Choosing Technical Stack

Scenario 1: Logging

To keep the log entries, a NoSQL database like mongodb may be used. Mongodb is an object-oriented, simple, dynamic, and scalable NoSQL database. The advantages of using mongodb are excessive performance, high availability, and automated scaling. I would use MongoDB to keep log entries and their function for data manipulation. MongoDB has some efficient and uncomplicated statistics manipulation functions which can be used to Add, Remove, Update and Query various logs which would be saved in the Database. I would allow the user to see their log entries in the terminal or server-side log screen. For this, I would probably use server-side Node as a web server along with Express. since NodeJs is very compatible with MongoDb

Scenario 2: Expense Reports

Use MongoDB to store each expense object based on its structure. Since MongoDB is document-oriented cross-platform database software that falls under the category of NoSQL database software, it is easier to archive and retrieve items in MongoDB. I would choose NodeJS as my webserver because of its compatibility with different javascript packages and its readily available node package manager also I have the most experience with it and it pairs well with MongoDB. Modules like Nodemailer and xoauth2 can be used to manage e-mails. Nodemailer is a module used to send emails, while xoauth2 is used to create, send and receive tokens.

To Handle PDF generation PDFKit or makePDF can be used, which can be installed using the npm package manager. For templating I would use Handlebars Using Handlebars, we can create dynamic webpages that render on the server side or client side. Using Handlebars' conditionals, loops, partials, and custom helper functions, our web pages become more than just static HTML.

Scenario 3: A Twitter Streaming Safety Service

I will use the Twitter Search API to get new tweets. To make sure the application is scalable, I would keep the server generalized so that other towns or departments will be able to use it easily. By relying as much as possible on other services, such as cloud storage expansion, it is simplified. To keep the system robust, I try to use technologies and services that have support available. I will use ExpressJS and NodeJS for my webserver. For databases, I would probably choose Redis for triggers because of its speed, but MySQL for historical logs because of its robustness and common usage, MySQL uses data much simpler for further reporting, integrating, and managing. MySQL Server gives you more advantages with your data management and insight. To report a real-time streaming incident, I would choose the WebSocket client to subscribe to events that correspond to the new Tweet. I store all the media in the MongoDB database because it has a capacity for very large amounts of data and it also supports storing multiple media, document types. Use NodeJS and ExpressJS for the web servers for ease of setup and customization. The reason to use ExpressJS is that you can specify the application path using HTTP methods and URLs. It also includes a variety of middleware that can perform additional tasks in response to requests and responses.

Scenario 4: A Mildly Interesting Mobile Application

To manage the geospatial nature of the data, the user's latitude and longitude as they are using an application can be mapped using the Google Maps API, where details can be displayed on site with high accuracy. Reverse Geocoding makes the process very fast and accurate. For the long term and cheap storage, images can be stored in cloud services like AWS, Google Cloud Platform, etc. These cloud services protect data, applications, infrastructure from fraudulent activity, spam, and abuse. Cloud’s networking, data storage, and compute services provide data encryption at rest, in transit, and use. I would use the MongoDB database to store images for short-term and fast retrieval because it follows the NoSQL structure, which speeds up data storage and retrieval for short and long-term use. NoSQL databases are easier to scale. They're designed to be fragmented across multiple data centers without much difficulty. I would write my API using ExpressJS and Node and use the express session to keep session records of the user when login in or using the application.