# CS554_A

# Lab3

Yuankai Ma

## Scenario 1: Logging

Because server will create, edit, or delete large number of log every day. So, it is very important to store the log entries properly. I will choose the MongoDB to store my log entries in JSON. MongoDB has flexible document schemas and powerful querying and analytics. For users to submit log entries. I will have them sent the data to the server in the web form, and then the server will store the data to the MongoDB. And because our database is non-relational. So, I can use NoSQL to submits lo entries. For the query part, it will be same as submit. I will use NoSQL to query the log entries. I will allow user to see the some of the log whatever they query. But some log likes password I won't allow them to see it. And for to log entries that I show them, it will be well organized and show on the website. For web server, I will use cloud server, and it can be Amazon's AWS. Because AWS's benefits is easy to use, flexible, cost-effective, reliable, scalable and high performance, and most important it is secure.

## Scenario 2: Expense Reports

Because the data structure is always the same, so I will use a relational database. Amazon Relational Database Service (RDS) or PostgreSQL will work. RDS make it easy to set up, operate, and scale a relation database in the cloud. It provides cost-efficient and

resizable capacity while managing time-consuming database administration tasks. So, for service I will use Amazon Relational Database Service. For the library that send the email, there are Nodemailer, EmailJs, Postmark, and Mailslurp-client. I will use Nodemailer, it is the solution most Node.js user turn to by default nowadays. Because it is heavy focus on security, because we don't want any gap that can let user to check other's expense. And the code is easily auditable because it is a single module with zero dependencies. And the most important benefit of Nodemailer is it embedded image attachments for HTML content because we need to send the expense pdf in email to the user. For generation PDF, I will use LaTex. It is a markup language, similar to HTML. LaTex is customizable, it has beautifully typeset output and it is good at dealing with mathematical notation. Layout and entry are generally easier using LaTeX than some other sort of equation editor. For handle all the templating for the web application, I will use VueJs, it is a JavaScript framework for building reactive web user interfaces. It is simple, flexibility. It is good to use it in building lightweight web application which is matching for our application.

## Scenario 3: A Twitter Streaming Safety Service

For the Twitter API, I will use the Academic Research of Twitter API V2. It allows the system to retrieve up to million tweets per month. It offers user the highest level of access. That makes system can access full-archive search Tweets. This API allows the system to access advanced filter operator. It can set up to 1000 rules for filter. That means system can custom a lot of filters. For expandable it to beyond my local precinct.

We can set a location filter for the tweet. And police department in other area, they can just change the location filter to their location. For make sure the system is constantly stable, I will use the reliable and stable web server. And set a test for the system with a large amount of data at regular intervals and periodically maintain the system. For web server technology, I will use AWS. And I will use MongoDB as my database, it is very suitable for system to store JSON format data including all the media in the tweet such as picture and URL. It can save all the tweet, the words that trigger the alert, and its investigation status. And another database to store all tweet that post. For those tweet that trigger the alert, the system can filter the location of the tweet and sent it to the officers in the local police department by email. The email technology I will use Nodemailer. After the officer finish and update the investigation status, the system will generate it into a pdf report using LaTeX which makes the report can be print by the user. For handle storing all the media that I have store, I will clear the sweet which did not trigger the alert at regular intervals.

## Scenario 4: A Mildly Interesting Mobile Application

For the data of the geospatial nature. I will store the long longitude and latitude into the database. By doing that I will use the Geolocation API. It allows the user to provide their location to the application and the system will get the longitude and latitude of the location. But it will need the user's permission to get their location. For the database, it can be MongoDB or Redis. But I will use MongoDB here. Because when comes to search data that in some condition (longitude and latitude), MongoDB is easy and better than

Redis. When we want to store image in MongoDB, we can use GridFS. It will allow user to store large-sized images, it will store the large file into small chunks. For long-term, cheap storage, MongoDB can do it very easily, and MongoDB can handle a very large amount of data. MongoDB can handle fast retrieval data, because MongoDB stores data in a single document for an entity and helps in faster data read or write. For short term data, MongoDB can set the TTL for it to make it expire after some time. When the user uploads an image, the system will not only store the image into MongoDB, but also will also store the location and the longitude and latitude. And when the user query for the pictures in certain location. The system will get the image in that match the geospatial information of the location that user provided. And finally I will write my API in NodeJS and ExpressJS.