Here is a little explanation, what the different parts of our app are, what they do and why we need them:

1. Creating the necessary folder structure

- **What it is:** We're making the directories (folders) on the server that will store our application files, configuration files, and HTML pages.
- **Why we need it:** This keeps everything organized and ensures our app has a clean, predictable place to live. We'll separate backend code, frontend static files, and logs so it's easy to maintain later.

2. Installing Python and dependencies

- **What it is:** Python is the programming language our Flask app is written in. Dependencies are extra packages (like Flask, Stripe, and Supabase libraries) that the app needs to function.
- **Why we need it:** Without Python and the correct dependencies, the server won't be able to run our application code. Installing them in a *virtual environment* keeps everything isolated from other software on the server.

3. Setting up NGINX as our web server

- **What it is:** NGINX (pronounced "Engine-X") is a high-performance web server that handles incoming web requests from users.
- **Why we need it:**
    - It serves our static files (HTML, CSS, JS) quickly and efficiently.
    - It forwards all API requests to our Flask app running in the background (reverse proxy).
    - It's designed to handle many simultaneous users, which Flask alone isn't great at.

4. Securing the app with HTTPS

- **What it is:** HTTPS encrypts the data between the user's browser and your server. We'll use a free SSL/TLS certificate from Let's Encrypt (via Certbot) to make this happen.
- **Why we need it:**
    - Protects sensitive information (like login credentials and payment data).
    - Modern browsers warn users if a site is not secure — HTTPS avoids that.
    - Stripe and most modern APIs require HTTPS for callbacks and webhooks.

5. Configuring the app to run as a service

- **What it is:** We'll set up a *systemd service* for our Flask app so it starts automatically when the server boots and restarts if it crashes. We'll run Flask through **Gunicorn**, a production-ready Python application server.
- **Why we need it:**
  - Without it, we'd have to manually start the app every time the server restarts.
  - It ensures 24/7 uptime for the application.
  - Gunicorn allows the app to handle multiple requests in parallel, which is essential for real users.

This is the folder structure we need to create on our cloud server:

```
/opt/gptsweetheart/                # Flask app (backend)
├─ app.py
├─ .env
├─ venv/                           # Python virtual env
└─ logs/
   └─ gunicorn.log

/var/www/gptsweetheart/            # Static site (frontend)
├─ sign-up.html
├─ login.html
├─ redirect.html
├─ dashboard.html
├─ assets/                         # css/js/images for frontend
└─ media/                          # generated images served at /media/*
```

For that we use the following bashes:

#1 - Main Project folder

```
sudo mkdir -p /opt/gptsweetheart
sudo chown $USER:$USER /opt/gptsweetheart
```

#2 - Backend folders (with dummy files that we replace)

```
sudo mkdir -p /opt/gptsweetheart/logs
sudo touch /opt/gptsweetheart/app.py /opt/gptsweetheart/.env /opt/gptsweetheart/logs/
gunicorn.log
sudo chown -R $USER:$USER /opt/gptsweetheart
```

#3 - Frontend folders (with dummy files that we replace)

```
sudo mkdir -p /var/www/gptsweetheart/assets /var/www/gptsweetheart/media
sudo touch /var/www/gptsweetheart/sign-up.html \
        /var/www/gptsweetheart/login.html \
        /var/www/gptsweetheart/redirect.html \
        /var/www/gptsweetheart/dashboard.html
sudo chown -R www-data:www-data /var/www/gptsweetheart
sudo chown -R $USER:$USER /var/www/gptsweetheart/*.html  # so you can edit pages
```

#4 - Create the .env file in the right folder

```
cd /opt/gptsweetheart && nano .env
```

#5 - Paste that with YOUR values and then safe and close with Control + O, Enter, Control + X

```
SUPABASE_URL=https://your-supabase-project-url.supabase.co
SUPABASE_SERVICE_ROLE_KEY=your-supabase-service-role-key

STRIPE_SECRET_KEY=sk_test_your_stripe_secret_key
STRIPE_PRICE_ID_PRO=price_your_stripe_price_id
STRIPE_WEBHOOK_SECRET=whsec_your_stripe_webhook_secret

FRONTEND_URL=https://gptsweetheart.com
```

#6 Navigate to the right folder
```
cd /opt/gptsweetheart
```

#7 Python + venv install
```
sudo apt-get update
sudo apt-get install -y python3-venv python3-pip
```

#8 Create venv + install deps
```
python3 -m venv venv
source venv/bin/activate
pip install --upgrade pip
pip install flask flask-cors supabase python-dotenv stripe gunicorn
```

#9 install NGINX:
```
sudo apt update
sudo apt install nginx -y
```

#10 NGINX create server block:
```
sudo tee /etc/nginx/sites-available/gptsweetheart >/dev/null <<'EOF'
server {
    listen 80;
    listen [::]:80;
    server_name gptsweetheart.com www.gptsweetheart.com;

    root /var/www/gptsweetheart;
    index index.html;

    # Pretty URLs: serve file.html when /file is requested
    location / {
        try_files $uri $uri.html =404;
    }

    # API -> Flask (gunicorn) on localhost:5002
    location /api/ {
        proxy_pass http://127.0.0.1:5002/;
        proxy_http_version 1.1;
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $remote_addr;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_read_timeout 180s;
    }

    # (Optional) serve generated images later
    location /media/ {
        alias /var/www/gptsweetheart/media/;
        add_header Cache-Control "public, max-age=31536000, immutable";
    }
}
EOF
```

#11 Disable default site if present
```
sudo rm -f /etc/nginx/sites-enabled/default
```

#12 Enable new site
```
sudo ln -sf /etc/nginx/sites-available/gptsweetheart /etc/nginx/sites-enabled/gptsweetheart
```

#13 Test & reload
```
sudo nginx -t
sudo systemctl reload nginx
```

#14 Firewall adjustment
```
sudo ufw allow 'Nginx Full'
```

#15 Install certbot (if not installed)

```
sudo apt-get update
sudo apt-get install -y certbot python3-certbot-nginx
```

#16 # Issue certs

```
sudo certbot --nginx -d gptsweetheart.com -d www.gptsweetheart.com
```

#17 Replace the dummy files

Replace the dummy files on the server with the real files (app.py and the html pages)

#18 Create the systems unit

```
sudo tee /etc/systemd/system/gptsweetheart.service >/dev/null <<'EOF'
[Unit]
Description=GPTSweetheart Flask (gunicorn)
After=network.target

[Service]
Type=simple
WorkingDirectory=/opt/gptsweetheart
Environment="PYTHONUNBUFFERED=1"
ExecStart=/opt/gptsweetheart/venv/bin/gunicorn -w 3 -b 127.0.0.1:5002 app:app --timeout 120
Restart=always
User=www-data
Group=www-data

[Install]
WantedBy=multi-user.target
EOF
```

#19 Permission and start

```
sudo chmod -R 750 /opt/gptsweetheart

# Start service
sudo systemctl daemon-reload
sudo systemctl enable --now gptsweetheart
sudo systemctl status gptsweetheart --no-pager
```