# Verdan Platform - App & Subscription Management

Generated on: 2025-02-07 07:07:06

## 1. Project File Structure

The project is organized to keep apps modular while ensuring a scalable and maintainable codebase.

/verdan\_platform  
│── /apps # Holds individual app modules  
│ ├── /freezer\_monitoring  
│ │ ├── \_\_init\_\_.py # Initialize app module  
│ │ ├── models.py # App-specific database models  
│ │ ├── routes.py # API routes for this app  
│ │ ├── services.py # Business logic  
│ │ ├── triggers.py # App-specific triggers (Temp Above, etc.)  
│ ├── /pond\_monitoring  
│ │ ├── \_\_init\_\_.py  
│ │ ├── models.py  
│ │ ├── routes.py  
│ │ ├── services.py  
│ │ ├── triggers.py  
│── /alerts # Global alerts module  
│ ├── models.py # Stores alert settings  
│ ├── routes.py # API for configuring alerts  
│ ├── services.py # Logic to send alerts via Email/SMS  
│ ├── email\_service.py # Handles email sending  
│ ├── sms\_service.py # Handles SMS sending  
│── /models # Global models (accounts, apps, devices, etc.)  
│── /services # Business logic for global entities  
│── /config.py # Configuration settings  
│── /main.py # Main Flask app  
│── /wsgi.py # WSGI entry point  
│── .env # Stores environment variables (excluded from Git)  
│── /requirements.txt

## 2. Database Models

### Accounts Table

CREATE TABLE accounts (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 name TEXT NOT NULL,  
 email TEXT UNIQUE NOT NULL,  
 stripe\_customer\_id TEXT UNIQUE NOT NULL,  
 created\_at TIMESTAMP DEFAULT NOW()  
 );

### Apps Table

CREATE TABLE apps (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 name TEXT NOT NULL,  
 description TEXT,  
 stripe\_product\_id TEXT UNIQUE NOT NULL,  
 monthly\_price DECIMAL(10,2) NOT NULL,  
 yearly\_price DECIMAL(10,2) NOT NULL,  
 created\_at TIMESTAMP DEFAULT NOW()  
 );

### Account Apps Table

CREATE TABLE account\_apps (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 account\_id UUID REFERENCES accounts(id) ON DELETE CASCADE,  
 app\_id UUID REFERENCES apps(id) ON DELETE CASCADE,  
 stripe\_subscription\_id TEXT UNIQUE NOT NULL,  
 created\_at TIMESTAMP DEFAULT NOW()  
 );

### Devices Table

CREATE TABLE devices (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 device\_id TEXT UNIQUE NOT NULL,  
 account\_id UUID REFERENCES accounts(id) ON DELETE CASCADE,  
 app\_id UUID REFERENCES apps(id) ON DELETE CASCADE,  
 created\_at TIMESTAMP DEFAULT NOW()  
 );

### Alert Triggers Table

CREATE TABLE alert\_triggers (  
 id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),  
 account\_id UUID REFERENCES accounts(id) ON DELETE CASCADE,  
 app\_id UUID REFERENCES apps(id) ON DELETE CASCADE,  
 trigger\_type TEXT NOT NULL,  
 trigger\_value FLOAT NOT NULL,  
 message TEXT NOT NULL,  
 notify\_emails TEXT[],  
 notify\_phones TEXT[],  
 notification\_type TEXT CHECK (notification\_type IN ('email', 'sms', 'both')) NOT NULL,  
 created\_at TIMESTAMP DEFAULT NOW()  
 );

## 3. Flask API Routes

### Install an App

POST /install\_app  
 Request Body:  
 {  
 "account\_id": "account-uuid",  
 "app\_id": "app-uuid"  
 }

### Uninstall an App

POST /uninstall\_app  
 Request Body:  
 {  
 "account\_id": "account-uuid",  
 "app\_id": "app-uuid"  
 }

### Create Alert Trigger

POST /alerts/create  
 Request Body:  
 {  
 "account\_id": "account-uuid",  
 "app\_id": "app-uuid",  
 "trigger\_type": "temp\_above",  
 "trigger\_value": 32,  
 "message": "Warning! Freezer temperature too high!",  
 "notify\_emails": ["user1@example.com", "user2@example.com"],  
 "notify\_phones": ["+12065550123", "+12065550124"],  
 "notification\_type": "both"  
 }

### Stripe Webhook

POST /stripe\_webhook  
 Handles subscription cancellations and updates.

## 4. Alert Services

The alert system allows users to set up triggers for email and SMS notifications based on device data.

### Send Alert

def send\_alert(alert, triggered\_value):  
 message = f"{alert.message}\nTriggered Value: {triggered\_value}"  
 if alert.notification\_type in ['email', 'both']:  
 for email in alert.notify\_emails:  
 send\_email(email, "Alert Notification", message)  
 if alert.notification\_type in ['sms', 'both']:  
 for phone in alert.notify\_phones:  
 send\_sms(phone, message)

### Send Email

def send\_email(to\_email, subject, body):  
 with smtplib.SMTP(SMTP\_SERVER, SMTP\_PORT) as server:  
 server.starttls()  
 server.login(SMTP\_USER, SMTP\_PASS)  
 server.sendmail(SMTP\_USER, to\_email, body)

### Send SMS via Twilio

def send\_sms(to\_phone, body):  
 client.messages.create(  
 body=body,  
 from\_=TWILIO\_PHONE\_NUMBER,  
 to=to\_phone  
 )

## 5. Security Best Practices

Environment variables should be stored in a `.env` file and excluded from Git using `.gitignore`.

Example `.env` file:  
TWILIO\_ACCOUNT\_SID=your\_account\_sid  
TWILIO\_AUTH\_TOKEN=your\_auth\_token  
TWILIO\_PHONE\_NUMBER=+12065550000  
  
SMTP\_SERVER=smtp.example.com  
SMTP\_PORT=587  
SMTP\_USER=alerts@example.com  
SMTP\_PASS=yourpassword  
  
Add `.env` to `.gitignore` to prevent accidental commits.