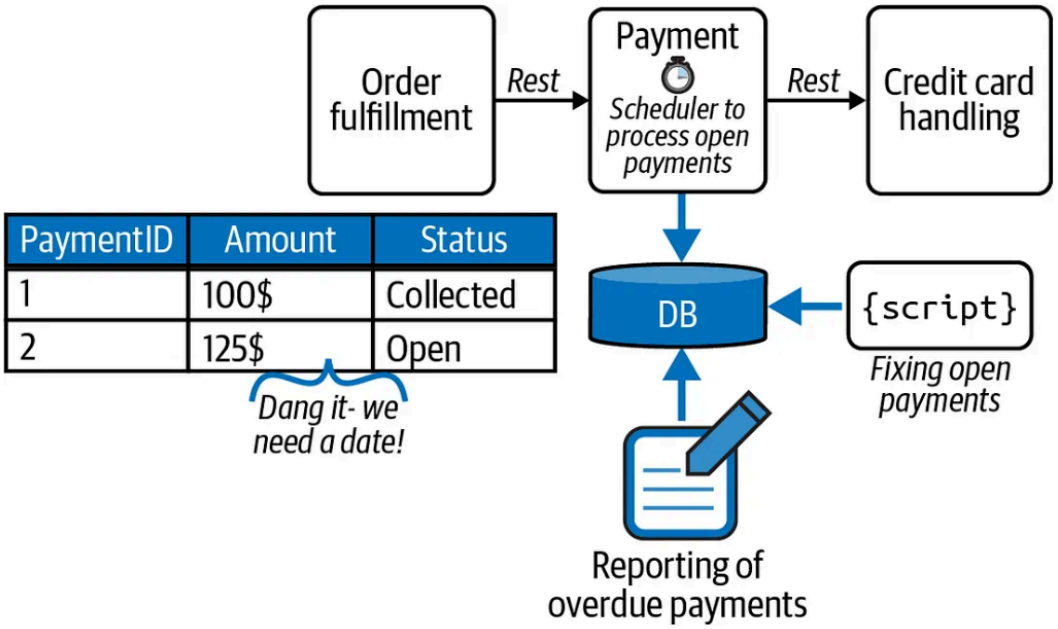
Notice: This story is taken from the book “Practical Process Automation by Bernd”

- Task assigned – Ash is asked to build a small backend to process credit card payments.
- Plan agreed – Ash and the order fulfillment team decide the easiest way is for Ash to provide them with a REST API.

- Warning received – A colleague warns Ash that the credit card service is unreliable.
- Quick fix – Ash adds simple retry logic and hopes the order fulfillment team can handle the rest.
- Problems in production – After launch, the order fulfillment team gets many “service unavailable” errors. Orders fail, the CEO complains, and the team doesn’t want to manually retry payments.
- Bigger fix – Ash creates a database to track payment status and adds a scheduler to retry failed payments asynchronously. The order fulfillment team agrees to use the updated process.



- New failure – A single bad payment crashes the scheduler. Payments pile up, orders can’t ship, and the order fulfillment team is stuck again.

- Patch and monitoring – Ash fixes the issue, adds alerts, and promises to monitor the system closely.

- Extra demands – Just before Ash’s vacation, the boss stops them. Only Ash understands the system, and now management wants detailed reports, SLA monitoring, and more reliability.

- End result – A simple payment API for the order fulfillment team has turned into a complex, fragile system requiring constant care.