SportHawk - Payments Lifecycle - Aug 2025

Aim: Payment / life-cycle needed in explicit and exhaustive detail (gFolder) of priority screens

See: Stripe Custom Checkout Components - https://stripe.com/gb/payments/elements

Movies in: Prototype Walkthroughs - Google Drive, flows:

Summary

1. Club/Team "Treasurer"s registers Stripe account
    a. FFC - two Stripe a/cs: David for Men's, Rew/Nicky for Women's and youth
2. Treasurer, raises a Payment Request to be paid by selected Members who get Notified
    a. Types: "Required", or "Optional"
    b. Cancel, sets the status to "Cancel" of the Payment Requests for Members who have not yet paid. People who have already paid are notified.
    c. Edit (e.g. correcting a mistake): Title, Description, Due by (future), Type, Members,, Amount
3. Notified Member sees list of requested payment, chooses individual (later bulk) payment, and (via Stripe) choose to pay by: card, Google Pay, or, Apple Pay
4. App, via Stripe, "knows" the completion status of the transactions (or not: cancel, fail), updates the Payment as "Paid" and notifies the Treasurer who raised it
5. Treasurer: receives the notifications Treasurer can send "reminder" notifications
6. Treasurer can view a list of payments that have been raised, choose to see detail of a single payment where they see who has/hasn't paid
7. Member can see payment history in profile/settings

Behind The Scenes of Member Payment To Team Bank A/c

Stripe for SportHawk
SportHawk (the App) has a Stripe account accessible via the Stripe API, info includes: account name,account number, public Stripe key, secret Stripe. This enables the App to use and manage Stripe transfers ("payments") from Member to Club/Team(s).

Stripe Connect To Team(s) Bank
Team Treasurer has created Stripe CONNECT a/c for a Team or Teams, which is a Stripe account connected to a bank account (to receive payments). SportHawk has recorded the Stripe a/c details, account number/id.

Onboard Team Stripe account as a usable destination by SportHawk Stripe Account
Onboard the Destination Account: SportHawk staff working with the Treasurer will connect the destination account (Team's) to the SportHawk Stripe. This process, known as "onboarding," links the Team account to SportHawk.

Team's Stripe Info in SportHawk
The Team Treasurer or Club Admins add a record to each SportHawk team which is the Stripe account id to accept payments for said Team. NB the same Stripe Connect account ID may be used for multiple Teams.

Who Can Manage Team's Settings
Configured Team(s) and Treasurer/Admin users are associated with the Stripe Connect a/c stored in SportHawk.

What is a "Payment" in the App?
A Payment Request in the App is a database row primarily associated with the Member (UUID), also referencing the Team (and through the Team references the Team's Stripe Account) and the user (Team Admin) who raised the Payment Request, the amount, status (requested) and other details such as due date for App management.

Member Makes Payment
For a specific Payment Request the Member sees the Payment Request details, particularly the amount, and chooses a supported payment method. The App asks Stripe (via SportHawk Stripe API) to present a payment form in a placeholder in the App screen, giving Stripe details:

- SportHawk Stripe API to use Stripe's functionality
- Stripe transaction type is "Destination Charge"
- Team's Stripe account number/id as the destination account
- the amount (and currency and tax info [i.e. not VAT registered])
- The Member's name

The user fills out his/her financial details in the Stripe form (likely unaware it is actually Stripe not the App) and clicks PAY NOW.

Monetary Transaction With Stripe From Member To

- Stripe takes the money from the Member's payment source
- Stripe charges SportHawk Stripe a/c the transaction fee
- Stripe puts the money in the Teams Stripe a/c

Worked example:

- Under the Stripe API with SportHawk credentials
- Amount is £25
- Stripe get £25 from the user
- Stripe charges its fee (e.g. £0.21) to SportHawk Stripe account
- Stripe takes/charges transaction fee of £0.21 from/to SportHawk Stripe a/c
- Stripe puts £25 in Team Stripe Connect a/c
- Treasurer uses Stripe App to see/move £25 to the Teams bank a/c

Treasurer Register

Manual setup, nothing in the App.

Ashton from SportHawk will assist a Team Treasurer in setting up a Stripe Connect a/c and then on-board that account as a usable destination in SportHawk's Strip account.

Ashton, using the Supabase Dashboard for the SportHawk BaaS project, updates the relevant Teams database rows with the Team Stripe Connect account number/id.

Treasurer Raises a Payment Request

Team Treasurer visits: Teams / Admins / '+' floating button on Figma 559-2966 / and on pop-up Figma 559-2927 taps the "Payment Request" card, to reach Figma 559-2744 "Create Request".

Code to be created in /app/payments/create-payment.tsx

Form fields:

- Title, req., text
- Description, opt. Multi-line text
- Due by, req., date / time input using pop-up OS date time controls

- Type, req., dropdown of "Required", "Optional"
- Members, req., card that goes to Figma 559-3204 (already implemented for Events in /app/events/edit-members.tsx)
- Stripe ID (read only), display only of Team's Stripe account number/id
- NOT NEEDED: Base Price and "Add Transaction Fee to Price"
- Amount, req., GBP monetary amount in pounds and pence

Top navigation to use Expo <Stack> component with custom arrow icon on the left (as per /app/events/create-event.tsx), screen title "Create Request", yellow action on right "Send".

Tapping "Send" will validate the form fields.

If fields are correct the Payment Request will be created.
First a row is inserted into the "payment_requests" table giving a Payment Request ID for the row.
For each Member that was marked as selected in the Select Member screen a row will be inserted in the "payments" table referencing the Payment Request ID and the User (UUID).

A Notification will be sent to each Member that was marked as selected in the Select Member screen, by inserting a row in the "notifications" table.

Member View Payments List

If there are no payments then as per Figma 559-3105,

If there are payments then as per Figma 559-3087, but default to All payments.
"Upcoming Payments" dropdown to work similarly to Teams/Events dropdown.
If the filter selects zero payments, but there are payments then show the body of Figma 559-3105 and how many payments there are overall as tappable text that when tapped will switch the filter to All.
Add a Filter option of "Required".

If the logged in user is an Admin for this Team, show a tab at the top left "Member", right is "Admin", default to "Member" view (as the Admin may also have payments to make) in a similar fashion to the "Recent" and "For you" "tabs" on the Home page. When the user selects "Admin" the body of the screen should be as per "Team Admin View Payments List" defined below.

Member View Detail & Payment

As per Figma 559-3055.
Top navigation menu with Expo <Stack> (as per technique in /app/events/create-event.tsx)

Member Profile Settings View Payment History

As per Figma 559-7147 for list of completed, failed, and cancelled payments.
Top navigation menu with Expo <Stack> (as per technique in /app/events/create-event.tsx)

As per Figma 559-7357 for payment details
Top navigation menu with Expo <Stack> (as per technique in /app/events/create-event.tsx)

Team Admin View Payments List

As per Figma 559-2776
Top navigation menu with Expo <Stack> (as per technique in /app/events/create-event.tsx)

Team Admin Payment Detail, Edit & Cancel

As per Figma 559-2792
Top navigation menu with Expo <Stack> (as per technique in /app/events/create-event.tsx)

From the dot/hamburger menu top right …
For "Send reminder", send a notification to those who haven't paid.

For "Edit Request" new screen as per 559-2709.

Edit (e.g. correcting a mistake).
These fields can be freely edited: Title, Description, Due by (future), Type.

The User cannot amend Members of Amount if the Due by date/time has already passed, they must adjust the Due by first.

Members can be edited with a Select Member like screen, except that it should also indicate if a Member has already paid.

On tapping Save there will be a pop-up alert for confirmation that additionally warns the user if they are removing Members who have already paid, or if they are changing the Amount and some Members have already paid.

Members who have not yet paid are sent a notification that the Payment details have changed.

Members who have paid receive a notification if they are removed from the Payment request, or the Amount has been changed.

The database tables are updated appropriately.

If the user taps Cancel there will be a pop-up alert for confirmation that additionally warns the user there are Members who have already paid.

On confirmation of Cancel all Members on the Payment Request are sent a notification that payment has been cancelled.

Known Limitations And Future Possibilities Re Payments

There are some edge cases that we are not yet handling:

- A Member who has paid is removed or the payment is cancelled - there is no current mechanism to refund them
- The Payment Request has the Amount edited leaving Members who have paid due a partial refund or owing more - not yet handled
- Changing the Team Stripe ID across a refund as there would be no Stripe continuity across the before and after accounts.
- Pre-payment for bad payers, or a club rules or guidelines
- Change of Team Admins - currently done by Aston manually in Supabase Dashboard

Info From Gemini: On Stripe Account and "Destination Charge"

Google Gemini Chat (AD a/c)

Yes, this configuration is absolutely possible using **Stripe Connect**.

Your business would act as the "platform," facilitating a payment from your app user to the destination account. The specific method to achieve your goal is called a **Destination Charge**.

How Destination Charges Work

With a destination charge, the payment flow is designed so your platform account handles the transaction and pays the fees.

1. **Payment Initiated:** Your app user decides to pay an amount (e.g., £100).
2. **Charge on Your Account:** Stripe creates the £100 charge on **your business's Stripe account** (the platform).
3. **Fees Deducted:** Stripe deducts its transaction fees (e.g., £1.70 for a standard UK card transaction) directly from **your account balance**.
4. **Funds Transferred:** The full, original amount (£100) is then automatically transferred from your account to the destination Stripe account.

The end result is exactly what you want:

- The **app user** pays £100.
- The **destination account** receives the full £100.
- **Your business account** pays the £1.70 Stripe fee.

What You Need to Do

To set this up, you'll need to use the Stripe Connect product. Here are the basic steps:

1. **Set up Stripe Connect:** Activate Stripe Connect on your existing UK Stripe account, choosing a platform model.
2. **Onboard the Destination Account:** You must connect the destination account to your platform. This process, known as "onboarding," links their account to yours. Depending on your needs, you can use Express or Custom accounts for this.
3. **Create the Payment:** When your app user makes a payment, your backend code will create a Stripe `PaymentIntent`. In the API call, you will specify the ID of the destination account in the `transfer_data[destination]` parameter. This tells Stripe to automatically transfer the funds after a successful charge on your platform.

This is a standard and well-supported use case for platforms and marketplaces built with Stripe.

Avoiding PCI-DSS Compliance Requirements

You should call the Stripe API to surface a Stripe-hosted UI component within your app. **Do not build your own form to collect financial information.**

Building your own form to collect raw credit card details (like the card number, CVC, and expiry date) would bring your app directly into the scope of **PCI DSS (Payment Card Industry Data Security Standard)** compliance. This is a complex, time-consuming, and expensive security standard to meet and maintain.

Stripe provides official libraries specifically for React Native that make this process secure and simple.

The Recommended Approach: Use Stripe's Payment Sheet

The best practice is to use Stripe's pre-built, native UI components. For most React Native apps, the **Payment Sheet** is the ideal solution.

- **What it is**: The Payment Sheet is a pre-built payment form, provided by the official `@stripe/stripe-react-native` library, that you present to the user within your app. It handles everything from entering card details to authenticating the payment with 3D Secure.
- **How it works**: Your server creates a `PaymentIntent` with Stripe, which is a record of the transaction. You pass a secret key from this `PaymentIntent` to your React Native app. Your app then uses this key to initialize and present the Payment Sheet to the user. The user enters their details directly into Stripe's secure UI, and your app never touches the sensitive data.
- **Key Benefits**:
  - **Drastically Reduced PCI Scope**: Since sensitive card data is sent directly from the user's device to Stripe's servers, your server and app avoid handling it, simplifying your PCI compliance obligations.
  - **Optimized for Conversion**: The form is designed for a smooth mobile experience and includes features like saved card details, Apple Pay, and Google Pay, which can significantly improve payment success rates.
  - **Global Payment Methods**: It automatically shows the most relevant payment methods to the user based on their location and device (e.g., iDEAL in the Netherlands, Bancontact in Belgium).
  - **Built-in Security & Authentication**: It automatically handles Strong Customer Authentication (SCA) requirements like 3D Secure 2 without you needing to build the complex logic yourself.