Back-end Engineer Technical Assessment | GeekyAir

Simplified items and orders Management System
Objective: Design and implement a simplified internal system API to manage items and orders.

Functional Requirements

1- Item/Inventory Management:

- Super Admin / Managers can perform CRUD operations on items (e.g., name, description, price, category, expiry date, stock quantity), While waiters can only view non-expired items.
- Implement filtering by category (others, food, beverages) and sorting by name, price, expiry date, or total stock value (price * quantity), supporting ascending/descending order for numeric and date fields.
- Expired or unavailable items cannot be added to orders. Admins/managers are also notified by email 5 days before an item expires, and again on the expiry date with the quantity/details of the expiring items.

2- Order Management:

- The cashier(s) can manage customer orders: add or remove non-expired items, set the quantity, mark the order as complete, and assign the order to the specific waiter who served the customer.
- Super Admin / Managers can view and manage all orders, including order details and status.
- The system must calculate the total cost. Orders can include multiple items with quantities. also implement a feature to automatically update the status of orders to "expired". if 4 hours have passed since creation time and they are still pending.

3- User Authentication:

- Implement user roles, and Only authenticated users with the appropriate roles can access specific functionalities as described in this document.
- Super Admin / Managers can view and manage all functionalities, and add cashier(s) / waiter(s) to the system.
- Email verification is required for new user registrations (cashiers/managers/waiters), and implement a password reset feature accessible upon user request.

4- Waiter Commission Report endpoint:

- Implement an API endpoint that accepts a date range (`startDate`, `endDate`) and an optional waiter name (partial match). It returns JSON by default. “If the optional parameters `export=true` and `format=csv` are passed, it returns a CSV export file”.
- Super Admins, managers, and cashiers can view all data, while waiters can see only their own.
- Use raw SQL to aggregate completed orders in the range, returning the following per waiter: total items sold, items per category, revenue, commissions (Others 0.25 %, Food 1 %, Beverages 0.5 %), and total commission.

5- Item Management via CSV Import/Export:

- Enable Admins/Managers to import/export item data (including all item details and stock levels) via CSV. The import should support creating new items and updating existing ones based on a unique identifier.

Non-functional Requirements

1. The system should be optimized for reading operations, focusing on efficient searching and listing.
2. Security: Ensure that user inputs are validated to prevent SQL injection or other potential security threats.

Technical Requirements

1. Programming: The task shall be implemented using NodeJs with the Express.js framework.
2. Database: Use a relational database system (e.g., PostgreSQL, MySQL), Also use **Sequelize**.
3. API: Implement a RESTful API to support all the above operations.
4. Error Handling: The system should gracefully handle errors and provide meaningful feedback.
5. Testing: Develop focused automated tests for two critical API endpoints, covering various scenarios.

Bonus - Ordered descending by value (Optional)

1. Integrate Gen AI API to create 3 promo messages (SMS/social media), and Triggered emails to admins upon: newly added 'Food' items to the system (min. price 200), or 500+ sales in 10 days.
2. Items expiring within 20 days receive an automatic 25% discount, showing both prices (original and discounted). Admins/managers are notified and can exclude items/categories.
3. Implement Automated Sales Report Export to Google Drive: Allow Admins/Managers to connect their Google Drive account via OAuth2, enabling the system to automatically export daily/weekly sales reports (e.g., CSV/PDF) directly to a specified Drive folder.
4. Implement Expiring Inventory Reminders via Google Calendar: Integrate with Google Calendar via OAuth2 to automatically add events to a shared calendar for items nearing expiry (e.g., "Use by 25/05: 50 sandwiches").
5. Deploy your API system and share the deployment link; also, attach the link to the README.md file.
6. Implement Swagger (OpenAPI) documentation.

Submission

- **Codebase**: Create a **private Github** repository and invite (backend@geekyair.com) as a collaborator.
- **Database**: Provide a schema diagram and any necessary setup scripts.
- **Documentation**: Include detailed instructions to set up and run the application. Document the API endpoints with expected inputs/outputs, This API Documentation should be interactive for example **Postman API Documentation** and must be using Postman environment variable for base URL.

Evaluation Criteria

- All functional requirements should be implemented.
- Code Quality: Clear naming conventions, modularity, and comments.
- Database Design: Schema design, normalization, and indexing.
- API Design: Proper use of HTTP methods, status codes, and endpoint structuring.