Identity Setup

---
2. DATABASE CONNECTION STRINGS

Where to put them: 'appsettings.json' inside "ConnectionStrings" section.

"ConnectionStrings": {
  "MongoDb": "mongodb://admin:password123@localhost:27017",
  "Redis": "localhost:6379"
}

---
3. CLOUDINARY SETUP (Images)

STEP A: Get Credentials
1. Sign up at Cloudinary.com.
2. Go to "Dashboard" → "Product Environment Credentials".
3. Copy: Cloud Name, API Key, API Secret.

STEP B: Where to put them (appsettings.json)
"CloudinarySettings": {
  "CloudName": "PASTE_CLOUD_NAME_HERE",
  "ApiKey": "PASTE_API_KEY_HERE",
  "ApiSecret": "PASTE_API_SECRET_HERE"
}

---
4. GMAIL SMTP SETUP (Email)

STEP A: Generate Password
1. Go to your Google Account (myaccount.google.com).
2. Click "Security" on the left.
3. Turn on "2-Step Verification" (if off).
4. Search for "App Passwords" in the top search bar.
5. Create new → Name it "MyApp".
6. Copy the 16-character code (e.g., "abcd efgh ijkl mnop").

STEP B: Where to put them (appsettings.json)
"EmailSettings": {
  "Server": "smtp.gmail.com",
  "Port": 587,
  "SenderName": "My App Support",
  "SenderEmail": "your.email@gmail.com",
  "AppPassword": "PASTE_16_CHAR_CODE_HERE"
}

---
5. POSTGRESQL (SEPARATE)

Since you wanted this separate, here is the standalone command to run Postgres by itself without affecting the others.

Command to run in terminal:
docker run -d \
  --name separate_postgres \
  -p 5432:5432 \
  -e POSTGRES_USER=admin \
  -e POSTGRES_PASSWORD=password123 \
  -e POSTGRES_DB=myappdb \
  postgres:15-alpine

Connection String for appsettings.json:
"Postgres":
"Host=localhost;Port=5432;Database=myappdb;Username=admin;Password=password123"

Here is exactly how to create them. You have two options: the **"Best Practice"** way (using a file) or the **"Fast"** way (using terminal commands).

Option 1: The "Best Practice" Way (Docker Compose)

Use this if you want them to start/stop together and save your data permanently.

1. Create a new empty file in your project folder named **docker-compose.yml**.
2. Paste this code into it:codeYaml

```
version: '3.8'
services:
  mongodb:
    image: mongo:latest
```

```
    container_name: my_mongo
    restart: always
    ports:
      - "27017:27017"
    environment:
      MONGO_INITDB_ROOT_USERNAME: admin
      MONGO_INITDB_ROOT_PASSWORD: password123
    volumes:
      - mongo_data:/data/db

  redis:
    image: redis:alpine
    container_name: my_redis
    restart: always
    ports:
      - "6379:6379"
    volumes:
      - redis_data:/data

volumes:
  mongo_data:
  redis_data:
```

3. Open your terminal (Command Prompt or PowerShell) in that folder.
4. Run this command:codeBash

```
docker-compose up -d
```

Option 2: The "Fast" Way (Terminal Commands)

Use this if you just want to spin them up quickly without creating a file.

1. Create MongoDB:

Run this single command in your terminal:

codeBash

```
docker run -d --name my_mongo -p 27017:27017 -e MONGO_INITDB_ROOT_USERNAME=admin -e MONGO_INITDB_ROOT_PASSWORD=password123 mongo:latest
```

2. Create Redis:

Run this single command:

codeBash

```
docker run -d --name my_redis -p 6379:6379 redis:alpine
```

How to check if they are running?

Type this in your terminal:

codeBash

```
docker ps
```

You should see two items listed under "NAMES": my_mongo and my_redis.

Your Connection URLs are now:

- **Mongo:** mongodb://admin:password123@localhost:27017
- **Redis:** localhost:6379