Detailed Report: Developing a Django-Based URL Shortening Service with Slack Integration

Context:

The customer requires an internal URL shortening service that integrates with Slack to allow employees to easily generate short links from long URLs. This service will be built using **Django**, hosted on **Heroku**, and integrated with a **Slack bot** to provide a seamless interface for users.

Requirements Breakdown:

1. **URL Shortening Service**:
   - The service will take a long URL as input and return a short URL.
   - It must be self-hosted (i.e., no third-party services like bit.ly).
   - The service should be simple to use and operate internally.
2. **Slack Bot Integration**:
   - Users will interact with the service through Slack.
   - Employees will send long URLs via the Slack bot, which will return the shortened version of the link.
   - The Slack bot needs to communicate with the URL shortening service hosted on Heroku.

Technical Steps to Implement the Solution:

1. Build the URL Shortening Service:

Django Project Setup:

- Start by creating a new Django project and app to handle URL shortening.

```
django-admin startproject url_shortener
cd url_shortener
django-admin startapp shortener
```

Model Design:

- Create a `URL` model to store long URLs and their corresponding shortened codes.

```
from django.db import models
import string
import random

def generate_short_code():
    return ''.join(random.choices(string.ascii_letters +
string.digits, k=6))

class URL(models.Model):
```

```
    long_url = models.URLField()
    short_code = models.CharField(max_length=6, unique=True,
default=generate_short_code)
    created_at = models.DateTimeField(auto_now_add=True)
```

Views and Logic:

- Add a view for the URL shortening functionality, which accepts a long URL, generates a short code, and stores it in the database.

```
from django.shortcuts import render, redirect
from .models import URL

def shorten_url(request):
    if request.method == 'POST':
        long_url = request.POST.get('url')
        url_obj, created =
URL.objects.get_or_create(long_url=long_url)
        return render(request, 'shortener/result.html', {'short_url':
request.build_absolute_uri(url_obj.short_code)})
    return render(request, 'shortener/index.html')

def redirect_url(request, short_code):
    try:
        url_obj = URL.objects.get(short_code=short_code)
        return redirect(url_obj.long_url)
    except URL.DoesNotExist:
        return render(request, 'shortener/404.html')
```

Templates:

- `index.html` allows the user to input a long URL, and `result.html` displays the shortened link.

URLs Setup:

- Define routes for shortening and redirecting URLs in the `urls.py` file.

```
from django.urls import path
from .views import shorten_url, redirect_url

urlpatterns = [
    path('', shorten_url, name='shorten_url'),
    path('<str:short_code>/', redirect_url, name='redirect_url'),
]
```

Heroku Deployment:

- **Procfile**: Create a `Procfile` to define the Heroku web server.

```
web: gunicorn url_shortener.wsgi
```

- **requirements.txt**: Ensure all necessary Python dependencies are listed.

```
pip freeze > requirements.txt
```

- **settings.py**: Add Heroku settings using `django-heroku`.

```
python
Copy code
import django_heroku
django_heroku.settings(locals())
```

- Deploy the application on Heroku:

```
bash
Copy code
heroku create your-app-name
git push heroku master
heroku run python manage.py migrate
```

---

2. Integrating the Slack Bot:

Create a Slack App:

- Go to the Slack API and create a new app.
- Enable **Bots** and grant the app **chat**

  permissions.

Bot Token and API Setup:

- In the **OAuth & Permissions** section, find your **Bot Token**.
- Install the `slack_sdk` Python package to interact with the Slack API.

```
pip install slack-sdk
```

Slack View for URL Shortening:

- Create a Django view to handle Slack bot requests.

```
python
Copy code
from django.http import JsonResponse
from slack_sdk import WebClient
from slack_sdk.errors import SlackApiError
from django.views.decorators.csrf import csrf_exempt
import json

SLACK_BOT_TOKEN = 'xoxb-your-slack-bot-token'
slack_client = WebClient(token=SLACK_BOT_TOKEN)

@csrf_exempt
def slack_shortener(request):
    if request.method == 'POST':
        data = json.loads(request.body)
        long_url = data['text']
        url_obj, created =
URL.objects.get_or_create(long_url=long_url)
        short_url = request.build_absolute_uri(url_obj.short_code)
```

```
        try:
            slack_client.chat_postMessage(channel=data['channel_id'],
text=f'Shortened URL: {short_url}')
        except SlackApiError as e:
            return JsonResponse({'error': str(e)})

        return JsonResponse({'message': 'URL shortened
successfully'})
```

Slack URL Configuration:

- Update your `urls.py` to include the Slack view.

```
from .views import slack_shortener

urlpatterns = [
    path('slack/', slack_shortener, name='slack_shortener'),
]
```

Slack Bot Interaction:

- Users can interact with the bot by sending a message in the format `/shorten https://longurl.com`, and the bot will respond with the shortened version of the URL.

Heroku Environment Variable:

- Set the **SLACK_BOT_TOKEN** as an environment variable on Heroku:

```
heroku config:set SLACK_BOT_TOKEN='xoxb-your-slack-bot-token'
```

---

3. Testing the Slack Bot:

Once the Slack app is installed in the workspace and connected with the URL shortener:

1. Employees can type a command in Slack with the long URL.
2. The bot sends the request to the Django service hosted on Heroku.
3. The service returns a shortened URL back to the Slack channel.

Conclusion:

This solution allows the internal employees to use a simple URL shortening service, fully hosted on Heroku, and integrates seamlessly with Slack for ease of access. The steps involve setting up a Django-based application for URL shortening, deploying it on Heroku, and integrating it with a Slack bot to handle user requests. The entire process ensures that no external URL shortening service is needed, fulfilling the customer’s requirement of maintaining internal control over the system.