

ChatGPT Prompts:

# My Prompt:

Now I am thinking a simple approach for OTP when user click otp login it will send a random otp with my mail after I gave that number in token from, I will allow to login

# ChatGPT Answer:

Sending a random OTP to the user's email for OTP-based login is a valid and simple approach. Here's a step-by-step guide on how to implement this approach in your Django project:

\*\*Step 1: Set Up Email Configuration\*\*

First, make sure you have configured email settings in your Django project's settings file (settings.py). You can use Django's built-in email configuration or third-party packages like `sendgrid`, `smtp`, etc. For example, you can configure Gmail for sending emails:

```python

# settings.py

EMAIL\_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'

EMAIL\_HOST = 'smtp.gmail.com'

EMAIL\_PORT = 587

EMAIL\_USE\_TLS = True

EMAIL\_HOST\_USER = 'your\_email@gmail.com'

EMAIL\_HOST\_PASSWORD = 'your\_email\_password'

```

\*\*Step 2: Generate and Send OTP via Email\*\*

In your OTP login view, you can generate a random OTP, send it to the user's email, and store it in the user's session for later verification:

```python

from django.contrib.auth.models import User

from django.core.mail import send\_mail

from django.shortcuts import render, redirect, HttpResponse

import random

from django.conf import settings

def otp\_login(request):

if request.method == 'POST':

# Generate a random OTP (6-digit)

otp = ''.join([str(random.randint(0, 9)) for \_ in range(6)])

# Get the user's email (you may use a form field to collect it)

email = request.POST.get('email')

# Send the OTP to the user's email

subject = 'Your OTP for Login'

message = f'Your OTP for login: {otp}'

from\_email = settings.EMAIL\_HOST\_USER

recipient\_list = [email]

try:

send\_mail(subject, message, from\_email, recipient\_list)

except Exception as e:

return HttpResponse("Error sending OTP email")

# Store the OTP in the user's session

request.session['otp'] = otp

# Redirect to OTP verification page

return redirect('otp\_verify')

# Render the OTP login form

return render(request, 'otp\_login.html')

```

In this example, we generate a 6-digit OTP, send it via email, and store it in the user's session as 'otp'. Replace `'otp\_verify'` with the URL of the OTP verification page.

\*\*Step 3: Create OTP Verification View\*\*

Create a view for OTP verification where the user enters the OTP received in their email and compare it with the OTP stored in their session:

```python

def otp\_verify(request):

if request.method == 'POST':

entered\_otp = request.POST.get('otp')

stored\_otp = request.session.get('otp')

if entered\_otp == stored\_otp:

# OTP is correct, log in the user or perform any desired actions

# ...

return redirect('index') # Replace with your desired redirect URL

else:

return HttpResponse("Invalid OTP. Please try again.")

# Render the OTP verification form

return render(request, 'otp\_verify.html')

```

\*\*Step 4: Create OTP Login Templates\*\*

Create templates for OTP login (`otp\_login.html`) and OTP verification (`otp\_verify.html`) forms.

\*\*Step 5: Update URLs\*\*

Update your project's URLs to include the OTP login and verification views.

With these steps, your users can request an OTP, receive it via email, and then use it to log in. Make sure to handle the user authentication and redirection after a successful OTP verification appropriately in your project.

## My Prompt:

give me OTP login and verification # step4

# ChatGPT:

Certainly! Below are example templates for OTP login (`otp\_login.html`) and OTP verification (`otp\_verify.html`) forms:

\*\*otp\_login.html (OTP Login Form):\*\*

```html

{% extends 'master.html' %}

{% block body %}

<section id="form">

<div class="container">

<div class="row">

<div class="signup-form">

<h2>Login with OTP</h2>

<form method="post">

{% csrf\_token %}

<label for="email">Enter Your Email:</label>

<input type="email" name="email" id="email" required>

<button class="btn btn-default" type="submit">Request OTP</button>

</form>

</div>

</div>

</div>

</section>

{% endblock %}

```

\*\*otp\_verify.html (OTP Verification Form):\*\*

```html

{% extends 'master.html' %}

{% block body %}

<section id="form">

<div class="container">

<div class="row">

<div class="signup-form">

<h2>Verify OTP</h2>

<form method="post">

{% csrf\_token %}

<label for="otp">Enter OTP:</label>

<input type="text" name="otp" id="otp" required>

<button class="btn btn-default" type="submit">Verify OTP</button>

</form>

</div>

</div>

</div>

</section>

{% endblock %}

```

These templates create simple forms where users can enter their email to request an OTP and enter the OTP to verify their login. You can further customize the design and styles to match your project's theme.

Make sure these templates are placed in the appropriate location within your project's template directory so that Django can find and render them correctly when requested.

My Prompt:

give the URL update

ChatGPT Answer:

To implement OTP login and verification, you'll need to update your URL configuration. Below are the URL patterns for OTP login and verification:

In your Django project's `urls.py`, add the following URL patterns:

```python

from django.urls import path

from . import views

urlpatterns = [

# ... Your existing URL patterns ...

# Add OTP login and verification URLs

path('otp-login/', views.otp\_login, name='otp\_login'),

path('otp-verify/', views.otp\_verify, name='otp\_verify'),

]

```

In this configuration, when users visit `/otp-login/`, they will see the OTP login form (`otp\_login.html`), and when they submit the form, it will send them to `/otp-verify/` for OTP verification (`otp\_verify.html`).

Now, you need to create views for `otp\_login` and `otp\_verify`. In your `views.py`, you can create these views as follows:

```python

from django.shortcuts import render, redirect

from django.contrib.auth.models import User

from django\_otp.plugins.otp\_totp.models import TOTPDevice

# Import any other necessary modules

# OTP login view

def otp\_login(request):

if request.method == 'POST':

email = request.POST.get('email')

user = User.objects.filter(email=email).first()

if user:

# Generate and send OTP to the user's email

otp\_device, created = TOTPDevice.objects.get\_or\_create(user=user, confirmed=True)

otp\_device.save()

otp\_device.throttle\_factor = 1 # You can adjust the throttling factor

otp\_device.save()

otp\_device.generate\_challenge()

otp\_device.save()

# Send the OTP to the user's email (you need to implement this)

# Example: send\_otp\_to\_email(otp\_device.token)

# Redirect to OTP verification page

return redirect('otp\_verify')

return render(request, 'otp\_login.html')

# OTP verification view

def otp\_verify(request):

if request.method == 'POST':

otp = request.POST.get('otp')

user = request.user

otp\_device = TOTPDevice.objects.filter(user=user).first()

if otp\_device and otp\_device.verify\_token(otp):

otp\_device.confirmed = True

otp\_device.save()

return redirect('index') # Replace with the desired redirect URL after successful login

return render(request, 'otp\_verify.html')

```

In these views, the `otp\_login` view handles generating and sending the OTP to the user's email (you need to implement the email sending part) and then redirects the user to the `otp\_verify` view. The `otp\_verify` view checks if the entered OTP is valid and, if so, logs the user in and redirects them to the index page.

Remember to replace `'otp\_verify'` and `'index'` with your actual URL names for the OTP verification page and the landing page after successful login.

Ensure that your `otp\_login.html` and `otp\_verify.html` templates are correctly set up with the input fields for email and OTP, as shown in the previous response.

My Prompt

Migrate UserCreation Form

from app.models import UserCreateFrom I am using this

ChatGPT

I see that you are trying to import `UserCreateFrom` from your models, but `UserCreateFrom` appears to be a form class, not a model. In Django, forms are typically placed in a separate forms.py file within your app.

Make sure that your `UserCreateFrom` form is defined in a file like `forms.py` within your app directory, and then import it in your `signup` view like this:

```python

from django.shortcuts import render, redirect

from django.contrib.auth import authenticate, login

from app.forms import UserCreateFrom # Import your UserCreateForm from the forms.py file

def signup(request):

if request.method == 'POST':

form = UserCreateFrom(request.POST)

if form.is\_valid():

new\_user = form.save()

new\_user = authenticate(

username=form.cleaned\_data['username'],

password=form.cleaned\_data['password1']

)

login(request, new\_user)

return redirect('index')

else:

form = UserCreateFrom()

context = {

'form': form,

}

return render(request, 'registration/signup.html', context)

```

Ensure that your `UserCreateFrom` form is correctly defined in the `forms.py` file within your app's directory, and that should resolve the issue

My Prompt:

{% extends 'master.html' %}

{% block body %}

<section id="form">

<!--form--> <div class="container"> <div class="row"> <div class="signup-form"><!--login form--> {% if user.is\_authenticated %} <!-- Redirect to OTP login since the user is authenticated --> <script> window.location.href = "{% url 'otp\_login' %}"; </script> {% else %} <h2>Login account</h2> <form method="post"> {% csrf\_token %} {% for field in form %} {{field.errors}} <label>{{field.label\_tag}}</label> <p>{{field}}</p> {% endfor %} <a href="{% url 'password\_reset' %}" style="float:right">Forgot password ? </a><br> <button class="btn-btn-default" type="submit">Login</button> <a href="{% url 'signup' %}"><span style="color:black;margin-left:500px">Create Your Account</span></a> </form> {% endif %} </div> </div> </div> </section> {% endblock %}

Make UI improvement make Login form under a Boxy Shadow. # A code

## ChatGPT Answer:

To add a boxy shadow to your login form, you can use CSS to style it. Here's an example of how you can modify your template to include a boxy shadow:

```html

{% extends 'master.html' %}

{% block body %}

<style>

/\* Add CSS for the boxy shadow \*/

.signup-form {

box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);

padding: 20px;

background-color: #fff;

border-radius: 5px;

}

</style>

<section id="form"><!--form-->

<div class="container">

<div class="row">

<div class="signup-form"><!--login form-->

{% if user.is\_authenticated %}

<!-- Redirect to OTP login since the user is authenticated -->

<script>

window.location.href = "{% url 'otp\_login' %}";

</script>

{% else %}

<h2>Login account</h2>

<form method="post">

{% csrf\_token %}

{% for field in form %}

{{field.errors}}

<label>{{field.label\_tag}}</label>

<p>{{field}}</p>

{% endfor %}

<a href="{% url 'password\_reset' %}" style="float:right">Forgot password ? </a><br>

<button class="btn-btn-default" type="submit">Login</button>

<a href="{% url 'signup' %}"><span style="color:black;margin-left:500px">Create Your Account</span></a>

</form>

{% endif %}

</div>

</div>

</div>

</section>

{% endblock %}

```

In this example, I added a `style` block within your `body` block and applied CSS to the `.signup-form` class. This CSS sets a box shadow, padding, background color, and border radius to create a boxy shadow effect around the login form. You can adjust the CSS properties as needed to achieve the desired visual effect.

My Prompt :

<!DOCTYPE html> <html lang="en"> <head> {% load static %} <meta charset="utf-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <meta name="description" content=""> <meta name="author" content=""> <title>Home | E-Shopper</title> <link href="/static/css/bootstrap.min.css" rel="stylesheet"> <link href="/static/css/font-awesome.min.css" rel="stylesheet"> <link href="/static/css/prettyPhoto.css" rel="stylesheet"> <link href="/static/css/price-range.css" rel="stylesheet"> <link href="/static/css/animate.css" rel="stylesheet"> <link href="/static/css/main.css" rel="stylesheet"> <link href="/static/css/responsive.css" rel="stylesheet"> <!--[if lt IE 9]> <script src="/static/js/html5shiv.js"></script> <script src="/static/js/respond.min.js"></script> <![endif]--> <link rel="shortcut icon" href="/static/images/ico/favicon.ico"> <link rel="apple-touch-icon-precomposed" sizes="144x144" href="/static/images/ico/apple-touch-icon-144-precomposed.png"> <link rel="apple-touch-icon-precomposed" sizes="114x114" href="/static/images/ico/apple-touch-icon-114-precomposed.png"> <link rel="apple-touch-icon-precomposed" sizes="72x72" href="/static/images/ico/apple-touch-icon-72-precomposed.png"> <link rel="apple-touch-icon-precomposed" href="/static/images/ico/apple-touch-icon-57-precomposed.png"> </head><!--/head--> <body> <header id="header"><!--header--> <div class="header\_top"><!--header\_top--> <div class="container"> <div class="row"> <div class="col-sm-6"> <div class="contactinfo"> <ul class="nav nav-tabs"> <li><a href="#"><i class="fa fa-phone"></i> 7908223750</a></li> <li><a href="#"><i class="fa fa-envelope"></i> [malakartanmoy97@gmail.com</a></li](mailto:malakartanmoy97@gmail.com</a></li)> </ul> </div> </div> <div class="col-sm-6"> <div class="social-icons pull-right"> <ul class="nav navbar-nav"> <li><a href="https://www.facebook.com/tanmoy.malakar.96/"><i class="fa fa-facebook"></i></a></li> <li><a href="https://www.instagram.com/tanmoy.malakar.t/"><i class="fa fa-Instagram"></i></a></li> </ul> </div> </div> </div> </div> </div><!--/header\_top--> <div class="header-middle"><!--header-middle--> <div class="container"> <div class="row"> <div class="col-sm-4"> <div class="logo pull-left"> <a href="{% url 'index' %ƒ}"><img src="/static/images/home/logo.png" alt="" /></a> </div> </div> </div> <div class="col-sm-8"> <div class="shop-menu pull-right"> <ul class="nav navbar-nav"> <li><a href="/order/"><i class="fa fa-crosshairs"></i> Order </a></li> <li><a href="/cart/cart-detail/"><i class="fa fa-shopping-cart"></i> Cart <span class="badge badge-secondary">{{request.session.cart|length}}</span></a></li> <li><a href="{% url 'password\_change' %}"><i class="fa fa-lock"></i> Change Password</a></li> {% if user.is\_authenticated %} <li><a href="{% url 'logout' %}"><i class="fa fa-lock"></i> Logout</a></li> {% else %} <li><a href="{% url 'otp\_login' %}"><i class="fa fa-lock"></i> Login </a></li> {% endif %} </ul> </div> </div> </div> </div> </div><!--/header-middle--> <div class="header-bottom"><!--header-bottom--> <div class="container"> <div class="row"> <div class="col-sm-9"> <div class="navbar-header"> <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse"> <span class="sr-only">Toggle navigation</span> <span class="icon-bar"></span> <span class="icon-bar"></span> <span class="icon-bar"></span> </button> </div> <div class="mainmenu pull-left"> <ul class="nav navbar-nav collapse navbar-collapse"> <li><a href="{% url 'index' %}" class="active">Home</a></li> <li><a href="/contact/">Contact</a></li> </ul> </div> </div> </div> </div> </div><!--/header-bottom--> </header><!--/header--> {% block body %}{% endblock%} <footer id="footer"><!--Footer--> <div class="footer-top"> <div class="container"> <div class="row"> <div class="col-sm-2"> <div class="companyinfo"> <h2><span>e</span>-shopper</h2> <p>Online Shopping System Powered By Django and GenAI</p> </div> </div> <div class="col-sm-7"> <div class="col-sm-3"> <div class="video-gallery text-center"> <a href="#"> <div class="iframe-img"> <img src="/static/images/home/iframe1.png" alt="" /> </div> <div class="overlay-icon"> <i class="fa fa-play-circle-o"></i> </div> </a> <p>Circle of Hands</p> <h2>24 DEC 2014</h2> </div> </div> <div class="col-sm-3"> <div class="video-gallery text-center"> <a href="#"> <div class="iframe-img"> <img src="/static/images/home/iframe2.png" alt="" /> </div> <div class="overlay-icon"> <i class="fa fa-play-circle-o"></i> </div> </a> <p>Circle of Hands</p> <h2>24 DEC 2014</h2> </div> </div> <div class="col-sm-3"> <div class="video-gallery text-center"> <a href="#"> <div class="iframe-img"> <img src="/static/images/home/iframe3.png" alt="" /> </div> <div class="overlay-icon"> <i class="fa fa-play-circle-o"></i> </div> </a> <p>Circle of Hands</p> <h2>24 DEC 2014</h2> </div> </div> <div class="col-sm-3"> <div class="video-gallery text-center"> <a href="#"> <div class="iframe-img"> <img src="/static/images/home/iframe4.png" alt="" /> </div> <div class="overlay-icon"> <i class="fa fa-play-circle-o"></i> </div> </a> <p>Circle of Hands</p> <h2>24 DEC 2014</h2> </div> </div> </div> <div class="col-sm-3"> <div class="address"> <img src="/static/images/home/map.png" alt="" /> <p>505 S Atlantic Ave Virginia Beach, VA(Virginia)</p> </div> </div> </div> </div> </div> <div class="footer-widget"> <div class="container"> <div class="row"> <div class="col-sm-2"> <div class="single-widget"> <h2>Service</h2> <ul class="nav nav-tabs nav-stacked"> <li><a href="#">Contact Us</a></li> <li><a href="#">FAQ’s</a></li> </ul> </div> </div> div> <div class="col-sm-2"> <div class="single-widget"> <h2>About Shopper</h2> <ul class="nav nav-pills nav-stacked"> <li><a href="#">Company Information</a></li> <li><a href="#">Careers</a></li> <li><a href="#">Store Location</a></li> <li><a href="#">Affiliate Program</a></li> <li><a href="#">Copyright</a></li> </ul> </div> </div> <div class="col-sm-3 col-sm-offset-1"> <div class="single-widget"> <h2>About Shopper</h2> <form action="#" class="searchform"> <input type="text" placeholder="Your email address" /> <button type="submit" class="btn btn-default"><i class="fa fa-arrow-circle-o-right"></i></button> <p>Get the most recent updates from <br />our site and be updated yourself...</p> </form> </div> </div> </div> </div> </div> <div class="footer-bottom"> <div class="container"> <div class="row"> <p class="pull-left">Copyright © 2013 E-SHOPPER Inc. All rights reserved.</p> </div> </div> </div> </footer><!--/Footer--> <script src="/static/js/jquery.js"></script> <script src="/static/js/bootstrap.min.js"></script> <script src="/static/js/jquery.scrollUp.min.js"></script> <script src="/static/js/price-range.js"></script> <script src="/static/js/jquery.prettyPhoto.js"></script> <script src="/static/js/main.js"></script> </body> </html> #A code

Add light and dark theme button on top navbar and genarate code that will coddectly work and we should display content properly   
ChatGPT Answer:

To add light and dark theme buttons to your top navbar and implement functionality to switch between themes, you can use JavaScript and CSS. Here's an example of how you can do this:

1. Modify your HTML template to include light and dark theme buttons:

```html

<!-- Add light and dark theme buttons to the top navbar -->

<div class="navbar-header">

<button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

</div>

<div class="mainmenu pull-left">

<ul class="nav navbar-nav collapse navbar-collapse">

<li><a href="{% url 'index' %}" class="active">Home</a></li>

<li><a href="/contact/">Contact</a></li>

</ul>

</div>

<!-- Add light and dark theme buttons -->

<div class="theme-buttons pull-right">

<button id="light-theme-button" class="theme-button" onclick="setLightTheme()">Light</button>

<button id="dark-theme-button" class="theme-button" onclick="setDarkTheme()">Dark</button>

</div>

```

2. Create CSS classes for your light and dark themes. You can define these classes in your existing CSS files or create separate CSS files for each theme (e.g., `light-theme.css` and `dark-theme.css`).

3. Add JavaScript functions to switch between themes. Here's an example of how you can do it:

```html

<script>

// Function to set the light theme

function setLightTheme() {

document.body.classList.remove("dark-theme");

document.body.classList.add("light-theme");

}

// Function to set the dark theme

function setDarkTheme() {

document.body.classList.remove("light-theme");

document.body.classList.add("dark-theme");

}

// Check the user's preferred theme from local storage (optional)

if (localStorage.getItem("theme") === "dark") {

setDarkTheme();

} else {

setLightTheme();

}

</script>

```

4. Add CSS rules for your light and dark themes in your CSS files. For example, you can create a `light-theme.css` and a `dark-theme.css` file:

```css

/\* light-theme.css \*/

.light-theme {

/\* Define styles for the light theme here \*/

background-color: #ffffff;

color: #000000;

/\* Add more styles as needed \*/

}

/\* dark-theme.css \*/

.dark-theme {

/\* Define styles for the dark theme here \*/

background-color: #000000;

color: #ffffff;

/\* Add more styles as needed \*/

}

```

5. Make sure to link the light and dark theme CSS files in your HTML template based on the selected theme. You can do this dynamically using JavaScript.

That's it! Now, users can switch between light and dark themes using the buttons, and your content will be displayed according to the selected theme. Additionally, you can store the user's theme preference in local storage to persist it across page reloads if needed.

My Prompt:

Now Make also something that all fonts and other text easily visible on dark mode

ChatGPT:

In dark mode, it's important to ensure that text and other elements remain easily visible. Here are some tips and CSS adjustments you can make to improve visibility in dark mode:

1. \*\*Contrast\*\*: Increase the contrast between text and background colors. In dark mode, you'll typically have light text on a dark background, so make sure the text color is significantly lighter than the background color.

2. \*\*Font Size\*\*: Consider increasing the font size slightly for better readability in low-light conditions.

3. \*\*Font Weight\*\*: You can use a slightly heavier font weight (e.g., `font-weight: bold;`) for text elements to make them stand out.

4. \*\*Text Shadow\*\*: Add a subtle text shadow to your text to make it pop against the dark background. For example:

```css

text-shadow: 1px 1px 1px rgba(255, 255, 255, 0.5);

```

Adjust the values as needed to achieve the desired effect.

5. \*\*Link Styles\*\*: Ensure that links are clearly distinguishable from regular text. You can change the color, underline, or add a background color to links on hover or active states.

6. \*\*Button Styles\*\*: Apply similar principles to buttons. Make sure button text stands out against the button background. You can adjust background color, text color, and border styles accordingly.

Here's an example of how you can adjust your CSS for better visibility in dark mode:

```css

/\* dark-theme.css \*/

.dark-theme {

/\* Dark background color \*/

background-color: #000000;

color: #ffffff; /\* Light text color \*/

/\* Increase font size slightly \*/

font-size: 16px;

/\* Apply a subtle text shadow \*/

text-shadow: 1px 1px 1px rgba(255, 255, 255, 0.5);

/\* Style links \*/

a {

color: #bb86fc; /\* Custom link color in dark mode \*/

text-decoration: underline; /\* Underline links \*/

}

/\* Style buttons \*/

.btn {

background-color: #6200ea; /\* Custom button background color in dark mode \*/

color: #ffffff; /\* Custom button text color in dark mode \*/

border: none;

}

}

```

These are just some basic adjustments to get you started. You can further fine-tune the styles to match your design preferences and ensure optimal visibility in dark mode. Testing your dark mode design across different devices and screen sizes is also important to ensure a consistent and readable experience for users.

My Prompt:

now Make some modification that text clearly visible and when using dark theme use a curvy border on button hyerlink on navbars so distinguisable

ChatGpt Answer:

To make text clearly visible and add curved borders to buttons and hyperlinks in the navbar for better distinction in dark mode, you can modify your CSS as follows:

\*\*CSS Changes:\*\*

1. Ensure text readability in both light and dark themes by adjusting the text color:

```css

/\* Light Theme Styles \*/

body.light-theme {

background-color: #fff;

color: #333;

}

/\* Dark Theme Styles \*/

body.dark-theme {

background-color: #333;

color: #fff;

}

```

2. Apply curved borders to buttons and hyperlinks in the navbar for better distinction:

```css

/\* Curved border for buttons and hyperlinks \*/

body.light-theme .btn,

body.light-theme .navbar-nav li a,

body.dark-theme .btn,

body.dark-theme .navbar-nav li a {

border-radius: 15px; /\* Adjust the radius as needed \*/

}

```

3. Modify the button styles to have a more pronounced appearance in dark mode:

```css

/\* Button styles for dark mode \*/

body.dark-theme .btn {

background-color: #ff6600;

color: #fff;

padding: 10px 20px;

border: none;

border-radius: 15px; /\* Adjust the radius as needed \*/

font-weight: bold;

transition: background-color 0.3s ease;

}

body.dark-theme .btn:hover {

background-color: #ff8844; /\* Change the hover color as needed \*/

}

```

\*\*JavaScript Changes:\*\*

No changes are required in the JavaScript code, as we've focused on CSS modifications for styling.

With these CSS modifications, you'll have improved text visibility and distinct curved borders for buttons and hyperlinks in both light and dark themes. You can further adjust colors, border radii, and other styles to match your design preferences.

My Prompt

still text not visible:

ChatGPT

I apologize for any confusion. If the text is not clearly visible in dark mode, you can further adjust the text color and background color in your CSS for the dark theme. Here's an updated example:

```css

/\* Light Theme Styles \*/

body.light-theme {

background-color: #fff;

color: #333;

}

/\* Dark Theme Styles \*/

body.dark-theme {

background-color: #333;

color: #fff;

}

/\* Curved border for buttons and hyperlinks \*/

body.light-theme .btn,

body.light-theme .navbar-nav li a,

body.dark-theme .btn,

body.dark-theme .navbar-nav li a {

border-radius: 15px; /\* Adjust the radius as needed \*/

}

/\* Button styles for dark mode \*/

body.dark-theme .btn {

background-color: #ff6600;

color: #fff;

padding: 10px 20px;

border: none;

border-radius: 15px; /\* Adjust the radius as needed \*/

font-weight: bold;

transition: background-color 0.3s ease;

}

body.dark-theme .btn:hover {

background-color: #ff8844; /\* Change the hover color as needed \*/

}

/\* Adjust text color for dark mode \*/

body.dark-theme a,

body.dark-theme p,

body.dark-theme h1,

body.dark-theme h2,

body.dark-theme h3,

body.dark-theme h4 {

color: #fff; /\* Adjust text color for readability \*/

}

```

In the updated CSS, I've added a section to adjust the text color for various HTML elements (links, paragraphs, headings) in dark mode. You can customize the `color` property values to ensure text readability in the dark theme.

My Prompt:

Make same changes in footer also that when click toggle it also change

ChatGpt:

To create a single button that toggles between dark and light themes and applies the theme to the entire page, including the header and footer, you can follow these steps:

1. Add the Toggle Button to Your HTML:

```html

<!-- Add this button wherever you want it on your page -->

<button id="theme-toggle-button" class="btn btn-default">Toggle Theme</button>

```

2. Include JavaScript for Theme Toggling (combine both header and footer theme toggling logic into a single function):

```html

<script>

// Function to toggle the theme when the button is clicked

function toggleTheme() {

if (document.body.classList.contains('dark-theme')) {

document.body.classList.remove('dark-theme');

localStorage.setItem('theme', 'light');

} else {

document.body.classList.add('dark-theme');

localStorage.setItem('theme', 'dark');

}

}

// Add an event listener to the toggle button

const themeToggleButton = document.getElementById('theme-toggle-button');

if (themeToggleButton) {

themeToggleButton.addEventListener('click', toggleTheme);

}

// Function to set the theme based on user preference in localStorage

function setTheme() {

const savedTheme = localStorage.getItem('theme');

if (savedTheme === 'dark') {

document.body.classList.add('dark-theme');

} else {

document.body.classList.remove('dark-theme');

}

}

// Call setTheme to set the initial theme when the page loads

setTheme();

</script>

```

3. Define CSS Styles for Dark and Light Themes (no changes needed in the styles, as they remain the same):

```css

/\* Light Theme Styles \*/

body.light-theme {

background-color: #fff;

color: #333;

}

/\* Dark Theme Styles \*/

body.dark-theme {

background-color: #333;

color: #fff;

}

/\* Curved border for buttons and hyperlinks \*/

body.light-theme .btn,

body.light-theme .navbar-nav li a,

body.dark-theme .btn,

body.dark-theme .navbar-nav li a {

border-radius: 15px; /\* Adjust the radius as needed \*/

}

/\* Button styles for dark mode \*/

body.dark-theme .btn {

background-color: #ff6600;

color: #fff;

padding: 10px 20px;

border: none;

border-radius: 15px; /\* Adjust the radius as needed \*/

font-weight: bold;

transition: background-color 0.3s ease;

}

body.dark-theme .btn:hover {

background-color: #ff8844; /\* Change the hover color as needed \*/

}

/\* Adjust text color for dark mode \*/

body.dark-theme a,

body.dark-theme p,

body.dark-theme h1,

body.dark-theme h2,

body.dark-theme h3,

body.dark-theme h4 {

color: #fff; /\* Adjust text color for readability \*/

}

```

With these changes, you will have a single "Toggle Theme" button that toggles between dark and light themes for the entire page, including the header and footer. Clicking the button will change the theme without causing a page refresh or redirect.

My Prompt

it is not working when click:

ChatGPT:

<script>

// Function to toggle the theme when the button is clicked

function toggleTheme() {

const savedTheme = localStorage.getItem('theme');

if (savedTheme === 'dark') {

document.body.classList.remove('dark-theme');

localStorage.setItem('theme', 'light');

} else {

document.body.classList.add('dark-theme');

localStorage.setItem('theme', 'dark');

}

}

// Function to set the theme based on user preference in localStorage

function setTheme() {

const savedTheme = localStorage.getItem('theme');

if (savedTheme === 'dark') {

document.body.classList.add('dark-theme');

} else {

document.body.classList.remove('dark-theme');

}

}

// Add an event listener to the toggle button

const themeToggleButton = document.getElementById('theme-toggle-button');

if (themeToggleButton) {

themeToggleButton.addEventListener('click', toggleTheme);

}

// Call setTheme to set the initial theme when the page loads

setTheme();

</script>

My Prompt

make footer section more readable:

ChatGpt:

/\* Footer Styles for Light Theme \*/

body.light-theme #footer {

background-color: #f5f5f5;

color: #333;

}

/\* Footer Styles for Dark Theme \*/

body.dark-theme #footer {

background-color: #222;

color: #fff;

}

/\* Adjust text color for footer links in both themes \*/

body #footer a {

color: inherit;

}

/\* Add padding and margin to the footer content \*/

body #footer .container {

padding: 20px 0;

}

/\* Style for the companyinfo section \*/

body #footer .companyinfo h2 {

color: #333; /\* Adjust text color for better readability \*/

}

/\* Style for the address section \*/

body #footer .address p {

color: #777; /\* Adjust text color for better readability \*/

}

My Promt:

In dark mode give boxy shadow 3d eddect to all images

ChatGpt:

To give a 3D box shadow effect to all images in dark mode, you can add a CSS style to target images specifically in dark mode. Here's how you can do it:

```css

/\* Box shadow effect for images in dark mode \*/

body.dark-theme img {

box-shadow: 5px 5px 15px rgba(0, 0, 0, 0.5); /\* Adjust shadow values as needed \*/

transition: box-shadow 0.3s ease;

}

/\* Hover effect for images in dark mode \*/

body.dark-theme img:hover {

box-shadow: 5px 5px 20px rgba(0, 0, 0, 0.7); /\* Adjust hover shadow values as needed \*/

}

```

This CSS code will add a 3D box shadow effect to all images when in dark mode, and the shadow will become slightly larger when you hover over the images. Adjust the `box-shadow` values to fine-tune the shadow effect according to your design preferences.

# My Prompt:

fix the code that visible checkout button

ChatGpt:

To make the "Checkout" button visible in your Django template, you need to place it outside of the for loop that iterates over the cart items. Here's the corrected code:

```html

{% extends 'master.html' %}

{% load cart\_tag %}

{% block body %}

<section id="cart\_items">

<div class="container">

<div class="breadcrumbs">

<ol class="breadcrumb">

<li><a href="{% url 'index' %}">Home</a></li>

<li class="active">Shopping Cart</li>

</ol>

</div>

<div class="table-responsive cart\_info">

<table class="table table-condensed">

<thead>

<tr class="cart\_menu">

<td class="image">Item</td>

<td class="description"></td>

<td class="price">Price</td>

<td class="quantity">Quantity</td>

<td class="total">Total</td>

<td></td>

</tr>

</thead>

<tbody>

{% for key, value in request.session.cart.items %}

<tr>

<td class="cart\_product">

<a href=""><img src="{{ value.image }}" alt="" height="300"></a>

</td>

<td class="cart\_description">

<h4><a href="">{{ value.name }}</a></h4>

<p>Web ID: 1089772</p>

</td>

<td class="cart\_price">

<p>₹ {{ value.price }}</p>

</td>

<td class="cart\_quantity">

<div class="cart\_quantity\_button">

<a class="cart\_quantity\_up"

href="{% url 'item\_increment' value.product\_id %}"> + </a>

<input class="cart\_quantity\_input" type="text" name="quantity"

value="{{ value.quantity }}" autocomplete="off" size="2">

<a class="cart\_quantity\_down"

href="{% url 'item\_decrement' value.product\_id %}"> - </a>

</div>

</td>

<td class="cart\_total">

<p class="cart\_total\_price">₹ {{ value.price|multiply:value.quantity }}</p>

</td>

<td class="cart\_delete">

<a class="cart\_quantity\_delete"

href="{% url 'item\_clear' key %}"><i class="fa fa-times"></i></a>

</td>

</tr>

{% endfor %}

</tbody>

<tfoot>

<tr>

<td colspan="4"></td>

<td>

<h3>Sub Total :</h3>

</td>

<td>

<h3>

{{ cart\_total\_amount }}

</h3>

</td>

</tr>

</tfoot>

</table>

</div>

</div>

</section> <!--/#cart\_items-->

<br>

<div class="container">

<div class="row">

<div class="col-md-6">

<a href="{% url 'cart\_clear' %}" class="btn btn-default update">Clear Cart</a>

</div>

<div class="col-md-6">

<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal">

Checkout

</button>

</div>

</div>

</div>

<div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel"

aria-hidden="true">

<!-- The rest of your modal code here -->

</div>

{% endblock %}

``

I've placed the "Checkout" button just below the "Clear Cart" button outside the loop, ensuring it's visible without duplication.

# My Prompt:

Button displayed but not visible the checkout form fix it

ChatGPT

If the checkout form is not visible when you click the "Checkout" button, it's likely due to an issue with the modal implementation. Ensure that you have the necessary Bootstrap modal components in place. Here's the corrected template code with the modal components properly structured:

```html

<!-- ... Previous code ... -->

<div class="container">

<div class="row">

<div class="col-md-6">

<a href="{% url 'cart\_clear' %}" class="btn btn-default update">Clear Cart</a>

</div>

<div class="col-md-6">

<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal">

Checkout

</button>

</div>

</div>

</div>

<!-- Modal -->

<div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel"

aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalLabel">Checkout</h5>

<button type="button" class="close" data-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div class="modal-body">

<!-- Checkout form goes here -->

<form method="post" action="/checkout/">

{% csrf\_token %}

<div class="form-group">

<label for="recipient-name" class="col-form-label">Phone:</label>

<input type="text" class="form-control" id="recipient-name" name="phone">

</div>

<div class="form-group">

<label for="message-text" class="col-form-label">Address:</label>

<textarea class="form-control" id="message-text" name="address"></textarea>

</div>

<div class="form-group">

<label for="message-text" class="col-form-label">Pincode:</label>

<textarea class="form-control" name="pincode"></textarea>

</div>

</form>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>

<button type="submit" class="btn btn-primary">Checkout</button>

</div>

</div>

</div>

</div>

{% endblock %}

```

Make sure that you have the necessary Bootstrap CSS and JavaScript included in your project to enable modal functionality. Also, double-check that your form fields and modal content are correctly structured within the modal body.

My Prompt:

In Dark Mode lebel not visible

ChatGPT

In dark mode or when using a dark background, you might need to adjust the label text color to make it visible. You can do this by applying custom CSS styles to your label elements. Here's an example of how you can modify your label styles to ensure they are visible in dark mode:

```html

<!-- ... Previous code ... -->

<style>

/\* Custom styles for label text in dark mode \*/

.dark-mode label {

color: white; /\* Set the label text color to white or a suitable contrasting color \*/

}

</style>

<!-- Modal -->

<div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel"

aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalLabel">Checkout</h5>

<button type="button" class="close" data-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div class="modal-body">

<!-- Checkout form goes here -->

<form method="post" action="/checkout/">

{% csrf\_token %}

<div class="form-group">

<label for="recipient-name" class="col-form-label">Phone:</label>

<input type="text" class="form-control" id="recipient-name" name="phone">

</div>

<div class="form-group">

<label for="message-text" class="col-form-label">Address:</label>

<textarea class="form-control" id="message-text" name="address"></textarea>

</div>

<div class="form-group">

<label for="message-text" class="col-form-label">Pincode:</label>

<textarea class="form-control" name="pincode"></textarea>

</div>

</form>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>

<button type="submit" class="btn btn-primary">Checkout</button>

</div>

</div>

</div>

</div>

{% endblock %}

```

In this example, I've added a custom CSS style block at the top of your template to set the label text color to white when in dark mode. You can adjust the color value to match your dark mode styling preferences. Additionally, make sure you have the logic in your application to detect whether dark mode is active and add the `dark-mode` class to the appropriate elements when it's active.

# My Prompt:

give me modified code of that form in dark mode

ChatGPT

Certainly, to modify your checkout form for better visibility in dark mode, you can adjust the label styles to have white text on a dark background. Here's an example of how you can modify your form:

```html

<!-- Modal -->

<div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel"

aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalLabel">Checkout</h5>

<button type="button" class="close" data-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div class="modal-body dark-mode">

<!-- Checkout form goes here -->

<form method="post" action="/checkout/">

{% csrf\_token %}

<div class="form-group">

<label for="recipient-name" class="col-form-label">Phone:</label>

<input type="text" class="form-control" id="recipient-name" name="phone">

</div>

<div class="form-group">

<label for="message-text" class="col-form-label">Address:</label>

<textarea class="form-control" id="message-text" name="address"></textarea>

</div>

<div class="form-group">

<label for="message-text" class="col-form-label">Pincode:</label>

<textarea class="form-control" name="pincode"></textarea>

</div>

</form>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>

<button type="submit" class="btn btn-primary">Checkout</button>

</div>

</div>

</div>

</div>

```

In this example, I've added the `dark-mode` class to the `modal-body` element to apply custom styles for dark mode. You can further adjust the text color, background color, or any other styles within the `dark-mode` class to ensure better visibility in dark mode.

Make sure you have appropriate dark mode detection and class application logic in your application to toggle the `dark-mode` class when dark mode is active. Additionally, you may need to adjust other styles in your project to ensure a consistent dark mode experience.

My Prompt:

give css

ChatGpt:

/\* Custom styles for dark mode \*/

.dark-mode {

background-color: #333; /\* Dark background color \*/

color: white; /\* White text color \*/

}

.dark-mode label {

font-weight: bold; /\* Make labels bold for better visibility \*/

}

.dark-mode input[type="text"],

.dark-mode textarea {

background-color: #444; /\* Darker background for input fields \*/

color: white; /\* White text color for input fields \*/

border: 1px solid #666; /\* Dark border for input fields \*/

}

.dark-mode .btn-primary {

background-color: #007bff; /\* Customize primary button background color \*/

border-color: #007bff; /\* Customize primary button border color \*/

}

.dark-mode .btn-primary:hover {

background-color: #0056b3; /\* Customize primary button hover background color \*/

border-color: #0056b3; /\* Customize primary button hover border color \*/

color: white; /\* White text color on hover \*/

}

My Prompt

#### give modified html

ChatGPT

<!-- Modal -->

<div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content dark-mode">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalLabel">Checkout</h5>

<button type="button" class="close" data-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div class="modal-body">

<!-- Checkout form goes here -->

<form method="post" action="/checkout/">

{% csrf\_token %}

<div class="form-group">

<label for="recipient-name" class="col-form-label">Phone:</label>

<input type="text" class="form-control" id="recipient-name" name="phone">

</div>

<div class="form-group">

<label for="message-text" class="col-form-label">Address:</label>

<textarea class="form-control" id="message-text" name="address"></textarea>

</div>

<div class="form-group">

<label for="message-text" class="col-form-label">Pincode:</label>

<textarea class="form-control" name="pincode"></textarea>

</div>

</form>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>

<button type="submit" class="btn btn-primary">Checkout</button>

</div>

</div>

</div>

</div>

My Prompt:

when click the pay button it will open the exixting form and then after filling the details when click checkout it gives that popupand then redirect to index

ChatGpt:

To achieve the behavior you described, you can modify your code as follows:

1. Keep the "Pay" button as it is but without the JavaScript function.

2. Modify the checkout form to include the "Pay" button.

3. Use JavaScript to display the alert after submitting the form.

Here's the modified code:

```html

{% extends 'master.html' %}

{% load cart\_tag %}

{% block body %}

<section id="cart\_items">

<!-- ... Your existing cart items code ... -->

</section> <!--/#cart\_items-->

<br>

<div class="container">

<div class="row">

<div class="col-md-6">

<a href="{% url 'cart\_clear' %}" class="btn btn-default update">Clear Cart</a>

</div>

<div class="col-md-6">

<!-- Keep the "Pay" button without JavaScript -->

<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal">

Pay: ₹ {{ cart\_total\_amount }}

</button>

</div>

</div>

</div>

<!-- Checkout Form Modal -->

<div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel"

aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content dark-mode">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalLabel">Checkout</h5>

<button type="button" class="close" data-dismiss="modal" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

<div class="modal-body">

<!-- Checkout form goes here -->

<form method="post" action="{% url 'checkout' %}">

{% csrf\_token %}

<div class="form-group">

<label for="recipient-name" class="col-form-label">Phone:</label>

<input type="text" class="form-control" id="recipient-name" name="phone">

</div>

<div class="form-group">

<label for="message-text" class="col-form-label">Address:</label>

<textarea class="form-control" id="message-text" name="address"></textarea>

</div>

<div class="form-group">

<label for="message-text" class="col-form-label">Pincode:</label>

<textarea class="form-control" name="pincode"></textarea>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>

<!-- Add a "Checkout" button inside the modal -->

<button type="submit" class="btn btn-primary">Checkout</button>

</div>

</form>

</div>

</div>

</div>

</div>

<script>

// JavaScript to display an alert after form submission

document.addEventListener("submit", function (event) {

if (event.target.tagName === "FORM") {

const cartTotal = {{ cart\_total\_amount }};

// Display an alert with the subtotal amount

alert(`Payment successful! Amount: ₹ ${cartTotal}`);

// Redirect to the index page after the alert

window.location.href = "{% url 'index' %}";

}

});

</script>

{% endblock %}

```

In this code, we've added a "Checkout" button inside the modal form. When the form is submitted, it will display an alert with the subtotal amount and then redirect to the index page.

My Prompt:

I want To send OTP to any mail address It is currently allow malakartanmoy97@gmail.com address only

ChatGpt:

To send OTP to any email address instead of a fixed one, you can modify your code to accept the target email address as input and then send the OTP to that address. Here's how you can do it:

1. Modify your `otp\_login` view function to accept the target email address as a parameter.

2. Update your OTP generation and email sending code to use the provided email address instead of the fixed one.

Here's an updated version of your `otp\_login` view function:

```python

import random

from django.conf import settings

from django.core.mail import send\_mail

from django.http import HttpResponse

from django.shortcuts import redirect, render

def otp\_login(request):

if request.method == 'POST':

# Generate a random OTP (6-digit)

otp = ''.join([str(random.randint(0, 9)) for \_ in range(6)])

# Get the user's email from the form input

email = request.POST.get('email')

# Send the OTP to the user's email

subject = 'Your OTP for Login'

message = f'Your OTP for login: {otp}'

from\_email = settings.EMAIL\_HOST\_USER

recipient\_list = [email]

try:

send\_mail(subject, message, from\_email, recipient\_list)

except Exception as e:

return HttpResponse("Error sending OTP email")

# Store the OTP and email in the user's session

request.session['otp'] = otp

request.session['target\_email'] = email

# Redirect to OTP verification page

return redirect('otp\_verify')

# Render the OTP login form

return render(request, 'otp\_login.html')

```

With this modification, the `otp\_login` view now accepts the email address from the form input and sends the OTP to that address. It also stores both the OTP and the email address in the user's session for later verification.

Make sure to update your `otp\_verify` view as well to retrieve the email address from the session when verifying the OTP.

Now, you can send OTPs to any valid email address entered by the user.

My Prompt:

{% extends 'master.html' %} {% block body %} <section id="form"> <div class="container"> <div class="row"> <div class="signup-form"> <h2>Signup account</h2> <form method="post"> {% csrf\_token %} <!-- Display non-field errors at the top --> {% if form.non\_field\_errors %} <div class="alert alert-danger" role="alert"> {% for error in form.non\_field\_errors %} {{ error }} {% endfor %} </div> {% endif %} <!-- Username --> <div class="form-group"> {{ form.username.label\_tag }} {{ form.username }} </div> <!-- Email --> <div class="form-group"> {{ form.email.label\_tag }} {{ form.email }} {% for error in form.email.errors %} <div class="alert alert-danger" role="alert"> {{ error }} </div> {% endfor %} </div> <!-- Password1 --> <div class="form-group"> {{ form.password1.label\_tag }} {{ form.password1 }} {% for error in form.password1.errors %} <div class="alert alert-danger" role="alert"> {{ error }} </div> {% endfor %} </div> <!-- Password2 --> <div class="form-group"> {{ form.password2.label\_tag }} {{ form.password2 }} {% for error in form.password2.errors %} <div class="alert alert-danger" role="alert"> {{ error }} </div> {% endfor %} </div> <button class="btn btn-default" type="submit">Signup</button> </form> </div> </div> </div> </section> {% endblock %}

Here modify rhe form and display text to user that If redirection happen after sign up then email address already use in database or password mismatch

ChatGPT

To display a message to the user in case of a redirection after signup due to a duplicate email or password mismatch, you can add a conditional check in your view and pass a message to the template. Here's how you can modify your view and template:

1. In your view, check for a `redirect\_msg` variable and pass it to the template if necessary.

```python

def signup(request):

redirect\_msg = None # Initialize the message variable

if request.method == 'POST':

form = UserCreateFrom(request.POST)

if form.is\_valid():

new\_user = form.save()

new\_user = authenticate(

username=form.cleaned\_data['username'],

password=form.cleaned\_data['password1']

)

login(request, new\_user)

return redirect('index')

else:

# Check for specific error cases and set the message accordingly

if 'email' in form.errors and 'unique' in form.errors['email'][0]:

redirect\_msg = "Email address is already in use."

elif 'password1' in form.errors and 'password2' in form.errors['password1'][0]:

redirect\_msg = "Passwords do not match."

else:

form = UserCreateFrom()

context = {

'form': form,

'redirect\_msg': redirect\_msg, # Pass the message to the template

}

return render(request, 'registration/signup.html', context)

```

2. In your template, you can add a section to display the `redirect\_msg` if it exists:

```html

<!-- Display non-field errors at the top -->

{% if redirect\_msg %}

<div class="alert alert-danger" role="alert">

{{ redirect\_msg }}

</div>

{% endif %}

```

With this code, if a user encounters an error during signup (e.g., duplicate email or password mismatch), the error message will be displayed at the top of the signup form.