Name: Siddhant Barua
Cwid: 10439929

CS-554 Lab 5

Scenario 1: Logging

In this scenario, you are tasked with creating a logging server for any number of other arbitrary pieces of technologies.

Your logs should have some common fields but support any number of customizable fields for an individual log entry. You should be able to effectively query them based on any of these fields.

How would you store your log entries? How would you allow users to submit log entries? How would you allow them to query log entries? How would you allow them to see their log entries? What would be your web server?

Solution:

In this scenario I would use a noSQL database like mongoDB for logging purposes. This would be stored in s JSON format. And the users can submit the log entries through a JQuery-Ajax form after which we can POST the data into the noSQL database.

We can allow the user to query their log entries through a text box, and a set of radio buttons where in we use GET, PUT, DELETE, PATCH and POST operations in the backend which returns the data requested by the user.

We can allow the user to see their log entries, by having a hyperlink named “SEE ENTRIES” which returns a static HTML containing the entries for that particular user.

We would use express server.

Scenario 2: Expense Reports

In this scenario, you are tasked with making an expense reporting web application.

Users should be able to submit expenses, which are always of the same data structure: `id`, `user`, `isReimbursed`, `reimbursedBy`, `submittedOn`, `paidOn`, and `amount`.

When an expense is reimbursed you will generate a PDF and email it to the user who submitted the expense.

How would you store your expenses? What web server would you choose, and why? How would you handle the emails? How would you handle the PDF generation? How are you going to handle all the templating for the web application?

Solution:

I would store the expenses in the form of a JSON object and store them in the mongoDB database.

Name: Siddhant Barua
Cwid: 10439929

Data
{

“Id”: “something “,

“user”: “something “,

“isReimbursed”: “something “,

“reimbursedBy”: “something “,

“submittedOn”: “something “,

“paidOn”: “something “,

“amount”: “something “

}

I would use express server and node mailer to handle the email operations.

For the PDF generation I would use latex, which converts the HTML into a PDF and this can be used for templating purposes as well.

LaTeX is a document preparation system. When writing, the writer uses plain text as opposed to the formatted text found in WYSIWYG word processors like Microsoft Word, LibreOffice Writer and Apple Pages.

Scenario 3: A Twitter Streaming Safety Service

In this scenario, you are tasked with creating a service for your local Police Department that keeps track of Tweets within your area and scans for keywords to trigger an investigation.

This application comes with several parts:

- An online website to CRUD combinations of keywords to add to your trigger. For example, it would alert when a tweet contains the words (`fight` **or** `drugs`) AND (`SmallTown USA HS` or `SMUHS`).
- An email alerting system to alert different officers depending on the contents of the Tweet, who tweeted it, etc.
- A text alert system to inform officers for critical triggers (triggers that meet a combination that is marked as extremely important to note).
- A historical database to view possible incidents (tweets that triggered an alert) and to mark its investigation status.
- A historical log of *all* tweets to retroactively search through.
- A streaming, online incident report. This would allow you to see tweets as they are parsed and see their threat level. This updates in real time.

Name: Siddhant Barua
Cwid: 10439929

- A long term storage of all the media used by any tweets in your area (pictures, snapshots of the URL, etc).

Which Twitter API do you use? How would you build this so its expandable to beyond your local precinct? What would you do to make sure that this system is constantly stable? What would be your web server technology? What databases would you use for triggers? For the historical log of tweets? How would you handle the real time, streaming incident report? How would you handle storing all the media that you have to store as well? What web server technology would you use?

Solution:

The twitter search API is useful for this purpose, and I would expand it beyond the local precinct by firstly making the application a public website or an app, this is accessible from all portions of the world not just restricted to our local precinct. Security can be handled by making sure only personnel with required security clearance can get access to the services. To make the application more stable I would design it to be as simple as possible hence making sure uninitiated users will not botch things up. Simplicity is the only way to ensure stability. The web server technology I would use is Bluehost web hosting which is a pretty good way of making sure the website is public and available to people outside the precinct. The historical log of tweets can be stored in the mongoDB database and real time incident report can be handled using react due to it's inherent speed. The storing of all the media can be done using AWS(Amazon web services) as they have an almost infinite cloud storage space.

Scenario 4: A Mildly Interesting Mobile Application

In this scenario, you are tasked with creating the web server side for a mobile application where people take pictures of mildly interesting things and upload them. The mobile application allows users to see mildly interesting pictures in their geographical location.

Users must have an account to use this service. Your backend will effectively amount to an API and a storage solution for CRUD users, CRUD 'interesting events', as well as an administrative dashboard for managing content.

How would you handle the geospatial nature of your data? How would you store images, both for long term, cheap storage and for short term, fast retrieval? What would you write your API in? What would be your database?

Solution:

The geospatial nature of the data can be handled using Google's places and Geofencing API's. It is best to store the images locally in your own personal server , as this makes for the fastest retrieval time. But assuming the capital for a giant local server doesn't exist I would use AWS or DropBox for this operation .I would use a noSQL database like MongoDB and I would use NodeJS to write my API.

Name: Siddhant Barua
Cwid: 10439929