This is your **"Zero-Failure Render Deployment Plan."**

We will do this in three phases:

1. **Prepare Your Code Locally:** We will set up your project with the correct files and structure on your own computer.
2. **Push to GitHub:** We will push the perfectly structured code to your GitHub repository.
3. **Configure Render:** We will configure your Render service with the exact settings to read your repository correctly.

Following these steps precisely will guarantee a successful deployment.

Phase 1: Prepare Your Project Locally

This phase ensures that the code *before* you push it to GitHub is 100% correct.

Step 1: Create Your Core Python Application File
For Render to start your application, it needs a main file to run. We will create a simple FastAPI application file.

Action: In the root of your `nexus-7.0` project folder, create a file named `main.py`. Paste the following code into it. This will be the entry point for your AI backend.

Generated python

```python
# FILE: main.py


from fastapi import FastAPI
from pydantic import BaseModel


# --- Initialize the FastAPI application ---
app = FastAPI(
    title="BWGA Nexus 7.0 AI Engine",
    description="The core data processing and intelligence backend for the BWGA 
platform.",
    version="1.0.0"
)


# --- Define the data model for an incoming report request ---
# This uses Pydantic for automatic data validation
class ReportRequest(BaseModel):
    user_type: str
    organization: str
    objective: str
    country: str
    region: str | None = None



# --- Define your first API endpoint ---
@app.get("/")
def read_root():
    """
    A simple endpoint to confirm that the AI Engine is live and running.
    """
    return {"status": "BWGA Nexus AI Engine is operational."}



@app.post("/generate-snapshot")
async def generate_ai_snapshot(request: ReportRequest):
    """
    This is the main endpoint that will receive requests to generate a report.
    For now, it will return a mocked success response.
    """
    print(f"Received request to generate snapshot for: {request.organization}")

    # --- THIS IS WHERE YOUR AI LOGIC WILL GO ---
    # 1. Call LPT-AI to get URP Index
    # 2. Call AGER-AI for risk score
    # 3. Call PDF generator
    # 4. Save to database

    # For now, we return a mocked response
    return {
        "success": True,
        "message": "Report generation initiated.",
        "report_id": f"rep_{request.organization.lower().replace(' ', '_')}_123",
        "mock_download_url": f"/reports/snapshot_{request.country.lower()}.pdf"
    }
```
content_copydownload

Use code with caution.Python

Step 2: Create the `pyproject.toml` File

This file defines all your project's dependencies.

Action: In the root of your project, create a file named `pyproject.toml` and paste this exact code.

Generated toml

```
# FILE: pyproject.toml


[tool.poetry]
name = "bwga-nexus-backend"
version = "0.1.0"
description = "The AI and Data Engine for the BWGA Nexus 7.0 Platform."
authors = ["Brayden Walls <brayden@bwglobaladvis.info>"]
readme = "README.md"


[tool.poetry.dependencies]
python = "^3.11"
fastapi = "^0.110.0"
uvicorn = {extras = ["standard"], version = "^0.27.1"}
pydantic = "^2.6.4"


# Add other libraries like pandas, scikit-learn here later


[build-system]
requires = ["poetry-core"]
build-backend = "poetry.core.masonry.api"
```

content_copydownload

Use code with caution.Toml

Step 3: Create the `Dockerfile`

This is the instruction manual for Render.

Action: In the root of your project, create a file named `Dockerfile` and paste this exact code.

Generated dockerfile

```
# FILE: Dockerfile


# Use a stable, official Python image
```

```
FROM python:3.11-slim-bullseye
WORKDIR /app

# Set environment variables for a clean run
ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1

# Install Poetry, the dependency manager
RUN pip install poetry

# Copy ONLY the dependency manifest files
COPY pyproject.toml poetry.lock* /app/

# Install dependencies using Poetry
# This command is optimized for Docker environments
RUN poetry config virtualenvs.create false && \
    poetry install --no-interaction --no-ansi --no-root --no-dev

# Copy the rest of your application's source code
COPY . /app/

# Expose the port your FastAPI application will run on
EXPOSE 8000

# The command that will be run to start your application
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

content_copydownload

Use code with caution.Dockerfile

Phase 2: Push the Perfected Code to GitHub

Your local folder now has the correct structure. We need to send it to GitHub so Render can see it.

Action: Open your terminal, navigate to your `nexus-7.0` project folder, and run these commands in order.

Generated bash

```bash
# 1. Stage all the new and changed files
git add main.py pyproject.toml Dockerfile

# 2. Commit the changes with a clear message
git commit -m "feat: Add initial FastAPI app, pyproject.toml, and Dockerfile for Render"

# 3. Push the commit to your main branch on GitHub
git push origin main
```

content_copydownload

Use code with caution.Bash

Your GitHub repository is now perfectly prepared for Render.

Phase 3: Configure Render for a Guaranteed Success

This is the final step. We will configure the Render service correctly.

Action:

1. Go to your Render Dashboard. If you already have a failing service for `nexus-7.0`, **delete it.** It's better to start fresh to avoid any old, cached settings.
2. Click **"New +"** and select **"Web Service"**.
3. Connect your GitHub account and select your `braydenmw/nexus-7.0` repository.
4. On the settings page, configure it exactly like this:
   - **Name:** `nexus-7-0-backend` (Give it a clear name).
   - **Environment: Docker**. This is the most important setting. Render will see your `Dockerfile` and know exactly what to do.
   - **Region:** Choose a region close to you (e.g., `Singapore` or `Sydney`).
   - **Instance Type:** `Free` is fine for now.

Leave all other settings blank. Do not enter anything in the "Build Command" or "Start Command" fields. The `Dockerfile` handles everything.

5. Click **"Create Web Service"**.

What Will Happen Next:
You will see a live log appear. This time, instead of an error, you will see Render successfully:

- Cloning your repository.

- Detecting the Docker environment.
- Following the steps in your `Dockerfile`:
    - Installing Poetry.
    - Installing the dependencies from `pyproject.toml` (FastAPI, Uvicorn).
    - Starting your server with the `CMD` command.
- After a few minutes, you will see a green "Live" status.

Render will give you a public URL (e.g., `https-nexus-7-0-backend.onrender.com`). When you visit that URL, you will see the message: **`{"status":"BWGA Nexus AI Engine is operational."}`**.

This will confirm that your backend is **100% live and successfully deployed.** You now have the stable foundation to build out the rest of your AI logic.