CURRENT IMPLEMENTATION · VERIFIED 8 NOVEMBER 2026

Product Requirements Document

Product purpose, implemented user journeys, and acceptance criteria for the current SMROS application.

01 · PRODUCT OVERVIEW

1. Product overview

The Student Meal Reservation and Ordering System (SMROS) is a server-rendered web application for a campus canteen. Students browse the current menu, select quantities, choose an available future pickup slot, submit an order, and track its status. Administrators maintain the menu, inventory, pickup capacity, student accounts, and order processing from a separate protected portal.

This PRD describes implemented behavior, not a future product roadmap. In the current model, a reservation is represented by an order linked to a pickup slot; there is no separate reservation or payment record.

02 · BACKGROUND

2. Project background and problem statement

Campus meal ordering involves coordination among a changing menu, finite meal stock, pickup times, and order progress. Without a shared application, students may have limited visibility into availability and staff must reconcile order details and pickup expectations manually.

SMROS centralizes menu, inventory, pickup-slot, and order records in one application. It gives students a self-service ordering and tracking flow and gives administrators operational views for processing orders. It does not process payments or provide real-time push updates.

03 · PRODUCT GOALS

3. Product goals and objectives

1. Allow students to discover available meals, prices, categories, and food images.
2. Capture meal quantities and a pickup choice with a server-calculated total.
3. Maintain stock and pickup capacity consistently during order placement and cancellation.
4. Give students clear order status, pickup information, order history, and account controls.
5. Give administrators tools to manage meals, categories, stock, students, pickup slots, orders, reports, and operational notifications.
6. Protect account and administrative actions with authentication, authorization, validation, and request-integrity controls.

04 · TARGET USERS

4. Target users and user roles

Role	Purpose	Implemented access
Student	Reserve campus meals and monitor pickup progress.	Student dashboard, meal exploration, orders and history, notifications, profile, and student password recovery.

Role	Purpose	Implemented access
Administrator	Operate the menu and fulfill campus orders.	Admin dashboard, meals and categories, pickup slots, orders, student accounts, reports, activity, email monitoring, SMTP diagnostics, settings, and database backup/merge restore.

Public registration assigns the admin role only when the users table is empty; subsequent public registrations are students. Initial deployment must secure this bootstrap path and administrator account.

05 · PRODUCT SCOPE

5. Product scope

Included

- Unified account login, role-based dashboards, student registration, profile editing, and account activation controls for administrators.
- Meal search, category filters, meal information and images, quantity selection, pickup scheduling, order totals, and order submission.
- Inventory and slot-capacity checks, student cancellation of pending orders, administrator order status processing, and history.
- Ready-for-pickup email notifications, student reset links by email, administrator reports and CSV downloads, activity history, and delivery monitoring.

Not implemented as product capabilities

- Online payments, stored payment methods, refunds, or payment transactions.
- A persistent or cross-device cart; selected quantities live in the current ordering form/session flow.
- Push/WebSocket updates, automatic background status refresh, or guaranteed real-time notification.
- Student-information-system integration, external identity providers, two-factor authentication, or an SLA.
- Forecasting/prediction of demand or a separate reservation table. The dashboard does include a low-stock warning for active meals at or below 10 units.

06 · PRODUCT REQUIREMENTS

6. Functional requirements

1. **Authentication:** accept email and password, validate a one-use arithmetic challenge, verify the PHP password hash, reject inactive accounts, regenerate the session identifier, and route by the stored role.
2. **Registration:** create an account from validated name, student number when supplied, email, and password. Only the first account in an empty users table receives the administrator role; later accounts receive the student role.
3. **Profile management:** allow a student to update name, student number, and email, and change a password after current-password verification.
4. **Catalog:** display available meals with category, description, price, stock/availability state, and an image or fallback placeholder; support text search and category filtering.

5. **Ordering:** validate quantities and future pickup slot, enforce slot capacity and current meal stock, calculate totals on the server, record line-item price snapshots, and deduct stock atomically.
6. **Tracking:** display active, completed, and cancelled student orders with pickup details and status; permit student cancellation only while pending.
7. **Administration:** create/edit/hide meals and categories; update prices, stock, availability, and images; manage student accounts and pickup capacity; search orders and perform valid status transitions.
8. **Operations:** report order and completed-sales summaries, filter transaction reports, download CSV output, inspect activity/email attempts, and run SMTP diagnostics or eligible notification retries.
9. **Recovery:** allow active student accounts to request single-use, time-limited password-reset links by email and submit a compliant replacement password.

07 · QUALITY REQUIREMENTS

7. Non-functional requirements

- **Security:** protected routes re-check the active account and role; state-changing forms use CSRF validation; SQL values use PDO prepared statements; rendered data is escaped.
- **Integrity:** database foreign keys preserve relationships; order placement, status changes, capacity checks, and stock restoration use transactions/row locks where implemented; duplicate submission tokens are one-use.
- **Usability:** student and administrator portals have distinct navigation; validation and email-delivery outcomes are communicated in the interface.
- **Accessibility and motion:** semantic controls, visible keyboard focus, responsive layouts, and reduced-motion handling are implemented in the existing interface.
- **Maintainability:** shared database, session, authorization, and view helpers are kept under includes/; migrations and CLI seeders are separate from request handlers.
- **Deployment:** database/SMTP settings are environment-provided; production requires suitable HTTPS, file permissions, database setup, Composer dependencies, and SMTP configuration for mail delivery.

The repository does not establish a performance target, supported user concurrency, uptime objective, or recovery-time objective. These require hosting-specific decisions and testing.

08 · STUDENT EXPERIENCE

8. Student portal features

- **Dashboard:** summary cards for orders, pickup times, and meals, plus links to the main student workflows.
- **Explore meals:** available menu cards, image display, text search, category filtering, quantity controls, calculated order amount, and pickup-slot selection.
- **Orders:** active orders show status progress and pickup information; completed/cancelled orders appear in order history. The data is fetched when the page loads or is refreshed.
- **Notifications:** students can see recorded ready-for-pickup email delivery state associated with their orders.
- **Account:** profile changes require current-password confirmation; the student can request a password reset if unable to sign in.

09 · ADMINISTRATOR EXPERIENCE

9. Administrator portal features

- **Dashboard:** student/order/pending/completed/cancelled counts, sales summaries based on completed orders, charts, top completed meals, hourly order patterns, and low-stock alerts.
- **Meals and categories:** meal/category create and edit workflows, availability and stock controls, validated image upload and replacement, and safeguards for records used by order history.
- **Pickup slots:** create daily slots at 11:00–12:00, 12:00–13:00, and 13:00–14:00; change capacity within validation limits; prevent capacity below active bookings and prevent deleting a used slot.
- **Orders:** search and filter order records, inspect order items, and apply server-validated status transitions. Cancelling an order restores its stock.
- **Students:** update student profile information, activate/deactivate accounts, or delete only accounts without order history.
- **Reports and operations:** date/status/student filters, CSV exports, printable reports, order-status activity, notification delivery monitoring/retry, SMTP diagnostics, and JSON database backup/merge restore.

Activity history records order status transitions; this is not a comprehensive audit of every read, login, or administrator action.

10 · MENU AND IMAGES

10. Meal management and food image display

Meals have a name, optional description, optional category, price, stock, availability flag, and optional filename in the existing meals.image_path column. Student menu cards and the administrator meal-management view display a corresponding local image; missing or invalid references resolve to the bundled placeholder.

Administrators can upload JPG/JPEG, PNG, or WEBP files up to 5 MiB. The server checks MIME type, matching extension, image metadata, dimensions (maximum 12,000 × 12,000 and 20 million pixels), and upload status; it generates a random 32-hex filename and stores files in uploads/meals/. Editing without a replacement preserves the existing image; deleting a meal image is an explicit action.

Development database snapshot verified 8 November 2026: all 60 of 60 meal rows have an image path and 54 distinct filenames; some duplicate dish variants share an appropriate dish image. This is a local development observation, not a statement about a separate production database or uploaded files.

11 · ORDERING WORKFLOW

11. Meal reservation and ordering workflow

1. The student browses available meals, searches or filters, chooses quantities, and selects a future pickup slot.
2. The server validates the one-use order token and submitted quantity/slot inputs.
3. Within a database transaction, the application locks the slot, checks capacity, inserts the order, locks each available meal row, verifies stock, deducts quantities, inserts order-item price snapshots, and calculates the total from stored prices.
4. On success, the transaction commits and the student is redirected to active orders. On validation or database failure, the transaction rolls back and the student receives an appropriate message.

5. Student cancellation is allowed only for a pending order. Cancellation locks the order and restores stock; administrators can cancel permitted active orders. Completed/cancelled orders are terminal.

Orders are not payment transactions. A unique session token prevents accidental resubmission of the same order form; it is not a persistent shopping cart.

12 · ORDER LIFECYCLE

12. Order status management

Status	Meaning in the interface	Implemented handling
Pending	Order submitted and awaiting preparation.	Student may cancel; administrator may advance, or cancel.
Preparing	Canteen is preparing the order.	Administrator may advance to Ready for Pickup, or cancel.
Ready for Pickup	Order is ready at the designated pickup counter.	Triggers one tracked ready-email notification attempt; administrator may complete or cancel.
Completed	Pickup is recorded as complete.	Terminal; completion timestamp is stored and included in completed-sales reporting.
Cancelled	Order was cancelled.	Terminal; stock is restored once when cancellation is processed.

The supported forward path is Pending → Preparing → Ready for Pickup → Completed. Cancelled is terminal. Valid changes are checked on the server and recorded in order_status_history.

13 · EMAIL DELIVERY

13. Email notifications

When an administrator changes an order to Ready for Pickup, SMROS commits the status change and queues a unique ready_for_pickup notification. PHPMailer sends the email over authenticated SMTP with order items, total, pickup date/time, and pickup instructions. Delivery status, attempt count, timestamps, and a safe failure reason are stored for the student notification page and administrator monitoring.

A failed email does not roll back the committed order status. Administrators may retry eligible failed messages while the order remains ready, up to five total attempts. A successful SMTP provider acceptance cannot guarantee inbox delivery or exactly-once delivery.

SMTP delivery requires administrator-host configured credentials and OpenSSL; supported encrypted pairs are port 587/TLS and port 465/SSL. Without valid SMTP, order status still updates but mail cannot be delivered.

14 · ACCOUNT RECOVERY

14. Password recovery

Active student accounts can request a reset link. The user-facing response is generic regardless of whether an account matches. Requests are rate-limited per hashed IP; prior active tokens for the student are invalidated. A cryptographically random 32-byte token is sent by email, while only its SHA-256 hash is stored. The link expires after 30 minutes and is single-use.

Reset completion checks the token and student account again, locks the token row, validates the new password, hashes it with PHP password_hash(), marks outstanding student tokens used in the same transaction, and sends a password-changed confirmation email. Mail depends on production SMTP and a valid HTTPS SMROS_BASE_URL.

15 · USER NEEDS

15. User stories

- As a student, I want to browse, search, and filter meals with prices and photos so I can choose what to reserve.
- As a student, I want to choose quantities and a pickup time and see a total before I submit an order.
- As a student, I want to view current status, pickup details, past orders, and recorded email delivery state.
- As a student, I want to cancel an eligible pending order so it is removed from active work and its stock is returned.
- As a student, I want to reset my password by email without exposing whether another email belongs to an account.
- As an administrator, I want to manage menu details, images, availability, and stock so the student catalog is current.
- As an administrator, I want to manage pickup capacity and process valid status changes so orders can be fulfilled safely.
- As an administrator, I want to manage student access and preserve account-linked order history.
- As an administrator, I want filtered reports, CSV exports, activity history, and email monitoring to review operations.

16 · VERIFICATION

16. Acceptance criteria

- Valid active accounts log in to the dashboard matching their stored role; inactive accounts and cross-role page requests are rejected.
- A student can search/filter available meals, adjust quantities, select a future slot, and submit an order with a server-computed total.
- Order placement fails safely when a meal is unavailable/out of stock or a pickup slot is full; failed transactions do not leave partial order/stock changes.
- Pending student cancellation and permitted administrator cancellation return stock exactly once; disallowed transitions are rejected and history is recorded.
- Meal image upload rejects mismatched, invalid, oversized, or unsafe images and stores accepted images in the local uploads directory with generated names.
- A Ready for Pickup transition remains committed even if SMTP fails; a notification attempt and visible failure state are recorded for operational follow-up.
- Student reset links are single-use and expire; the database stores a token hash rather than the raw token.
- Report filters and CSV exports use selected criteria; activity and notification views show recorded data without implying complete audit coverage or guaranteed mail receipt.

17. Expected outcomes

- Students can make pickup reservations against current menu stock and slot capacity and can follow recorded order progress.
- Administrators can maintain the operational catalog and student access while retaining linked order records.
- Order totals, stock changes, and status history are backed by stored transactional records.
- Ready-notification attempts and operational failures are visible for review without making unsupported delivery guarantees.
- Current application behavior remains distinguishable from future work such as payments, push updates, or demand prediction.