📘 Product Design Document (PDD)

Product Name

***Mobile_User* – Order Management & Delivery Address Flow**

Document Owner

Product Owner

Version

1.0

Status

Final

---

1. Product Vision

Provide a **simple, reliable, and user-friendly order experience** where:

- A delivery address is always available before checkout
- Users can optionally change the delivery address after ordering
- Order data is persistent, transparent, and auditable
- The system remains scalable for enterprise growth

The product must reduce confusion, prevent lost deliveries, and maintain clean separation between **default addresses (Set A)** and **optional override addresses (Set B)**.

---

2. Business Objectives

- Ensure every order has a valid delivery address
- Minimize checkout friction
- Allow address updates without modifying original user data
- Maintain data integrity and auditability
- Support REST-based API architecture

---

3. User Personas

3.1 End User (Customer)

- Places orders
- Expects a confirmed delivery address
- May want to change delivery location after ordering

3.2 System Administrator

- Manages products, orders, and tracking
- Requires consistent data models

3.3 API Consumer

- Uses Postman / external services
- Needs predictable API responses

---

4. Functional Requirements

4.1 Address Sets

Set A – Default Address

- Mandatory before checkout
- Automatically applied to new orders
- Not editable from the Orders UI
- Stored in addresses table

Set B – Whitelisted Addresses

- Optional
- Can override delivery address per order
- Applied **only if user explicitly selects**
- Stored in addresses_whitelist table

4.2 Order Creation Flow

1. User adds products to cart

2. System calculates total price
3. User selects **Set A address** before checkout
4. Order is created with:
    - address_id (Set A)
    - whitelist_address_id = NULL
5. Order becomes immutable except for whitelist override

4.3 Order Viewing Flow

- User navigates to **My Orders**
- Each order displays:
    - Order ID
    - Status
    - Tracking Number
    - Total Amount
    - Delivery Address (resolved logic)

Delivery Address Resolution Logic:

- If whitelist_address_id exists → use Set B
- Else → fallback to Set A

4.4 Address Update Flow (Post-Order)

- UI displays link: **“Want to change/update the delivery address?”**
- Clicking reveals Set B dropdown
- Selecting address:
    - Updates only orders.whitelist_address_id
    - Does NOT modify Set A
    - Persists across refresh

4.5 Order Cancellation

- Available until order is shipped
- Soft delete implementation
- Affects:
  - orders.deleted_at
  - order_tracking.deleted_at

4.6 Authentication & Session

- Session-based authentication
- All order APIs require login
- Logout invalidates session

---

5. Non-Functional Requirements

5.1 Performance

- Order list fetch ≤ 500ms
- Optimized joins for address resolution

5.2 Security

- Session validation on every API
- User isolation by user_id
- No direct address mutation via Orders UI

5.3 Reliability

- Address persistence guaranteed
- No UI-only state changes
- Database is source of truth

---

6. Technical Architecture

6.1 Frontend

- HTML + CSS + Vanilla JavaScript
- Fetch API
- Stateless rendering

6.2 Backend

- PHP (REST API)
- PDO + MySQL
- JSON communication

6.3 Database Tables

- users
- products
- cart
- orders
- order_items
- order_tracking
- addresses (Set A)
- addresses_whitelist (Set B)

---

7. API Design

7.1 GET /api/orders_api.php

Returns all active orders with:

- Delivery address resolved
- Order items
- Available whitelist addresses

7.2 PATCH /api/orders_api.php

Updates:

- orders.whitelist_address_id

7.3 DELETE /api/orders_api.php

Soft cancels order

---

8. UX Principles

- No unnecessary dropdowns
- Default behavior must be automatic
- Optional actions must be explicit
- System state must survive refresh

---

9. Success Metrics

- 0 orders without delivery address
- 100% address persistence after refresh
- Reduced support tickets related to delivery
- Clear audit trail for address changes

---

10. Out of Scope (Current Phase)

- Address history timeline
- Delivery ETA calculation
- Multi-shipment orders
- Admin dashboard

---

11. Future Enhancements

- Lock address after shipping
- Address change cutoff rules
- Email notification on address update
- Delivery confirmation

---

12. Final Product Statement

This implementation delivers a **robust, enterprise-ready order and delivery system** that balances automation with user control while preserving data integrity and scalability.

Approved by Product Owner

I've prepared a **complete Product Design Document (PDD)** for your entire workflow from a **Product Owner perspective** and placed it in the canvas.

What this PDD covers (aligned with everything you built):

- ✅ Cart → Checkout → Orders → Tracking (end-to-end)
- ✅ **Set A (default address before checkout)** – mandatory
- ✅ **Set B (whitelisted address after order)** – optional override
- ✅ Correct delivery address resolution logic
- ✅ Persistent behavior after refresh
- ✅ Cancel order & logout retained
- ✅ REST API + PHP + MySQL architecture
- ✅ UX rules that explain *why* Set A is hidden and Set B is optional
- ✅ Enterprise-ready structure (scalable, auditable)

This PDD is suitable for:

- 📌 Final project submission
- 📌 Enterprise documentation
- 📌 System handover
- 📌 Resume / portfolio explanation
- 📌 Product review or demo