رفیق، اگه بخوای میتونم همین الان برات ورک‌فلو شاخه‌بندی Git توی PyCharm رو هم بچینم که هر ماژول پروژه فرح پرو رو توی شاخه جدا بسازی و بعد Merge کنی، این باعث میشه همیشه main تمیز و پایدار باشه. میخوای این رو هم بچینم؟

**ChatGPT said:**

باشه رفیق 😎  
بیاین یه **ورک‌فلو شاخه‌بندی (Branch Workflow)** شسته‌رفته برای پروژه **فرح پرو** بچینیم که همیشه main تمیز باشه و هر اپ/ماژول رو توی شاخه خودش توسعه بدی، بعد Merge کنی.

**🔹 ساختار شاخه‌ها**

من پیشنهاد می‌کنم این سه شاخه اصلی رو همیشه داشته باشیم:

1. **main** → نسخه پایدار و آماده برای انتشار.
2. **develop** → نسخه در حال توسعه که همه فیچرهای تکمیل‌شده رو قبل از انتشار جمع می‌کنیم.
3. **شاخه‌های فیچر (feature branches)** → برای هر اپ یا قابلیت جدید یک شاخه مجزا.

📌 نام‌گذاری شاخه‌های فیچر:

bash

CopyEdit

feature/accounts

feature/organizations

feature/chart-of-accounts

feature/invoices

...

**🔹 چرخه کاری (Workflow)**

1. همیشه کد پایدار توی main میمونه.
2. برای توسعه بلندمدت از develop استفاده می‌کنیم.
3. برای هر ماژول یا تغییر مهم → یک شاخه جدید از develop می‌گیری.
4. بعد از اتمام کار اون فیچر → Merge میکنی به develop.
5. وقتی چند فیچر کامل شد و تست‌ها پاس شد → develop رو Merge میکنی به main.

**🔹 اجرای این سیستم در PyCharm**

**1) ساخت شاخه‌ها**

* پایین سمت راست PyCharm روی اسم شاخه (مثلاً main) کلیک کن.
* گزینه **+ New Branch** رو بزن.
* برای شروع، شاخه develop رو از روی main بساز.

**2) ساخت شاخه فیچر**

* وقتی خواستی روی یک اپ کار کنی:
  1. روی اسم شاخه کلیک → **+ New Branch**
  2. اسم بذار مثلاً:

bash

CopyEdit

feature/accounts

* 1. مطمئن شو **Create from: develop** انتخاب شده.

**3) تغییرات و Commit & Push**

* توی همون شاخه فیچر تغییراتت رو انجام بده.
* **Commit** و **Push** فقط به همون شاخه (نه main).

**4) Merge به develop**

* بعد از اتمام کار فیچر، از پایین سمت راست روی شاخه کلیک → **Merge into Current** و شاخه فیچر رو به develop Merge کن.
* اگر کانفلیکت پیش اومد، PyCharm ابزار Resolve Conflicts رو باز میکنه، میتونی انتخاب کنی کدوم تغییر بمونه.

**5) Merge به main**

* وقتی همه فیچرها کامل و تست شدن → از develop به main Merge کن.

**🔹 نکات مهم برای این پروژه**

* هر اپ **فرح پرو** (accounts، organizations، transactions و …) باید توی شاخه خودش ساخته و تست بشه.
* main رو نباید مستقیم Commit بزنی، فقط از Merge استفاده کن.
* هر روز قبل از شروع کار → develop رو **Pull** کن که با آخرین تغییرات به‌روز باشه.

📌 اگر بخوای میتونم **الان لیست دقیق شاخه‌های فیچر** برای همه اپ‌های پروژه رو برات بسازم و حتی دستور ایجادشون رو هم بدم که یک‌باره همه آماده باشن.  
اینطوری از فردا مستقیم میتونی وارد توسعه بشی.

خب رفیق، این هم لیست کامل شاخه‌های Git که بر اساس اپ‌های پروژه **فرح پرو** برات چیدم تا از روز اول همه آماده باشن و فقط بری سراغ توسعه.

**1️⃣ شاخه‌های اصلی (یک‌بار ساخته میشن)**

bash

CopyEdit

# از main شروع می‌کنیم

git checkout main

# ساخت develop از main

git checkout -b develop

git push -u origin develop

**2️⃣ شاخه‌های فیچر برای هر اپ**

📌 همه این‌ها باید از روی develop ساخته بشن.

bash

CopyEdit

# Core/Common

git checkout develop

git checkout -b feature/core-common

git push -u origin feature/core-common

# Accounts

git checkout develop

git checkout -b feature/accounts

git push -u origin feature/accounts

# Organizations

git checkout develop

git checkout -b feature/organizations

git push -u origin feature/organizations

# Stores (در صورت نیاز)

git checkout develop

git checkout -b feature/stores

git push -u origin feature/stores

# Chart of Accounts

git checkout develop

git checkout -b feature/chart-of-accounts

git push -u origin feature/chart-of-accounts

# Transactions

git checkout develop

git checkout -b feature/transactions

git push -u origin feature/transactions

# Invoices

git checkout develop

git checkout -b feature/invoices

git push -u origin feature/invoices

# Reporting

git checkout develop

git checkout -b feature/reporting

git push -u origin feature/reporting

# Imports/Exports

git checkout develop

git checkout -b feature/imports-exports

git push -u origin feature/imports-exports

# API

git checkout develop

git checkout -b feature/api

git push -u origin feature/api

# Web Frontend

git checkout develop

git checkout -b feature/web-frontend

git push -u origin feature/web-frontend

# Integrations

git checkout develop

git checkout -b feature/integrations

git push -u origin feature/integrations

# Tasks

git checkout develop

git checkout -b feature/tasks

git push -u origin feature/tasks

# Admin Panel

git checkout develop

git checkout -b feature/admin-panel

git push -u origin feature/admin-panel

# Tests

git checkout develop

git checkout -b feature/tests

git push -u origin feature/tests

# AuditLog

git checkout develop

git checkout -b feature/auditlog

git push -u origin feature/auditlog

**3️⃣ نتیجه نهایی شاخه‌ها**

بعد از اجرای همه این‌ها، توی GitHub/GitLab/Bitbucket یا هر ریپویی که داری، این ساختار رو می‌بینی:

bash

CopyEdit

main

develop

feature/core-common

feature/accounts

feature/organizations

feature/stores

feature/chart-of-accounts

feature/transactions

feature/invoices

feature/reporting

feature/imports-exports

feature/api

feature/web-frontend

feature/integrations

feature/tasks

feature/admin-panel

feature/tests

feature/auditlog

**4️⃣ کار روزمره‌ت چطور میشه؟**

1. قبل از شروع کار →

bash

CopyEdit

git checkout develop

git pull

1. شاخه فیچر موردنظر رو بگیر و واردش شو:

bash

CopyEdit

git checkout feature/transactions

1. کد بزن، Commit کن و Push بزن.
2. بعد از اتمام کار اون ماژول → Merge به develop.
3. در نهایت، وقتی چند ماژول کامل شد → develop رو Merge کن به main.

اگه بخوای، میتونم همین شاخه‌ها رو طوری توی **PyCharm 2025** ست کنم که همه‌شون با دو کلیک ساخته و Push بشن و حتی رنگ‌بندی هم بدم که بدونی الان کدوم شاخه پایداره و کدوم در حال توسعه.

عالی، پس بریم ورک‌فلو شاخه‌بندی Git رو توی **PyCharm 2025** طوری بچینیم که بدون نیاز به تایپ دستور، فقط با چند کلیک شاخه‌ها ساخته، سوئیچ، و Push بشن.

**🛠 تنظیم ورک‌فلو Git Branching در PyCharm**

**1) فعال کردن Git در پروژه**

1. از منو **VCS → Enable Version Control Integration…** رو انتخاب کن.
2. توی لیست، **Git** رو انتخاب کن.
3. الان PyCharm پروژه رو Git-ready میکنه و کنار فایل‌ها علامت **V** رنگی میاد.

**2) اضافه کردن Remote (مثلاً GitHub یا GitLab)**

1. از منو **VCS → Git → Remotes…** رو بزن.
2. روی **+** کلیک کن.
3. آدرس ریپوی Remote رو وارد کن، مثل:

scss

CopyEdit

git@github.com:USERNAME/FarahPro.git

1. OK رو بزن.

**3) ساخت شاخه‌ها در PyCharm (بدون ترمینال)**

برای هر شاخه‌ای که میخوای:

1. پایین سمت راست، روی اسم شاخه (الان احتمالاً main) کلیک کن.
2. گزینه **+ New Branch** رو بزن.
3. اسم بذار (مثلاً develop) و تیک **Checkout branch** رو بزن.
4. بعد از ساخته شدن، دوباره روی اسم شاخه کلیک و گزینه **Push** رو بزن تا به Remote بره.
5. همین کار رو برای شاخه‌های فیچر تکرار کن (همه از روی develop ساخته بشن).

**4) رنگ‌بندی شاخه‌ها برای شناسایی سریع**

1. برو به **File → Settings → Version Control → Git**.
2. توی بخش **Branch Colors** برای شاخه‌ها رنگ انتخاب کن:
   * main → سبز تیره (پایدار)
   * develop → آبی (در حال توسعه کلی)
   * فیچرها → رنگ‌های روشن مختلف (برای تشخیص راحت)

**5) تغییر شاخه و Merge با کلیک**

* برای **Switch Branch** → پایین سمت راست اسم شاخه رو بزن و شاخه جدید رو انتخاب کن.
* برای **Merge** → روی اسم شاخه فعلی کلیک، **Merge into Current** رو انتخاب کن.
* اگر Conflict پیش اومد، PyCharm پنجره **Resolve Conflicts** رو باز میکنه که میتونی گرافیکی حلش کنی.

**6) پیشنهاد من**

می‌تونم همین الان یک **Branch Template Sheet** (لیست کامل شاخه‌ها با ساختار و توضیح هرکدوم) برات بسازم و داخل پروژه ذخیره کنم، تا همیشه بدونی کدوم شاخه برای چی هست.  
این باعث میشه حتی بعد از چند ماه هم اشتباهی روی main کد نزنی 😄

باشه رفیق 😎  
الان برات یه **Branch Template Sheet** می‌سازم که بذاری داخل پروژه‌ت (مثلاً توی ریشه پروژه یا پوشه docs/) تا همیشه ساختار شاخه‌ها و قوانینش جلوی چشمت باشه.

**📄 فایل: git\_branch\_template.md**

markdown

CopyEdit

# FarahPro - Git Branch Template

## شاخه‌های اصلی

| Branch Name | توضیح | وضعیت |

|-------------|-------|-------|

| `main` | نسخه پایدار، همیشه قابل انتشار | 🟢 Stable |

| `develop` | نسخه توسعه اصلی، ادغام فیچرها | 🔵 In Progress |

---

## شاخه‌های فیچر (Feature Branches)

📌 همه فیچرها باید از روی `develop` ساخته شوند.

| Branch Name | توضیح |

|-------------|-------|

| `feature/core-common` | کدهای پایه، middleware، ابزارهای مشترک |

| `feature/accounts` | سیستم کاربران، نقش‌ها، مجوزها |

| `feature/organizations` | مدیریت شرکت/کسب‌وکار (Multi-Tenant) |

| `feature/stores` | مدیریت فروشگاه‌ها (در صورت نیاز جدا) |

| `feature/chart-of-accounts` | سرفصل‌ها، حساب‌ها، ترازها |

| `feature/transactions` | تراکنش‌ها، ثبت و ویرایش |

| `feature/invoices` | فاکتورها، رسیدها، پرداخت‌ها |

| `feature/reporting` | گزارش‌های مالی، ترازنامه |

| `feature/imports-exports` | ایمپورت/اکسپورت CSV و اکسل |

| `feature/api` | REST/GraphQL API و نسخه‌بندی |

| `feature/web-frontend` | صفحات و رابط کاربری وب |

| `feature/integrations` | درگاه پرداخت، پیامک، ایمیل |

| `feature/tasks` | پردازش‌های پس‌زمینه و زمان‌بندی |

| `feature/admin-panel` | پنل مدیریت سیستم |

| `feature/tests` | تست‌های خودکار و Fixtures |

| `feature/auditlog` | ثبت لاگ تغییرات و حسابرسی |

---

## قوانین کلی

1. \*\*هر شاخه فیچر\*\* از `develop` ساخته شود.

2. \*\*Commit\*\*‌ها باید واضح و توصیفی باشند.

3. بعد از اتمام فیچر → Merge به `develop`.

4. قبل از انتشار نسخه پایدار → Merge `develop` به `main`.

5. روی `main` مستقیم کد نزن مگر برای Hotfix فوری.

---

## رنگ‌بندی پیشنهادی در PyCharm

- `main` → سبز تیره 🟢

- `develop` → آبی 🔵

- فیچرها → رنگ‌های روشن متنوع 🌈

---

📅 تهیه‌کننده: [Your Name]

🛠 برای پروژه \*\*FarahPro\*\*

پیشنهاد من اینه که این فایل رو داخل پروژه در مسیر docs/git\_branch\_template.md بذاری و همین الان Commit & Push کنی.  
میخوای من همین الان دستورهاشو هم بهت بدم که این فایل رو اضافه و Push کنی؟

خب رفیق، این کار رو مرحله‌به‌مرحله انجام میدیم تا هم فایل ساخته بشه، هم توی **Git** ثبت بشه و بره روی Remote.

**1️⃣ ساخت پوشه docs و فایل git\_branch\_template.md**

توی PyCharm:

1. روی پوشه اصلی پروژه (FarahPro) راست‌کلیک → **New → Directory** → اسم رو بزن docs.
2. روی پوشه docs راست‌کلیک → **New → File** → اسم رو بزن git\_branch\_template.md.
3. محتوایی که بالا برات نوشتم رو توش کپی کن و ذخیره کن.

**2️⃣ ثبت فایل در Git**

توی PyCharm:

1. سمت چپ (Project View)، کنار فایل جدید علامت قرمز رنگ هست یعنی هنوز **Unversioned** هست.
2. روی فایل راست‌کلیک → **Git → Add** (الان سبز میشه یعنی آماده Commit هست).

**3️⃣ Commit کردن**

1. منو **VCS → Commit…** رو بزن (یا Ctrl + K).
2. توی پنجره Commit، مطمئن شو فقط همین فایل تیک خورده.
3. توی قسمت Message بنویس:

pgsql

CopyEdit

Add Git Branch Template Sheet to docs

1. دکمه **Commit** یا **Commit and Push** رو بزن.

**4️⃣ Push به Remote**

اگر فقط Commit کردی و Push نکردی:

* منو **VCS → Git → Push** (یا Ctrl + Shift + K).
* انتخاب شاخه (main یا develop) → Push.

📌 نکته: پیشنهاد من اینه که این Commit رو روی develop بزنی نه main، تا از همین اول ساختار شاخه‌بندی رعایت بشه.

میخوای همین الان بهت بگم چطور develop رو بسازی و این فایل رو اونجا Commit و Push کنی؟  
اینطوری از همون لحظه اول پروژه رو با ساختار حرفه‌ای شروع می‌کنیم.

**ChatGPT said:**

پس بزن بریم رفیق 😎  
الان گام‌به‌گام **ساخت شاخه develop** و **Commit & Push** رو انجام میدیم.

**1️⃣ ساخت شاخه develop**

توی PyCharm:

1. پایین سمت راست (Git Branch Indicator) روی اسم شاخه فعلی کلیک کن (احتمالاً main هست).
2. **New Branch…** رو انتخاب کن.
3. اسم رو بزن:

nginx

CopyEdit

develop

1. تیک **Checkout Branch** رو بزن تا بعد از ساخت، بری روی این شاخه.

**2️⃣ اضافه کردن فایل به develop**

الان که روی develop هستی:

1. پوشه docs و فایل git\_branch\_template.md رو طبق مراحلی که گفتم بساز.
2. روی فایل راست‌کلیک → **Git → Add**.

**3️⃣ Commit روی develop**

1. Ctrl + K رو بزن (یا منو VCS → Commit).
2. پیام Commit رو بزن:

pgsql

CopyEdit

Add Git Branch Template Sheet to docs

1. دکمه **Commit** رو بزن.

**4️⃣ Push شاخه develop به Remote**

1. Ctrl + Shift + K رو بزن (یا VCS → Git → Push).
2. توی پنجره Push، مطمئن شو شاخه develop انتخاب شده.
3. دکمه **Push** رو بزن.

✅ الان Remote دو شاخه خواهد داشت:

* main (پایدار، فعلاً خالی یا با کد اولیه)
* develop (شاخه توسعه، شامل فایل Template جدید)

اگه بخوای میتونم برات الان **workflow کار روی هر ماژول** رو هم مرحله‌به‌مرحله بچینم که از develop شاخه فیچر بسازی، روش کار کنی، بعد Merge کنی.  
این باعث میشه کدت همیشه تمیز و کنترل‌شده باشه.

باشه رفیق، اینو جوری می‌چینم که هر بار خواستی روی یک ماژول جدید پروژه فرح پرو کار کنی، فقط با چند کلیک از روی **develop** یه شاخه فیچر بسازی، کارتو بکنی، بعد تمیز بیاری توی develop، و main همیشه تمیز بمونه.

**🛠 ورک‌فلو شاخه‌بندی برای هر ماژول**

**1️⃣ شروع کار روی یک ماژول (Feature)**

1. مطمئن شو روی develop هستی.
   * پایین سمت راست (Git Branch Indicator) رو نگاه کن → اگه develop نیست، انتخاب کن.
2. **ساخت شاخه فیچر جدید**:
   * روی اسم شاخه کلیک کن → **New Branch…**
   * اسم رو به این فرمت بزن:

cpp

CopyEdit

feature/<module-name>

مثال:

bash

CopyEdit

feature/accounts

feature/organizations

feature/chart-of-accounts

* + تیک **Checkout Branch** رو بزن.

**2️⃣ توسعه و Commit**

* تغییراتت رو انجام بده (کد، تست، فایل‌ها).
* بعد از هر تغییر مهم:
  + **Add** → **Commit** با پیام واضح:

sql

CopyEdit

Implement user model in accounts app

Add API endpoints for organization list

* + Push کن به Remote.

**3️⃣ اتمام فیچر و ادغام با develop**

وقتی کار فیچر تموم شد:

1. برو روی develop:
   * **Checkout Branch → develop**
2. شاخه فیچر رو Merge کن:
   * منو **Git → Merge into Current…**
   * شاخه فیچر رو انتخاب کن → Merge.
3. تست‌ها رو اجرا کن (python manage.py test یا از PyCharm).
4. Push کن:
   * develop رو Push کن به Remote.

**4️⃣ انتشار نسخه پایدار روی main**

وقتی چند فیچر تموم شد و میخوای نسخه پایدار بدی:

1. برو روی main.
2. Merge کن از develop به main.
3. Tag بزن (مثلاً v1.0.0).
4. Push کن.

**📌 نکات مهم**

* روی main کد نزن مگر برای Hotfix.
* هر شاخه فیچر کوچک و متمرکز باشه (نه همه پروژه توی یه شاخه).
* نام فیچر واضح باشه و با ماژول تطبیق داشته باشه.
* اگر وسط کار نیاز شد تغییرات develop رو بیاری تو فیچر، از **Merge develop into feature** استفاده کن.

اگه بخوای الان میتونم **لیست شاخه‌های پیشنهادی برای همه اپ‌های ERD** رو هم آماده کنم تا مستقیم بسازی و هر کدوم رو جدا توسعه بدی.  
اینطوری حتی قبل از نوشتن اولین خط کد، نقشه شاخه‌ها آماده‌ست.

آها، رفیق، الان آخرین مرحله گیت رو برات به صورت کامل و مرحله‌به‌مرحله میچینم تا همه چی تمیز و کامل باشه.

**🎯 آخرین مرحله مدیریت Git برای پروژه FarahPro**

**1. ساخت شاخه‌های فیچر آماده برای همه ماژول‌ها**

این دستورات رو توی ترمینال PyCharm بزنی، همه شاخه‌ها ساخته میشن و می‌تونی روی هر کدوم جدا کار کنی:

bash

CopyEdit

git checkout develop

git branch feature/core-common

git branch feature/accounts

git branch feature/organizations

git branch feature/stores

git branch feature/chart-of-accounts

git branch feature/transactions

git branch feature/invoices

git branch feature/reporting

git branch feature/imports-exports

git branch feature/api

git branch feature/web-frontend

git branch feature/integrations

git branch feature/tasks

git branch feature/admin-panel

git branch feature/tests

git branch feature/auditlog

**توجه:** شاخه‌ها هنوز Checkout نشده‌اند، فقط ساخته شدند.

**2. Push همه شاخه‌ها به Remote (گیت‌هاب یا هر سرویس دیگه)**

برای هر شاخه‌ای که ساختی، باید Push کنی:

bash

CopyEdit

git push -u origin feature/core-common

git push -u origin feature/accounts

git push -u origin feature/organizations

git push -u origin feature/stores

git push -u origin feature/chart-of-accounts

git push -u origin feature/transactions

git push -u origin feature/invoices

git push -u origin feature/reporting

git push -u origin feature/imports-exports

git push -u origin feature/api

git push -u origin feature/web-frontend

git push -u origin feature/integrations

git push -u origin feature/tasks

git push -u origin feature/admin-panel

git push -u origin feature/tests

git push -u origin feature/auditlog

با -u شاخه روی Remote ست میشه و دفعه‌های بعد راحت Push و Pull انجام میدی.

**3. کار با شاخه‌ها**

* وقتی میخوای روی هر فیچر کار کنی، فقط Checkout کن و تغییرات رو انجام بده:

bash

CopyEdit

git checkout feature/accounts

* بعد از اتمام کار، تغییرات رو Commit و Push کن.
* سپس برو روی develop و Merge کن.

**4. نکات تکمیلی**

* شاخه main فقط برای نسخه‌های پایدار.
* شاخه develop برای ادغام فیچرها و تست‌ها.
* هر فیچر باید کوچک و متمرکز باشه.
* همیشه قبل از Merge به develop، آخرین تغییرات develop رو به فیچر بیار.

اگر خواستی، همین الان می‌تونم یه اسکریپت ساده شل برای ساخت و Push همه شاخه‌ها بدم که فقط کپی کنی اجرا کنی.

باشه رفیق، اینم اسکریپت ساده‌ی Bash برای ساخت و Push همه شاخه‌های فیچر که فقط کافیه توی ترمینال PyCharm یا هر محیط bash-compatible بندازی و اجرا کنی.

bash

CopyEdit

#!/bin/bash

# مطمئن شو روی develop هستی

git checkout develop

# آرایه شاخه‌ها

branches=(

feature/core-common

feature/accounts

feature/organizations

feature/stores

feature/chart-of-accounts

feature/transactions

feature/invoices

feature/reporting

feature/imports-exports

feature/api

feature/web-frontend

feature/integrations

feature/tasks

feature/admin-panel

feature/tests

feature/auditlog

)

# ساخت شاخه‌ها و push به remote

for branch in "${branches[@]}"

do

echo "Creating and pushing $branch"

git branch "$branch"

git push -u origin "$branch"

done

echo "All branches created and pushed successfully!"

**چطوری اجراش کنی؟**

1. این کد رو توی یه فایل متنی مثلاً create\_push\_branches.sh ذخیره کن.
2. توی ترمینال برو به ریشه پروژه (جایی که .git هست).
3. به فایل اجازه اجرا بده:

bash

CopyEdit

chmod +x create\_push\_branches.sh

1. اسکریپت رو اجرا کن:

bash

CopyEdit

./create\_push\_branches.sh

اگه تو ویندوز هستی و Bash نداری، می‌تونیم یه نسخه PowerShell هم آماده کنیم یا این کارها رو دستی توی PyCharm انجام بدی.