AeroAspire

SDE Intern

Goutham V

Week 4 – Day4 (16th October)

Task:

Learn/use Flask-Migrate or write simple script to seed DB; migrations when schema changes

Reflections,

1. What is a migration? How it works: generating migration file → applying it → version control.

A **migration** is a way to update the database structure (like adding or removing columns) without losing existing data.
It helps keep your database in sync with changes in your code.

The process usually goes like this:

1. *Generate migration file* — Flask-Migrate or Alembic scans model changes and creates a migration script.
2. *Apply migration* — The migration script runs and updates the database schema.
3. *Version control* — Each migration is tracked so you can roll back or reapply changes when needed.

This makes it easy to manage database changes safely and systematically.

2. How to seed data: why and how.

Seeding data means adding some sample or default data into the database.

Why:

- It helps in testing and demoing features without entering data manually each time.
- It ensures consistent test data for developers and testers.

How:
You can write a simple script (like `seed_data.py`) to insert predefined records into the database using SQLAlchemy or direct SQL commands. Run it once after setting up the database to load your initial data.

3. If you need to add a new column to tasks table after app is in use, how do you do that safely?

To add a new column safely:

1. *Create a migration* that includes the new column definition in your model.
2. *Run the migration* using Flask-Migrate (for example, `flask db migrate` and `flask db upgrade`).
3. *Provide a default value* or make the column nullable to prevent breaking existing data.
4. *Test the app* to make sure old data and new schema work together.

This approach avoids data loss and ensures your application continues running smoothly while updating the schema.