Development Journey: Miniature Invoicing Module

This document outlines the step-by-step process I followed to build the Miniature Invoicing Module on the Frappe Framework.

Phase 1: Server and Environment Setup

This was the foundational and most challenging part of the project. I chose to use a **Google Cloud Platform (GCP)** free tier server running **Ubuntu 22.04** to simulate a real-world deployment.

Problem 1: Out of Memory Errors (SIGKILL: 9)

- **Cause:** The server's default RAM was insufficient for the installation process.
- **Solution:** I created a 2GB swap file to provide additional virtual memory, which allowed the installation to complete successfully.

Problem 2: Out of Disk Space

- **Cause:** The default hard drive size (8 GB) was too small for the Frappe and ERPNext installations.
- **Solution:** I resized the server's hard drive to 30 GB to provide enough storage space.

Problem 3: Missing Dependencies

- **Cause:** The honcho and nano commands were missing, and the pip environment was not correctly configured.
- **Solution:** I manually installed the necessary packages and configured the Python virtual environment to ensure all Frappe dependencies were met.

Phase 2: Application Core Development

With the server running, I focused on building the core of the application using Frappe's tools.

Data Modeling

I used Frappe's DocType system to define the application's data structure, modeling relationships between key entities:

- **Customer**: A master DocType to store client information.
- **Invoice Item**: A child DocType for individual line items.
- **Invoice**: The main transactional DocType that links to a customer and contains a table of line items.

Backend Business Logic

To automate the calculations, I implemented a Python function that runs on the server.

- I used the validate hook, a key Frappe feature, to ensure the calculation runs just before an invoice is saved.
- The function iterates through each Invoice Item to calculate its amount and then sums these values to compute the grand_total.

Phase 3: User Interface and Showcase

The final phase involved making the application presentable and showcasing my understanding of the Frappe platform.

Custom UI Components

- **Dashboard**: I built a custom dashboard with number cards and a donut chart to provide a quick, visual overview of key metrics.
- **Permissions**: I configured a custom role (Invoice Viewer) to demonstrate Frappe's powerful role-based permissions system, limiting a user to only read invoices.
- **Website**: I created a public-facing website