**🧑‍💻 MANGESH SETUP GUIDE – POLYRISE PROJECT**

✅ You will be accessing and contributing to a shared Django project hosted on GitHub.

**🔍 1. BEFORE CLONING – CHECK SYSTEM & EXTENSIONS**

**✅ Requirements on Mangesh PC**

* ✅ Python installed (Python 3.10 or above)
  + Check: python --version in terminal
* ✅ Git installed
  + Check: git --version
* ✅ VS Code installed
* ✅ These VS Code Extensions installed:
  + 🟦 Python (by Microsoft)
  + 🟦 Pylance (for IntelliSense)
  + 🟦 Django (optional but helpful)
  + 🟦 GitLens (optional for Git UI)
  + 🟦 Prettier (optional for formatting)

**🧠 Tip:**

To install extensions, go to VS Code → Extensions sidebar → search and install.

**📥 2. CLONE THE PROJECT**

1. **Open Terminal in VS Code**
2. **Run:**

bash

CopyEdit:

git clone https://github.com/YOUR\_USERNAME/PolyRise.git

cd PolyRise

✅ Now you are inside the project folder.

**🛠️ 3. CREATE VIRTUAL ENVIRONMENT**

bash

CopyEdit

python -m venv env

* This creates a folder env/ which contains isolated Python packages.

**🚀 4. ACTIVATE THE ENVIRONMENT**

On Windows:

bash

CopyEdit

.\env\Scripts\activate

You will see (env) before the path in terminal if it's activated.

**📦 5. INSTALL DEPENDENCIES**

bash

CopyEdit

pip install -r requirements.txt

✅ This installs Django, Pillow, etc., which are required for the project to run.

**🔐 6. CREATE .env FILE (IMPORTANT)**

Since .env is **not pushed** to GitHub (for security), create it manually.

1. In the root project folder:

bash

CopyEdit

touch .env

1. Paste the values (ask your friend to send):

env

CopyEdit

SECRET\_KEY=your-secret-key

DEBUG=True

**🧱 7. APPLY DATABASE MIGRATIONS**

bash

CopyEdit

python manage.py makemigrations

python manage.py migrate

✅ Creates db.sqlite3 and required tables.

**🖥️ 8. RUN THE SERVER**

bash

CopyEdit

python manage.py runserver

✅ Visit: <http://127.0.0.1:8000/>

🎉 You’re now running PolyRise on your local PC!

**🔄 COLLABORATION GUIDE**

**✅ Before You Start Editing:**

bash

CopyEdit

git pull

✅ Gets the latest changes from the team.

**✅ After You Make Changes:**

bash

CopyEdit

git add .

git push

**🚫 FILES NEVER TO PUSH TO GITHUB**git commit -m "Add feature: xyz"-(here what you work write in short but in “……”)

These must stay **locally only**, never commit or push:

| **File/Folder** | **Why** |
| --- | --- |
| env/ | Local environment only |
| .env | Secret credentials |
| db.sqlite3 | Contains local database (optional to share) |
| \_\_pycache\_\_/ | Auto-generated cache |

✅ Already handled in .gitignore

**🧠 EXTRA TIPS**

* Always **activate your virtual environment** before running commands.
* Use **git status** to check which files changed.
* You can edit Python files and Django templates inside accounts, core, or other apps.

**✅ QUICK SUMMARY (FOR FRIEND)**

| **Task** | **Required?** | **You Do It?** |
| --- | --- | --- |
| Clone repo | ✅ | Yes |
| Install extensions | ✅ | Yes |
| Setup virtual env | ✅ | Yes |
| Install from requirements.txt | ✅ | Yes |
| Create .env manually | ✅ | Yes |
| Migrate database | ✅ | Yes |
| Run project | ✅ | Yes |
| **✅ What happens when your friend installs new packages?**  If your friend installs a new Python/Django package in his environment using:  bash  CopyEdit  pip install somepackage  Then **by default**, it will install **only on his PC** — you won’t see it unless he updates the requirements.txt.  **✅ How to sync those new packages to you?**  To make sure **you also get those new packages**, he must run:  bash  CopyEdit  pip freeze > requirements.txt  This command will regenerate the requirements.txt with all currently installed packages.  Then he must push the updated file:  bash  CopyEdit  git add requirements.txt  git commit -m "Updated requirements with new packages"  git push origin main  **✅ Then on your side:**  You can run:  bash  CopyEdit  git pull origin main  pip install -r requirements.txt  That will install the **same new packages** on your side too.  **🚫 Important Note:**   * If he installs a package but **forgets to update and push requirements.txt**, then you **won’t know**. * You may face errors like ModuleNotFoundError when trying to run parts of his code.   **✅ Best Practice (Team Rule):**  Tell your friend:  “Whenever you install a new package, always run pip freeze > requirements.txt and push it.”  **📌 Summary Table:**   | **Action** | **You See It?** | **What to Do** | | --- | --- | --- | | Friend installs package | ❌ No | Must run pip freeze > requirements.txt | | Friend pushes updated requirements | ✅ Yes | You pull and run pip install -r | |  |  |