Ask a Question

# Payment Service and Fraud Detection in Uber Design

Learn how the payment service of Uber works and how they deal with payment fraud.

**We'll cover the following**

Our goal in this lesson is to highlight the need to secure the system against malicious activities. We'll discuss fraudulent activities in the context of payment. We'll first briefly go through how the payment system works, and then we'll cover how to guard our system against malicious activity.

The payment service deals with the payments side of the Uber application. It includes collecting payment from the rider to give it to the driver. As the trip completes, it collects the trip information, calculates the cost of the journey, and initiates the payment process.

## Major functionality

The Uber payments platform includes the following functionality:

- **New payment options**: It adds new payment options for users.
- **Authorization**: It authorizes the payments for transactions.
- **Refund**: It refunds a payment that was authorized before.
- **Charging:** It moves money from a user account to Uber.

## What to prevent

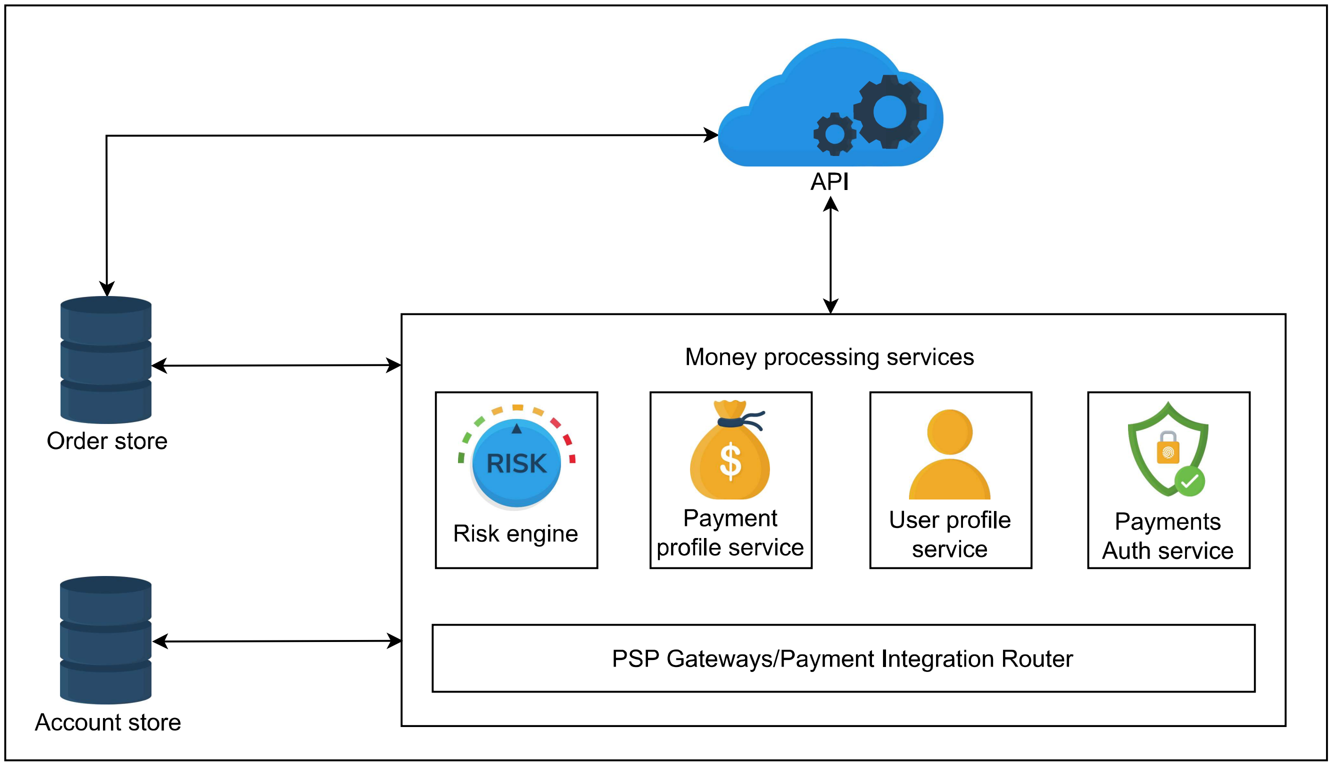
We need to prevent the following scenarios for the success of the payment system:

- Lack of payment
- Duplicate payments
- Incorrect payment
- Incorrect currency conversion
- Dangling authorization

The Uber payment platform is based on the double-entry bookkeeping method.

## Design

Let's look at the design of the payment platform in Uber and learn how different components work together to complete the transactions:



The payment platform in Uber

The key components used in the payment service are as follows:

- **API**: The Uber API is used to access the payment service.
- **Order store**: This stores all the orders. Orders collect payments and contain information regarding money flow between different riders and drivers.
- **Account store**: This stores all the accounts of riders and drivers.
- **Risk engine**: The risk engine analyzes different risks involved in collecting payment from a particular rider. It checks the history of the rider—for example, the rating of the rider, if the rider has sufficient funds in their rider account, any pending dues, if the rider cancels the rides frequently, and so on.
- **Payment profile service**: This provides information on the payment mechanisms, such as credit and debit cards.
- **User profile service**: This provides information regarding users' payments.
- **Payment authorization service**: This offers payment authentication servic
- **PSP gateways**: This connects with payment service providers

## Workflow

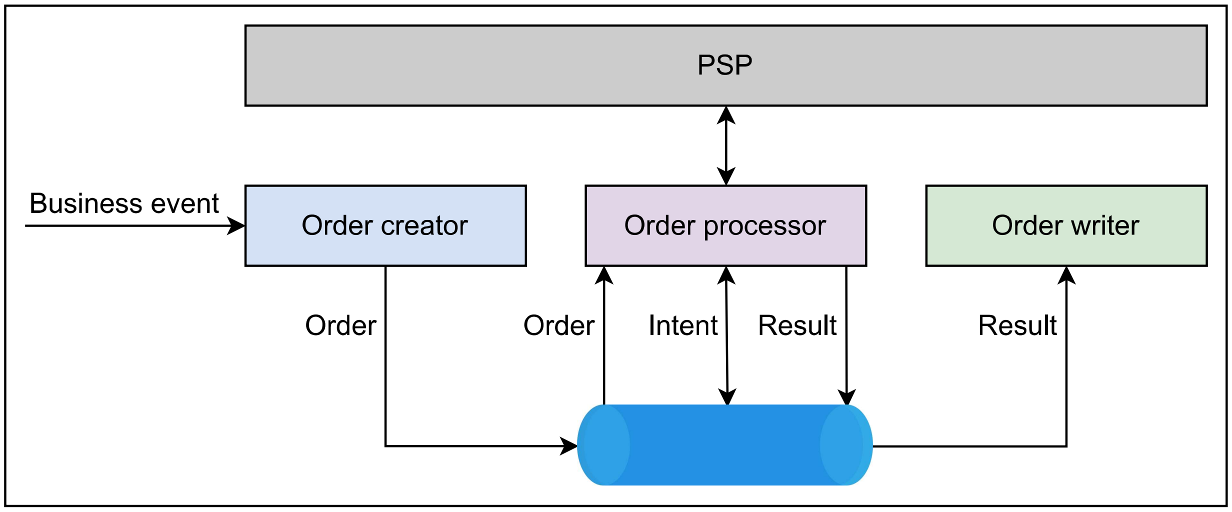
The following steps show the workflow of the payment service:

1. When a rider requests a ride, the Uber application uses the Uber API to request the risk engine to check the risks involved.
2. The risk engine obtains user information from the user profile service and evaluates the risk involved.
3. If the risks are high, the rider's request isn't entertained.
4. If the risks are low, the risk engine creates an authorization token and sends it to the payment profile service (for record-keeping), which fetches that token and sends it to the payment authorization service.
5. The payment authorization service sends that request to the PSP gateway, which contacts the service provider for authorization.
6. The PSP gateway sends the authorization token back to the payment authorization service, which sends the token back to the Uber application that the trip request is approved.
7. After the trip is completed, the Uber application uses the API to send the payment request to the PSP gateway with authorization data.
8. The PSP gateway contacts the service provider and sends the status back to the Uber application through the API.

The Uber payment service is constructed with a set of microservices grouped together as a stream-processing architecture.

## Apache Kafka

**Kafka** is an open-source stream-processing software platform. It's the primary technology used in payment services. Let's see how Kafka helps to process an order:



Payment service with Kafka

The order creator gets a business event—for example, the trip is finished. The order creator creates the money movement information and the metadata. This order creator publishes that information to Kafka. Kafka processes that order and sends it to the order processor. The order processor then takes that information from Kafka, processes it, and sends it as intent to Kafka. From that, the order processor again processes and contacts the PSP. The order processor then takes the answer from PSP and transmits it to Kafka as a result. The result is then saved by the order writer.

The key capabilities of Kafka that the payment service uses are the following:

- It works like message queues to publish and subscribe to message queues.
- It stores the records in a way that is fault tolerant.
- It processes the payment records asynchronously.

## Fraud detection

Fraud detection is a critical operational component of our design. Payment fraud losses are calculated as a percentage of gross amounts. Despite fraud activities accounting for a tiny portion of gross bookings, these losses considerably influence earnings. Moreover, if malicious behavior isn't promptly detected and handled, it may be further leveraged, which results in significant losses for the business.

Earlier, we discussed the risk engine that detects fraud before the trip starts. Here, we'll focus on the fraud detection that happens during trips or at the end of the trips.

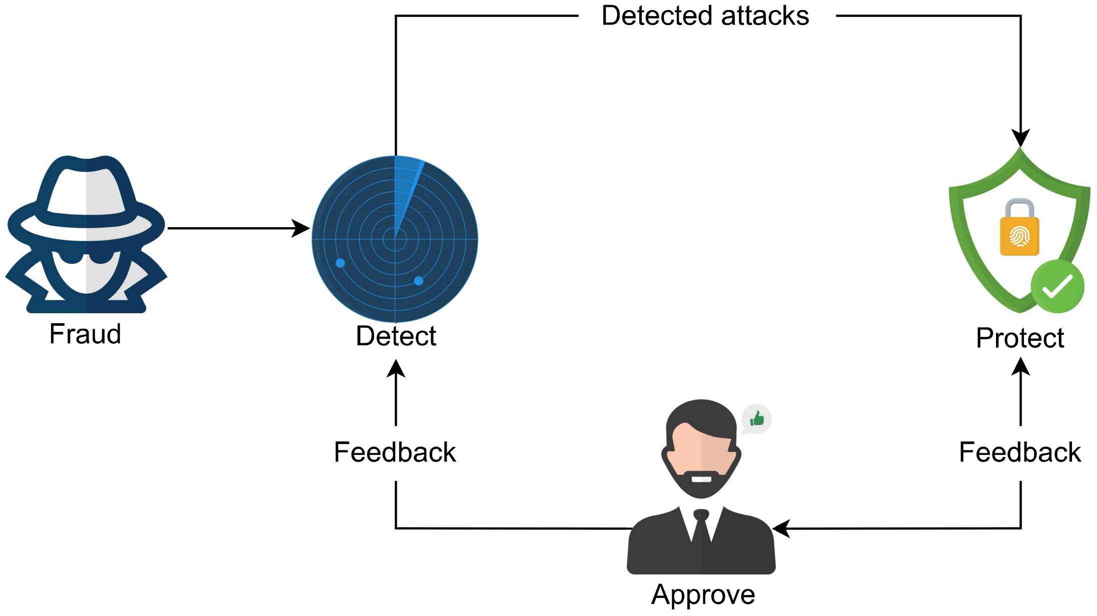
Fraud detection is challenging because many instances of fraud are like detecting zero-day security bugs. Therefore, our system needs intelligence to detect anomalies, and it also needs accountability for automated decisions in the form of

The activities that are considered fraudulent by Uber are as follows:

- A driver deliberately increases the time or distance of a trip in a dishonest manner—for example, if they take a much longer route than necessary to the rider's destination.
- A driver manipulates the GPS data or uses fake location GPS apps.
- A driver confirms trips with no intention of completing them, forcing riders to cancel.
- A driver creates false driver or rider accounts for fraudulent purposes.
- A driver provides incorrect or inaccurate information about themselves when opening their account.
- A driver drives a vehicle that hasn't been approved by Uber.
- A driver claims fraudulent fees or charges, such as unwarranted cleaning fees.
- A driver willfully confirms or completes fraudulent trips.

## RADAR

Uber introduced **RADAR**, a human-assisted AI fraud detection and mitigation solution to handle the discussed scenarios of fraud. The RADAR system detects fraud by analyzing the activity time series of the payment system. It then generates a rule for it and stops it from further processing. This proactive approach can hel to detect unseen fraud in real time. This detection model also uses human knowledge for continuous improvements. Here, we'll briefly discuss how RADAR works:



The RADAR fraud detection solution

RADAR recognizes the beginning of a fraud attempt and creates a rule to prevent it. The fraud analyst is involved in the next step. They review the rule and approve or reject it if required. They then send feedback (approved or not approved) to the protection system. The feedback is also sent to the fraud detection system by the fraud analysts to improve detection.

Furthermore, the system examines the data in the following time dimensions:

- It examines the trip time when a trip has been completed. Multiple factors are involved in real-time trip monitoring. For instance, the system can check whether or not the driver and rider locations are the same. The system can also monitor the speed and analyze the real traffic to check if the driver is intentionally driving slow or if there's traffic congestion.