🥦 GroceryMate - Smart Grocery Management System

A full-stack web application to manage groceries, track fridge and pantry inventory, create shopping lists, and discover healthy recipes based on available ingredients.

🛠️ Tech Stack

Frontend

- **Vue.js 3** - Progressive JavaScript framework
- **Vite** - Next-generation frontend tooling
- **Tailwind CSS** - Utility-first CSS framework
- **Vue Router** - Official router for Vue.js
- **Axios** - HTTP client

Backend

- **FastAPI** - Modern Python web framework
- **PostgreSQL** - Relational database
- **SQLAlchemy** - SQL toolkit and ORM
- **Pydantic** - Data validation

DevOps

- **Docker Desktop** - Containerization platform
- **Docker Compose** - Multi-container orchestration

📋 Prerequisites

Before starting, ensure you have:

1. **Node.js** (v18 or higher) - Download
2. **Python** (v3.10 or higher) - Download
3. **Docker Desktop** - Download
4. **Git** (optional) - Download
5. **Code Editor** - VS Code recommended

🚀 Setup Instructions

Step 1: Create Project Directory

Open PowerShell or Command Prompt:

```powershell
# Create and navigate to project directory
mkdir grocery-mate
cd grocery-mate
```

Step 2: Setup Backend

1. **Create backend structure:**

```powershell
mkdir backend
cd backend
mkdir app
cd app
mkdir routers
```

2. **Create necessary files:**

Create the following files and copy content from artifacts:

- `requirements.txt` (in backend folder)
- `app/__init__.py` (empty file)
- `app/database.py`
- `app/models.py`
- `app/schemas.py`
- `app/main.py`
- `app/routers/__init__.py` (empty file)
- `app/routers/ingredients.py`
- `app/routers/shopping_lists.py`
- `app/routers/recipes.py`

3. **Create .env file in backend folder:**

```env
DATABASE_URL=postgresql://grocery_user:grocery_pass@localhost:5432/grocery_db
```

4. **Setup Python virtual environment:**

```powershell
# Navigate to backend folder
cd .. # (you should be in backend folder)

# Create virtual environment
python -m venv venv

# Activate virtual environment
# For PowerShell:
.\venv\Scripts\Activate.ps1

# For Command Prompt:
.\venv\Scripts\activate.bat

# If you get execution policy error in PowerShell, run:
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
```

5. **Install Python dependencies:**

```powershell
pip install -r requirements.txt
```

Step 3: Setup Database with Docker

1. **Start Docker Desktop** (make sure it's running)
2. **Create docker-compose.yml in root folder:**

Navigate back to project root and create `docker-compose.yml` with the provided content.

```powershell
cd .. # Go back to grocery-mate folder
```

3. **Start PostgreSQL container:**

```powershell
docker-compose up -d
```

4. **Verify database is running:**

```powershell
docker ps
```

You should see a container named `grocery_postgres` running.

Step 4: Setup Frontend

1. **Create frontend structure:**

```
powershell

mkdir frontend
cd frontend
```

2. **Create package.json:**

Create `package.json` with the provided content.

3. **Install Node dependencies:**

```
powershell

npm install
```

4. **Create configuration files:**

Create these files in frontend folder:

- `vite.config.js`
- `tailwind.config.js`
- `postcss.config.js`
- `index.html`

5. **Create src structure:**

```
powershell

mkdir src
cd src
mkdir assets
mkdir components
mkdir views
mkdir services
mkdir router
```

6. **Create frontend files:**

Create these files with provided content:

- `src/main.js`
- `src/App.vue`
- `src/assets/main.css`
- `src/services/api.js`
- `src/router/index.js`
- `src/views/Dashboard.vue`
- `src/views/Ingredients.vue`
- `src/views/ShoppingLists.vue`
- `src/views/Recipes.vue`

🎯 Running the Application

Terminal 1 - Backend Server

```powershell
# Navigate to backend folder
cd path\to\grocery-mate\backend

# Activate virtual environment (if not already activated)
.\venv\Scripts\Activate.ps1

# Run FastAPI server
uvicorn app.main:app --reload --host 0.0.0.0 --port 8000
```

The backend API will be available at:

- **API**: http://localhost:8000
- **API Docs**: http://localhost:8000/docs (Interactive Swagger UI)
- **ReDoc**: http://localhost:8000/redoc

Terminal 2 - Frontend Development Server

```powershell
# Navigate to frontend folder
cd path\to\grocery-mate\frontend

# Run Vite development server
npm run dev
```

The frontend will be available at:

- **App**: http://localhost:5173

📱 Using the Application

1. Dashboard

- View total ingredients count
- See items in fridge vs pantry
- Get alerts for expiring items
- Quick access to main features

2. Ingredients Management

- Add new ingredients with details (name, category, location, quantity, expiry date)
- Filter by location (Fridge/Pantry)
- Edit or delete ingredients
- Track expiration dates

3. Shopping Lists

- Create multiple shopping lists
- Add items with quantities
- Mark items as purchased
- Delete completed lists

4. Recipes

- Browse healthy recipes
- Find recipes matching available ingredients
- Load sample recipes
- View detailed cooking instructions

🧪 Testing the Application

1. **Test Backend API:**

Open http://localhost:8000/docs in your browser to:

- Test all API endpoints

- View request/response schemas
- Execute test requests

2. **Test Frontend:**

- Navigate through all pages
- Add sample ingredients
- Create shopping lists
- Load sample recipes
- Test recipe matching feature

🐛 Troubleshooting

Backend Issues

Problem: ModuleNotFoundError

```powershell
# Make sure virtual environment is activated
.\venv\Scripts\Activate.ps1

# Reinstall dependencies
pip install -r requirements.txt
```

Problem: Database connection error

```powershell
# Check if Docker container is running
docker ps

# Restart container
docker-compose down
docker-compose up -d
```

Frontend Issues

Problem: Module not found

```powershell
```

```
# Delete node_modules and reinstall
rm -r node_modules
npm install
```

Problem: Port already in use

```
powershell

# Frontend will auto-assign next available port
# Or kill the process using the port
```

📚 API Endpoints

Ingredients

- `GET /ingredients/` - Get all ingredients
- `GET /ingredients/{id}` - Get specific ingredient
- `POST /ingredients/` - Create ingredient
- `PUT /ingredients/{id}` - Update ingredient
- `DELETE /ingredients/{id}` - Delete ingredient
- `GET /ingredients/expiring/soon` - Get expiring items

Shopping Lists

- `GET /shopping-lists/` - Get all lists
- `POST /shopping-lists/` - Create list
- `POST /shopping-lists/{id}/items` - Add item to list
- `PUT /shopping-lists/items/{id}` - Update item status
- `DELETE /shopping-lists/{id}` - Delete list

Recipes

- `GET /recipes/` - Get all recipes
- `POST /recipes/` - Create recipe
- `GET /recipes/match/ingredients` - Find matching recipes
- `POST /recipes/seed-sample` - Load sample recipes

🔒 Security Notes

- This is a **development setup** - not production-ready
- Database credentials are in plain text (use environment variables in production)
- No authentication/authorization implemented
- CORS is open for development

📝 Project Structure Summary

```
grocery-mate/
├── backend/              # FastAPI backend
│   ├── app/
│   │   ├── routers/      # API route handlers
│   │   ├── database.py   # Database configuration
│   │   ├── models.py     # SQLAlchemy models
│   │   ├── schemas.py    # Pydantic schemas
│   │   └── main.py       # FastAPI application
│   ├── venv/             # Python virtual environment
│   └── requirements.txt  # Python dependencies
│
├── frontend/             # Vue.js frontend
│   ├── src/
│   │   ├── views/        # Page components
│   │   ├── services/     # API service layer
│   │   ├── router/       # Vue Router config
│   │   └── assets/       # Static assets
│   └── package.json      # Node dependencies
│
└── docker-compose.yml    # Database container config
```

🎓 Learning Resources

- **Vue.js**: https://vuejs.org/guide/
- **FastAPI**: https://fastapi.tiangolo.com/
- **Tailwind CSS**: https://tailwindcss.com/docs
- **SQLAlchemy**: https://docs.sqlalchemy.org/
- **Docker**: https://docs.docker.com/

💡 Features to Add (Exercises)

1. Add user authentication
2. Implement recipe categories
3. Add nutrition tracking
4. Create meal planning feature
5. Add barcode scanning
6. Implement data export (CSV/PDF)
7. Add email notifications for expiring items

📄 License

Educational project - Free to use and modify

🤝 Support

If you encounter issues:

1. Check error messages carefully
2. Verify all files are created correctly
3. Ensure Docker Desktop is running
4. Check that all dependencies are installed
5. Review the API documentation at `/docs`

Happy Coding! 🚀