Scenario 1: Logging

Since logs being submitted have common fields as well as variable number of customizable fields, I would prefer to use a NoSQL database such as MongoDB in which the logs will be stored in JSON format.In the USER Collection, each user will have a logs array to keep track of their logs, every log is going to have an id, timeAndDate, common fields and customizable fields.

```
{
        “_id”: "7b7997a2-c0d2-4f8c-b27a-6a1d4b5b6310",
        “username”:”JohnDoe431”
        “password”:”hashPassword”,
        “logs”: [
                        {
                                “_id”:”7b7997a2-c0d2-4f8c-b27a-6a1d4b5b6311",
                                “timeAndDate”:”03/23/2020 14:45:34”
                                “commonField”:log,
                                “customField1”:log,
                                “customField2”:log
                        },
                        {
                        “_id”:”7b7997a2-c0d2-4f8c-b27a-6a1d4b5b6311",
                        “timeAndDate”:”03/23/2020 14:45:34”
                        “commonField”:log,
                        “customField1”:log,
                        “customField2”:log,
                        “customField3”:log
                        }
                ]
}
```

Users can submit generated logs through file upload/text submission forms by providing customizable field and its value in a web site built using HTML, CSS, JavaScript. Setting the customizable field as an object property the user can later query based on the common fields as well as the customizable fields.By providing search and filter options, users will be able to query log entries using keywords, common fields or customizable fields. They will also be able to query using the filter option where they can query using the common or customizable fields or date and time the log was created.

Users will be able to view detailed information of all their logs ina paginated format and also be able to query a specific log through a web interface built using React.jsI would prefer to user Express server in Node.js for that backend as it is compatible with MongoDB and easier to make API calls, and React.js for the frontend as it provides easier DOM manipulation.

Scenario 2: Expense Reports

Since the expenses being submitted are of the same data structure, I would use a relational database such as MySQL, with proper datatypes for all the mentioned fields. “id” would be the primary key with auto incrementing enabled to maintain a unique id for each expense.Node.js is what I would use because it integrates databases well and works well with SQL databases.

I would use the Nodemailer package in Node.js to send emails when an expense is reimbursed. By setting up a transporter object with my SMPT details such as email and password. We can add the generated PDF as an attachment to the email being sent.Using the pdfkit package in Node.js, we can generate PDF using the data present in the database. PDF will contain all the details such as, user, reimbursedBy, submittedOn, paidOn and amount.Express-handlebars can be used to handle all the templating and creating a dynamic single page web applicaton.

Scenario 3: A Twitter Streaming Safety Service

I would use Twitter Streaming API to stream through real time tweets.To make sure the application is expandable I would design the system in which each component runs independently and easier addition of new components when required and use distributed servers managed by a master server. For the stability I would add a load balancer to ensure the servers don’t get overwhelmed by the number of requests and automating tasks with constant updates to catch and fix bugs.

For the web server I would use Apache HTTP server as it can handle high volume of traffic.I would use a NoSQL database such as MongoDB to store the triggers, and a relational database such as MySQL to store historical log of the tweets.I would use a WebSocket server to handle real time streaming incident report, the server can parse the tweet and check for specific keyword and update incident report and send alerts via text message.To handle storing of all media, I would use a cloud storage service such as Google cloud storage or Dropbox, this would ensure long term storage of data as well as high volumes of data.

Scenario 4: A Mildly Interesting Mobile Application

To handle the geospatial data I would use MongoDB which support geospatial models and using the latitude and longitude information present in the image metadata to map the coordinates. For long term and cheap storage I would use a cloud storage service such as Google cloud storage or Dropbox, and for short term storage and faster retrieval I would use redis database. I would use Node.js to write API calls and also use express-session for user authentication. MongoDB would be my go to database as this application has user generated content which is unstructured data and redis cache for faster retrieval.