Functional and Nonfunctional Requirements

Carolina Wings Web Application - Requirements Document

Functional Requirements

---

1. User Management

- The system shall allow users to create an account using an email and password with a verification code upon creation.
- The system shall allow users to log in securely
- The system shall allow users to log out securely.
- The system should allow a variety of roles to be applied to accounts

2. User Stories

- Root user
  - The system shall allow this user to have root access to the system
- Admin
  - The system shall allow a user with this role to view
  - The system shall allow a user with this role access to orders and history of orders
  - The system shall gives a user with this role the ability to make changes to the menu
  - The system shall allow a user with this role the ability to edit prices in the menu
  - The system shall allow a user with this role the ability to add or remove menu items on a per location basis
  - The system shall allow a user with this role to view total sales
  - The system shall allow a user with this role to print reports of sales
- Customer
  - The system shall allow a user with this role to create orders
  - The system shall allow a user with this role to view the menu
  - The system shall allow a user with this role to create an account
  - The system shall allow a user with this role to login with a valid email and password
  - The system shall allow a user with this role to sign up for a weekly newsletter

- The system shall allow a user with this role to reset their password via email

3. Cart & Order Management

- The system shall allow users to add items to the cart
- The system shall allow users to remove items from the cart
- The system shall allow users to add special requests or notes to items in the cart
  - E.g. allergies or memos
- The system shall update item quantities in the cart in real time.
- The system shall display a cart with a live summary with total cost of items
- The system shall show all components of a menu item in the cart (name, cost, desc, image)

4. Payment Processing

- The system shall allow users to checkout using Credit/Debit Cards (Toast most likely or Stripe)
- The system shall allow user to checkout using Apple Pay
- The system shall allow users to checkout using Google Pay
- The system shall allow user to checkout using PayPal
- The system shall allow for secure and tokenized payments
- The system shall securely send payment info to a third party payment processor
- The system shall generate an order based on successful checkout/confirmed payment
- The system shall format the order correctly after generation
- The system and sent to the correct restaurant where it will print.

5. Availability Logic

- The system shall allow for orders to only be placed during open hours. (11-11 f-sat) (11-10 sun-thu)
- The system shall inform customers if ordering is currently unavailable.

7. Admin Dashboard

- The system shall allow an Admin user to view orders
- The system shall allow an Admin user to manage orders
- The system shall allow an Admin user to add menu items
- The system shall allow an Admin user to remove menu items
- The system shall allow an Admin user to update menu items
- The system shall allow an Admin user to edit hours online
- The system shall allow a Customer user to view full menu items

- The system shall allow a Customer user to view the static website content
- The system shall allow a Customer to create an account
- The system shall allow a Customer to still view the menu without an account
- The system shall require a Customer to be logged in to sign up for newsletter

---

Non-Functional Requirements

1. Performance

- The system shall support 20–30 concurrent users
- The system shall maintain menu/cart interactions under 300ms latency.

2. Scalability

- The system shall have backend services and database services scalable using AWS services The system shall be cost optimized
- The system shall have static assets served using S3

3. Security

- The system shall employ HTTPS for all data transmission
- The system shall use JWT-based authentication (tokens stored securely on priv subnet).
- The system shall employ input validation APIs and forms.
- The system shall sanitize APIs and forms
- The system shall employ Role-based access control (RBAC) for admin routes and root routes
- The system shall distinguish customer routes from admin routes as customers should not have access to any admin routes

4. Offline Access

- The system shall contain a site shell and menu cached with S3 to ensure offline access
- The system shall serve static website after hours unless Admin account logs in
- The system shall deploy an instance for the admin
- The system shall send a notification when the system goes down
- The system shall disable order functionality after hours

5. Availability & Reliability

- The system shall maintain 99% uptime
- The system shall use AWS CloudWatch for monitoring
- The system shall alert for downtime/errors.
- The system shall use a SNS service when there is downtime to notify service is down

6. Maintainability

- The system shall be built using React, Spring Boot, PostgreSQL as tech stack
- The system's code shall adhere to SOLID principles
  - SOLID
    -
- The system shall be well-documented
- The system shall maintain a RESTful API design

2. Menu & Item Browsing

- The system shall categorize menu items based on what submenu they belong to
  - The system shall include the submenu Wings
  - The system shall include the submenu Ribs
  - The system shall include the submenu Sandwiches
  - The system shall include the submenu Appetizers
  - The system shall include the submenu Wraps
  - The system shall include the submenu Salads
  - The system shall include the submenu Burgers
  - The system shall include the submenu Side items
  - The system shall include the submenu Premium side items
- The system shall include an overall menu where each item displays: name, short description, price, and image and what subcategory it is
- The system shall include the submenu Menu is fetched from backed (DB)

Tech Stack

- **Frontend:** React
- **Backend:** Spring Boot
- **Database:** PostgreSQL
- **Auth:** JWT
- **Hosting:** AWS (EC2/ECS, S3)

- **Payments:** Stripe, Apple Pay, Google Pay, PayPal
- **Payment Processor**: Toast or other third party (Heartland is currently in use)

(INSERT GRAPH)

5. Order Handling

- Orders are sent to a backend endpoint.
- Orders are printed at the restaurant via ticket printers.
- Admin can view order status: Pending, In Progress, Completed.