Scenario 1: Logging

For this scenario I would use the tech stack of MongoDB, Node.js, express and HTML templating.

For storing the log entries, I would use MongoDB because it helps me in saving the data in JSON form and it would provide high performance, high availability as well as auto scaling, and moreover new fields can be added to the documents without making any change in the existing documents which is the requirement of the application, that individuals can have customized additional fields. Through MongoDB CRUD operation it will allow my application to perform submission of log entries as well as querying. MongoDB provides a rich collection of query operations through which I can perform different types of querying based on the application requirements.

With the help of HTML templating express-handlebars my client-side pages will be rendered for the user so that user can provide the log entries and submit it and can check query and check the log entries. I have chosen HTML templating because it is easier to render the JSON data into html data which is easier for the user to understand.

My server would be Express because, through that I can use MongoDB, Node.js and HTML templating easily and it will allow me to perform different routing methods like GET, POST, PUT etc.

And also, I can use middleware function with express to verify the request and response before transferring it to the database.

Scenario 2: Expense Reports

For expense report web application, I would use MongoDB, Node.js, Express, React, wkhtmltopdf, nodemailer.

To store my expense reports I will use MongoDB which is a NoSQL database and stores data in the document form which is easier to read as my application will be saving expense reports. And MongoDB CRUD operations are very efficient, through which I can provide an option to user to update or delete any expense report if it is not correct or the user has entered some wrong value.

For web server I will use express with server side Node.js which has a very rich open-source library through that I will be able to manage the user operations, expense report submission, creation, deletion, updation. Express helps in writing handlers for requests with different HTTP verbs at different URL paths.

For emailing I will use nodemailer, which is a Node.js module which gives the ability to send email without hassle and SMTP protocol is the main transport used by nodemailer for delivering messages. I chose Nodemailer because it also gives me an option to attach the document while sending email, which is the requirement of my application.

For pdf generation I will use wkhtmltopdf which is library that uses Webkit to take HTML and print it to PDF and I can easily use this with Node.js

For templating I will use React to render my documents stored in data base to show it to the user because through React I can change the user interface dynamically as soon as the user submits the expense report, they can see the updated status of their expense.

Scenario 3: A Twitter Streaming Safety Service

For Twitter Streaming Safety Service I will use MongoDB, express, React, Node.js, nodemailer, AWS cloud storage, Twilio

Enterprise PowerTrack API I will use which Provides complete coverage of public Tweets as they happen, filterable using a wide range of operators and rules. With PowerTrack API I can retrieve tweets which are based on a wide variety of attributes, including user attributes, geo-location, language, and many others.

For scaling my application beyond local precinct I can use PowerTrack API which provides myriad number of parameters to query the data from API location, place, place_country, lang, bio_location, profile_country, profile_region, profile_locality, profile_subregion etc. With these parameters I can expand my application to any location, or any country based on the requirements.

To make my application constantly stable I will use Big Data Analytics to Predict Outages - By collecting large volumes of data from across each system, both when they are running properly and when they fail, I can use analytics tools to discover trends that help predict future outages.

I will use express with server side Node.js as my webserver, express helps in writing handlers for requests with different HTTP verbs at different URL paths.

I will use MongoDB NoSQL database for storing possible events as well as history of all the tweets which triggered an alert in JSON format for retrieving, querying, creating, logging, and updating purpose. And through query options I will use the MongoDB search operation for querying tweets.

For real time streaming incident report, I will use React and its functional components, where I can leverage the functionality provided by hooks useState and useEffect which will allow my application to update the content in real time.

For email alert system, I will use nodemailer, which is Node.js module which gives the ability to send email without hassle and SMTP protocol is the main transport used by nodemailer. For text alert system to inform officers about critical tweets I would use Twilio package, using Twilio's Rest API I can send outgoing text alert from my Twilio phone number to mobile phones around the world.

For the long-term storage of Media, I will use AWS cloud storage through which I can store, access, govern, and analyze my data to reduce costs and increase agility. Amazon Simple Service Storage (S3) offers industry-leading scalability, data availability, security, and performance. This means in my application customers of all sizes and industries can use it to store the required data.

Scenario 4: A Mildly Interesting Mobile Application

For Mildly Interesting Mobile Application I will use MongoDB, Bootstrap, HTML Templating, Node.js, Express, AWS cloud storage, ImageMagick

For handling the geospatial nature of my data, I will use Google MAP API which will help my application to detect location of the user's device through which I can allow users to see mildly interesting pictures in their geographical location. With Google MAP API I can also allow my user to change the location using maps which will enable the user to see images from different locations as well.

For uploading images to the mobile application, I will use ImageMagick which is a powerful image manipulation utility. Images can be compressed, resized, cropped by this highly efficient program rather than doing these operations in programming language.

For long term storage of images, I will use Amazon Simple Service Storage (S3) where I have the option of autoscaling because through my mobile application users will upload plenty of images, and AWS provides the option of pay for what you use, so it is a cheap and efficient storage for my application. I am not using MongoDB for long term because it might affect my performance of the application as the data grows.

For short term storage of images, I will use MongoDB which gives me the option to store all types of data and I can store the images in binary form which is an efficient way and querying and retrieving of data is also easier. I can also use Redis for a session when the user is logged in to show the data from cache which is a very fast technique for displaying data to the user, my application would not have to query the database for the user's data.
My database will be MongoDB again because I can easily perform all the CRUD operations with it.

I will write my API in server side Node.js with express web framework and also, I will use express-session to store the session of the user when the user is logged in so that application's dashboard can be changed according to the user's login status. And also, I will use HTML templating with bootstrap which will allow my application to fit in any size of the device, my application will not be restricted to only mobile devices.