Progress Tracker: Week 37, 2025

**✅ Day 1 – Budget App Journal Checklist**

**📝 Planning**

* Write down **MVP features**:
  + User signup/login
  + Add income/expense
  + Categorize transactions
  + Monthly summary
  + Export CSV/PDF (optional)
* Note **non-MVP features** (bank APIs, AI, recurring transactions, notifications).

**⚙️ Setup Tools**

* Install **Python** (latest version).
* Install **Flutter SDK**.
* Install **VS Code** + extensions (Python, Flutter).
* Install **Git**.
* (Optional) Install **Postman** for testing APIs.
* Create a folder: BudgetAppProject/.

**🏗 Backend Setup (Flask)**

* Create folder budget\_app\_backend/.
* Create and activate a Python virtual environment.
* Install dependencies (flask, flask-restful, flask-jwt-extended, flask-cors, sqlalchemy).
* Create app.py with “Hello World” route.
* Test by running python app.py → confirm in browser.
* Create requirements.txt.

**🎨 Frontend Setup (Flutter)**

* Create folder budget\_app\_frontend/ using flutter create budget\_app\_frontend.
* Run default Flutter app (flutter run) → confirm it works.

**🌐 GitHub Setup**

* Create **GitHub repo**: budget\_app\_backend.
* Create **GitHub repo**: budget\_app\_frontend.
* Initialize Git in backend folder → commit → push.
* Initialize Git in frontend folder → commit → push.

**🗂 Documentation**

* Write a simple **README.md** for backend (how to run Flask app).
* Write a simple **README.md** for frontend (how to run Flutter app).
* Journal entry: note what worked, what was tricky, and what you learned.

👉 By the end of Day 1, you should have:  
✔ A “Hello World” Flask backend running.  
✔ A default Flutter app running.  
✔ Both projects uploaded to GitHub.  
✔ Your journal updated with progress + reflections.

✅ Day 2 – Budget App Journal Checklist

### 🎯 Goal

Set up the **Backend MVP** with authentication and transaction handling.

### 🏗 Backend Development Tasks

* Create **models** with SQLAlchemy:
  + User → id, email, password\_hash
  + Category → id, name
  + Transaction → id, user\_id, amount, type, category\_id, date, note
* Set up **database connection** (database.py)
* Apply **migrations** or auto-create **SQLite database**

### 🔑 Authentication

* Install & configure Flask-JWT-Extended
* Create **signup endpoint** (POST /auth/signup)
* Create **login endpoint** (POST /auth/login)
* Return **JWT token** on successful login

### 🔄 Transactions API

* Create **GET /transactions** → fetch user’s transactions
* Create **POST /transactions** → add new income/expense
* Ensure **JWT authentication** is required for all endpoints

### 📖 API Documentation

* Install **Flasgger** (or switch to **FastAPI** if preferred)
* Add **Swagger/OpenAPI docs** for:
  + Signup
  + Login
  + Transactions

### 🧪 Testing

* Test **signup & login** with Postman
* Confirm JWT token works for **protected routes**
* Test **creating a transaction** and **fetching transactions**

### 🗂 Documentation & Journal

* Update **backend README.md** with new endpoints
* Journal entry:
  + Endpoints created & tested
  + Bugs or issues encountered

### 👉 By the end of Day 2, you should have:

* ✔ User signup & login working with JWT
* ✔ Ability to add and view transactions (linked to user)
* ✔ API documented (Swagger/OpenAPI)
* ✔ Tested endpoints in Postman