Jacob Roessler
CS 554
Lab 3

I pledge my honor that I have abided by the Stevens Honor System.

Scenario 1: Logging

I would go with our typical MEAN (without angular) stack that we are used to for this scenario. This means storing the log entries in MongoDB allows for easy modifying of individual entries as JSON objects with customizable fields making it so it doesn't have to be a set number of fields per entry. Using the combination of express and the MongoDB node driver also allows for easy querying using fields, so users providing query strings to the express server will allow them to find entries based on fields. Using middlewares in express will also be useful for logging things like date/time or applying any filters to the information. To actually be able to see each log entry, a small web template could be created using express-handlebars to insert their data into viewable HTML pages and styled with CSS if necessary, or a simple JSON response page could be rendered if visuals are not the focus.

Scenario 2: Expense Reports

For storage, both MongoDB and SQL make sense here, but SQL would be a little nicer because the number of fields is static. Whereas MongoDB introduces unnecessary features for dynamic schemas which could just serve to complicate creating the database, SQL's simplicity and strictness are preferred. Since I chose SQL, I will also choose the LAMP stack. This means using an apache web server. The server will serve a reactive HTML template form page implemented with jquery and ajax that will be used to provide feedback and only allow the user to enter the same required data structures with strict type checking on the client and server-side to only allow valid data into the SQL database. I would pipe the data from the server into a latex compiler which would handle creating the pdfs. Doing some searching, PHP has a mail function but this does not work on a local server so if the server is local I would use PHPmailer. To handle the templating a combination of PHP and client side javascript would be used. The fixed data vertical scalability of this application makes for a perfect application of the LAMP stack.

Scenario 3: A Twitter Streaming Safety Service

To ensure stability with Twitter I would query the official Twitter API because any other 3rd party API might add an extra layer of instability. To ensure expandability and rapid searching/indexing of all tweets using the provided keywords, I would use Elasticsearch to maintain the data. I would use express for the webserver with middlewares for marking sensitive tweets, catching updates, and scheduled refresh timers to make sure information is up to date. Nodemailer provides a module to send emails and will be used for the email alerting system. Sendgrid and Twilio provide a module to send SMS messages for the text alerting system. To keep the data reactive I would render an HTML page then use jquery to make ajax requests to the server to get new information. This ajax page would be responsible for maintaining the live feed of tweets with their threat levels. Elasticsearch provides good methods to log and index data in separate collections with a UI for viewing analytics for different precincts allowing for the database to be expandible. Similar to SQL's views elasticsearch provides something called "Filtered Aliases" which would allow filtering incident tweets into a historical database.

Scenario 4: A Mildly Interesting Mobile Application

The google maps API is capable of handling iPhone and android geolocation data. This will be used to populate the user portion of the app. For an android app, I would use Java and for ios, I would use swift for developing the app and administrative dashboard itself. Since people have different phone models with different quality cameras image quality and resolution will vary. Images would be passed through ImageMagick to create cohesive dimensions for images of different sizes to fit cleanly into storage and the UI of the app. For the short term, fast retrieval I would use Redis to cache/store the images. This might need to be scaled to a large cloud provider in the long term like dropbox, google drive, etc. The actual API itself would be written in Java for good cross-platform compatibility.