

Gas Utility Management System

Introduction

The Gas Utility Management System is a Django-based web application designed to improve the customer service experience for a gas utility company. The system addresses the challenges of high volumes of customer service requests by providing a streamlined platform for managing service requests, tracking their status, and offering customer support tools.

Features

For Customers:

1. **Service Requests**
 - Submit requests online.
 - Select the type of service required.
 - Provide details and attach files.
2. **Request Tracking**
 - View the status of submitted requests.
 - See submission and resolution timestamps.

For Customer Support Representatives:

1. **Request Management**
 - View, update, and resolve service requests.
 - Change request statuses (e.g., pending, in-progress, resolved).
-

Functionalities

Customer Views

- **Signup/Login:** Users can create accounts and log in to access the system.
- **Submit Requests:** A form allows users to submit detailed service requests.
- **View Requests:** Customers can view all their submitted requests and their current statuses.

Admin/Support Views

- **Manage Requests:** Admins can view, update, and resolve service requests.
 - **Permissions:** Only authorized users (e.g., staff) can access the admin panel and manage customer requests.
-

Application Structure

Project Structure

```
GasUtilitySystem/
├── customer_service/
│   ├── migrations/
│   ├── templates/
│   │   ├── registration/ # Login & Signup Pages
│   │   └── customer_service/ # Service request templates
│   ├── admin.py
│   ├── models.py
│   ├── views.py
│   ├── forms.py
│   ├── urls.py
│   └── tests.py
├── static/
├── media/
├── manage.py
└── db.sqlite3
```

Key Files and Their Purpose

- **models.py:** Defines the `ServiceRequest` model for customer requests.
 - **views.py:** Contains logic for handling user requests and interactions.
 - **forms.py:** Includes forms for user input, such as service requests.
 - **urls.py:** Maps URLs to their respective views.
 - **templates:** HTML files for rendering the front-end.
-

Technical Details

Models

`ServiceRequest`

```
class ServiceRequest(models.Model):
```

```

customer = models.ForeignKey(User, on_delete=models.CASCADE)
service_type = models.CharField(max_length=255)
details = models.TextField()
attachment = models.FileField(upload_to='attachments/', blank=True, null=True)
status = models.CharField(
    max_length=50,
    choices=[('Pending', 'Pending'), ('In Progress', 'In Progress'), ('Resolved', 'Resolved')],
    default='Pending'
)
created_at = models.DateTimeField(auto_now_add=True)
updated_at = models.DateTimeField(auto_now=True)

```

Forms

- **ServiceRequestForm:** For submitting and updating service requests.

Views

- **home:** Renders the homepage or dashboard.
- **signup/login:** Handles user authentication.
- **create_service_request:** Allows customers to submit requests.
- **service_requests:** Displays all requests for a logged-in user.
- **update_service_request:** Enables staff to update the status of requests.

URL Patterns

```

urlpatterns = [
    path("", views.home, name='home'),
    path('signup/', views.signup, name='signup'),
    path('login/', views.login_view, name='login'),
    path('service/create_service_request/', views.create_service_request, name='create_service_request'),
    path('service/service_requests/', views.service_requests, name='service_requests'),
    path('service/<int:request_id>/', views.service_request_detail, name='service_request_detail'),
    path('service/<int:request_id>/update/', views.update_service_request, name='update_service_request'),
]

```

Deployment

1. **Setup:** Install Django and required dependencies.
2. **Run Server:** Start the development server using `python manage.py runserver`.
3. **Database:** Migrate the database using `python manage.py migrate`.
4. **Admin Panel:** Use the Django admin interface for advanced request management.

Future Enhancements

1. **Account Management**
 - Add account information such as billing details and usage history.
 2. **Notifications**
 - Notify customers via email or SMS when their request status changes.
 3. **Analytics Dashboard**
 - Provide customer support with insights into service request trends.
-

Conclusion

This application effectively addresses the high volume of customer service requests for the gas utility company by providing an efficient, user-friendly platform for request submission, tracking, and management. Future enhancements can further improve customer experience and operational efficiency.