Case Study - Gas Utility Application

Introduction

In todays time, customer satisfaction, customer safety, service and customer retention plays an important role in any service based company. A Gas Utility company is facing significant challenges due to High volume of customer service requests. And the existing system is not able to manage it properly.
This case study will provide an effect solution to develop an application using Django to overcome all this problem.

Problem Statement:

A gas utility company is experiencing a high volume of customer service requests. The company's current system is not able to handle the volume of requests, and customers are experiencing long wait times and poor service

- Long wait times
- Inefficient request management
- Poor customer experience

These issues affect customer retention and impact the overall reputation of the company.

Features Required:

1. Service Requests
    - Submit Service requests online
        - Ability to select the type of service requests
        - Provide details about the request
        - Attach Files
2. Request tracking
    - Track the status of their service requests.
        - Ability to see the status of the requests
        - Date and time of the request submitted
        - Data and time the request was resolved

3. View Customer account information
4. The application will equip customer support representatives to manage requests and assist customers effectively.

Why should we use **Django (Python Framework)** to develop this application ?

- Primary Goal - Data Driven Database based web application.
- Easy to switch between database
- Focus on Reusability, Pluggability, and on Maintenance
- Based on Do Not Repeat Yourself (DRY principal)
- Django is Based on model view template.

> Django is a Backend Frame work but we need HTML and CSS to develop working front end for fully functional application.

Possible solution:

1. **Service Request Submission - Client side**

   - Customers will be able to submit service requests online.
   They can select the type of request - (New service, Issue with the product, exchange, etc.)
   - Provide Detailed Description - (About the problem)
   - Attach relevant files - (Images and Video to give an idea about the problem)

In application an Form feature will take this request and this requests will be write (saved) on the database.

2. **Request Tracking - Client side**

   - Application will allow the customers to track the status of their requested service in real time.
   - Which shows Current status, Data - time of Submission and Resolution of request.
   - To provide this support we can give access to Admin to update the status of request in real time.
   - With transparent request tracking, Customers gets the feeling satisfaction and more informed.

3. **Customer Account Information - Client side**

   - Dashboard
   - Customers will have access to their account information
   - Function → Update personal details, View service history, Billing information.

- Settings, Pre-order, and all user related function will be present on the dashboard

4. **Customer Support - Client side**

   - A bot can be prepared on frequency asked Questions so similar queries can be answered by the bot and if user has a unique query which i unsolved by the bot, then Customer support representative will reach him out for solving the problem.

5. **Admin Dashboard - Admin side**

   - Customer Support representative will have extra functionality, with access to each stage and part of the application.
   - To manage incoming requests, Change statuses as request is getting processed, Can communicate directly with customers regarding there requests.

6. **User Authentication**

   - To secure the data and restrict the access in certain regions we need secure login system for User and Admin.

Implementation - Project

1. Project Setup - We require an IDE to set up the project (Py-charm and Web-strom).
2. Define Model
3. Create Form - For user input.
4. View Implementation - To display requests and handle forms submissions.
5. Template Development - To design user friendly templates - Using Front end technology.
6. Testing - to make sure each function is working properly.
7. Deployment - To make it official available to actual customer for there daily use.

Our expected Result

- To Increase the efficiency of the application.
- Make customer satisfaction much better than previous application.
- To connect User and Admin seamless to solve problems and maintain an transparent environment.
- Reduced Waiting Time
- Always access to an representative for any queries.

System Design (Architecture of Application)

```
gas_utility
│
├── manage.py
├── README.md
├── db.sqlite3
│
├── gas_utility
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   ├── wsgi.py
│   ├── asgi.py
│
├── requests
│   ├── migrations
│   │    ├── __init__.py
│   │    ├── 0001_initial.py
│   │
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── forms.py
│   ├── models.py
│   ├── tests.py
│   ├── views.py
│   └── urls.py
│
└── templates
    ├── base.html
    └── requests
          ├── request_form.html
          ├── request_list.html
          └── request_detail.html
```

Conclusion:

The development of an application based on Django for managing customer service requests presents an effective solution to over come the challenges faced by the existing Gas Utility System. By enhancing the customer experience through improved, fast and transparent system company can retain their existing customers and also attract new customers with there Interactive, Fast, Secure service.

By Krishna Birla.