**Cloud Application Development**

**Project**

**Sending Notifications to patients using push notifications**

**Problem Statement** – Designing an architecture on AWS, which can send notifications to patients using push notifications based on Doctor Feedback.

High-level architecture for sending notifications to patients using push notifications based on doctor feedback:

**Designed Application on pubic cloud: -**

**Front-end:** A mobile app for patients that is able to receive push notifications.

**Back-end:** An AWS-based serverless architecture that includes the following components:

**AWS Lambda:** To run the serverless functions that send push notifications to patients.

**AWS SNS:** To send the push notifications to the mobile app.

**AWS DynamoDB:** To store doctor feedback and patient information.

**AWS API Gateway**: To expose a REST API for the front-end to access the back-end services.

**Workflow: -**

The doctor provides feedback in the form of a record in DynamoDB.

The Lambda function is triggered by a DynamoDB stream, reads the feedback, and retrieves the relevant patient information from DynamoDB.

The Lambda function formats the push notification using the patient information and the doctor feedback, and sends the notification to the relevant patient via SNS.

The patient receives the push notification on their mobile app.

This architecture provides a scalable, secure, and highly available solution for sending notifications to patients based on doctor feedback.

**why deploying it on cloud is beneficial: -**

**Scalability:** Cloud platforms can easily scale up or down as per the usage and traffic, making it easier to handle sudden spikes in demand.

**Cost-effective:** Cloud computing eliminates the need for physical infrastructure and its associated costs, providing cost savings in terms of maintenance and upgrades.

**Easy maintenance:** Cloud platforms allow for easy maintenance and updates, reducing downtime and improving the overall user experience.

**Cloud Platform we will be using is AWS**

For the problem statement, I would recommend using Amazon Web Services (AWS) as the public cloud platform. AWS provides a wide range of services that can be utilized to build a scalable, secure, and reliable notification system.

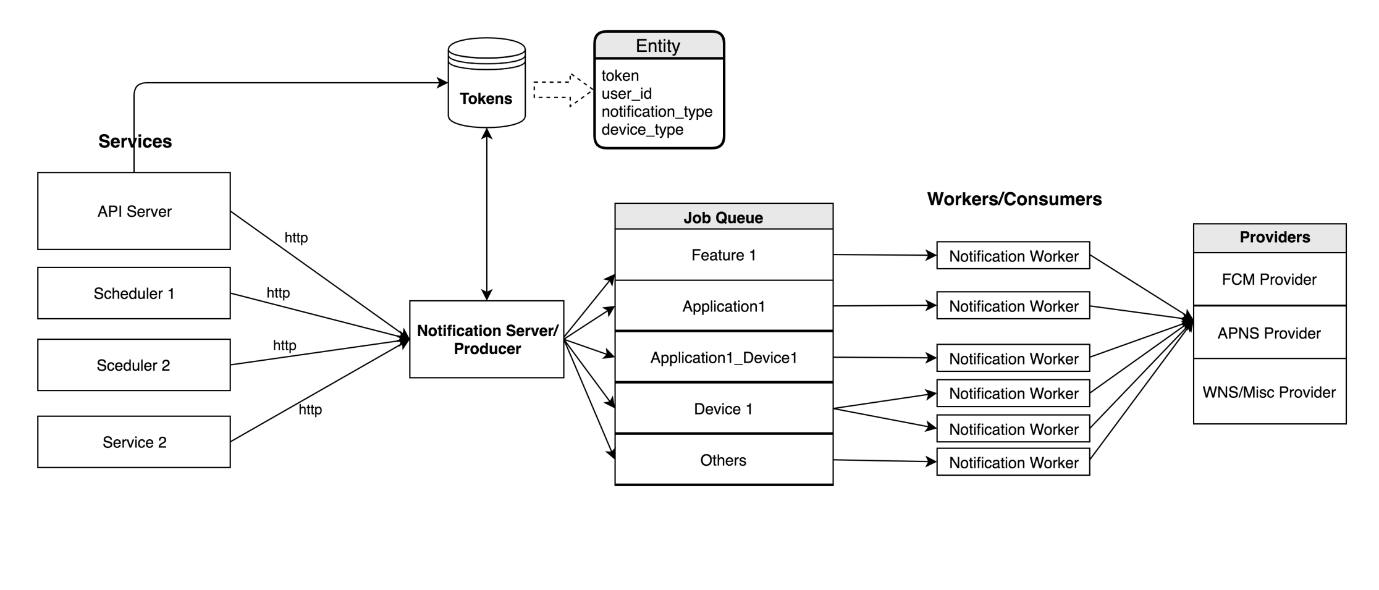
AWS provides services such as Amazon Simple Notification Service (SNS), Amazon Lambda, Amazon DynamoDB, and Amazon API Gateway, which can be used to implement the proposed architecture for sending notifications to patients. These services are highly scalable and can be easily integrated to create a complete solution.

Additionally, AWS has a strong security posture and provides various security features, such as VPCs, IAM, and encryption, to ensure the privacy and security of patient data.

AWS also has a large and growing global infrastructure, making it a reliable choice for delivering notifications to patients around the world. The platform is also widely adopted and has a large community of developers and partners, providing access to a wealth of resources and support.

Overall, AWS is the preferred platform for this problem statement due to its comprehensive set of services, strong security posture, and global infrastructure.

**Architecture: -**

****

Token Store

Whenever user logs in to Application, client sends registration token along with other metadata such as device type, user Id, and application type to server. Server stores above information in Token Store.

Services

Whenever any notification is to be triggered, different services or schedulers will send notification object over http to Notification Server.

Notification Server

Notification server exposes rest apis to send push notifications to client. Registration tokens and other data is fetched from token store based on passed user Id if we want to send targeted notifications. Server schedules a job on one of the job queues based on current use case.

Job Queue

We can use any queue management service like Kafka, RabbitMQ as an exchange for our job queues.

The number of queues can be decided based on features, device type, or application types in order to have even load on all queues and clear segregation based on business logic. The maximum number of queues can be dependent on number of cores available for efficient usage.

Notification Workers/Consumer

The worker processes consume message from job queues and send out messages to respective providers based on device \_type. If particular queue is loaded heavily (One feature/ [device type]/application used more frequently than others) we can have multiple workers consuming messages from same queue.

Notification Providers

Notification providers will send push notification to client devices. Providers will use different SDK and different mechanism to send notification based on the type of device they cater.

Yes, SIMD technology can be used in a project that involves sending notifications to patients using push notifications. By using SIMD, the processing of large amounts of data related to the notifications can be performed in parallel, making the process more efficient and faster.

For example, the system can use SIMD to process multiple patients' data and send push notifications to each of them simultaneously, rather than processing each patient's data and sending notifications one by one. This can help in reducing the overall time required to send notifications to all patients.

In addition, SIMD can also be used to process the data related to the notifications, such as patient information, appointment details, and more, to ensure that the notifications are accurate and relevant to each individual patient.

Overall, the use of SIMD in such a project can lead to improved performance, reduced processing time, and enhanced user experience.