**Chapter 10**

Design a Notification System

*Notification types*

*iOS push notification*

* 3 components required to send an iOS push notification
  + *Provider* – builds and sends notification requests to Apple Push Notification Service (APNS). Provides:
    - Device token – unique identifier used to send push notifications
    - Payload – a JSON dictionary containing a notification’s payload
  + *APNS* – a remote service provided by Apple to propagate push notifications to iOS devices
  + *iOS* – the end client receiving the push notifications (e.g. iPhone)

*Android push notification*

* Same flow as iOS – Firebase Cloud Messaging (FCM) instead of APN

*SMS message*

* Third-party SMS services (e.g. Twilio, Nexmo)

*Email*

* Can setup own email servers or opt for commercial services
  + Sendgrid, Mailchimp – popular third-party services

*Basic high-level design*

A diagram of a system

Description automatically generated

*Components*

* *Services* – can be a micro-service, cron job, or distributed system that triggers notification sending events
* *Notification system* – the centerpiece of sending/receiving notifications
  + Provides APIs for services
  + Builds notification payloads for third-party services
* *Third-party services*
  + Extensibility is important – system should easily allow third-party services to be plugged or unplugged
  + Used in case when certain services are not available in certain markets

*Problems in this design*

* *Single point of failure* – only one server handles all notifications
* *Hard to scale* – challenging to scale databases, caches, and different notification components independently
* *Performance bottleneck* – constructing HTML pages and waiting for responses from third-party could take time
  + Handling everything in one system/server can result in the system overload, especially during peak hours

*Revised High Design*

*A diagram of email communication

Description automatically generated*

*Solutions*

* Move database and cache tier out of the notification server
* Added more notification servers and set up automatic horizontal scaling
* Introduce message queues to decouple the system component

*Notification servers*

* Provide APIs for services to send notifications
* Provide basic validations for emails, phone numbers, etc.
* Query the database or cache to fetch data
* Put notification data to message queues for parallel processing

*Cache*

* User, device info and notification templates are cached

*Message queues*

* Remove dependencies between components
* Serve as buffers to handle high volumes of notifications
* Each notification type has its own message queue
  + Outage on one notification will not affect the rest

*Workers*

* List of servers that pull notification events from message queues
* Send them to the corresponding third-party services

***Deep Dive Considerations***

*Reliability*

*Prevent data loss*

* Notification system persists notification data in a database and implements a retry mechanism

*Prevent duplicate notifications*

* Dedupe mechanism to reduce duplication occurrence
* Check whether an event has been seen before via event ID. If seen before, it is discarded

*Additional Components*

*Notification template*

* Introduced to avoid building every notification from scratch
* A preformatted notification with customizable parameters, styling and tracking links, etc.
* Helps to maintain consistent format, reduce margin errors, and saves time

*Notification setting*

* Users can opt-in and out of different notifications
* Information is stored in a notification setting table

*Rate limiting*

* Limit the number of notifications a user can receive

*Retry mechanism*

* When a third-party service fails to send a notification, it will be added to message queue to retry
* If problems persist, an alert is sent to the developers

*Security*

* For iOS and Android apps, *appKey* and *appSecret* are used to secure push notification APIs

*Monitor queued notifications*

* Key metric is the number of queued notifications at any given time
* Too much traffic would indicate that more workers need to be added

*Event tracking*

* Notification metrics such as open rate, click rate and engagement are important to understand customer behaviors

*Deep dive diagram*

A diagram of a software system

Description automatically generated