**Project Description**

**Project Name: TrimQ**

**Overview:**

**TrimQ** is a lightweight, admin-facing web application designed to manage customer queues in a barbering shop. The system enables barbershop clerks to register walk-in customers, assign them to barbers, and track service progress — all through a clean and intuitive interface. TrimQ improves customer experience by streamlining operations and reducing confusion over who’s next in line. The system is designed with scalability in mind, allowing it to serve multiple branches with ease.

**Core Features (MVP)**

**Customer Intake & Queue Management**

1. **Customer Registration**:
   * Admin inputs **customer name** and **phone number** when they arrive.
   * New customers are stored in the system for future reference.
2. **Service Selection**:
   * Admin selects the requested service (e.g., *shave*, *down cut*, *fade*).
3. **Automatic Queue Placement**:
   * Once the details are saved, the customer is added to the **waiting queue**.
   * The system orders the queue by timestamp (first-come, first-served).
4. **Assigning Barbers**:
   * Admin can assign customers to a specific barber once they’re ready.
   * The system updates the customer’s status from *waiting* → *assigned* → *completed*.
5. **Queue Display Screen**:
   * A dedicated screen shows the current queue in real-time.
   * Useful for the shop’s waiting area to let customers see their turn.

**Architecture and Tech Stack**

**Backend**

* **Flask** – lightweight, flexible web server framework.
* **Psqcopg2** – **PostgreSQL database adapter** for the Python programming language.
* **PostgreSQL** – production-ready, robust relational database system.

**Frontend**

* **HTML & Jinja2 Templates** – for rendering dynamic views server-side.
* **CSS** – basic custom styles (may use Bootstrap for simplicity).
* **JavaScript (vanilla)** – minimal use for refreshing the public queue screen.

**Environment & Config**

* .env file for secret keys and database URLs.
* Secure defaults with room for Dockerization in the future.

**Deployment Options**

* Designed for easy deployment to:
  + Local hosting on **localhost**

**Multi-Branch Support**

* Each customer is registered under a **branch name**.
* Filtering and queue display per branch is supported.

**Development Approach (Option A: Ease of Development)**

* Focused on building an MVP quickly and cleanly using Flask.
* PostgreSQL chosen for scalable, robust data storage.
* UI designed to be functional, admin-friendly, and responsive on tablets/laptops.