
🧭 TICKIFY DEVELOPMENT ROADMAP (Backend Plan)

✅ PHASE 1 — Core Accounts & Auth (Done)

Tables used: `users`, `merchants`, `documents`, `audit_logs`
Features Completed:

- ✅ Register: customer, merchant, admin (seeded)
- ✅ Login with JWT
- ✅ Auth middleware (token-based)
- ✅ Role-based access (`requireRole`)
- ✅ Merchant verification flow (Admin can approve/reject)
- ✅ Merchant can create/update/delete events
- ✅ Protected routes working perfectly

✅ **Status:** Complete 🎉

🏗️ PHASE 2 — Ticketing System (Next Step)

Tables involved: `events`, `ticket_types`

Goal: Allow merchants to create and manage **ticket types** for each event.

Endpoints:

Method	Endpoint	Role	Description
`POST`	`/api/tickets`	Merchant	Create a new ticket type for an event
`GET`	`/api/tickets/:eventId`	Public	View all ticket types for a specific event
`PUT`	`/api/tickets/:ticketId`	Merchant	Update ticket details (price, quantity, etc.)

`DELETE`	`/api/tickets/:ticketId`	Merchant	Delete a ticket type

✅ **Expected Output:**
Merchants can set ticket tiers (VIP, Standard, Early Bird, etc.)
Customers can view available tickets for each event.

💳 PHASE 3 — Orders & Payments

Tables involved: `orders`, `order_items`, `ticket_types`

Goal: Let customers buy tickets and place orders.

Endpoints:

Method	Endpoint	Role	Description
`POST`	`/api/orders`	Customer	Create a new order for one or more tickets
`GET`	`/api/orders`	Customer	View all customer orders
`GET`	`/api/orders/:id`	Customer	View specific order details
`PUT`	`/api/orders/:id/cancel`	Customer	Cancel a pending order
`POST`	`/api/orders/:id/pay`	Customer	Simulate payment and update order to "paid"

✅ **Expected Output:**

- When order is created, reduce `available_quantity` in `ticket_types`.
- When paid, store payment reference (for now, mock it).
- Generate a `qr_code_data` for each ticket in `order_items`.

🎟️ PHASE 4 — QR Code & Ticket Validation

Tables involved: `order_items`, `orders`, `ticket_types`

Goal: Generate QR codes for purchased tickets and validate them on event entry.

Endpoints:

Method	Endpoint	Role	Description
`GET`	`/api/tickets/:orderId`	Customer	View QR codes for purchased tickets
`POST`	`/api/tickets/scan`	Merchant	Validate ticket (mark `ticket_status` as "used")

✅ **Expected Output:**
When scanned by the merchant at the venue, ticket becomes "used".
Helps prevent duplicate entries.

💰 PHASE 5 — Refunds System

Tables involved: `refunds`, `orders`

Goal: Customers can request refunds and merchants/admins can approve/reject them.

Endpoints:

Method	Endpoint	Role	Description
`POST`	`/api/refunds/:orderId`	Customer	Request a refund
`GET`	`/api/refunds`	Admin/Merchant	View refund requests
`PUT`	`/api/refunds/:id`	Admin/Merchant	Approve or reject a refund

✅ **Expected Output:**
Refund workflow complete, with status tracking (`requested`, `approved`, etc.).

📜 PHASE 6 — Audit Logs (Auto-Tracking)

Tables involved: `audit_logs`

Goal: Automatically track all major user activities.

Examples:

- Admin verifies merchant → log: *"Merchant verified"*
- Merchant creates event → log: *"Event created"*
- Customer purchases ticket → log: *"Order created"*

✅ **Expected Output:**
Every major action writes to `audit_logs` table automatically.

🧩 PHASE 7 — Admin Dashboard Endpoints

Tables involved: All

Goal: Build admin endpoints to view and manage everything.

Endpoints:

Method	Endpoint	Description
`GET`	`/api/admin/users`	List all users
`GET`	`/api/admin/merchants`	List all merchants
`GET`	`/api/admin/events`	List all events
`GET`	`/api/admin/orders`	List all orders
`GET`	`/api/admin/logs`	View all audit logs

✅ **Expected Output:**
Admin gets a 360° view of system activity.

🚀 FINAL STEP — Integration Tests & Polishing

Once all backend logic is solid:

- Add integration tests (Postman / Jest)

- Add pagination & filtering for big lists
- Add error handling & response standardization
- Prepare API documentation (Swagger or Postman Collection)

📘 Summary: Current Status & Next Action

Phase	Name	Status
1	Auth, Merchant Verification	✅ Done
2	Ticket Management	🔜 **Next**
3	Orders & Payments	⏳ Later
4	QR Validation	⏳ Later
5	Refund System	⏳ Later
6	Audit Logs	⏳ Later
7	Admin Dashboard	⏳ Later

Would you like me to draw the **Phase 2 (Ticket Management)** implementation plan next — showing how merchants will create and manage ticket types (including sample JSON, routes, and controller logic)?