Sai Vignesh Golla
CWID: 20008561

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 customizeable fields for an individual log entry. You should be able to effectively query them based on any of these fields.

How would you store your log entries? How would you allow users to submit log entries? How would you allow them to query log entries? How would you allow them to see their log entries? What would be your web server?

Ans:

- I would store log entries with these fixed attributes

  Data = { device,
  IPaddress,
  auth-success,
  timestamp,
  userID,
  otherDOC } in a mongoDB and save the extra or custom attributes if needed.

- I would first verify if the user is a admin or not with help of session-id (cookies) node express server and if verified that the user is admin and has access to submit log entries, I'll let him change it in DB with PUSH request.
- First verify if the user has access to see their log entries, if yes, I'll request the log entries with userID as filter from the mongo DB database.
- I would use JSON, mongoDB for data purposes cause the log entries here are a bit dynamic in fields and no sql database that is the mongo DB is more suitable than sql databases for dynamic databases.
- I would use Express server as it's vast and has highly supportive open-source community and would be able to integrate several third-party applications and services with Express.JS.

Sai Vignesh Golla
CWID: 20008561

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.

How would you store your expenses? What web server would you choose, and why? How would you handle the emails? How would you handle the PDF generation? How are you going to handle all the templating for the web application?

Ans:

- Here since the data is always structured we could use an sql based database server but since I'm comfortable with mongoDB as of now I'll use a mongoDB server to save the data. Both are good choices, you cannot go wrong with any of those. Prefer sql based cause it's more suitable for structured data types.
- I would use Express server as it's vast, highly supportive open-source community and would be able to integrate several third-party applications and services with Express.JS.
- I would use Nodemailer module to send emails because it lets me add pdf attachments and it's also free to use and it's open source, so no trust issues.
- I would use pdfkit to generate pdf which is a simple free to use module in npm, we can also define several parameters in this module to customise our pdf generation.
- I would use express handle-bars to create web templates which is well known for it's versatility and easy to use nature. Here we can actually use react too and make it a single page web application but for this scenario I'll prefer the express-handle bars cause it's easier to handle multiple templates that way.

Sai Vignesh Golla
CWID: 20008561

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).

Which Twitter API do you use? How would you build this so its expandable to beyond your local precinct? What would you do to make sure that this system is constantly stable? What would be your web server technology? What databases would you use for triggers? For the historical log of tweets? How would you handle the real time, streaming incident report? How would you handle storing all the media that you have to store as well? What web server technology would you use?

Ans:

- I would use the official twitter api and I would store tweets with potential risky tweets based on some key words, all I need to do is inject the query parameters into the api, but it comes at a price for frequent requests.
- I would use Nodemailer module to send emails because it lets me add pdf attachments and it's also free to use and it's open source, so no trust issues.
- I would use MessageBird API or Twilio API or some other sms sending api's to send sms. I'd prefer MessageBird cause of it's cheaper pricing and immediate response.
- I would use MongoDB cause it has versitiltiy of having dynamic documents in database to save the potential risky tweets in history db.

- MongoDB again to log the data, sql database here is not so apt cause tweets are so dynamic resulting in lot of null values if sql is selected.
- Since api from twitter is on real time, streaming can be done in real time by continuously updating the info for every constant small period.
- I would use mongoDB for long term data storage but I would prefer the twitters inbuilt api cause twitter has long term storage of it's tweets and all I need to do is search for that tweet or save the tweet's link in my DB.
- I would use Redis for caching and temporary data cause it's really good with handling temporary data storage and it has automatic expiries based on the time we set which comes very handy.
- MongoDB is good with querying words, so it can be used for CRUD, or twitter api can be used for searching directly, all it needs are some key words.
- Express server, due to its vast advantages and good support of multiple libraries for server which would also enable us to use multiple third party applications which are supported by express server.

Sai Vignesh Golla
CWID: 20008561

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?

Ans:

- Assuming data is requested as api from web application I would use nodemon, mongoose to create web api and to handle download or upload requests I'll create separate api's for them, I would write my API using nodemon, express and in JavaScript and JSON.
- I would use JSON, mongoDB for data purposes cause the data here are a bit dynamic and no sql database that is the mongo DB is more suitable than sql databases for dynamic databases.
- I would use Express server as it's vast, highly supportive open-source community and would be able to integrate several third-party applications and services with Express.JS.
- I would save data based on cities or I would tag the events with location tag and while querying in mongo db I can just use the location tag to filter. I would save the events with location info.
- I would use Redis for caching and temporary data cause it's very good with handling temporary data storage and it has automatic expiries based on the time we set which comes very handy.