Muhammad Farooq

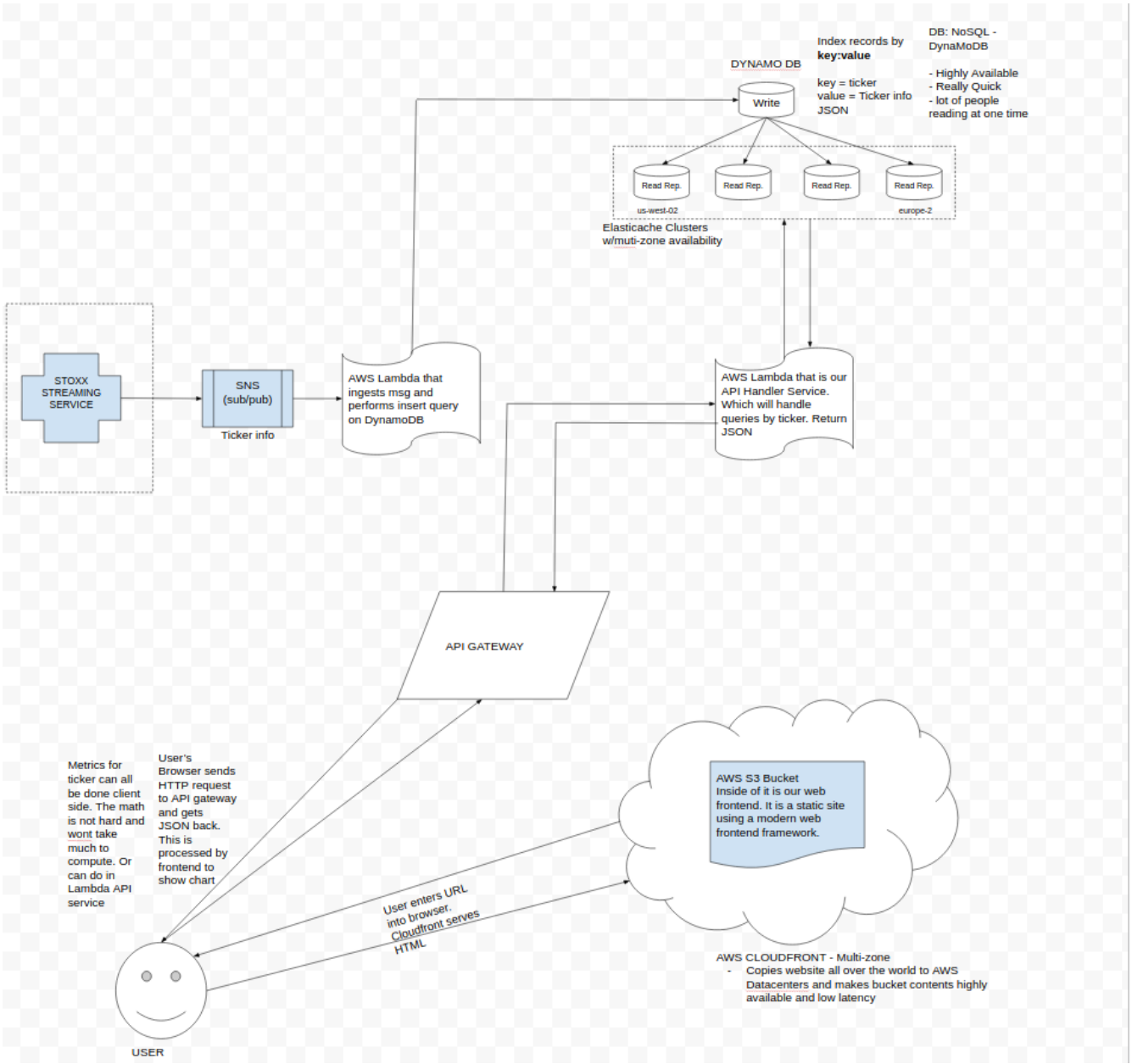
This document is going to help us with making stoxx web application with the help of using functional requirements such as charting, News Feed, and user management. After the application is designed, it should let the client interact with UI and get information and news for financial market.

Let start the application with looking at whats provided to us, we have been given a target application called stoxx which provides us with the financial information. Stoxx sends us tickers every second and we are going to bring it into the system and setup and RSS feed so we can setup subscription to get notifications from the service. We will be using Amazon Simple notification Service (SNS) so we can have data exchanges with the quote streams. Afterwards we are going to use AWS Lambda that ingests message and performs insert query on Dynamo DB because we want the information to be highly available and accessible. We need to have very robust data base because this is global so we can be receiving millions of tickers so we don't want our database to crash. That we are going to have elasticache clusters with multi zone availability and we are going to have various different read replicas only so it can handle the incoming traffic and users from everywhere around the globe. Since lambda is serverless so it can run multiple instances. This is why we need API Gateway which will consists of multiple lambdas, so whenever users browser sends HTTP request and to API GATEWAY and then it gets JSON back. For the news feed, we just need to create another SNS service which will communicate with our database. It can can feed the news from 200 streams as a key value and our lambda will be able to handle the situation accordingly. We will have standard user management for this application. We will use cookies in our database so we can save and manage the users data and this will also give them custom-ability over their interface.

We are using the S3 buckets to send requests from the client browser to the API gateway, the API will handle the requests from relevant read replica. It will return that info and return it to the front end. We will also use AWS cloudfront with S3 buckets. This service will make the buckets content highly available. Cloud front is like elasticache for S3. This will help with latency since people from all over the globe be accessing this application.

To make this application even better I would use Amazon elastic Beanstalk application. It allows for auto scaling and handles a lot of other complexities. It does all that automatically. Its main purpose is to help automate the web application deployment.

Below is a flowchart/diagram to help visualize this documentation.


DYNAMO DB
Index records by key:value
key = ticker
value = Ticker info JSON
DB: NoSQL - DynaMoDB
- Highly Available
- Really Quick
- lot of people reading at one time
Write
Read Rep.
Read Rep.
Read Rep.
Read Rep.
us-west-02
europe-2
Elasticache Clusters w/muti-zone availability
STOXX STREAMING SERVICE
SNS (sub/pub)
Ticker info
AWS Lambda that ingests msg and performs insert query on DynamoDB
AWS Lambda that is our API Handler Service. Which will handle queries by ticker. Return JSON
API GATEWAY
Metrics for ticker can all be done client side. The math is not hard and wont take much to compute. Or can do in Lambda API service
User's Browser sends HTTP request to API gateway and gets JSON back. This is processed by frontend to show chart
AWS S3 Bucket
Inside of it is our web frontend. It is a static site using a modern web frontend framework.
User enters URL into browser. Cloudfront serves HTML
AWS CLOUDFRONT - Multi-zone
- Copies website all over the world to AWS Datacenters and makes bucket contents highly available and low latency
USER