Password Reset

Password Reset Functionality in Django

Implementation

Creating a password reset functionality in Django involves several steps. This guide will walk you through the entire process from start to finish, covering everything needed for a complete and secure implementation of password reset functionality.

1. Setting Up Django Authentication

Django's built-in authentication system provides a ready-to-use password reset feature. To begin, ensure that the `django.contrib.auth` and `django.contrib.sessions` apps are included in your `INSTALLED_APPS` list in `settings.py` :

2. Configuring Email Settings

Password reset requires sending an email with a reset link. Set up email backend settings in `settings.py` :

```python
# settings.py
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'smtp.your_email_provider.com'
EMAIL_PORT = 587
EMAIL_USE_TLS = True
EMAIL_HOST_USER = 'your-email@example.com'
EMAIL_HOST_PASSWORD = 'your-email-password'


# Alternatively, for development/testing, use:
# EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'
```

Using `console.EmailBackend` is useful during development as it prints the email to the console instead of sending it.

3. URL Patterns for Password Reset

Django provides built-in views to handle password reset. In your `urls.py`, include the following:

```python
# urls.py
from django.contrib.auth import views as auth_views
from django.urls import path

urlpatterns = [
    # Password reset URLs
    path('password_reset/', auth_views.PasswordResetView.as_view(), name='password_reset'),
    path('password_reset/done/', auth_views.PasswordResetDoneView.as_view(), name='password_reset_done'),
    path('reset/<uidb64>/<token>/', auth_views.PasswordResetConfirmView.as_view(), name='password_reset_confirm'),
    path('reset/done/', auth_views.PasswordResetCompleteView.as_view(), name='password_reset_complete'),
]
```

4. Adding Templates for Password Reset

You need templates for rendering the password reset forms and emails. Django looks for these templates in your templates directory:

1. `password_reset_form.html` - For users to enter their email.
2. `password_reset_done.html` - Confirmation page after a reset email is sent.
3. `password_reset_confirm.html` - For entering the new password.
4. `password_reset_complete.html` - Confirmation page after the password is successfully reset.
5. `password_reset_email.html` - Email template with the reset link.

Example Template Files

- `templates/registration/password_reset_form.html` :

```
{% block content %}
<h2>Reset your password</h2>
<form method="post">
  {% csrf_token %} {{ form.as_p }}
  <button type="submit">Reset Password</button>
</form>
{% endblock %}
```

- `templates/registration/password_reset_done.html` :

```
{% block content %}
<p>An email has been sent with instructions to reset your password.</p>
{% endblock %}
```

- `templates/registration/password_reset_confirm.html`:

```
{% block content %}
<h2>Enter new password</h2>
<form method="post">
  {% csrf_token %} {{ form.as_p }}
  <button type="submit">Change Password</button>
</form>
{% endblock %}
```

- `templates/registration/password_reset_complete.html`:

```
{% block content %}
<p>
  Your password has been successfully reset. You can now
  <a href="{% url 'login' %}">log in</a> with your new password.
</p>
{% endblock %}
```

- `templates/registration/password_reset_email.html` :

```
{% block content %}
<p>Hello,</p>
<p>
  You're receiving this email because you requested a password reset for your
  account. Click the link below to reset your password:
</p>
<p>
  <a
    href="{{ protocol }}://{{ domain }}{% url 'password_reset_confirm' uidb64=uid token=token %}"
    >Reset Password</a
  >
</p>
<p>If you didn't request a password reset, please ignore this email.</p>
{% endblock %}
```

5. Customizing the Password Reset View

You can customize the password reset views by overriding the default class-based views provided by Django. For example:

```
# views.py
from django.contrib.auth.views import PasswordResetView
from django.urls import reverse_lazy

class CustomPasswordResetView(PasswordResetView):
    template_name = 'registration/password_reset_form.html'
    success_url = reverse_lazy('password_reset_done')
    email_template_name = 'registration/password_reset_email.html'
```

Then, update the URL configuration to use your custom view:

```
# urls.py
path('password_reset/', CustomPasswordResetView.as_view(), name='password_reset'),
```

6. Testing the Password Reset Functionality

To test the password reset feature:

1. Visit the password reset page: `/password_reset/` .
2. Enter an email associated with a user account.
3. Check the console (if using the console backend) or your email inbox for the reset link.
4. Follow the link to reset the password.

7. Advanced Customizations (Optional)

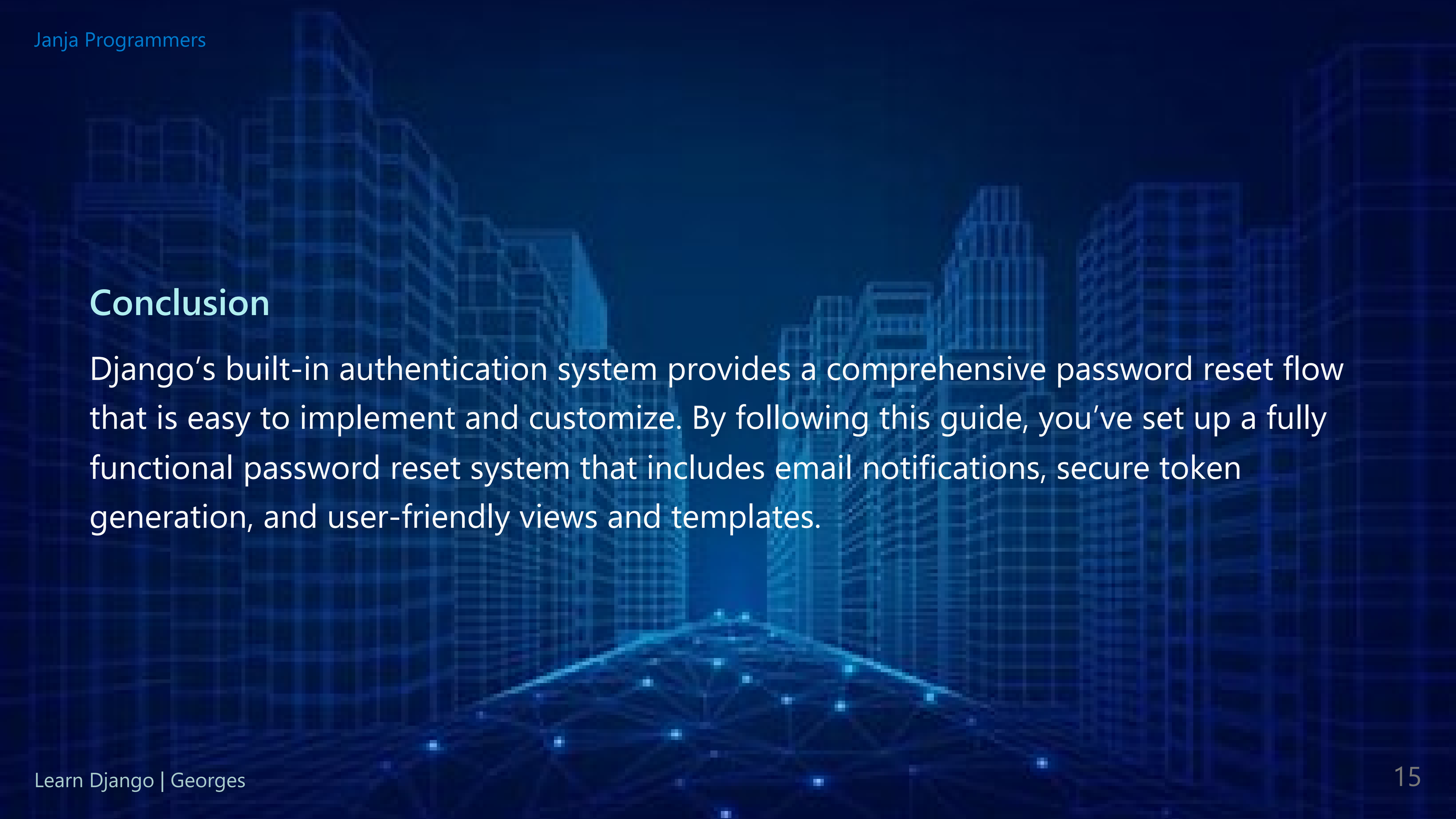
- **Customizing the Email Subject:** You can customize the subject line by overriding the `subject_template_name` :

```
class CustomPasswordResetView(PasswordResetView):
    subject_template_name = 'registration/password_reset_subject.txt'
```

- **Customizing the Email Context:** You can modify the context used in the email by overriding the `get_context_data` method.

8. Securing the Password Reset Feature

- Ensure your email settings use TLS/SSL.
- Implement rate limiting to prevent abuse (e.g., django-ratelimit).
- Use strong password validation with Django's password validators.



Conclusion

Django’s built-in authentication system provides a comprehensive password reset flow that is easy to implement and customize. By following this guide, you’ve set up a fully functional password reset system that includes email notifications, secure token generation, and user-friendly views and templates.