CS-554 Lab 5

Choosing Technical Stacks

Scenario 1: Logging

In this scenario, you are tasked with creating a logging server for any number of other arbitrary pieces of technologies.
Your logs should have some common fields but support any number of customizable fields for an individual log entry. You should be able to effectively query them based on any of these fields.

Technology Stack I would prefer

Storing log entries:

- I will use MongoDB database to store any number of customizable fields for an individual log entry. As MongoDB supports JSON format (storing objects).
- React JS: User will be allowed to submit log entries using input fields or forms containing time/date field and store logging information.
- We can query log entries using a search bar.

Web Server: I will use Express as web server.

Scenario 2: Expense Reports

In this scenario, you are tasked with making an expense reporting web application.

Users should be able to submit expenses, which are always of the same data structure: id, user, isReimbursed, reimbursedBy, submittedOn, paidOn, and amount.

When an expense is reimbursed you will generate a PDF and email it to the user who submitted the expense.

Technology Stack I would prefer

XAMP (X for any operating system, Apache, MySQL, PHP)

- For storing the same data structure, I will be using MySQL database and Apache server.
- For generating expense report I will use LaTeX which is a document preparation system for creating printable documents. LaTeX is a content first approach, which can then be combined with a set of styles. To compile LaTeX to pdf, we need LaTeX compiler.
- I will use PHPMailer to mail the PDF file generated using LaTeX to the user who submitted the expense.

Scenario 3: A Twitter Streaming Safety Service

In this scenario, you are tasked with creating a service for your local Police Department that keeps track of Tweets within your area and scans for keywords to trigger an investigation.

This application comes with several parts:

- An online website to CRUD combinations of keywords to add to your trigger. For example, it would alert when a tweet contains the words (fight **or** drugs) AND (SmallTown USA HS or SMUHS).
- An email alerting system to alert different officers depending on the contents of the Tweet, who tweeted it, etc.
- A text alert system to inform officers for critical triggers (triggers that meet a combination that is marked as extremely important to note).
- A historical database to view possible incidents (tweets that triggered an alert) and to mark its investigation status.
- A historical log of *all* tweets to retroactively search through.
- A streaming, online incident report. This would allow you to see tweets as they are parsed and see their threat level. This updates in real time.
- A long-term storage of all the media used by any tweets in your area (pictures, snapshots of the URL, etc).

Technology Stack I would prefer

MEAN (Mongo, Express, Angular, Node)

- For using Twitter APIs, I will use server-side scripting language like angular to make request to twitter API and that results would be in JSON format.
- Database: MongoDB or Redis database should be used to store the above JSON results.
- Web Server: Express to handle multiple request at the same time. To store historical data, we can use MySQL and for querying it we can use Elasticsearch
- Real time streaming: I would use WebSocket client that subscribes to event corresponding to new tweet.
- For alerting the officers we can use nodemailer or mail() function.

Scenario 4: A Mildly Interesting Mobile Application

In this scenario, you are tasked with creating the web server side for a mobile application where people take pictures of mildly interesting things and upload them. The mobile application allows users to see mildly interesting pictures in their geographical location.

Users must have an account to use this service. Your backend will effectively amount to an API and a storage solution for CRUD users, CRUD 'interesting events', as well as an administrative dashboard for managing content.

How would you handle the geospatial nature of your data? How would you store images, both for long term, cheap storage and for short term, fast retrieval? What would you write your API in? What would be your database?

Technology Stack I would prefer

- For handling geospatial nature of the data, I would use Google API to identify the area and map with our data.
- For long term storage of the images I can use cloud technologies like AWS, Google Drive, Dropbox. For short term, cheap and fast retrieval storage we can use Redis.
- I would like to write my API using NodeJS and Express as a web server.
- I would use MongoDB as a database to store images, location.