Production Deployment Guide

This document outlines the steps to deploy the `platform_core` backend application to a production server.

Prerequisites

Before deploying, ensure the following are installed on your production server:

- Python 3.x
- pip (Python package installer)
- Git
- A database system (e.g., SQLite, PostgreSQL, MySQL) compatible with SQLAlchemy.

Deployment Steps

1. Clone the Repository

First, clone your project repository to your production server:

```
 git clone https://github.com/alphazee09/platform_core.git
cd platform_core
```

2. Set up Environment Variables

Create a `.env` file in the root directory of your project and populate it with your production environment variables. Replace the placeholder values with your actual secrets and configurations:

```
SESSION_SECRET=your_production_session_secret_key
DATABASE_URL=sqlite:///platform.db  # Or your production database URL (e.g., postgresql://user:password@host:port/database_name)
STRIPE_PUBLISHABLE_KEY=pk_live_your_stripe_publishable_key
STRIPE_SECRET_KEY=sk_live_your_stripe_secret_key
```

3. Install Dependencies

Install the required Python packages using pip:

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

4. Database Migrations

Apply database migrations to set up your database schema. Ensure your `FLASK_APP` environment variable is set to `manage.py`.

```
export FLASK_APP=manage.py
flask db upgrade
```

5. Seed Database (Optional)

If you need to populate your database with initial data (e.g., admin users), run the seeding script:

```
python seed.py
```

6. Run the Application with Gunicorn

For production, it is recommended to use a production-ready WSGI HTTP server like Gunicorn. You can run the application using the following command:

```
gunicorn -w 4 -b 0.0.0.0:5000 main:app
```

- `-w 4`: Specifies 4 worker processes. Adjust this based on your server's CPU cores and expected load.
- `-b 0.0.0.0:5000`: Binds the application to all network interfaces on port 5000. You might want to bind it to `127.0.0.1` if you are using a reverse proxy like

Nginx.

- `main:app`: Refers to the `app` object within the `main.py` file.

7. Reverse Proxy (Recommended: Nginx)

In a production environment, it's highly recommended to use a reverse proxy like Nginx in front of Gunicorn. Nginx can handle static files, SSL termination, load balancing, and improve security and performance.

Here's a basic Nginx configuration example (e.g., `/etc/nginx/sites-available/your_project`):

```
server {
    listen 80;
    server_name your_domain.com www.your_domain.com;

    location / {
        proxy_pass http://127.0.0.1:5000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /static/ {
        alias /home/ubuntu/platform_core/static/;
    }

    location /uploads/ {
        alias /home/ubuntu/platform_core/uploads/;
    }
}
```

After creating the Nginx configuration, enable it and restart Nginx:

```
sudo ln -s /etc/nginx/sites-available/your_project /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl restart nginx
```

8. Process Management (Recommended: Systemd)

To ensure Gunicorn runs continuously and restarts automatically on server reboot or crashes, use a process manager like Systemd.

Create a Systemd service file (e.g., `/etc/systemd/system/your_project.service`):

```
[Unit]
Description=Gunicorn instance for your_project
After=network.target

[Service]
User=ubuntu
Group=www-data
WorkingDirectory=/home/ubuntu/platform_core
Environment="PATH=/usr/bin/python3:/home/ubuntu/platform_core/venv/bin"
ExecStart=/home/ubuntu/platform_core/venv/bin/gunicorn -w 4 -b 127.0.0.1:5000
main:app
Restart=always

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

- `User` and `Group`: Set these to appropriate user and group on your server.
- `WorkingDirectory`: Path to your project directory.
- `Environment="PATH=..."`: Ensure the correct Python and Gunicorn paths are included, especially if using a virtual environment.
- `ExecStart`: The command to start Gunicorn. Adjust the path to `gunicorn` if you installed it in a virtual environment.

Reload Systemd, enable, and start your service:

```
sudo systemctl daemon-reload
sudo systemctl start your_project
sudo systemctl enable your_project
```

9. Security Considerations

- **Firewall:** Configure your firewall (e.g., `ufw`) to allow incoming connections only on necessary ports (e.g., 80 for HTTP, 443 for HTTPS).
- **SSL/TLS:** Implement SSL/TLS certificates (e.g., using Certbot with Let's Encrypt) for HTTPS to encrypt traffic.
- **Secrets Management:** Do not hardcode sensitive information. Use environment variables or a dedicated secrets management solution.
- **Logging:** Set up proper logging for your application and server to monitor performance and troubleshoot issues.

This guide provides a general overview. Specific configurations may vary based on your server environment and specific requirements.