Spitbraai Automation API Reference

Overview

This document provides detailed information about the automation system's internal APIs and service methods.

Core Services

EmailService

Methods

`sendTeamNotification(booking: Booking): Promise<void>`

Sends a notification email to the team when a new booking is received.

Parameters:
- `booking` : Booking object containing customer and event details

Email Content:
- Customer information
- Event details
- Package selection
- Next steps for the team

`sendCustomerThankYou(booking: Booking): Promise<void>`

Sends a thank you email to the customer after booking submission.

Template: `templates/thankYou.html`

`sendBookingFeeReminder(booking: Booking): Promise<void>`

Sends a reminder for the R500 booking fee payment.

Trigger: 3 days after booking if booking fee not paid
Template: `templates/reminder1.html`

`sendDepositReminder(booking: Booking): Promise<void>`

Sends a reminder for the 50% deposit payment.

Trigger: 14 days before event if deposit not paid
Template: `templates/reminder2.html`

`sendFinalPaymentReminder(booking: Booking): Promise<void>`

Sends a reminder for the final payment.

Trigger: 2 days before event if final payment not made
Template: `templates/reminder3.html`

CalendarService

Methods

`createEvent(booking: Booking): Promise<calendar_v3.Schema$Event>`

Creates a Google Calendar event for the spitbraai booking.

Event Details:
- Duration: 10 AM - 6 PM on event date
- Location: Venue address from booking
- Attendees: Customer and business email
- Reminders: 1 day and 1 hour before
- Color: Red (#11) for spitbraai events

`updateEvent(eventId: string, updates: Partial<Booking>): Promise<calendar_v3.Schema$Event>`

Updates an existing calendar event.

`deleteEvent(eventId: string): Promise<void>`

Deletes a calendar event.

PayFastService

Methods

`setupPaymentTracking(booking: Booking): Promise<PaymentLinks>`

Creates payment links for all payment stages.

Returns:

```
{
  booking_fee_link: string;
  deposit_link: string;
  final_payment_link: string;
}
```

`createPaymentLink(booking: Booking, paymentType: string, amount: number): Promise<string>`

Creates a PayFast payment link for a specific payment type.

Payment Types:
- `booking_fee` : R500 to secure the date
- `deposit` : 50% of total amount
- `final` : Remaining 50% of total amount

`verifyPayment(paymentData: any): Promise<boolean>`

Verifies a PayFast payment using their validation API.

`handleWebhook(webhookData: any): Promise<boolean>`

Processes PayFast webhook notifications for payment status updates.

ShoppingListService

Methods

`generateShoppingList(booking: Booking): Promise<ShoppingItem[]>`

Generates a comprehensive shopping list based on the menu package and guest count.

Categories:
- Meat (calculated per person with 10% buffer)
- Sides (salads, bread, vegetables)
- Condiments (sauces, seasonings)
- Equipment & Disposables (plates, cutlery, charcoal)

`formatShoppingListForEmail(shoppingList: ShoppingItem[]): string`

Formats the shopping list as HTML for email inclusion.

`exportShoppingListToPDF(shoppingList: ShoppingItem[], booking: Booking): string`

Exports the shopping list as a formatted text document (PDF generation can be added).

Data Types

Booking Interface

```
interface Booking {
  id: string;
  name: string;
  email: string;
  phone: string;
  event_date: string;
  guest_count: number;
  package_name: string;
  venue_name?: string;
  venue_address: string;
  total_amount: number;
  payment_status: 'pending' | 'deposit_paid' | 'fully_paid';
  booking_fee_paid: boolean;
  created_at: string;
  additional_notes?: string;
}
```

ShoppingItem Interface

```
interface ShoppingItem {
  category: string;
  item: string;
  quantity: string;
  unit: string;
  notes?: string;
}
```

Menu Packages

Essential Celebration

Target: Standard celebrations, 50+ guests
Meat per person:
- Beef (Silverside/Topside): 0.25kg
- Pork Shoulder: 0.15kg
- Chicken (Whole): 0.2kg
- Boerewors: 0.1kg

Premium Feast

Target: Premium events, special occasions
Meat per person:
- Beef (Ribeye/Sirloin): 0.3kg
- Lamb Leg: 0.2kg
- Pork Ribs: 0.25kg
- Chicken (Free Range): 0.25kg
- Boerewors (Premium): 0.15kg

Budget Braai

Target: Cost-conscious events
Meat per person:
- Beef (Chuck/Brisket): 0.2kg
- Chicken (Portions): 0.2kg
- Boerewors: 0.15kg

Automation Triggers

New Booking Processing

Trigger: New record in `bookings` table with `processed = false`
Actions:
1. Send team notification email
2. Send customer thank you email
3. Create Google Calendar event
4. Generate shopping list
5. Setup PayFast payment tracking
6. Mark booking as processed

Payment Reminders

Schedule: Daily at 9:00 AM

Booking Fee Reminder

Condition: `booking_fee_paid = false` AND `created_at` > 3 days ago
Action: Send booking fee reminder email

Deposit Reminder

Condition: `payment_status = 'pending'` AND `booking_fee_paid = true` AND `event_date` <= 14 days away
Action: Send deposit reminder email

Final Payment Reminder

Condition: `payment_status = 'deposit_paid'` AND `event_date` <= 2 days away
Action: Send final payment reminder email

Error Handling

Logging

All errors are logged to the `automation_logs` table with:
- `booking_id` : Associated booking
- `action` : What was being performed
- `status` : ‘error’
- `error_message` : Detailed error information
- `created_at` : Timestamp

Retry Logic

- Email sending: 3 retries with exponential backoff
- Calendar API: 2 retries with 1-second delay
- PayFast API: 3 retries with 2-second delay

Graceful Degradation

- If email fails, log error but continue processing
- If calendar creation fails, log error but continue
- If payment link generation fails, log error and notify team

Configuration

Environment Variables

See `.env.example` for all required configuration variables.

Cron Schedules

- New booking check: Every 5 minutes
- Payment reminder check: Daily at 9:00 AM

Rate Limits

- Gmail API: 250 quota units per user per 100 seconds
- Google Calendar API: 1,000,000 queries per day
- PayFast API: No published limits

Security

Data Protection

- All sensitive data encrypted in transit
- Environment variables never logged
- PayFast webhooks validated with signatures
- Supabase RLS policies enforced

Access Control

- Service account has minimal required permissions
- Gmail app password used instead of full OAuth
- PayFast sandbox mode for testing

Monitoring

Health Checks

Monitor these endpoints/metrics:
- Supabase connection status
- Gmail SMTP connection
- Google Calendar API availability
- PayFast API response times

Key Metrics

- Booking processing success rate
- Email delivery rate
- Calendar event creation success
- Payment reminder effectiveness
- Average processing time per booking

Testing

Unit Tests

```
npm test
```

Integration Tests

Test with sandbox/development environments:
- PayFast sandbox mode
- Test Gmail account
- Development Google Calendar
- Supabase test project

Manual Testing

1. Create test booking in Supabase
2. Verify all automation steps execute
3. Check email delivery
4. Confirm calendar event creation
5. Test payment link generation