Lab 4 Choosing Technical Stacks CS-554

Scenario 1: Logging

How would you store your log entries?

I would store the log entries using NoSQL database. The main reason behind choosing it is that it will allow logs to have some common fields, and also support any number of customizable fields for an individual log entry. The popular NoSQL database is MongoDB which can store key-value pair for the log entries.

How would you allow users to submit log entries?

I would allow users to submit the log entries using HTML forms using the post method.

How would you allow them to query log entries?

I would allow the users to query log entries using the connection between front-end and back-end. The users can submit the request using front-end HTML page/form and then using some middleware to authenticate the parameters/body, the data can be filtered from the database using the routes to call the appropriate function, to query the required log entries.

How would you allow them to see their log entries?

I can allow them to see their log entries on the application using front-end framework for web application. It usually includes some way to structure the files (for example, via components or a CSS preprocessor), style the components, and associate data with DOM elements. It could be either HTML form or table or some relatable component.

What would be your web server?

I would like to use Express as the web server.

Scenario 2: Expense Reports

How would you store your expenses?

Here, I would use relational DB to store data/expenses. The reason for choosing relational DB is that the users wants to submit expenses, which are always of the same data structure. The relational DB which I would use is PostgreSQL.

What web server would you choose, and why?

I would choose E**xpress** as a web server. The reason behind using Express is that it is a lightweight package that does not obscure the core Node. It is a web application framework which will allow to spin up robust APIs, handling AJAX requests and managing routes.

How would you handle the emails?

I would handle the emails using Nodemailer. It is a single module with zero dependencies for Node.js, designed for sending emails.

How would you handle the PDF generation?

I would handle the PDF generation using PDFKit or npm `pdf-creator-node`.

How are you going to handle all the templating for the web application?

I can handle the templating using the template engine like handlebars. Handlebars compiles templates into JavaScript functions which will make the template execution faster.

Scenario 3: A Twitter Streaming Safety Service

Which Twitter API do you use?

I would use `https://api.twitter.com/1.1/search/tweets.json` API.

How would you build this so its expandable to beyond your local precinct?

I would use Node-geolocation to build this so its expandable to beyond the local precinct and keep track of latitude and longitude.

What would you do to make sure that this system is constantly stable?

I would ensure that this system is constantly stable by enforcing end-to-end procedures, proper server monitoring, mapping and monitoring network and using bid data analytics to predict outages.

What would be your web server technology?

My web server technology will be Express.

What databases would you use for triggers? For the historical log of tweets?

I would use MongoDB and Redis for triggers and for the historical log of tweets.

How would you handle the real time, streaming incident report?

I would handle real time, streaming incident report by following Ajax polling and WebSocket,

How would you handle storing all the media that you have to store as well?

I would handle storing all the media that I have to store as well by using MongoDB, Redis and Express.

What web server technology would you use?

As mentioned above, I would use Express as my web server technology.

Scenario 4: A Mildly Interesting Mobile Application

How would you handle the geospatial nature of your data?

I would use geolocation API like Google Maps API to handle the data for mobile application and Geolocation API for web server.

How would you store images, both for long term, cheap storage and for short term, fast retrieval?

I would use AWS and Client-side storage API or PostgreSQL database for long term and cheap storage and for short term, fast retrieval I would use MongoDB.

What would you write your API in?

I would write my API in Node.js.

What would be your database?

My database would be MongoDB.