**✅ CiliaSync: Personal Development Plan**

**🧭 Phase 1: Project Bootstrapping (Week 1)**

**🔹 1.1. Set Up Your Local Dev Environment**

* ✅ Install:
  + Python 3.10+
  + Node.js (v18+)
  + PostgreSQL (or SQLite for now)
  + .NET SDK (for C# CLI)
  + VS Code (or IDE of choice)
* ✅ Create virtual environments:

bash

CopyEdit

python -m venv venv

source venv/bin/activate

**🔹 1.2. Initialize Repos**

* 🗂 Create folder structure:

arduino

CopyEdit

ciliasync/

├── client/

├── backend/

├── image-core/

├── data/

├── notebooks/

* ✅ Initialize git repo & push to GitHub

**🌐 Phase 2: Frontend Setup (Week 1–2)**

**🔹 2.1. Setup React + Tailwind**

bash

CopyEdit

cd client

npx create-react-app .

npm install -D tailwindcss postcss autoprefixer

npx tailwindcss init -p

**🔹 2.2. Build UI Components**

* Image Upload component
* Switch/tabs: “Cilia Count” / “Co-localization”
* Result card with:
  + Uploaded image preview
  + Analysis result text/stats
  + Placeholder chart

✅ Use dummy JSON from backend for now.

**🔬 Phase 3: Backend Setup (Week 2–3)**

**🔹 3.1. Django Backend (Main API, DB)**

* Init Django project: django-admin startproject django\_core
* Add analysis app
* Set up PostgreSQL connection
* Create Django REST endpoints:
  + /api/upload
  + /api/results
* Connect to database & test with dummy image stats

**🔹 3.2. FastAPI ML Subservice**

* Init FastAPI project in fastapi\_core/
* Create /predict endpoint
* Set up ASGI to run Django + FastAPI together

**🖼 Phase 4: Core Image Analysis (Week 4–5)**

**🔹 4.1. Cilia Detection (Python)**

* Use OpenCV:
  + Convert to grayscale
  + Thresholding + dilation/erosion
  + Blob detection
* Output:
  + Image with bounding boxes
  + Cilia count
* Save processed image to /media

**🔹 4.2. Co-localization (Classic)**

* Handle dual-channel RGB input
* Define membrane zone (1–2 px edge)
* Measure:
  + Pearson correlation
  + Jaccard index
  + % overlap
* Return overlay image + metrics

**💡 Phase 5: PyTorch ML Module (Week 6–7)**

**🔹 5.1. Prepare Training Data**

* Label a small image set:
  + With pixel-wise masks or binary classification for overlap
* Store under /data/train

**🔹 5.2. Train Initial Model**

* Write train.py:
  + CNN or UNet
  + DataLoader + augmentations
* Train for 10–20 epochs
* Save best .pt model

**🔹 5.3. Inference API (FastAPI)**

* Accept image, run model
* Output prediction map + metrics
* Return overlay + heatmap URL

**⚙️ Phase 6: C# Integration (Week 7–8)**

**🔹 6.1. Build CLI Tool**

* Parse CLI args (input path, output JSON path)
* Run edge detection / segmentation
* Save JSON:

json

CopyEdit

{

"membrane\_area": 1203,

"coloc\_percent": 0.84

}

**🔹 6.2. Integrate with Python Backend**

* Call via subprocess.run()
* Read and merge JSON results
* Optionally log timings for comparison (Python vs C#)

**📊 Phase 7: Visualization Dashboard (Week 9)**

**🔹 7.1. Connect DB to Frontend**

* Use Axios to fetch data
* Filter by:
  + Tags
  + Condition
  + Date range

**🔹 7.2. Add Charts**

* 📈 Cilia count per condition
* 📉 Membrane vs Cytoplasm colocalization
* 📊 Scatterplot: Coloc vs other variables

**🚀 Phase 8: Deployment (Week 10)**

**🔹 8.1. Deploy Frontend**

* Host on Netlify:
  + Connect GitHub repo
  + Enable continuous deployment

**🔹 8.2. Deploy Backend**

* Render (or Railway) for Django + FastAPI
* Supabase or Render PostgreSQL instance

**🔹 8.3. C# Deployment**

* Upload compiled CLI binary to backend host
* Ensure permissions for subprocess execution

**🧪 Bonus: Testing & ML Improvements (Optional Phase)**

* Add unit tests for:
  + Cilia detection
  + ML model output
* Improve:
  + Training with more images
  + Add dropout/regularization
* Later: Add auto-labeling or annotation UI

**🧠 Study Along the Way**

**Topics to Learn**

* Django REST Framework basics
* FastAPI with PyTorch
* OpenCV image morphology
* C# CLI basics (.NET Core)
* PyTorch training workflows
* Docker (optional for deployment later)