Day 1 — FULLY detailed, step-by-step, with **all commands, folder structures, config files, and starter code**.

🚀 DAY 1 — Project Setup (FULL DETAILS + CODE)

Today we will create:

✓ Project folder structure
✓ GitHub repo
✓ Full React + TypeScript + Vite + Tailwind setup
✓ FastAPI backend structure
✓ Base API file (`main.py`)
✓ Clean separation between frontend & backend

Let's start.

🔹 STEP 1 — Create Main Project Structure

On your machine:

mkdir FocusFlow

cd FocusFlow

mkdir frontend backend

touch README.md

Your folder structure now:

FocusFlow/

frontend/

backend/

README.md

🔹 STEP 2 — Initialize a GitHub Repository

Inside the `FocusFlow/` folder:

git init

git add .

git commit -m "Initial project structure"

Then create a GitHub repo named **focusflow** and push:

git remote add origin https://github.com/YOUR_USERNAME/focusflow.git

git branch -M main

git push -u origin main

🟦 PART A — FRONTEND SETUP (React + TypeScript + Vite + Tailwind + Zustand + React Query)

🔹 STEP 3 — Create React App (Vite + TypeScript)

cd frontend

npm create vite@latest

Error:

In case, an error generated due to the current user having an undefined `ExecutionPolicy`.

In PowerShell *as Administrator*, you could try the following:

Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy Unrestricted

Select:

- **Project name:** `frontend`
- **Framework:** React
- **Variant:** TypeScript

Then install dependencies:

npm install

🔹 STEP 4 — Install UI + State Libraries

npm install axios react-query @tanstack/react-query zustand

npm install tailwindcss postcss autoprefixer / npm install -D tailwindcss@3

npx tailwindcss init -p

🔹 STEP 5 — Configure TailwindCSS

Edit `tailwind.config.js`:

```
/** @type {import('tailwindcss').Config} */
export default {
  content: [
    "./index.html",
    "./src/**/*.{js,ts,jsx,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

Edit `src/index.css`:

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

🔹 STEP 6 — Create Basic Page Structure

Inside `src/`, create a `pages/` folder:

```
mkdir src/pages
```

Add the following files:

src/pages/Login.tsx

```
export default function Login() {
  return (
    <div className="h-screen flex items-center justify-center bg-gray-100">
      <h1 className="text-3xl font-bold">Login Page</h1>
    </div>
  );
}
```

---

src/pages/Register.tsx

```
export default function Register() {
  return (
    <div className="h-screen flex items-center justify-center bg-gray-100">
      <h1 className="text-3xl font-bold">Register Page</h1>
    </div>
  );
}
```

---

src/pages/Dashboard.tsx

```
export default function Dashboard() {
  return (
    <div className="min-h-screen bg-gray-50 p-4">
      <h1 className="text-3xl font-bold">Dashboard</h1>
    </div>
```

```
  );
}
```

🔹 STEP 7 — Set Up Routing

Install React Router:

```
npm install react-router-dom
```

Edit `src/main.tsx`:

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Login from './pages/Login';
import Register from './pages/Register';
import Dashboard from './pages/Dashboard';
import './index.css'

ReactDOM.createRoot(document.getElementById('root')!).render(
  <React.StrictMode>
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Login />} />
        <Route path="/register" element={<Register />} />
        <Route path="/dashboard" element={<Dashboard />} />
      </Routes>
```

```
    </BrowserRouter>
  </React.StrictMode>,
)
```

Frontend base is READY ✓

🟩 PART B — BACKEND SETUP (FastAPI + SQLAlchemy)

🔷 STEP 8 — Create Python Virtual Environment

cd ../backend

python -m venv venv

Activate:

Mac/Linux:

source venv/bin/activate

Windows:

venv\Scripts\activate

🔹 STEP 9 — Install Backend Dependencies

pip install fastapi uvicorn python-dotenv sqlalchemy psycopg2-binary pydantic passlib[bcrypt]

🔹 STEP 10 — Create Backend Folder Structure

Inside `backend/`:

backend/

app/

main.py

database.py

models/

routes/

auth/

Create folders:

mkdir app app/models app/routes app/auth

touch app/main.py app/database.py

🔹 STEP 11 — Base FastAPI App

app/main.py

```
from fastapi import FastAPI


app = FastAPI()


@app.get("/")
def home():
    return {"message": "FocusFlow API is running 🚀"}
```

Run server:

```
uvicorn app.main:app --reload
```

You should see:

http://127.0.0.1:8000

🔹 STEP 12 — Configure Database Connection

app/database.py

```
from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker
import os
from dotenv import load_dotenv
```

```
load_dotenv()


DATABASE_URL = os.getenv("DATABASE_URL")


engine = create_engine(DATABASE_URL)

SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)

Base = declarative_base()


def get_db():

    db = SessionLocal()

    try:

        yield db

    finally:

        db.close()
```

Create `.env` file:

```
DATABASE_URL=postgresql://username:password@host/dbname
```

Example:

```
DATABASE_URL=postgresql://postgres:myPassword@db.supabase.com:5432/focusflow_db
```

Here is a **simple, clear, step-by-step guide** to install PostgreSQL on Windows, create a database, and connect it to your FastAPI project without errors — including how to set up your **.env** file correctly.

✅ 1. Install PostgreSQL on Windows

Step 1 — Download PostgreSQL

1. Go to: https://www.postgresql.org/download/windows/
2. Download the installer from **EDB**.

Step 2 — Run the Installer

During installation, you will be asked:

- **Installation Directory** → leave default
- **Components** → make sure these are checked:
  ✓ PostgreSQL Server
  ✓ pgAdmin 4
  ✓ Command Line Tools
- **Superuser password** → write this down!
  Example: `mysecretpassword`
- **Port** → default is `5432`
- **Locale** → default

Click **Next** → **Finish**.

✅ 2. Open pgAdmin and Create a Database

Step 1 — Open pgAdmin 4

- You'll find it in the Start menu.

Step 2 — Connect

- It will ask for the PostgreSQL **master password** you set during installation.

Step 3 — Create a New Database

In pgAdmin:

1. Left sidebar → expand **Servers**
2. Expand **PostgreSQL 16** (or your version)
3. Right-click **Databases** → **Create** → **Database…**
4. Fill:

- **Database name:** `myappdb`
- **Owner:** `postgres`

Click **Save**.

✅ 3. Find Your PostgreSQL Connection Information

You need:

Item	Value
USERNAME	`postgres`
PASSWORD	(the password you entered)
HOST	`localhost`
PORT	`5432`
DATABASE NAME	`myappdb`

✅ 4. Create the `.env` File

Where to create it?

In your project root folder:

my-project/

│── app/

│ └── main.py

│── .env <-- HERE

│── requirements.txt

Name must be exactly:

.env

> Not `.env.txt` or `env.txt` — only `.env`.

✅ 5. What to Put Inside `.env` for Windows

Write this **exact format** (replace the password):

DATABASE_URL=postgresql://postgres:YOURPASSWORD@localhost:5432/myappdb

Example:

DATABASE_URL=postgresql://postgres:mysecretpassword@localhost:5432/myappdb

⚠️ **No quotes, no spaces, no extra characters.**

✅ 6. Install the Required Packages

Run:

pip install fastapi uvicorn psycopg2-binary python-dotenv sqlalchemy

---

✅ 7. Test the Connection

In your `main.py` (example):

```
from fastapi import FastAPI
from sqlalchemy import create_engine
from dotenv import load_dotenv
import os

load_dotenv()

DATABASE_URL = os.getenv("DATABASE_URL")
engine = create_engine(DATABASE_URL)

app = FastAPI()

@app.get("/")
def home():
    return {"status": "ok"}
```

🎉 You're Ready to Use PostgreSQL with FastAPI!

If you want, I can now help you:

✅ Generate your SQLAlchemy models
✅ Generate migrations (Alembic)
✅ Create CRUD functions
✅ Build an API endpoint that uses the database

"Let's continue Day 1 database setup"

🔹 STEP 13 — Create a Test Model

`app/models/user.py`

```
from sqlalchemy import Column, Integer, String

from ..database import Base


class User(Base):

    __tablename__ = "users"


    id = Column(Integer, primary_key=True, index=True)

    email = Column(String, unique=True, index=True)

    hashed_password = Column(String)
```

In `main.py`, add:

```
from app.database import Base, engine

from app.models.user import User
```

Base.metadata.create_all(bind=engine)

🔹 STEP 14 — Test Backend

Run again:

uvicorn app.main:app --reload

Visit:

http://localhost:8000

You should see:

{

"message": "FocusFlow API is running 🚀"

}

✓ Backend initialized
✓ Database connected
✓ First table created

🎉 DAY 1 IS COMPLETE:

FRONTEND

✓ React + TS + Vite
✓ TailwindCSS configured
✓ Pages created
✓ Routing working

BACKEND

✓ FastAPI running
✓ PostgreSQL connection configured
✓ Base models created
✓ API root endpoint working

This is a **perfect professional foundation**.

👌 NEXT STEP (Day 2)

We will build:

✓ Full database setup
✓ User model
✓ Task model
✓ Habit model
✓ Category model
✓ Database migrations (optional)
