Gas Utility System Documentation

Table of Contents

Introduction

The Gas Utility System is a web-based application designed to manage gas utility service requests. It allows users to submit gas service requests, view the status of their requests, and provides an admin interface for managing users and requests. The system is built using the Django framework and follows best practices in web application development to ensure scalability, security, and ease of use.

Key Features:

- **Request Submission: Users can submit requests for gas services.**
- **Request Status: Users can track the status of their requests.**
- **Admin Interface: Admins can manage requests and user data.**
- **Responsive Design: The system is designed to work across a variety of devices.**

Technologies Used

The following technologies are used to build and deploy the Gas Utility System:

- **Django (Web framework for Python)**
- **Python (Programming language)**
- **HTML/CSS/JavaScript (Frontend technologies)**
- **SQLite (Database, default)**
- **Bootstrap (CSS framework for responsive design)**
- **Git (Version control)**
- **Heroku/AWS (Optional for deployment)**

Setup Instructions

Follow these steps to set up the project locally.

3.1. Installing Dependencies

Clone the repository and install the necessary dependencies using the following commands:

1. **Clone the project repository:**

git clone https://github.com/yourusername/gas-utility-system.git

cd gas-utility-system

2. **Create a virtual environment (optional but recommended):**

python -m venv venv

source venv/bin/activate # On Windows: venv\Scripts\activate

3. **Install the required dependencies:**

pip install -r requirements.txt

3.2. Setting Up the Database

Django uses SQLite by default. If you want to use a different database, modify the DATABASES setting in the settings.py file.

To set up the database:

1. **Run Django migrations:**

python manage.py migrate

3.3. Running the Application

To run the application locally, use the following command:

python manage.py runserver

Visit http://127.0.0.1:8000/ in your browser to view the application.

Project Structure

Here is the file structure of the Gas Utility System:

```
gas_utility/
├── gas_utility/              # Core project files (settings, urls, wsgi)
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
├── customers/                # Customer-facing app (request handling, views)
│   ├── migrations/
│   ├── __init__.py
│   ├── admin.py              # Admin panel configuration
│   ├── apps.py
│   ├── models.py             # Database models
│   ├── views.py              # Views for request submission, status, etc.
│   ├── urls.py               # URL routing for customers app
│   └── templates/            # HTML templates for views
│       └── home.html
├── db.sqlite3                # Database file (SQLite by default)
├── manage.py                 # Django project manager script
└── requirements.txt          # List of required Python packages
```

Features

1. **User Registration and Login: Users can register, log in, and manage their accounts.**
2. **Request Submission: Users can submit gas service requests with their address and other details.**
3. **Request Tracking: Users can track the status of their submitted requests.**
4. **Admin Panel: Admins can manage users, requests, and view data using Django's built-in admin interface.**
5. **Responsive Design: The system is responsive, ensuring compatibility with mobile devices.**

API Endpoints

- **POST /api/submit-request: Submits a gas request with the provided details.**
- **GET /api/request-status/{request_id}: Retrieves the status of a specific gas request by its ID.**

Example Request for submitting a request (POST):

POST /api/submit-request

{

"user_id": 1,

"address": "123 Main St, City, Country"

}

Example Response:

{

"request_id": 101,

"status": "Pending"

}

Example Request for checking status (GET):

GET /api/request-status/101

Example Response:

{

"request_id": 101,

"status": "Pending",

"address": "123 Main St, City, Country",

"requested_on": "2025-01-20T10:00:00"

}

Admin Panel

Django's built-in admin panel allows administrators to manage users, gas requests, and other data. To access the admin panel:

1. **Ensure that a superuser has been created using the python manage.py createsuperuser command.**
2. **Access the admin panel at http://127.0.0.1:8000/admin/.**

Testing

Testing is essential for ensuring the quality and reliability of the Gas Utility System. Django includes a testing framework that makes it easy to write and execute tests.

To run tests:

python manage.py test

Tests for models, views, and APIs can be added in the tests.py file within the customers app.

Deployment

The Gas Utility System can be deployed on various cloud platforms, such as Heroku or AWS. Here is an example of how to deploy to Heroku:

1. **Install the Heroku CLI.**
2. **Log in to your Heroku account:**

heroku login

3. **Create a new Heroku app:**

heroku create gas-utility-system

4. **Push the code to Heroku:**

git push heroku master

5. **Open the application:**

heroku open

For more detailed deployment instructions, consult the Heroku documentation or similar guides for AWS.

Security

To secure the Gas Utility System, the following measures should be taken:

1. **Use HTTPS: Ensure the app uses HTTPS to secure user data in transit.**
2. **Sanitize Input: Always sanitize user input to prevent injection attacks.**
3. **Store Passwords Securely: Django handles password security, but always ensure you use a strong hashing algorithm.**
4. **Update Dependencies Regularly: Regularly update all dependencies to their latest versions to patch any security vulnerabilities.**

License

The Gas Utility System is released under the MIT License. See the LICENSE file for more details.

Conclusion

The Gas Utility System is a comprehensive web application for managing gas utility service requests. It provides users with an easy-to-use interface for submitting and tracking requests, while administrators can manage users and requests via the Django admin panel. The system is designed to be scalable, secure, and responsive.

Next Steps

- **User Authentication: Implement user authentication and login/logout functionality.**
- **Deployment: Deploy the system to a production server like Heroku or AWS.**
- **Enhancements: Add features such as email notifications, request history, or advanced reporting.**