**Notifications Service**

This is never a standalone system but is embedded in some designs like E-Commerce Website and other applications. This service will be used to notify our users.

Functional Requirements

1. Send Notifications.
2. Pluggable (SMS, Email are already there but we want to introduce App notification or whatsapp notification and our system should be easily extensible to add a new type of notification)
3. SaaS (Who is sending what number of notifications and we should be able to rate limit since this service can be given to other companies as a product and also internally we must know which service is sending how many notifications to the users – A particular user should not get X promotional notifications in a day. Transactional Notifications is fine.)
4. Prioritization: Some messages are more priority, and some are not. OTP is more priority while promotions are low priority.

Non-Functional Requirements

1. High Availability
2. Many Clients

**System Design**

Diagram

Description automatically generatedA picture containing text, whiteboard

Description automatically generated

1. Client 1, 2 .. N wants to send out a notification. They will talk with the Notification Service. There a few kind of requests that can be made:
   1. Send a “email/sms/app” type of notification with user having ID emailID/number/phonetype with content “...”. This type of requests will be made by the companies whom we will give our product as a SaaS.
   2. Send a notification to this User ID with content “…”. This type of requests would be made by our own company where different services will use our notification service.
2. Notification Service: This will take the request and put it into Kafka and respond to the client saying the request is taken and it will be sent in a couple of seconds at max. Keeping a synchronous flow would keep our client waiting for some time so its better to keep it async. This service also does a basic verification of the fields (e.g.: User is not null, phone number is valid, etc.)
3. Notification Validator and Prioritizer: Based on some attribute within the message, this service would assign a priority to the messages (OTP would have more priority than a normal promotional notification). It will put these messages on a Kafka topic specific to each priority. Consumers are going to consume high priority messages first and then the low priority messages.
4. Rate Limiter:
   1. It does two kinds of rate limiting.
      1. Is the client allowed to call me these many times?
      2. The user whom I am going to send this notification, am I allowed to send him this notification, these many times?
   2. Both these rate limiting will be implemented in a similar way. It has a REDIS where there would be key having a userID, ClientId and we increment it for a certain timestamp. The moment we exceed a threshold, we drop that request.
   3. It also does request counting as well for the “client companies” who pay us X for each request.
5. Notification Handler & User Preferences:
   1. User can have preferences like:
      1. Don’t send me any kind of notification.
      2. Don’t send me sms, rather send me emails.
   2. We will put the user’s preference in a Preference DB (MySQL). This will mostly be used by our company, not the “SaaS product” since they will handle this at their own end.
   3. It will talk with a user service with which we can fetch the user related data like emailID, phone number, etc.
   4. Now we have the exact details of whom we want to send, what we want to send, and how we want to send. So, we put the request into another KAFKA for sending it out.
   5. Why so many Kafkas? => If there are certain spikes for certain types of notifications, then adding hardware is one option but we don’t want to do that all the time. We put it into a Kafka and we let the notification handler read and consume the messages at its own pace.
6. Notification Handlers:
   1. SMS Handler: We can have multiple vendors to deal with the SMSs in Asia, Europe, etc so we can integrate it with the SMS Handler Service.
   2. Email Handler: Takes all the email requests and forward it to the email vendor OR our own SMTP server.
   3. In-App Handler: FireBase for Android, Apple Push Notification Service.
   4. IVRS Handler: For handling IVR calls to users for confirming their order.
   5. Whatsapp Handler: Integration of NEW Handler is now very simple since we just want to write a new handler which does that handling and introduce a new “type” of notification in the notification service. Hence, the pluggable requirement is satisfied.
7. Notification Tracker: To know all the communication that has been sent out. We put the data into a Cassandra. Here the data will mostly be in write mode except for some cases when there is a audit when it will be read (Write-heavy).
8. BULK Notifications:
   1. If we want to send notifications to all the users who have bought “X” item in the last 24hrs?
   2. Bulk Notification Service: Bulk Notification UI will talk with a Build Notification Service with a filter criterion and a notification and then send it out. Filter criteria would be something like: “Find all the users who have ordered Potato’s 14 days back”
   3. User Transaction Data: It will put all the suer bought this and that kind of details into a Kafka and it will be read by something called as Transaction Data Parser.
   4. Transaction Data Parser: It parses all that information and puts it into a database (Mongo) or a elastic search on top of which it can do search and other aggregated queries.
   5. Query Engine: It takes a query (filter). It will have its own DSL and query the data store and would return a list of users which match the criterion.
   6. BULK Notification Service now has everything and will call Notification service with the details.