**PROJECT REPORT:**

**UTILITY – VERSATILE WEB APP THAT PROVIDES VARIOUS UTILITY TOOLS AND FUNCTIONS TO USERS**

**By**

**SAI TEJA JEEGURU**

**ID:STB01T0003**

**Abstract**

The Utility App project aims to provide various utility tools and functions to users developed using Django framework. This report outlines the project's objectives, key features, technologies used, implementation details, challenges faced, and future enhancements.

**Table of Contents**

1.Introduction

2.Project Overview

3.Technologies Used

4.System Architecture

5.Implementation Details

6.Challenges Faced

7.Future Enhancements

8.Conclusion

9.References

1. **Introduction**

The Utility App project is various utility tools and functions to users developed using Django framework. It incorporates various sections such as Home page, Notes, Currency Converter, Calculator, About Us Page and Contact Us Page to provide users with a comprehensive utility functionalities experience.

1. **Project Overview**

The project integrates Django framework along with Bootstrap for frontend styling and responsiveness. It features an enhanced admin interface using Django Rosette for easy customization. The App supports multiple tools and functionalities for storing notes data, converting currency and small calculations using calculator. Key functionalities include Notes, Currency Converter, Calculator. Currency Rate are provided using Django's built-in Forex python converter packages and middleware.

1. **Technologies**

Django 5.0.1

Bootstrap 5

SQLite3 (default database engine)

H.T.M.L

C.S.S

JavaScript

Django 5.0.1: Django is a high-level Python web framework that facilitates rapid development of secure and maintainable websites and web applications. It follows the model-view-template (MVT) architectural pattern.

Bootstrap 5: Bootstrap is a popular front-end framework for building responsive and mobile-first web projects. It provides pre-designed templates and components using HTML, CSS, and JavaScript for creating user interfaces.

SQLite3: SQLite is a lightweight, self-contained SQL database engine used as the default database engine in Django projects. It's suitable for small to medium-sized web applications or prototypes.

HTML: HTML (Hypertext Markup Language) is the standard markup language used to create the structure and content of web pages. It defines the elements and tags used to display text, images, links, and other multimedia content on the web.

CSS: CSS (Cascading Style Sheets) is a stylesheet language used to style the presentation and layout of HTML documents. It defines the styles, such as colours, fonts, margins, and padding, to enhance the visual appearance of web pages.

JavaScript: JavaScript is a high-level programming language used to add interactivity, dynamic behaviour, and functionality to web pages. It allows developers to manipulate the HTML and CSS of a webpage, handle user interactions, and perform asynchronous operations.

1. **System Architecture**

The system architecture follows a typical Django MVC (Model-View-Controller) pattern. It involves models for database management, views for handling user requests and rendering templates, and templates for HTML rendering. Forex python converter for currency rates by using currency exchange rate API.

1. **Implementation**

Home Page: The home page contains the nav bar with all the functionalities notes, currency converter, calculator, about us, contact us and log in. The body of the home page contains about us details, frequently asked questions, feedback page and contact us.

Log In Page: This page consists of text box to enter user name and password. The users only able to use the functionalities or tools after login.

Sign Up Page: This page consists of name, email, password and conform password. The new user can create an account and after sign up the page redirects to login page.

Notes: Easily manage your notes for organization and productivity. Access, edit, delete, and create new notes effortlessly. Keep your workspace tidy by removing unnecessary content and capture thoughts and tasks quickly with note creation. Stay organized and productive with our utility app's notes feature.

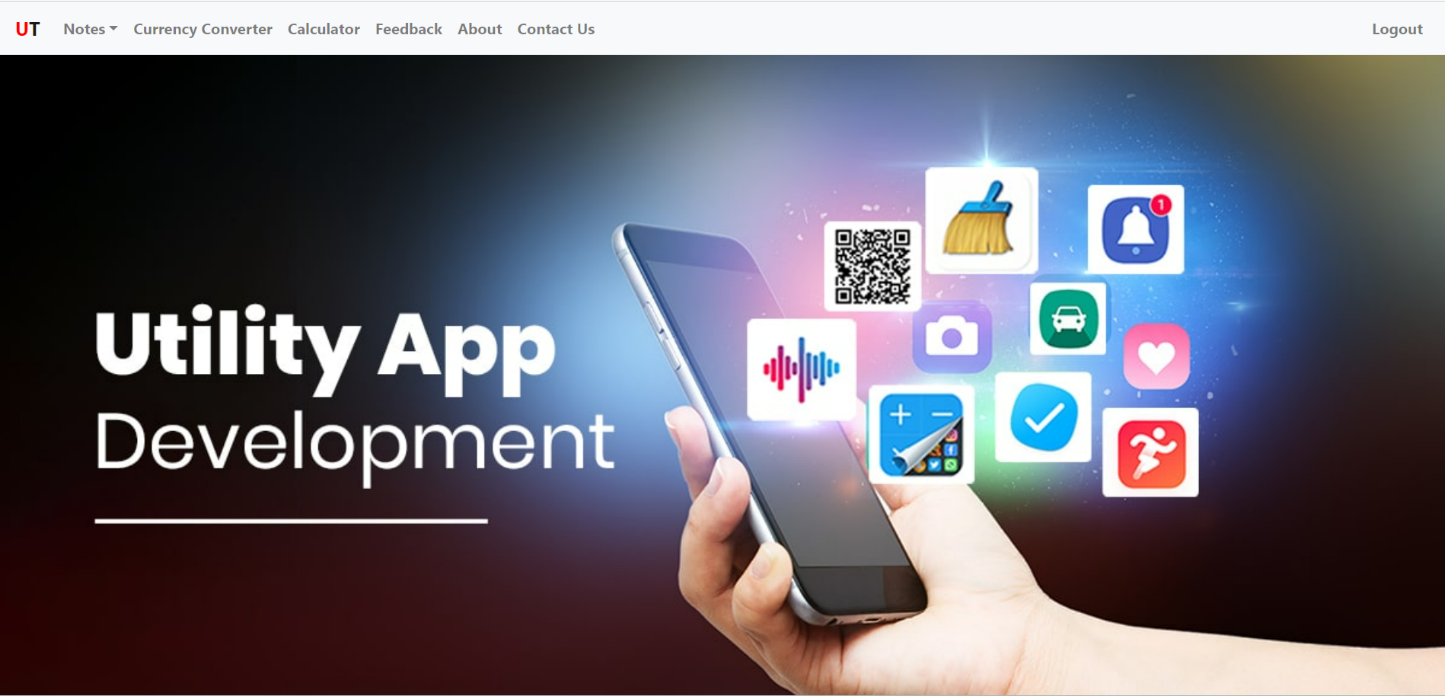
Calculator:Need to crunch some numbers? Our built-in calculator is here to help. Perform basic arithmetic operations, complex calculations, or even solve equations quickly and accurately. With a user-friendly interface, you can perform calculations with ease and precision.

Currency Converter: Traveling abroad or dealing with international transactions? Our currency converter makes it easy to convert between different currencies. Simply select the currencies you want to convert, enter the amount, and instantly see the converted value. Stay up-to-date with accurate exchange rates and streamline your currency conversions.

Feedback Page: The user can give there feedback in this page as good or bad or to improve the UI or more functionalities which can be checked by the admin in the admin page.

About Us: Presents mission and vision of Utility App through visually appealing layout and Bootstrap components. Language options are provided to cater to users from different linguistic backgrounds.

Contact Us: The user can contact to the support team of the mail or a phone number which are provided in the contact us.

Fig 1: Home page

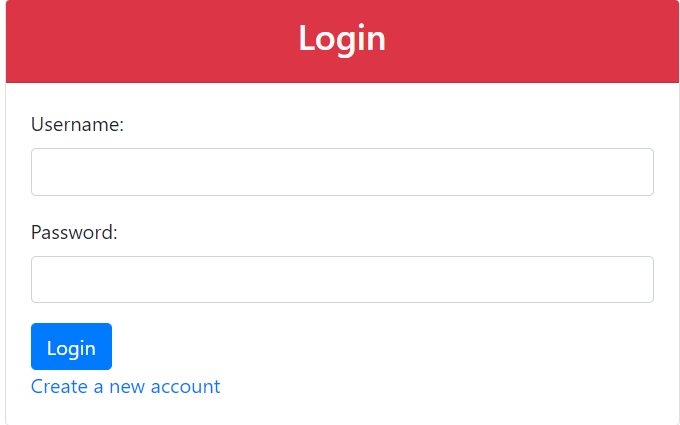


Fig2. Login page

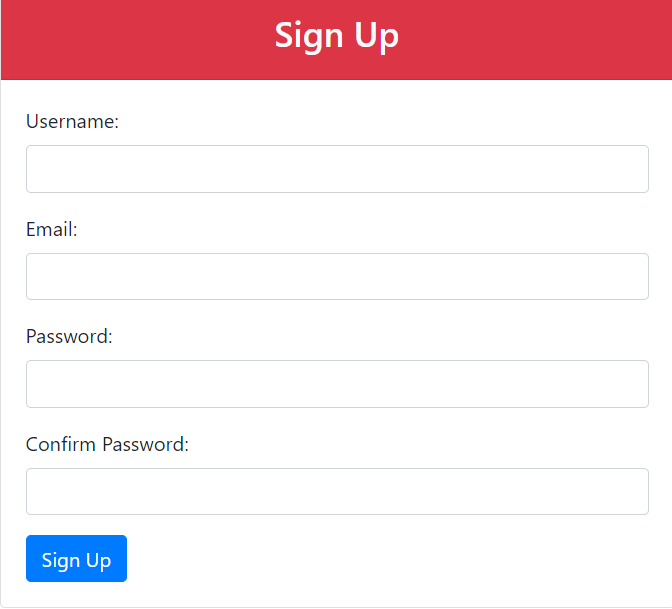


Fig3. Sign up page

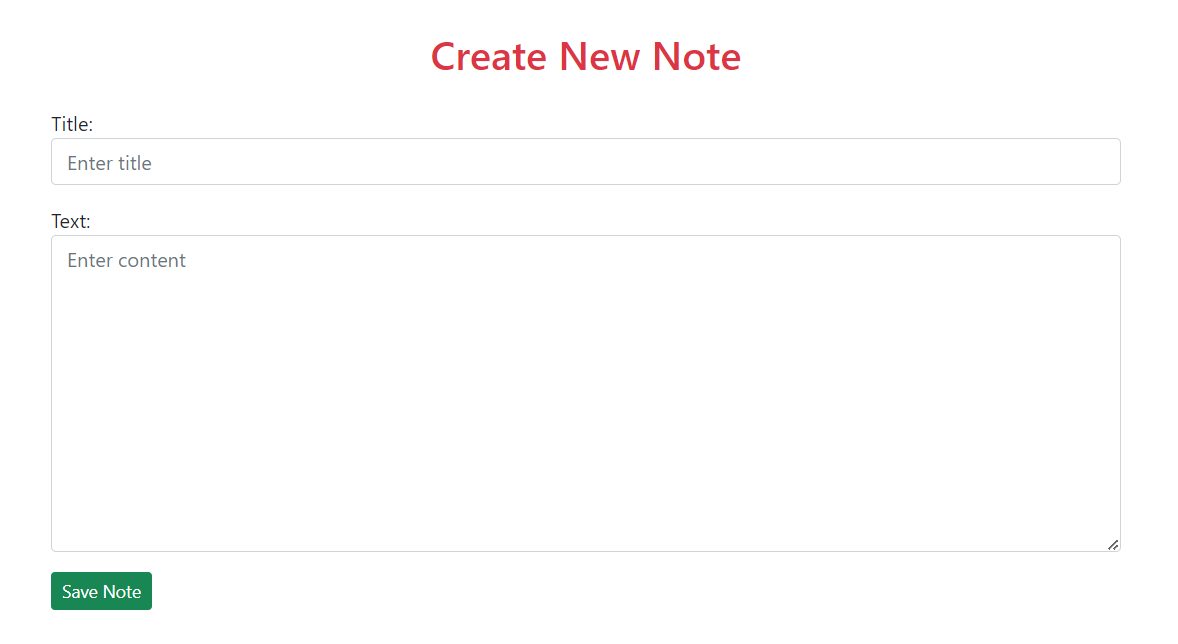


Fig4. Create new notes

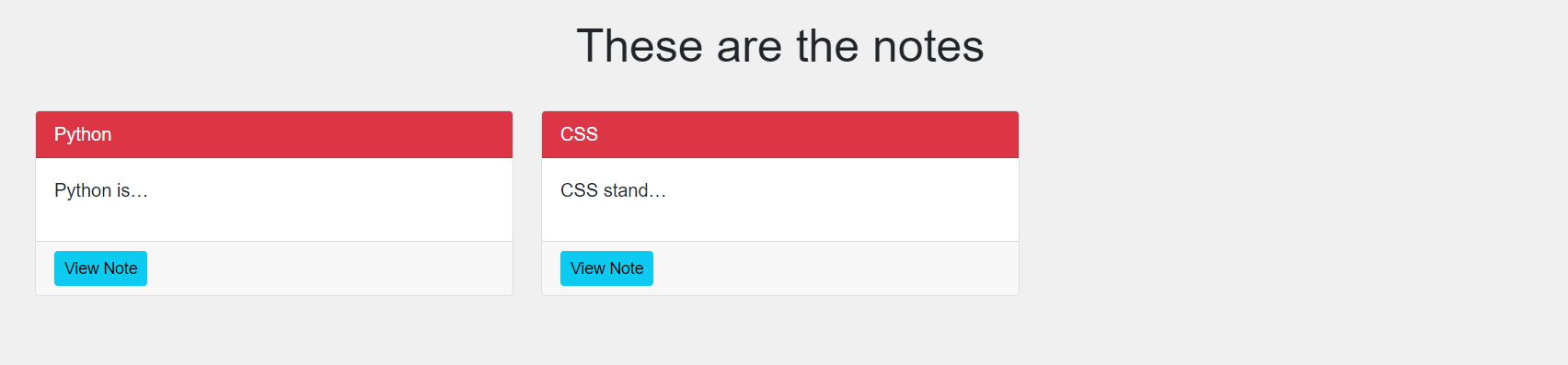


Fig5. All previous notes

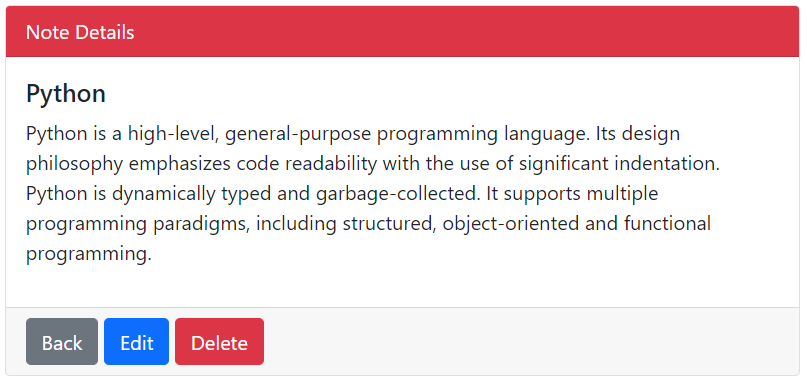


Fig6. Note Details page

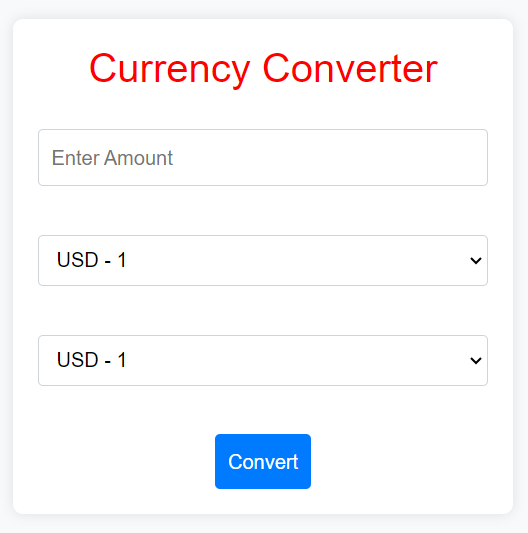


Fig7. Currency Converter page

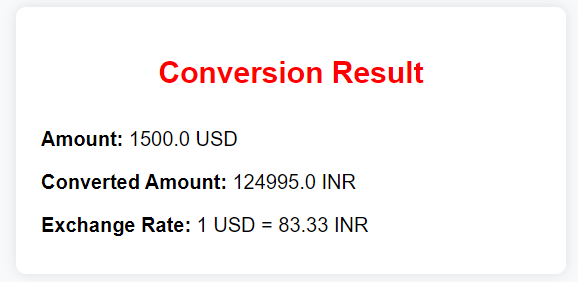


Fig8. Conversion Result Page

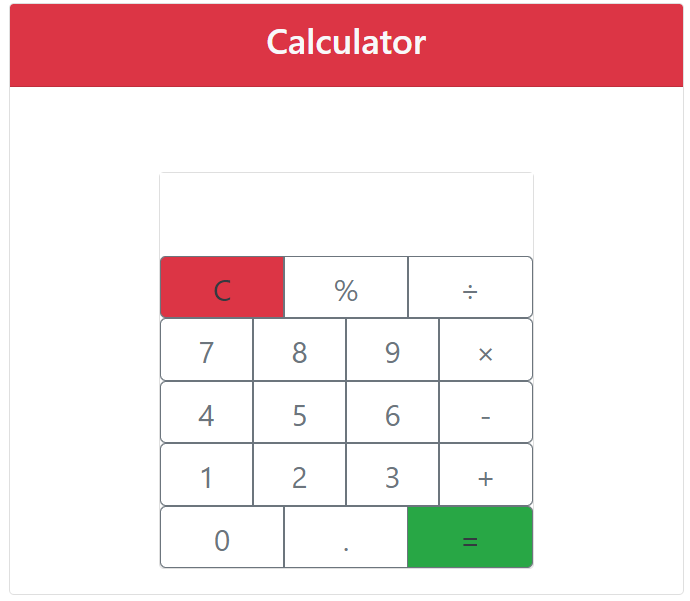
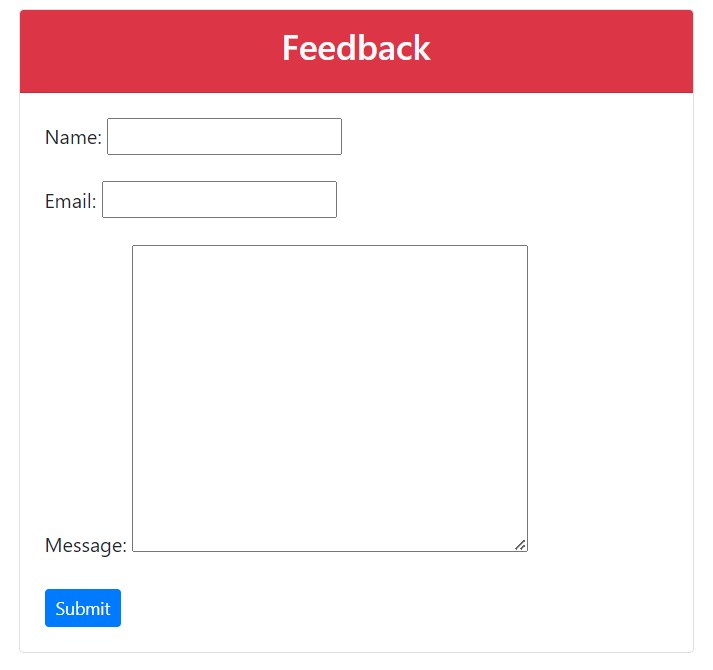


Fig9. Calculator Page

Fig10. Feedback Page

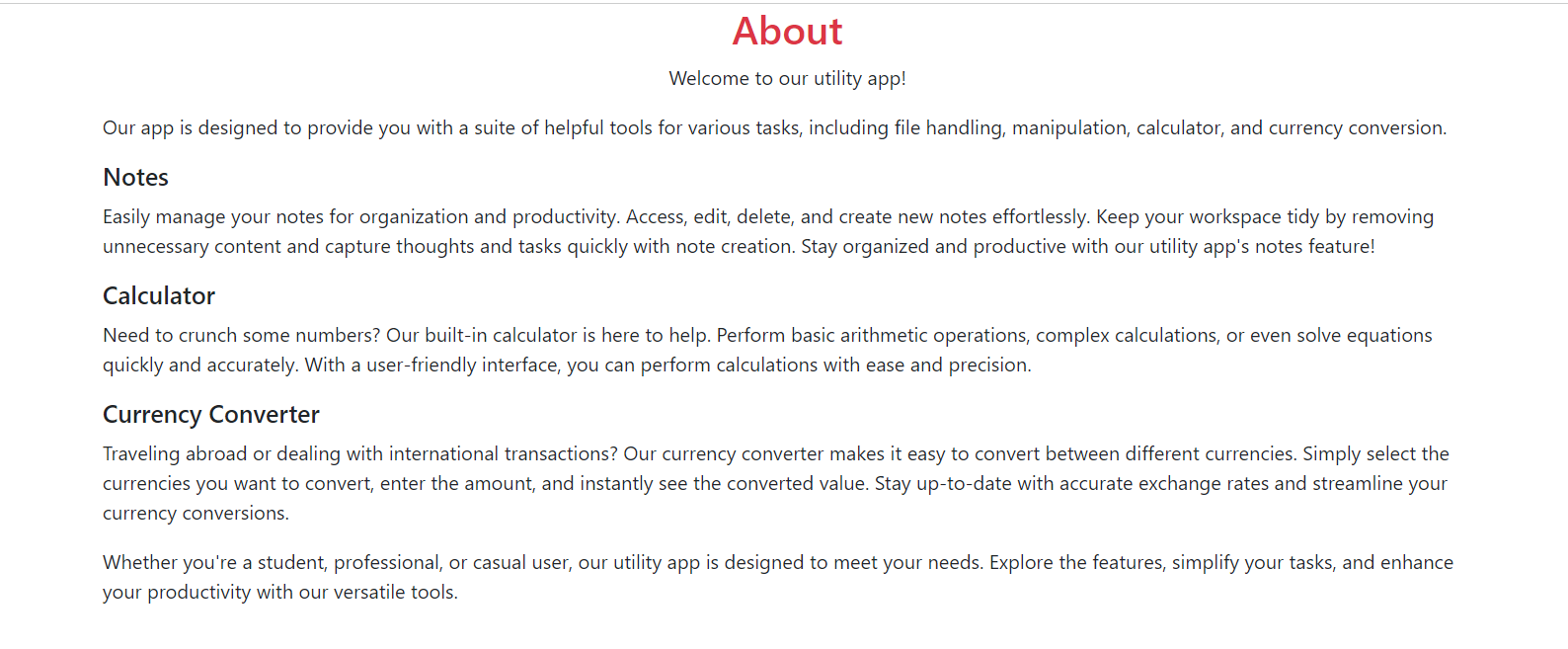


Fig11. About us page

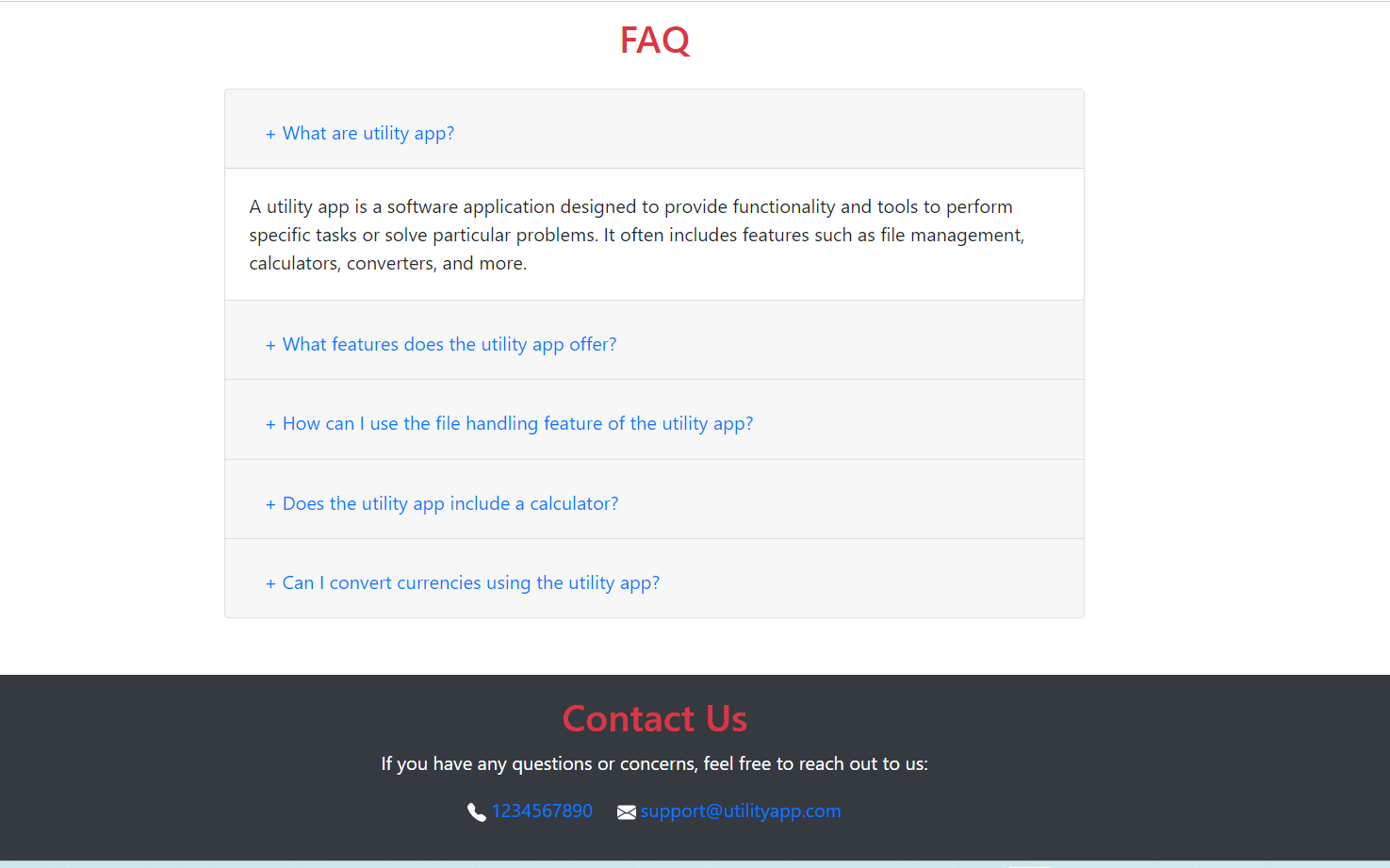


Fig12. FAQ & Contact Us Page

**Home Page:**

**Functionality:**

The home page contains the nav bar with all the functionalities notes, currency converter, calculator, about us, contact us and log in. The body of the home page contains about us details, frequently asked questions, feedback page and contact us.

In notes, user can create a new notes and view previous notes, the user can edit and delete the previous notes.

**Technologies Used:**

Frontend: HTML, CSS, Bootstrap

Backend: Django

Views.py

from django.shortcuts import render

from django.contrib.auth.decorators import login\_required

@login\_required  
def home(request):  
 return render(request, 'home.html')

**Sign up Page:**

**Functionality:**

Renders sign up page

**Technologies Used:**

Frontend: HTML, CSS, Bootstrap

Backend: No specific backend functionality mentioned for this page.

Views.py

from django.shortcuts import render

from django.contrib.auth.forms import UserCreationForm

def user\_signup(request):  
 if request.method == 'POST':  
 form = UserCreationForm(request.POST)  
 if form.is\_valid():  
 form.save()  
 # Redirect to login page after successful signup  
 return redirect('login')  
 else:  
 form = UserCreationForm()  
 return render(request, 'signup.html', {'form': form})

**Login Page:**

**Functionality:**

Allows admin to open the home page using Django admin interface.

**Technologies Used:**

Frontend: HTML, CSS, JavaScript

Backend: Django, Django Admin

views.py

|  |
| --- |
| from django.contrib.auth import authenticate, login  def user\_login(request):  if request.method == 'POST':  username = request.POST.get('username')  password = request.POST.get('password')  user = authenticate(request, username=username, password=password)  if user is not None:  login(request, user)  return redirect('home') # Redirect to home page after successful login  else:  # Display error message using messages framework  messages.error(request, 'Invalid username or password. Please try again.')  return redirect('login') # Redirect back to login page  else:  return render(request, 'login.html') |

**Notes Page:**

**Functionality:**

Users can create a new notes and can save in the files.

Users can edit or delete this previous notes.

**Technologies Used:**

Frontend: HTML, CSS, JavaScript

Backend: Django, Django Forms, Django Models

**Model Fields:**

Views.py

from django.shortcuts import render, redirect, get\_object\_or\_404

from .forms import NoteForm

from .models import Notes

@login\_required  
def new\_notepad(request):  
 user\_notes = Notes.objects.filter(user=request.user)  
 return render(request, 'new\_notepad.html', {'notes': user\_notes})  
  
@login\_required  
def open\_notepad(request, pk):  
 note = Notes.objects.get(pk=pk)  
 return render(request, 'open\_notepad.html', {'note' : note})  
  
@login\_required  
def create\_note(request):  
 if request.method == 'POST':  
 form = NoteForm(request.POST)  
 if form.is\_valid():  
 note = form.save(commit=False)  
 note.user = request.user  
 form.save()  
 return redirect('new\_notepad')  
 else:  
 form = NoteForm()  
 return render(request, 'new\_note.html', {'form': form})  
  
@login\_required  
def edit\_note(request, pk):  
 note = get\_object\_or\_404(Notes, pk=pk)  
 if request.method == 'POST':  
 form = NoteForm(request.POST, instance=note)  
 if form.is\_valid():  
 form.save()  
 return redirect('open\_notepad', pk=pk)  
 else:  
 form = NoteForm(instance=note)  
 return render(request, 'edit\_note.html', {'form': form})  
  
  
@login\_required  
def delete\_note(request, pk):  
 note = get\_object\_or\_404(Notes, pk=pk)  
 if request.method == 'POST':  
 note.delete()  
 return redirect('new\_notepad')  
 return render(request, 'confirm\_delete\_note.html', {'note': note})

models.py

from django.db import models  
from django.contrib.auth.models import User

class Notes(models.Model):  
 title = models.CharField(max\_length=200)  
 text = models.TextField()  
 created = models.DateTimeField(auto\_now\_add=True)  
 user = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name='new\_notepad')

forms.py

from django import forms  
from .models import Notes  
  
class NoteForm(forms.ModelForm):  
 class Meta:  
 model = Notes  
 fields = ['title', 'text']  
 widgets = {  
 'title': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Enter title'}),  
 'text': forms.Textarea(attrs={'class': 'form-control', 'placeholder': 'Enter content', 'rows': 10})  
 }

**Currency Converter:**

**Functionality:**

The primary function is to convert the value of one currency into another.

The converter then displays the converted amount based on the current exchange rates.

These rates may update regularly to reflect current market conditions.

**Technologies Used:**

Frontend: HTML, CSS, JavaScript

Backend: Django, Currency Rate

Views.py

def currency\_converter(request):  
 response = requests.get('https://api.exchangerate-api.com/v4/latest/USD')  
 data = response.json()  
 currencies = data['rates']  
 if request.method == 'POST':  
 amount = float(request.POST['amount'])  
 from\_currency = request.POST['from\_currency']  
 to\_currency = request.POST['to\_currency']  
  
 # Make API request to get exchange rate  
 response = requests.get(f'https://api.exchangerate-api.com/v4/latest/{from\_currency}')  
 data = response.json()  
 exchange\_rate = data['rates'][to\_currency]  
  
 # Perform conversion  
 converted\_amount = round(amount \* exchange\_rate, 2)  
  
 context = {  
 'amount': amount,  
 'from\_currency': from\_currency,  
 'to\_currency': to\_currency,  
 'converted\_amount': converted\_amount,  
 'exchange\_rate': exchange\_rate,  
 'currencies': currencies  
 }  
 return render(request, 'result.html', context)  
 else:  
 return render(request, 'converter.html', {'currencies': currencies})

**Calculator:**

**Functionality:**

User can use calculator for addition, subtraction, multiplication, division and modules division

**Technologies Used:**

Frontend: HTML, CSS, Bootstrap

Backend: Django

Views.py

def calculator\_view(request):  
 return render(request, 'calculator.html')

**Feedback Page:**

**Functionality:**

The user can give there feedback in this page as good or bad or to improve the UI or more functionalities which can be checked by the admin in the admin page.

**Technologies Used:**

Frontend: HTML, CSS, Bootstrap

Backend: Django, Django Forms, Django Models

Views.py

|  |
| --- |
| from django.shortcuts import render    # Create your views here.  def feedback(request):  if request.method == 'POST':  form = FeedbackForm(request.POST)  if form.is\_valid():  form.save()  return redirect('feedback\_success')  else:  form = FeedbackForm()  return render(request, 'feedback.html', {'form': form}) |

Models.py

from django.db import models  
from django.contrib.auth.models import User

class Feedback(models.Model):  
 name = models.CharField(max\_length=100)  
 email = models.EmailField()  
 message = models.TextField()  
 created\_at = models.DateTimeField(auto\_now\_add=True)  
  
 def \_\_str\_\_(self):  
 return self.name

forms.py

from django import forms  
from .models import Feedback

class FeedbackForm(forms.ModelForm):  
 class Meta:  
 model = Feedback  
 fields = ['name', 'email', 'message']

1. **Challenges Faced**

1.Integration currency converter for different countries using currency exchange rate APIs and forex python converter.

2.Customization of admin interface using Django Rosette while ensuring compatibility with internationalization features.

1. **Future Enhancements**

1.Integration of more currency exchange rates APIs for enhanced accuracy.

2. Implementation of user authentication, enabling users to create accounts, log in, and access the utility tools.

3.Implementation tasks such as file management, text manipulation.

4.Implementation of user friendly calculator.

1. **Conclusion**

The Utility project successfully delivers a user-friendly and used to write the notes, can get currency converter and calculator using Django framework. Its modular design, integration of various technologies, and focus on user experience make it a valuable tool for multiple functionalities.

1. **References**

1.Django Documentation: [https://docs.djangoproject.com/en/5.0/]

2.Bootstrap Documentation: [https://getbootstrap.com/docs/4.1/gettingstarted/introduction/]

3.Currency exchange rate API link: [https://api.exchangerate-api.com/v4/latest/USD]