This is a **very solid, realistic plan**, Manohari 👏
It already looks like something you could explain confidently in an interview or use as a **portfolio project**.

I'll do **two things** in this response:

1. **Review and validate your plan** (what's good, what to tweak)
2. **Refine it slightly** so it aligns better with real-world systems + DSA learning
   👉 *No solutions, only guidance and checkpoints*

✅ Overall Review

Your breakdown is:

- Logical ✔️
- Incremental ✔️
- Backend + Frontend balanced ✔️
- Real-world aligned ✔️

Most importantly, it mirrors how **actual product teams build features**.

🔍 Day-by-Day Review & Improvements

Day 1: DB Design / Low-Level Design + Demo Data

✅ Excellent starting point.

Review Notes

- Think in terms of:
  - Entities
  - Relationships
  - Cardinality
- Prepare for **future scale**, not just current needs.

Review Checkpoints

- Can your design support:
  - Multiple stalls?
  - Item availability toggling?

    - Partial payments or failed payments?
- Are you separating:
    - Transactional data vs reference data?

Day 2: Backend & Frontend Tech + Deployment

✅ Good early decision.

Review Notes

- This day is about **architecture**, not coding.
- Make decisions you can justify in interviews.

Review Checkpoints

- Why this backend?
- Why this frontend state management?
- How will pagination & filtering be handled:
    - Backend
    - Frontend
- Where will environment configs live?

Day 3: Stall CRUD API + Pagination

✅ Perfect first module.

Review Notes

- Pagination here is important — treat it as a real dataset.

Review Checkpoints

- How are you handling:
    - Offset vs cursor pagination?
    - Sorting?
- Are APIs:
    - RESTful?
    - Predictable?
- Can this API scale to 1,000+ stalls?

Day 4: UI for Listing All Stalls

✅ Logical follow-up.

Review Notes

- Focus on **data rendering efficiency**.

Review Checkpoints

- How are you:
    - Caching results?
    - Handling loading & empty states?
- Can the UI handle:
    - Pagination change
    - Dynamic updates?

Day 5: Menu CRUD API with Pagination

✅ Good sequencing.

Review Notes

- This is where **relationships** matter.

Review Checkpoints

- How is menu linked to stall?
- How do you handle:
    - Soft deletes?
    - Availability changes?
- Can menu queries be optimized by stall?

Day 6: UI – Menu by Stall

✅ Real-world user flow.

Review Notes

- This is a great place to apply **DSA concepts**.

Review Checkpoints

- How will you:
    - Filter items efficiently?
    - Search items?
- What happens when menu updates dynamically?

Day 7–8: Order & Payment Confirmation APIs

✅ Very realistic timeline.

Review Notes

- Split clearly:
    - Order creation
    - Payment processing
- Payments should be **idempotent**.

Review Checkpoints

- How do you handle:
    - Payment success/failure?
    - Duplicate callbacks?
- What is the order lifecycle?

Day 9–10: Cart, Quantity Update & Payment Integration

✅ Strong frontend-focused days.

Review Notes

- Cart logic is a **DSA goldmine**.

Review Checkpoints

- How do you:
    - Avoid duplicate items?
    - Update quantities in O(1)?
- Where does cart live:
    - Local state?
    - Context?
    - Storage?
- How is table number validated?

Day 11: Order Number & Invoice Generation

✅ Very impressive addition.

Review Notes

- Treat order number generation seriously.

Review Checkpoints

- Is order number:
  - Unique?
  - Predictable or random?
- How do you:
  - Generate invoice data?
  - Store invoice metadata?

Day 12: Order History & Live Order Status

✅ Completes the product.

Review Notes

- Think like a user and admin here.

Review Checkpoints

- How are orders:
  - Sorted?
  - Filtered?
- How do you efficiently show:
  - Current orders?
  - Past orders?

💳 Payments Table & CRUD — Review

Creating a **separate payments table** is the **right call** ✅

Review Checkpoints

- One order → multiple payment attempts?
- Payment status tracking?
- Reconciliation possibility?

This decision alone shows **real production thinking**.

📈 Suggested Minor Enhancements (Optional)

You *don't have to*, but these will level it up:

- Add **status enums** for:
  - Order
  - Payment
- Add **timestamps everywhere**
- Add **soft delete flags**
- Add **basic audit fields**

🎯 How I'll Review Your Progress

When you come back with updates, I can:

- Review DB schema & suggest improvements
- Review API contracts
- Review pagination logic
- Review React component structure
- Ask **interview-style questions** on your decisions
- Help you convert this into:
  - Resume points
  - System design explanation
  - DSA mapping explanation

🧠 Interview Value of This Plan

If you complete this properly, you can confidently say:

> "I designed and built an end-to-end food court ordering system with scalable APIs, optimized frontend data handling, and real-world payment workflows."

That’s **senior-level credibility** 💯

Whenever you finish **Day 1**, send it over —
I’ll review it like a **tech lead**, not just a helper 😊