Product Requirements Document: Booking Links (Flowkey Platform Integration)

Version: 2.3 **Date:** May 28, 2025 **Author:** Flowkey Product & Engineering Team (AI Assisted)
Status: Updated Draft for Review & Build

Table of Contents

1. Introduction & Vision

1.1. Overview

This document outlines the requirements for the "Booking Links" feature, designed to be seamlessly integrated into the existing Flowkey platform. This feature will empower Flowkey users (admins/instructors) to generate and share one primary, persistent URL for their business. Clients visiting this link will land on a booking page where they can view all services offered by the business, select a service, and then book available time slots. Bookings will automatically update the Flowkey admin's calendar, ensuring real-time synchronization and accurate availability.

1.2. Problem Statement

Currently, Flowkey users often rely on manual processes (email, phone calls, messaging) to manage client bookings. This approach is time-consuming, prone to errors (e.g., double bookings, miscommunication about availability), and lacks the professionalism and efficiency that a modern SaaS platform should offer. Clients also face friction in discovering availability and securing appointments across potentially multiple services offered by a business.

1.3. Proposed Solution

The "Booking Links" feature will provide each Flowkey business with a single, shareable booking link. This link directs clients to a dedicated booking page that lists all publicly available services offered by that business. From this page, clients can select their desired service, view real-time availability (adjusted for their timezone), and complete the booking process. This system, deeply integrated with the admin's backend configurations, aims to automate and streamline the entire booking lifecycle.

1.4. Goals & Objectives

Primary Goals:

- Allow Flowkey users to easily generate and share one persistent booking link for their business, which dynamically presents all their offered services.
- Enable clients to first select a service from the business's offerings on the booking page, and then self-select available time slots based on real-time availability for that specific service.
- Automate the creation of session instances and update the admin's FullCalendar view upon confirmed bookings.
- Ensure booked slots (respecting session capacity defined in session/SessionCategory or equivalent, which already supports N attendees) are no longer presented as available.
- Provide clients with comprehensive booking confirmations (including details and "Add to Calendar" options for Google Calendar and .ics).

Secondary Goals:

- Significantly reduce administrative time spent by Flowkey users on manual scheduling and follow-ups.

- Enhance the professional image and service offering for Flowkey users.
- Improve client satisfaction by providing a convenient, 24/7 booking option for all their services through one entry point.
- Establish a robust foundation for future integration of online payments with bookings.

1.5. Success Metrics

- **Adoption Rate:** % of active Flowkey users (with BusinessProfile) who generate and have at least one active booking link within 3 months post-launch.
- **Booking Volume:** Number of successful client-initiated bookings made via booking links per week/month.
- **Admin Time Saved:** (Qualitative) Survey feedback from users on perceived time savings. (Quantitative) Reduction in manually created/modified sessions by admins that could have been client-booked.
- **Client Conversion Rate:** % of clients visiting a booking link who complete a booking.
- **System Performance:**
    - Booking link page load time (target < 2s).
    - Availability calculation API response time (target < 500ms for P95).
    - Booking confirmation time (end-to-end, target < 3s).
- **Error Rate:** Number of failed booking attempts due to system errors (target < 0.1%).
- **Feature Usage:** Frequency of use for advanced settings (e.g., approval workflow, buffer times).

1.6. Target Users

- **Flowkey Admins/Users:** Instructors, studio owners, and other service providers using the Flowkey platform to manage their business, staff, clients, and sessions. They interact with the React SPA and rely on the Django backend.
- **Clients:** End-customers of Flowkey Admins who wish to book services. They will interact with the public-facing booking link pages.

1.7. Glossary

- **Booking Link:** A single, unique, persistent URL per business that a Flowkey Admin can share, leading to a client-facing booking page listing all offered services.
- **Service:** A bookable offering defined by the Flowkey Admin, corresponding to a SessionCategory within the session/ app, including duration, capacity, and price (future).
- **Availability Block:** Time periods defined by the Flowkey Admin in their session/ schedule (e.g., WorkingHours model) as being open for bookings, excluding ExceptionDates.
- **Time Slot:** A specific, bookable instance of a Service on a given day, calculated based on Service duration and Availability Blocks.

- **Capacity:** The maximum number of clients that can book a single Time Slot (e.g., for group classes), managed within the SessionCategory or Session model. Existing system supports N attendees per session/slot.
- **PRD:** Product Requirements Document.
- **SPA:** Single Page Application (referring to the existing Flowkey admin frontend and the new client booking interface).
- **DRF:** Django REST Framework.
- **JWT:** JSON Web Token.

2. User Stories & Flows

2.1. Flowkey Admin (User Persona: Sarah, a swim coach using Flowkey)

User Stories:

- **US-A1 (Link Generation & Management):** "As Sarah, I want to view and copy my business's unique, persistent booking link from my Flowkey settings to easily share it. I also want to configure global settings for this link, like booking approval and buffer times."
- **US-A2 (Multi-Service Presentation):** "As Sarah, I want my single booking link to lead clients to a page where they can see and select from all my services that I've marked as 'publicly bookable' (SessionCategory items)."
- **US-A4 (Availability Sync):** "As Sarah, I want my booking links to automatically reflect my availability based on my WorkingHours and ExceptionDates configured in the Flowkey session/ app."
- **US-A5 (Session Duration Control):** "As Sarah, I want the duration defined in my SessionCategory (e.g., '60 minutes') to dictate the length of bookable time slots on the link for that specific service."
- **US-A6 (Buffer Time Config):** "As Sarah, I want to set a global buffer time (time gap between sessions) for my business, which automatically applies when availability is calculated for any service booked via my link."
- **US-A7 (Booking Approval Workflow):** "As Sarah, I want an option to require my manual approval for bookings. When a booking needs approval, I want to receive an in-platform notification (top nav icon) and see it on my dashboard. I also want to configure how long a pending request remains active before expiring. (Future: WhatsApp notifications)."
- **US-A8 (Booking Notifications):** "As Sarah, I want to receive immediate email notifications (via SendGrid) and in-platform notifications (via Mantine Notifications / Zustand) when a client books (or requests a booking if approval is on)."
- **US-A9 (Calendar Integration):** "As Sarah, I want confirmed bookings from links to automatically create/update corresponding events in my Flowkey FullCalendar view and backend Session records, respecting the defined SessionCategory.capacity."
- **US-A11 (Client Info Capture):** "As Sarah, I want new clients booking via a link to be added to my client/ list, or matched to existing clients if their email/phone matches."

Admin User Flow (Booking Link Settings - Illustrative):

1. Sarah logs into the Flowkey React SPA (flowkey.vercel.app).
2. Navigates to "Settings" -> "Public Booking Page" (or similar, new section).
3. **View:** Displays her business's single persistent booking link (e.g., `flowkey.com/book/sarahs-swim-school` or `[business_slug].flowkey.com`). Options to copy link.
4. **Configuration Options (Mantine Form elements):**
    - Booking Approval Required: Mantine Switch (Yes/No).
    - Pending Booking Expiry (hours): Mantine NumberInput (e.g., 24, 48 hours).
    - Global Buffer Time Between Sessions (minutes): Mantine NumberInput.
    - Custom Client Message (Optional): Mantine Textarea (for display on booking/confirmation page).
    - Enable Public Booking Page: Mantine Switch (to activate/deactivate the entire booking link functionality).
5. Sarah saves settings. Axios PUT request to update business-level booking settings (e.g., to BusinessBookingSettings model). TanStack React Query invalidates relevant queries.

[Diagram Placeholder: Admin User Flow - Booking Link Settings. Flowchart: Login -> Navigate to Public Booking Page Settings -> View Link & Configure Options (Approval, Buffer, Expiry) -> Save Settings.]

2.2. Client (User Persona: John, looking for swim lessons)

User Stories:

- **US-C1 (Access Link):** "As John, I want to click the business's booking link Sarah shared and be taken to a clean, mobile-friendly page listing all her services."
- **US-C2 (Service Selection):** "As John, from the initial page, I want to easily browse and select the specific service I'm interested in (e.g., 'Adult Beginner Swim') to then see its availability."
- **US-C3 (Time Zone Clarity):** "As John, after selecting a service, I want the availability calendar to automatically detect my timezone and display slots in my local time, with an option to change it."
- **US-C4 (Calendar Navigation):** "As John, I want an intuitive calendar to navigate dates and see availability for the chosen service."
- **US-C5 (Slot Selection):** "As John, I want to clearly see and click on available time slots for the selected service to proceed."
- **US-C6 (Booking Form):** "As John, I want to fill a simple form (Name, Email, Phone - validated) to provide my details. Notes field should be optional."
- **US-C7 (Confirmation - On-Screen):** "As John, after submitting: If approval is NOT needed, I see 'Booking Confirmed!'. If approval IS needed, I see 'Booking pending approval'."
- **US-C8 (Confirmation - Calendar):** "As John, upon final confirmation (either immediate or after approval), I want 'Add to Google Calendar' and '.ics download' options."

- **US-C9 (Confirmation - Email/WhatsApp):** "As John, I want to receive an email for my booking status. If approved, it's a confirmation with calendar options. If rejected, the email explains (optionally with a reason) and includes a link to rebook. (Future: WhatsApp for all these)."

Client Booking Flow (Illustrative):

1. John clicks the business's booking link (e.g., `flowkey.com/book/sarahs-swim-school`).
2. **Booking Page - Step 1: Service Selection:**
    - Displays Admin's branding (Business Name, Logo).
    - Lists all publicly available services (SessionCategory items where `is_publicly_bookable=True`) offered by the business, possibly with durations. John selects one.
3. **Booking Page - Step 2: Availability View (for selected service):**
    - Calendar displays available dates for the selected service.
    - Client's timezone is auto-detected; option to change.
    - On date selection, available time slots (factoring duration, admin availability, existing bookings, global buffer, service capacity) are shown.
    - John selects a time slot.
4. **Booking Page - Step 3: Booking Form (Mantine Form with React Hook Form):**
    - Fields: Name, Email, Phone, Notes. Validation messages displayed.
    - John submits. Axios POST to a public booking API endpoint.
5. **Booking Page - Step 4: Confirmation/Status Update:**
    - If auto-confirmed: "Booking Confirmed!" with details and calendar links. Email sent.
    - If approval needed: "Booking pending approval." Email sent acknowledging request.
    - (If pending) Later, John receives an email upon approval (with calendar links) or rejection (with rebook link and optional reason).

[Diagram Placeholder: Client Booking Flow. Flowchart: Click Link -> View/Select Service -> View Calendar/Slots for Service -> Select Slot -> Fill Form -> Submit -> Confirmation Page/Status & Notifications.]

3. Key Features (Detailed)

3.1. Booking Link Generation & Management

Description: Each Flowkey Admin/Business will have one primary, persistent, shareable URL for clients to access their multi-service booking page. Settings for this page are managed globally by the admin. **Requirements:**

- **FR1:** System automatically generates or allows admin to customize a unique, persistent URL slug for their business's booking page (e.g., `flowkey.com/book/[business_slug]`). This slug is associated with their `BusinessProfile` or a new `BusinessBookingSettings` record.
- **FR2:** Admin UI (React/Mantine) to view and copy this single link. Configuration options (booking approval, global buffer time, pending request expiry, custom client message, activate/deactivate public booking page) are managed in a dedicated settings area. **Technical Notes:** A new model `BusinessBookingSettings` linked 1-to-1 with `BusinessProfile` is preferred to hold these global settings. The `public_booking_slug` would be a field on this model.

3.2. Service Presentation & Selection on Booking Page

Description: The client-facing booking page, accessed via the single business link, dynamically lists all services (derived from `session.SessionCategory`) offered by the business for client selection as the first step. **Requirements:**

- **FR1:** The booking page must fetch and display all `SessionCategory` items for the linked business where `is_publicly_bookable=True`.
- **FR2:** Each listed service should clearly show its name, duration, and (future) price.
- **FR3:** Client must select a service before viewing the availability calendar for that specific service. The subsequent availability display is filtered by this selected service. **Technical Notes:** The primary public API endpoint for the booking page (`/book/{business_slug}/services/`) will return the list of these services.

3.3. Session Duration & Slot Calculation

Description: The system dynamically calculates and displays bookable time slots for the selected service, based on its duration, the admin's comprehensive availability, the global buffer time, and the service's capacity. **Requirements:**

- **FR1:** Algorithm segments admin's available time blocks into discrete slots matching the selected `SessionCategory.duration_minutes`.
- **FR2:** The global buffer time (configured by admin) must be factored in when calculating gaps needed between any two sessions.
- **FR3:** Algorithm respects `SessionCategory.capacity` (which already supports N attendees) and existing `Session` instances to determine remaining spots for the selected service. **Technical Notes:** Backend availability calculation service takes `session_category_id` as a key input.

3.4. Availability Management & Time Zone Handling

(Largely the same as v2.2, principles apply directly)

3.5. Client Booking Interface

(Largely the same, but emphasizes the multi-step flow: 1. Select Service -> 2. Select Date/Time -> 3. Fill Details -> 4. Confirmation/Status)

3.6. Booking Form & Validation

(Largely the same as v2.2)

3.7. Booking Confirmation (Email, Calendar Integration, Optional WhatsApp)

Description: Provides immediate feedback to clients based on booking status (confirmed or pending approval) and integrates with their calendars upon final confirmation. All notifications are handled asynchronously. **Requirements:**

- **FR1:** On booking submission:
    - If auto-confirmed: Client sees on-screen "Booking Confirmed!" summary.
    - If approval needed: Client sees on-screen "Booking pending approval."
- **FR2:** Upon final confirmation (either immediate or after admin approval): Confirmation page/update includes "Add to Calendar" buttons (Google Calendar intent link, server-generated .ics file).
- **FR3:** Automated email (via SendGrid & Celery):
    - For pending: "Booking request received."
    - For approved: "Booking Confirmed!" with details and calendar links.
    - For rejected: "Booking Rejected" (admin can optionally add a reason, email includes a link to rebook).
    - For expired pending request: "Booking Request Expired" (optional, with link to rebook).
- **FR4:** (P1 - Post-MVP) WhatsApp confirmations mirroring the email flows, triggered by Celery.
- **FR5:** Admin receives email (via SendGrid & Celery) and in-platform notifications (Mantine Notifications/Zustand, potentially via dashboard update) for new pending requests and directly confirmed bookings.

3.8. Admin Booking Management (Approval Workflow)

Description: Admins can opt to manually approve bookings. The system provides notifications and tools for managing these pending requests, including configurable expiry for pending slots. **Requirements:**

- **FR1:** If booking approval is enabled (global business setting `BusinessBookingSettings.requires_approval`):

    - Client sees "Booking pending approval" on submission. `Booking.status` is `pending_approval`.
    - Admin receives in-platform notification (e.g., badge on top nav icon) and the request appears on their dashboard/dedicated pending list. (Future: WhatsApp notifications).
- **FR2:** Admin UI (React/Mantine Table or List) to view pending bookings with details (Client Name, Service, Requested Time). Actions: Approve, Reject.
- **FR3:** On Approve: `Booking.status` changes to `confirmed`. Client receives "Booking Confirmed!" email (with calendar links). Corresponding `session.Session` instance is created/finalized in the Flowkey calendar. Slot is firmly booked.
- **FR4:** On Reject: `Booking.status` changes to `rejected`. Client receives "Booking Rejected" email (admin can optionally add a reason from the UI; email includes a link to rebook). Slot is released.
- **FR5:** Admin can configure `BusinessBookingSettings.pending_booking_expiry_hours`. A scheduled task (Celery Beat) will periodically check for pending bookings exceeding this duration.
- **FR6:** Expired pending bookings: `Booking.status` changes to `expired_pending`. Slot is released. Client is notified via email (optional, with rebook link). Admin may also be notified. **Technical Notes:** `Booking.status` field with new 'expired_pending' choice. `Booking.pending_request_expires_at`: `DateTimeField` set when status becomes `pending_approval`. Celery Beat task for managing expirations.

3.9. Buffer Time Configuration (Global)

Description: Admins can set a single, global buffer time representing the time gap between sessions. This buffer is automatically factored into availability calculations for all services offered by the business. **Requirements:**

- **FR1:** Admin can set a `global_buffer_time_minutes_between_sessions` in their `BusinessBookingSettings`.
- **FR2:** The availability slot calculation algorithm must use this global buffer when determining the necessary gap required before the start of a potential slot if there's a preceding session, and after the end of a potential slot if there's a subsequent session. **Technical Notes:** This setting is stored in the `BusinessBookingSettings` model (or on `BusinessProfile`).

3.10. Conflict Handling & Real-time Updates

(Largely the same as v2.2, principles apply directly. Real-time updates for MVP will rely on TanStack React Query's refetching; WebSockets are post-MVP).

4. Technical Architecture & Design (Integrating Booking Links)

4.1. Current System Architecture Context

(Remains the same as v2.2 - references the provided ASCII diagram and stack summary)
[Diagram: Current System Architecture - ASCII diagram provided by user context to be inserted here or recreated visually if tool allows.]

4.2. Technology Stack Integration for Booking Links

(Largely the same as v2.2, emphasizing the new `booking/` app and how admin/client UIs fit into existing React SPA patterns or as a new public section).

4.3. Proposed Data Models & Modifications

New Model: `business.BusinessBookingSettings` (or extend `BusinessProfile`)

```
Unset
class BusinessBookingSettings(models.Model):
    business_profile = models.OneToOneField(
        'business.BusinessProfile',
        on_delete=models.CASCADE,
        primary_key=True
    )
    public_booking_slug = models.SlugField(
        max_length=100,
        unique=True,
        null=True,
        blank=True
    ) # e.g., "sarahs-swim-school"
    requires_approval = models.BooleanField(default=False)
    global_buffer_time_minutes_between_sessions =
models.PositiveIntegerField(default=0)
    pending_booking_expiry_hours =
models.PositiveIntegerField(default=24)
    custom_message_client = models.TextField(blank=True,
null=True)
```

```
    is_booking_page_active = models.BooleanField(default=True)
    updated_at = models.DateTimeField(auto_now=True)
```

Model: `booking.Booking`

```
Unset
class Booking(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid.uuid4,
editable=False)
    business_profile = models.ForeignKey(
        'business.BusinessProfile',
        on_delete=models.CASCADE
    )
    session_category = models.ForeignKey(
        'session.SessionCategory',
        on_delete=models.PROTECT
    )
    flowkey_user = models.ForeignKey(
        'users.User',
        related_name='admin_bookings',
        on_delete=models.CASCADE
    ) # The admin who owns the business_profile
    client = models.ForeignKey(
        'client.Client',
        on_delete=models.SET_NULL,
        null=True,
        blank=True
    )
    client_name = models.CharField(max_length=255)
    client_email = models.EmailField()
    client_phone = models.CharField(max_length=30, blank=True,
null=True)
    notes_from_client = models.TextField(blank=True, null=True)
    start_time_utc = models.DateTimeField()
```

```
    end_time_utc = models.DateTimeField()
    booked_in_client_timezone = models.CharField(max_length=100)
    STATUS_CHOICES = [
        ('pending_approval', 'Pending Approval'),
        ('confirmed', 'Confirmed'),
        ('cancelled_by_client', 'Cancelled by Client'),
        ('cancelled_by_admin', 'Cancelled by Admin'),
        ('completed', 'Completed'),
        ('rejected', 'Rejected'),
        ('expired_pending', 'Expired Pending Approval'),
    ]
    status = models.CharField(
        max_length=20,
        choices=STATUS_CHOICES,
        default='confirmed'
    )
    pending_request_expires_at = models.DateTimeField(null=True,
blank=True) # Set when status becomes pending_approval
    rejection_reason = models.TextField(blank=True, null=True) #
Admin-provided reason for rejection
    gcal_event_id_client = models.CharField(max_length=255,
blank=True, null=True)
    created_session = models.OneToOneField(
        'session.Session',
        on_delete=models.SET_NULL,
        null=True,
        blank=True
    )
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

Modifications to `session.SessionCategory`:

```
Unset
class SessionCategory(models.Model):
    # ... existing fields ...
    is_publicly_bookable = models.BooleanField(default=True) #
Default to true for ease of setup, admin can disable.
    duration_minutes = models.PositiveIntegerField() # Ensure
robustly defined
    capacity = models.PositiveIntegerField(default=1) # Ensure
robustly defined
    # ... existing fields ...
```

[Diagram Placeholder: ERD - Showing BusinessBookingSettings, Booking, and their relationships to BusinessProfile, users.User, session.SessionCategory, client.Client, session.Session.]

4.4. API Design for Booking Links

Admin Endpoints (JWT Authenticated):

- `GET /api/v1/business-booking-settings/`: Retrieve current business's booking settings.
- `PUT /api/v1/business-booking-settings/`: Update business's booking settings. (Manages slug, approval, buffer, expiry, active status).
- `GET /api/v1/bookings/pending-approval/`: List Bookings with `status='pending_approval'` for the admin.
- `POST /api/v1/bookings/{booking_id}/approve/`: Change `Booking.status` to `confirmed`.
- `POST /api/v1/bookings/{booking_id}/reject/`: Body: `{"reason": "Optional rejection reason"}`. Change `Booking.status` to `rejected`.

Client-Facing Public Endpoints (No JWT, Rate Limited):

- `GET /book/{business_slug}/services/`: Returns list of publicly bookable `SessionCategory` items for the business (name, duration, description, future price).
- `GET /book/{business_slug}/availability/`: Query Params: `session_category_id`, `start_date`, `end_date`, `client_timezone`.
- `POST /book/{business_slug}/bookings/`: Request Body: Client details, `session_category_id`, `start_time_utc`.

4.5. Key Algorithms for Booking Links

- **Availability Slot Generation (Backend):**
    - Inputs now include `business_slug` (to fetch `BusinessBookingSettings` like global buffer) and `session_category_id`.
    - The global buffer is applied between potential slots.
- **Conflict Resolution (Booking Submission):** *(Largely the same as v2.2)*
- **Pending Booking Expiry (Backend - Celery Beat Task):**
    - Scheduled task (e.g., every 5-15 minutes).
    - Queries `Booking` records where `status='pending_approval'` AND `pending_request_expires_at <= now()`.
    - For each expired booking:
        - Update status to `expired_pending`.
        - Enqueue Celery task to notify client (optional).
        - Enqueue Celery task to notify admin (optional).
        - (Availability is implicitly freed up as these bookings are no longer 'holding' slots).

(Authentication & Authorization, Validation & Error Handling, Logging & Monitoring, Async Tasks, Data Flow & Caching, Security, Scalability sections will be adapted from v2.2, ensuring they align with the single business link, global buffer, and detailed approval/expiry workflow. Celery's role for notifications and expiry management becomes more critical.)

5. Design & UX Considerations

(Largely the same as v2.2, but client booking page now has a clear Step 1 for Service Selection. Admin settings UI simplifies to managing one link's global parameters.)

5.2. Wireframes/Mockup Descriptions (Key Screens)

- **Admin - Public Booking Page Settings (React/Mantine):**
    - Displays the single, copyable business booking link.
    - Form elements for: Booking Approval Required (Switch), Pending Booking Expiry (hours) (NumberInput), Global Buffer Time Between Sessions (minutes) (NumberInput), Custom Client Message (Textarea), Enable Public Booking Page (Switch).
- **Client - Public Booking Page - Step 1: Service Selection (React/Mantine/Tailwind):**
    - Clear list/cards of all publicly bookable services for the business. Each shows name, duration. Client clicks to select one.
- **Client - Public Booking Page - Step 2: Date/Time Selection (React/Mantine/Tailwind):**
    - Calendar and time slots shown for the service selected in Step 1.
- **Admin - Dashboard / Notifications Area:**

    - Clear indication of new pending booking requests (e.g., a "Pending Bookings" widget or counter).
    - List of pending bookings with client name, service, time, and Approve/Reject actions.

6. Considerations for Booking Links Feature Development

(Updated based on new clarifications)

6.1. Strengths of Existing Platform to Leverage

(Largely the same, but explicitly note that `SessionCategory.capacity` already supports N attendees).

6.2. Gaps to Address & Recommended Architecture Additions

(Largely the same, but emphasize Celery Beat for pending booking expiry, and in-platform notifications for admins).

7. Release & Iteration Plan

7.1. MVP Scope

- **Core Link Functionality:**
    - Admin configures settings for their single business booking link (`BusinessBookingSettings`: slug, approval, global buffer, pending expiry, active toggle).
    - Client page (Step 1) lists all `is_publicly_bookable` services for that business.
- **Availability & Booking:** *(Largely same, uses global buffer).*
- **Confirmations (Email Only for MVP):**
    - Includes flow for "pending approval" status and subsequent approved/rejected emails (with rebook link and optional reason).
    - Email for "booking request expired" (if implemented for MVP).
- **Admin Booking Management (Approval Workflow - CRITICAL FOR MVP):**
    - In-platform notifications (top nav icon) and dashboard visibility for new pending requests.
    - UI to Approve/Reject pending bookings.
    - Configurable expiry for pending requests (Celery Beat task to manage this).
- **Essential Technical Foundations (for MVP viability):**

    - Celery + Redis/Broker: For asynchronous email notifications and managing pending request expirations.
    - Basic Rate Limiting: For public booking endpoints.

7.2. Future Enhancements (Post-MVP)

(Largely same, WhatsApp notifications remain P1 post-MVP)

7.3. Out of Scope for MVP (but considered for future)

(Largely same, but "Manual booking approval workflow" is now IN MVP scope)

8. Open Questions & Assumptions

- **Assumption:** The "global buffer time" applies between any two distinct sessions, regardless of service.
- **Question (Dashboard Integration):** Specific UI design for how pending approval requests are displayed on the admin's dashboard for high visibility and quick action (e.g., a dedicated widget/card, count badges).
- **Question (Pending Expiry Action):** When a pending booking expires:
    - Should the client always be notified? (Recommendation: Yes, with a rebook link).
    - Should the admin be notified of auto-expirations? (Recommendation: Optional, perhaps a summary or log).
- **Question (Rebook Link Content):** The rebook link in rejection/expiry emails: should it pre-fill any information (like the originally desired service) or just lead to the main service selection page? (Recommendation: Lead to main service selection page for simplicity in MVP).
- **Question (Initial Slug Generation):** How should the initial `public_booking_slug` for a business be generated? From business name? Should it be editable by admin (with uniqueness checks)? (Recommendation: Auto-generate from sanitized business name, make editable with uniqueness validation).

9. Document History

Version	Date	Author(s)	Summary of Changes
...	...	...	...

2.2	May 28, 2025	AI (Enhanced with Detailed System Context)	Integrated specific Flowkey platform architecture, technology stack, data models, API structure, and development considerations based on prior context.
2.3	**May 28, 2025**	**AI (Incorporating User Clarifications)**	**Current Version.** Updated based on user feedback regarding: single persistent link per business, multi-service presentation on client page, existing N-attendee capacity, detailed booking approval workflow, and global buffer time. Refined user stories, features, data models, and API design accordingly.