Scenario 1: Logging

How would you store your log entries?

-using MongoDB as a persistent storage because it supports customizeable fields

How would you allow users to submit log entries?

Generally, I can make web page and have users to submit their logs as .txt file

How would you allow them to query log entries?

Make another web page and use ElasticSearch on MongoDB to query log entries.

How would you allow them to see their log entries?

I will create a Logentries account and add the log to logentries, then invite users in my team who can see log entries.

What would be your web server?

Express.js

Scenario 2: Expense Reports

How would you store your expenses?

I store all the field names (id, user, isReimbursed, reimbursedBy, submittedOn, paidOn, and amount.) together in one collection in MongoDB. the reason why I use MongoDB is because it is simple to work with JavaScript comparing with the other database. Moreover, I am more familiar with MongoDB.

What web server would you choose, and why?

Express.js, I choose it because I am familiar with it. Moreover, it is very flexible and pluggable

How would you handle the emails?

I will use mailgen in Node.js package to generate clean, responsive HTML e-mails for sending transactional mail.

How would you handle the PDF generation?

I will use PDFKit which is a PDF document generation library for Node and the browser that makes creating complex, multi-page, printable documents easy, The PDFKit API is designed to be simple, so generating complex documents is often as simple as a few function calls.

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

I will use handlebars because it is simpler and easy to implement and for this task we don't need to consider speed. Based on that, I choose to use handlebars

Scenario 3: A Twitter Streaming Safety Service

Which Twitter API do you use?

1. I need to use search tweet to get the recent tweet published in 7 days, then with Using PowerTrack rules to filter Tweet ensures that customers receive all of the data, and *only* the data they need for the app.
2. Using filter real-time Tweet to query real-time tweet based on the required keywords.

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

Here I will use the two endpoints:
GET geo/reverse_geocode: Given a latitude and a longitude, searches for up to 20 places that can be used as a place_id when updating a status. This request is an informative call and will deliver generalized results about geography.
GET geo/search: Search for places that can be attached to a statuses/update. Given a latitude and a longitude pair, an IP address, or a name, this request will return a list of all the valid places that can be used as the place_id when updating a status.

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

- solve a minuscule subset of the full problem space first and get that deployed to production than it is to have a large project die
- fix minor error in time, because once mistake deployed, they can alter production data and this makes them prohibitely expensive to fix, in the worst case I have to rewrite years worth of data which is wasted time.
- Control for risk by knowing where I am not pretty handled and what is the stable part of the software
- Components close to the core requires more testing, slower development and more attention to error.
- Know the code quality and data quality of parts I will interact with. Be on guard and look out for bad APIs.
- I will need a supervision/restart strategy.
- Avoid the temptation to add more and more functionality. Build a sibling tool instead

What would be your web server technology?

Express.js

What databases would you use for triggers?

MySQL

For the historical log of tweets?

I will store the tweets in MySQL which contains the keywords

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

With tweet realtime filter API to filter the relevant tweets

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

Compress the media to make it as small as possible for storage and use Amazon s3

What web server technology would you use?

Express.js

Scenario 4: A Mildly Interesting Mobile Application

How would you handle the geospatial nature of your data?

I can store the image in s3, and store the path to the image in my MongoDB, and there in my MongoDB I can store the geo stuff

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

Amazon S3 is cloud storage for the internet. I can upload my data (photos, videos, documents etc.) by firstly creating a bucket in one of the AWS Regions by using Amazon S3 APIs then upload any number of objects to the bucket. S3 uses unique Ids called Keys to retrieve files from the bucket

What would you write your API in?

Write my API in node.js

What would be your database?

MongoDB