Payment System

We design a payment system. E-commerce & other services online have exploded in popularity across the world. What makes every transaction possible is a payment-system running behind the scenes. A reliable, scalable, and flexible payment system is essential.

What is a payment system? According to Wikipedia, “A payment system is any system used to settle financial transactions through the transfer of monetary value. This includes the institutions, instruments, people, rules, procedures, standards, and technologies that make its exchange possible”.

A payment gateway is a service that processes online payments. It enables merchants to accept credit cards and other forms of electronic payments securely over the Internet. Popular payment gateways include PayPal, Stripe, and Square.

The payment system is easy to understand on the surface but is also intimidating for many developers to work on. A small slip could potentially cause significant revenue loss and destroy credibility among users.

# Key Pointers:

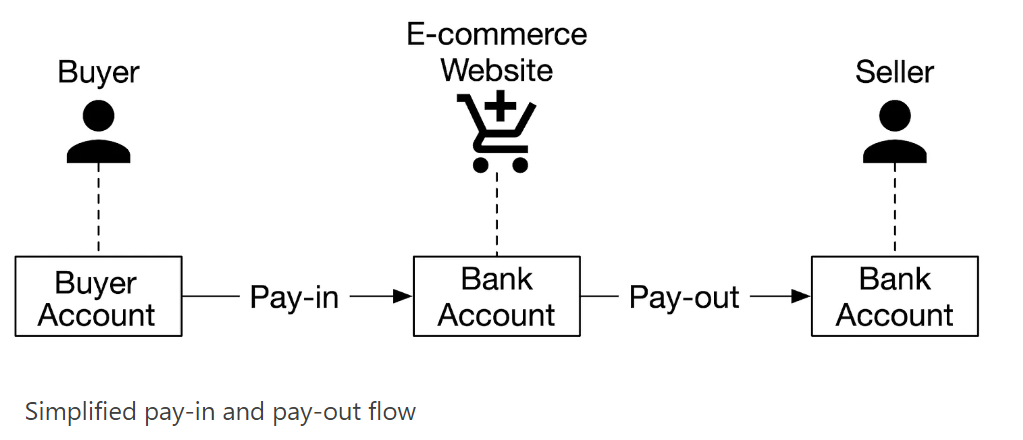
* Assume you are building a payment backend for an e-commerce application like Amazon.com. When a customer places an order on Amazon.com, the payment system handles everything related to money movement.
* The payment system should support all of these options in real life. Like Cards or Alternate Payment Methods (APM).
* For Card Processing, we generally tend to use third-party payment processors. Due to extremely high security and compliance requirements, we do not store card numbers directly in our system. We rely on third-party payment processors to handle sensitive credit card data. Storing CC at your end needs Payment Card Industry Data Security Standard (PCI DSS).
* For Alternate Payment Methods (APM) processing, we integrate with the corresponding 3rd party provider. We need to understand end-to-end integration with each of the 3rd party. And build a secure, stable integration and enable the end customer to make payment via that 3rd party provider.
* Load of around 1 million transactions per day. 10-15 transactions per second.
* PaymentMethod: APM, Card
* Payment Service Provider (PSP): Trustly, Paypal, Stripe, etc
* PaymentType: SALE (One time payment), REFUND, RECC (Subscriptions)
* Hosted Page - the Payment page provided by the PSP. Ecom application redirects the customer to the hosted page, where they complete their transactions. And would return back to the ecom website.
* Reliability and fault tolerance. Failed payments need to be carefully handled.

## Functional requirements

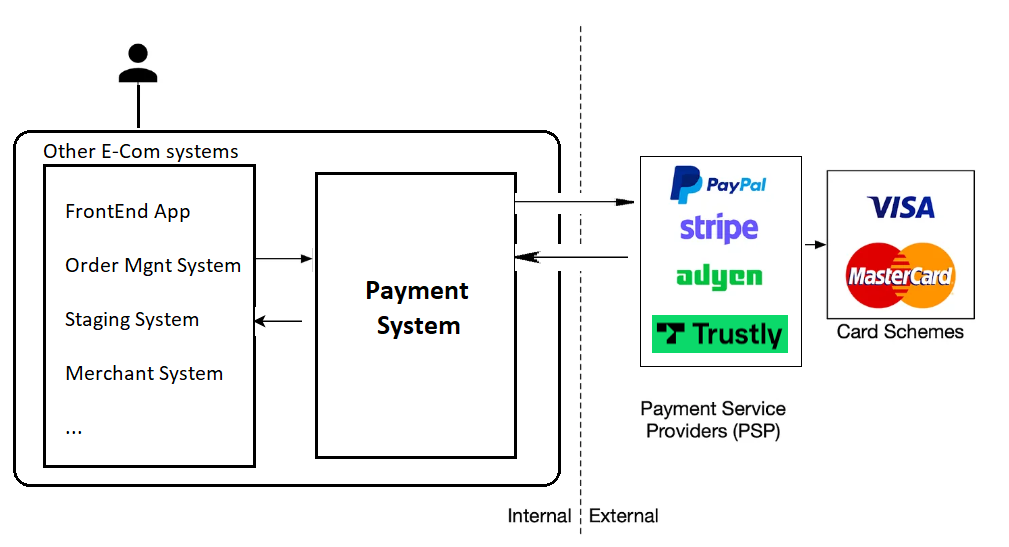
* To build a Pay-in / Deposit / Sale / Checkout flow: Money moves from the end customer account to the payment system account (on behalf of seller)

Overview:

* Pay-in flow: payment system receives money from customers on behalf of sellers.
* Pay-out flow: payment system sends money to sellers around the world.



## Pay-in / Deposit / Sale / Checkout flow (Payment System Overview)



## 

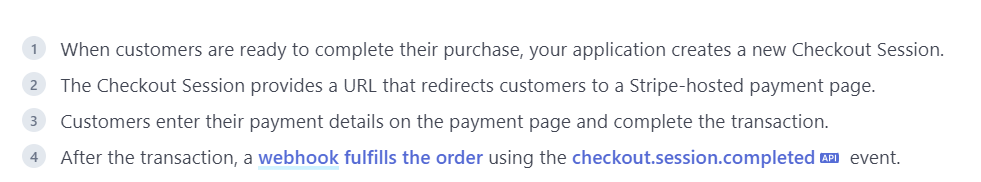
PSP integration

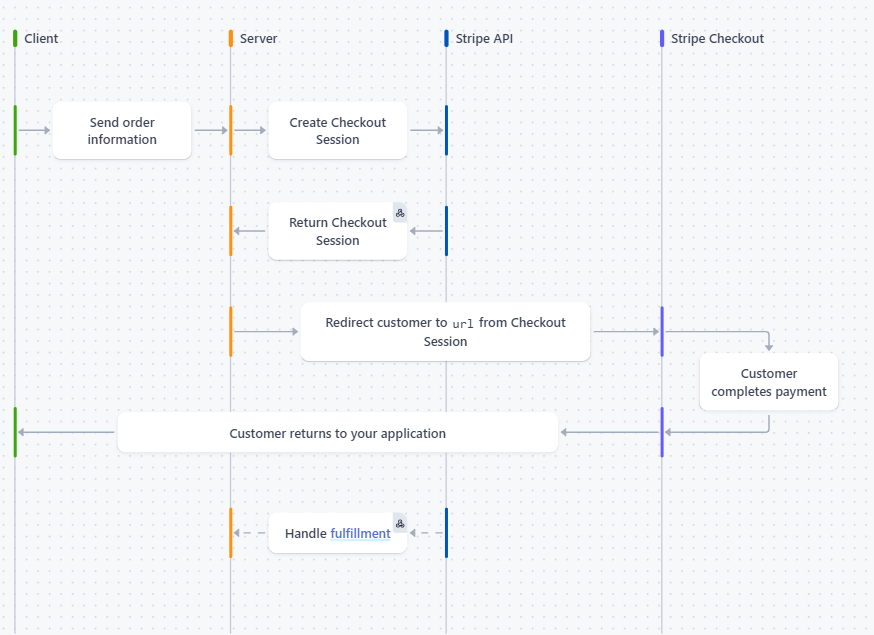
We are integrating with Stripe PSP. Stripe is a leading payment service provider renowned for its seamless integration and robust infrastructure, offering businesses around the globe a versatile platform to accept online payments securely and efficiently. With its user-friendly interface and extensive range of features, Stripe empowers merchants of all sizes to effortlessly manage transactions, handle subscriptions, and expand their customer base, thereby revolutionizing the e-commerce landscape. Trusted by millions of businesses worldwide, Stripe continues to set the standard for payment processing solutions, driving innovation and simplifying the complexities of online commerce..

**Stripe Payment Integration:**

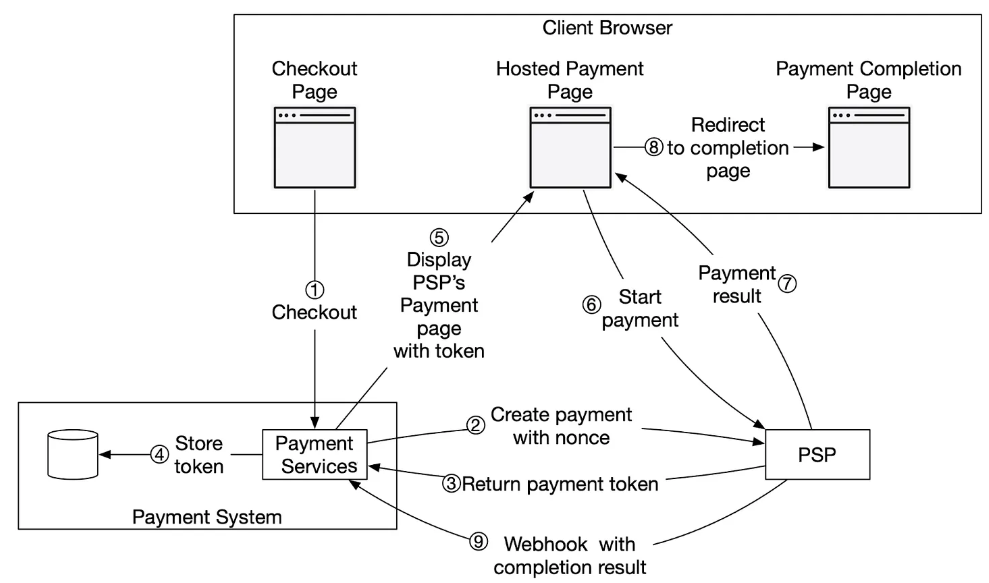
* <https://docs.stripe.com/payments/checkout/how-checkout-works>
* <https://docs.stripe.com/api>
* <https://docs.stripe.com/api/checkout/sessions>

Stripe Checkout Flow:





End-to-end payment Flow



1. The user clicks the “checkout” button in the client browser. The client calls the payment service with the payment order information.
2. After receiving the payment order information, the payment service sends a payment registration request to the PSP. This registration request contains payment information, such as the amount, currency, etc, and the redirect URL. Because a payment order should be registered only once, there is a unique field(id of payment order) to ensure the exactly-once registration.
3. The PSP returns a token back to the payment service. A token is an id on the PSP side that uniquely identifies the payment registration. We can examine the payment registration and the payment execution status later using this token.
4. The payment service stores the token in the database before calling the PSP-hosted payment page.
5. Once the token is persisted, the client displays a PSP-hosted payment page.
6. The user fills in the payment details on the PSP’s web page, logs in to the bank, and authorizes the payment, by clicking the pay button. The PSP starts the payment processing.
7. The PSP returns the payment status.
8. The web page is now redirected to the redirect URL. The payment status that is received in step 7 is typically appended to the URL.
9. Asynchronously, the PSP calls the payment service with the payment status via a webhook. The webhook is an URL on the payment system side that was registered with the PSP during the initial setup with the PSP. When the payment system receives payment events through the webhook, it extracts the payment status and updates the *payment\_order\_status* field in the Payment Order database table.