Stripe Payment Flow in the Customer App
==========

This document reflects the Stripe integration that powers card payments during ride booking in the ZoomZap customer Flutter app.

1. Entry Points in the App

- Payment happens from the booking screen (`booking_screen.dart`). Riders choose between `cash` and `stripe` via radio buttons bound to `BookingController.paymentMethod`.
- The “Confirm Booking” button calls `BookingController.hitConfirmBookingApi()`, which orchestrates payment and final booking confirmation.

2. Amount Determination

- `BookingController` resolves the chargeable amount before contacting Stripe.
  * If a coupon or override populates `fairePrice`, that value is used.
  * Otherwise it reads the selected vehicle’s `totalPrice` from `SelectVehicleController`.
- Invalid or zero amounts trigger a toast and abort the flow.

3. Payment Intent Request

- When `paymentMethod == 'stripe'`, `hitConfirmBookingApi()` calls `_createPaymentIntentIfNeeded()`.
- `_createPaymentIntentIfNeeded()` posts to the backend endpoint `/payment/create-intent` via `APIRepository().createPaymentIntent(...)`.
  * Payload: `amount` (double) and `currency` (defaults to `usd`).
  * The mobile app expects a response shaped like:
    ```
    {
      "detail": {
        "client_secret": "...",
        "publishable_key": "...", // or publishableKey/pk
        ...
      },
      "message": "..."
    }
    ```
- The publishable key is mandatory. The controller updates `Stripe.publishableKey` at runtime using the value returned. Missing keys raise a toast and stop the flow.

4. PaymentSheet Initialization and Confirmation

- With a client secret in hand, `_createPaymentIntentIfNeeded()` calls:
  * `Stripe.instance.initPaymentSheet(...)` using the client secret and merchant display name `ZoomZap`.
  * `Stripe.instance.presentPaymentSheet()` to display Stripe's native UI for card selection, authentication, and confirmation.
- Stripe handles 3DS/SCA within the sheet. No manual card fields are shown in-app (legacy card UI is disabled).
- Success: execution resumes without error, meaning the Payment Intent is confirmed and automatically captured (default Stripe behavior).
- Failure scenarios:
  * User cancellation (`FailureCode.Canceled`) shows "Payment Cancelled" and throws to abort booking.
  * Other errors surface Stripe's localized message, toast "Payment failed", and rethrow.
- Any exception in this block stops `hitConfirmBookingApi()` from contacting the booking endpoint.

5. Booking Confirmation Call

- After a successful PaymentSheet flow, `_paymentClientSecret` remains populated.
- `hitConfirmBookingApi()` sends a FormData payload to `/ride/confirm` (`confirmEndPt`) with:
  * `Ride[car_type_id]`: selected vehicle ID.
  * `Ride[enable_disable_otp]`: OTP preference.
  * `PriceDetail[final_amount]`: same amount sent to Stripe.
  * `Payment[method]`: `stripe` or `cash`.
  * `Payment[client_secret]`: included only when Stripe succeeds, allowing the server to reconcile with the Stripe intent.
- The backend response (`MessageResponseModel`) drives final UI feedback and navigation (`Get.offAllNamed(AppRoutes.mainScreen)`).
- Any backend failure is surfaced via toast and the booking stays unconfirmed.

6. Backend Responsibilities (Expected)

- Maintain the customer's Stripe setup: create or reuse Stripe Customers and PaymentMethods as needed (the app does not manage saved cards directly).
- Create Payment Intents with automatic capture to match the client's expectation. The app does not support manual capture right now.
- Validate the client secret received during booking confirmation against Stripe (e.g., lookup the Payment Intent status, verify amount, rider, ride ID).

- Dispatch drivers, send receipts, trigger payout workflows, and listen to Stripe webhooks for reliability (`payment_intent.succeeded`, `payment_intent.payment_failed`, `charge.refunded`, etc.).

7. Error Handling and UX Notes

- Toasts communicate issues such as missing vehicle data, courier details, invalid amounts, missing publishable key, payment cancellation, or Stripe errors.
- If the rider toggles back to `cash`, `_createPaymentIntentIfNeeded()` short-circuits and no Stripe request is made.
- `_paymentClientSecret` is only sent when Stripe succeeds; cash bookings omit it.
- Keep the Stripe publishable key and webhook secrets off the client—only the publishable key is injected via API response.