

# DJANGO USER REGISTRATION WITH EMAIL CONFIRMATION

- NAME: BASINA MAHA LAKSHMI VENKATA VALLI

- ROLL NO: 21T91A1205

* COLLEGE: GIET Engineering College
* FULL STACK PYTHON INTERNSHIP

# ABLE OF CONTENTS

**T**

1. Introduction



* + Project Overview
  + Objectives
  + Scope

1. Setup Development Environment
   * Install Necessary Software
   * Set Up Integrated Development Environment (IDE)
2. Project Initialization
   * Create Project Directory
   * Initialize Version Control System
   * Create Django Project
   * Create Django App for Todo
3. Integrate Frontend Code
   * Move Frontend Files
   * Configure Static and Templates
   * Update URL Patterns
4. Create Models, Views, and Forms
   * Create Models
   * Create Forms
   * Create Views
   * Create HTML Template



1. Create Django Admin and Superuser
   * Register Models in Admin
   * Create and Apply Migrations
   * Create Superuser
2. Run the Development Server
3. Testing
   * Unit Testing
   * Integration Testing
   * User Acceptance Testing
4. Conclusion
   * Future Enhancements
5. References



# INTRODUCTION

* 1. **Project Overview**

This documentation provides a comprehensive guide to the To-Do application built using Django. The application allows users to manage tasks by viewing a list of tasks and details for individual tasks.

# Objectives

* To develop a user-friendly registration system.
* To ensure secure user authentication.
* To implement robust backend functionality using Django.
* To integrate frontend and backend seamlessly.

# Scope

This document covers the steps for setting up the development environment, initializing the project, integrating frontend code, creating models, views, and forms, setting up Django admin, and running the development server.



# SETUP DEVELOPMENT ENVIRONMENT

* 1. **Install Necessary Software**
* Install Python from [python.org](https://[www.python.org/downloads/).](http://www.python.org/downloads/))
* Install Django using pip:
* pip install django

# Set Up Integrated Development Environment (IDE)

* Install an IDE like Visual Studio Code, PyCharm, or IntelliJ IDEA.



# PROJECT INTIALIZATION

* 1. **Create Project Directory**

mkdir todo\_application

cd todo\_application

# Initialize Version Control System

git init

# Create Django Project

django-admin startproject todo\_application.

# Create Django App for Todo

python manage.py startapp todo



# INTEGRATE FRONTEND CODE

* 1. **Move Frontend Files**

Move HTML, CSS, and JavaScript files into the `todo` app's `templates` and `static` directories.

# Configure Static and Templates

In `settings.py`:

```python import os

STATIC\_URL = '/static/'

STATICFILES\_DIRS = [

os.path.join(BASE\_DIR, 'todo/static'),

]

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates', 'DIRS': [os.path.join(BASE\_DIR, 'todo/templates')],

# Move Frontend Files

Move HTML, CSS, and JavaScript files into the `todo` app's `templates` and `static` directories.

# Configure Static and Templates

In `settings.py`:

```python import os

STATIC\_URL = '/static/'

STATICFILES\_DIRS = [

os.path.join(BASE\_DIR, 'accounts/static'),

]

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates', 'DIRS': [os.path.join(BASE\_DIR, 'accounts/templates')],



'APP\_DIRS': True, 'OPTIONS': {

'context\_processors': [ 'django.template.context\_processors.debug', 'django.template.context\_processors.request', 'django.contrib.auth.context\_processors.auth', 'django.contrib.messages.context\_processors.messages',

],

},

},

]

```

# Update URL Patterns

In `user\_registration/urls.py`:

```python

from django.contrib import admin from django.urls import path, include

urlpatterns = [

path('admin/', admin.site.urls), path('accounts/', include('accounts.urls')),



'APP\_DIRS': True, 'OPTIONS': {

'context\_processors': [ 'django.template.context\_processors.debug', 'django.template.context\_processors.request', 'django.contrib.auth.context\_processors.auth', 'django.contrib.messages.context\_processors.messages',

],

},

},

]

```

# 4.3 Update URL Patterns

In `todo/urls.py`:

```python

from django.urls import path

from . import views

urlpatterns = [

path('tasks/', views.task\_list, name='task\_list'),

path('tasks/<int:id>/', views.task\_detail, name='task\_detail'),

]



In `todo\_project/urls.py`:

```python

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

path('admin/', admin.site.urls),

path('', include('todo.urls')),

]```

---



]

```

In `accounts/urls.py`:

```python

from django.urls import path from . import views

urlpatterns = [

path('register/', views.register, name='register'),

]

```

---



# CREATE MODELS, VIEWS, AND FORMS

* 1. **Create Models**

In `todo/models.py`:

```python

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

from .models import Task

def task\_list(request):

tasks = Task.objects.all()

return render(request, 'todo/task\_list.html', {'tasks': tasks})

def task\_detail(request, id):

task = get\_object\_or\_404(Task, id=id)

return render(request, 'todo/task\_detail.html', {'task': task})```

# CREATE MODELS, VIEWS

* 1. **Create Models**

In `todo/models.py`:

```python

from django.db import models

class Task(models.Model):

title = models.CharField(max\_length=100)

description = models.TextField(blank=True)

due\_date = models.DateField(null=True, blank=True)

def \_\_str\_\_(self):

return self.title```



# Create Views

In `todo/views.py`:

```python

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

from .models import Task

def task\_list(request):

tasks = Task.objects.all()

return render(request, 'todo/task\_list.html', {'tasks': tasks})

def task\_detail(request, id):

task = get\_object\_or\_404(Task, id=id)

return render(request, 'todo/task\_detail.html', {'task': task})



```

# Create HTML Template

In `todo/templates/todo/task\_list.html`:

```html

<!DOCTYPE html>

<html>

<head>

<title>Task List</title>

<link rel="stylesheet" href="{% static 'styles/main.css' %}">

</head>

<body>

<h1>Task List</h1>

<ul>

{% for task in tasks %}

<li><a href="{% url 'task\_detail' task.id %}">{{ task.title }}</a></li>

{% endfor %}

</ul>

</body>

</html>

```



# 5.4 Create HTML Template

In ` todo/templates/todo/task\_detail.html

```html

<!DOCTYPE html>

<html>

<head>

<title>Task Detail</title>

<link rel="stylesheet" href="{% static 'styles/main.css' %}">

</head>

<body>

<h1>{{ task.title }}</h1>

<p>{{ task.description }}</p>

<p>Due Date: {{ task.due\_date }}</p>

<a href="{% url 'task\_list' %}">Back to Task List</a>

</body>

</html>

</body>



</html>

```

# CREATE DJANGO ADMIN AND SUPERUSER

* 1. **Register Models in Admin**

In ` todo/ models.py`:

```python

from django.db import models

class Task(models.Model):

title = models.CharField(max\_length=100)

description = models.TextField(blank=True)

due\_date = models.DateField(null=True, blank=True)

def \_\_str\_\_(self):

return self.title```

# Create and Apply Migrations

```sh

python manage.py makemigrations

python manage.py migrate

```

# Create Superuser



python manage.py createsuperuser



# RUN THE DEVELOPMENT SERVER Start the Django development server:

```sh

python manage.py runserver

```

Visit `http://localhost:8000/accounts/register/` to see the integrated frontend and backend registration form.



# TESTING

* 1. **Unit Testing**
* Verify that tasks are created with the correct attributes and that model methods work as expected.

-Verify that views return the correct status codes, use the correct templates, and contain the expected content.

* Verify that forms validate input correctly and save data to the database..

# Integration Testing

* Test the full process of creating a task through the application’s UI and ensure it is correctly saved in the database.

-Test the process of updating a task through the application’s UI and ensure the changes are correctly saved in the database.

* Test the process of deleting a task through the application’s UI and ensure it is correctly removed from the database.

# User Acceptance Testing

* Gather feedback from users to ensure the registration system is user-friendly.
* Make adjustments based on user feedback to improve the user experience.



# Conclusion

The To-Do application project serves as a comprehensive demonstration of building a full-stack web application using Django. Throughout the development process, we focused on creating a well-structured, functional, and user-friendly application that allows users to manage their tasks efficiently. The project included key aspects of web development

# Future Enhancements

* User Authentication and Authorization.
* Improve the user interface for better user experience.
* Task Categorization and Tagging
* Due Dates and Reminders
* Performance Optimization
* Deployment and Scaling