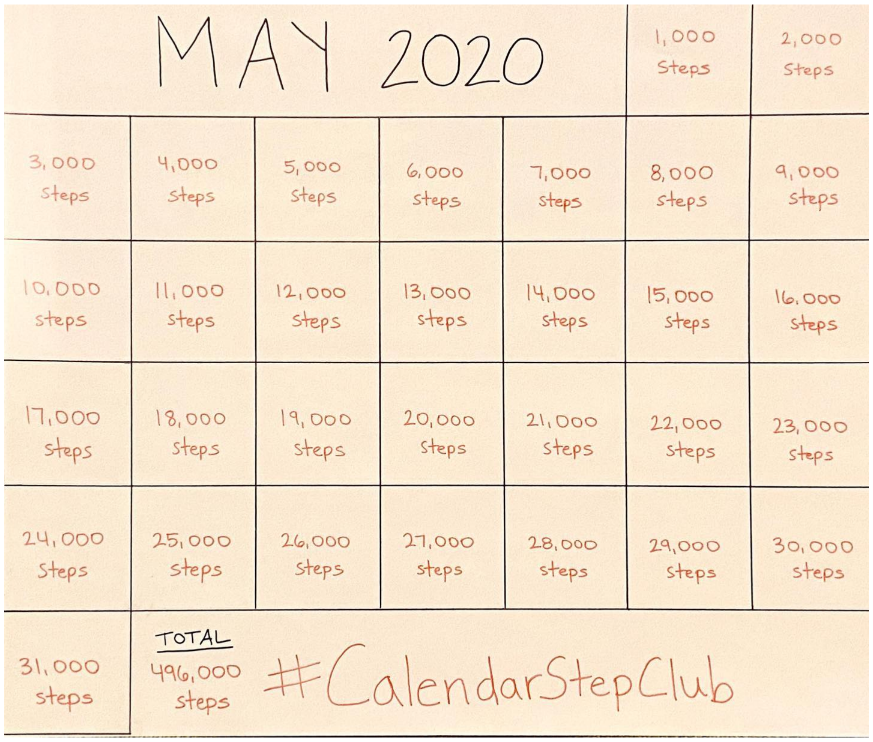
Decoupling

Decoupling in the context of cloud architecture usually means designing the different parts of an application such that they can function independently of each other. Decoupling is so ubiquitous in the modern tech environment that its actually hard even to imagine an example of a software architecture with different parts of the application tightly knit and dependent on one another.

Lets say for example that we were designing a pedometer application which counts the number of steps each day and then displays it on a calendar even if the device is offline. Then, it might benefit us to package the database and front-end application together within the same distribution (.apk, .exe file) such that these two parts of the application are shipped together and run together, without any middlemen. A mock UI of said app is given below:



MAY 2020					1,000 Steps	2,000 Steps
3,000 Steps	4,000 steps	5,000 Steps	6,000 steps	7,000 steps	8,000 steps	9,000 steps
10,000 steps	11,000 Steps	12,000 steps	13,000 steps	14,000 steps	15,000 Steps	16,000 steps
17,000 Steps	18,000 steps	19,000 steps	20,000 Steps	21,000 steps	22,000 steps	23,000 steps
24,000 Steps	25,000 steps	26,000 Steps	27,000 steps	28,000 steps	29,000 steps	30,000 steps
31,000 steps	TOTAL 496,000 steps	#CalendarStepClub				

This however means that the application and the database code are coupled together, and that if an issue were to happen with say, the database, then the front-end application would go down as well and vice versa since separation between the two parts is non-existent.

Now, it is not hard to see how this could be a problem in more modern software paradigms which rely on quite complex architectures with many different moving parts. After all, risking the operation of the entire application every time a small component within it faces issues.

Decoupling is extremely important within an AWS environment in particular, as we often rely on a wide array of offerings, often belonging to different category of services and serving unique purposes and during such situations it becomes almost a necessity to ensure that issues in one service does not lead to the failure of the entire application.

This also allows us to manage and scale AWS services independently of one another, enhancing the flexibility as well as the reliability of our architecture. Now, there are three major offerings provided by AWS that enable us to efficiently decouple our AWS environment. These are as follows: **Simple Queue Service (SQS)**, **Simple Notification Service (SNS)** and **EventBridge**. All three will be discussed in the next section.