**DigiServe Development Plan**

**Phase 1: Foundation Setup (Week 1)**

**Objective:** Finalize the basic structure and core functionalities to enable sales input, roles, and stock management.

1. **Finalize Environment Setup:**
   * Ensure **Django** is installed and running.
   * Verify **MS SQL Server** or **SQLite** as the database.
   * Set up **Git version control** to track changes and development.
2. **App Structure:**
   * sales app is already created and will handle core features (sales, stock, users, and reporting).
   * **User Roles**: Use Django's built-in authentication for managing roles (waitstaff, manager, admin).
   * **Database Models:**
     + Define models for:
       - Sale (transaction with item, quantity, user, total amount)
       - Stock (items with quantities and stock entries)
       - User (extend Django’s User model with roles)
       - Report (generated reports for admin and managers)
     + Example:

python

Copy code

# sales/models.py

from django.db import models

from django.contrib.auth.models import User

class Item(models.Model):

name = models.CharField(max\_length=100)

price = models.DecimalField(max\_digits=10, decimal\_places=2)

stock = models.IntegerField(default=0)

class Sale(models.Model):

item = models.ForeignKey(Item, on\_delete=models.CASCADE)

quantity = models.PositiveIntegerField()

total\_amount = models.DecimalField(max\_digits=10, decimal\_places=2)

user = models.ForeignKey(User, on\_delete=models.CASCADE)

timestamp = models.DateTimeField(auto\_now\_add=True)

1. **Initial Templates & Views:**
   * Create basic views for:
     + Entering a sale.
     + Displaying a list of sales.
     + Basic stock management.
   * Start with basic HTML and ensure the UI is responsive (for mobile and desktop).
2. **Authentication and Role Management:**
   * Implement user authentication with Django’s built-in auth system.
   * Define permissions for each role (waitstaff, manager, admin).

**Deliverables:**

* Database models for Sale, Stock, and User.
* Basic sales input form.
* User authentication with roles (waitstaff, admin, manager).

**Phase 2: Sales Management & Reporting (Weeks 2-3)**

**Objective:** Build the core sales functionality and real-time reporting for managers.

1. **Sales Input and Display:**
   * Create a view and form for waitstaff to input sales.
   * Display sales on the dashboard for review.
   * **Real-time Updates:** Integrate **Socket.IO** for real-time updates to the manager's dashboard.
2. **Stock Management:**
   * Develop stock management views for admins/managers to add incoming stock.
   * Automatically update stock levels after each sale.
   * Notify managers/admins of low stock or discrepancies.
3. **Reporting System:**
   * Build out the reporting views for managers and admins.
   * Generate reports based on daily, weekly, monthly sales.
   * Show financial calculations (profit, loss, etc.).
4. **UI Enhancements:**
   * Ensure the app is responsive and mobile-friendly.
   * Use **Bootstrap** or **Tailwind CSS** to speed up styling.

**Deliverables:**

* Functional sales input and real-time updates.
* Stock management system.
* Basic reporting for managers and admins.
* Responsive and user-friendly UI.

**Phase 3: Financial Calculations & Security (Weeks 4-5)**

**Objective:** Implement financial logic and secure the application.

1. **Financial Calculations:**
   * Add logic to calculate profit, loss, and other business metrics.
   * Build backend logic for generating financial statements (e.g., gross profit, net profit).
   * Example calculation in the Sale model:

python

Copy code

class Sale(models.Model):

# ...fields

@property

def gross\_profit(self):

return self.total\_amount - (self.item.cost\_price \* self.quantity)

1. **Security Enhancements:**
   * Implement security best practices:
     + Use **Django’s CSRF protection**.
     + Enforce strong password policies.
     + Secure endpoints with role-based access control.
2. **Optimize Database Queries:**
   * Use Django's **QuerySet API** to optimize reporting and data retrieval.
   * Index key fields to ensure quick lookups.
3. **Prepare for Deployment:**
   * Start containerizing the app with **Docker** for easier deployment.
   * Test the application in different environments.

**Deliverables:**

* Accurate financial reports (profit, loss, etc.).
* Security improvements (role-based access, CSRF protection).
* Optimized queries and efficient data retrieval.

**Phase 4: Testing, Feedback & Deployment (Week 6)**

**Objective:** Prepare the app for production by testing, iterating, and deploying.

1. **Testing:**
   * Thoroughly test all features (sales, stock, reports) for bugs and performance issues.
   * Perform user acceptance testing (UAT) with potential users (waitstaff, managers).
2. **Gather Feedback:**
   * Gather feedback from real users and make necessary adjustments.
   * Optimize for user experience and workflow.
3. **Deployment:**
   * Deploy the application on **AWS**, **Azure**, or another cloud service.
   * Use **Docker** for deployment to ensure consistency across environments.
   * Monitor the app for performance and scalability issues.
4. **Write Documentation:**
   * Write comprehensive documentation for the app.
   * Include setup instructions, how-to guides, and developer documentation.

**Deliverables:**

* Fully tested and deployed application.
* Comprehensive documentation.
* Final feedback from users.

**Optional: Phase 5 - Future Enhancements**

Once the prototype is stable and meets client expectations, additional features can be added:

* **Advanced Reporting:** Use charts for data visualization (e.g., using **Chart.js**).
* **Integrations:** Integrate with external APIs (e.g., payment gateways).
* **Mobile/Desktop Apps:** Develop native apps using **React Native** or **Electron**.
* **Analytics:** Implement advanced analytics for business insights.

**Project Directory Structure:**

Here's the proposed structure after initial setup:

txt

Copy code

DigiServe/

├── DigiServe/

│ ├── \_\_pycache\_\_/

│ ├── \_\_init\_\_.py

│ ├── asgi.py

│ ├── settings.py

│ ├── urls.py

│ └── wsgi.py

├── sales/

│ ├── migrations/

│ ├── templates/

│ │ └── sales/

│ │ └── sales\_list.html

│ ├── \_\_init\_\_.py

│ ├── admin.py

│ ├── apps.py

│ ├── models.py

│ ├── tests.py

│ ├── urls.py

│ └── views.py

├── venv/

├── db.sqlite3

├── manage.py

├── requirements.txt

└── Dockerfile

Let me know when you'd like to dive deeper into a specific feature or start coding the next phase! 🔥