**🧱 1. Tạo repository**

Tại <https://github.com/new>

**Phần 1 — General**

* **Owner:** chọn tài khoản GitHub của bạn (hoặc tổ chức nếu có).
* **Repository name:**  
  👉 FamilyBudget *(ngắn gọn, chuyên nghiệp)*
* **Description (khuyến khích thêm):**
* FamilyBudget – Ứng dụng quản lý chi tiêu gia đình và gợi ý tối ưu ngân sách cá nhân.

**⚙️ 2. Configuration**

**Visibility:**

Chọn:

* 🔒 **Private** nếu bạn chưa muốn người khác xem.
* 🌍 **Public** nếu bạn muốn chia sẻ hoặc làm portfolio.

💡 Gợi ý: chọn **Public** nếu đây là đồ án hoặc bạn muốn ghi điểm về khả năng dev chuyên nghiệp.

**Add README: ✅ Bật ON**

Vì bạn sẽ mô tả dự án, hướng dẫn cài đặt và mục tiêu ban đầu.

**Add .gitignore: ✅ Bật ON**

Chọn template phù hợp:

* Nếu bạn dùng **Node.js + Firebase/React/React Native**, chọn:  
  👉 Node
* Nếu bạn dùng **Java Spring Boot**, chọn:  
  👉 Java
* Nếu bạn định làm **đa nền tảng (web + mobile)** → chọn Node là an toàn nhất.

**Add license: ⚖️**

Nếu chỉ là đồ án cá nhân → có thể để **None**.  
Nếu bạn muốn người khác dùng mã nguồn:

* MIT License (phổ biến nhất)
* GPLv3 (nếu muốn yêu cầu mã nguồn mở theo chuỗi)

**📂 3. Clone về máy và tạo cấu trúc thư mục**

Sau khi tạo repo:

git clone https://github.com/<yourname>/FamilyBudget.git

cd FamilyBudget

Tạo cấu trúc thư mục cơ bản:

mkdir api-server mobile-app web-admin docs

touch README.md docker-compose.yml

**🌿 4. Tạo nhánh chuẩn**

Tạo các branch chính để làm việc theo mô hình Git Flow:

git checkout -b develop

git push -u origin develop

git branch feature/database

git branch feature/api

git branch feature/mobile

git branch feature/web

git push -u origin feature/database

git push -u origin feature/api

git push -u origin feature/mobile

git push -u origin feature/web

**📝 5. Soạn README.md ban đầu**

Ví dụ nội dung ban đầu:

# 🌿 FamilyBudget

Ứng dụng quản lý chi tiêu gia đình và gợi ý tối ưu ngân sách cá nhân.

## 🚀 Cấu trúc dự án

FamilyBudget/

├── api-server/ # Backend (Node.js/Express hoặc Firebase Functions)

├── web-admin/ # Web dashboard (React)

├── mobile-app/ # Mobile app (React Native hoặc Flutter)

├── docs/ # Tài liệu thiết kế, mô tả yêu cầu

└── docker-compose.yml

## 🔧 Cài đặt ban đầu

```bash

git clone https://github.com/<yourname>/FamilyBudget.git

cd FamilyBudget

npm install

**🧩 Nhánh chính**

* main – phiên bản ổn định
* develop – nhánh phát triển
* feature/database – cấu trúc cơ sở dữ liệu
* feature/api – REST API / Firebase Functions
* feature/mobile – ứng dụng di động
* feature/web – giao diện quản trị web

**📦 Docker (tùy chọn)**

Khởi chạy môi trường phát triển:

docker-compose up --build

**🧩 Mục tiêu:**

* Một **API server** (Node.js hoặc Express)
* Một **database** (MongoDB hoặc PostgreSQL tuỳ chọn)
* Một **web client** (React hoặc Next.js)
* Dễ mở rộng sau này (mobile app không cần docker)

**✅ Ví dụ file docker-compose.yml (cấu trúc cơ bản, chạy được ngay)**

Tạo (hoặc mở) file:

FamilyBudget/docker-compose.yml

Thêm nội dung sau:

version: '3.9'

services:

api-server:

build: ./api-server

container\_name: familybudget\_api

restart: always

ports:

- "5000:5000"

environment:

- NODE\_ENV=development

- DB\_HOST=db

- DB\_USER=admin

- DB\_PASS=admin123

depends\_on:

- db

volumes:

- ./api-server:/usr/src/app

web-admin:

build: ./web-admin

container\_name: familybudget\_web

restart: always

ports:

- "3000:3000"

depends\_on:

- api-server

volumes:

- ./web-admin:/usr/src/app

db:

image: mongo:6

container\_name: familybudget\_db

restart: always

ports:

- "27017:27017"

volumes:

- db\_data:/data/db

environment:

- MONGO\_INITDB\_ROOT\_USERNAME=admin

- MONGO\_INITDB\_ROOT\_PASSWORD=admin123

volumes:

db\_data:

**📁 Sau đó tạo khung thư mục:**

mkdir api-server web-admin

echo "FROM node:20\nWORKDIR /usr/src/app\nCOPY . .\nRUN npm install\nCMD [\"npm\", \"start\"]" > api-server/Dockerfile

echo "FROM node:20\nWORKDIR /usr/src/app\nCOPY . .\nRUN npm install\nCMD [\"npm\", \"start\"]" > web-admin/Dockerfile

**▶️ Cuối cùng, chạy lại:**

docker-compose up --build

Sau vài phút, bạn sẽ thấy container:

* familybudget\_api chạy ở http://localhost:5000
* familybudget\_web chạy ở http://localhost:3000
* familybudget\_db (MongoDB) ở localhost:27017
* .

S

**🧱 1️⃣ Tạo thư mục đúng cách trong PowerShell**

Trong PowerShell, lệnh tương đương mkdir là **New-Item** hoặc New-Item -ItemType Directory, nhưng bạn vẫn có thể dùng mkdir nếu viết đúng cú pháp:

mkdir api-server, web-admin

⚠️ Dấu **phẩy ( , )** trong PowerShell cho phép tạo nhiều thư mục một lúc.

**🧩 2️⃣ Tạo Dockerfile đúng cú pháp cho PowerShell**

Lệnh echo trong PowerShell không hiểu ký tự \n như trên Linux.  
Bạn nên dùng **heredoc** hoặc ghi file trực tiếp bằng @""@.

**Lệnh thay thế đúng:**

@"

FROM node:20

WORKDIR /usr/src/app

COPY . .

RUN npm install

CMD ["npm", "start"]

"@ | Out-File -Encoding UTF8 api-server/Dockerfile

@"

FROM node:20

WORKDIR /usr/src/app

COPY . .

RUN npm install

CMD ["npm", "start"]

"@ | Out-File -Encoding UTF8 web-admin/Dockerfile

**🧩 3️⃣ Kiểm tra lại docker-compose.yml**

Thông báo empty compose file nghĩa là file của bạn trống.  
→ Hãy mở file docker-compose.yml trong VS Code hoặc Notepad và dán nội dung sau:

version: '3.9'

services:

api-server:

build: ./api-server

container\_name: familybudget\_api

restart: always

ports:

- "5000:5000"

environment:

- NODE\_ENV=development

- DB\_HOST=db

- DB\_USER=admin

- DB\_PASS=admin123

depends\_on:

- db

volumes:

- ./api-server:/usr/src/app

web-admin:

build: ./web-admin

container\_name: familybudget\_web

restart: always

ports:

- "3000:3000"

depends\_on:

- api-server

volumes:

- ./web-admin:/usr/src/app

db:

image: mongo:6

container\_name: familybudget\_db

restart: always

ports:

- "27017:27017"

volumes:

- db\_data:/data/db

environment:

- MONGO\_INITDB\_ROOT\_USERNAME=admin

- MONGO\_INITDB\_ROOT\_PASSWORD=admin123

volumes:

db\_data:

Lưu lại file rồi chạy:

docker-compose up --build

**✅ Nếu tất cả đúng:**

Bạn sẽ thấy log tương tự:

[+] Building 2.5s (10/10)

=> => naming to docker.io/library/familybudget\_api

familybudget\_db | {"t":{"$date":"..."},"s":"I",...}

familybudget\_web | > npm start

✅ Hoàn hảo — thông báo này không phải lỗi nghiêm trọng đâu nhé.  
Nó chỉ nói rằng hai thư mục **api-server** và **web-admin** **đã tồn tại**, nên PowerShell **không tạo lại nữa**.

Nói cách khác:  
👉 Bạn **đã có sẵn** hai thư mục này rồi, không cần tạo lại nữa.

**🔍 Kiểm tra nhanh:**

Gõ lệnh sau để xác nhận:

Get-ChildItem

Bạn sẽ thấy danh sách như:

api-server

web-admin

docker-compose.yml

README.md

**✅ Bước tiếp theo:**

Bây giờ chỉ cần đảm bảo bạn đã có **Dockerfile** trong 2 thư mục này và **docker-compose.yml** ở gốc, ví dụ:

FamilyBudget/

├── api-server/

│ └── Dockerfile

├── web-admin/

│ └── Dockerfile

└── docker-compose.yml

**⚙️ Nếu Dockerfile chưa có, chạy lại lệnh này (copy & paste nguyên khối):**

@"

FROM node:20

WORKDIR /usr/src/app

COPY . .

RUN npm install

CMD ["npm", "start"]

"@ | Out-File -Encoding UTF8 api-server/Dockerfile

@"

FROM node:20

WORKDIR /usr/src/app

COPY . .

RUN npm install

CMD ["npm", "start"]

"@ | Out-File -Encoding UTF8 web-admin/Dockerfile

**🧩 Sau đó, chạy lại:**

docker-compose up --build

Nếu vẫn còn báo "empty compose file" → nghĩa là docker-compose.yml đang **trống hoặc sai đường dẫn**.  
Hãy kiểm tra bằng:

notepad docker-compose.yml

và xác nhận rằng bạn **đã dán nội dung YAML đầy đủ** (mình gửi ở tin nhắn trước).

**✅ 1️⃣ Kiểm tra nội dung file docker-compose.yml**

Trong PowerShell, chạy:

notepad docker-compose.yml

Sau đó đảm bảo nội dung **đúng như sau (đầy đủ, không trống):**

version: '3.9'

services:

api-server:

build: ./api-server

container\_name: familybudget\_api

restart: always

ports:

- "5000:5000"

environment:

- NODE\_ENV=development

- DB\_HOST=db

- DB\_USER=admin

- DB\_PASS=admin123

depends\_on:

- db

volumes:

- ./api-server:/usr/src/app

web-admin:

build: ./web-admin

container\_name: familybudget\_web

restart: always

ports:

- "3000:3000"

depends\_on:

- api-server

volumes:

- ./web-admin:/usr/src/app

db:

image: mongo:6

container\_name: familybudget\_db

restart: always

ports:

- "27017:27017"

volumes:

- db\_data:/data/db

environment:

- MONGO\_INITDB\_ROOT\_USERNAME=admin

- MONGO\_INITDB\_ROOT\_PASSWORD=admin123

volumes:

db\_data:

⚠️ **Lưu ý:** YAML rất nhạy cảm với **thụt dòng (indent)** và **tab** → chỉ dùng **space**, không dùng tab.

**✅ 2️⃣ Kiểm tra Dockerfile**

Trong hai thư mục api-server và web-admin, mỗi nơi phải có **file Dockerfile**.

Chạy:

Get-ChildItem api-server, web-admin

Nếu bạn thấy:

Dockerfile

→ là ổn.  
Nếu **chưa có**, chạy lại đoạn sau để tạo đúng cú pháp PowerShell:

@"

FROM node:20

WORKDIR /usr/src/app

COPY . .

RUN npm install

CMD ["npm", "start"]

"@ | Out-File -Encoding UTF8 api-server/Dockerfile

@"

FROM node:20

WORKDIR /usr/src/app

COPY . .

RUN npm install

CMD ["npm", "start"]

"@ | Out-File -Encoding UTF8 web-admin/Dockerfile

**✅ 3️⃣ Cuối cùng, chạy Docker Compose**

docker-compose up --build

Nếu mọi thứ đúng, bạn sẽ thấy:

[+] Building api-server

[+] Building web-admin

familybudget\_db | {"t":{"$date":"..."}, "msg":"Waiting for connections"}

Nếu vẫn báo "empty compose file", thử chạy:

docker-compose -f .\docker-compose.yml up --build

→ để chắc chắn Docker biết file nằm ở đúng vị trí.

**⚠️ Giải thích ngắn gọn lỗi**

error during connect: Get "http://.../dockerDesktopLinuxEngine":

The system cannot find the file specified.

→ Nghĩa là: Docker CLI (dòng lệnh) **không tìm thấy Docker Engine** đang chạy.  
Thường xảy ra khi:

* Docker Desktop **chưa được khởi động**
* Docker service **bị tắt hoặc lỗi kết nối WSL2**
* Hoặc bạn đang ở PowerShell mà Docker chưa được bật trong session hiện tại

**✅ Cách khắc phục nhanh**

**1️⃣ Mở Docker Desktop**

* Nhấn **Start → tìm “Docker Desktop” → mở lên**
* Chờ vài giây đến khi thấy dòng “Docker is running” (biểu tượng cá voi 🐳 màu xanh trong khay hệ thống).

👉 Sau đó chạy lại lệnh:

docker-compose up --build

**2️⃣ Kiểm tra Docker có chạy chưa**

Chạy lệnh:

docker info

Nếu bạn thấy lỗi kiểu "Cannot connect to the Docker daemon"  
→ Docker Desktop **vẫn chưa chạy**.

**3️⃣ Nếu đã mở Docker Desktop mà vẫn lỗi**

Chạy lệnh khởi động lại service Docker:

Restart-Service com.docker.service

rồi chạy lại:

docker-compose up --build

**4️⃣ Cập nhật cảnh báo nhỏ (version bị obsolete)**

Cảnh báo này không lỗi, chỉ báo rằng khóa version: không còn cần thiết.  
Bạn có thể **xóa dòng đầu tiên** trong docker-compose.yml:

version: '3.9'

→ giữ nguyên phần services: là đủ.

**✅ Sau khi Docker chạy ổn, bạn sẽ thấy log như:**

[+] Building 5.6s (10/10)

=> => naming to docker.io/library/familybudget\_api

familybudget\_db | {"t":{"$date":"..."}, "msg":"Waiting for connections"}

familybudget\_web | > npm start

Nếu bạn mở Docker Desktop rồi mà **chạy lại vẫn lỗi**, gửi giúp mình output của lệnh này nhé:

docker info

→ Mình sẽ chẩn đoán xem Docker daemon đang lỗi ở đâu (Windows service, WSL2, hay quyền truy cập).

**💡 Giải thích kỹ hơn:**

**🧱 1️⃣ Vì sao api-server và web-admin có Dockerfile**

Hai thành phần này chạy trên **môi trường server hoặc web**, nên Docker hóa giúp:

* Dễ triển khai (deploy) trên cloud
* Dễ chạy kèm database
* Dễ test môi trường backend và frontend web đồng thời

**📱 2️⃣ Nhưng mobile-app khác:**

Ứng dụng di động:

* Cần **Android SDK / Xcode / Metro bundler / Emulator**
* Các công cụ này **nặng và phụ thuộc hệ điều hành**, nên Docker không phù hợp để build app di động.

Do đó, thư mục mobile-app thường được chạy trực tiếp bằng Node/npm hoặc Android Studio chứ **không cần Dockerfile**.

**✅ 3️⃣ Khi nào có thể thêm Dockerfile cho mobile-app**

Nếu bạn **thật sự muốn chạy thử trong container**, có thể tạo một Dockerfile “phát triển” (developer image) — ví dụ với React Native (Expo):

# mobile-app/Dockerfile

FROM node:20

# Cài đặt Yarn và Expo CLI

RUN npm install -g yarn expo-cli

WORKDIR /usr/src/app

COPY . .

# Cài dependencies

RUN yarn install

# Cổng cho Metro bundler

EXPOSE 19000 19001 19002

CMD ["yarn", "start"]

⚠️ Nhưng Docker này **không build được app Android/iOS** – chỉ dùng để chạy Expo Dev Server cho frontend mobile.

**📂 Tóm lại: cấu trúc của bạn đang hợp lý**

FamilyBudget/

├── api-server/ # Có Dockerfile

├── web-admin/ # Có Dockerfile

├── mobile-app/ # Không cần Dockerfile

├── docs/

└── docker-compose.yml

Rất tốt — log này nói rõ lỗi đang nằm ở **bước build của web-admin**.  
Cụ thể:

npm error enoent Could not read package.json

Error: ENOENT: no such file or directory, open '/usr/src/app/package.json'

👉 **Nói cách khác:**  
Khi Docker chạy RUN npm install trong container, **không tìm thấy file package.json** trong thư mục /usr/src/app.  
Điều đó có nghĩa là **bạn chưa copy mã nguồn đúng cách vào image**.

**🔍 Nguyên nhân phổ biến nhất**

Trong Dockerfile của web-admin, bạn **chỉ định COPY . .**, nhưng **context build** trong docker-compose.yml lại **chưa trỏ đúng thư mục** (hoặc trỏ sai thư mục rỗng).

Ví dụ:  
Nếu bạn có cấu trúc:

FamilyBudget/

│

├─ web-admin/

│ ├─ package.json

│ ├─ src/

│ └─ Dockerfile

Thì trong docker-compose.yml, phần build phải **trỏ đến thư mục chứa Dockerfile và mã nguồn** như sau:

services:

web-admin:

build:

context: ./web-admin

dockerfile: Dockerfile

ports:

- "3000:3000"

Nếu bạn chỉ để:

build: .

thì Docker chỉ copy **toàn bộ thư mục FamilyBudget**, nhưng **bên trong đó không có file package.json** tương ứng với WORKDIR /usr/src/app.

**✅ Cách khắc phục**

**1️⃣ Mở Dockerfile của web-admin**  
Đảm bảo nó có nội dung kiểu như sau:

FROM node:20

WORKDIR /usr/src/app

COPY package\*.json ./

RUN npm install

COPY . .

CMD ["npm", "run", "dev"]

**2️⃣ Sửa docker-compose.yml phần web-admin:**

services:

web-admin:

build:

context: ./web-admin

dockerfile: Dockerfile

ports:

- "3000:3000"

**3️⃣ Chạy lại:**

docker-compose up --build

Nếu bạn gửi mình xem nội dung hiện tại của:

* docker-compose.yml
* web-admin/Dockerfile

c

Câu hỏi của bạn rất hay 👏 — bạn đang muốn thiết kế **Docker setup chuẩn cho dự án FamilyBudget** gồm nhiều thành phần:

* **API Server (Node.js + Express)**
* **Web Admin (ReactJS)**
* **Mobile App (React Native / Expo)**
* **Database (MongoDB)**

Dưới đây là hướng dẫn chi tiết cách tạo **cấu trúc Docker chuyên nghiệp, dễ mở rộng, phù hợp môi trường dev + deploy thực tế** 👇

**🧱 1. Cấu trúc thư mục chuẩn**

FamilyBudget/

│

├─ api-server/

│ ├─ src/

│ ├─ package.json

│ ├─ Dockerfile

│ └─ .dockerignore

│

├─ web-admin/

│ ├─ src/

│ ├─ package.json

│ ├─ Dockerfile

│ └─ .dockerignore

│

├─ mobile-app/ ← Không cần Docker build (Expo chạy riêng)

│ ├─ App.js

│ ├─ package.json

│ └─ ...

│

├─ docker-compose.yml

└─ README.md

**🐳 2. Dockerfile cho từng service**

**🧩 api-server/Dockerfile**

FROM node:20

WORKDIR /usr/src/app

COPY package\*.json ./

RUN npm install

COPY . .

EXPOSE 5000

CMD ["npm", "run", "start"]

**🌐 web-admin/Dockerfile**

FROM node:20 AS build

WORKDIR /usr/src/app

COPY package\*.json ./

RUN npm install

COPY . .

RUN npm run build

# Dùng nginx để serve build React

FROM nginx:alpine

COPY --from=build /usr/src/app/dist /usr/share/nginx/html

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]

⚠️ Nếu bạn dùng create-react-app, thư mục build là build/  
Nếu dùng vite, là dist/.  
Hãy chỉnh lại cho đúng framework.

**⚙️ 3. docker-compose.yml hoàn chỉnh**

version: "3.9"

services:

api-server:

build:

context: ./api-server

container\_name: familybudget\_api

restart: always

ports:

- "5000:5000"

environment:

- NODE\_ENV=development

- DB\_HOST=db

- DB\_USER=admin

- DB\_PASS=admin123

depends\_on:

- db

volumes:

- ./api-server:/usr/src/app

- /usr/src/app/node\_modules

web-admin:

build:

context: ./web-admin

container\_name: familybudget\_web

restart: always

ports:

- "3000:80"

depends\_on:

- api-server

db:

image: mongo:6

container\_name: familybudget\_db

restart: always

ports:

- "27017:27017"

volumes:

- db\_data:/data/db

environment:

- MONGO\_INITDB\_ROOT\_USERNAME=admin

- MONGO\_INITDB\_ROOT\_PASSWORD=admin123

volumes:

db\_data:

**📱 4. Mobile App (React Native / Expo)**

Phần này **không cần Docker**, vì Expo CLI cần môi trường thiết bị / emulator.  
Bạn chỉ cần:

cd mobile-app

npm start

và kết nối với api-server qua địa chỉ IP của máy chủ Docker, ví dụ:

http://192.168.1.10:5000

**🚀 5. Quy trình chạy**

# 1. Khởi động toàn bộ môi trường backend + web

docker-compose up --build

# 2. Kiểm tra container

docker ps

# 3. Truy cập

API server: http://localhost:5000

Web Admin: http://localhost:3000

**💡 6. Gợi ý mở rộng sau này**

| **Mục tiêu** | **Gợi ý** |
| --- | --- |
| CI/CD | Dùng **GitHub Actions** để build image và deploy lên Firebase Hosting / Render / Railway |
| Env riêng cho dev/prod | Tạo file .env.dev và .env.prod, rồi dùng env\_file: trong Compose |
| Cache npm nhanh hơn | Dùng npm ci thay npm install trong Dockerfile |
| Theo dõi log | Dùng docker logs -f familybudget\_api |

Nếu bạn gửi mình:

* Framework bạn dùng cho **web-admin** (CRA, Vite, hay Next.js)
* Framework bạn dùng cho **api-server** (Express, NestJS, hay Fastify)

→ Mình có thể tinh chỉnh Dockerfile tối ưu hơn (build nhỏ hơn, cache tốt hơn, khởi động nhanh hơn).  
Bạn có muốn mình làm giúp phần đó luôn không?

Rất tốt — bạn đã **chạy được Docker Compose**, nhưng lỗi bạn gặp là:

npm error enoent Could not read package.json: Error: ENOENT: no such file or directory, open '/usr/src/app/package.json'

👉 Nghĩa là:  
Docker **không tìm thấy file package.json trong thư mục web-admin/** khi thực hiện lệnh npm install.

**✅ Giải thích ngắn gọn**

Khi bạn có dòng này trong docker-compose.yml:

web-admin:

build:

context: ./web-admin

thì Docker sẽ:

1. Tìm Dockerfile trong **./web-admin**
2. Sao chép nội dung trong thư mục đó (COPY . .) vào /usr/src/app trong container
3. Sau đó chạy npm install

Vì vậy, nếu **không có file package.json trong ./web-admin**, lệnh npm install sẽ lỗi như bạn thấy.

**🧭 Cách khắc phục**

**🔹 Bước 1. Kiểm tra thư mục web-admin**

Chạy lệnh này trong PowerShell:

dir web-admin

Nếu bạn **không thấy file package.json**, thì đó chính là nguyên nhân.

**🔹 Bước 2. Tạo hoặc khởi tạo dự án ReactJS**

Bạn cần khởi tạo frontend trong thư mục web-admin trước khi build Docker.

Chạy một trong các lệnh sau tùy công nghệ bạn dùng:

**👉 Nếu dùng Vite (nhanh, gọn)**

cd web-admin

npm create vite@latest .

# chọn React, TypeScript hoặc JavaScript tuỳ ý

npm install

cd ..

**👉 Nếu dùng Create React App**

cd web-admin

npx create-react-app .

cd ..

Sau bước này, thư mục web-admin/ sẽ có:

* package.json
* src/, public/
* vite.config.js hoặc README.md

**🔹 Bước 3. Kiểm tra lại Dockerfile trong web-admin**

Nên có nội dung như sau (cho Vite hoặc CRA):

FROM node:20 AS build

WORKDIR /usr/src/app

COPY package\*.json ./

RUN npm install

COPY . .

RUN npm run build

FROM nginx:alpine

COPY --from=build /usr/src/app/dist /usr/share/nginx/html

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]

⚠️ Nếu dùng Create React App thì đổi dist thành build:

COPY --from=build /usr/src/app/build /usr/share/nginx/html

**🔹 Bước 4. Build lại**

Quay về thư mục chính của dự án rồi chạy:

docker-compose down -v

docker-compose up --build

**🧩 Tổng kết**

| **Bước** | **Mục tiêu** | **Trạng thái** |
| --- | --- | --- |
| 1️⃣ | Kiểm tra web-admin có package.json chưa | ❌ (chưa có) |
| 2️⃣ | Tạo dự án ReactJS (Vite hoặc CRA) | ✅ |
| 3️⃣ | Kiểm tra Dockerfile đúng cấu trúc | ✅ |
| 4️⃣ | Build lại bằng docker-compose up --build | ✅ |

Nếu bạn gửi mình đoạn Dockerfile hiện tại trong web-admin/  
và cho biết bạn dùng **React + Vite** hay **Create React App**,  
mình sẽ **viết lại Dockerfile chính xác 100%** cho bạn (chạy được ngay).  
Bạn muốn dùng loại nào để mình soạn cho đúng?

Tốt lắm 👍 — nhìn vào ảnh thì mình thấy thư mục web-admin/ của bạn **đã đầy đủ** để chạy React + Vite (có package.json, vite.config.js, src, public, v.v.).  
Giờ ta chỉ cần **sửa lại Dockerfile** cho đúng kiểu Vite là được.

**✅ Bước 1: Sửa Dockerfile trong web-admin**

Mở file web-admin/Dockerfile và thay toàn bộ nội dung bằng:

# --- Giai đoạn build ---

FROM node:20 AS build

WORKDIR /usr/src/app

# Copy file cấu hình trước để tận dụng cache

COPY package\*.json ./

RUN npm install

# Copy toàn bộ source code vào

COPY . .

# Build project

RUN npm run build

# --- Giai đoạn chạy (production) ---

FROM nginx:alpine

# Copy file build ra thư mục web mặc định của nginx

COPY --from=build /usr/src/app/dist /usr/share/nginx/html

# Expose port 80

EXPOSE 80

# Khởi động nginx

CMD ["nginx", "-g", "daemon off;"]

**✅ Bước 2: Kiểm tra lại docker-compose.yml ở thư mục gốc**

Mở file docker-compose.yml và đảm bảo có phần như sau:

version: "3.9"

services:

web-admin:

build:

context: ./web-admin

ports:

- "5173:80" # 5173 là port ngoài, 80 là port nginx trong container

container\_name: familybudget-web-admin

restart: always

api-server:

build:

context: ./api-server

ports:

- "5000:5000"

container\_name: familybudget-api

restart: always

**✅ Bước 3: Build lại từ đầu**

Trong PowerShell, tại thư mục D:\DoAN\PJ\FamilyBudget, chạy:

docker-compose down -v

docker-compose up --build

**✅ Bước 4: Mở trình duyệt**

Khi build xong, mở trình duyệt truy cập:

👉 <http://localhost:5173>

Nếu hiện trang React của bạn (ví dụ “Vite + React” hay giao diện bạn đang làm), là thành công 🎉

**⚠️ Nếu build vẫn lỗi**

Hãy gửi mình phần **log lỗi trong PowerShell** hoặc **nội dung docker-compose.yml** bạn đang dùng để mình kiểm tra chi tiết giúp.

Bạn có muốn mình giúp luôn phần api-server/Dockerfile chuẩn (Node.js + Express + auto-reload dev) cho đồng bộ với frontend không?

Tốt lắm 👍 — log bạn gửi rất rõ.  
Lỗi chính là:

npm error enoent Could not read package.json

👉 Nghĩa là: trong thư mục api-server/ **không có file package.json**, nên Docker không thể chạy npm install.

**✅ Giải thích nhanh**

Dòng trong Dockerfile của bạn:

WORKDIR /usr/src/app

COPY . .

RUN npm install

Docker đang copy toàn bộ thư mục api-server (theo context: ./api-server trong docker-compose),  
nhưng bên trong đó lại **thiếu file package.json**, nên npm install không thể chạy.

**✅ Cách kiểm tra & sửa**

**🧩 Bước 1: Kiểm tra lại cấu trúc thư mục**

Trong VS Code, mở thư mục api-server và xem có không:

api-server/

├── Dockerfile

├── package.json ❌ ← hiện tại đang thiếu file này

├── package-lock.json

├── src/

├── server.js (hoặc app.js)

Nếu chưa có package.json, thì bạn cần tạo bằng cách chạy lệnh này trong thư mục api-server:

cd api-server

npm init -y

**🧩 Bước 2: Cài các thư viện cần thiết (ví dụ nếu bạn dùng Express)**

npm install express cors mongoose dotenv

Sau đó file package.json sẽ trông như sau (ví dụ):

{

"name": "api-server",

"version": "1.0.0",

"main": "server.js",

"type": "module",

"scripts": {

"start": "node server.js"

},

"dependencies": {

"cors": "^2.8.5",

"dotenv": "^16.0.3",

"express": "^4.19.2",

"mongoose": "^8.5.0"

}

}

**🧩 Bước 3: Kiểm tra Dockerfile của api-server**

Hãy mở api-server/Dockerfile và đảm bảo nó có nội dung như sau:

FROM node:20

WORKDIR /usr/src/app

COPY package\*.json ./

RUN npm install

COPY . .

EXPOSE 5000

CMD ["npm", "start"]

**🧩 Bước 4: Build lại**

Tại thư mục gốc FamilyBudget, chạy lại:

docker-compose down -v

docker-compose up --build

Sau khi build thành công, bạn sẽ thấy:

✅ familybudget\_api container chạy (API server)  
✅ familybudget\_web\_admin container chạy (React admin site)

Bạn có muốn mình giúp tạo sẵn một **server.js mẫu** cho API backend (Express + MongoDB + dotenv) để bạn có thể chạy được luôn không?

Rất tốt 👌 — nếu bạn đang muốn cấu hình **Firebase cho api-server**, thì mình cần làm rõ thêm một chút để hướng dẫn đúng hướng:

Bạn đang định dùng **Firebase theo cách nào**?

**🔸 Có 2 trường hợp phổ biến:**

**1️⃣ Dùng Firebase như backend chính (no server needed)**

* Bạn không cần api-server Node.js.
* Firebase (Firestore, Authentication, Cloud Functions, v.v.) sẽ xử lý hết backend.
* App của bạn (web-admin, mobile-app) sẽ giao tiếp trực tiếp với Firebase qua SDK.

→ **Trường hợp này** bạn chỉ cần cấu hình Firebase trong web-admin và mobile-app (không cần Docker cho api-server).

**2️⃣ Dùng Firebase chỉ làm cơ sở dữ liệu hoặc auth provider (qua API server riêng)**

* Bạn có api-server Node.js làm trung gian (gateway/backend riêng).
* api-server dùng **Firebase Admin SDK** để truy cập Firestore hoặc Authentication.
* Docker build api-server sẽ vẫn cần package.json.

→ **Trường hợp này**, bạn cần:

1. Cài firebase-admin trong api-server
2. cd api-server
3. npm init -y
4. npm install firebase-admin dotenv express cors
5. Tạo file serviceAccountKey.json (Firebase Admin Key) trong api-server/
   * Lấy file này trong Firebase Console → Project Settings → Service accounts → Generate new private key
   * **⚠️ Không commit file này lên Git** (.gitignore phải có serviceAccountKey.json)
6. Tạo file server.js như sau:

import express from "express";

import cors from "cors";

import admin from "firebase-admin";

import dotenv from "dotenv";

dotenv.config();

const app = express();

app.use(cors());

app.use(express.json());

// Khởi tạo Firebase Admin SDK

admin.initializeApp({

credential: admin.credential.cert("./serviceAccountKey.json"),

});

const db = admin.firestore();

app.get("/test", async (req, res) => {

const snapshot = await db.collection("users").get();

const users = snapshot.docs.map((doc) => ({ id: doc.id, ...doc.data() }));

res.json(users);

});

const PORT = process.env.PORT || 5000;

app.listen(PORT, () => console.log(`Server running on port ${PORT}`));

1. Tạo Dockerfile trong api-server/:

FROM node:20

WORKDIR /usr/src/app

COPY package\*.json ./

RUN npm install

COPY . .

EXPOSE 5000

CMD ["npm", "start"]

1. Cập nhật package.json:

{

"name": "api-server",

"version": "1.0.0",

"main": "server.js",

"type": "module",

"scripts": {

"start": "node server.js"

},

"dependencies": {

"cors": "^2.8.5",

"dotenv": "^16.0.3",

"express": "^4.19.2",

"firebase-admin": "^12.0.0"

}

}

👉 Sau khi xong:

docker-compose down -v

docker-compose up --build

Server sẽ chạy và có thể truy cập Firebase Firestore qua REST API của bạn.

Để mình giúp chi tiết hơn:

Bạn chọn hướng nào —  
🔹 “Dùng Firebase trực tiếp (frontend-only)”  
hay 🔹 “Dùng Firebase qua API server riêng (Node.js)” ?

Câu hỏi rất hay 👏 — và **rất quan trọng** khi bắt đầu đồ án kiểu “quản lý chi tiêu + gợi ý ngân sách”.  
Mình sẽ giải thích **rõ – ngắn – thực tế** để bạn chọn hướng phù hợp.

**🔸 1️⃣ “Dùng Firebase trực tiếp (frontend-only)”**

**📌 Cấu trúc**

Không có backend riêng.  
👉 App web (web-admin) và app mobile (mobile-app) **kết nối thẳng đến Firebase** bằng **Firebase SDK**.

**⚙️ Cách hoạt động**

* Frontend đăng nhập người dùng bằng **Firebase Authentication** (Google / Email, v.v.)
* Dữ liệu (chi tiêu, thu nhập, nhóm chi phí, v.v.) lưu vào **Firestore** hoặc **Realtime Database**.
* Firebase Rule kiểm soát quyền truy cập.
* Logic như “tính tổng chi tiêu, thống kê tháng, gợi ý tiết kiệm” có thể viết:
  + trực tiếp trong frontend, hoặc
  + dùng **Firebase Cloud Functions** (viết bằng JS, chạy trên cloud – không cần server riêng).

**✅ Ưu điểm**

* **Cực nhanh để triển khai** (chỉ cần frontend và Firebase config).
* Không cần DevOps, không cần Docker cho backend.
* Firebase lo luôn:
  + Authentication
  + Database
  + Hosting
  + Security Rules

**❌ Nhược điểm**

* Khó xử lý logic phức tạp (ví dụ: tính toán gợi ý AI, tổng hợp nhiều bảng, xử lý dữ liệu nặng).
* Không có API REST tùy chỉnh để dùng chung (nếu sau này cần mở rộng).
* Bị **ràng buộc** vào hệ sinh thái Firebase.

**🔸 2️⃣ “Dùng Firebase qua API server riêng (Node.js)”**

**📌 Cấu trúc**

Có backend riêng (api-server) — nơi bạn kiểm soát toàn bộ logic và quyền truy cập.

👉 Kiến trúc:  
web-admin / mobile-app → gửi request → api-server (Node.js + Express) → Firebase (Firestore/Auth)

**⚙️ Cách hoạt động**

* Frontend gọi API như /api/income/add, /api/spending/list.
* Backend dùng **Firebase Admin SDK** để đọc/ghi dữ liệu Firestore.
* Logic tính toán (gợi ý ngân sách, AI, thống kê) được viết ở server.
* Có thể dễ dàng thêm chức năng sau này (ví dụ: gửi email, export báo cáo PDF).

**✅ Ưu điểm**

* **Kiểm soát tuyệt đối** luồng dữ liệu và bảo mật.
* Dễ mở rộng (thêm AI, chatbot, phân tích, tích hợp ngân hàng…)
* Không bị ràng buộc bởi frontend — bạn có thể làm mobile, web, API khác dùng chung.

**❌ Nhược điểm**

* Cần setup backend (Express, Docker, deploy…)
* Cần code thêm phần API.
* Tốn thêm thời gian ban đầu.

**🧭 Gợi ý chọn hướng**

| **Mục tiêu đồ án** | **Nên chọn hướng** |
| --- | --- |
| Bạn chỉ cần CRUD (thêm/sửa/xóa/hiển thị thu nhập, chi tiêu, thống kê biểu đồ) | ✅ **Frontend-only với Firebase SDK** |
| Bạn muốn làm đồ án chuyên sâu (có gợi ý ngân sách, machine learning, API riêng) | ✅ **Firebase + Node.js API server** |

👉 **Tóm gọn dễ nhớ:**

* **Frontend-only:** nhanh, dễ, phù hợp demo.
* **Có API server:** chuyên nghiệp, mạnh, mở rộng dễ.

Bạn muốn mình giúp **setup theo hướng nào**?

* 🔹 *Firebase frontend-only (web + mobile kết nối trực tiếp)*
* 🔹 *Firebase qua API server Node.js (backend trung gian)*

Tuyệt vời 🎯 — bạn chọn hướng **Firebase qua API server (Node.js backend)** là hướng **chuyên nghiệp** nhất, đúng chuẩn mô hình *Fullstack Cloud-Integrated*.  
Mình sẽ giúp bạn đi từng bước để cấu hình chuẩn.

**⚙️ 1️⃣ Cấu trúc tổng thể dự án (chuẩn cho Docker + Firebase)**

FamilyBudget/

│

├── api-server/ # Backend (Node.js + Express)

│ ├── Dockerfile

│ ├── package.json

│ ├── src/

│ │ ├── index.js # điểm vào chính (Express server)

│ │ ├── routes/

│ │ ├── controllers/

│ │ └── services/firebase.js # Firebase Admin SDK

│ └── .env # Firebase credentials + env config

│

├── web-admin/ # ReactJS dashboard (quản trị)

│ └── Dockerfile

│

├── mobile-app/ # React Native app người dùng

│

├── docker-compose.yml # quản lý service backend & web

└── README.md

**🔑 2️⃣ Tạo Firebase Project & Credentials**

1. Truy cập: https://console.firebase.google.com
2. **Tạo project mới** → ví dụ: familybudget
3. Trong phần **Project Settings** → tab **Service accounts** → chọn:
4. Generate new private key

→ tải file .json (đây là khóa truy cập Admin SDK)

1. Đặt file đó vào thư mục:
2. api-server/firebase-service-account.json

**⚙️ 3️⃣ Cài đặt Firebase Admin SDK cho API Server**

Trong thư mục api-server, tạo package.json nếu chưa có:

cd api-server

npm init -y

npm install express cors firebase-admin dotenv

**📄 4️⃣ File cấu hình backend**

**Dockerfile (trong api-server/)**

FROM node:20

WORKDIR /usr/src/app

COPY package\*.json ./

RUN npm install

COPY . .

EXPOSE 5000

CMD ["npm", "start"]

**.env**

PORT=5000

FIREBASE\_SERVICE\_ACCOUNT=./firebase-service-account.json

FIREBASE\_DATABASE\_URL=https://familybudget.firebaseio.com

**src/index.js**

import express from "express";

import cors from "cors";

import dotenv from "dotenv";

import admin from "firebase-admin";

import { readFileSync } from "fs";

dotenv.config();

const app = express();

app.use(cors());

app.use(express.json());

// 🔥 Khởi tạo Firebase Admin

const serviceAccount = JSON.parse(readFileSync(process.env.FIREBASE\_SERVICE\_ACCOUNT));

admin.initializeApp({

credential: admin.credential.cert(serviceAccount),

databaseURL: process.env.FIREBASE\_DATABASE\_URL,

});

const db = admin.firestore();

// 🧩 API thử nghiệm

app.get("/", (req, res) => {

res.json({ message: "FamilyBudget API Server is running 🚀" });

});

app.get("/users", async (req, res) => {

const users = await db.collection("users").get();

res.json(users.docs.map(doc => ({ id: doc.id, ...doc.data() })));

});

app.listen(process.env.PORT, () => {

console.log(`✅ API Server running at http://localhost:${process.env.PORT}`);

});

**🐳 5️⃣ docker-compose.yml (cập nhật lại)**

services:

api-server:

build: ./api-server

container\_name: familybudget\_api

restart: always

ports:

- "5000:5000"

volumes:

- ./api-server:/usr/src/app

environment:

- NODE\_ENV=development

- PORT=5000

- FIREBASE\_SERVICE\_ACCOUNT=/usr/src/app/firebase-service-account.json

- FIREBASE\_DATABASE\_URL=https://familybudget.firebaseio.com

web-admin:

build: ./web-admin

container\_name: familybudget\_web

restart: always

ports:

- "3000:3000"

depends\_on:

- api-server

**🧩 6️⃣ Kiểm tra nhanh**

Sau khi cấu hình xong:

docker-compose up --build

Nếu mọi thứ OK:

* http://localhost:5000/ → trả JSON { "message": "FamilyBudget API Server is running 🚀" }
* http://localhost:5000/users → trả danh sách user trong Firestore (nếu có)

**🔮 7️⃣ Bước tiếp theo (mình có thể hướng dẫn bạn chi tiết)**

| **Bước** | **Mục tiêu** |
| --- | --- |
| 🔹 Bước 1 | Kết nối API → Firestore (CRUD thu nhập / chi tiêu) |
| 🔹 Bước 2 | Tích hợp Firebase Auth (đăng nhập, token verify) |
| 🔹 Bước 3 | Kết nối web-admin → API qua axios |
| 🔹 Bước 4 | Mobile app sync dữ liệu Firestore qua API |
| 🔹 Bước 5 | Thêm gợi ý ngân sách / AI mini bằng Python hoặc rule |

👉 Bạn muốn mình hướng dẫn **bước kế tiếp** là:

* (A) Tạo CRUD API kết nối Firestore
* (B) Thiết lập xác thực Firebase Auth cho API (bảo vệ request)
* (C) Cấu hình phần web-admin kết nối API

Bạn chọn hướng nào tiếp theo?

✅ Hoàn hảo — bạn đã **khởi tạo thành công backend Node.js** và cài đầy đủ các thư viện cần thiết (express, cors, firebase-admin, dotenv) 🎯

Giờ bạn làm tiếp 3 bước ngắn sau để hoàn tất phần backend cơ bản nhé:

**🧩 Bước 1 — Tạo cấu trúc thư mục**

Trong thư mục api-server, tạo cấu trúc này:

api-server/

│

├── src/

│ └── index.js

│

├── .env

├── firebase-service-account.json ← (file tải từ Firebase console)

└── Dockerfile

👉 Lệnh nhanh (nếu bạn đang dùng PowerShell):

mkdir src

ni src\index.js -ItemType File

ni .env -ItemType File

ni Dockerfile -ItemType File

**⚙️ Bước 2 — Dán nội dung vào từng file**

**📄 .env**

PORT=5000

FIREBASE\_SERVICE\_ACCOUNT=./firebase-service-account.json

FIREBASE\_DATABASE\_URL=https://familybudget.firebaseio.com

**📄 src/index.js**

import express from "express";

import cors from "cors";

import dotenv from "dotenv";

import admin from "firebase-admin";

import { readFileSync } from "fs";

dotenv.config();

const app = express();

app.use(cors());

app.use(express.json());

// 🔥 Khởi tạo Firebase Admin

const serviceAccount = JSON.parse(readFileSync(process.env.FIREBASE\_SERVICE\_ACCOUNT));

admin.initializeApp({

credential: admin.credential.cert(serviceAccount),

databaseURL: process.env.FIREBASE\_DATABASE\_URL,

});

const db = admin.firestore();

// 🧩 API thử nghiệm

app.get("/", (req, res) => {

res.json({ message: "FamilyBudget API Server is running 🚀" });

});

app.get("/users", async (req, res) => {

const snapshot = await db.collection("users").get();

const users = snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() }));

res.json(users);

});

app.listen(process.env.PORT, () => {

console.log(`✅ API Server running at http://localhost:${process.env.PORT}`);

});

**📄 Dockerfile**

FROM node:20

WORKDIR /usr/src/app

COPY package\*.json ./

RUN npm install

COPY . .

EXPOSE 5000

CMD ["node", "src/index.js"]

**🚀 Bước 3 — Kiểm tra chạy thử**

Nếu bạn muốn **chạy thử trực tiếp (chưa cần Docker):**

node src/index.js

Kết quả mong đợi:

✅ API Server running at http://localhost:5000

Sau đó vào trình duyệt:

* http://localhost:5000/ → thấy JSON { "message": "FamilyBudget API Server is running 🚀" }
* http://localhost:5000/users → sẽ trả danh sách người dùng (nếu Firestore có sẵn)

Khi mọi thứ ổn rồi, bước tiếp theo là:

🔹 Thêm CRUD API (thu nhập / chi tiêu)  
🔹 hoặc 🔹 Thêm Firebase Auth bảo vệ API

👉 Bạn muốn mình hướng dẫn **tạo API CRUD đầu tiên (thu nhập / chi tiêu)** hay **xác thực người dùng (Firebase Auth)** trước?

**🧩 Nguyên nhân**

File này là **Service Account Key** — Firebase dùng nó để cho phép backend (server Node.js) truy cập Firestore, Auth, Storage, v.v.  
Hiện bạn **chưa tạo hoặc chưa đặt file đó** vào thư mục api-server.

**✅ Cách khắc phục từng bước**

**1️⃣ Tạo Service Account trên Firebase Console**

1. Vào Firebase Console
2. Chọn **project của bạn** (ví dụ: *FamilyBudget*)
3. Vào **⚙️ Project settings** (biểu tượng bánh răng bên cạnh “Project Overview”)
4. Chọn tab **Service accounts**
5. Bấm **"Generate new private key"**  
   → Firebase sẽ tải về file .json, ví dụ:
6. familybudget-firebase-adminsdk-xyz12.json
7. Đặt file này vào đúng thư mục backend:
8. D:\DoAN\PJ\FamilyBudget\api-server\firebase-service-account.json

**2️⃣ Đảm bảo file .env có đúng đường dẫn**

Mở .env trong api-server và kiểm tra:

PORT=5000

FIREBASE\_SERVICE\_ACCOUNT=./firebase-service-account.json

FIREBASE\_DATABASE\_URL=https://<your-project-id>.firebaseio.com

⚠️ Thay <your-project-id> bằng ID thật của dự án Firebase (xem ở Firebase console, góc trên).

**3️⃣ Chạy lại**

node src/index.js

Nếu cấu hình đúng, bạn sẽ thấy:

✅ API Server running at http://localhost:5000

**🔒 Lưu ý bảo mật**

* **Không đẩy file firebase-service-account.json lên GitHub.**  
  → Hãy thêm dòng này vào .gitignore:
* firebase-service-account.json
* .env

Bạn có muốn mình hướng dẫn **thêm Firebase Auth (xác thực người dùng)** vào server ngay sau khi server chạy được không?  
Điều này giúp đảm bảo chỉ người dùng đăng nhập mới được gọi API.

Tuyệt vời 👍 — bạn đã **đi đúng quy trình triển khai dự án Node.js + React bằng Docker Compose**.  
Mình sẽ **tổng hợp toàn bộ các lệnh** mà bạn đã thực hiện, theo **thứ tự thực tế**, kèm **ý nghĩa chi tiết từng lệnh**.

**🧩 1️⃣ Tạo cấu trúc dự án ban đầu**

PS D:\DoAN\PJ\FamilyBudget> mkdir api-server web-admin

* **Tạo hai thư mục con**:
  + api-server: chứa mã nguồn **server backend** (Node.js, Express, Firebase Admin).
  + web-admin: chứa mã nguồn **frontend admin panel** (React + Vite).
* Lệnh này giúp chia tách backend và frontend rõ ràng, dễ Docker hóa.

**🧩 2️⃣ Tạo Dockerfile cho từng phần**

echo "FROM node:20\nWORKDIR /usr/src/app\nCOPY . .\nRUN npm install\nCMD [\"npm\", \"start\"]" > api-server/Dockerfile

echo "FROM node:20\nWORKDIR /usr/src/app\nCOPY . .\nRUN npm install\nCMD [\"npm\", \"start\"]" > web-admin/Dockerfile

* **Tạo 2 Dockerfile** (một cho mỗi dịch vụ).
* Ý nghĩa từng dòng trong Dockerfile:

| **Dòng** | **Giải thích** |
| --- | --- |
| FROM node:20 | Dùng image Node.js 20 làm nền. |
| WORKDIR /usr/src/app | Đặt thư mục làm việc trong container. |
| COPY . . | Sao chép toàn bộ mã nguồn vào container. |
| RUN npm install | Cài đặt các package trong package.json. |
| CMD ["npm", "start"] | Khi container khởi chạy, tự động chạy npm start. |

**🧩 3️⃣ Tạo file cấu hình Docker Compose**

notepad docker-compose.yml

* Mở file docker-compose.yml để chỉnh sửa nội dung cấu hình nhiều container (web, api, db, v.v.) cùng lúc.
* **docker-compose** là công cụ giúp bạn build & run nhiều container chỉ bằng 1 file YAML.

**🧩 4️⃣ Chạy build và khởi động container**

docker-compose up --build

* Lệnh này **build lại image** và **khởi động toàn bộ dịch vụ** được mô tả trong docker-compose.yml.
* Thực tế bạn gặp lỗi:
* npm error enoent Could not read package.json

→ Do trong thư mục api-server chưa có file package.json.

**🧩 5️⃣ Kiểm tra container đang chạy**

docker ps

* Hiển thị danh sách các container đang hoạt động, gồm CONTAINER ID, IMAGE, PORTS, STATUS...
* Kết quả: **trống**, nghĩa là build lỗi nên container chưa start được.

**🧩 6️⃣ Sửa lỗi bằng cách khởi tạo Node.js cho api-server**

cd api-server

npm init -y

npm install express cors firebase-admin dotenv

* npm init -y → Tạo file package.json mặc định.
* npm install express cors firebase-admin dotenv → Cài các package backend:

| **Gói** | **Vai trò** |
| --- | --- |
| express | Framework tạo REST API. |
| cors | Cho phép frontend gọi API khác domain. |
| firebase-admin | Kết nối server với Firebase. |
| dotenv | Đọc biến môi trường từ file .env. |

**🧩 7️⃣ Tạo cấu trúc mã nguồn backend**

mkdir src

ni src\index.js -ItemType File

ni .env -ItemType File

* src/index.js → Tập tin khởi đầu cho API server.
* .env → Nơi lưu các biến môi trường (PORT, DB\_URL, FIREBASE\_KEY\_PATH, ...).
* ni là alias của New-Item trong PowerShell, dùng để tạo file mới.

**🧩 8️⃣ Chạy thử API server**

node src/index.js

* Chạy trực tiếp file backend bằng Node.
* Ban đầu lỗi:
* Error: ENOENT: no such file or directory, open 'firebase-service-account.json'

→ Nghĩa là thiếu file khóa Firebase, sau đó bạn thêm vào và server chạy thành công:

✅ API Server running at http://localhost:5000

**🧩 9️⃣ Chạy frontend (React + Vite)**

cd D:\DoAN\PJ\FamilyBudget\web-admin

npm run dev

* npm run dev → Chạy server Vite ở chế độ phát triển (hot reload).
* Sau khi thành công, hiển thị:
* ➜ Local: http://localhost:5173/
* ➜ Network: use --host to expose
* Vite chạy rất nhanh (0.4–1.2s), cho phép bạn sửa file React và xem kết quả ngay.

**🧩 10️⃣ Thông báo “hmr update” trong log**

[vite] (client) hmr update /src/components/SidebarMenu.jsx

* **HMR (Hot Module Replacement)**: Vite tự động reload module khi bạn chỉnh sửa file (không cần reload trang).
* Giúp quá trình phát triển frontend nhanh và liên tục.

**✅ Tổng kết quy trình bạn vừa làm**

| **Giai đoạn** | **Mục tiêu** | **Kết quả** |
| --- | --- | --- |
| 1️⃣ | Tạo cấu trúc thư mục | Có 2 module riêng biệt: api-server, web-admin. |
| 2️⃣ | Tạo Dockerfile | Định nghĩa cách build từng service. |
| 3️⃣ | Tạo docker-compose.yml | Quản lý toàn bộ container cùng lúc. |
| 4️⃣ | Build docker lần đầu | Thất bại do thiếu package.json. |
| 5️⃣ | Cài và cấu hình Node backend | Tạo API server Express hoạt động. |
| 6️⃣ | Chạy backend trực tiếp | Server chạy thành công tại localhost:5000. |
| 7️⃣ | Chạy frontend React | Web chạy tại localhost:5173. |
| 8️⃣ | HMR hoạt động | Mọi thay đổi cập nhật ngay lập tức. |