Fogon Inventory Management System – Developer Guidelines:

https://github.com/Landy2233/Fogon-Inventory-Management-System.git

This document provides all information required for developers contributing to the Fogon IMS project. It is fully self-contained, includes complete setup instructions, testing guidance, repository structure, and CI details.

---

1. System Overview

Fogon IMS is a full-stack Inventory Management System consisting of:

- **Backend:** Python + Flask
- **Mobile App:** React Native (Expo)
- **Database:** SQLite (local)
- **Dev Tools Required:** Git, Python 3.10+, Node.js, npm, Expo, Xcode (macOS), Expo Go (Windows/macOS)

This document explains how to install, run, test, and contribute to the project.

---

2. Repository Structure

```
Fogon-Inventory-Management-System/
│
├── backend/ or root backend files
│   ├── app.py
│   ├── requirements.txt
│   ├── instance/
│   ├── database.db (generated after init)
│   ├── app/
│   │   ├── routes/
│   │   ├── models/
│   │   ├── utils/
│   │   └── __init__.py
│
└── FogonIMSMobile/
    ├── src/
```

```
|   ├── api/
|   |   └── client.ts
|   ├── screens/
|   ├── components/
|   ├── hooks/
|   └── navigation/
├── package.json
├── app.json
└── node_modules/
```

Explanation

- **The backend folder** contains all Flask server files and database logic.
- **FogonIMSMobile** contains all React Native screens, API calls, navigation, and UI components.
- The repository is structured so backend and mobile can be tested independently.

This structure is clear, organized, and easy for contributors to understand.

---

3. Install & Run the Project (Full Setup)

Below are the exact developer setup steps (included exactly as required):

3.1 Install Required Tools

Git

Verify:

```
git --version
```

If missing (macOS):

```
brew install git
```

Python 3.10+

Verify:

```
python3 --version   # macOS/Linux
python --version    # Windows
```

Node.js + npm + Expo

Verify:

```
node -v
npm -v
npx -v
npx expo --version
```

If missing (macOS):

```
brew install node
```

Install Expo CLI:

```
npm install -g expo-cli
```

Xcode (macOS only) – required for iOS Simulator

Download from Mac App Store.

After installation:

```
sudo xcodebuild -runFirstLaunch
xcode-select -p
```

Expo Go (Windows/macOS) – Required for phone testing

Install Expo Go from the App Store / Google Play.

3.2 Download the Project

```
git clone
https://github.com/Landy2233/Fogon-Inventory-Management-System.git
```

```
cd Fogon-Inventory-Management-System
```

3.3 Start the Backend (Flask)

Create virtual environment

macOS/Linux:

```
python3 -m venv venv
source venv/bin/activate
```

Windows:

```
python -m venv venv
venv\Scripts\activate
```

Upgrade pip

Windows:

```
python3.exe -m pip install --upgrade pip
```

Mac:

```
pip install --upgrade pip
```

Install dependencies

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

Initialize the database

```
flask --app app init-db
```

Run the backend

```
python app.py
```

Backend URLs:

- http://localhost:5001
- http://YOUR_WIFI_IP:5001 (required for phone testing)

Leave this terminal running.

3.4 Run the Mobile App (Expo)

Open a new terminal:

```
cd Fogon-Inventory-Management-System/FogonIMSMobile
npm install
```

Update API client

Edit:
`FogonIMSMobile/src/api/client.ts`

Set:

```
const DEV_LAN = "http://YOUR_WIFI_IP:5001/api";
```

Find Wi-Fi IP:
macOS → `ifconfig`
Windows → `ipconfig`

Start Expo

```
npx expo start
```

Open the app:

- Press **i** → iOS Simulator (macOS)
- OR scan QR code with **Expo Go App** (Windows/macOS)

---

4. Demo Logins (Seeded Test Accounts)

These accounts allow developers to test manager vs. cooker flows.

Role	Email	Password
Manager	manager_demo@fogon.com	Manager123!
Cooker	cooker_demo@fogon.com	Cook123!

Developers can also register their own accounts.

---

5. How to Add a New Test Case

Backend tests follow a standard structure:

1. Create a new test file

Inside:

```
backend/tests/
```

Example:

```
test_inventory.py
```

2. Import app + test client

```
from app import app

def test_sample():
    client = app.test_client()
    response = client.get("/api/items")
    assert response.status_code == 200
```

3. Run tests

```
pytest
```

4. Add test data

Use:

`backend/tests/data/`

This structure makes testing simple and consistent.

---

6. Continuous Integration (CI) Setup

If GitHub Actions is used, CI files are located in:

`.github/workflows/`

Example workflow keywords:

- install Python
- install npm
- run backend tests
- run mobile lint checks
- build Expo bundle

CI Build History

Go to:

GitHub → Actions Tab

You will see a complete build history for:

- backend checks
- mobile build checks
- test execution

---

7. Developer Responsibilities

- Follow repository structure
- Write tests for new features
- Do not push directly to main
- Create feature branches

- Make clear pull requests
- Ensure backend + mobile both run before submitting code

---

8. Filing Bug Reports

Developers should open issues following this format:

Title: Short and descriptive
Steps to Reproduce:
Expected Result:
Actual Result:
Screenshots or Logs:
Environment: OS, device, Expo version, backend state

This ensures consistent and reliable tracking of bugs.
