HUrricane

System-Wide Requirements Specification

# Introduction

This section of the Software Requirements Specification document (henceforth referred to as SRS) outlines the purpose, scope, definitions of terms and abbreviations, and an overview of the document’s structure for the HUrricane

* 1. **Purpose**

● To define the quality attributes and constraints that the HU-FDS must satisfy to achieve its business goals and operational objectives.

● To capture functional requirements and their behavioral aspects using use case models (detailed in the Use Case Definitions document).

● To specify functional requirements not addressed by use case specifications.

● To evaluate and select from competing design options for the system.

● To assess the feasibility and viability of the proposed HU-FDS.

● To establish service-level requirements for the operational management and scalability of the solution.

* 1. **Scope**

HUrricane is an online food delivery platform designed and developed by PentaCode team for the BBM384 Software Engineering Laboratory course term project under the guidance of the teaching staff. The system aims to connect restaurants, customers, and couriers efficiently, enabling users to browse restaurants, place orders, and manage deliveries seamlessly. HUrricane includes distinct modules for administrators, restaurants, customers (registered and non-registered), and couriers, each tailored to their specific needs. The application emphasizes flexibility, maintainability, and scalability while adhering to the Open-Close Principle, ensuring it is closed for modification and open for extension.

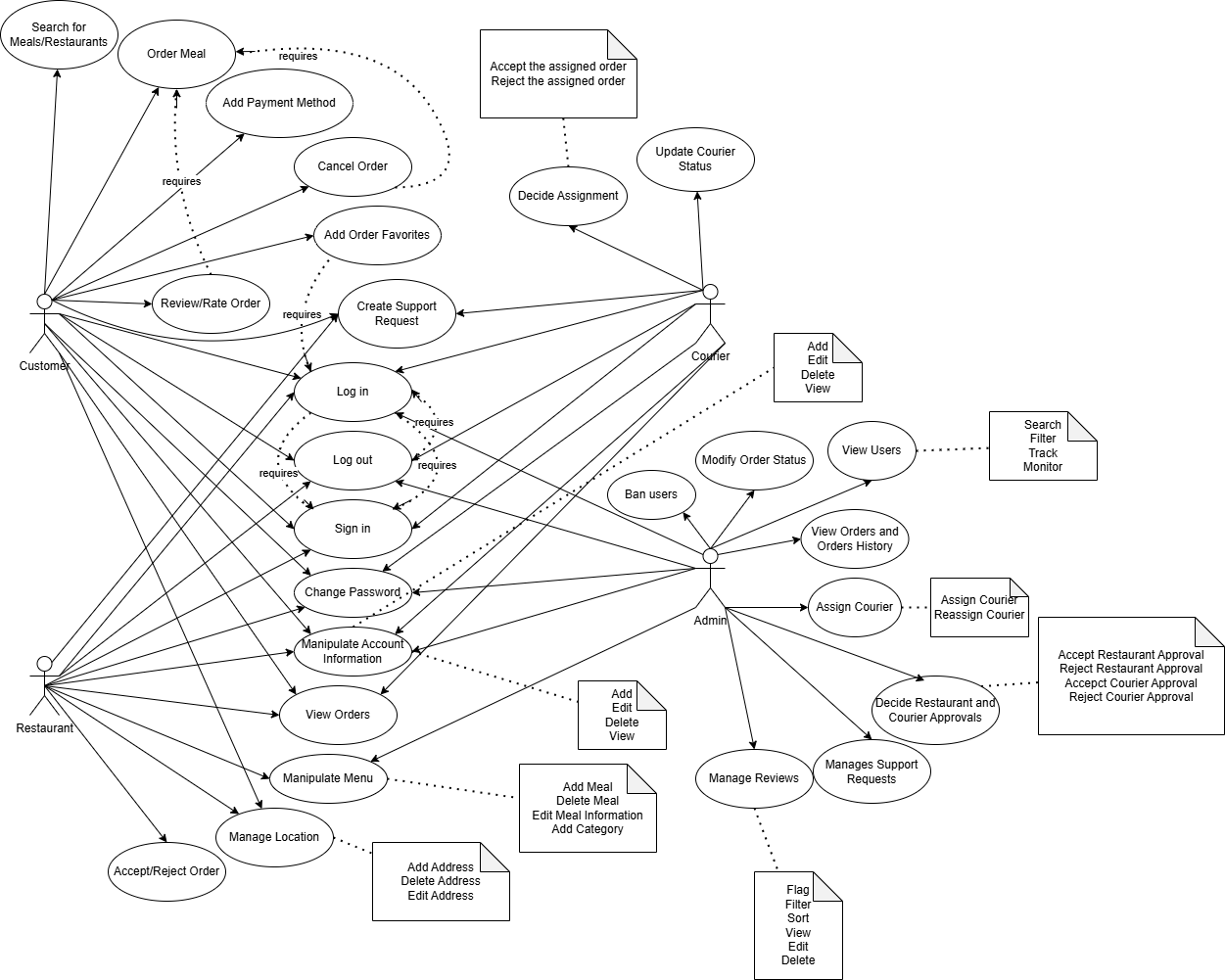
* 1. **Overview**

This document, together with its attachments—Use Case Definitions, Graphical User Interface Design, and Test Case Definitions—specifies the functional requirements (features and functionalities), non-functional requirements (quality attributes), testing criteria, and constraints for the HUrricane development. The system’s requirements are modeled using a use-case-driven approach, with most functional requirements detailed in the Use Case Definitions document attached to this SRS. Section 2, “System-Wide Functional Requirements,” addresses functional requirements.

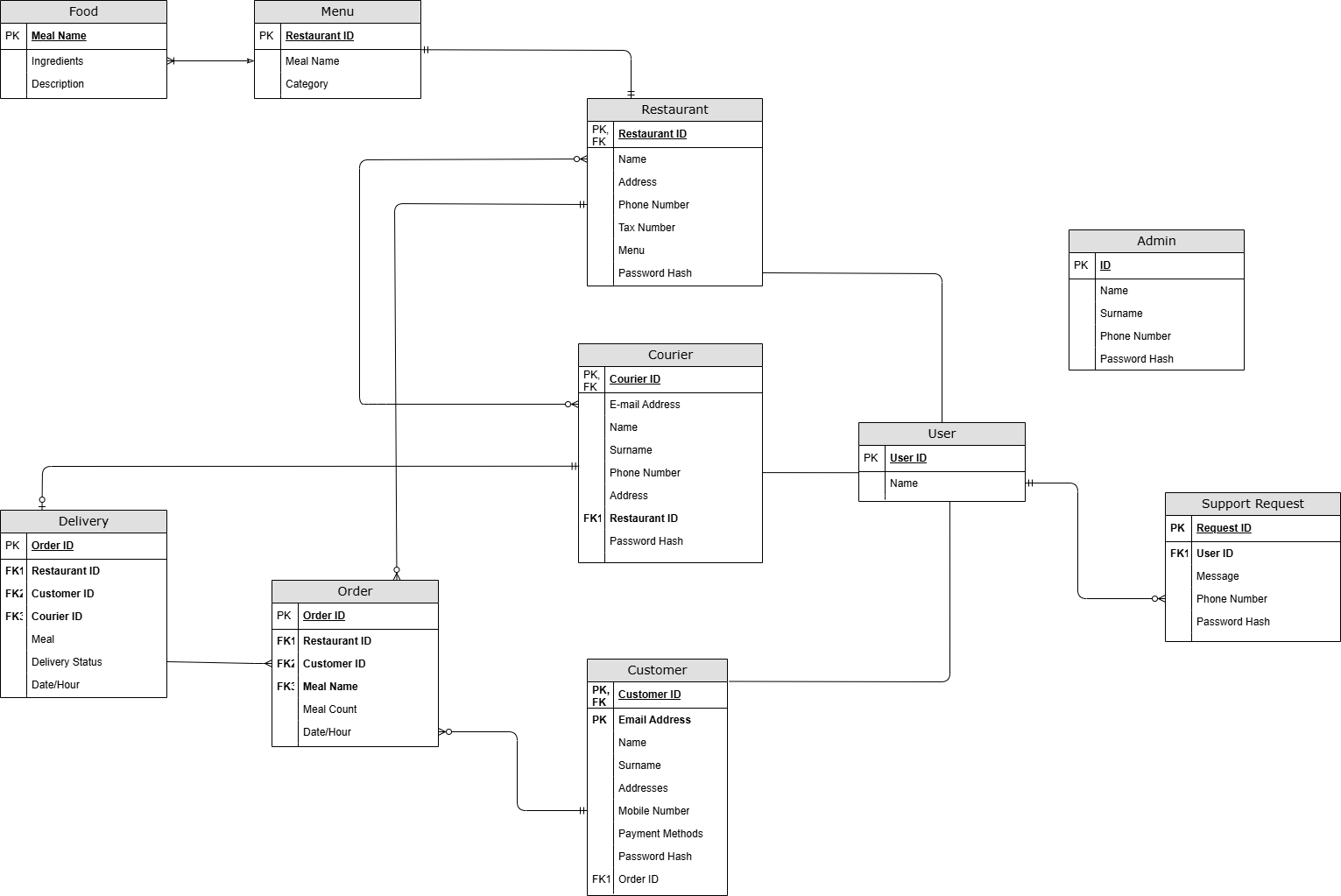
Section 3, “System Qualities,” describes non-functional requirements such as performance, usability, and scalability, presented with both quantitative and qualitative measures. Each requirement in sections 2 and 3 is assigned a unique identifier for traceability, with similar traceability provided for use cases (e.g., UC1.2 refers to the second path item of use case UC1).

Section 4, “System Interfaces,” outlines requirements for user interfaces and external system integrations. Section 5, “Business Rules,” specifies the domain-specific rules and policies the system must follow. Section 6, “System Constraints,” details design, implementation, and deployment limitations. Section 7, “System Compliance,” addresses applicable standards and legal requirements. Finally, Section 8, “System Documentation,” covers requirements for user manuals and supporting documentation.[[1]](#bookmark=id.2tp0asxtetip)

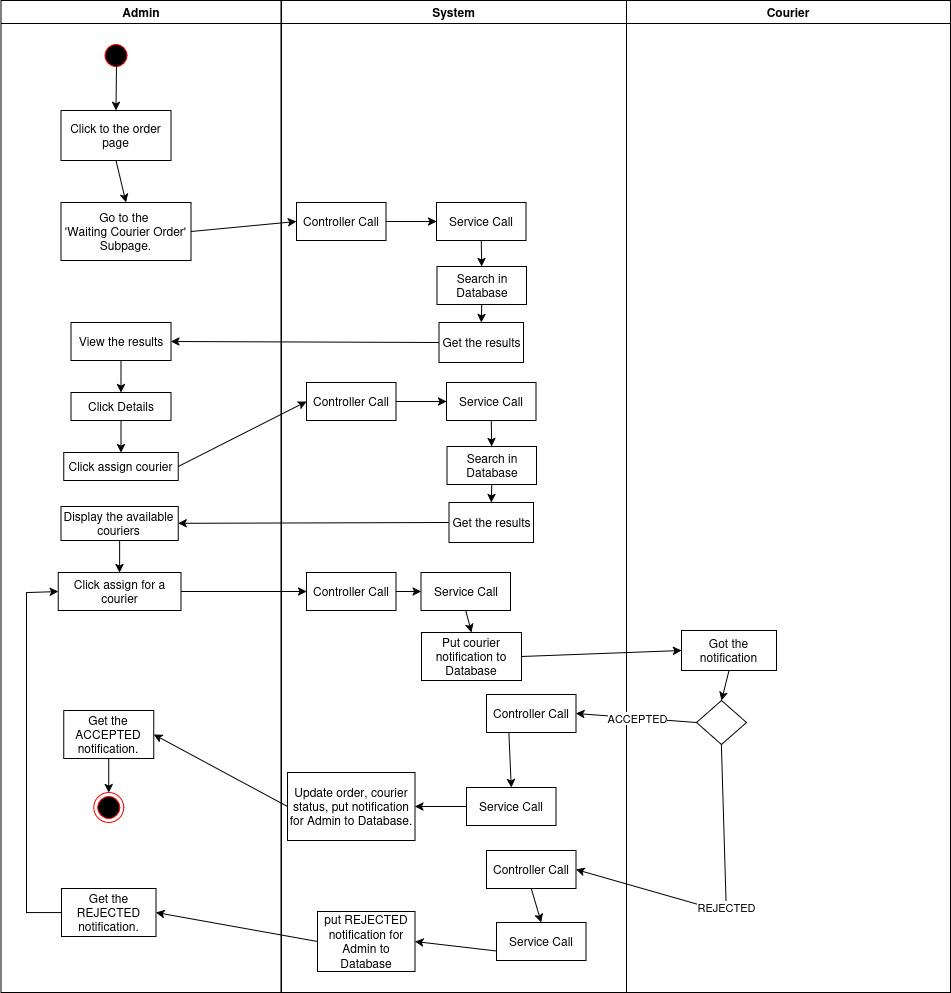
# System-Wide Functional Requirements[[5]](#bookmark=id.o4paeom21mvs)

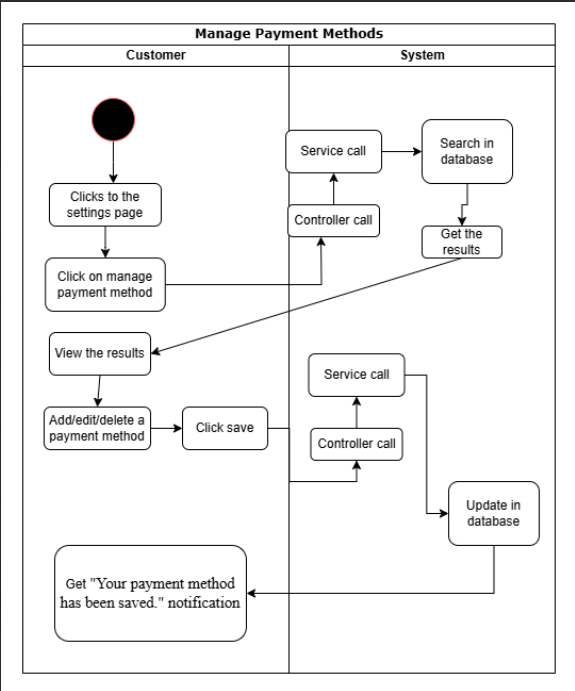
**

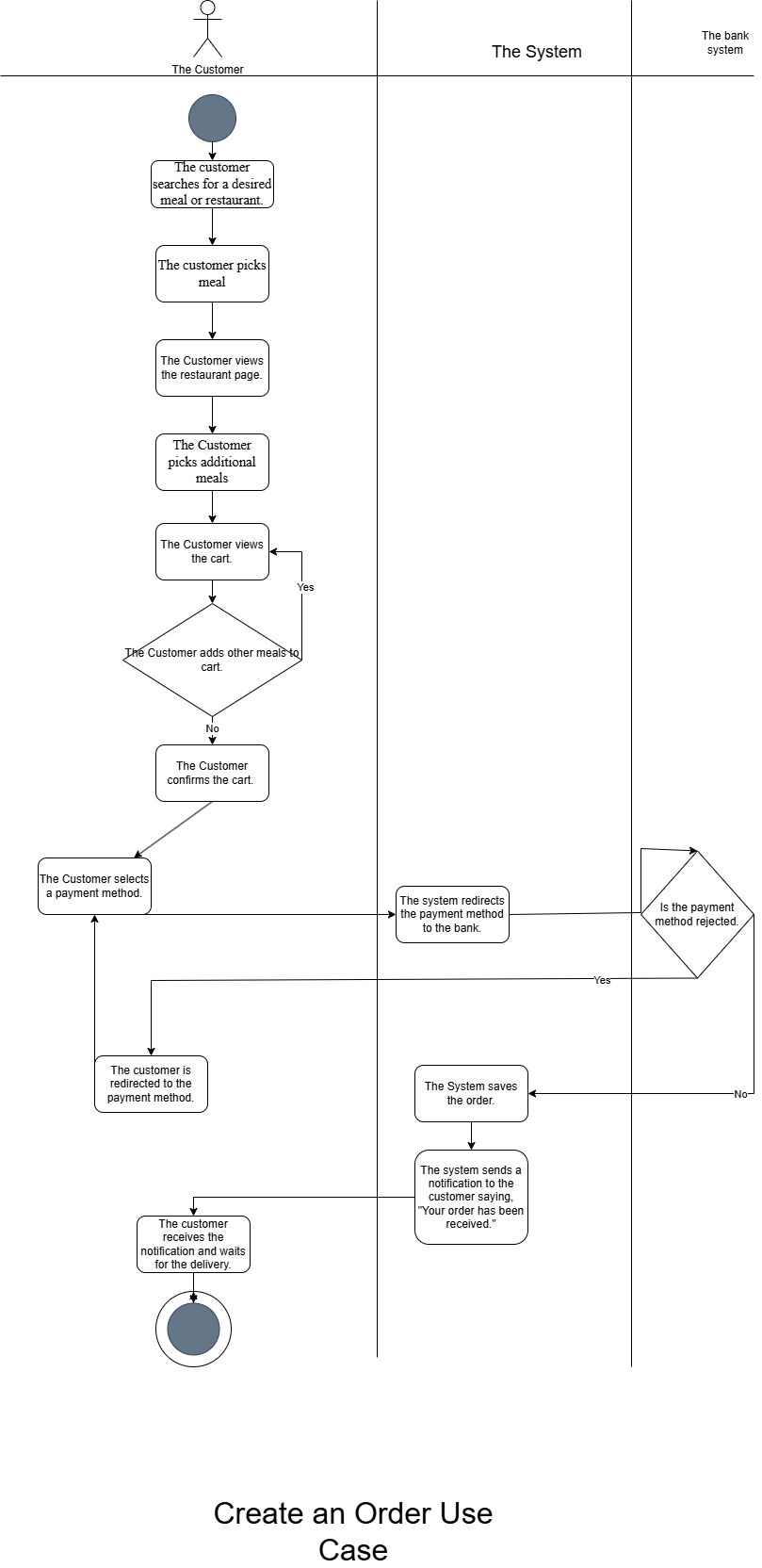
ER Diagram of The System

*Activity Diagram for UC -1*

Assign Courier Activity Diagram

**

****

****

**Restaurant Requirements**

| **Version** | **Feature** | **Description.** |
| --- | --- | --- |
| 1.1 | Apply for Collaboration | The restaurant manager shall submit a **registration form** with a **TC number, business license, and required fields**. |
| 1.2 | Validate Restaurant Application | The system shall validate the submitted documents before approving the restaurant manager. |
| 2.1 | Manage Food Items | The restaurant manager shall be able to **add, edit, update and remove** food items from the inventory. |
| 2.2 | Food Information | Food shall have fields of name, small explanation, choices, image, and primary key etc. Each food item shall have **a price field**, which the restaurant manager can update. |
| 2.3 | Food Category | The system shall allow restaurant managers to **assign food items to categories** (e.g., Burgers, Beverages, Desserts). |
| 2.4 | Menu Information | The restaurant manager shall be able to **create menus** by grouping multiple food items into a single offering. |
| 2.5 | Menu Categories | The restaurant manager shall be able to assign menus to **categories** (e.g., Lunch Menus, Family Combos). |
| 2.6 | Menu Composition | A menu shall contain **multiple food items**, with the ability to specify **required and optional selections** (e.g., a burger menu with a choice of drink and side). |
| 2.7 | Menu Customization | The restaurant manager shall be able to **set default items in a menu** while allowing customers to **customize their selections** (e.g., choose between fries or onion rings). |
| 2.8 | Menu Pricing | The restaurant manager shall be able to set a **price for the entire menu**, which may be lower than the sum of individual items. |
| 2.9 | Menu Image & Description | Each menu shall have an **image and description** to provide customers with more details. |
| 3.1 | Log in | The restaurant manager shall be able to **log in using an email and password**. |
| 3.2 | Password Recovery | The restaurant manager shall have a **password recovery option**. The 6 digit code shall be sent to the restaurant manager’s email and used for password reset. |
| 4.1 | Restaurant Deactivation | If the restaurant is removed, all associated **food items** shall be removed. Also all associated **couriers** for that restaurant shall be removed. |
| 4.2 | Account Deactivation | If the restaurant is removed, the restaurant manager’s account shall be **disabled**. |
| 5.1 | Manage Orders | The restaurant manager shall be able to **accept, update, cancel, and postpone** orders. Each order must have time, restaurant, food, and courier information. |
| 5.2 | Order Status Notifications | The system shall notify **customers and couriers** about order status updates. |
| 5.3 | Generate Order Reports | The restaurant manager shall have access to **detailed order reports** after each completed order. |
| 5.4 | Assign Couriers | The restaurant manager shall be able to **assign couriers** for deliveries. |
| 5.5 | Request Additional Couriers | The restaurant manager shall be able to **request extra couriers** when needed. The system shall assign the couriers for deliveries. |
| 5.6 | View Customer Details | The restaurant manager shall be able to **view details of customers** related to their orders. |
| 5.7 | View Courier Details | The restaurant manager shall be able to **view details of assigned couriers**. |
| 5.8 | View Order History | The restaurant manager shall be able to view the **order history for a specific customer and overall order history**. |
| 5.9 | Order Information | The order shall have detailed **information of the courier, customer and the cart.** |
| 5.10 | Staff Order Management | The restaurant manager shall be able to **update an order’s status to "Waiting," "Preparing," or "On the Courier."** |
| 6.1 | Update Restaurant Information | The restaurant manager shall be able to **update the restaurant name, address, and contact details**. |
| 7.1 | Create Campaigns | The restaurant manager shall be able to **create promotional campaigns and discounts**. |

**Customer Requirements**

| 1.1 | Register | The customer shall be able to **register using a valid email and phone number**. |
| --- | --- | --- |
| 1.2 | Log in | The customer shall be able to **log in using a username and password**. |
| 1.3 | Password Recovery | The customer shall have a **password recovery option**. The 6 digit code shall be sent to the customer’s email and used for password reset. |
| 2.1 | Place Orders | The customer shall be able to **place orders from available restaurants**. |
| 2.2 | Order Information | The order shall have the information of the customer, restaurant and assigned the courier. |
| 2.3 | Select Restaurant | The customer shall be able to **browse and select** a restaurant from the list of available restaurants by name, cuisine or location. |
| 2.3 | Browse Food Items | The customer shall be able to browse food items. |
| 2.4 | View Menu | The customer shall be able to **view the restaurant’s menu**, including food items, descriptions, prices, and availability. |
| 2.5 | Add Items to Cart | The customer shall be able to **add food items and menus** to their cart before placing an order. If the items are out of the stock, the warning error message shall display. |
| 2.6 | Customize Order | The customer shall be able to **customize their order** (e.g., remove ingredients, add extra toppings, choose meal options). |
| 2.7 | Apply Discounts & Promotions | The customer shall be able to **apply valid promotional codes or discounts** before checkout. |
| 2.8 | Review Order | The customer shall be able to **review their selected items, total price, and delivery details** before confirming the order. |
| 2.9 | Select Payment Method | The customer shall be able to **choose a payment method**, including credit card, online payment, or cash on delivery. |
| 2.10 | Confirm Order | The customer shall be able to **finalize the order** by confirming their selection and payment details. |
| 2.11 | Order Confirmation Notification | The system shall **notify the customer** when the restaurant accepts or rejects the order. |
| 2.12 | Estimated Delivery Time | The system shall **display an estimated delivery time** based on restaurant preparation time and courier availability. |
| 2.13 | Order Status Tracking | The customer shall be able to **track their order status** in real time (e.g., "Waiting for Confirmation," "Preparing," "On the Way"). |
| 2.14 | Modify Order (Before Confirmation) | The customer shall be able to **modify or cancel their order** before the restaurant confirms it. |
| 2.15 | Reorder Past Orders | The customer shall be able to **reorder previous meals** from their order history. |
| 2.16 | Request Refund or Report Order issue | The customer shall be able to request refunds for deliveries or report issues. |
| 2.17 | Leave Ratings | The customer shall be able to **rate** both **restaurants** and **couriers** based on service quality. |
| 2.18 | Save favorites | The customer shall be able to save favorite restaurants and food items to be accessed in future. |
| 3.1 | Order Cancellation | The customer shall be able to **cancel an order** within predefined constraints. |
| 3.2.1 | Time-Based Restriction | The customer shall only be able to cancel an order **before the restaurant starts preparing it**. |
| 3.2.2 | Time-Based Restriction | The customer shall only be able to cancel an order if **the order is more than 10 mins late.** |
| 3.3 | Order History Update | Canceled orders shall be **recorded in the customer’s order history**, along with the cancellation reason. |
| 3.4 | Notification on Cancellation | The system shall notify **the restaurant, courier, and customer** when an order is canceled. |
| 4.1 | Access Customer Support | The customer shall be able to **contact customer support through a live chat**. |
| 4.2 | AI Agent | The customer shall be able to **communicate with an AI agent** for assistance. |
| 4.3 | Feedback Submission | The customer shall be able to **submit feedback** about their support experience. |
| 4.4 | Order-Related Assistance | The customer shall be able to **get support regarding orders**, including cancellations, refunds, and status inquiries. |

**Courier Requirements**

| 1.1 | Register | The courier shall be able to **register using a valid motorcycle ID and personal details (e.g., TC number)**. |
| --- | --- | --- |
| 1.2 | Login | The courier shall be able to **login using a valid email and password.** |
| 2.1 | Update Availability | The courier shall be able to **set their availability status** (Available/Busy/Offline) in the system. |
| 2.2 | Automatic Status Update | The system shall automatically mark the courier as **Busy** when they are assigned an order. |
| 2.3 | Notify System of Availability Changes | When a courier updates their availability, the system shall **reflect the changes in real-time** and adjust order assignments accordingly. |
| 2.4 | Contact Customers & Managers | The courier shall be able to **contact customers and restaurant managers** regarding deliveries. |
| 3.1 | Order Assignment | The system shall allow restaurant managers to **assign delivery orders** to available couriers. The order has the minimum and the maximum time it takes to deliver. |
| 3.2 | Courier Order Notification | The system shall notify **the assigned courier** of a new delivery order, including details like customer name, address, restaurant details and items to be delivered. |
| 3.3 | Courier Accept/Reject Order | The courier shall be able to **accept or reject** the assigned delivery order. Rejected orders will be reassigned to other couriers. |
| 3.4 | Courier Order Confirmation | Upon successful delivery, the courier shall **confirm order completion** by entering the customer’s code to the system, triggering the system to update the order status to "Delivered" and notify the customer. |
| 3.5 | Report Delivery Issues | The courier shall be able to **report issues with deliveries to restaurant managers**. |

**Admin Requirements**

| 1.1 | Register | The admin accounts will be provided by the **system’s owner** to the people that work. |
| --- | --- | --- |
| 1.2 | Login | After the user gets the **credentials** from the system it can login using credentials. |
| 2.1 | User Management | The admin shall be able to **create, update, delete, and deactivate user accounts for customers, restaurant managers, and couriers.** |
| 2.2 | View User Profiles | The admin shall be able to **view** detailed user profiles, including account information, registration date, and activity logs. |
| 2.3 | User Account Status Management | The admin shall be able to **suspend or reactivate** user accounts due to policy violations, fraud, or suspicious activity. |
| 2.4 | Search & Filter Users | The admin shall be able to **search and filter** users based on criteria such as name, email, role, status (active/suspended), and registration date. |
| 2.5 | Approve or Reject Restaurant Registrations | The admin shall be able to **review restaurant registration** requests, including submitted documents, and either approve or reject the application. |
| 2.6 | Approve or Reject Courier Applications | The admin shall be able to **review courier registration** requests, verify required documents (e.g., vehicle information, licenses), and either approve or reject the application. |
| 3.1 | Order & Delivery Management | The admin shall be able to view **a list of all orders, including their status** (pending, preparing, out for delivery, completed, canceled). |
| 3.2 | Modify Order Status | The admin shall be able to manually **update an order status** in case of technical issues or restaurant/courier failure (e.g., change from "Preparing" to "Out for Delivery"). |
| 3.3 | Order Search & Filtering | The admin shall be able to **search and filter orders** based on order ID, customer name, restaurant, courier, status, and date range. |
| 3.4 | Assign or Reassign Couriers | The admin shall be able to manually **assign or reassign a courier** to an order in case of delays or unavailability. |
| 3.5 | View Customer Order History | The admin shall be able to view a **customer's order history**, including previous orders, payment details, and complaints. |
| 3.6 | Monitor Delivery Progress | The admin should be able to **track live delivery updates**, including courier locations, estimated delivery times, and route changes. |
| 3.7 | Monitor Courier Performance | The admin should be able to **track courier performance**, including delivery times, customer ratings, and on-time percentages. |
| 3.8 | Monitor Restaurant Review | The admin should be able to **track restaurant performance**, including order preparation times, customer ratings, and order fulfillment accuracy. |
| 4.1 | Support Management | The admin shall be able to **access a support dashboard** displaying open tickets, resolved issues, pending escalations, and customer/courier/restaurant complaints. |
| 4.2 | Manage Customer Support Requests | The admin shall be able to **receive, review, and respond** to customer complaints regarding orders, payments, delivery issues, or technical problems. |
| 4.3 | Manage Restaurant Support Requests | The admin shall be able to **handle support tickets** from restaurants regarding order delays, system issues, payout concerns, or menu updates. |
| 4.4 | Manage Courier Support Requests | The admin shall be able to address courier concerns related to delayed payments, route issues, delivery disputes, or app functionality problems. |
| 4.5 | Ticket Categorization & Prioritization | The system shall allow the **admin to categorize tickets** (e.g., payment issue, order delay, technical issue) and prioritize them based on urgency. |
| 4.6 | Track Ticket Status | The admin shall be able to **monitor the status of support tickets** (e.g., Open, In Progress, Resolved, Escalated) and ensure timely resolution. |
| 5.1 | Review Management | The admin shall have access to **all reviews submitted by customers**, including ratings, comments, and feedback for restaurants, food items, and couriers. |
| 5.2 | Filter and Sort Reviews | The system shall allow the admin to **filter** reviews by date, rating, restaurant, food item, or customer. The admin can **sort** reviews from high to low or vice versa. |
| 5.3 | Review Summary Dashboard | The admin shall be able to view a **summary dashboard** that provides an overview of reviews, including average ratings, top-rated items, and restaurants, and a count of positive, neutral, and negative reviews. |
| 5.4 | Flag Inappropriate Reviews | The system shall **automatically flag reviews containing** inappropriate language, spam, or content that violates platform policies. Admins can also manually flag reviews for review. |

# System Qualities[[6]](#bookmark=id.o4paeom21mvs)

## Usability

**Ease of Use:** The application must feature an intuitive interface, allowing users to navigate effortlessly. The average time to complete a primary task (e.g., placing an order) should not exceed 30 seconds. The application should maintain a task completion rate of 95% without user confusion or assistance. The application should cache relevant information in a page (e.g currently selected filters, entered yet not saved fields, current scroll amount in the page) such that if the page closes and reopens these information stay.

**Ease of Learning:** New users should be able to understand and complete basic tasks (such as placing an order) without external guidance. Onboarding should take no longer than 2 minutes, and the application should achieve a user satisfaction rate of at least 90% in terms of ease of learning in usability surveys.

**Usability Standards:** The application must follow recognized usability standards, to ensure efficiency, accessibility, and user satisfaction. It should maintain a usability score of 85% or higher in user feedback surveys.

**Localization:** The application must provide localization for at least 3 different languages and regions, including currency and date formatting. All localized versions should have a translation accuracy rate of 99% or higher, and the system should adapt automatically to the user's region without requiring manual adjustments.

## Reliability

**Availability:** The application must be available 24/7 with 99.9% uptime, ensuring continuous access without interruptions. During peak dining hours (e.g., breakfast, lunch, and dinner), the system should handle up to 10,000 concurrent users without performance degradation. Any downtime should not exceed 1 hour per month.

**Failure Frequency and Severity:** The application should experience no more than 0.1% failure rate in production. When errors occur, the system must provide clear, actionable error messages within 5 seconds, ensuring users are informed of any issues like payment failures. The user should not experience unexpected crashes or freezes more than once every 1,000 interactions.

**Recoverability:** The system should be capable of recovering from both client-side and server-side failures within 5 seconds. In case of a failure, the application must restore the user’s last action (such as cart review or payment) without data loss. Recovery time should be less than 5 seconds, and no critical data should be lost during failure scenarios.

## Performance

**Response Time:** The application must respond within 2-3 seconds for all interactions, including search functionality, order processing, and real-time updates. Any delay beyond 3 seconds should occur in less than 5% of interactions. The average response time must remain below 2.5 seconds for 95% of all user requests.

**Throughput:** The system should handle at least 1,000 concurrent users without noticeable performance degradation during peak traffic. It should maintain consistent response times, even when processing up to 500 requests per second. The throughput should remain stable with a failure rate below 0.5% during peak hours.

**Capacity:** The application should scale to support 10,000+ concurrent users with a minimal increase in response time. The system must dynamically allocate resources (such as server load balancing) to handle growth in user numbers and data storage without compromising performance.

**Startup and Shutdown Times:** The application should start up within 3 seconds and shut down within 5 seconds, without interrupting ongoing transactions or causing data loss. All processes should be completed during the shutdown process, ensuring that no user data is lost or unsaved actions are discarded.

## Supportability

**Adaptability and Upgrading:**The system should support seamless updates to menu items and UI components without causing downtime. Updates should be completed within 1 hours and not affect the core operations. The system should handle updates with no more than 2 minutes of service interruption.

**Compatibility:**The app must be compatible with Android, iOS, Windows, and macOS, as well as function properly on various screen sizes. It should maintain a response time of less than 3 seconds across all devices and browsers, with no critical compatibility issues.

**Configurability:**Users must be able to update personal details (e.g., address) and modify food preferences. Configurations should be applied within 3 seconds, and changes should be reflected instantly in the app without requiring a restart.

**Scalability:**The system should be able to handle at least 1,000 concurrent users during peak hours. The order processing time should remain under 5 seconds per order, even when processing high traffic volumes. The system should scale automatically based on demand without performance degradation.

**Installation and Maintenance:**Installation time should not exceed 2 minutes for any platform. Maintenance should have less than 1% downtime per month, with automatic background updates occurring without disrupting user activity.

**Level of Support:**The app must provide at least 95% of queries through AI-powered chat within 1 minute. For live support, response time should be under 5 minutes, with a resolution time of under 30 minutes for urgent issues. Non-urgent issues should be resolved within 24 hours.

# System Interfaces[[2]](#bookmark=id.5mn6jjbrbu46)

This section defines the interface requirements for the HUrricane system, categorized under the "+" in the FURPS+ classification of supporting requirements. It specifies the user interfaces, external system interfaces, software interfaces, hardware interfaces, and communication interfaces that the system must support. These requirements are detailed with sufficient specificity, including protocols, ports, logical addresses, and design constraints, to ensure the software can be developed and verified against the interface requirements.

## User Interfaces

The user interfaces for HUrricane are designed to provide an intuitive, efficient, minimalistic and visually appealing experience for all user types—administrators, restaurants, customers, and couriers. The following subsections outline the requirements for the look and feel, layout, navigation, consistency, and personalization of the user interfaces.

### Look & Feel

The HUrricane user interface must embody a vibrant and energetic spirit, reflecting the dynamic nature of a food delivery platform. The interface shall use a color scheme dominated by shades of red and orange to convey energy, urgency, and appetite appeal, with a light coffee-colored background to provide a warm and inviting contrast. Specifically:

* **Primary Colors**: Red (e.g., #D32F2F) for buttons, headers, and interactive elements to draw attention and indicate actionable items.
* **Secondary Colors**: Orange (e.g., #FF9800) for highlights, notifications, and secondary actions to create a sense of warmth and urgency.
* **Background:** Light coffee (e.g., #F5F5DC) to ensure readability and a cohesive aesthetic that complements the food theme.
* **Typography**: The interface must use a modern, sans-serif font (e.g., Roboto or Open Sans) with a minimum font size of 14px for readability. Text in red or orange must have sufficient contrast against the light coffee background to meet WCAG 2.1 accessibility standards (contrast ratio of at least 4.5:1).
* **Interaction:** The interface must provide immediate visual feedback for user actions (e.g., button hover effects changing from red to a darker red shade, #B71C1C) and use subtle animations (e.g., fade-in effects for order status updates) to enhance the user experience without overwhelming the user.

### Layout and Navigation Requirements

The layout of HUrricane must be structured to facilitate quick access to key functionalities for each user type. The major screen areas and their grouping are as follows:

**Header Section:** Located at the top of all pages, the header must display the HUrricane logo (in red and orange) on the left and a "View All" link on the right for navigating to comprehensive lists (e.g., all orders). The header background must be white with a red border to distinguish it from the main content.

Navigation Sidebar: A fixed vertical sidebar on the left side of the screen, occupying 15% of the screen width, must contain navigation options such as "Account," "Restaurant Menu," "Orders," and "Couriers." Each option must be represented by a red button with white text, turning orange on hover. The sidebar background must match the light coffee color of the main background for visual continuity.

**Main Content Area:** Occupying the remaining 85% of the screen width, this area must display the primary content (e.g., current orders, past orders). It must be divided into sections with red headers (e.g., "Current Orders," "Past Orders") and use a grid layout for displaying order cards. Each order card must include an image of the food item, order details, and a "View Details" button in red.

Status Indicators: Order status (e.g., "Preparing," "On the Road," "Delivered") must be displayed prominently on each order card using green text for positive statuses (e.g., "Delivered") and orange text for in-progress statuses (e.g., "On the Road").

**Navigation Flow:** Users must be able to navigate between sections (e.g., from "Orders" to "Restaurant Menu") with a single click. The system must maintain a breadcrumb trail (e.g., "Home > Orders > Order Details") in the main content area, displayed in orange text, to help users track their navigation path.

### Consistency

Consistency in the user interface is critical to ensure predictability and ease of use across all modules of HUrricane. The following requirements must be adhered to:

**Navigation Controls:** All navigation buttons in the sidebar must have a uniform size (e.g., 200px wide, 50px tall) and padding (10px). The active section (e.g., "Orders") must be highlighted with an orange background, while inactive sections remain red.

**Screen Areas:** Order cards in the main content area must have a fixed width of 300px and height of 400px, with a 10px margin between cards. The layout must remain consistent across all user types, with only the content (e.g., order details vs. courier assignments) changing.

**Data Presentation:** Order details (e.g., "Order ID," "Customer Name," "Total Amount") must follow a consistent format: labels in bold red text, values in black text, and aligned in a two-column layout within each order card.

**Terminology:** The system must use consistent terminology across all interfaces (e.g., "Order Status" instead of alternating with "Delivery Status"). A glossary of terms must be maintained in the system documentation to ensure uniformity.

**External Consistency:** The interface must align with common web application standards, such as placing the navigation sidebar on the left and using a "View All" link in the top-right corner, to meet user expectations based on industry norms.

### User Personalization & Customization Requirements

HUrricane must provide personalization and customization options to enhance the user experience:

**Automatic Display Based on User Attributes**: The system must automatically display relevant content based on the user’s role. For example, restaurant users must see a "Restaurant Menu" section by default, while couriers see a "Delivery Requests" section. The default view must be determined by the user’s role upon login.

**Customizable Content:** Registered customers must be able to save favorite restaurants, which are then displayed in a "Favorites" section on their homepage. This section must appear above the "Browse Restaurants" section and be collapsible with a toggle button.

**Personalization Options:** Users must be able to customize their dashboard by selecting preferred themes (e.g., a darker shade of the light coffee background, #E6D7B9) and adjusting font sizes (e.g., small: 12px, medium: 14px, large: 16px). These settings must be saved in the user’s profile and applied across all sessions.

**Order Preferences**: Customers must be able to set default payment methods and delivery addresses, which are automatically populated during checkout. These preferences must be editable via the "Account" section and applied within 3 seconds of saving.

## Interfaces to External Systems or Devices

HUrricane must interface with several external systems to support its functionality. The following requirements specify the nature of these interfaces:

**Payment Gateway:** The system must integrate with a third-party payment gateway (e.g., Stripe or PayPal) to process customer payments. The interface must use HTTPS protocol over port 443 and exchange data in JSON format. The payment gateway must return a transaction ID and status (e.g., "Success," "Failed") within 5 seconds of a payment request.

**Email Service:** The system must interface with an email service (e.g., SendGrid) for password reset and order confirmation emails. The interface must use SMTP protocol over port 587 with TLS encryption. Email payloads must be in JSON format, including fields such as "to," "subject," and "body," and the service must confirm delivery within 10 seconds.

**Geolocation Service:** The system must integrate with a geolocation API (e.g., Google Maps API) to calculate delivery distances and provide real-time tracking for couriers. The API must be accessed via HTTPS over port 443, using RESTful requests with JSON responses. The API must return coordinates and estimated travel times within 2 seconds.

**Constraints:** All external interfaces must support retry mechanisms (up to 3 retries with a 2-second delay between attempts) in case of failures. The system must log all interactions with external systems for auditing purposes, as specified in the system-wide requirements.

### Software Interfaces

HUrricane must interact with several software components, including third-party libraries and frameworks:

**Spring Data JPA**: Used for database access, this component must interface with the PostgreSQL database via JDBC over port 5432. The interface must support CRUD operations with a maximum latency of 100ms per query.

**JWT (JSON Web Token)**: Used for authentication, JWT must be integrated into the Spring Security framework. The system must generate and validate tokens via HTTP headers, with a token expiration time of 24 hours.

**React Router:** Used for client-side routing, this library must handle navigation between pages (e.g., from "Orders" to "Restaurant Menu") without full page reloads. The routing must complete within 200ms.

**Axios:** Used for HTTP requests from the React frontend to the backend REST API, Axios must communicate over HTTPS on port 8080. Requests must include authentication headers (JWT tokens) and return responses in JSON format within 3 seconds.

**Tailwind CSS**: Used for styling, Tailwind CSS must be integrated into the React frontend to apply the red, orange, and light coffee color scheme consistently across all components.

### Hardware Interfaces

HUrricane is primarily a software system and does not require direct hardware interfaces beyond standard web application requirements. However, the following constraints apply:

**Device Compatibility:** The system must support devices with screen resolutions ranging from 320x480px (mobile) to 1920x1080px (desktop). The interface must adapt responsively using Tailwind CSS media queries.

**Geolocation Hardware:** For courier tracking, the system must interface with the device’s GPS hardware via the browser’s Geolocation API. The API must provide location updates with an accuracy of at least 10 meters and a refresh rate of 5 seconds.

**Expected Behavior:** The system must handle GPS failures gracefully by displaying a fallback message (e.g., "Unable to retrieve location, please try again") and logging the error for debugging.

### Communications Interfaces

HUrricane must support the following communication interfaces to ensure seamless operation:

**Local Area Network (LAN):** The backend server must communicate with the PostgreSQL database over a LAN using JDBC on port 5432. The connection must be secured with SSL/TLS to prevent data interception.

**Internet Communication:** All client-server communication must occur over HTTPS on port 443, using RESTful APIs. The system must support a minimum bandwidth of 1Mbps to ensure smooth operation during peak usage.

**WebSocket for Real-Time Updates**: The system must use WebSocket protocol on port 8080 to provide real-time order status updates (e.g., "Preparing" to "On the Road").

# Business Rules[[6]](#bookmark=id.4i6u2zf64wpi)

## 5.1 Food Order Rules

### **5.1.1** **Minimum Order Rule**

**Description:** If a customer’s order is under 300TL, add a 20TL delivery fee.  
**Condition:** Order total < TL300  
**Action:** Add 20TL delivery fee

#### **5.1.2 Free Delivery Rule**

**Description:** Orders above 500TL qualify for free delivery.  
**Condition:** Order total ≥ 500TL  
**Action:** Waive the delivery fee.

#### **5.1.3 Restaurant Operating Hours Rule**

**Description:** Orders can only be placed within the restaurant’s operating hours.  
**Condition:** Current time outside restaurant operating hours  
**Action:** Disable ordering and display a message: "Restaurant is currently closed."

#### **5.1.4 Out-of-Stock Rule**

**Description:** Customers cannot add out-of-stock food items to their cart.  
**Condition:** Food item status = Out of stock  
**Action:** Display a message: "This item is currently unavailable."

#### **5.1.5 Minimum Delivery Time Rule**

**Description:** Orders must have a minimum preparation and delivery time based on restaurant estimates.  
**Condition:** Delivery time < Estimated preparation time  
**Action:** Adjust estimated delivery time accordingly.

#### **5.1.6 Order Confirmation Time Limit**

**Description:** If a restaurant does not confirm an order within 10 minutes, the order is automatically canceled.  
**Condition:** Order status = "Pending Confirmation" for 10+ minutes  
**Action:** Cancel the order and notify the customer.

#### **5.1.7 Loyalty Program Rule**

**Description:** Customers earn 1 point for every 10TL spent, redeemable for future orders.  
**Condition:** Order total ≥ 10TL  
**Action:** Add 1 loyalty point for every 10TL spent.

### **5.2 Food Delivery Rules**

#### **5.2.1 Delivery Area Rule**

**Description:** Delivery is only available within a specified radius from the restaurant.  
**Condition:** Customer address within the delivery radius (e.g., 10 km).  
**Action:** Enable delivery service for eligible addresses; disable it for addresses outside the radius.

#### **5.2.2 Delivery Time Estimate Rule**

**Description:** The system provides an estimated delivery time based on restaurant preparation time and courier availability.  
**Condition:** Order placed and status is "Preparing."  
**Action:** Display estimated delivery time to the customer.

#### **5.2.3 Late Delivery Penalty Rule**

**Description:** If delivery exceeds the estimated time by more than 10 minutes, offer a cancellation on current order.  
**Condition:** Delivery exceeds estimated time by > 10 minutes.  
**Action:** Offer a cancellation on the current order.

#### **5.2.4 Delivery Tracking Rule**

**Description:** Customers can track the real-time status of their order delivery.  
**Condition:** Order status is "Out for Delivery."  
**Action:** Allow customer to view delivery progress via tracking system.

**5.2.5 Courier Free Rule**

**Description:** Courier must be available after 15 mins of delivery. **Condition:** Minutes passed after delivery time < 15 **Action:** Keep the courier busy.

### 

### **5.3 Customer Loyalty and Discount Rules**

#### **5.3.1 Customer Loyalty Discount Rule**

**Description:** Loyal customers receive a 10% discount on orders after placing 10 orders.  
**Condition:** Customer placed ≥ 10 orders.  
**Action:** Automatically apply a 10% discount on the next order.

#### **5.3.2 Loyalty Point Discount Rule**

**Description:** Customers can buy discount tickets by collecting loyalty points.  
**Condition:** Have loyalty points > 150  
**Action:** Apply a discount based on the ticket.

#### **5.3.3 First Order Discount Rule**

**Description:** New customers receive a 15% discount on their first order.  
**Condition:** First order by a customer.  
**Action:** Automatically apply a 15% discount on the first order.

#### **5.3.4 Overlap Discount Rule**

**Description:** Customers can’t apply 2 different discounts at the same time. Example scenario: Discount from 11th order and discount from loyalty point, customer can’t apply both of them at the same time.  
**Condition:** 2 different discounts tried to apply.  
**Action:** Display an error message.

# System Constraints[[7]](#bookmark=id.jkhqfsy5tiyc)

This section defines the mandated design, implementation, and deployment constraints for HUrricane, per the FURPS+ classification.

* **Languages**: Backend must use Java Spring Boot; frontend must use React with Tailwind CSS for styling (red, orange, light coffee scheme).
* **Tools**: Use Maven for Spring Boot, Create React App/Vite for React, VS Code for frontend, IntelliJ IDEA for backend, Git with GitHub Flow (branch, pull requests, merge to master). PostgreSQL with DBeaver/pgAdmin for database.
* **Libraries**: Spring Data JPA, JWT for authentication, React Router, Axios for API calls, Stripe, Google Maps API for payments and geolocation.
* **Architecture**: Must follow MVC pattern and Open-Close Principle.
* **Platform**: Must run on laptops (Windows/macOS/Linux, 8GB RAM, 2.0 GHz processor, 20GB disk). Support Chrome, Firefox, Edge; deployable on AWS/Azure for future scalability. Backend on localhost:8080, frontend on localhost:3000, PostgreSQL on localhost:5432.
* **Resource Limits**: Backend ≤ 500MB memory, frontend loads in ≤ 3s (1Mbps), database ≤ 5GB. API rate limit: 100 requests/min/user.
* **Security**: Use JWT (24h expiry), Bcrypt for passwords, HTTPS with TLS 1.3, sanitize inputs.
* **Process**: Follow GitHub Flow; 9–12 critical use cases.
* **Deployment**: Must run locally on laptops; document setup in Developer Guide.

# System Compliance[[3]](#bookmark=id.bbu1jfldiery)

This section outlines the compliance requirements for the HUrricane system, ensuring that the software adheres to licensing, legal, and regulatory standards, as well as industry best practices. It also specifies the requirements for system documentation to support user understanding and future maintenance.

## Licensing Requirements

HUrricane must comply with licensing requirements for all software components, frameworks, and third-party libraries used in its development, ensuring proper usage and distribution rights. The following licensing requirements must be adhered to:

**Open-Source Libraries:** The system uses several open-source libraries, including Spring Data JPA, React, and Tailwind CSS. All open-source components must comply with their respective licenses:

* Spring Data JPA and Spring Boot are licensed under the Apache License 2.0, which permits modification and distribution but requires the inclusion of the license and copyright notice in the system documentation.
* React is licensed under the MIT License, allowing free use, modification, and distribution, provided the MIT License and copyright notice are included in the software distribution.
* Tailwind CSS is also licensed under the MIT License, with the same requirements as React.
* Third-Party Services: The system integrates with third-party services such as payment gateways (e.g., Stripe) and geolocation APIs (e.g., Google Maps API). These services must be used in accordance with their terms of service:
* Stripe’s API usage requires a commercial license, and the system must comply with Stripe’s usage restrictions, such as prohibiting the storage of sensitive cardholder data (e.g., full credit card numbers) on HUrricane servers.
* Google Maps API usage must adhere to Google’s terms of service, including API request limits (e.g., 100,000 requests per month under the free tier) and the display of Google’s branding in the UI where maps are used.

**HUrricane Software Licensing:** The HUrricane system itself is developed as a course project for BBM384 and is not intended for commercial distribution. However, the source code must be licensed under the MIT License to allow future use by the teaching staff and other students, with proper attribution to the development team (PentaCode). The license file must be included in the project’s GitHub repository root directory.

**Usage Restrictions:** The system must enforce usage restrictions to prevent unauthorized access. Only registered users (customers, restaurants, couriers, and administrators) with valid credentials can access the system. The system must implement JWT-based authentication to enforce this restriction, ensuring that API endpoints are inaccessible without a valid token.

## Legal, Copyright, and Other Notices

HUrricane must include appropriate legal disclaimers, copyright notices, and trademark acknowledgments to protect intellectual property and inform users of their rights and responsibilities. The following requirements apply:

**Copyright Notice:** The system must display a copyright notice on all user interfaces and in the system documentation, stating:

"© 2025 PentaCode Team. All rights reserved."

This notice must be placed in the footer of all web pages and on the cover page of all documentation.

**Trademark and Logo Compliance:** The HUrricane logo and name must be treated as a trademark of the PentaCode team. The logo must be displayed consistently across all interfaces (as seen in the provided GUI image) and must not be altered by third parties without permission. A trademark notice ("HUrricane™") must be included in the "About" section of the application and in the system documentation.

**Third-Party Notices:** The system must acknowledge third-party components and services:

A "Third-Party Notices" section must be included in the system documentation, listing all open-source libraries (e.g., Spring Boot, React, Tailwind CSS) with their respective licenses and copyright notices.

Google Maps API usage must include the Google logo and attribution (e.g., "Powered by Google") on all pages where maps are displayed, as per Google’s branding guidelines.

**Legal Disclaimer:** The system must include a disclaimer in the "Terms of Use" section, accessible via the footer of all pages, stating:

"HUrricane is a student project developed for educational purposes under the BBM384 Software Engineering Laboratory course. The PentaCode team is not liable for any damages, losses, or issues arising from the use of this system, including but not limited to order delays, payment failures, or data breaches."

**Warranty Disclaimer:** The system must explicitly state that it is provided "as-is" without any warranties:

"HUrricane is provided without any express or implied warranties, including warranties of merchantability, fitness for a particular purpose, or non-infringement."

This disclaimer must be included in the "Terms of Use" section and in the system documentation.

## Applicable Standards

HUrricane must comply with relevant standards to ensure quality, interoperability, security, and usability. The following standards, referenced from the experiment sheet and industry best practices, apply:

**IEEE Standards (as per the experiment sheet):**

IEEE Std 830-1998 (Recommended Practice for Software Requirements Specifications): The SRS document must conform to this standard, particularly in its structure (e.g., purpose, scope, functional and non-functional requirements) and traceability requirements (e.g., unique identifiers for requirements).

IEEE Std 1016-1998 (Recommended Practice for Software Design Descriptions): The Architecture Notebook, as specified in the experiment sheet, must follow this standard for documenting the system’s design, including architectural views, component interactions, and design rationale.

IEEE Std 829-1998 (Standard for Software Test Documentation): The Software Test Report (STR) and Test Case Definitions must adhere to this standard, ensuring that test cases, test scripts, and test results are documented systematically with clear pass/fail criteria.

**Security Standards:**

OWASP Top Ten: The system must comply with the OWASP Top Ten security guidelines to mitigate common vulnerabilities, such as injection attacks (e.g., SQL injection) and broken authentication. For example, all user inputs must be sanitized, and JWT tokens must be securely stored and validated.

GDPR Compliance: Since the system handles personal data (e.g., customer names, addresses, payment details), it must comply with the General Data Protection Regulation (GDPR). This includes obtaining explicit user consent for data collection, providing a "Right to be Forgotten" option (e.g., account deletion), and ensuring data encryption (e.g., HTTPS, Bcrypt for passwords).

**Usability Standards:**

ISO 9241-11 (Ergonomics of Human-System Interaction): The system must meet usability criteria defined in this standard, such as effectiveness (e.g., 95% task completion rate), efficiency (e.g., placing an order in under 30 seconds), and satisfaction (e.g., 90% user satisfaction rate in surveys).

WCAG 2.1 (Web Content Accessibility Guidelines): The user interface must achieve at least Level AA compliance, ensuring accessibility for users with disabilities. This includes providing sufficient color contrast (e.g., 4.5:1 ratio for red text on a light coffee background), keyboard navigation support, and alt text for all images (e.g., food item images in order cards).

**Interoperability Standards:**

REST API Standards: The system’s RESTful APIs must follow the OpenAPI Specification (OAS) 3.0 for defining endpoints, ensuring interoperability with external systems (e.g., payment gateways). API responses must use JSON format with standard HTTP status codes (e.g., 200 for success, 401 for unauthorized).

HTTPS Compliance: All network communications must use HTTPS with TLS 1.3 to ensure secure data transmission, as mandated by modern web standards.

**Internationalization:**

The system must support localization for at least three languages (e.g., English, Turkish, Spanish), as specified in the experiment sheet’s functional requirements. Date formats (e.g., MM/DD/YYYY for the US, DD/MM/YYYY for Europe) and currency formats (e.g., USD, TRY) must adapt to the user’s region, following the Unicode Common Locale Data Repository (CLDR) standards.

# System Documentation[[4]](#bookmark=id.tm3plpt6meql)

This section outlines the requirements for HUrricane’s documentation, including online user documentation, help systems, help about notices, and developer documentation. It ensures the system is usable and maintainable while identifying responsibilities for documentation creation.

**8.1 Online User Documentation**

HUrricane must provide online user documentation to assist all user types (administrators, restaurants, customers, couriers):

* A "Help" section, accessible via the footer, must include role-specific guides (e.g., placing orders for customers, managing menus for restaurants) with step-by-step instructions and screenshots.
* A searchable FAQ section must address at least 15 common questions (e.g., "How do I reset my password?"), returning results within 2 seconds.
* Documentation must support three languages (English, Turkish, Spanish) with 99% translation accuracy and comply with WCAG 2.1 Level AA for accessibility (e.g., alt text for images, 14px font size).

**8.2 Help Systems**

The system must offer contextual help:

* A "Help" icon on each page must open a modal with relevant guidance (e.g., how to track orders on the "Orders" page), loading within 1 second.
* Tooltips on interactive elements (e.g., "View Details" button) must appear within 0.5 seconds, using a red background with white text.
* Error messages (e.g., "Incorrect login") must include a "Help" link directing to the relevant FAQ within 1 second.

**8.3 Help About Notices**

An "About HUrricane" page, accessible via the footer, must include:

* Version number (e.g., "HUrricane Version 1.0, March 2025").
* Description: "HUrricane is an online food delivery platform developed by the PentaCode team for the BBM384 Software Engineering Laboratory course."
* Support email and copyright notice: "© 2025 PentaCode Team. HUrricane™ is a trademark of the PentaCode team."
* The page must use the system’s color scheme (red, orange, light coffee background).

**8.4 Developer Documentation**

A "Developer Guide" in the GitHub repository must include:

* System architecture overview (referencing the Architecture Notebook).
* API documentation (e.g., POST /orders) using OpenAPI Specification 3.0.
* Setup instructions (e.g., running Spring Boot backend, React frontend, PostgreSQL).
* Extensibility guide for adding new features per the Open-Close Principle.
* The guide must be in Markdown format in the /docs directory.

**8.5 Responsibility and Format**

The PentaCode team is responsible for creating and maintaining all documentation, with teaching assistants reviewing it during system acceptance (per Section 8.6 of the experiment sheet).

User documentation must be available online and as a downloadable PDF; developer documentation must be in Markdown on GitHub. All content must meet WCAG 2.1 Level AA accessibility standards.

# Traceability Table

|  | Use Case Definition | Test Case Definition | System Req Spesification | GUI Design | Summation |
| --- | --- | --- | --- | --- | --- |
| Yusuf Küçüköner | 10h | 0h | 10h | 0h | 20 |
| Salih Eren Yüzbaşıoğlu | 10h | 4h | 6h | 0h | 20 |
| Şükriye Öztürk | 0h | 0h | 6h | 14h | 20 |
| Mustafa Furkan Ateş | 0h | 0h | 0h | 20h | 20 |
| Bedirhan Gençaslan | 11h | 10h | 0h | 0h | 21 |

# Prompts

* [Grok 3: “can you write the introduction of my SRS document for the project in the pdf. here is an example introduction of another team: …”](https://grok.com/share/bGVnYWN5_f37fa01a-f130-42e2-beeb-647deb2e6044)

* [Grok 3: “Write this part for mu such that….”](https://grok.com/share/bGVnYWN5_e592d602-6e70-43ec-a96a-f33d7f3f7c27)

* Grok 3: “write System Compliance based on the experiment sheet and prior knowledge professionally…” (Same link as above 2)

* Grok3: “Write system documentation based on the experiment sheet and…” (Same link)

* ChatGPT: <https://chatgpt.com/share/67e6f5eb-0f14-8000-9cc2-f819a0e3831d>
* ChatGPT: <https://chatgpt.com/share/67e6f788-c41c-8000-a348-6cd8e11fea70>

* Grok 3. “Write the System Documentation part professionally, short and dense for reference we these languages: ….”