(Name: Chakravarti Kothadiya   CWID:20012229)

Choosing Technical Stacks - CS-554 Lab 3

Scenario 1

Answer: To store log data I will use a JSON supported NoSQL database - MongoDB as it allows for easy storage and retrieval of JSON-like documents and can handle large volumes of data with ease. Using MongoDB would allow us to store log entries in a flexible, schema-less format that can accommodate any number of customizable fields. Now for allowing users to send the data to be logged I will create API endpoint through POST routes using Express.js. To get the data where the user can enter in his required fields, I will create a UI using HTML handlebars views. After entering the data we can collect it through the req.body from the POST Route and then further insert it into the MongoDB database. We will use Express.js to set the server as it has built-in support for creating RESTful APIs. For the user to see their log entries I will create a GET route which will extract and query the required data form the MonogoDB database and to display it to the user I will render a handlebar sending all the extracted data to be displayed and then display it. If required that the data needed to be extracted based on certain conditions or matching fields then instead of using a GET route, we can use a POST route where the user can enter the required fields information and then according to that we can query the data further and display accordingly. For allowing the user to access only the authorized or personal data I will create middleware wherein I will store and check the user credential data in sessions and if matched then only allow access to authorized pages and data.

Scenario 2

Answer: For storing the expense data I will use MonogoDB as it allows for easy storage and retrieval of JSON-like documents and can handle large volumes of data with ease. Using MongoDB would allow us to store log entries in a flexible, schema-less format that can accommodate any number of customizable fields. For starting a web server, I will use Django framework as it is a simple way to create RESTful APIs and web-based UIs, and can be easily integrated with other libraries and tools. For sending email I will use Django's built in library EmailMessage and create a function to send emails. I will create an API endpoint through which I will call this function and send and handle emails. For generating PDFs of reimbursed bill, I will use PhantomJS thrid party library which allows to generate PDFs from HTML templates. I would create a template for the expense report, populate it with the data from the expense, and generate the PDF. The PDF would be attached to the email and made available for download from the web application. For handling all the templates, Django has its built-in template which are easy to use and we can define reusable template blocks, use template inheritance to create a consistent look and feel across entire application, and render dynamic content based on data from database. So, I will handle all the view templates using Django's templates.

Scenario 3

Answer: For Twitter API, I would use the Twitter Streaming API, which allows real-time access to a sample of the Twitter firehose, filtered by keywords and other parameters. This would enable the application to listen to tweets in real-time and capture relevant tweets based on the keywords that have been set up. I would design the system as a cloud-based microservice architecture, which can be easily deployed to multiple instances in the cloud. This would allow the system to scale as necessary, based on the volume of tweets, and handle additional precincts with ease. To ensure that the system is constantly stable, I would implement automated testing and continuous integration and deployment practices. This would include automated testing of the system's functionality, performance, and security, as well as automated deployment to production. For the web server technology, I would use Django framework. This framework provides built-in support for templating, and authentication, making it easy to develop and maintain complex web applications. For text alert system, in order to send critical triggers to the police I will use a third-party API called Twilio, which will provide a SMS service provider service by configuration of credentials. For storing the triggers and the historical log of tweets, I would use a relational database such as PostgreSQL. These databases are reliable, scalable, and can handle large volumes of data. To handle the real-time, streaming incident report, I would use a real-time data processing framework such as Apache Kafka. This would allow the system to process the incoming tweets in real-time and provide an incident report that is always up-to-date. To handle the storage of all the media used by any tweets in the area, I would use a cloud-based storage solution such as Amazon S3 storage as these services are reliable, scalable, and can handle large volumes of data.

Scenario 4

Answer: To handle the geospatial nature of the data in this scenario, I would use a geospatial database such as PostGIS, which is an extension to the PostgreSQL database that adds support for geospatial data. This would allow us to store the coordinates of the location where the picture was taken and easily perform spatial queries, such as finding all interesting events within a certain radius of a given location. To store the images, I would use a combination of long-term, cheap storage and short-term, fast retrieval. For long-term storage, I would use a cloud-based object storage service such as Amazon S3 Storage. These services are designed for storing large volumes of data and are cost-effective. For short-term, fast retrieval, I would use a content delivery network (CDN) such as Cloudflare, which would cache the images in multiple locations around the world for fast access by users. For the API, I would use a modern and widely-used web Django framework. This framework provides built-in support for building RESTful APIs and are easy to use and maintain. For checking if the user has account and has logged in, I will create middleware functions which will store/check the users login credentials from the sessions data. For the database, I would use a relational database such as PostgreSQL. This database is reliable, scalable, and provide strong data consistency guarantees. Additionally, it also has built-in support for geospatial data through extensions such as PostGIS. For the administrative dashboard, I would use React web framework. It provides a rich set of components and tools for building dynamic and interactive user interfaces. The dashboard would allow administrators to manage users, interesting events, and other content, as well as view analytics and other metrics related to the usage of the mobile application.