REGRESSION-APP

A full-stack regression web application built with FastAPI, React + Vite + TailwindCSS, and PostgreSQL (Neon).

It allows users to upload datasets, run regression experiments, view metrics & plots, and track experiment history.

PROJECT STRUCTURE

```
regression-app/
├── backend/
│   ├── app.py           # FastAPI app entrypoint
│   ├── auth.py          # JWT authentication (register/login)
│   ├── deps.py          # Dependency utilities (current user, auth)
│   ├── db.py            # SQLAlchemy engine & session
│   ├── models.py        # Database models
│   ├── schemas.py       # Pydantic request/response schemas
│   ├── routers/
│   │   ├── datasets.py          # Dataset upload endpoints
│   │   └── experiments.py       # Run experiments & fetch results
│   ├── ml/
│   │   ├── pipeline.py          # Regression pipeline
│   │   └── plots.py             # Generate metrics plots (PNG)
│   ├── tests/                   # Unit & integration tests
│   ├── migrations/              # Alembic migration scripts
│   ├── alembic.ini              # Alembic configuration
│   ├── requirements.txt         # Python dependencies
│   ├── .env                     # Environment variables (local only)
│   └── init.py                  # Package marker
│
├── frontend/
│   ├── index.html
│   ├── package.json
│   ├── vite.config.ts
│   ├── tsconfig.json
│   ├── postcss.config.js
│   ├── tailwind.config.js
│   ├── src/
│   │   ├── main.tsx
│   │   ├── App.tsx
│   │   ├── api.ts               # Axios base instance (JWT token)
│   │   ├── pages/               # Pages: Auth, Upload, Configure, Results, History
│   │   └── components/          # Reusable components (MetricCards, Tables)
│   └── public/                  # Static assets
│
├── .gitignore
```

Features

Authentication: JWT-based register/login.

Dataset Management: Upload CSV/XLSX datasets; store metadata & column types.

Regression Experiments:

- Train/test split
- Regression pipeline (scaling, one-hot encoding)
- Metrics: R^2, MAE, MSE, RMSE
- Plots: Predicted vs Actual, Residuals
- Store trained model & plots in DB

History Tracking: View past runs, metrics, and experiment artifacts.

PDF Export: (Optional) Download experiment reports.

SETUP INSTRUCTIONS

1. Clone Repository

```
git clone <repo-url> regression-app

cd regression-app
```

2. Setup Backend

```
cd backend

python -m venv .venv

# Activate

# macOS/Linux:

source .venv/bin/activate

# Windows:

.venv\Scripts\activate

pip install -r requirements.txt
```

3. Configure .env

Create backend/.env:

```
DATABASE_URL=postgresql+psycopg://USER:PASSWORD@HOST/DB?sslmode=require

JWT_SECRET=your-secret-key

JWT_EXPIRE_MIN=30
```

```
CORS_ORIGINS=http://localhost:5173
```

4. Run Database Migrations (Alembic)

```
alembic upgrade head
```

5. Start Backend

```
uvicorn app:app --reload --port 8000
```

6. **Setup Frontend**

```
cd ../frontend

npm install

npm run dev
```

End-to-End Flow

1. **Register/Login** → JWT token stored in browser.
2. **Upload Dataset** → CSV/XLSX → metadata saved in DB.
3. **Configure Experiment** → choose target column, features, regression algorithm, train/test split.
4. **Run Experiment** → backend computes metrics, generates plots, stores model & artifacts.
5. **Results Page** → shows metrics and plots.
6. **History Page** → list past runs and details.

Database (Neon / PostgreSQL)

- Tables: users, datasets, dataset_columns, experiments, experiment_metrics, experiment_artifacts.
- Check Neon console for rows:

```
SELECT * FROM experiments ORDER BY created_at DESC LIMIT 5;

SELECT octet_length(model_bin) FROM experiment_artifacts ORDER BY 1 DESC LIMIT 5;
```