Scenario 1: Logging

- To achieve this, we need following technologies: Express server, MongoDB, HTML5/Bootstrap4, Node
- HTML5/Bootstrap4 – we can create a simple but powerful logging form using it. Bootstrap4 will help you style it as awesome as you want.
- Express- Easy to implement, works well with node and simple to send the request to server
- MongoDB- Adding fields into MongoDB is as simple as you think. We can set some common fields for each technology and can allow them to add other fields well of their choice.

How would you store your log entries?

Ans- MongoDB

How would you allow users to submit log entries?

Ans- Form-submit, Jquery -Ajax or Window-Fetch

How would you allow them to query log entries?

Ans- MongoDB CRUD operations with Express Server- GET POST PUT DELELE

How would you allow them to see their log entries?

Ans- Get the data from MongoDB using express GET and pass it to HTML (you can use bootstrap4 forms etc to style it)

What would be your web server?

Ans- Express!!

Scenario 2: Expense Reports

This scenario can easily be achieved by using Express Server, Redis, PDFKit, NodeMailer and of course NODEJS.

- Redis- As Redis is fast in performance and works well with the structured fields.
- Express- Easy to implement, works well with node and simple to send the request to server.
- PDFKit- I found it to be very easy and simple to implement. It works well with the node as it provides node package to work with. Easy to understand code. (http://pdfkit.org/demo/browser.html)
- We can use Latex as well.
- NodeMailer – Node mailer is by far the best package to work with when it comes to sending emails. It's easy to set up, simple in understanding.

Explanation:

We will simply create a HTML form using bootstrap4 containing all the mentioned fields and upon submit we will pass all the data to back-end using express server. Upon submit we will save the submitted data into Redis as Redis works well with the structured fields and fast in performance as well and at the same time pass the data to worker which will then handle creation of PDF.

To create a PDF, we will use LaTeX which will convert the HTML content into PDF. We can use PDFKit as well as it is very easy to setup and require very less amount of code for creation of complex pdf. Once the PDF is generated we will pass it to another worker which will then use the node mailer to send the pdf. I suppose we need to save the pdf at some physical location as node mailer require physical location of pdf to be attach.

How would you store your expenses?

Ans- in Redis

What web server would you choose, and why?

Ans- Express for the above-mentioned reason.

How would you handle the emails?

Ans- Using Node Mailer

How would you handle the PDF generation?

Ans- using PDFKit or LaTeX with the help of worker

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

Ans- I will create a separate collection dynamically in MongoDB for each web application as they may contain different fields in database.

Scenario 3: A Twitter Streaming Safety Service

The design can be as follow:

- We will create a database/tables which will contains all the words which can results into triggering an alert.
- We will use Twitter's public streaming API to get the tweets. As we get tweets into the system, we will parse it and check against the words database created above. If there is a match we will create an alert. ElasticSearch can be used for faster performance.
- We will use Redis pub-sub which will continuously listen to the alerting channel. Once we find any tweet containing alarming word, we will publish it to the channel. We will create 2 workers which are subscribed to alerting channel and upon getting tweet it will analyses it and send email and text to corresponding officers.

- We will store the tweet into our database along with investigation status as well let’s say in alert database.

Which Twitter API do you use?

Ans- Twitter public steaming API

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

Ans- Handling large is what I’m interpreting with the question. Hadoop or AWS will do the job I believe.

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

Ans- Stability means continuous support. Continuous support can only be provided when we have resource availability at all the time. We can distribute our data and resources so that failure of one can be taken care by another. Apache Hadoop.

What would be your web server technology?

Ans- Apache Tomcat/ Express server

What databases would you use for triggers?

Ans- redis pub-sub for trigger, mongoDB as a storage

For the historical log of tweets?

Ans- Elasticsearch for historical log of tweets

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

Ans- We can either use Socket.IO which is the best option for streaming reports in real time or MapReduce job to handle real time streaming incidents.

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

Ans- AWS for media storage

Twilio API for text message, Node Mailer for email

Scenario 4: A Mildly Interesting Mobile Application

We are concerned about creating a server side for a mobile application which makes me think of using no other than Node as it gives a lot of simplicity to write API in node.

How would you handle the geospatial nature of your data?

There are many ways to get the current location of user like turf.js, IBMS etc but the best way is to use google API to get the current location. Once have the current location we can sort the data from database by passing the current location of user.

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

For short term, fast retrival, I would use MongoDB GridFS as it allows user to store binary data I the database in BSON format. For long term, cheap stoarage I would use cloud stoarage like AWS, dropbox etc by comparing price of it.

What would you write your API in?

Node, as we are just concerned about writing server side of mobile application. Djnago can be another option as well.

What would be your database?

MongoDB GridFS storage.