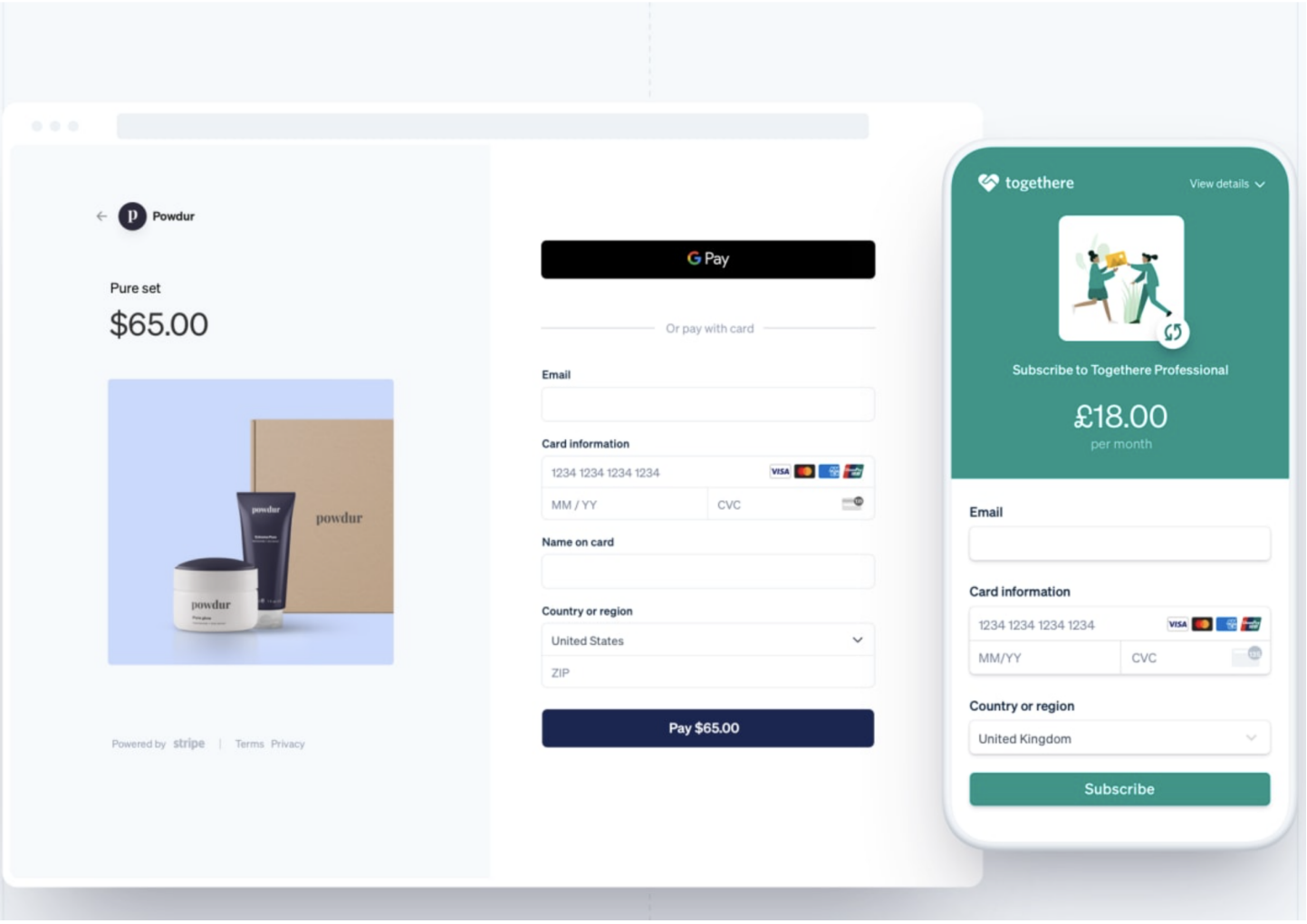
#systemDesign #programming

Introduction to Payment Systems



What we will not talk about

- Storing credit card information
- Fraud checks
- Connect to bank accounts
- Charge credit cards
- Let Stripe / Paypal handle the actual payment

What we will talk about

- Why use a third party Payment Service?
- How our app integrates with a Payment Service (PayPal, Stripe, etc)

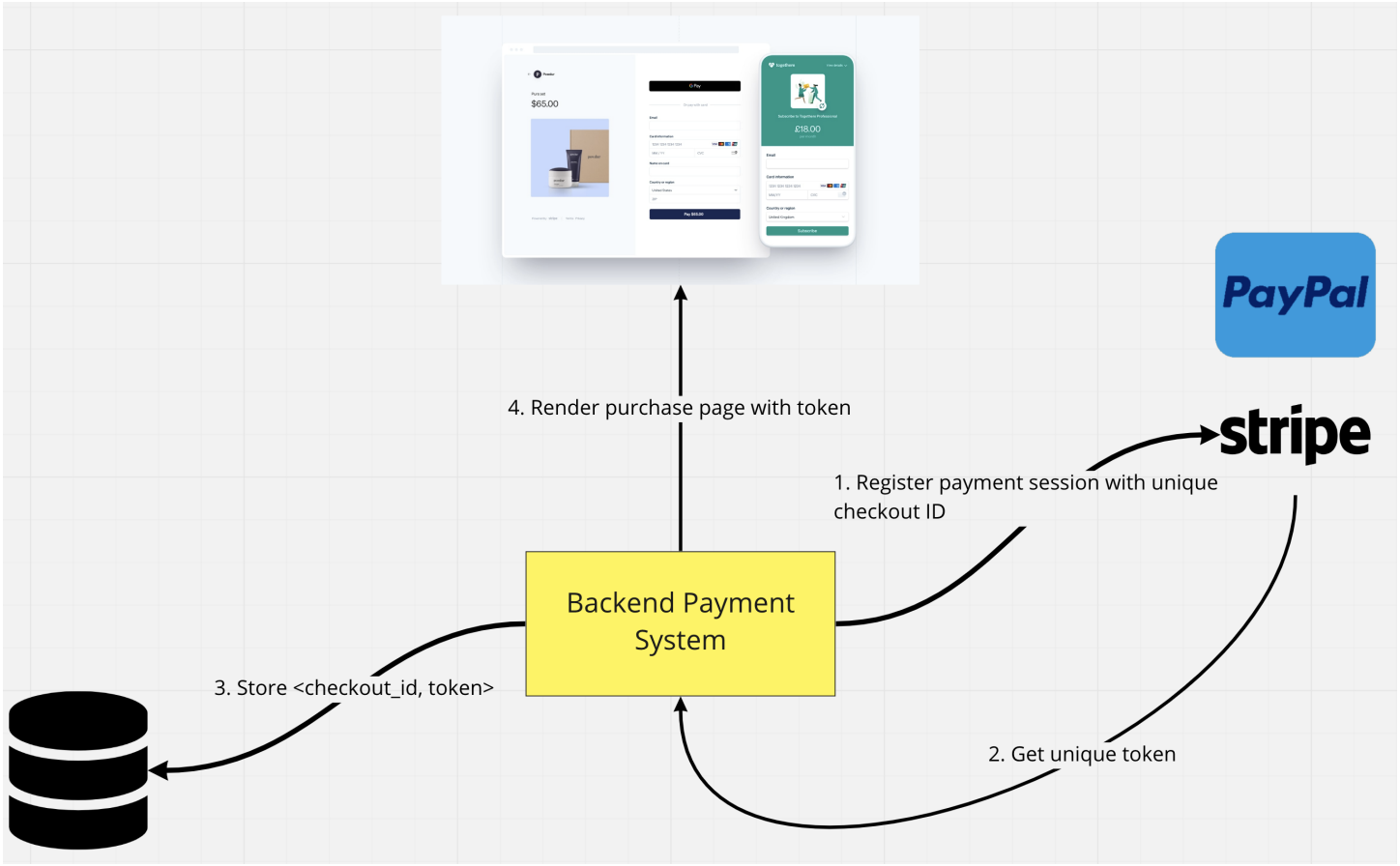
- What kind of database to use?
- What data types to use for currency?
- Synchronous vs Asynchronous system
- How to avoid double charge issue?

Why use a Payment Service?

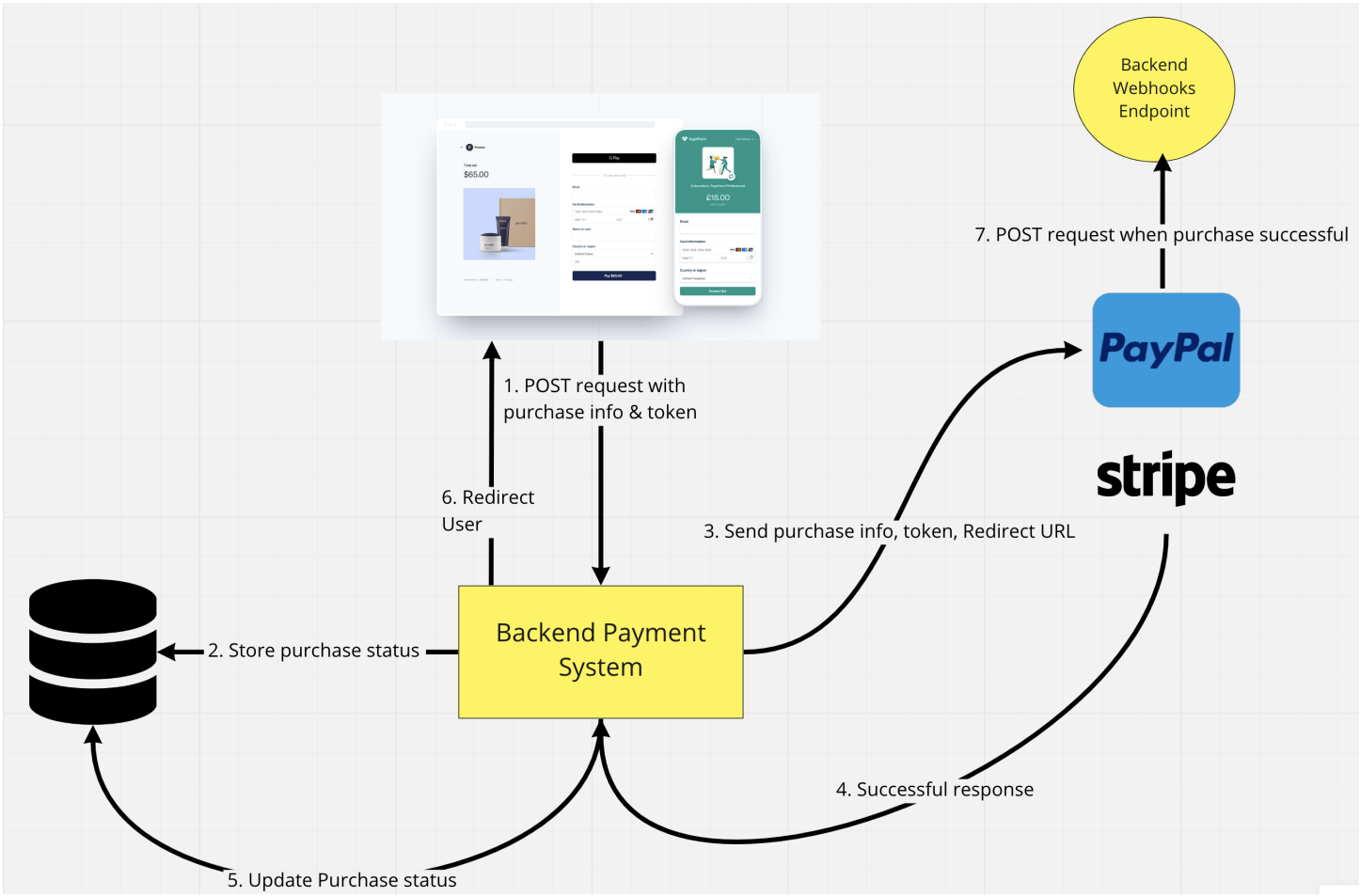
- Sensitive credit card information
- Lots of data privacy laws
- Complex credit card systems
- Lots of integrations: VISA, Discover, Mastercard, AMEX, etc
- Focus on core logic rather than reinventing the wheel

How our app integrates?

What happens when user is about to checkout?



What happens when user clicks "Buy Now" or "Purchase"?



What does the database look like?

checkout_id	psp_token	amount	currency	status
1	abc123	"56.00"	USD	PENDING
2	eee909	12.00	CAD	COMPLETED
3	gf98d	10.00	USD	FAILED

A few things to note:

- checkout_id has unique constraint
- psp_token generated by PayPal, Stripe
- Store amount as a string
    - Only convert to number when doing calculations
    - Not safe to store amount as number
    - Not safe to transmit amount as number
- Webhooks endpoint updates status to COMPLETE

Why use Relational Database?

- For most applications your QPS will be relatively low for purchase
- Transactional guarantees
- Consistency
- Replication lag?
    - Read / Write from master for critical paths

Microservice Communications

- We kept things simple with one backend
- Realistically you might have multiple microservices in the purchase flow
- Synchronous communication
    - Relies on synchronous HTTP requests
    - Pro
        - Easy to setup
    - Cons
        - Any component in critical part can fail
        - Timeouts
        - Very brittle
        - Snowball effect
        - Prone to inconsistencies
- Asynchronous communication
    - Relies on some messaging system like Kafka
    - Pros
        - Better failure handling
        - Retries
        - Decoupled
    - Cons
        - Difficult to setup
        - Difficult to debug

Double Spend

Every purchase page will have a uniquely generated `checkout_id` and a corresponding `token` generated by the PSP

- How does it occur?
  - Network errors
    - Between frontend and backend
    - Between backend and PSP
  - Client page not updated
- What happens when the user clicks "Buy Now" twice?
  - Always send PSP generated token to PSP for every request
  - PSP "recognizes" token
    - Treats the request as a "retry"
    - If purchase was successful, request thrown away