💪 Awesome — here’s a **clean step-by-step guide** to set up **Flask-Migrate (Alembic)** with your existing project.  
This will let you do schema migrations from now on, and never write raw ALTER TABLE by hand.

**🚀 1️⃣ Install the required packages**

Run this in your virtual environment:

pip install Flask-Migrate

**🚀 2️⃣ Integrate Flask-Migrate in your app.py**

At the top, add:

from flask\_migrate import Migrate

Then **below your db = SQLAlchemy(app)**, add:

migrate = Migrate(app, db)

So that block becomes:

db = SQLAlchemy(app)

migrate = Migrate(app, db)

✅ Done. That’s it for code changes.

**🚀 3️⃣ Initialize Alembic migrations folder**

Run this from your project directory (where your app.py is):

flask db init

This will create a migrations/ folder with scripts.  
(If you don’t have a FLASK\_APP set, you may need to run:

set FLASK\_APP=app.py # on Windows

export FLASK\_APP=app.py # on Linux/mac

)

**🚀 4️⃣ Create your first migration script**

This will compare your database **models (Python code)** with your **actual database**, and create a migration file.

flask db migrate -m "Initial migration"

This generates a migration script in migrations/versions/.

**🚀 5️⃣ Apply the migration to your database**

This actually runs the migration and updates the schema.

flask db upgrade

🎉 Now your database is synced with your models!

**✅ Future: whenever you change your models**

Just do:

flask db migrate -m "Added ondelete cascade to notification"

flask db upgrade

It will generate ALTER TABLE statements automatically.

**💡 Notes for your existing change**

Since your database **already has tables**, your first migration will recognize them as existing, and might generate a baseline.

If you **already dropped and re-added constraints manually**, that’s fine — your next migrations will track from here.

✅ That’s it!  
If you want, I can even write you a small **README.md snippet** to keep in your repo on **how to run migrations**. Just tell me. 🚀