



**CSE 299.5**

**Junior Design Project**

**Speech Petrol Pump Management System**

**Group # 7**

**Fahmida Sultana**

**ID # 1812829642**

**Sharmin Akter**

**ID # 1812349042**

Faculty Advisor:

Zunayeed Bin Zahir

Lecturer

ECE Department

Summer, 2021

# Acknowledgement

At the very beginning, we express our gratefulness to almighty Allah for the blessings which makes us possible to complete our desire project.

We are grateful and wish our profound indebtedness to Zunayeed Bin Zahir sir, Senior Lecturer, Department of ECE, North South University, Dhaka. Being interest on doing website and seeing user welfare to carry out this project. His guidance, constant supervision, enthusiastic encouragement, sagacious advice and an effective surveillance throughout the entire period of the project have made it possible to complete this project.

At last we must express our sincere heartfelt gratitude to all the staff members of the Computer Engineering Department who helped us directly or indirectly during this course of work.

# Abstract

In this report we present a petrol management system which is be a Website. The website name is “GAS N GO”. This website was done by using Django framework. Nowadays, People are very busy with their work and many issues. In our daily life, people face many problems which is sometimes unbearable to us. There are one of the most common problem in our country which is traffic jam. For this, people can’t move anywhere on time. Especially, those people who has vehicles, they wait so long time to take fuel from the petrol pump because of gathering. After reaching fuel station they also need to stand up in a queue to pour gases, fuel, petrol, and so on. Sometimes it seems gas is over also so for that they have to move to look out another station. For the betterment of people we thought to build up such a project where they can book first with their current location and get to aware all the information. Following this goal, we implemented a website where user can create his own account in our website and book a fuel station according to his choice of place and can also see the available amount of fuel left in the fuel station. Users can also choose preferred fuel and get the total payment cost. Moreover user data can be stored in our website so user can see his fuel refill history. Similarly admin can store required information.

# Table of Content

Chapter 1: Project Overview .....	1
1.1 Introduction.....	2
1.2 Our proposed project.....	9
1.2.1 Description of the idea: .....	9
1.2.2 Problem Statement .....	10
1.3 Motivation.....	10
1.4 Summary .....	11
Chapter 2: Related work .....	12
2.1 Introduction.....	13
2.2 Existing work related to Petrol Pump Management .....	13
2.2.1 Esheba.cnsbd.com .....	<b>Error! Bookmark not defined.</b>
2.2.2 Trivago.com.....	<b>Error! Bookmark not defined.</b>
2.2.3 MobiKwik .....	14
2.3 Research .....	14
2.3.1 Collecting address of petrol pump station.....	14
2.3.2 Information on fuel .....	15
2.3.3 Django API .....	15
2.3.4 Image Collection .....	15
2.4 Summary .....	15
Chapter 3: Theory .....	16
3.1 Introduction.....	17
3.2 How Django Look Like .....	17-16
3.3 Collected Data.....	17
3.3.1 Place Name .....	17
3.3.2 Fuel Price .....	17
3.3.3 Fuel Amount .....	18
3.4 Summary .....	25
Chapter 4: Structure of the system.....	26
4.1 Introduction.....	27
4.2 Requirements .....	27
4.2.1 Design Requirements .....	27
4.2.2 Hardware Requirements.....	27
4.2.3 Software Requirements .....	281
4.2.4 Stakeholders .....	281

4.3 User Story .....	22-23
4.4 Summary .....	24
<b>Chapter 5: Login and registration .....</b>	<b>Error! Bookmark not defined.25</b>
5.1 Introduction.....	32
5.2 Screenshot.....	32
5.3 Description of Process .....	32
5.4 Address.....	33
5.5 Summary .....	<b>Error! Bookmark not defined.</b>
<b>Chapter 6: Homepage.....</b>	<b>28</b>
6.1.Introduction.....	29
6.2 Eminities .....	29
6.2.1 Example of Choosing place .....	380
6.2.2 Backend.....	31
6.3 Avaiable amount .....	32
6.3.1 Example of choosing amount of fuel.....	32
6.3.2 Backend and Frontend.....	33
6.4 Address.....	
6.5 Summary .....	404
<b>Chapter 7: Booking Page .....</b>	<b>Error! Bookmark not defined.5</b>
7.1 Introduction.....	426
7.2 Design bakcend-frontend .....	427
7.4 Address.....	38
7.5 Summary .....	38
<b>Chapter 8: Booking Information .....</b>	<b>39</b>
8.1 Introduction.....	40
8.2 View of booking history .....	40
8.3 Backend and Frontend .....	40
8.4 Summary .....	40
<b>Chapter 9: Django Administration .....</b>	<b>41</b>
9.1 Introduction.....	42
9.2 Superuser .....	42
9.2.1 How to create superuser in Django.....	43
9.3 Data at django Administration.....	44
9.3.1 Customer.....	44
9.3.2 Superuser.....	45
9.3.3 Contact.....	46
9.3.4 Eminities.....	47
9.4 Gass.....	48
9.4 Summary.....	48
<b>Chapter 10: API.....</b>	<b>48</b>
10.1 Introduction.....	49

10.2 Json Response .....	49
10.2.1 Django rest framework with Json for API .....	49
10.2.2 Json Set save equal to fault .....	50
10.3 Summary .....	50
Chapter 11: Results and Discussion .....	51
11.1 Introduction.....	52
11.2 Result Analyses.....	52-58
11.3 Summary .....	59
Chapter 12: Conclusion.....	60
Bibliography.....	62
Appendix.....	64

# *Chapter 1*

## **Project Overview**

## 1.1 Introduction

“Gas N Go” petrol pump management system is a website which was built by using django framework. Petrol pump management system is developed to manage daily records of petrol pump. Using this system admin can check the employee attendance, quantity of petrol and diesel sold in any day, week or month. Using sales module admin can check how much diesel or petrol sold. Admin can also check total oil purchase by the petrol pump. Using this system admin can find full details of users like his\her name, password and also email address. Admin can also check what amount of petrol, diesel, oil has sold. User can generate reports easily form this petrol pump management system.

By this website users able to reserve a time whenever they will be in need to take gas or oil and also this project users is able to find out petrol pump according to their current location. It is shown in the project, it was build the frontend part of petrol management system from scratch. At first, there is home page and from there we can visit other pages. Here it contains the location and amount, how much available. In this website any kind of user is able to login and hope they didn't face any problem while doing registration. Users have to pay after reload gas or fuel by cash. There is system where they can pay this via online payment system (Bkash, Rocket or Master/Visa card). Through this system got proficient application that hold overall location of petrol pump. This is a very user-friendly website. After completing their payment, they will get a confirmation message. Our main goal is to satisfy the needs of customers who wants to save their time and get a better service.



## **1.2 Our proposed project**

The main idea of our project is to build a petrol management website which will help user to save their valuable time and take fuel easily by booking and knowing all the information which they can relate.

### **1.2.1 Description of the idea:**

Dhaka city or many other areas people face hassle while filling up their vehicles such as they stuck in the traffic jam while searching petrol pump station and hardly can reach in the station and also sometimes saw that there is no fuel or gas left in the station. So, we get our idea from this problem. If we make a website which can show users nearest fuel station then they doesn't have to search here and there by facing traffic jam and they can also know which petrol pump station is available for filling up their vehicle's fuel or gas. They also aware the quantity of products they need is available or not to their nearest location. In this software users able to fulfill their needed requirements very smoothly.

### **Capability of the petrol pump management:**

The website has the capability to --

- To save user's valuable time.
- To find out nearest petrol pump station.
- To see whether fuel or gas left in the station.
- Seeing that user is able to book smoothly what they need.
- Take the oil/gas for vehicles on the suitable time.
- To pay their payment easily in cash or in online based such as using different cards or by bKash.

## 1.2.2 Problem Statement

The level of difficulty of this project was very high, as petrol pump management is a completely new project. While doing the project we face problem for less information are available in the online. There are some risk of losing customer's data as we don't have any data holder and strong management system. So, sometimes it could be happen users lost their data they provide or given before. Data security is very low so for the reason it can be hacked very easily. There can be also bugs or errors from 3rd party API's.

As Django follows a high level model structure and it architecture is too high. At the beginning of the project we faced problem to cope up with Django documentation, how it works and so on. After a lots of research and watching videos for Django, after that we came up a state where we was able to do and then we started our implementation. We faced problem while doing API set up. Admin panel emenities set up,image upload problem, console log problem, automatic logout etc. But ended up this problem by researching, from other student who is aware of doing website and others stuffs.

## 1.3 Motivation

Tec The level of difficulty of this project was very high, as petrol pump management is a completely new project. While doing the project we face problem for less information are available in the online. There are some risk of losing customer's data as we don't have any data holder and strong management system. So, sometimes it could be happen users lost their data they provide or given before. Data security is very low so for the reason it can be hacked very easily. There can be also bugs or errors from 3rd party API's.

As Django follows a high level model structure and it architecture is too high. At the beginning of the project we faced problem to cope up with Django documentation, how it works and so on. After a lots of research and watching videos for Django, after that we came up a state where we was able to do and then we started our implementation. We faced problem while doing API set up. Admin

panel amenities set up, console log problem, automatic logout etc. But ended up this problem by researching, from other student who is aware of doing website and others stuffs.

## **1.4 Summary**

In this chapter, we have briefly described the basics of our petrol management system, how we built the project and the main idea on which our project is built. We have described the capabilities our project, what motivated us to design and build this system, and our accomplishments in here. The following chapters describe the theory and details of the components used, designs, and the overall structure of the system.

*Chapter 2*  
**Related work/Literature Review**

## **2.1 Introduction**

The existing work related to petrol management system is that we had discovered and found useful that are described in this chapter. As we searched for similar systems, we found some related existing systems (not exact) where they can provide services to their users. We went through some similar type of projects which gave us some insights. We also searched for systems and existing papers to localize the sound source, we got some. The broad description for it is given below.

### **2.2 Existing work related to Speech Recognition on robot**

#### **2.2.1 Esheba.cnsbd.com**

For travelling in a railway we need to book or cut down ticket first then able to travel. For ticket booking user have to go to the railway ticket counter from which railway station user want to travel. The passenger needs to go counter before the journey. If user's Journey has more important, they have to book a ticket earlier. Otherwise, there had a chance to miss the Ticket or expected seat. Sometimes it does happen that they didn't find their desire-able seat or seat are not available. So, for their betterment Esheba.cnsbd.com website was built so that they can book a ticket in online or by going to the station. This website will provide them with all the information about the train ticket booking system online. They help to make trip better and easier. The online train ticket booking system is much easier than others.

### **2.2.2 Trivago.com**

Trivago is a hotel price comparison site with an extensive hotel search. The prices shown come from numerous hotels and booking websites. This means that while users decide on Trivago which hotel best suits their needs, the booking process itself is completed through the booking sites (which are linked to our website). By clicking on the “view deal” button, user will be forwarded onto a booking site; from there they will be able to get review and book the room best offer shown on Trivago. Without any hassle by Trivago people can booked hotel room and can get all update from the website

### **2.2.3 MobiKwik**

Online gas booking can be instantly and easily made using the MobiKwik app or webpage. MobiKwik allows its users to make LPG booking online and get the LPG gas booking in not more than 5 steps. MobiKwik also offers huge offers for LPG gas bookings and enable the users to make their gas bill payment online without any hassle. Making online gas booking through MobiKwik is an easy and instant process and offers great savings. Thus, MobiKwik is the best platform for online gas payment

## **2.3 Research**

As our project is very new we had to research a lot on specific things like data collection, how to save data, how to search location, how can we know about amount of fuel left on fuel station, how we can show total amount of price according to given amount of fuel. Moreover we research on better way to set our API.

### **2.3.1 Collecting address of petrol pump station**

In this paper address are collected petrol pump is a necessary which is collected from Google Map. At first some places are selected then addresses of some fuel station that are available on place are collected as our data.

## **2.3.2 Information on Fuel**

In this paper, available fuel on fuel station and price are needed. Known the name of the available fuel by asking people and price from Google. Government has fixed the prices of the fuels.

## **2.3.3 Django API**

API may stand for Application Programming Interface. Many documentation are seen for setting API. Many frameworks allow you to easily build API. Json response can be used to build Django Rest Framework. Django-Json-api uses Django's ORM interfaces as inspiration to serialize, request and deserialize entities from databases of other microservices using (and benefiting from) the JSON:API specification. Not used Django serializer instead made own serializer with Json.

## **2.3.4 Image Collection & Materialize CSS**

In this project images are provided to each fuel station. So image of each fuel station that are mentioned in project was collected from Google Map. But as Json response is used for building API and uploading image is not supported in Json response so image is collected from Pixels. From Pixels image link is used.

In this project we also research on materialize CSS

## **2.4 Summary**

The existing work related to speech recognition on mobile robot and sound source detection that we found useful have been briefly described in this section. The next chapter elaborates more on the theoretical part of our project.

## *Chapter 3*

# **Theory**



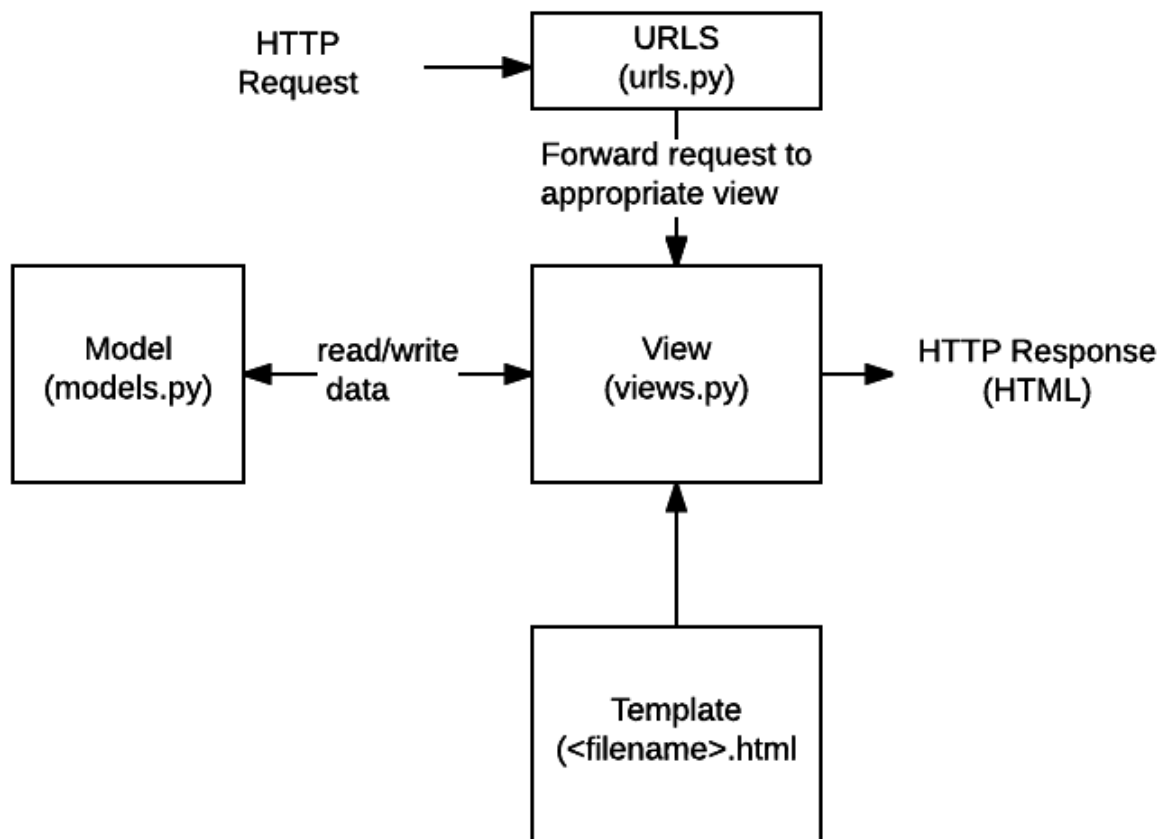
## 3.1 Introduction

The details of the theory of our system are discussed in this chapter. All of the information are given below:

The system had been built fully by Django Framework and python languages for backend and HTML, CSS, JS and MySQL for front-end portion. We used Django because it is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel.

## 3.2 What does Django code look like?

In a traditional data-driven website, a web application waits for HTTP requests from the web browser (or other client). When a request is received the application works out what is needed based on the URL and possibly information in `POST` data or `GET` data. Depending on what is required it may then read or write information from a database or perform other tasks required to satisfy the request. The application will then return a response to the web browser, often dynamically creating an HTML page for the browser to display by inserting the retrieved data into placeholders in an HTML template. Django web applications typically group the code that handles each of these steps into separate files:



- **URLs:** While it is possible to process requests from every single URL via a single function, it is much more maintainable to write a separate view function to handle each resource. A URL mapper is used to redirect HTTP requests to the appropriate view based on the request URL. The URL mapper can also match particular patterns of strings or digits that appear in a URL and pass these to a view function as data.
- **View:** A view is a request handler function, which receives HTTP requests and returns HTTP responses. Views access the data needed to satisfy requests via *models*, and delegate the formatting of the response to *templates*.
- **Models:** Models are Python objects that define the structure of an application's data, and provide mechanisms to manage (add, modify, delete) and query records in the database.
- **Templates:** A template is a text file defining the structure or layout of a file (such as an HTML page), with placeholders used to represent actual content. A *view* can dynamically create an

HTML page using an HTML template, populating it with data from a *model*. A template can be used to define the structure of any type of file; it doesn't have to be HTML!

## Sending the request to the right view (urls.py)

### Code:

```
urlpatterns = [

    path('admin/', admin.site.urls),

    path('book/<int:id>/', views.book_detail, name='book_detail'),

    path('catalog/', include('catalog.urls')),

    re_path(r'^([0-9]+)/$', views.best),

]
```

## Handling the request (views.py)

**Views are the heart of the web application, receiving HTTP requests from web clients and returning HTTP responses. In between, they marshal the other resources of the framework to access databases, render templates etc.**

### Code:

```
# filename: views.py (Django view functions)

from django.http import HttpResponse

def index(request):

    # Get an HttpRequest - the request parameter

    # perform operations using information from the request.

    # Return HttpResponse

    return HttpResponse('Hello from Django!')
```

## Defining data models (models.py)

The code snippet below shows a very simple Django model for a `Team` object.

The `Team` class is derived from the django class `models.Model`. It defines the team name and team level as character fields and specifies a maximum number of characters to be stored for each record. The `team_level` can be one of several values, so we define it as a choice field and provide a mapping between choices to be displayed and data to be stored, along with a default value.

### Code:

```
# filename: models.py

from django.db import models

class Team(models.Model):

    team_name = models.CharField(max_length=40)

    TEAM_LEVELS = (

        ('U09', 'Under 09s'),

        ('U10', 'Under 10s'),

        ('U11', 'Under 11s'),

        ... #list other team levels

    )

    team_level = models.CharField(max_length=3, choices=TEAM_LEVELS, default='U11')
```

## Querying data (views.py)

The Django model provides a simple query API for searching the associated database. This can match against a number of fields at a time using different criteria (e.g. exact, case-insensitive, greater than, etc.), and can support complex statements (for example, you can specify a search on U11 teams that have a team name that starts with "Fr" or ends with "al")

The code snippet shows a view function (resource handler) for displaying all of our U09 teams. The `list_teams = Team.objects.filter(team_level__exact="U09")` line shows how we can use the model query API to filter for all records where the `team_level` field has exactly the text 'U09' (note how this criteria is passed to the `filter()` function as an argument, with the field name and match type separated by a double underscore: `team_level__exact`).

Code:

```
## filename: views.py

from django.shortcuts import render

from .models import Team

def index(request):

    list_teams = Team.objects.filter(team_level__exact="U09")

    context = {'youngest_teams': list_teams}

    return render(request, '/best/index.html', context)
```

## Rendering data (HTML templates)

Template systems allow you to specify the structure of an output document, using placeholders for data that will be filled in when a page is generated. Templates are often used to create HTML, but can also create other types of document. Django supports both its native templating system and another popular Python library called Jinja2 out of the box (it can also be made to support other systems if needed).

## Code:

```
## filename: best/templates/best/index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="utf-8">

    <title>Home page</title>

</head>

<body>

    {% if youngest_teams %}

        <ul>

            {% for team in youngest_teams %}

                <li>{{ team.team_name }}</li>

            {% endfor %}

        </ul>

    {% else %}

        <p>No teams are available.</p>

    {% endif %}

</body>

</html>
```

## 3.3 Collected Data

### 3.3.1 Places Name

Gazipur, Savar, Uttara, Banani, Gulshan, Mirpur, Mohammadpur, Motijheel. This places are the emenities in the project.

### 3.3.2.Fuel Price

Fuel name	Price
Petrol	57
Octane	95
Diesel	65
Natural Gas	65
LPG	41

### 3.3.3 Fuel Amount

Place	Name	Available amount of fuel
Banani	Royal filling Station	9
Motijheel	Hazipara CNG Filling Station	2
Gulshan	Ideal filling Station	5
Gulshan	Crescent Petrol Pump(Meghna)	14
Mirpur	Ariya CNG Filling Station	12
Mirpur	Semano LPG Autogas Filling Station	0
Mirpur	Purbachal Filling Station	6
Motijheel	Dhaka's 1st Petrol Station	7
Mohammadpur	Sonar Bangla Service Station	7
Motijheel	M/S Pubali Filling Station	5
Uttara	Islam Brother's LPG Filling Station	3
Uttara	Khandakar CNG & Filling Station	7
Savar	Afzal brothers filling station	6
Savar	Nabinogor CNG Refuelling & Filling Station	1
Gazipur	Gazipur city filling station	5



## **3.4 Summary**

In this chapter, the theoretical part of our project has been described. We have tried to explain all the requirements needed in our project.

## *Chapter 4*

# **Structure of the system**

## **4.1 Introduction**

Structure of website is an important to discuss for website.

## **4.2 Requirements**

Before describing the workflow and algorithms of the system, how the system works has been explained first. For the sake of clarity, we have also explained the fifteen verbal commands the robot recognizes and the corresponding functions it performs.

### **4.2.1 Design Requirements:**

- i. HTML
- ii. CSS (SCSS)
- iii. JAVA SCRIPTS(JS)

### **4.2.2 Hardware Requirements:**

- i. i5 Processor Based Computer or higher
- ii. Memory: 8 GB RAM
- iii. Hard Drive: 150 GB
- iv. Internet Connection

### **4.2.3 Software Requirements:**

- i. Windows 10
- ii. Visual Studio 2020
- iii. SQLite Server

### **4.2.4 Stakeholders:**

. External

- Customer
- Service Provider

- Admin
- Servicer
- Cashier

## ii. Internal

- Project Manager
- Software Engineer
- Project Developer

## 4.3 User Stories:

A user story is open, simple language information of one or more than one opinion of a software System. Basically, user story represents the class of users, what they need, and why they need that. It supports generating a clear explanation of a condition. So, here we have created 3 characters of User stories. Which are: Admin, User and Service Provider.

### i. For Customer:

\*\* As a new customer

I want to register my account

So that I can get benefit from here

\*\* As a Registered customer

I want to take services

So that I can save my valuable time.

\*\* As a Registered customer

I want to change my password

So that I can keep my account secured

\*\* As a Registered customer

I want to get a system to enter my personal information and payment details

So that I can take services and make payment without any hassle

\*\* As a customer

I want to choose a product

So that I can place an order

\*\* As a customer

I want to add preferred products to my cart

So that I can initiate the transaction later

\*\* As a customer

I want to get a payment page

So that I can purchase this successfully

\*\* As a customer

I want a page to enter my personal information and payment details

So that I can easily transact

\*\* As a customer

I want to get a membership card

So that I can get more offer from here

\*\* As a customer

I want to give some feedback

So that people can see their review

\*\* As a customer

I want to get a receipt

So that I can see my transaction confirmation

## **ii. For Service Provider:**

\*\* As a User

I want a page

So that I can see all order

**\*\* As a User**

I want a transaction page

So that I can see the accounting costPage | 8

**iii. For Admin:**

**\*\* As an Admin**

I want to see the admin dashboard

So that I can update some records if necessary

**\*\* As an Admin**

I want to collect every feedback

So that I can work with all the feedback

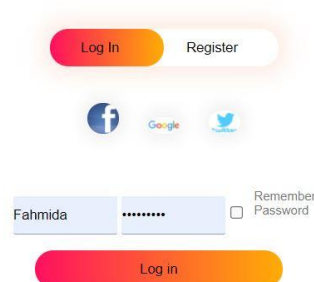
## *Chapter 5*

# **Login & Registration**

## 5.1 Introduction

This chapter discusses the login & registration related detail. In this page user at 1<sup>st</sup> register if he has no account than login with the same name and password.

## 5.2 Screenshot



## 5.3 Description of process

For making this page, at first made an environment in Django environment .Then made an app name accounts and then in the app made some folder such as static, templates. In the static folder made two more folder which is images and styles. In images part I took some photos which is related to my login and registration part and in styles folder I made a file which name is login.css which works for customize and design for login and registration part. In Templates folder I made a file which is login.html. In this portion I designed for login and registration both. Then linked up with login.css. I worked on views.py to view my design and in urls.py I created some method to go on home page so that view the model which I want to design. Thus I completed my accounts app completed this page. The registered user detail will be saved in our Django administration.



### **5.46.3 Adresses**

Link to homepage:

[Login And Registration Form](#)

## **5.5 Summary**

Login and registration is a primary and important part of website. Almost every website have login and registration. Login and registration plays a key role in your digital security strategy.

## *Chapter 6*

# HomePage

## 6.1 Introduction

In this project homepage plays the main functions of our project. Homepage contain the option of finding fuel station according to place and fuel amount.

## 6.2 Select Eminities


GAS N GO

[Logout](#)

[Booking Information](#)

Select Eminities  
Choose your place


Amount



Gazipur city filling station

Available amount  
- 5


Address:  
Madda



Nabinogor CNG Refuelling & Filling Station

Available amount  
- 1

Address:  
Nabinagar-Chandra Rd,Dendabor




Afzal brothers filling station

Available amount  
- 6

Address:  
N5, Savar Union

CONFIRMATION



Khandakar CNG & Filling Station

Available amount  
- 7


Address:  
Abdullahpur Over Bridge, Dhaka 1230

CONFIRMATION

CONFIRMATION

Address:  
Dar-Us-Salam Road, Technical More, Mirpur Technical Bus Station, Mirpur, Dhaka-1216


CONFIRMATION



Sonar Bangla Service Station

Available amount  
- 7

Address:  
50, 1 Asad Ave, Dhaka 1207




Ideal Filling Station

Available amount  
- 5

Address:  
191 Bir Uttam Mir Shawkat Sarak, Gulshan, Dhaka 1208


CONFIRMATION



Dhaka's 1st Petrol Station

Available amount  
- 7

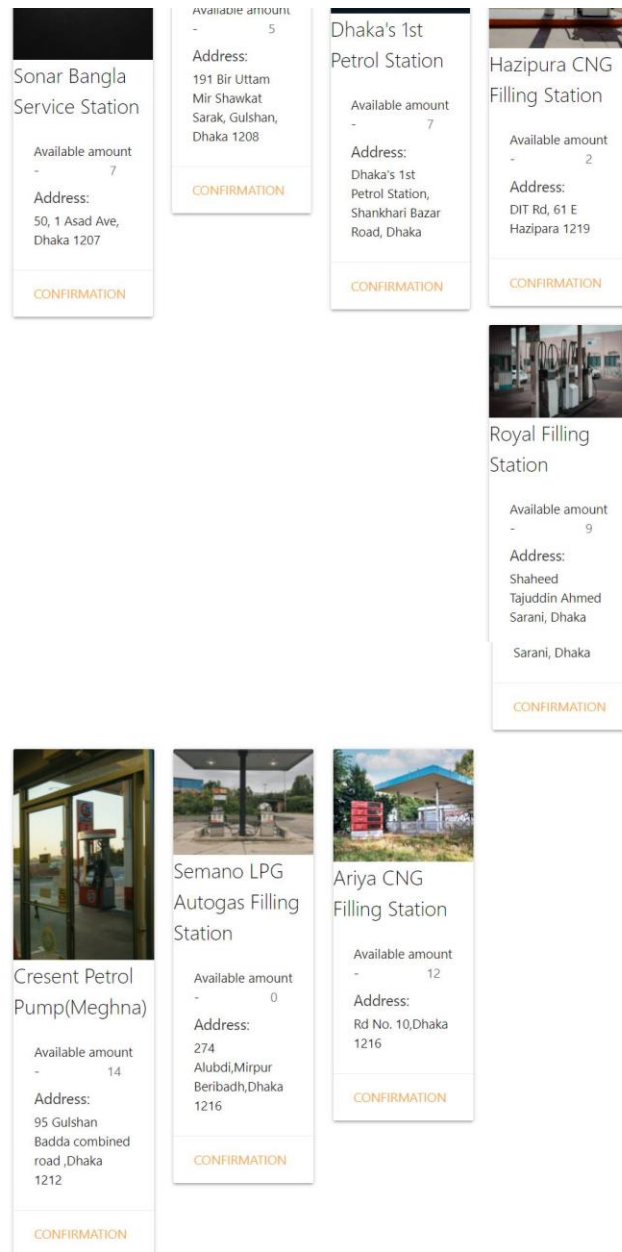
Address:  
Dhaka's 1st Petrol Station, Shankhari Bazar Road, Dhaka



Hazipura CNG Filling Station

Available amount  
- 2

Address:  
DIT Rd, 61 E Hazipara 1219



Here from the dropdown menu named “choose your place” desired place can be able to choose. There are total 15 fuel stations are given. It is possible to add more fuel station or any information.

## 6.1.2 Example of choosing place

Suppose, if user choose place “Uttara” then only that fuel station will be shown which are at Uttara.

GAS N GO

[Logout](#)

[Booking Information](#)

Select Emenities

✓ Choose your place

☐ option

☐ Savar

☒ Uttara

☐ Banani

☐ Gulshan


☐ Mirpur

☐ Mohammadpur

☐ Motijheel

CONFIRMATION

Amount



Islam Brother's  
LPG Filling  
Station

Available amount  
- 3

Address:  
kamarpara,  
Vatuulia ,Uttara

CONFIRMATION

GAS N GO


[Logout](#)

[Booking Information](#)

Select Emenities

Choose your place, Uttara

Amount




Khandakar CNG  
& Filling Station

Available amount  
- 7

Address:  
Abdullahpur  
Over Bridge,  
Dhaka 1230

CONFIRMATION



Islam Brother's  
LPG Filling  
Station

Available amount  
- 3

Address:  
kamarpara,  
Vatuulia ,Uttara

CONFIRMATION

Here khandakar CNG & Filling Station and Islam brother’s LPG filling Station are at Uttara which has 7litre and 3 litre fuel respectively. Now if user want 3 litre fuel he can move slider to value 3 than the fuel station which has 3litre fuel will be shown.

## 6.1.2 Backend

At first fields are taken at model.py within class “Gas”.Return data to pump\_name..Here

Each place can have many fuel station.

```

class Emenities(models.Model):
    name = models.CharField(max_length=100)

    def __str__(self):
        return self.name

class Gas(models.Model):
    pump_name = models.CharField(max_length=100)
    pump_image = models.CharField(max_length=500)
    pump_address = models.TextField()
    available_amount = models.IntegerField()
    emenities = models.ManyToManyField(Emenities)

    def __str__(self):
        return self.pump_name

```

## 6.2 Available Amount

The slider named “Amount” where user can give the amount they want for fuel.


### 6.2.1 Example of choosing amount of fuel

If user want 3 litre fuel at Uttara he can move slider to value 3 than the fuel station which has 3litre fuel will be shown.

Select Emmenties

Choose your place, Uttara

Amount



Islam Brother's  
LPG Filling  
Station

Available amount  
- 3

Address:  
kamarpara,  
Vatuulia ,Uttara

CONFIRMATION


- 5

Address:  
N105,Gazipur

CONFIRMATION

Rd,Dendabor

CONFIRMATION




Islam Brother's  
LPG Filling  
Station

Available amount  
- 3

Address:  
kamarpara,  
Vatuulia ,Uttara

CONFIRMATION




M/S Pubali  
Filling Station

Available amount  
- 5

Address:  
79/A Motijheel  
C/A, Dhaka 1000

CONFIRMATION



Purbachal Filling  
Station

Available amount

## 6.2.2 Backend & frontend

Here we have taken 0 to 15 as the amount of fuel. Range field is used to build this slider. The onchange event occurs when the value of an element has been changed. Frontend is designed at home.html file.

39

```

</div>
<div class="col m4">
<label>Amount</label>
<p class="range-field">
<input type="range" onchange="getGas()" id="available_amount" min="0" max="15" value="15">
</p>
</div>

```

## 6.3 Addresses

Link to homepage:

[Gas-N-GO](#)

```

urlpatterns = [
    path('', include('home.urls')),
    path('accounts/', include('accounts.urls')),
    path('booking/', include('booking.urls')),
    path('admin/', admin.site.urls),
]

```

Homepage path given here.

## 6.4 Summary

There is the logo at the above and also logout booking information which user can logout and see his booking history respectively. Before filtering by place and amount of fuel the page contain all the fuel station that are given in Django administration. After filtering according to amount and place only that filtered fuel station will be shown. All the data of places are given in Eminities and fuel station name address amount are given in Gas in Django administration. API from backend which is Json data. When we pass query than it will filter out data in that way

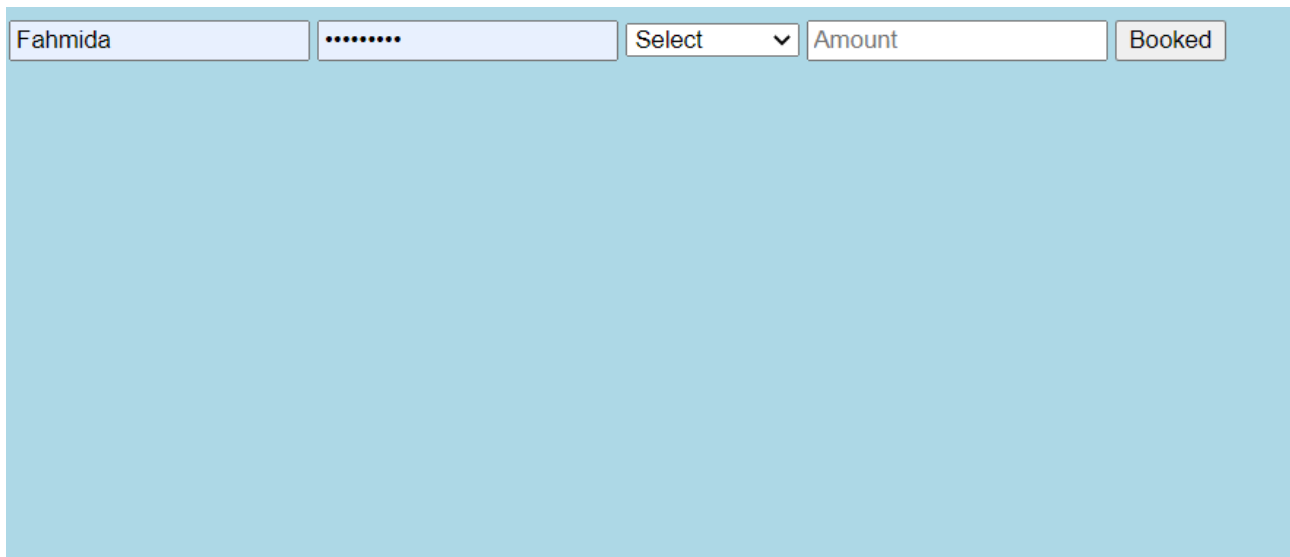


*Chapter 7*  
**Booking Page**

## 7.1 Introduction

In this page user can get fuel types and select required one. User can make booking for his vehicles fuel refill.

## 7.2 Design, backend & frontend

A screenshot of a web form for fuel booking. The form has a light blue background. At the top, there are four input fields: a text box containing 'Fahmida', a password box with seven dots, a dropdown menu with 'Select' and a downward arrow, and a text box containing 'Amount'. To the right of these fields is a button labeled 'Booked'. Below the input fields is a large, empty light blue rectangular area.

If user has an account than name and password arrives automatically at booking page as data is saved in Django administration.

Given a dropdown menu for selecting fuel.

```
<select name="fuel_type" id="fuel_type">
  <option value="select">Select</option>
  <option value="petrol">Petrol</option>
  <option value="octane">Octane</option>
  <option value="diesel">Diesel</option>
  <option value="natural_gas">Natural Gas</option>
  <option value="lpg">LPG</option>
</select>
```

User can give his amount of needed fuel than book. Frontend is designed inside booking app at cat.html. The fields are taken at model.py in booking app.

```

# Create your models here.
class Contact(models.Model):
    name = models.CharField(max_length=124)
    password = models.CharField(max_length=20)
    fuel_type = models.CharField(max_length=20)
    fuel_amount = models.CharField(max_length=20)
    price = models.CharField(max_length=20)

    def __str__(self):
        return self.name

```

## 7.3 Calculated Price

User can get total amount money for payment. In the method book we run a elif condition for the calculation of price. Here user given amount of fuel is multiplied with per litre price of fuel.

```

def book(request):
    if request.method == 'POST':
        name = request.POST['name']
        fuel_type = request.POST['fuel_type']
        fuel_amount = request.POST['fuel_amount']
        password = request.POST['password']
        if fuel_type == 'petrol':
            price = int(int(fuel_amount) * 52)
        elif fuel_type == 'octane':
            price = int(int(fuel_amount) * 95)
        elif fuel_type == 'diesel':
            price = int(int(fuel_amount) * 65)
        elif fuel_type == 'natural_gas':
            price = int(int(fuel_amount) * 64)
        else:
            price = int(int(fuel_amount) * 42)

        book = Contact.objects.create(name=name, password=password, fuel_type=fuel_type, fuel_amount=fuel_amount, price=price)
        book.save()
        return redirect('preview')
    else:
        return render(request, 'cat.html')

```

For example if user take 3 litre LPG than his total cost will be  $(3 \times 41) = 126$ . [41 is the government decide price for per litre LPG.]

Fig. 7.2. Bluetooth module hc-05

## 7.4 Address

### Link to booking page:

[Document](#)

## 7.5 Summary

In this chapter, we have described the booking page. All the data will be saved in booking of Django administration.

*Chapter 8*  
**Booking Information page**

## 8.1 Introduction

By this booking information page any user can see all the information or history how many fuel, gas , octane and so on they booked.

Address

[Document](#)

## 8.2 View of booking history

In this page we viewed users Id number and what type of products they are going to book and it's amount also converted price with amount.

14
Fuel Type: octane
Amount: 3 (ltr)
Price: 285(Taka)
15
Fuel Type: octane
Amount: 3 (ltr)
Price: 285(Taka)
16
Fuel Type: lpg
Amount: 3 (ltr)
Price: 126(Taka)
17
Fuel Type: select
Amount: 6 (ltr)

*Fig. 8.2*

## 8.3 Back-end and Front-end.

For backend and frontend work we created a file in booking app templates which name is book\_preview.html. In this file we work for viewing booking information so that user can see the history.

## 8.4 Summary

So in short we can say that we showed the view of booking information. Without it users couldn't able to know gas fuel amount is available or not and price of the amount.



## *Chapter 9*

# **Django Administration**



## 9.1 Introduction

One of the most powerful parts of Django is the automatic admin interface. It reads metadata from your models to provide a quick, model-centric interface where trusted users can manage content on your site. The admin's recommended use is limited to an organization's internal management tool. It's not intended for building your entire front end around. The admin has many hooks for customization, but beware of trying to use those hooks exclusively.

Address:

[Site administration](#) | [Django site admin](#)

## 9.2 Super User

Creating a Super User in Django Django's prominent feature is the admin interface, which makes it stand out from the competition.

### 9.2.1 How to create super user in Django?

For creating superuser, first reach the same directory as that of **manage.py** and run the following command:

```
python manage.py createsuperuser
```

Then enter the Username of your choice and press enter.

```
Username: sristi
```

Then enter the Email address and press enter.(It can be left blank)

```
Email address: fahamida.sultana3@gmail.com
```

Next, enter the Password in-front of the Password field and press enter Enter a strong password so as to keep it secure.

```
Password: *****
```

Then again enter the same Password for confirmation.

```
Password (again): *****
```

Superuser created successfully if above fields are entered correctly.

```
PS C:\Users\R AND S\Desktop\Travel> cd travel
PS C:\Users\R AND S\Desktop\Travel\travel> python manage.py createsuperuser
Username (leave blank to use 'rands'): srishti
Email address: example@gmail.com
Password:
Password (again):
Superuser created successfully.
PS C:\Users\R AND S\Desktop\Travel\travel> █
```

*Image shown after above steps done*

Now we can login into our Django Admin page by running the command **python manage.py runserver** . Then, open a Web browser and go to “/admin/” on your local domain – e.g., **http://127.0.0.1:8000/admin/** and then enter the same Username and Password.



Django administration

Username:

srishti

Password:

.....

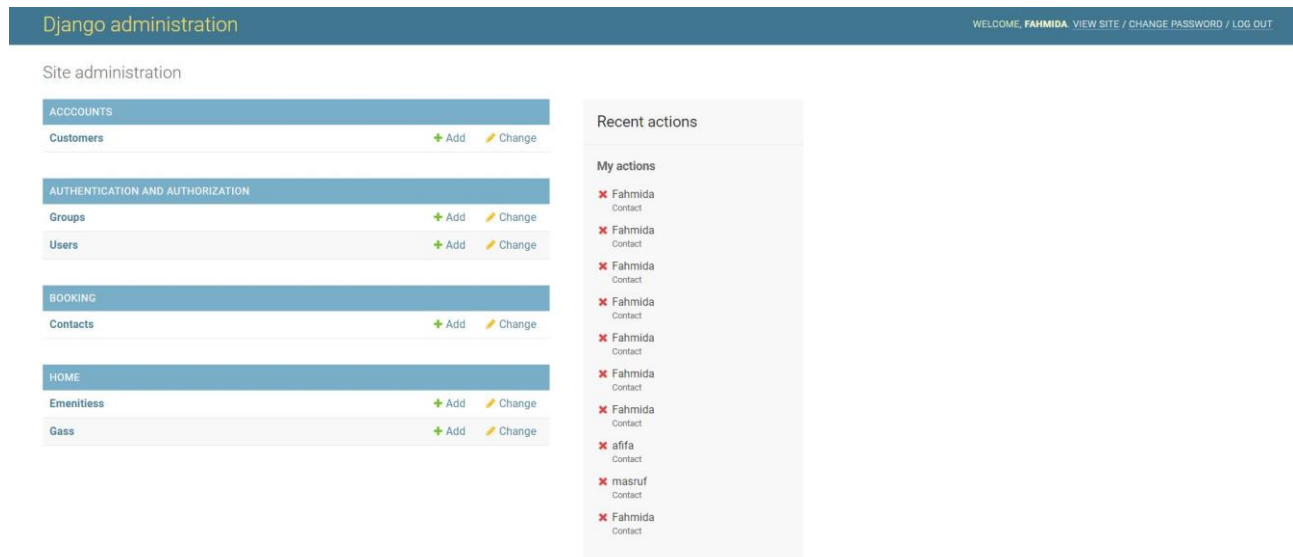
Log in

*Django admin page*

## 9.3 Data at Django administration

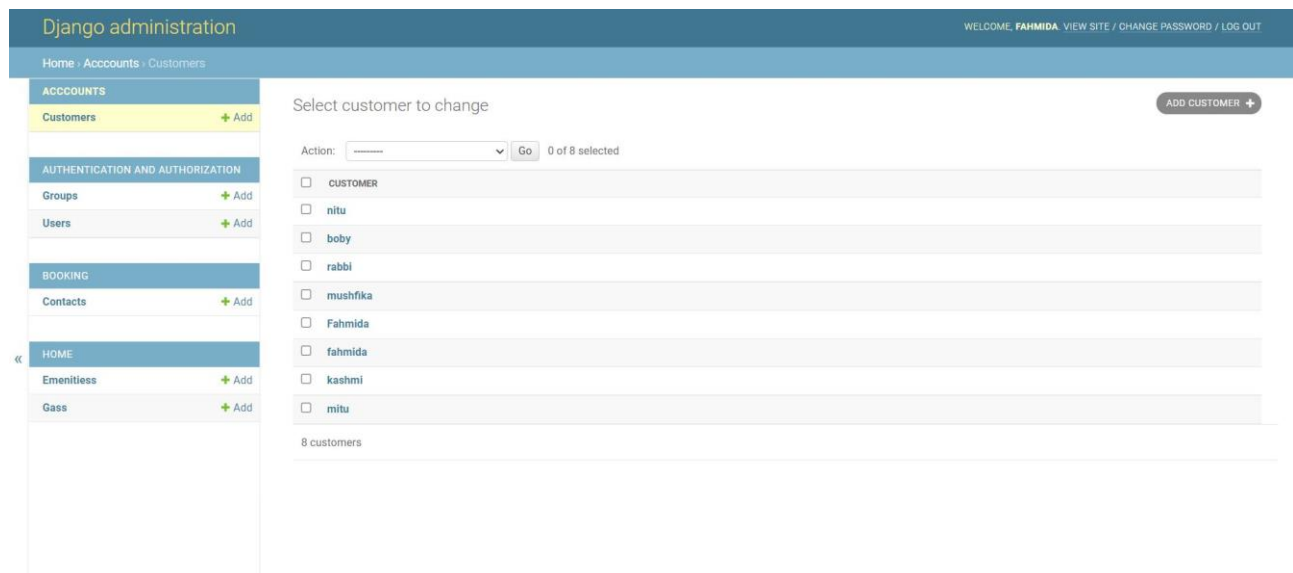
In this project all data are saved in Django administration. Here Customers for Login & Registration

Contacts for Confirmation history, Emenities for place and Gass for fuel station detail are given.



### 9.3.1 Customer

Here the user details who have registered will be stored. Data can be deleted if admin want to.



Django administration
WELCOME, FAHMIDA. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Accounts > Customers > mitu

ACCOUNTS

Customers + Add

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

BOOKING

Contacts + Add

HOME

Ementiless + Add

Gass + Add

Change customer

mitu

Username:

mitu

Email:

mitu6093@gmail.com

Password:

5678

Delete

Save and add another

Save and continue editing

SAVE

## 9.3.2 Super User

Django administration
WELCOME, FAHMIDA. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Authentication and Authorization > Users

ACCOUNTS

Customers + Add

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

BOOKING

Contacts + Add

HOME

Ementiless + Add

Gass + Add

Select user to change

ADD USER +

Q

Search

Action:

Go

0 of 1 selected

	USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
<input type="checkbox"/>	Fahmida	fahmida.sultana3@northsouth.edu			

1 user

FILTER

By staff status

All

Yes

No

By superuser status

All

Yes

No

By active

All

Yes

No

## 9.3.3 Contacts

Here users confirmation history that is name,password and total price will be stored after booking. This data is provided to show Booking History.

Django administration WELCOME, FAHMIDA. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Booking > Contacts

ACCOUNTS

Customers [+ Add](#)

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Users [+ Add](#)

BOOKING

Contacts [+ Add](#)

HOME

Emenitiess [+ Add](#)

Gass [+ Add](#)

Select contact to change ADD CONTACT +

Action:  Go 0 of 8 selected

☐ CONTACT

☐ Fahmida

☐ Fahmida

☐ Fahmida

☐ Fahmida

☐ Fahmida

☐ Fahmida

☐ Fahmida

8 contacts

Django administration WELCOME, FAHMIDA. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Booking > Contacts > Fahmida

ACCOUNTS

Customers [+ Add](#)

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Users [+ Add](#)

BOOKING

Contacts [+ Add](#)

HOME

Emenitiess [+ Add](#)

Gass [+ Add](#)

Change contact HISTORY

**Fahmida**

Name:

Password:

Fuel type:

Fuel amount:

Price:

## 9.3.4 Emminities

Here some selected place name has been taken.

Django administration WELCOME, FAHMIDA. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Home > Emenitiess

ACCOUNTS

Customers [+ Add](#)

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Users [+ Add](#)

BOOKING

Contacts [+ Add](#)

HOME

Emenitiess [+ Add](#)

Gass [+ Add](#)

Select emenities to change ADD EMENTIES +

Action:  Go 0 of 8 selected

☐ EMENTIES

☐ Motijheel

☐ Mohammadpur

☐ Mirpur

☐ Gulshan

☐ Banani

☐ Uttara

☐ Savar

☐ Gazipur

8 emenitiess

# 9.3.5 Gass

Here 15 fuel station and there details .This details always edited by admin.

Django administration

WELCOME, FAHMIDA VIEW SITE / CHANGE PASSWORD / LOG OUT

Home Home Gass

ACCOUNTS

Customers Add

AUTHENTICATION AND AUTHORIZATION

Groups Add

Users Add

BOOKING

Contacts Add

HOME

Emenitiess Add

Gass Add

Select gas to change

ADD GAS +

Action: Go 0 of 15 selected

☐ GAS

☐ Ariya CNG Filling Station

☐ Semano LPG Autogas Filling Station

☐ Cresent Petrol Pump(Meghna)

☐ Royal Filling Station

☐ Hazipura CNG Filling Station

☐ Dhaka's 1st Petrol Station

☐ Ideal Filling Station

☐ Sonar Bangla Service Station

☐ Purbachal Filling Station

☐ M/S Pubali Filling Station

☐ Islam Brother's LPG Filling Station

☐ Khandakar CNG & Filling Station

☐ Afzal brothers filling station

☐ Nabinogor CNG Refuelling & Filling Station

☐ Gazipur city filling station

Change gas

HISTORY

Ariya CNG Filling Station

Pump name: Ariya CNG Filling Station

Pump image: https://images.pexels.com/photos/5777101/

Pump address: Rd No. 10,Dhaka 1216

Available amount: 12

Emenities: 

Gazipur Savar Uttara Banani Guleshan Mirpur Mohammadpur Motijheel

Delete

Save and add another

Save and continue editing

SAVE

# 9.5 Summary

This is the most important portion of project. Only admin can edit this and user will be provided information stored here.

## *Chapter 10*

# **API**

## 10.1 Introduction

API stands for Application Programming Interface, which is a software application that we interact programmatically, instead of using a graphical interface.

## 10.2 Json Response

The `json()` method of the `Response` interface takes a `Response` stream and reads it to completion. It returns a promise which resolves with the result of parsing the body text as JSON

### 10.2.1 Django Rest Framework with Json for API

At first import `JsonResponse`

```
from django.http import JsonResponse
```

```
def api_gas(request):
    gas_objs = Gas.objects.all()
    ... available_amount = request.GET.get('available_amount')
    ... if available_amount:
        gas_objs = gas_objs.filter(available_amount__lte=available_amount)

    emenities = request.GET.get('emenities')
    if emenities:
        emenities = emenities.split(',')
        em = []
        for e in emenities:
            try:
                em.append(int(e))
            except Exception as e:
                pass
        gas_objs = gas_objs.filter(emenities__in=em).distinct()

    payload = []
    for gas_obj in gas_objs:
        result = {}
        result['pump_name'] = gas_obj.pump_name
        result['pump_image'] = gas_obj.pump_image
        result['pump_address'] = gas_obj.pump_address
        result['available_amount'] = gas_obj.available_amount
        payload.append(result)
    return JsonResponse(payload, safe=False)
```



Payload variable is a loop for gas object where all gas are appended

### 10.2.2 Json set as **safe=False**

The JSON Response in Django set **safe=True** by default, and the safe parameter as a data influencer makes JSON accept the Python Data-Type {Dictionaries} and nothing less. So, at this point any data sent contrary to {Dictionaries} would actually fire up errors.

So, setting the safe parameter to **False** actually influences JSON to receive any Python Data Type.

Personally and professionally, it is advisable you set it the safe parameter to **False** because it makes JSON accept both {Dictionaries} and others.

```
return JsonResponse(<'your python data-type'>, safe = False
```

## 10.3 Summary

The API formation is discussed here. And many more information on API is given.

*Chapter 11*  
**Results and Discussion**

## 11.1 Introduction

The results and Analyses of our project are discussed in this chapter. Our main target was to implement such an application where we can find out their (user) nearest gas and fuel station and then can book and reserve slot for filling gas and fuel. Users can also aware of whether fuel is available or not. After filling gas user can pay in person or via online system. Thus they can save their valuable time. After implementing our project we can say that we were able to implement all of the functions that we thought, just we couldn't able to work on online payment system because we thought that customer can pay in person so that they have not face any hassle. We worked on online payment system also but not completed due to Bkash and Nogod API needed. Then we thought also to make our project responsive but we unfortunately we couldn't do that. Our future aspect will be to complete the online payment system and make our project responsive.

## 11.2 Result Analysis

To see Gas N Go website user has two option:

1.Create an environment on Django.Than activate the environment.

```
venv/scripts/activate
```

Run the website by:

```
Python manage.py runserver
```

2.Using the URL

After running the website at first we see the home page





GAS N GO

[Logout](#)

[Booking Information](#)

Select Emenities  
Choose your place

Amount

 Gazipur city filling station Available amount - 5 Address: .....	 Nabinogor CNG Refuelling & Filling Station Available amount - 1 Address: Nabinagar- Chandra Rd,Dendabor	 Afzal brothers filling station Available amount - 6 Address: N5, Savar Union CONFIRMATION	 Khandakar CNG & Filling Station Available amount - 7 Address: Abdullahpur Over Bridge, Dhaka 1230 CONFIRMATION
----------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Here suppose user want to register then he will logout.Then Login option arrived.

GAS N GO

[Login](#)

After clicking in login user get registration option where he can register and store data in Django administration.

Log In

Register





Fahmida

fahmidamitu18@

\*\*\*\*\*

☐
I agree to the terms & conditions

Register

Django administration
WELCOME, FAHMIDA. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Accounts > Customers > mitu

ACCOUNTS

Customers + Add

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

BOOKING

Contacts + Add

HOME

Ementieess + Add

Gass + Add

Change customer

mitu HISTORY

Username:

mitu

Email:

mitu6093@gmail.com

Password:

5678

Delete

Save and add another

Save and continue editing

SAVE

As the data of register stored. User who have registered can login.



In the homepage user has the major function that is, search fuel according to place and available fuel .User can also see the booking information and logout option.If user choose Uttara

GAS N GO


[Logout](#)

[Booking Information](#)

Select Emenities

Choose your place, Uttara

Amount




Khandakar CNG & Filling Station

Available amount  
- 7

Address:  
Abdullahpur  
Over Bridge,  
Dhaka 1230

CONFIRMATION



Islam Brother's LPG Filling Station

Available amount  
- 3

Address:  
kamarpara,  
Vatuulia ,Uttara

CONFIRMATION

If user give 3litre n

GAS N GO

[Logout](#)

[Booking Information](#)

Select Emenities

✓ Choose your place

☐ option

☐ Savar

☒ Uttara

☐ Banani


☐ Gulshan

☐ Mirpur

☐ Mohammadpur

☐ Motijheel

Amount



Islam Brother's LPG Filling Station


Available amount  
- 3

Address:  
kamarpara,  
Vatuulia ,Uttara

CONFIRMATION

Select Emmenties  
 Choose your place, Uttara

Amount



Islam Brother's  
LPG Filling  
Station

Available amount  
- 3

Address:  
kamarpara,  
Vatuulia ,Uttara

CONFIRMATION

After Choosing user can make confirmation.

Fahmida

.....

Select ▼

Amount

Booked

Give name,password and fuel type and amount then make booking and datat is stored in admin



Fahmida

.....

LPG

3

Booked

Django administration

WELCOME, FAHMIDA. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home » Booking » Contacts » Fahmida

ACCOUNTS

Customers [+ Add](#)

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Users [+ Add](#)

BOOKING

Contacts [+ Add](#)

HOME

Ementless [+ Add](#)

Gass [+ Add](#)

Change contact

Fahmida [HISTORY](#)

Name:

Fahmida

Password:

fahmida05

Fuel type:

lpg

Fuel amount:

3

Price:

126

Delete

Save and add another

Save and continue editing

SAVE

Here total price is saved after calculation. Multiplied  $3 \times 41 = 126$

41 is the per litre price of lpg

After that user get this confirmation history.

<b>14</b> Fuel Type: octane Amount: 3 (ltr) Price: 285(Taka)
<b>15</b> Fuel Type: octane Amount: 3 (ltr) Price: 285(Taka)
<b>16</b> Fuel Type: lpg Amount: 3 (ltr) Price: 126(Taka)
<b>17</b> Fuel Type: select Amount: 6 (ltr)

By getting this history can understand that his booking is confirmed.

## 11.4 Summary

In this chapter, we have described and discussed the results of our project, what we wanted to at first and what have been achieved and give a short demonstration

## *Chapter 12*

# **Conclusion**

In this project, we have focused to make an application which is user friendly viewing user betterment. This website is very much helpful as it saves our time. From our childhood to till we always have to remember on proverb that is "Time and tide waits for none", so make the best use of time and save it. In this website we can easily find out the nearest petrol pump to their location and can login or registration very easily. User can easily book a slot for pouring or filling fuel, gas and oil to their respective vehicles. They need not to stand in queue or sit for a long time in the jam and also to find out filling station.

While doing the project we learn a lot. We had to research on it so much, thus not only this project related things we were able to know other thing also. We taught how Django works, taught Python language, html, CSS and JS. Although we had to face some problems, could able to work to reach our goal, couldn't work on some of the parts.

We teach one more thing that is if you research more you will know more, you will face some problem but your goal motivates you that you have to that. What we could able to do it is okay but what we couldn't, in future we will focus more. We also planning to transaction system on our website and our plan is to make promotions through social media to aware people. Thus, the development of this field can open up unlimited possibilities and open up new applications that can be very useful and have a big impact on people's lives.

## **Github Link:**

<https://github.com/sharminafifa25/Gas-N-Go/tree/homepage>

## *Bibliography*

- [1] <https://www.w3schools.com/>.
- [2] <https://materializecss.com/>
- [3] <https://www.pexels.com/search/petrol%20pump%20station/>
- [4] <https://code.jquery.com/>
- [5] <https://www.django-rest-framework.org/>
- [6] <https://djangostars.com/blog/rest-apis-django-development/>
- [7] <https://pypi.org/project/django-json-api/>
- [8] <https://howtojsonapi.com/django.html>
- [9] <https://nordicapis.com/the-benefits-of-using-json-api/>
- [10] <https://developer.mozilla.org/en-US/docs/Web/API/Response/json>
- [11] <https://stackoverflow.com/questions/63229231/how-to-return-custom-json-response-with-django-rest-framework>
- [12] <https://chryzcodez.hashnode.dev/django-json-response-safefalse>
- [13] <https://docs.djangoproject.com/en/3.2/ref/contrib/admin/>
- [14] <https://www.geeksforgeeks.org/how-to-create-superuser-in-django/>
- [15] <https://getbootstrap.com/>
- [16] <https://www.w3schools.com/Js/>
- [17] <https://www.w3schools.com/css/default.asp>
- [18] <https://code.visualstudio.com/docs/python/tutorial-django>
- [19] <https://www.javatpoint.com/django-virtual-environment-setup>
- [20] <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction?fbclid=IwAR3HWcj1ydGrzWOi4jqtme3S8L-pnzfJp5F5WppJiDVLRbtmlovDIOUgQY4>

## *Appendix*

## Codes Related to Core

core/asgi.py

```
"""
ASGI config for core project.

It exposes the ASGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/3.2/howto/deployment/asgi/
"""

import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'core.settings')

application = get_asgi_application()
```

core/settings.py

```
"""
Django settings for core project.

Generated by 'django-admin startproject' using Django 3.2.6.

For more information on this file, see
https://docs.djangoproject.com/en/3.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.2/ref/settings/
"""

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-(pjv5odapw51e!fu@fz-1#-cwrk-y%ux(g$*ahzz4g5cs$j7^'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
```



```

ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'home',
    'accounts',
    'booking',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'core.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'core.wsgi.application'

# Database
# https://docs.djangoproject.com/en/3.2/ref/settings/#databases

```

```

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}

# Password validation
# https://docs.djangoproject.com/en/3.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

# Internationalization
# https://docs.djangoproject.com/en/3.2/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.2/howto/static-files/

STATIC_URL = '/static/'

# Default primary key field type
# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field

```

```
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

## core/urls.py

```
"""core URL Configuration
```

The `urlpatterns` list routes URLs to views. For more information please see:  
<https://docs.djangoproject.com/en/3.2/topics/http/urls/>

Examples:

Function views

1. Add an import: `from my_app import views`
2. Add a URL to `urlpatterns`: `path('', views.home, name='home')`

Class-based views

1. Add an import: `from other_app.views import Home`
2. Add a URL to `urlpatterns`: `path('', Home.as_view(), name='home')`

Including another URLconf

1. Import the `include()` function: `from django.urls import include, path`
2. Add a URL to `urlpatterns`: `path('blog/', include('blog.urls'))`

```
"""
```

```
from django.contrib import admin
```

```
from django.urls import path, include
```

```
urlpatterns = [  
    path('', include('home.urls')),  
    path('accounts/', include('accounts.urls')),  
    path('booking/', include('booking.urls')),  
    path('admin/', admin.site.urls),  
]
```

## core/wsgi.py

```
"""
```

WSGI config for core project.

It exposes the WSGI callable as a module-level variable named `application`.

For more information on this file, see

<https://docs.djangoproject.com/en/3.2/howto/deployment/wsgi/>

```
"""
```

```
import os
```

```
from django.core.wsgi import get_wsgi_application
```

```
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'core.settings')
```

```
application = get_wsgi_application()
```

## Codes Related to accounts

accounts/static/styles/login.css

```

*{
    margin: 0;
    padding: 0;
    font-family: sans-serif;
}
.afifa{
    height: 100%;
    width: 100%;
    background-image: linear-
gradient(rgba(0,0,0,0.4),rgba(0,0,0,0.4))url("/static/acccounts/images/banner.jpg")
;
    background-position: center;
    background-size: cover;
    position: absolute;
}
.form-box{
    width: 380px;
    height: 480px;
    position: relative;
    margin: 6% auto;
    background: #fff;
    padding: 5px;
    overflow: hidden;
}
.button-box{
    width: 220px;
    margin: 35 auto;position: relative;
    box-shadow: 0 0 20px 9px #ff61241f;
    border-radius: 30px;
}
.toggole-btn{
    padding: 10px 30px;
    cursor: pointer;
    background: transparent;
    border: 0;
    outline: none;
    position: relative;
}
#btn{
    top: 0;
    left: 0;
    position: absolute;
    width: 110px;
    height: 100%;
}
```

```

    background: linear-gradient(to right, #ff105f, #ffad06);
    border-radius: 30px;
    transition: .5s;
}
.social-icons{
    margin: 30px auto;
    text-align: center;
}
.social-icons img{
    width: 30px;
    margin: 0 12px;
    box-shadow: 0 0 20px 0 #7f7f7f3d;
    cursor: pointer;
    border-radius: 50%;
}
.input-group{
    top: 180px;
    position: absolute;
    width: 280px;
    transition: .05s;
}
.input-field{
    width: 100px;
    padding: 10px 0;
    margin: 5px 0;
    border-left: 0;
    border-top: 0;
    border-right: 0;
    border-bottom: 1px solid #999;
    outline: none;
    background: transparent;
}
.submit-btn{
    width: 85%;
    padding: 10px 30px;
    cursor: pointer;
    display: block;
    margin: auto;
    background: linear-gradient(to right, #ff105f, #ffad06);
    border: 0;
    outline: none;
    border-radius: 30px;
}
.check-box{
    margin: 30px 10px 30px 0;
}

```

```

span{
    color: #777;
    font-size: 12px;
    bottom: 68px;
    position: absolute;
}
#login{
    left: 50px;
}
#register{
    left: 450px;
}

```

## accounts/templates/login.html

```

{% load static %}
<html>
    <head>
        <title>Login And Registration Form</title>
        <link rel="stylesheet" href="{% static 'accounts/styles/login.css' %}">
    </head>
    <body>
        <div class="afifa">
            <div class="form-box">
                <div class="button-box">
                    <div id="btn">
                        <button type="button" class="toggole-
btn" onclick="login()">Log In</button>
                        <button type="button" class="toggole-
btn" onclick="register()">Register</button>
                    </div>
                    <div class="social-icons">
                        
                        
                        
                    </div>

                    <form action="login" method="POST" id="login" class="input-group">
                        {% csrf_token %}
                        <input type="text" name="username" class="input-
field" placeholder="User Name" required>
                        <input type="password" name="password" class="input-
field" placeholder="Enter Password" required>
                        <input type="checkbox" class="check-
box"><span>Remember Password</span>

```

```

        <button type="submit" class="submit-btn">Log in</button>

    </form>
    <form action="register" method="POST" id="register" class="input-
group">
        {% csrf_token %}
        <input type="text" name="username" class="input-
field" placeholder="User Name" required>
        <input type="email" name="email" class="input-
field" placeholder="Email Id" required>
        <input type="password" name="password" class="input-
field" placeholder="Enter Password" required>
        <input type="checkbox" class="check-
box"><span>I agree to the terms & conditions</span>
        <button type="submit" class="submit-btn">Register</button>
    </form>

</div>

</div>
<script>
    var x = document.getElementById("login");
    var y = document.getElementById("register");
    var z = document.getElementById("btn");
    function register(){
        x.style.left = "400px";
        y.style.left = "50px";
        z.style.left = "110px";

    }
    function login(){
        x.style.left = "50px";
        y.style.left = "450px";
        z.style.left = "0px";

    }

</script>
</body>
</html>

```

accounts/admin.py

```

from django.contrib import admin
from .models import *

```

```
admin.site.register(Customer)
```

accounts/apps.py

```
from django.apps import AppConfig

class AccountsConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'accounts'
```

accounts/models.py

```
from django.db import models

# Create your models here.
class Customer (models.Model):
    username = models.CharField(max_length=50, unique=True)
    email = models.EmailField(max_length=50)
    password = models.CharField(max_length=20)

    def __str__(self):
        return self.username
```

accounts/tests.py

```
from django.test import TestCase

# Create your tests here.
```

accounts/urls.py

```
from django.contrib import admin
from django.urls import path
from . import views
from .views import *

urlpatterns = [
    path('', acc, name="acc_view"),
    path('login', login, name="login"),
    path('register', register, name="register"),
    path('logout', logout, name='logout'),
]
```



accounts/views.py

```
from django.shortcuts import redirect, render
from django.contrib.auth.models import auth
from .models import Customer

# Create your views here.
def acc(request):
    return render(request, 'login.html')

def login(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']

        user = auth.authenticate(username=username, password=password)
        if user is not None:
            auth.login(request, user)
            return redirect('home')
        else:
            return redirect('acc_view')
    else:
        return render(request, 'login.html')

def register(request):
    if request.method == 'POST':
        username = request.POST['username']
        email = request.POST['email']
        password = request.POST['password']
        user = Customer.objects.create(username=username, password=password, email=
email)
        user.save()
        return redirect('login')
    else:
        return render(request, 'login.html')

def logout(request):
    auth.logout(request)
    return redirect('home')
```

## Codes related to home

home/static/styles/payment.css

```

* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: 'Poppins', sans-serif;
}
body{
  width: 100%;
  height: 100vh;
  display: flex;
  justify-content: center;
  align-items: center;
  background-color: azure;
}
.container{
  width: 750px;
  height: 500px;
  border: 1px solid;
  background-color: white;
  display: flex;
  flex-direction: column;
  padding: 40px;
  justify-content: space-around;
}
.container h1{
  text-align: center;
}
.first-row{
  display: flex
}
.owner{
  width: 100%;
  margin-right: 40px;
}
.input-filed{
  border: 1px solid #999;
}
.input-filed input{
width: 100%;
border: none;
outline: none;
padding: 10px;
}
.selection{
  display: flex;
  justify-content: space-between;
  align-items: center;
}

```

```

}
.selection select{
  padding: 10px 20px;
}
a{
  background-color: blueviolet;
  color: white;
  text-align: center;
  text-transform: uppercase;
  text-decoration: none;
  font-size: 18px;
  transition: 0.5s;
}
a:hover{
  background-color: dodgerblue;
}
.cards img{
width: 100px;
}

```

home/templates/home.html

```

<!DOCTYPE html>
<html lang="en">

<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <!--<meta http-equiv="X-UA-Compatible" content="IE=edge"> -->
    <title>Gas-N-GO</title>
    <script
      src="https://code.jquery.com/jquery-3.6.0.min.js"
      integrity="sha256-/xUj+30JU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4="
      crossorigin="anonymous"></script>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.min.css">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/materialize.min.js"></script>
    <link rel="icon" href="static/Gas-N-Go/favicon.ico">

  </head>
  <body>
    <nav>
      <div class="nav-wrapper">
        <a href="/" class="brand-logo p3">GAS N GO</a>
      </div>
    </nav>

```

```

{% if user.is_authenticated %}
<!-- TODO: design logout button -->
<h5 style="background-color:rgba(255, 99, 71, 0.5);color: black;">
<a href="{% url 'logout' %}">Logout</a>
</h5>
<h5 style="background-color:rgba(255, 99, 71, 0.5);color:black;text-align:left;">
<a href="{% url 'preview' %}">Booking Information</a>
</h5>
<div class="container" style="margin-top:50px">
    <div class="row">
        <div class="col m4">
            <div class="input-field col s12">
                <select multiple id="emenities" onchange="getGas()" >
                    <option value="" disabled selected>Choose your place</o
ption>

                    <option value="">option</option
on>
                    <!-- {% for emenitie in emenities %}
                    <option value="{{emenitie.id}}">{{emenitie.name}}</opti
on>

                    {% endfor %}-->
                </select>
                <label>Select Emenities</label>
            </div>
        </div>
        <div class="col m4">
            <label>Amount</label>
            <p class="range-field">
                <input type="range" onchange="getGas()" id="available_amount" min="
0" max="15" value="15">
            </p>

            </div>
        </div>
        <div class="container">
            <div class="row" id="show_gas_here"></div>
        </div>
    </div>
{% else %}
<!-- TODO: design login button -->
<h2>
<a href="{% url 'login' %}">Login</a>
</h2>
{% endif %}

<script>

var show_gas_here = document.getElementById("show_gas_here")
$(document).ready(function(){
    $('select').formSelect();
});

```

```

function getGas(){
    var available_amount = document.getElementById('available_amount')
    var instance =M.FormSelect.getInstance(document.getElementById('emeniti
es'))

    //if(instance)
    // console.log(instance.getSelectedValues())
    var emenities = ''
    var html = ''
    if(instance){
        emenities = (instance.getSelectedValues())
    }
    fetch(`/api/gas/?emenities=${emenities}&available_amount=${available_am
ount.value}`)
        .then(result => result.json())
        .then(response => {
            // console.log(response)
            // })
            for(var i=0; i <response.length; i++){
                html += `
                <div class="col s5 m3"
                <div class="col s4 m4">
                    <div class="card">
                        <div class="card-image">
                            
                        </div>
                        <span class="card-title black-
text">${response[i].pump_name}</span>

                        <div class="card-content">
                            Available amount - <span class="badge">${response[i].a
vailable_amount}</span>

                            <h6>Address:</h6>
                            <p>${(response[i].pump_address).substring(0 , 250)}</p>

                        </div>
                        <div class="card-action">
                            <a href="% url 'contact' %">Confirmation</a>
                        </div>
                    </div>

                </div>
                </div>
            `
            }
            show_gas_here.innerHTML = html
            //console.log(html)
        })
}

```

```

    }
    getGas()
  </script>

</body>
</html>

```

home/templates/payment.html

```

{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Payment Form</title>
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link href="https://fonts.googleapis.com/css2?
family=Poppins:ital,wght@0,500;0,600;0,700;1,400&display=swap" rel="stylesheet">
  <link rel="stylesheet" href="{% static 'home/styles/payment.css' %}">
</head>
<body>
  <div class="container">
    <h1>Confirm Your Payment</h1>
    <div class="first-row">
      <div class="owner">
        <h3>Owner</h3>
        <div class="input-field">
          <input type="text">
        </div>
      </div>
      <div class="cvv">
        <h3>CVV</h3>
        <div class="input-feild">
          <input type="password">
        </div>
      </div>
    </div>
    <div class="second-row">
      <div class="card-number">
        <h3>Card Number</h3>
        <div class="input-fileld">
          <input type="text">
        </div>
      </div>
    </div>
  </div>

```

```

        </div>
    </div>
    <div class="third-row">
        <h3>Expiration Date</h3>
        <div class="selection">
            <div class="date">
                <select name="months" id="months">
                    <option value="Jan">Jan</option>
                    <option value="Feb">Feb</option>
                    <option value="Mar">Mar</option>
                    <option value="Apr">Apr</option>
                    <option value="May">May</option>
                    <option value="Jun">Jun</option>
                    <option value="Jul">Jul</option>
                    <option value="Aug">Aug</option>
                    <option value="Sep">Sep</option>
                    <option value="Oct">Oct</option>
                    <option value="Nov">Nov</option>
                    <option value="Dec">Dec</option>
                </select>
                <select name="years" id="years">
                    <option value="2021">2021</option>
                    <option value="2020">2020</option>
                    <option value="2019">2019</option>
                    <option value="2018">2018</option>
                    <option value="2017">2017</option>
                    <option value="2016">2016</option>
                </select>
            </div>
            <div class="cards">
                
                
                
            </div>
        </div>
    </div>

    </div>
    <a href="">Confirm</a>

</div>

</body>
</html>

```

home/admin.py

```
from django.contrib import admin
from .models import *

admin.site.register(Emenities)
admin.site.register(Gas)
```

home/apps.py

```
from django.apps import AppConfig

class HomeConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'home'
```

home/models.py

```
from django.db import models

# Create your models here.

class Emenities(models.Model):
    name = models.CharField(max_length=100)

    def __str__(self):
        return self.name

class Gas(models.Model):
    pump_name = models.CharField(max_length=100)
    pump_image = models.CharField(max_length=500)
    pump_address = models.TextField()
    available_amount = models.IntegerField()
    emenities = models.ManyToManyField(Emenities)

    def __str__(self):
        return self.pump_name
```

home/tests.py

```
from django.test import TestCase

# Create your tests here
```



## home/urls.py

```
from django.contrib import admin
from django.urls import path
from . import views
from .views import *

urlpatterns = [
    path('', home , name="home"),
    path('cart/', cart , name="cart"),
    path('payment/', payment , name="payment"),
    path('api/gas/', api_gas)
]
```

## home/views.py

```
from django.shortcuts import render
from .models import *
from django.http import JsonResponse

# Create your views here.

def home(request):
    emenities = Emenities.objects.all()
    context = {'emenities': emenities}
    return render(request, 'home.html', context)

def cart(request):
    return render(request, 'cart.html')

def payment(request):
    return render(request, 'payment.html')

def api_gas(request):
    gas_objs = Gas.objects.all()
    available_amount = request.GET.get('available_amount')
    if available_amount:
        gas_objs = gas_objs.filter(available_amount__lte=available_amount)

    emenities = request.GET.get('emenities')
    if emenities:
        emenities = emenities.split(',')
        em = []
        for e in emenities:
            try:
                em.append(int(e))
```

```

        except Exception as e:
            pass
        gas_objs = gas_objs.filter(emenities__in=em).distinct()

    payload = []
    for gas_obj in gas_objs:
        result = {}
        result['pump_name'] = gas_obj.pump_name
        result['pump_image'] = gas_obj.pump_image
        result['pump_address'] = gas_obj.pump_address
        result['available_amount'] = gas_obj.available_amount
        payload.append(result)
    return JsonResponse(payload, safe=False)

```

## Codes Related for Booking

booking/static/booking/styles/style.css

```

body {
    background-color: lightblue;
}

```

booking/templates/book\_view.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Booking Information</h1>
    <h2>{{ user.username }}</h2>
    {% for item in items %}
    <div class="box" style="border: 5px solid black;">
    <h2>{{ item.id }}</h2>
    <h3>Fuel Type: {{ item.fuel_type }}</h3>
    <h5>Amount: {{ item.fuel_amount }} (ltr)</h5>
    <h5>Price: {{ item.price }} (Taka)</h5>
    </div>
    {% endfor %}
</body>
</html>

```

## booking/templates/cat.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="{% static 'booking/styles/style.css' %}">
</head>
<body>
    <form action="book" method="POST" id="register" class="input-group">
        {% csrf_token %}
        <input type="text" name="name" class="input-
field" placeholder="User Name" required>
        <input type="password" name="password" class="input-
field" placeholder="Enter Password" required>
        {% comment %} <input type="text" name="fuel_type" class="input-
field" placeholder="Fuel Type" required> {% endcomment %}
        <select name="fuel_type" id="fuel_type">
            <option value="select">Select</option>
            <option value="petrol">Petrol</option>
            <option value="octane">Octane</option>
            <option value="diesel">Diesel</option>
            <option value="natural_gas">Natural Gas</option>
            <option value="lpg">LPG</option>
        </select>
        <input type="text" name="fuel_amount" class="input-
field" placeholder="Amount" required>
        {% comment %} <input type="text" name="fuel_price" class="input-
field" placeholder="Price" required> {% endcomment %}
        <button type="submit" class="submit-btn">Booked</button>
    </form>
</body>
</html>
```

## booking/admin.py

```
from django.contrib import admin
from .models import *

admin.site.register(Contact)
```

## booking/apps.py

```
from django.apps import AppConfig

class BookingConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'booking'
```

## booking/forms.py

```
from django import forms
from .models import Contact

class ContactModelForm(forms.ModelForm):
    class Meta:
        model = Contact
        fields = '__all__'
```

## booking/models.py

```
from django.db import models

# Create your models here.
class Contact(models.Model):
    name = models.CharField(max_length=124)
    password = models.CharField(max_length=20)
    fuel_type = models.CharField(max_length=20)
    fuel_amount = models.CharField(max_length=20)
    price = models.CharField(max_length=20)

    def __str__(self):
        return self.name
```

## booking/tests.py

```
from django.test import TestCase
```

## booking/urls.py

```
from django.contrib import admin
from django.urls import path
from . import views
```

```

from .views import *

urlpatterns = [
    path('', contact , name='contact'),
    path('bookingview', preview , name='preview'),
    path('book', book , name='book'),
]

```

## booking/views.py

```

from typing import ContextManager
from django.http import JsonResponse
from django.shortcuts import render, redirect
from .forms import ContactModelForm
from .models import Contact

def contact(request):
    return render(request, 'cat.html')

def preview(request):
    context = {
        'items': Contact.objects.all()
    }
    return render(request, 'book_view.html', context)

def book(request):
    if request.method == 'POST':
        name = request.POST['name']
        fuel_type = request.POST['fuel_type']
        fuel_amount = request.POST['fuel_amount']
        password = request.POST['password']
        if fuel_type == 'petrol':
            price = int(int(fuel_amount) * 52)
        elif fuel_type == 'octane':
            price = int(int(fuel_amount) * 95)
        elif fuel_type == 'diesel':
            price = int(int(fuel_amount) * 65)
        elif fuel_type == 'natural_gas':
            price = int(int(fuel_amount) * 64)
        else:
            price = int(int(fuel_amount) * 42)
    
```

```

        book = Contact.objects.create(name=name, password=password, fuel_type=fuel_
type, fuel_amount=fuel_amount, price=price)
        book.save()
        return redirect('preview')
    else:
        return render(request, 'cat.html')

```

## Codes for venv:

### yvenv.cfg

```

home = c:\users\user\anaconda3
implementation = CPython
version_info = 3.8.8.final.0
virtualenv = 20.7.2
include-system-site-packages = false
base-prefix = c:\users\user\anaconda3
base-exec-prefix = c:\users\user\anaconda3
base-executable = c:\users\user\anaconda3\python.exe

```

### venv/manage.py

```

"""Django's command-line utility for administrative tasks."""
import os
import sys

def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'core.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)

if __name__ == '__main__':
    main()

```

### requirements.txt

```

asgiref==3.4.1
Django==3.2.6

```

```
pytz==2021.1  
sqlparse==0.4.1
```