ChatGPT

Complete Step-by-Step Guide: Setting Up the Portal API (Django) Project

This guide walks you through creating a Django-based Portal API service from scratch, covering environment setup, dependencies, project structure, database configuration, migrations, and common troubleshooting (including xlsxwriter installation).

Prerequisites

- **Python 3.10+** installed and added to your PATH
- **MySQL** server (or compatible) running locally or remotely
- **Git** (optional, for version control)
- Basic familiarity with the command line (bash, PowerShell, etc.)

Step 1: Create and Enter the Project Directory

```
# Choose a parent directory, then:
mkdir portal_api_service
cd portal_api_service
```

This isolates your project in its own folder.

Step 2: Set Up a Python Virtual Environment

```
# Create a virtual environment named 'venv'
python -m venv venv

# Activate it:
# Windows
venv\Scripts\activate
# macOS / Linux
source venv/bin/activate
```

Ensure your shell prompt indicates the environment is active (e.g., `(venv)` prefix).

Step 3: Define Project Dependencies

Create a file named `requirements.txt` in the project root with these lines:

```
Django>=4.2,<5.0
djangorestframework>=3.14.0
djangorestframework-simplejwt>=5.2.2
mysqlclient>=2.1.1
django-cors-headers>=4.0.0
python-dotenv>=1.0.0
celery>=5.2.7
django-filter>=23.2
drf-yasg>=1.21.5
pytest>=7.3.1
pytest-django>=4.5.2
xlsxwriter>=3.0.3
```

> **Tip:** Keep this list updated as you add new packages.

Step 4: Install Dependencies

With the virtual environment active, run:

```
pip install -r requirements.txt
```

Watch for any errors—if a module fails to install, address it before proceeding.

Step 5: Start the Django Project and Main App

```
# Create the Django project named 'portal_api'
django-admin startproject portal_api .

# (Optional) Create core Django apps; adjust names as needed:
python manage.py startapp users
python manage.py startapp demo
python manage.py startapp exams
python manage.py startapp adminpanel
```

Your directory structure should now look like:

```
portal_api_service/
├── portal_api/
│   ├── __init__.py
│   ├── asgi.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
├── users/
├── demo/
├── exams/
├── adminpanel/
├── manage.py
└── requirements.txt
```

Step 6: Configure Settings

Open `portal_api/settings.py` and ensure the following:

1. **Installed Apps**:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # Third-party
    'rest_framework',
    'corsheaders',
    'django_filters',
    # Your apps
    'users', 'demo', 'exams', 'adminpanel',
]
```

1. **Middleware** (include CORS):

```
MIDDLEWARE = [
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
```

```
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

1. **Database** (using MySQL):

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'portal_db',
        'USER': 'api_service_user',
        'PASSWORD': 'mypassword',
        'HOST': 'localhost',
        'PORT': '3306',
    }
}
```

1. **Static & Media** (adjust as needed):

```
STATIC_URL = '/static/'
MEDIA_URL = '/media/'
```

1. **Environment Variables** (using python-dotenv)

2. Create a `.env` file for secrets (never commit to Git!).

3. Load them in `settings.py` with `python-dotenv`.

Step 7: Create `manage.py` (If Missing)

If you started with `django-admin startproject`, `manage.py` is created automatically. Otherwise, ensure `manage.py` contains:

```
#!/usr/bin/env python
import os, sys

def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'portal_api.settings')
    from django.core.management import execute_from_command_line
    execute_from_command_line(sys.argv)
```

```
if __name__ == '__main__':
    main()
```

Make it executable on UNIX-like systems:

```
chmod +x manage.py
```

Step 8: Set Up MySQL Database & User

Log into MySQL and run:

```
CREATE DATABASE portal_db;
CREATE USER 'api_service_user'@'localhost' IDENTIFIED BY 'mypassword';
GRANT ALL PRIVILEGES ON portal_db.* TO 'api_service_user'@'localhost';
FLUSH PRIVILEGES;
```

> **Security Tip:** Use a strong password and restrict host access in production.

Step 9: Run Migrations

Generate and apply migrations for all apps:

```
python manage.py makemigrations users demo exams adminpanel
python manage.py migrate
```

This creates the necessary database tables.

Step 10: Install & Verify `xlsxwriter`

If you encounter the `ModuleNotFoundError: No module named 'xlsxwriter'`, install it:

```
pip install xlsxwriter
```

Then rerun any command (e.g., `createsuperuser`) to verify the error is resolved.

Step 11: Create a Superuser & Start the Server

1. **Create superuser**:

```
python manage.py createsuperuser
```

2. **Run development server**:

```
python manage.py runserver
```

Open http://127.0.0.1:8000/admin/ in your browser, log in with your superuser credentials, and verify access.

Optional: Celery & Redis Setup

1. Install and run **Redis** (or another broker).
2. In `.env`, add `CELERY_BROKER_URL=redis://localhost:6379/0`.
3. Create `portal_api/celery.py`:

```
from celery import Celery
import os

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'portal_api.settings')
app = Celery('portal_api')
app.config_from_object('django.conf:settings', namespace='CELERY')
app.autodiscover_tasks()
```

4. Launch a worker:

```
celery -A portal_api worker --loglevel=info
```

Troubleshooting & Tips

- **Dependency Conflicts**: Use `pip check` to find mismatches.
- **Static Files**: Run `python manage.py collectstatic` in production.
- **Environment Management**: Use `.env` and `django-environ` or `python-dotenv`.
- **Testing**: Configure `pytest` and `pytest-django`—add `pytest.ini` with:

```
[pytest]
DJANGO_SETTINGS_MODULE = portal_api.settings
```

You're all set! Your Django-based Portal API is configured and ready for development. Happy coding!