perplexity

Using this project plan, give me detailed instructions step by step for an inexperienced dev for Phase 1.2

Absolutely! Here are **detailed, step-by-step instructions for Phase 1.2: Database Setup & Configuration** for an inexperienced developer, using your project plan and assuming your main directory is:

```
C:\Users\TechFast Australia\Desktop\LifeTriage
```

Phase 1.2: Database Setup & Configuration

Goal:
Set up a Supabase PostgreSQL database, connect it to your Flask backend, create models for tasks and categories, and verify database operations.

1. Create a Supabase Account and Project

1. **Go to** Supabase **and sign up for a free account.**
2. **Create a new project:**
    - Click "New Project."
    - Choose your organization or create one.
    - Name your project (e.g., `lifetriage-db`).
    - Set a strong password (save this!).
    - Choose a region close to you.
    - Click "Create new project."
3. **Wait for your project to initialize.**
4. **In your Supabase dashboard, click the project you just created.**

2. Design and Create Database Schema

a. Create Tables

1. **Go to the “Table Editor” in the left sidebar.**
2. **Click “New Table.”**
   - **Table name:** `categories`
   - **Columns:**
     - `id` (type: `uuid`, default: `gen_random_uuid()`, primary key)
     - `name` (type: `text`, not null)
   - Click “Save.”
3. **Click “New Table” again.**
   - **Table name:** `tasks`
   - **Columns:**
     - `id` (type: `uuid`, default: `gen_random_uuid()`, primary key)
     - `description` (type: `text`, not null)
     - `category_id` (type: `uuid`, foreign key to `categories.id`)
     - `priority` (type: `integer`, default: `1`)
     - `completed` (type: `boolean`, default: `false`)
     - `created_at` (type: `timestamp with time zone`, default: `now()`)
   - Click “Save.”

3. Get Supabase Database Connection Details

1. **In your Supabase project dashboard, click “Project Settings” → “Database.”**
2. **Copy the “Connection string” (it will look like:**

   ```
   postgresql://postgres:[YOUR_PASSWORD]@db.[randomstring].supabase.co:5432/postgres
   ```

   **)
3. **You’ll use this in your Flask backend.**

4. Configure Database Connection in Flask

a. Install SQLAlchemy and psycopg2

1. **Open Command Prompt.**
2. **Navigate to your backend folder:**

   ```
   cd C:\Users\TechFast Australia\Desktop\LifeTriage\backend
   ```

3. **Activate your virtual environment:**

```
venv\Scripts\activate
```

4. **Install dependencies:**

```
pip install sqlalchemy psycopg2-binary
```

5. **Add these to your `requirements.txt`:**

```
pip freeze > requirements.txt
```

b. Set Up Environment Variables

1. **Open (or create) the `.env` file in your backend folder.**
2. **Add your database URL:**

```
DATABASE_URL=postgresql://postgres:[YOUR_PASSWORD]@db.[randomstring].supabase.co:5432
```

(Replace with your actual connection string.)

c. Create a Database Utility

1. **In `backend/app/utils/`, create a file called `database.py`.**
2. **Paste this code:**

```
import os
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
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)
```

5. Create SQLAlchemy Models

a. Create Models Folder

1. **In `backend/app/models/`, create `task.py` and `category.py`.**

b. Define Models

`category.py`:

```
from sqlalchemy import Column, String
from sqlalchemy.dialects.postgresql import UUID
import uuid
from app.utils.database import engine
from sqlalchemy.ext.declarative import declarative_base

Base = declarative_base()

class Category(Base):
    __tablename__ = "categories"
    id = Column(UUID(as_uuid=True), primary_key=True, default=uuid.uuid4)
    name = Column(String, nullable=False)
```

`task.py`:

```
from sqlalchemy import Column, String, Integer, Boolean, ForeignKey, DateTime
from sqlalchemy.dialects.postgresql import UUID
import uuid
from app.utils.database import engine
from sqlalchemy.ext.declarative import declarative_base
from datetime import datetime

Base = declarative_base()

class Task(Base):
    __tablename__ = "tasks"
    id = Column(UUID(as_uuid=True), primary_key=True, default=uuid.uuid4)
    description = Column(String, nullable=False)
    category_id = Column(UUID(as_uuid=True), ForeignKey("categories.id"))
    priority = Column(Integer, default=1)
    completed = Column(Boolean, default=False)
    created_at = Column(DateTime, default=datetime.utcnow)
```

c. Create Tables in Database

1. **In** `backend/app/models/__init__.py`**, add:**

    ```
    from .task import Task
    from .category import Category
    from app.utils.database import engine
    from sqlalchemy.ext.declarative import declarative_base

    Base = declarative_base()

    def create_tables():
        Base.metadata.create_all(bind=engine)
    ```

2. **In** `backend/app.py`**, add:**

```
from app.models import create_tables

if __name__ == "__main__":
    create_tables()
    app.run(debug=True)
```

3. **Run your Flask app to create the tables:**

```
python app.py
```

6. Seed Categories Table with Default Data

1. **In** `backend/app/services/task_service.py` **or a new** `seed.py` **file, add:**

```
from app.utils.database import SessionLocal
from app.models.category import Category

def seed_categories():
    db = SessionLocal()
    default_categories = ["Health", "Work", "Relationships", "Finance", "Personal Gro
    for name in default_categories:
        if not db.query(Category).filter_by(name=name).first():
            db.add(Category(name=name))
    db.commit()
    db.close()

if __name__ == "__main__":
    seed_categories()
```

2. **Run the seed script:**

```
python app/services/seed.py
```

(Adjust the path if needed.)

7. Test Database Connection and CRUD

1. **In your Flask app, try creating a route to test database access:**

```
@app.route("/api/test-db")
def test_db():
    from app.utils.database import SessionLocal
    from app.models.category import Category
    db = SessionLocal()
    categories = db.query(Category).all()
    db.close()
    return {"categories": [c.name for c in categories]}
```

2. **Restart your Flask server and visit** `http://localhost:5000/api/test-db`**.**
   - You should see your default categories listed.

8. Document Your Database Setup

1. **In `docs/api.md` or `docs/deployment.md`, describe:**
    - Your Supabase setup steps
    - The schema for `tasks` and `categories`
    - How to run the seed script

✅ Success Criteria

- [ ] **Database connects successfully from Flask app**
- [ ] **Can create, read, update, delete tasks via database**
- [ ] **Categories are populated with default data**

Congratulations!
You have completed Phase 1.2 and your Flask backend is now connected to a real PostgreSQL database on Supabase, with your schema and seed data in place.

⁂