**Foodbank Management Backlog**  
  
**Hardware & Software Requirements**

a. Hardware Requirements

* Tablets or Smartphones – Devices capable of scanning IDs.
* Barcode Scanners – Used for validating ID information.
* Label Printer – Generates tickets for registrants.

b. Software Requirements

* Cloud Database (MySQL, SQL Server, Oracle, etc.) – Organizes, stores, and manages information efficiently.
* Real-time Sync on Microsoft 365 – Ensures multiple devices can update simultaneously, reducing wait times.
* OCR (Optical Character Recognition) Software – Scans passports or non-barcoded IDs.
* Export Functionality – Allows data to be exported to platforms such as Microsoft Excel.

**Phase 1 (Winter 2025): Initial Development**

#### **1. System Architecture Setup (Priority: HIGHEST)**

* **Process:**
  + Set up a cloud-based backend using **AWS** to store and manage user data.
  + Configure a **Linux** server to host backend scripts and manage the database.
  + Establish a **MongoDB** database for storing scanned user information.
  + Initialize a **Progressive Web App (PWA)** that allows staff to interact with the system via a web browser on tablets or computers.
* **Acceptance Criteria:**  
  ✅ Development environment is fully functional.  
  ✅ Database connection is established successfully.  
  ✅ PWA is initialized with basic working features.

#### **2. User Authentication System (Priority: HIGH)**

* **Process:**
  + Implement a **secure login system** where staff and admins authenticate using **username and password**.
  + Develop a **One-Time Password (OTP)** system using an **email/SMS API (e.g., AWS SNS, Twilio)** for an added layer of security.
  + Establish **role-based access control (RBAC)** so that different users (staff, providers, admins) have different privileges within the system.
* **Acceptance Criteria:**  
  ✅ Secure login system is operational.  
  ✅ OTP authentication works correctly.  
  ✅ Role-based access control functions properly.

#### **3. Registration Interface (Priority: HIGH)**

* **Process:**
  + Create a **digital registration form** accessible via a web interface.
  + Use **barcode scanning called OCR (Optical Character Recognition) via AWS Textract** to extract key information from ID cards.
  + Implement **address verification** using an external API like **Google Maps API** or a government database to confirm residency.
  + Prevent duplicate registrations by checking stored data against new entries.
* **Acceptance Criteria:**  
  ✅ System captures all required registration details.  
  ✅ Real-time ID verification functions correctly.  
  ✅ Synchronization across multiple devices is seamless.

#### **4. Service Provider Interface (Priority: HIGH)**

* **Process:**
  + Develop a **dashboard** where service providers can view and manage registered users.
  + Implement **ticket validation** by generating a **unique barcode or QR code** on printed labels.
  + Scan tickets using barcode scanners at distribution points to ensure one redemption per household.
  + Maintain **service tracking records** in the database to monitor food distribution.
* **Acceptance Criteria:**  
  ✅ Dashboard updates in real-time.  
  ✅ Ticket validation system works efficiently.  
  ✅ Service tracking maintains accurate records.

#### **5. Admin Dashboard (Priority: MEDIUM)**

* **Process:**
  + Create an **admin control panel** to manage users, review logs, and generate reports.
  + Implement **reporting tools** (such as AWS QuickSight or Tableau) to track food distribution and user statistics.
  + Enable **system monitoring** through logging tools (such as AWS CloudWatch) to detect errors or performance issues.
* **Acceptance Criteria:**  
  ✅ Full user management capabilities are available.  
  ✅ Reporting system generates accurate insights.  
  ✅ System monitoring tools function properly.

**Phase 2 (Summer 2025): Enhancement & Testing**

**6. System Testing (Priority: HIGH)**

* **Process:**
  + Conduct **unit testing** for individual components (e.g., ID scanning, ticket generation).
  + Perform **integration testing** to ensure seamless communication between AWS, MongoDB, and the PWA.
  + Run **performance testing** to optimize response times for large volumes of users.
  + Ensure **security testing** by simulating data breaches and validating encryption protocols.
* **Acceptance Criteria:**  
  ✅ All test cases pass successfully with minimal errors.

#### **7. System Optimization**

* **Process:**
  + Optimize database queries to **reduce processing time** for ID scanning and ticket generation.
  + Improve **UI/UX design** for easier navigation and user experience.
  + Identify and **fix bugs** based on user feedback from testing.
* **Acceptance Criteria:**  
  ✅ System runs efficiently with faster processing times.  
  ✅ User interface is more intuitive and user-friendly.  
  ✅ Any identified bugs are resolved.

**Project Timeline (Gantt Chart)**

**February 2025**

Week 1-2:

* Project setup, initial client meeting, requirements documentation.
* System architecture design finalized.

Week 3-4:

* Database setup completed.
* User authentication system development starts.
* Client meeting to review progress.

**March 2025**

Week 1-2:

* Registration interface development.
* Continued authentication system improvements.
* Begin service provider interface.

Week 3-4:

* Admin dashboard development.
* Requirements documentation refinement.
* Drafting initial poster for presentations.

**April 2025**

Week 1-2:

* System integration and initial testing.
* Client review and feedback session.

Week 3-4:

* Phase 1 completion and deployment.
* Final presentations and handoff.

**Summer 2025 (May-August)**

* May: Planning for enhancements and advanced testing.
* June: Security improvements, UI/UX refinements, integration testing.
* July: Performance testing, bug fixes, and system optimization.
* August: Final testing, documentation updates, production deployment.

**Technical Approach**

* Frontend: Progressive Web App (PWA).
* Backend: Node.js.
* Database: MongoDB.
* Authentication: Username/Password + Daily OTP.
* User Roles:

1. Registration Staff
2. Service Providers
3. System Administrators

\* Backlog conducted by Mohamed Mezahi, Abdulmajeed Almaweri, Ali Almaweri, Anika Tahrin