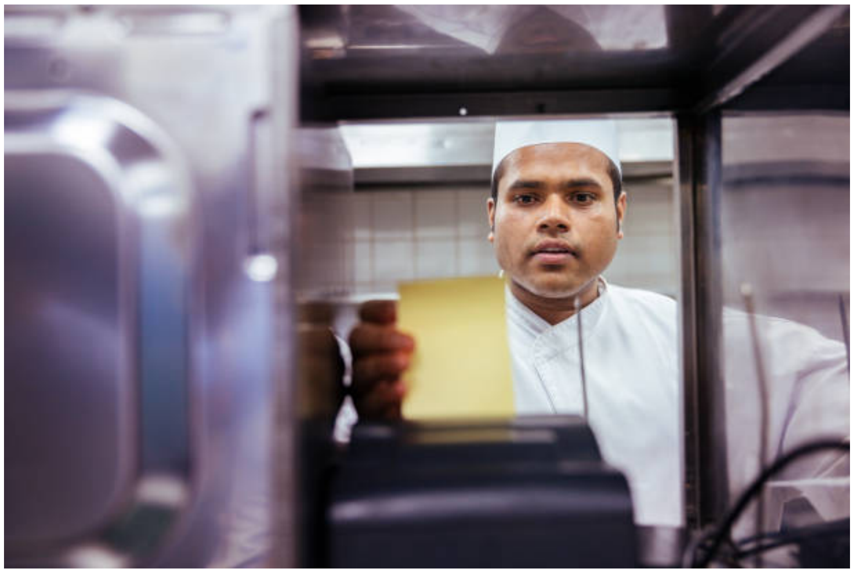
# OOPS (Online Order Printing Software)

Team 16 - brj46, hfd2, jjo1, ros3

Haust 2025



# 1 Project Vision

## 1.1 Vision Statement

To empower businesses to process online orders instantly and accurately, making every transaction seamless and stress-free. The system is designed to remove barriers for both restaurant owners and customers, providing a smooth and reliable experience from order creation to fulfillment.

## 1.2 Long-Term Goal

The long-term vision of this project is to enable restaurant owners to receive and manage orders without the need for direct phone calls or manual input. Customers will be able to place orders online via a mobile app or website, reducing staff workload and minimizing errors.

## 1.3 Value for Stakeholders

**Customers:** Can browse menus, place orders, see totals, and save favorites. In the beta version, payments will be made at pickup to simplify adoption.

**Restaurant Staff:** Gain a login to manage pending and closed orders, adjust order queue times, and mark items as temporarily sold out.

**Administrators:** Have full control over menus, pricing, and opening hours. They also gain access to sales reports and item popularity analytics, empowering data-driven decisions.

## 1.4 Competitive Advantage

Unlike traditional phone-based ordering or existing delivery platforms that often take a commission, this system gives restaurants direct control over their online orders. With built-in analytics and flexible administration features, it not only streamlines the ordering process but also provides valuable insights into customer behavior and sales trends.

## 2 Use Case Document

### UC1: Place Order (Pay at Pickup)

**Scope:** Restaurant Online Ordering System
**Level:** User goal
**Primary actor:** Customer

**Stakeholders & interests:**

- Customer: Wants to browse menu, create order, see total price, and receive clear pickup time without errors.
- Restaurant Staff: Wants accurate, complete orders, and realistic queue times.
- Restaurant Owner/Admin: Wants increased order throughput and fewer phone calls; accurate sales and item popularity data.

**Preconditions:**

- Menu and opening hours are available.
- System is accepting orders.

**Success guarantee:**

- Order recorded with items, prices, and total.
- Order status set to "Pending."
- Pickup time estimated.
- Customer receives confirmation.

**Main success scenario:**

1. System displays current menu.
2. Customer adds items to basket.
3. System shows running total and pickup time.
4. Customer enters phone number.
5. Customer reviews and confirms order.
6. System validates store and availability.
7. System creates order, assigns ID.
8. System presents confirmation and notifies staff.

**Extensions / alternate scenarios:**

(2a) Item sold out:

1. System notifies customer that the item is unavailable.
2. Customer either replaces the item or removes it from basket.

(4a) Invalid phone number:

1. System detects phone number does not match valid format.
2. System requests correction from the customer.
3. Customer re-enters phone number.

(5a) Store just closed:

   1. System checks current store opening hours.
   2. System rejects the order and notifies customer of closure.

(6a) Inventory threshold hit:

   1. System detects item availability is below threshold.
   2. System prevents checkout of that item.
   3. Customer edits basket to adjust or remove the item.

(7a) Notification delivery fails:

   1. System attempts to send confirmation to customer.
   2. System fails to deliver via chosen channel.
   3. System shows confirmation on screen and logs error.

**Special requirements:**

- Pickup time shown $<$ 1 second after basket update.
- Confirmation within 2 seconds.
- Accessibility requirements met.

**Technology & data variations:**

- International phone formats supported.
- Future: online payments.
- Reorder favorites feature possible.

**Frequency:** High during open hours.
**Open issues:** Phone number retention, taxes, notification channels.

## UC2: Manage Menu (Add/Update/Remove Items)

**Scope:** Restaurant Online Ordering System
**Level:** User goal
**Primary actor:** Admin

**Stakeholders & interests:**

- Admin: Needs control over menu, prices, availability.
- Customers: Need accurate menus.
- Staff: Wants fewer manual corrections.

**Preconditions:** Admin authenticated.
**Success guarantee:** Menu changes are persisted and reflected. Logged for audit.

**Main success scenario:**

1. Admin signs in.
2. Opens menu management.
3. Creates or edits item.
4. Updates properties (name, price, etc.).
5. System validates inputs.

6. Admin saves.
7. System persists changes.
8. System confirms success.
9. Customers see update.

**Extensions / alternate scenarios:**

(3a) Remove item with open orders:
   1. Admin attempts to delete an item.
   2. System checks if the item is part of any open orders.
   3. If yes, system prevents deletion and notifies admin.
   4. Admin may mark the item as unavailable instead.

(5a) Validation error:
   1. System detects invalid input (e.g., empty name, negative price).
   2. System highlights the error field and prompts correction.
   3. Admin corrects and retries save.

**Special requirements:**

- Audit trail of changes.
- Update latency $<$ 5 seconds.
- Role-based access control.

**Frequency:** Moderate (weekly/daily).
**Open issues:** Price rounding, image hosting.

## UC3: Set Order Queue Time (Adjust Estimated Prep/Wait)

**Scope:** Restaurant Online Ordering System
**Level:** User goal
**Primary actor:** Staff/Admin

**Stakeholders & interests:**

- Staff: Needs realistic pickup times.
- Customers: Want accurate estimates.
- Owner/Admin: Wants fewer cancellations.

**Preconditions:**

- Staff authenticated.
- Store open.

**Success guarantee:**

- Queue time updated.
- Applied to new orders.
- Logged with timestamp and actor.

**Main success scenario:**

1. Staff signs in.
2. Opens dashboard.
3. Reviews pending orders.
4. Adjusts queue time.
5. System validates bounds.
6. System applies change.
7. Confirmation shown.

**Extensions / alternate scenarios:**

(5a) Out-of-range input:

1. Staff enters a queue time outside valid limits.
2. System rejects the input and suggests a valid range.
3. Staff re-enters a valid value.

(6b) Scheduled rules override:

1. A scheduled rule (e.g., lunch rush adjustment) is active.
2. System prompts staff for new queue time during a scheduled rule.
3. Staff may adjust the schedule if necessary.

**Special requirements:**

- Change applied in $< 2$ seconds.
- All changes traceable.

**Frequency:** Variable, often during peaks.
**Open issues:** Policy on updating existing orders, scheduled vs. ad-hoc overrides.

**UC4: View Menu** A customer selects a restaurant and the system shows the current menu with categories, item names, descriptions, prices, and real-time availability. The customer can expand items to see options/modifiers and add items to their basket. The system ensures only currently available items are shown and reflects temporary "sold out" statuses immediately.

**UC5: Sign In (Staff/Admin)** A staff member or admin provides credentials; the system authenticates and grants access to role-appropriate features (orders dashboard for staff; menu and reports for admin). Failed attempts are limited and logged; on success, the user is taken to the relevant workspace.

**UC6: View Sales Report** An admin opens the reporting section and selects a date range. The system aggregates sales totals, average order value, item-level popularity, and trends. The admin can export results and use insights to adjust prices, menu, or staffing.

**UC7: Adjust Opening Hours and Exceptions (Admin)** An admin updates regular opening hours and sets one-time exceptions such as holidays or early closures. The system validates the new schedule, applies the changes to ordering eligibility, and ensures customers cannot place orders outside the adjusted hours.

**UC8: Manage Roles and Access (Admin)** An admin assigns or revokes roles (Staff, Admin) for other users. The system updates access permissions immediately, ensuring that each user can only access features appropriate to their role. All role changes are logged for security and accountability.

**UC9: View Order Status (Customer)** A customer enters their order ID (and optionally phone number) to check the current status of their order. The system shows whether it is Pending, Preparing, Ready, or Closed, along with the estimated pickup time.

**UC10: Set Order Status (Staff)** A staff member updates the status of an order, such as moving it from Pending to Preparing, from Preparing to Ready, and finally to Closed once picked up. The system records each update, ensures proper sequence of states, and notifies the customer if applicable.

## 3 Project Estimation and Prioritization

We use a **grouped priority system** where a lower number means higher priority (P1 > P2 > P3). Time estimates are in **person hours**.

| Use Case | Time Estimation (hours) | Priority |
|---|---|---|
| UC1: Place Order (Pay at Pickup) | 10 | P0 |
| UC2: Manage Menu | 15 | P0 |
| UC3: Set Order Queue Time | 12 | P1 |
| UC4: View Menu | 5 | P0 |
| UC5: Sign In (Staff/Admin) | 6 | P1 |
| UC6: View Sales Report | 8 | P1 |
| UC7: Adjust Opening Hours & Exceptions (Admin) | 6 | P2 |
| UC8: Manage Roles & Access (Admin) | 6 | P2 |
| UC9: View Order Status (Customer) | 5 | P1 |
| UC10: Set Order Status (Staff) | 6 | P1 |

## 4 Project Plan and Schedule

The project spans **10 weeks** with 4 sprints (Sprint 1: 2 weeks, Sprint 2: 3 weeks, Sprint 3: 3 weeks, Sprint 4: 2 weeks). Each sprint has a designated Product Owner (P.O.).

| Week | Use Cases / Activities | Expected Hours | P.O. | Sprint | Consultation |
|---|---|---|---|---|---|
| 1 | None – setup & planning | ~5 | BRJ | 1 | **A1 Presentation** |
| 2 | UC1: Place Order; Android skeleton | ~20 | BRJ | 1 | Model Drafts |
| 3 | UC2: Manage Menu; UC3: Set Queue Time | ~25 | JJO | 2 | **A2 Presentation** |
| 4 | UC4: View Menu; UC5: Sign In; UC6: Reports | ~20 | JJO | 2 | Dev support |
| 5 | Testing UC1–UC6; integration | ~15 | JJO | 2 | Dev support |
| 6 | UC7: Opening Hours; UC8: Roles; UC9: Order Status (Cust); UC10: Order Status (Staff) | ~20 | HFD | 3 | **A3 Presentation** |
| 7 | Scaling improvements; performance checks | ~15 | HFD | 3 | Dev support |
| 8 | Final features; Print orders | ~15 | HFD | 3 | Dev support |
| 9 | Final testing; bug fixes | ~20 | ROS | 4 | A4 Prep |
| 10 | Delivery; deployment; presentation | ~15 | ROS | 4 | Final Presentation |

## 5 Project Skeleton

**GitHub Repository:** `https://github.com/Reynirjr/Hugbo1-team16`

**Team Members and Accounts:**

- BRJ – `Reynirjr`
- JJO – `jonnishmurda`
- HFD – `helgarfri`
- ROS – `Robertorri`

All team members have access and have made at least one commit (e.g., updating the README, adding skeleton files, or first endpoints). The skeleton was generated using **Java Spring** and uploaded to GitHub for collaborative development.