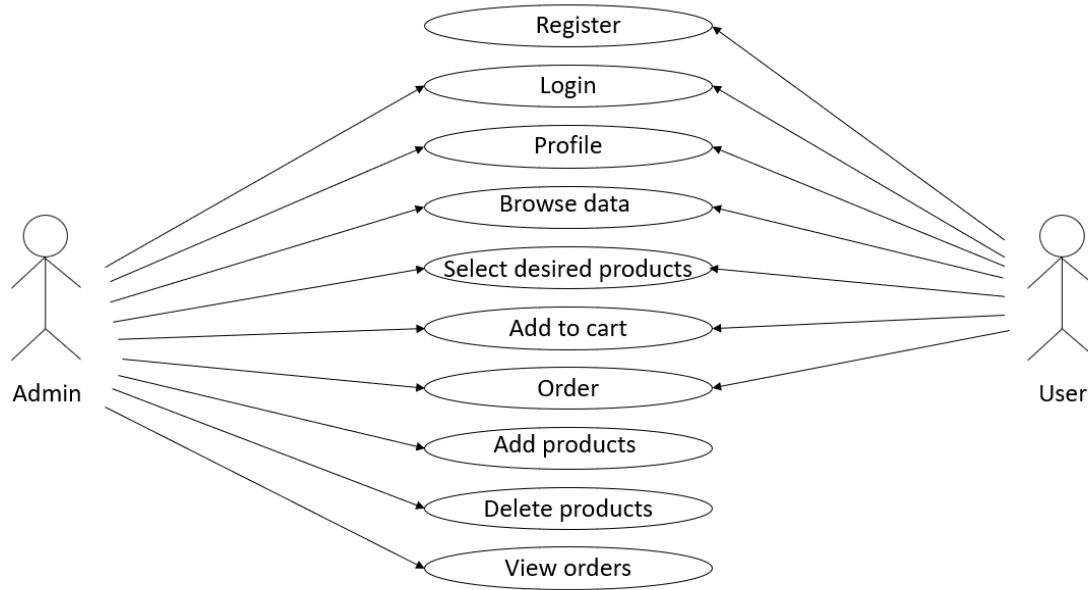


Figure 4 use case diagram of the system

## Database in django:

Tables in django are created in models.py. The class is a Table name and the variables inside it are the column names and they are given datatypes in the models.py. The data can be entered through the user or directly in the database.

The following picture is the pets table created in the models.py.

```
class Pets(models.Model):
    name = models.CharField(max_length=150)
    slug = models.SlugField(max_length=150, null=True, blank=True)
    description = models.TextField(null=True, blank=True)
    image = models.ImageField(upload_to='pets', null=True, blank=True)
    price = models.DecimalField(max_digits=10, decimal_places=2, null=True, blank=True)
    stock = models.PositiveIntegerField(null=True, blank=True)
    available = models.BooleanField(default=True)
    created = models.DateTimeField(auto_now_add=True, null=True, blank=True)
    updated = models.DateTimeField(auto_now=True, null=True, blank=True)

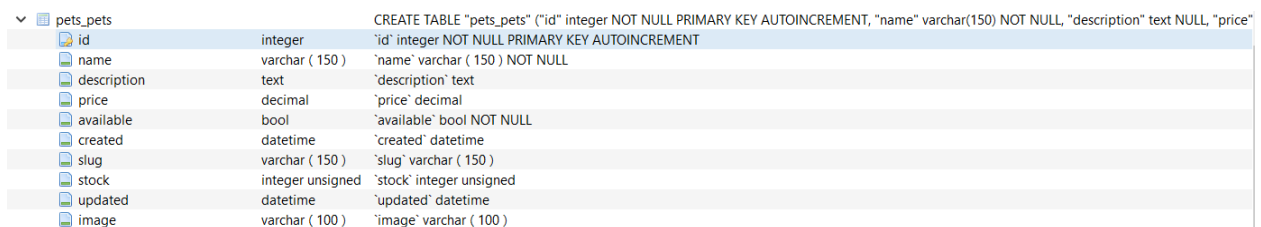
    class Meta:
        ordering = ('-created',)
        index_together = (('id', 'slug'),)

    def __str__(self):
        return self.name

    def get_absolute_url(self):
        return reverse('pets:pets_detail', args=(self.pk, ))
```

## Database structure

The following picture shows the pets table in SQLite browser.



Field Name	Django Type	SQLite Type
id	integer	integer NOT NULL PRIMARY KEY AUTOINCREMENT
name	varchar ( 150 )	'name' varchar ( 150 ) NOT NULL
description	text	'description' text
price	decimal	'price' decimal
available	bool	'available' bool NOT NULL
created	datetime	'created' datetime
slug	varchar ( 150 )	'slug' varchar ( 150 )
stock	integer unsigned	'stock' integer unsigned
updated	datetime	'updated' datetime
image	varchar ( 100 )	'image' varchar ( 100 )

The following picture has the authenticated users tables which are built in features of django.

Tables (22)	
> auth_group	CREATE TABLE "auth_group" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "name" varchar(80) NOT NULL UNIQUE)
> auth_group_permissions	CREATE TABLE "auth_group_permissions" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "group_id" integer NOT NULL REFERENCES "auth_group", "permission_id" integer NOT NULL REFERENCES "auth_permission")
> auth_permission	CREATE TABLE "auth_permission" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "content_type_id" integer NOT NULL REFERENCES "django_content_type", "codename" varchar(100) NOT NULL)
> auth_user	CREATE TABLE "auth_user" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "password" varchar(128) NOT NULL, "last_login" datetime NULL, "is_superuser" boolean NOT NULL, "username" varchar(150) NOT NULL UNIQUE, "first_name" varchar(30) NULL, "last_name" varchar(30) NULL, "email" varchar(254) NULL)
> auth_user_groups	CREATE TABLE "auth_user_groups" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "user_id" integer NOT NULL REFERENCES "auth_user", "group_id" integer NOT NULL REFERENCES "auth_group")
> auth_user_user_permissions	CREATE TABLE "auth_user_user_permissions" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "user_id" integer NOT NULL REFERENCES "auth_user", "permission_id" integer NOT NULL REFERENCES "auth_permission")

The following picture has the tables that are created dynamically through models.py.

> carts_cart	CREATE TABLE "carts_cart" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "quantity" integer NOT NULL, "accessory_id" integer NULL REFERENCES "pets_accessories", "cart_id" integer NOT NULL REFERENCES "carts_order")
> carts_order	CREATE TABLE "carts_order" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "cart_id" integer NOT NULL REFERENCES "carts_cart", "date_created" datetime NOT NULL, "total_price" decimal(10,2) NOT NULL)
> django_admin_log	CREATE TABLE "django_admin_log" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "action_time" datetime NOT NULL, "object_id" text NULL, "object_repr" text NULL, "action_flag" smallint NOT NULL, "change_message" text NULL)
> django_content_type	CREATE TABLE "django_content_type" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "app_label" varchar(100) NOT NULL, "model" varchar(100) NOT NULL)
> django_migrations	CREATE TABLE "django_migrations" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "app" varchar(255) NOT NULL, "name" varchar(255) NOT NULL)
> django_session	CREATE TABLE "django_session" ("session_key" varchar(40) NOT NULL PRIMARY KEY, "session_data" text NOT NULL, "expire_date" datetime NOT NULL)
> pets_about	CREATE TABLE "pets_about" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT)
> pets_accessories	CREATE TABLE "pets_accessories" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "name" varchar(150) NOT NULL, "description" text NULL, "price" decimal(10,2) NOT NULL)
> pets_nutrition	CREATE TABLE "pets_nutrition" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "name" varchar(150) NOT NULL, "description" text NULL, "price" decimal(10,2) NOT NULL)
> pets_pets	CREATE TABLE "pets_pets" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "name" varchar(150) NOT NULL, "description" text NULL, "price" decimal(10,2) NOT NULL)
> sqlite_sequence	CREATE TABLE sqlite_sequence(name,seq)
> users_profile	CREATE TABLE "users_profile" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "image" varchar(100) NOT NULL, "user_id" integer NOT NULL REFERENCES "auth_user")

# Chapter 5 - System Implementation

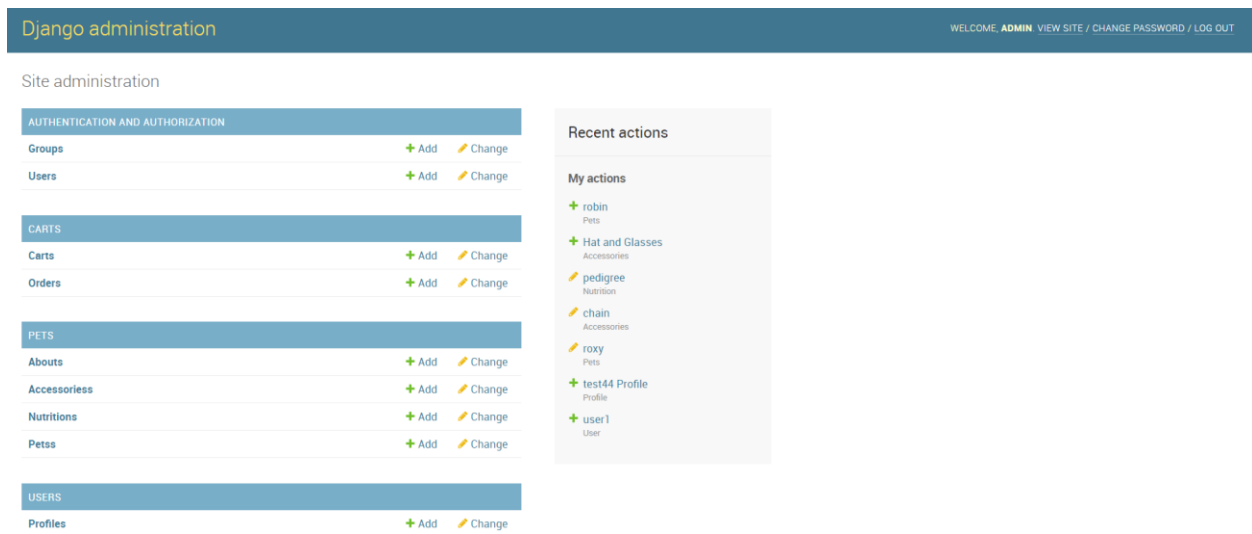
## Modules

### Admin

Admin account has control over the data of the website and can add or delete the products, view user list and their orders. Admin is created using the following commands.

```
`python manage.py createsuperuser`
```

After running this command, it will ask for username, email address and password. These username and password should be used as admin credentials.



### Users

Users module consists of all the users that are registered with the website. It also has user profile which has user picture, email and other details.

In the below image, 'register' method is to register a user in the system. The form to acquire the details from the user is passed into the template through which the user can enter their details, the submitted form by the user is passed to this method and it checks if the form is valid, if not it

shows the message about what is invalid and redirects to register form. If the register form is valid, then the system redirects the user to login page so that they can login and use the website. The 'profile' method allows the user add an image to their profile and update the existing information. This method has two forms, namely, UserUpdateForm and ProfileUpdateForm. These forms are passed into the template and are visible to the user where they can add or update the information, which are again passed into this method when they click on update button. These forms are again checked if they are valid and returns a message saying that their account has been updated. (<https://docs.djangoproject.com/en/2.2/ref/request-response/>)

```
def register(request):
    if request.method == 'POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            form.save()
            username = form.cleaned_data.get('username')
            messages.success(request, f'Your account has been created for {username}! You can now log in')
            return redirect('login')
    else:
        form = UserRegisterForm()
    return render(request, 'users/register.html', {'form': form})

@login_required
def profile(request):
    if request.method == 'POST':
        u_form = UserUpdateForm(request.POST, instance=request.user)
        p_form = ProfileUpdateForm(request.POST,
                                   request.FILES,
                                   instance=request.user.profile)
        if u_form.is_valid() and p_form.is_valid():
            u_form.save()
            p_form.save()
            messages.success(request, f'Your account has been updated')
            return redirect('profile')
    else:
        u_form = UserUpdateForm(instance=request.user)
        p_form = ProfileUpdateForm(instance=request.user.profile)

    context = {
        'u_form': u_form,
        'p_form': p_form,
    }

    return render(request, 'users/profile.html', context)
```

## **Products**

Products are divided into three modules, namely, pets, accessories and nutrition. Pets module has all the pets that are available and are ready to be purchased. Accessories module has all the pet accessories that are available and are ready to be purchased. Nutrition module has all the pet nutrition and care products that are available and are ready to be purchased.

Below image has the backend code of the home page and pets module. IndexView is the home page. It takes model Pets which is the pets table in the database. It is connected with the template index which is the frontend that the user sees on the website. IndexView shows the sending of data from the database to template(html) which can be seen by the user. The same is being done in the PetsView which can be seen in the pets module. PetsDetailView shows the backend code of what happens when the user clicks on any product card. When a user clicks on a product card, that product id is fetched from the database and is stored in a variable and it is passed to the template which shows the details of the product to the user.

```

class IndexView(generic.ListView):
    model = Pets
    template_name = 'pets/index.html'

    def get_context_data(self, **kwargs):
        context = super(IndexView, self).get_context_data(**kwargs)
        context['pets'] = Pets.objects.all()
        return context

class PetsView(generic.ListView):
    model = Pets
    template_name = 'pets/pets.html'

    def get_context_data(self, **kwargs):
        context = super(PetsView, self).get_context_data(**kwargs)
        context['pets'] = Pets.objects.all()
        return context

class PetsDetailView(generic.DetailView):
    model = Pets
    template_name = 'pets/pets_detail.html'

    def get_context_data(self, **kwargs):
        context = super(PetsDetailView, self).get_context_data(**kwargs)
        pet = get_object_or_404(Pets, pk=self.kwargs['pk'])
        context['pet'] = pet
        return context

```



## Cart

Any desired product that is added to the cart will be stored in the cart. When all the items are added to the cart. Then the user can confirm the order.

Below image is to show the functionality of the cart, 'add\_to\_cart' method is used to add the selected product of the user to the cart. When 'Add to cart' button is clicked, it passes that product id and this method is called. When this method is called, it fetches the product details from the database based on the id passed when clicked on the button. Those products are added to the cart of the user. The 'remove\_from\_cart' method is also called in the same way. It fetches that particular id from the database and deletes those products from the cart. The 'get\_cart' method displays the cart page on the website. (<https://docs.djangoproject.com/en/2.2/topics/db/queries/>)

```
@login_required
def add_to_cart(request):
    pet_id = request.GET.get('pet_id', None)
    accessory_id = request.GET.get('accessory_id', None)
    nutrition_id = request.GET.get('nutrition_id', None)

    kwargs = {'quantity': 1}

    if pet_id:
        kwargs['pet'] = Pets.objects.get(pk=pet_id)
    if accessory_id:
        kwargs['accessory'] = Accessories.objects.get(pk=accessory_id)
    if nutrition_id:
        kwargs['nutrition'] = Nutrition.objects.get(pk=nutrition_id)

    cart = Cart.objects.filter(Q(pet__pk=pet_id) | Q(accessory__pk=accessory_id) | Q(nutrition__pk=nutrition_id)).first()
    if cart:
        cart.add(user=request.user, **kwargs)
    else:
        cart = Cart.objects.create(user=request.user, **kwargs)
        cart.save()
    return render(request, 'pets/add_cart.html')

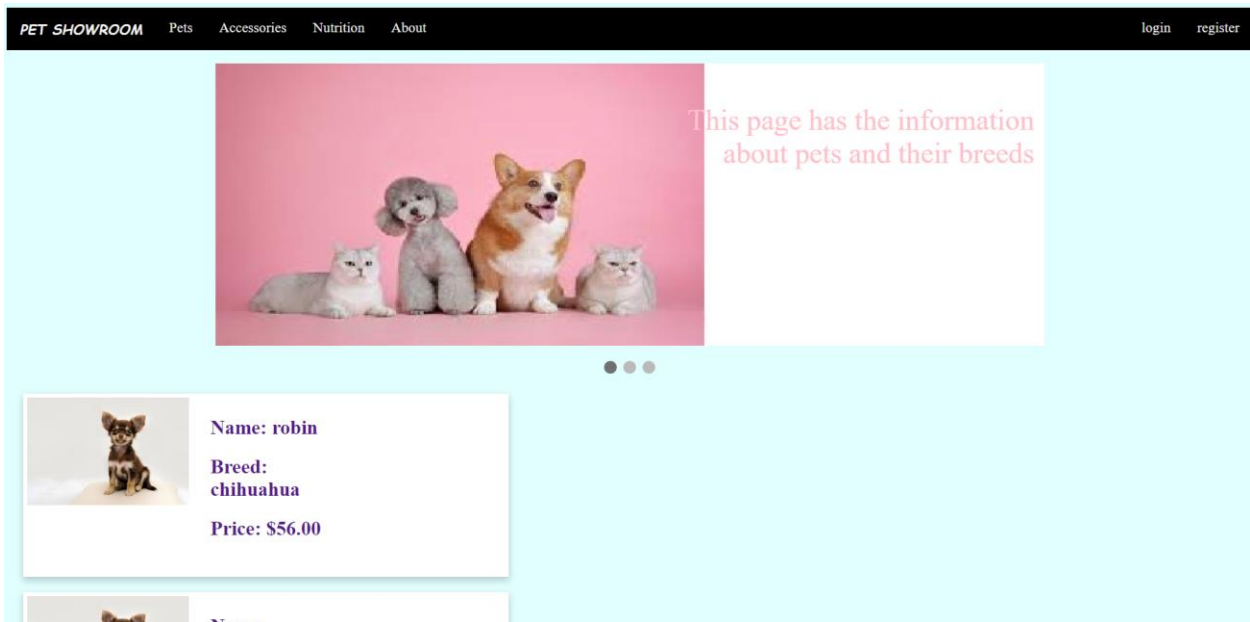
@login_required
def remove_from_cart(request):
    cart_id = request.GET.get('cart_id', None)
    cart = Cart.objects.get(id=cart_id)
    if cart:
        cart.remove(cart)
    return render(request, 'pets/remove_cart.html')

def get_cart(request):
    return render(request, 'pets/cart.html', {'carts': Cart.objects.filter(user=request.user)})
```

# Chapter 6 - Results

## Home Page

Following image is the home page of the website. It has the top navigation bar with different modules. PET SHOWROOM is the hyperlink to the home page which redirects to this page. And the other modules are explained below. The website has a background image of pets. There is a slideshow of three modules of this website, namely, Pets, Accessories and Nutrition. By clicking on this slideshow or the modules present on the top navigation redirects to their respective pages. Below are the cards of the products, on clicking these cards redirects to their detail view which is also present below.



## Login

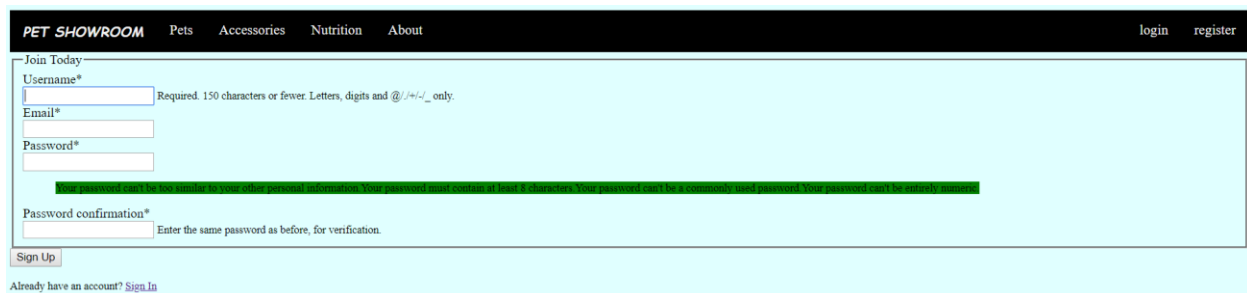
This page is accessed by the users when they are not logged in. And only authenticated users can login to the website. Below it says that 'Don't have an account? Sign Up'. On clicking sign up, the user will be redirected to register page.



The screenshot shows the login interface for the PET SHOWROOM website. At the top, a black navigation bar contains the site name "PET SHOWROOM" and menu items: "Pets", "Accessories", "Nutrition", and "About". On the right side of the navigation bar are links for "login" and "register". Below the navigation bar is a light blue login form. The form is titled "Log In" and contains two input fields: "Username\*" with the value "admin" and "Password\*" with masked characters "\*\*\*\*\*". A "Login" button is positioned below the password field. At the bottom of the form, there is a link: "Don't have an account? [Sign Up](#)".

## Registration

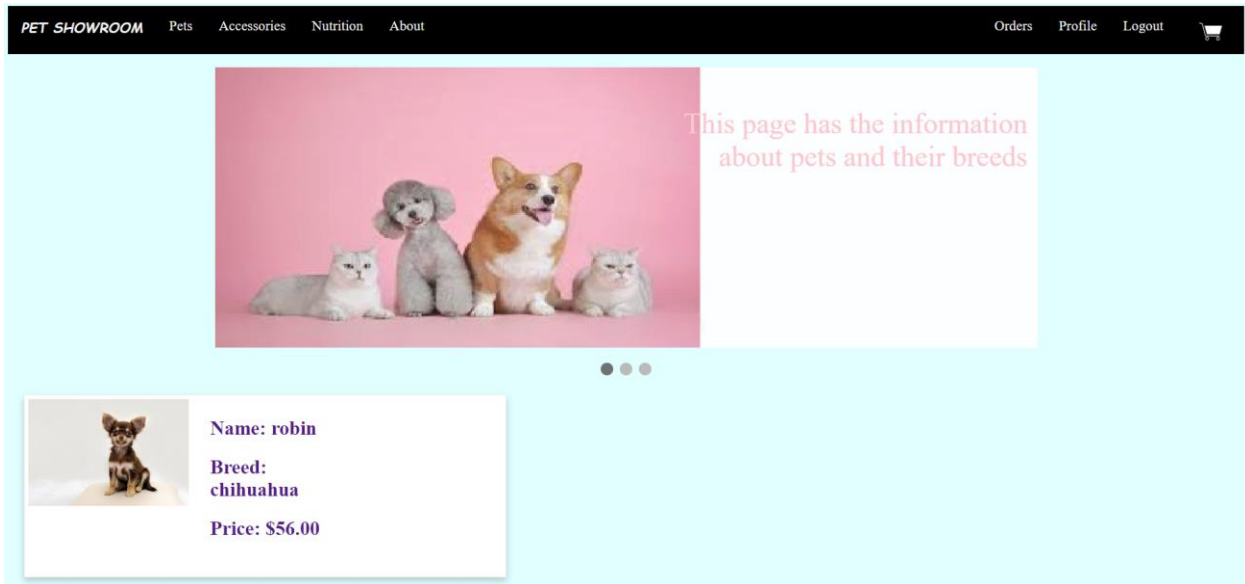
This page can only be accessible when the user is not logged in. It has 4 fields, username, email, password and password confirmation. It has form validation, when the two passwords fields are not the same, it gives a message that they are not same. If the email is not in email format, it gives a message that the email is not in the correct format. Below, it has Sign up button which sends the user details into the database. There is 'Already have an account? Sign In'. On clicking sign in, the user will be redirected to login page. Even after registering, the user is redirected to login page.



The screenshot shows the registration interface for the PET SHOWROOM website. At the top, a black navigation bar contains the site name "PET SHOWROOM" and menu items: "Pets", "Accessories", "Nutrition", and "About". On the right side of the navigation bar are links for "login" and "register". Below the navigation bar is a light blue registration form. The form is titled "Join Today" and contains four input fields: "Username\*" with a validation message "Required. 150 characters or fewer. Letters, digits and @/+/./\_ only", "Email\*", "Password\*", and "Password confirmation\*" with a validation message "Enter the same password as before, for verification." A "Sign Up" button is positioned below the password confirmation field. At the bottom of the form, there is a link: "Already have an account? [Sign In](#)".

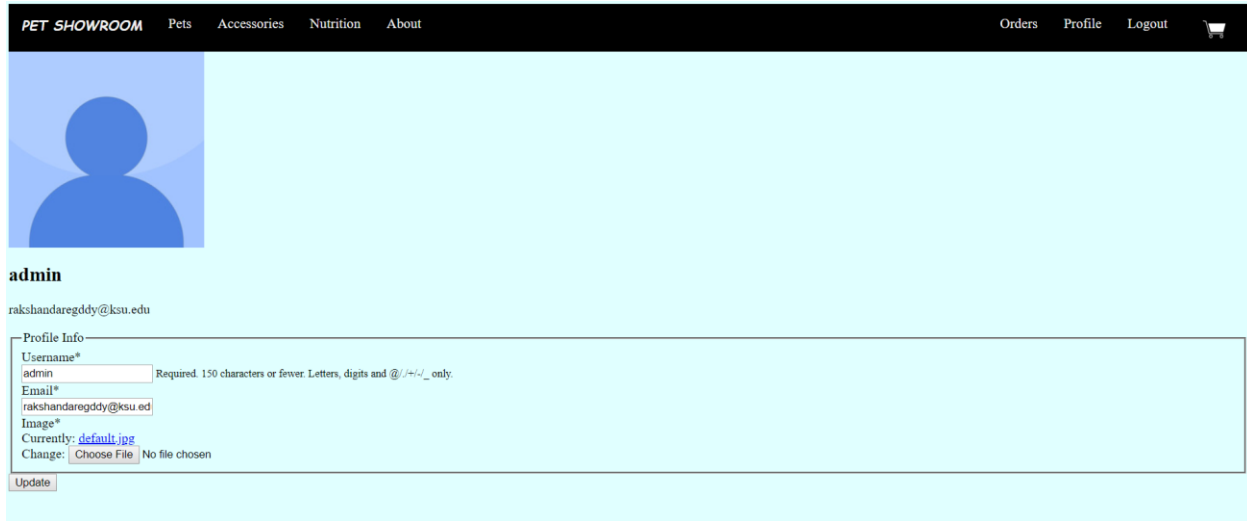
## After logging in

In the top navigation bar, on the right side it was login register when the user is not logged in. Once the user is logged in, it is changed to orders, profile and logout. Below image is the home page for the logged in users.



## Profile

When a user is logged in they have access to profile page. This page can be used to update the user information. Profile has a default image if the user has not uploaded their picture. When they update any of the details in the below forms. It displays a message saying that the account has been updated.




The screenshot shows a web application interface for a user profile. At the top, there is a dark navigation bar with the text "PET SHOWROOM" on the left and "Orders Profile Logout" on the right, accompanied by a shopping cart icon. Below the navigation bar, the profile page has a light blue background. On the left side, there is a circular profile picture placeholder with a blue silhouette of a person. To the right of the profile picture, the name "admin" is displayed in bold, followed by the email address "rakshandaregddy@ksu.edu". Below this information, there is a form titled "Profile Info" with a border. The form contains the following fields and labels: "Username\*" with the value "admin" and a note "Required. 150 characters or fewer. Letters, digits and @/+/./\_ only."; "Email\*" with the value "rakshandaregddy@ksu.edu"; "Image\*" with the value "default.jpg"; and "Change:" with a "Choose File" button and the text "No file chosen". At the bottom left of the form, there is an "Update" button.

## After uploading an image

Below image is the profile page after updating the profile picture. Here, it displays a message saying that the account has been updated.

**PET SHOWROOM** Pets Accessories Nutrition About Orders Profile Logout

Your account has been updated



**admin**  
rakshandareddy@ksu.edu

Profile Info

Username\*  
admin Required: 150 characters or fewer. Letters, digits and @/./+/\_ only

Email\*  
rakshandareddy@ksu.edu

Image\*  
Currently: [profile\\_pic:download.jpg](#)  
Change: [Choose File](#) No file chosen

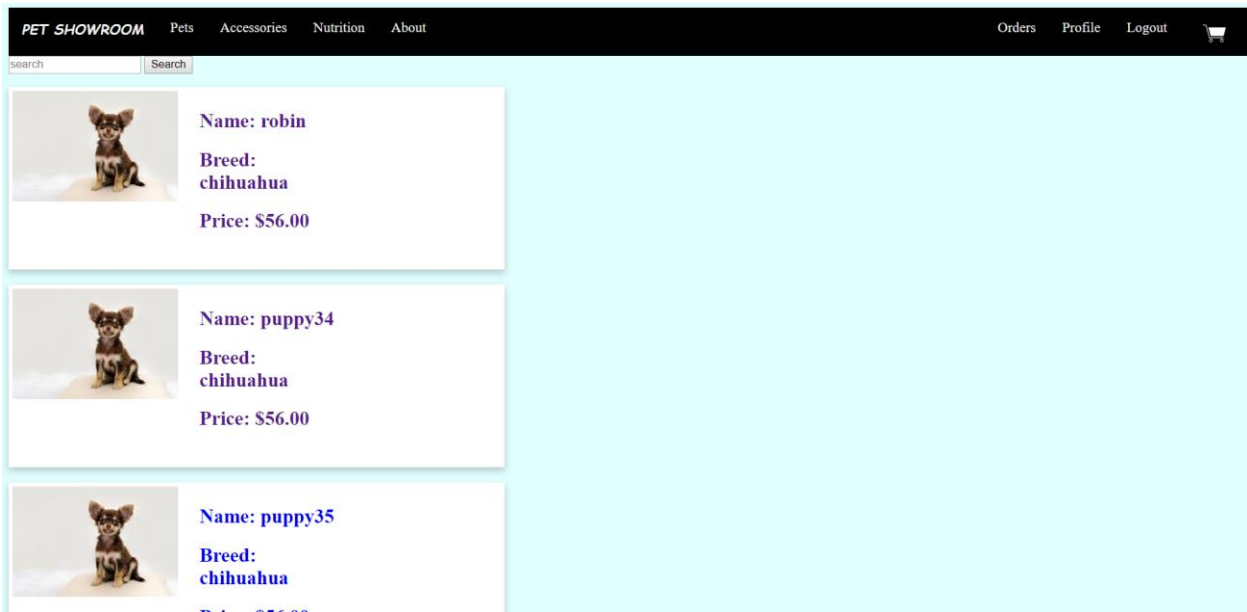
Update

## Modules

All the users have access to browse through out the website. There are three different pages a user can choose products from. They are:

## Pets

User can find all the pets on this page to choose from. There is a search field present on this page which allows the user to type in the breed of the pet and choose accordingly.



The screenshot displays the 'PET SHOWROOM' website interface. At the top, there is a navigation menu with links for 'Pets', 'Accessories', 'Nutrition', and 'About'. On the right side of the header, there are links for 'Orders', 'Profile', and 'Logout', along with a shopping cart icon. Below the navigation, a search bar is visible with the text 'search' and a 'Search' button. The main content area features three product listings, each consisting of a small image of a Chihuahua puppy and a text box containing the following information:

- Name:** robin
- Breed:** chihuahua
- Price:** \$56.00

The second listing shows:

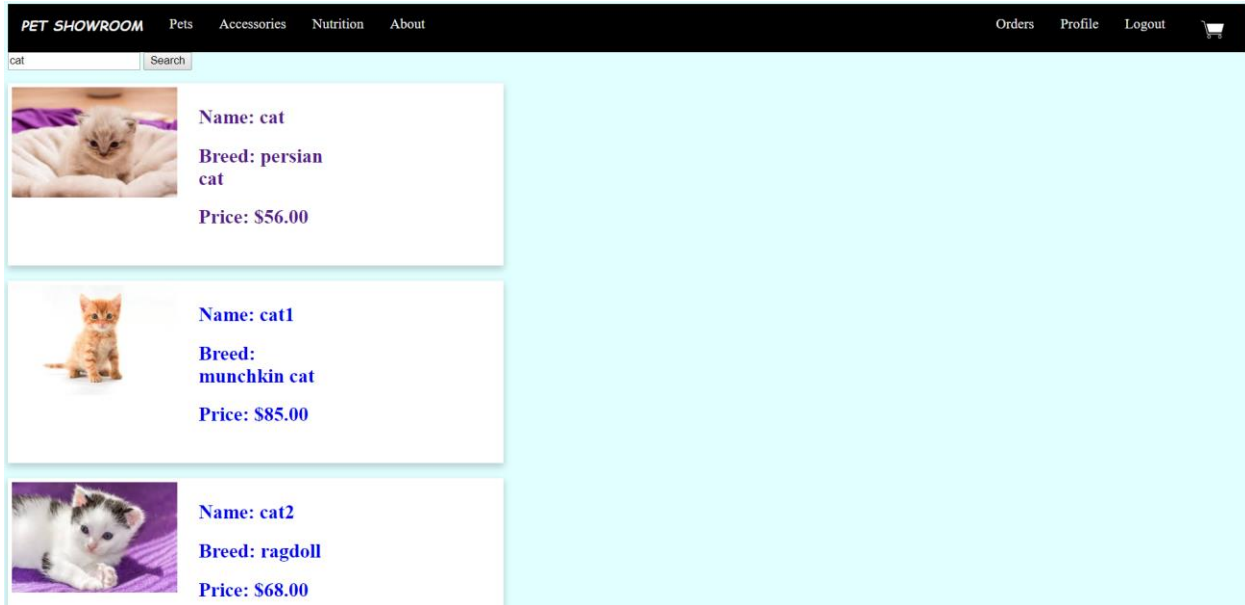
- Name:** puppy34
- Breed:** chihuahua
- Price:** \$56.00

The third listing shows:

- Name:** puppy35
- Breed:** chihuahua
- Price:** \$56.00

## Searching in Pets

The search field does not require the whole name of the breed, it will find the products as long as the letters are within that particular breed. For example, there is a breed 'german shepherd', the user does not have to type in the whole thing. They can just type some letters from it in order as shown below.



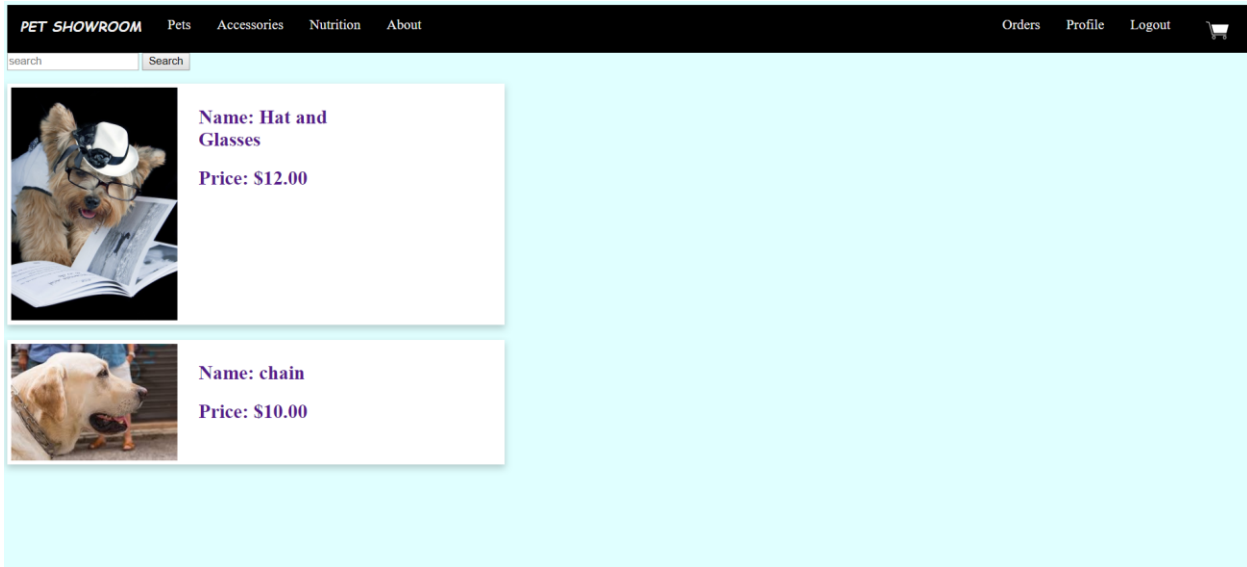
The screenshot shows a website header for 'PET SHOWROOM' with navigation links: Pets, Accessories, Nutrition, About, Orders, Profile, Logout, and a shopping cart icon. A search bar contains the text 'cat' and a 'Search' button. Below the search bar, three product cards are displayed, each with a small image of a cat and its details:

Name	Breed	Price
cat	persian cat	\$56.00
cat1	munchkin cat	\$85.00
cat2	ragdoll	\$68.00



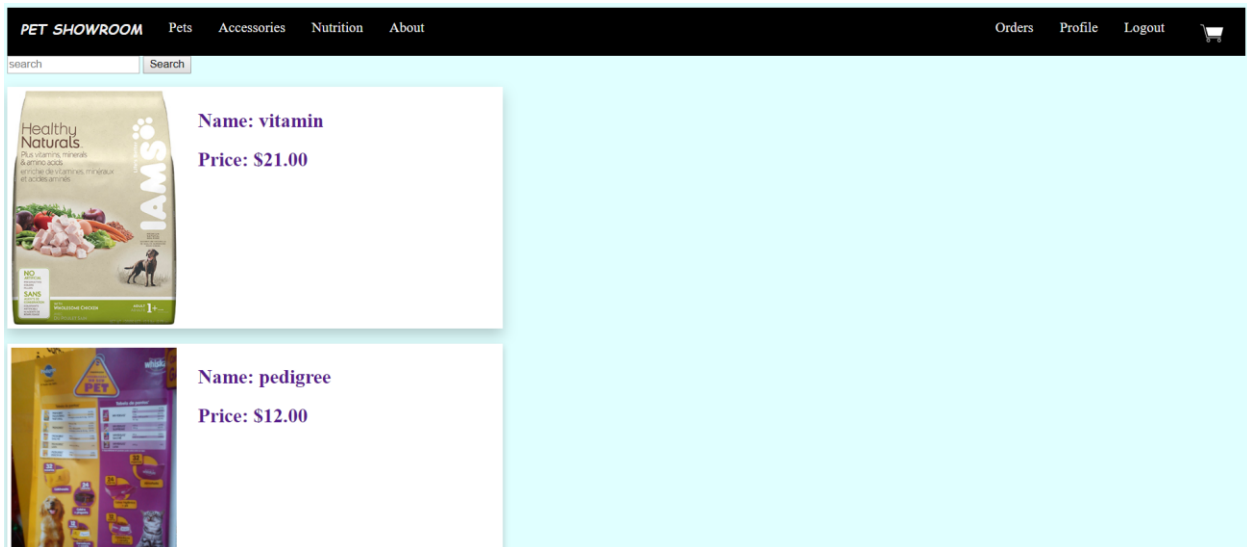
## Accessories

This page allows the user to choose from the pet accessories. It also has search field which can search the products with their name.



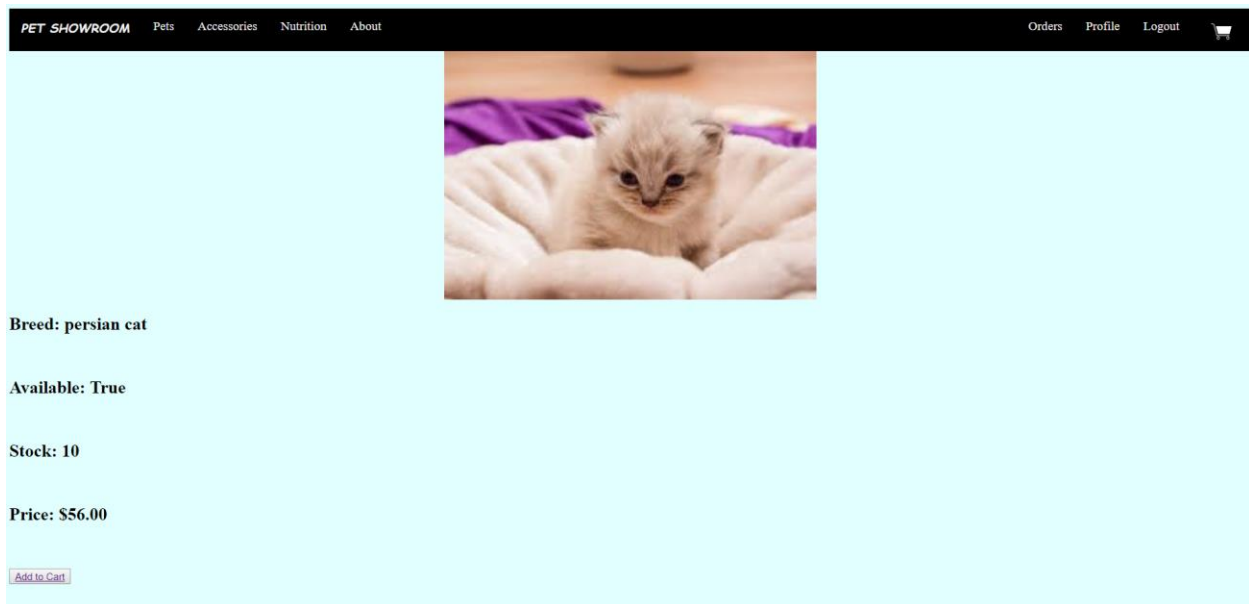
## Nutrition

This page allows the user to choose from the pet nutrition products on the website.



## Detail Page

Each of the three above modules has a detail page, that is, when a product is selected by the user, it shows a page where it has details of that particular as shown in the below image. It also has 'add to cart' button which is used to add the particular product to the cart.



## Cart

On clicking the cart symbol on the right corner of the top navigation bar, the user can get to this page. Every product added to the cart is displayed here. Each product has a button 'remove from cart' which removes that particular product from the cart. There is a confirm order button which is used to confirm the order.

**PET SHOWROOM** Pets Accessories Nutrition About Orders Profile Logout

Name: cat  
Quantity: 1  
Price: 56.00  
[Remove from cart](#)

Name: Hat and Glasses  
Quantity: 1  
Price: 12.00  
[Remove from cart](#)

[Confirm Order](#)

## Orders

The confirmed order items are shown in the orders page.

Image	Name	Quantity	Price
	cat	1	56.00
	Hat and Glasses	1	12.00

**Your order has been confirmed!**

## Chapter 7 - Security and Exception Handling

**Security** (<https://docs.djangoproject.com/en/2.2/topics/security/>)

Django Framework provides some built-in security features which are as follows,

(<https://docs.djangoproject.com/en/2.2/topics/http/middleware/>)

### 1. Cross site scripting (XSS) protection

XSS attacks allow a user to inject client side scripts into the browser of other users.

This attack is usually achieved by storing the malicious scripts in the database where it will be retrieved and displayed to other users, or by getting users to click on a link which will cause the attacker's JavaScript to be executed by the user's browser. They can originate from any untrusted source of data, such as cookies or Web services, whenever the data is not sufficiently sanitized before including in a page. Using django templates protects against XSS attacks.

### 2. Cross site request forgery (CSRF) protection

CSRF attacks allow a user to execute actions using the credentials of another user without the knowledge of that user. Django has protection from the CSRF attacks. It works by checking for secret in each POST request.

### 3. SQL injection protection

SQL injection attacks allow a user to execute arbitrary SQL code on a database.

This can result in leakage of data. Django querysets protect from the SQL injection attacks since their queries are constructed using query parameterization.

### 4. Clickjacking protection

Clickjacking attack is a attack where a malicious site wraps another site in a frame.

This results in a user being tricked into performing unintended actions on the target side.

Django contains clickjacking protection in the X-Frame-Options middleware which supports a browser to prevent a site being rendered inside a frame.

## 5. Host header validation

Even secure web server configurations are susceptible to fake Host headers. Django validates Host headers against the `ALLOWED_HOSTS` setting in the `django.http.HttpRequest.get_host()` method.

### Exception Handling (<https://docs.djangoproject.com/en/2.2/ref/exceptions/>)

Django raises its own exceptions and standard python exceptions. The django core exception classes are defined in `django.core.exceptions`. Some of them are,

#### 1. `ObjectDoesNotExist`

This occurs when a user is trying to get a product whose id is not present in the database. This mostly occurs if the user manipulates the url and changes the id to something that does not exist.

#### 2. `EmptyResultSet`

This exception occurs when a query does not return any results.

#### 3. `FieldDoesNotExist`

This exception is raised when a model is requested with a field that does not exist in the respective table of the database.

#### 4. `MultipleObjectsReturned`

This is raised when a query encountered multiple objects when it is supposed to be receiving one object when returning an object.

## **5. PermissionDenied**

This exception is raised when a user performs an action that they do not have permission to.

## **6. NoReverseMatch**

This exception is raised when a matching URL requested by the user is not present in the project.

## Chapter 8 - Testing and Evaluation

(<https://docs.djangoproject.com/en/2.2/topics/testing/>)

### Unit Testing

Unit testing is a level of software testing used to test individual units/components of a software.

Below, is the test to give a forbidden error message when an unauthorized user tries to access cart page by manipulating the url.

```
class TestRegisteredUser():
    def setUp(self):
        self.user = User.objects.create(username='testuser', password='password', email='email@gmail.com')
        self.anonymous_user = AnonymousUser()
        self.url = self.reverse('pets:cart')

        with self.login(self.user):
            self.get(self.url)
            self.response_200()

        with self.login(self.anonymous_user):
            self.get(self.url)
            self.response_403()
```

### Correctness Testing

By using the results from unit testing and manual testing by users, i have gained the following results.

Test Case	Expected Results	Result
Loading of the home page	Should render fine	Works
Hyperlinks for all the pages in the top navigation bar	All work fine	Works

Login Page	checks if authenticated user or not	Works
Products in Pets, Nutrition and Accessories	Should be in the form of cards and should highlighted when hovered over	Works
On clicking a product	Should redirect to the details page of that particular product	Works
Add to cart button	should add that particular product to the cart and display product added	Works
Search field on Pets, Accessories and Nutrition pages	should show the related results	Works
Confirm order button	should redirect to orders page showing the products that are ordered and display the message that the order is confirmed	Works

Table 1 Test cases for correctness testing



## Performance Testing and scalability

When there were only 10 products on the database, the website returned the results in 0.2 milliseconds. When there were 100 products, it returned the results in 0.3 milliseconds.

<b>Products count on the database</b>	<b>Time taken to return the results</b>
10	0.2 ms(millisecons)
100	0.3 ms
200	0.3 ms
300	0.2 ms
500	0.3 ms
800	0.3 ms
1000	0.3 ms
2000	0.3 ms
4000	0.3 ms
5000	0.3 ms
10000	0.3 ms

Table 2 Scalability of the products

### **What did i learn?**

I learnt how to develop a website from scratch using Django framework. I was able to know a lot about django built-in features, security and exceptional handling. I learnt a lot of backend programming using python. I also learnt a lot about how request and response work from frontend to backend.

### **Lines of code:**

<b>Language</b>	<b>Lines of Code</b>
Python	1,133
HTML	492
CSS	178

## **Chapter 9 - Accessibility of the Application**

### **Color blind people accessibility**

The web application is easier to access for the colorblind people as the color contrasts have been used keeping them in mind.

### **Elder people accessibility**

There are pictures of the pets which makes it easy for the elderly people to choose a product easily. Whenever the user hovers over a product, it is highlighted which makes it easier for them to understand which product they are choosing.

## **Chapter 10 - Conclusion**

The PET SHOWROOM Web application is an online pet store which has pets, their accessories and nutrition products. This website is for the pet lovers who can find everything at one place.

The web application is user friendly, basic user can navigate through out the website and find the desired products. This website is really useful for pet lovers and to sell pets.

With this web application development, one can learn python programming with django framework and many built-in features about the framework. It is really helpful in getting hands on experience for a developer.

## **Chapter 11 - Future Work**

At present, user can order products but there is no payment method. In future, i would like to add a payment method using PayPal.

To improve user experience, I would like to add keyboard shortcuts throughout the website which makes it easier for the user to navigate.

To work more on the frontend(CSS) which makes the website more appealing.

## Chapter 12 - Bibliography

[1] Documentation about Django framework

<https://docs.djangoproject.com/en/2.1/>

[2] Request and response objects

<https://docs.djangoproject.com/en/2.2/ref/request-response/>

[3] Making queries and creating objects

<https://docs.djangoproject.com/en/2.2/topics/db/queries/>

[4] Django built-in security features

<https://docs.djangoproject.com/en/2.2/topics/security/>

[5] Django built-in exceptions

<https://docs.djangoproject.com/en/2.2/ref/exceptions/>

[6] Middleware for request/response in Django(Security)

<https://docs.djangoproject.com/en/2.2/topics/http/middleware/>

[7] Testing in Django

<https://docs.djangoproject.com/en/2.2/topics/testing/>