Box Cricket Booking System - Full Technical Specification (v2)

1. Overview

This document defines the system architecture, technology stack, multi-tenant routing model, database schema, APIs, and user flows for the Box Cricket Booking SaaS platform.

The system supports:

- Multiple businesses, each mapped to a subdomain: abc.domainname.com
- Each business can have multiple venues, and each venue can have multiple courts.
- Role-based access: SUPER_ADMIN, MANAGER, USER.
- Dynamic pricing (peak/off-peak), recurring availability, one-time blocks (tournaments/maintenance).
- Online payments for Indian users and PDF invoice generation.
- Timezone: IST.

2. Multi-Tenant Routing by Subdomain

Businesses are represented as organizations and are accessed via subdomains:

abc.domainname.com -> organization with slug = 'abc'

Routing rule:

- On each request, the backend reads the Host header, extracts the subdomain (e.g. 'abc'), and resolves it to an organization row (business) using the slug field.
- All queries for venues, courts, bookings, pricing, etc., are always filtered by organization_id.

Onboarding flow for a new business (by SUPER_ADMIN):

1. SUPER_ADMIN opens the admin panel and clicks "Add Business".
2. Fills: business name, desired subdomain (slug), contact details.
3. System creates:
- organizations row with name, slug = 'abc'
- primary manager user with role = MANAGER
4. SUPER_ADMIN or manager then creates:
- One or more venues (address, city, pincode)
- One or more courts under those venues
5. While assigning a manager, SUPER_ADMIN can select multiple courts from a dropdown (multi-select). This populates the court_managers mapping table.

3. Tech Stack

Frontend (Web): React or Next.js + MUI + React Query

Mobile App (Phase 2): React Native / Expo (same APIs)

Backend: FastAPI (Python)

Database: PostgreSQL (preferred) or MySQL

Authentication: Google OAuth (backend verifies, issues JWT)

Payments (India): Razorpay / Cashfree / PayU / PhonePe Gateway

PDF Invoices: Python PDF library (e.g., WeasyPrint / ReportLab)

Hosting: AWS or GCP (decision pending)

Timezone: IST only

4. User Roles & Permissions

SUPER_ADMIN:

- Can create organizations (businesses) and assign primary managers.
- Can create venues and courts under any organization.
- Can view all data across organizations.

MANAGER:

- Belongs to a single organization.
- Can manage only courts assigned to them (via court_managers mapping).
- Can configure recurring availability, one-time overrides, and pricing rules.
- Can see bookings and revenue for their courts.
- Can manually block/unblock slots (including for tournaments).
- Cannot issue refunds through the app (refunds handled outside).

USER:

- Belongs implicitly to an organization via subdomain.
- Can view venues and courts for that organization.
- Can view availability, select slots, make a single booking for one court at a time.
- Can pay, then download invoice and view booking history.

5. Database Entities (High-Level)

organizations (businesses mapped to subdomains)

users

venues

courts

court_managers

court_recurring_availability

court_date_overrides

pricing_rules

bookings

payments

5.1 organizations

```
organizations (
  id                uuid PK,
  name              varchar NOT NULL,
  slug              varchar UNIQUE NOT NULL,  -- used for subdomain, e.g. 'abc'
  is_active         boolean DEFAULT true,
  created_at        timestamptz,
  updated_at        timestamptz
)
```

5.2 users

```
users (
  id                uuid PK,
  organization_id   uuid FK -> organizations(id),
  google_id         varchar UNIQUE,          -- or firebase_uid
  email             varchar UNIQUE NOT NULL,
  first_name        varchar NOT NULL,
  last_name         varchar,
  phone             varchar,
  address           text,
  pincode           varchar(10),
  role              varchar CHECK (role IN ('SUPER_ADMIN','MANAGER','USER')),
  is_active         boolean DEFAULT true,
  last_login_at     timestamptz,
  created_at        timestamptz,
  updated_at        timestamptz
)
```

5.3 venues

```
venues (
  id                uuid PK,
  organization_id   uuid FK -> organizations(id),
  name              varchar NOT NULL,
  address_line1     varchar,
  address_line2     varchar,
  city              varchar,
  state             varchar,
  pincode           varchar(10),
  latitude          numeric(10,7),     -- optional
  longitude         numeric(10,7),     -- optional
  is_active         boolean DEFAULT true,
  created_at        timestamptz,
  updated_at        timestamptz
)
```

5.4 courts

```
courts (
  id                   uuid PK,
  venue_id             uuid FK -> venues(id),
  name                 varchar NOT NULL,
  description          text,
  min_booking_minutes  int NOT NULL,
  max_booking_minutes  int NOT NULL,
  is_active            boolean DEFAULT true,
  created_at           timestamptz,
  updated_at           timestamptz
)
```

5.5 court_managers

```
court_managers (
  id                uuid PK,
  court_id          uuid FK -> courts(id),
  manager_id        uuid FK -> users(id),
  UNIQUE (court_id, manager_id)
)
```

5.6 court_recurring_availability

```
court_recurring_availability (
  id                uuid PK,
  court_id          uuid FK -> courts(id),
  day_of_week       smallint CHECK (day_of_week BETWEEN 0 AND 6),
  start_time        time NOT NULL,
  end_time          time NOT NULL,
  is_active         boolean DEFAULT true,
  created_at        timestamptz,
  updated_at        timestamptz
)
```

5.7 court_date_overrides

```
court_date_overrides (
  id                uuid PK,
  court_id          uuid FK -> courts(id),
  date              date NOT NULL,
  override_type     varchar CHECK (override_type IN ('OPEN','CLOSE')),
  start_time        time,
  end_time          time,
  reason            varchar,
  created_by        uuid FK -> users(id),
  created_at        timestamptz
)
```

5.8 pricing_rules

```
pricing_rules (
  id                uuid PK,
  court_id          uuid FK -> courts(id),
  organization_id   uuid FK -> organizations(id),
  rule_type         varchar CHECK (rule_type IN ('RECURRING','ONE_TIME')),
  day_of_week       smallint,
  date              date,
  start_time       time NOT NULL,
  end_time         time NOT NULL,
  price_per_30_min numeric(10,2) NOT NULL,
  is_peak          boolean DEFAULT false,
  priority         int DEFAULT 0,
  is_active        boolean DEFAULT true,
  created_by       uuid FK -> users(id),
  created_at       timestamptz,
  updated_at       timestamptz
)
```

5.9 bookings

```
bookings (
  id               uuid PK,
  organization_id  uuid FK -> organizations(id),
  user_id          uuid FK -> users(id),
  court_id         uuid FK -> courts(id),
  venue_id         uuid FK -> venues(id),
  start_time       timestamptz NOT NULL,
  end_time         timestamptz NOT NULL,
  total_price      numeric(10,2) NOT NULL,
  status           varchar CHECK (status IN (
                       'PENDING_PAYMENT',
                       'CONFIRMED',
                       'FAILED',
                       'CANCELLED_MANUAL'
                     )),
  invoice_number   varchar,
  notes            text,
  created_at       timestamptz,
  updated_at       timestamptz
)
```

5.10 payments

```
payments (
  id               uuid PK,
  booking_id       uuid FK -> bookings(id),
  gateway          varchar,
  gateway_order_id varchar,
  gateway_payment_id varchar,
  amount           numeric(10,2),
  currency         varchar(3) DEFAULT 'INR',
  status           varchar CHECK (status IN ('CREATED','SUCCESS','FAILED')),
  raw_request      jsonb,
  raw_response     jsonb,
  created_at       timestamptz,
  updated_at       timestamptz
)
```

6. Booking & Business Rules

- Time slots: 30-minute increments, flexible start and end.

- Buffer: 5 minutes inside the booking.

- Multi-slot = single booking; one booking bound to a single court.

- User cannot cancel after payment; refunds are handled offline.
- Admin/Manager manually frees slots by setting status to CANCELLED_MANUAL.
- Overlap prevention only considers PENDING_PAYMENT and CONFIRMED bookings.
- FAILED and CANCELLED_MANUAL do not block new bookings.
- No hard deletes; entities are soft-deleted via is_active or status.

7. API Overview

All APIs are scoped by organization (business), resolved from subdomain.

AUTH:

- POST /auth/google-login
- GET /me

USER:

- GET /venues
- GET /courts?venue_id=
- GET /courts/{court_id}
- GET /availability?court_id=&date;=YYYY-MM-DD
- POST /bookings/initiate
- POST /payments/create
- POST /payments/webhook
- GET /bookings/history
- GET /bookings/{id}
- GET /bookings/{id}/invoice

MANAGER:

- GET /manager/courts
- GET /manager/dashboard-metrics
- POST /manager/courts
- PATCH /manager/courts/{id}
- POST /manager/availability/recurring
- POST /manager/availability/override
- POST /manager/pricing-rules
- GET /manager/pricing-rules
- GET /manager/bookings?court_id=&from;=&to;=
- PATCH /manager/bookings/{id}/status

SUPER_ADMIN:

- POST /admin/organizations
- POST /admin/organizations/{id}/managers
- POST /admin/venues

- POST /admin/courts
- PATCH /admin/users/{id}/role
- GET /admin/organizations
- GET /admin/exports/bookings
- GET /admin/exports/payments

8. Client User Flow with API Calls

1) User opens abc.domainname.com.

- Frontend extracts 'abc' and backend resolves organization via subdomain.

2) Login with Google:

- POST /auth/google-login
- Backend verifies Google token, upserts user, returns JWT.

3) User selects venue and court:

- GET /venues
- GET /courts?venue_id={venue_id}

4) Check availability:

- GET /availability?court_id={court_id}&date;=YYYY-MM-DD

5) Initiate booking:

- POST /bookings/initiate with court_id, start_time, end_time.
- Backend validates overlap, price, creates PENDING_PAYMENT booking and CREATED payment.

6) Create payment order:

- POST /payments/create (or merged with initiate).
- Returns gateway payload for Razorpay/other.

7) Payment:

- User completes payment on gateway UI.

8) Webhook:

- POST /payments/webhook
- On success: mark payment SUCCESS and booking CONFIRMED, assign invoice_number.
- On failure: mark payment FAILED and booking FAILED (slot freed).

9) Booking history & invoice:

- GET /bookings/history
- GET /bookings/{id}/invoice to download PDF with QR.

9. Manager & Super Admin Flows (Summary)

Manager (abc.domainname.com):

- GET /manager/courts
- Configure recurring availability and overrides.
- Configure pricing rules.
- View bookings and revenue for assigned courts.
- Manually reopen slots via PATCH /manager/bookings/{id}/status.

Super Admin:

- Onboard new businesses via POST /admin/organizations with slug used for subdomain.
- Create primary manager for each business via POST /admin/organizations/{id}/managers.
- Create venues and courts.
- Assign managers to multiple courts via court_managers (multi-select dropdown in UI).
- Export bookings and payments via export endpoints.