Coffee Supply Chain Management System - Functional Specification

1. System Overview

The Coffee Supply Chain Management System is a comprehensive web application designed to streamline inventory tracking, operational workflow optimization, and data management for coffee roasteries and retail shops.

2. User Roles and Permissions

2.1 User Types

- **Roastery Owner**: Full system access, user management
- **Roaster**: Manages roasting processes and orders
- **Shop Manager**: Manages retail inventory and shop operations
- **Barista**: Basic inventory viewing and updates

2.2 Role-Based Access Control

Detailed access permissions for each role are enforced throughout the application.

3. Database Structure

3.1 Core Tables

Users Table

```
- id (Primary Key)
- username (Unique)
- password (Hashed)
- role (Enum: roasteryOwner, roaster, shopManager, barista)
- isActive (Boolean)
- isPendingApproval (Boolean)
- createdAt (Timestamp)
```

Shops Table

```
- id (Primary Key)
- name
- location
- isActive (Boolean)
- desiredSmallBags (Integer)
- desiredLargeBags (Integer)
- createdAt (Timestamp)
```

Green Coffee Table

```
- id (Primary Key)
- name
- producer
- country
- currentStock (Decimal)
- minThreshold (Decimal)
- isActive (Boolean)
- grade (Enum: Specialty, Premium, Rarity)
- createdAt (Timestamp)
```

Orders Table

```
- id (Primary Key)
- shopId (Foreign Key)
- greenCoffeeId (Foreign Key)
- smallBags (Integer)
- largeBags (Integer)
- status (Enum: pending, roasted, dispatched, delivered)
- createdAt (Timestamp)
- createdById (Foreign Key)
- updatedById (Foreign Key)
```

Retail Inventory Table

```
- id (Primary Key)
- shopId (Foreign Key)
- greenCoffeeId (Foreign Key)
- smallBags (Integer)
- largeBags (Integer)
- updatedById (Foreign Key)
- updatedAt (Timestamp)
- updateType (Enum: manual, dispatch)
- notes (Text)
```

Roasting Batches Table

```
- id (Primary Key)
- greenCoffeeId (Foreign Key)
- plannedAmount (Decimal)
- actualAmount (Decimal)
- roastingLoss (Decimal)
- status (Enum: planned, in_progress, completed)
- roastedAt (Timestamp)
- smallBagsProduced (Integer)
- largeBagsProduced (Integer)
- createdAt (Timestamp)
```

3.2 Relationships

- Users can be assigned to multiple shops (Many-to-Many through UserShops)
- Each shop can have multiple inventory records

- Orders are linked to shops and specific coffee types
- Roasting batches track the processing of green coffee

4. Core Features

4.1 Authentication & Authorization

- Secure login/logout functionality
- Password hashing using scrypt
- Session management with PostgreSQL session store
- Role-based access control

4.2 Inventory Management

- Real-time tracking of coffee stock levels
- Automatic alerts for low inventory
- Batch tracking and management
- Historical inventory data

4.3 Order Processing

- Order creation and tracking
- Status updates (pending → roasted → dispatched → delivered)
- Automatic inventory updates on delivery
- Order history and reporting

4.4 Roasting Operations

- Roasting batch planning
- Production tracking
- Yield calculations
- Loss monitoring

4.5 Shop Management

- Multi-shop support
- Individual shop inventory tracking
- Desired stock level management
- Shop-specific reporting

5. Technical Implementation

5.1 Frontend

- React with TypeScript
- Shadcn UI components
- TanStack Query for data fetching
- Zod for validation
- Zustand for state management

5.2 Backend

- Node.js with Express
- Drizzle ORM for database operations
- PostgreSQL database
- Session-based authentication
- RESTful API endpoints

5.3 Security

- CORS protection
- Password hashing
- Session management
- Input validation
- Role-based access control

6. API Endpoints

Authentication

- POST /api/login
- POST /api/logout
- POST /api/register
- GET /api/user

Inventory Management

- GET /api/inventory
- POST /api/inventory/update
- GET /api/inventory/history

Orders

- GET /api/orders
- POST /api/orders
- PATCH /api/orders/:id/status

Roasting

- GET /api/roasting/batches
- POST /api/roasting/batches
- PATCH /api/roasting/batches/:id

Shop Management

- GET /api/shops
- POST /api/shops
- PATCH /api/shops/:id

7. Data Validation

All data validation is handled using Zod schemas defined in shared/schema.ts, ensuring type safety and data integrity throughout the application.

8. Error Handling

Comprehensive error handling is implemented across all layers of the application, with appropriate error messages and status codes returned to the client.

9. Monitoring and Logging

Detailed logging is implemented throughout the application to track:

- Authentication attempts
- Inventory changes
- Order status updates
- Roasting batch progress
- Error conditions

10. Future Enhancements

- Advanced analytics dashboard
- Mobile application support
- Integration with roasting equipment
- Automated order scheduling
- Customer portal for order tracking