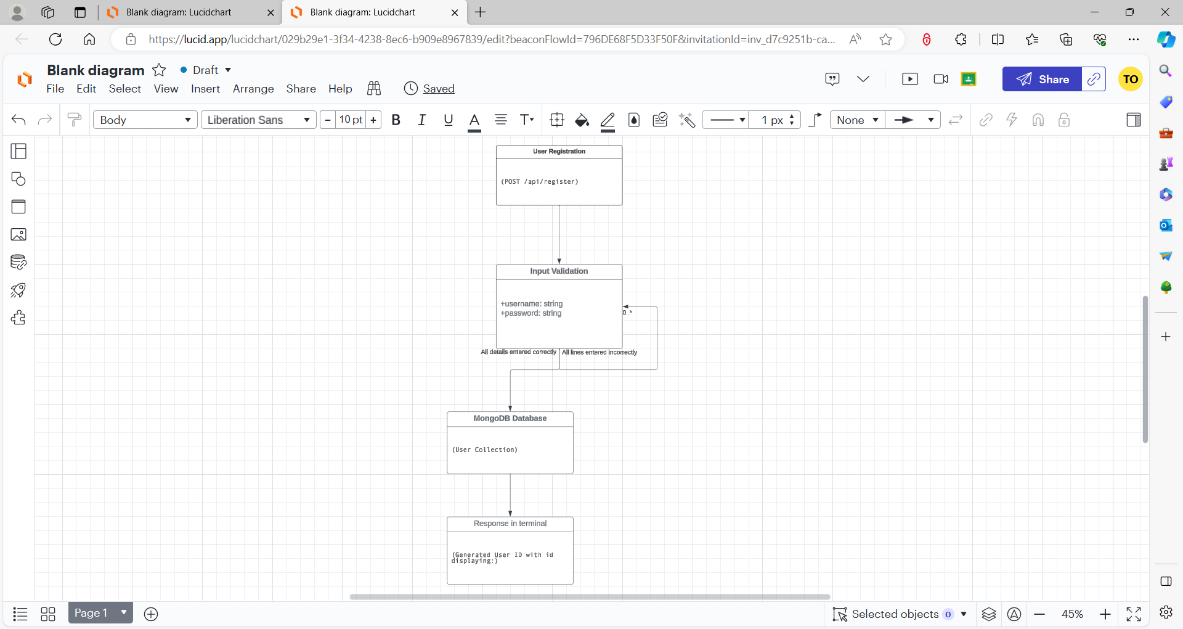
Service-Centric & Cloud Computing Report

Terrence Oppon N1004932

**Brief function description**

**Generating a unique user**

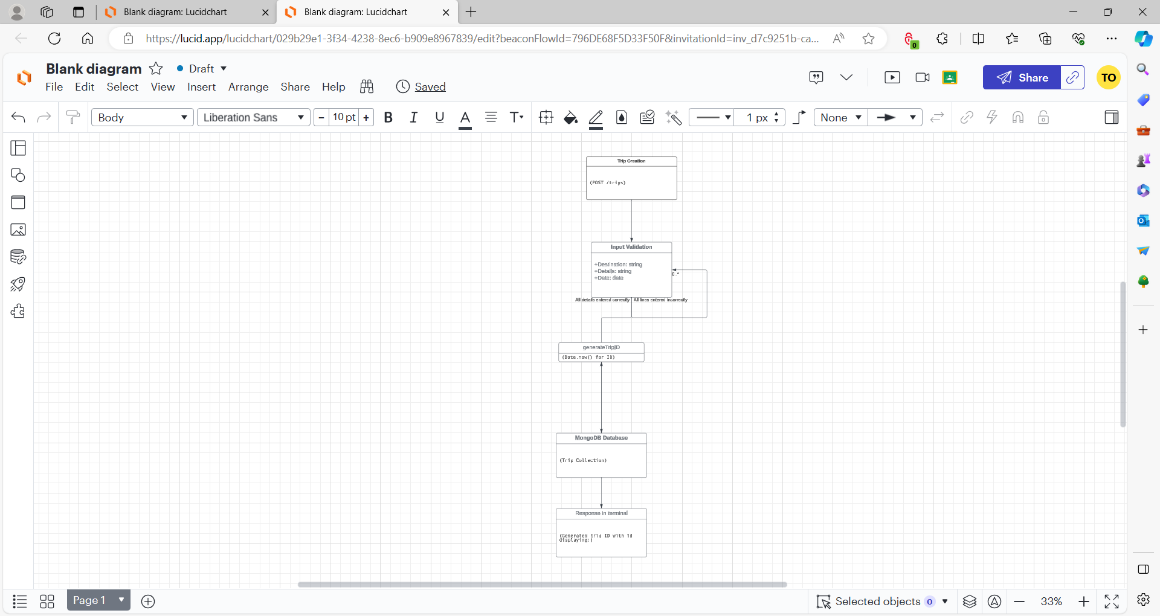
In my project, new users can create an account using the user registration feature when entering their username and password. The input is checked to make sure both fields are supplied when a POST request is sent to the /api/register endpoint. By calling the generateUserId() function, which sends a request to the external api random.org, yields a unique user ID. This assigns the new user a random number between 1 and 100. After that, the user data is saved and pushed in JSON format to a User database in the MongoDB database. The response contains the generated user ID.



**Trip Creation**

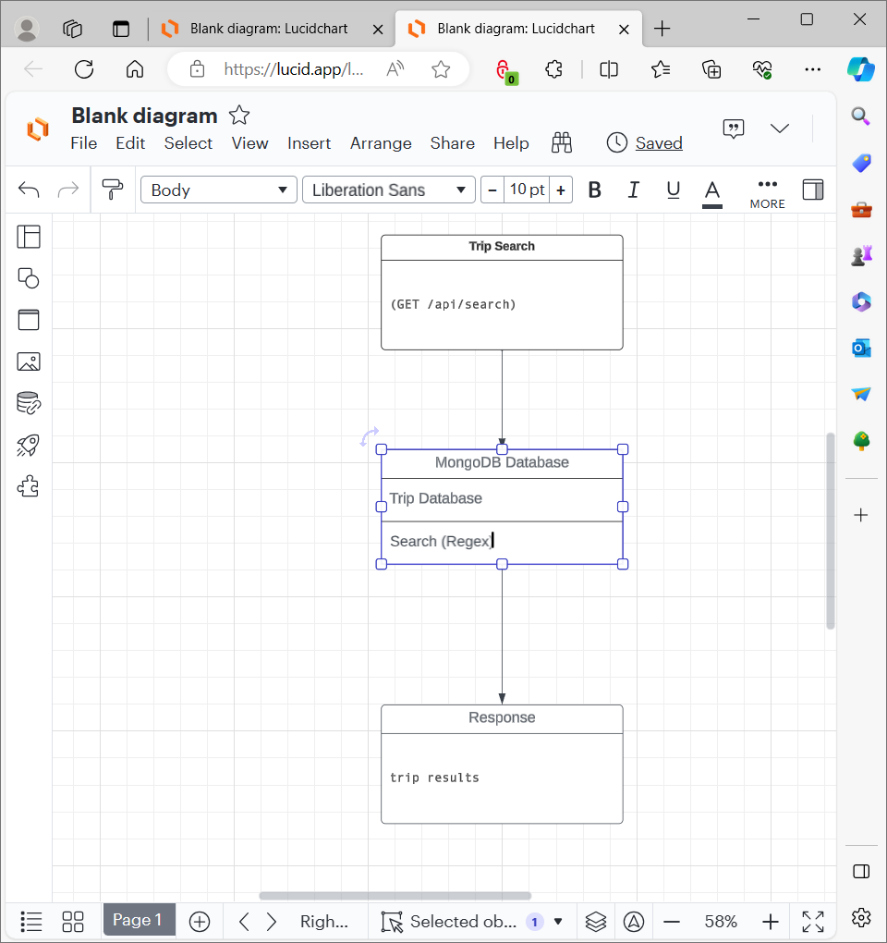
The trip creation feature allows users that are logged in to create new trip records using the trip creation feature. The input is validated to make sure a destination, details, and the trip date are provided when a POST request is made to /trips. Also, when a destination is typed in, it will connect to the weather org. External api to get the weather details for that specific location.

Meanwhile for creating a unique ID for the trips, the Date.now(), which returns the current timestamp in milliseconds, is used to generate the unique ID. You can make multiple trips without running into problems because this timestamp is guaranteed to be distinct. When a new Trip document is created, the tripId property is assigned to each trip. Then, a Trip document in JSON format containing the trip details, the user ID and the weather forecast for that location and the generated ID is saved to the database. The response includes the trip ID back.



**Trip Search**

Finding trips by destination is possible with the trip search feature. The request query parameters may contain a search term. Using a regular expression for case-insensitive matching, the /api/search endpoint filters the database's trips where the destination matches the search term. The response contains the results that have been filtered.



**Express interest**

Authenticated users can express interest in a trip using the express interest feature. The trip ID is passed in the URL parameters of a POST request made to the /api/expressInterest/:id endpoint. The authenticated request yields the user's ID. The user's interests' array in the database is then updated with the trip ID to reflect the user's interest. To keep track of who is interested, the user's ID is also added to the interested Users array for the trip. This makes it possible for users and trips related to their interests to link in both directions. If the update is successful, the updated user and trip documents are returned in the response; if the user or trip cannot be located, a 404 error is returned.

**Section C**

**JMeter testing**

I used JMeter to analyse my results. I tested the GET trips port and edited the thread group so there is the 1 Thread Group containing 100 threads (virtual users). This allows checking application performance with a significant load. and changed the ramp-up-period to 10 seconds. Some of the things I found out after running the JMeter test are:

* In ten seconds, the test increased the user count by 100. This mimics a traffic build-up over time.
* Every thread runs in less than a second. This suggests that the response time for each request is extremely quick.
* In one minute, all 100 threads (users) were able to finish. Errors were not reported. This shows that there will not be any problems with the application's ability to manage 100 simultaneous users.
* The application has enough capacity to handle the configured load, as each thread is ramping up and finishing quickly. There are no visible bottlenecks.
* But these logs alone cannot tell us anything about hardware limitations or true scalability. It is necessary to progressively increase the load to 500, 1000, or more users to determine that and to track response times.
* One JMeter engine was used for the test. The load must be divided among several JMeter engines that are contacting various application instances to properly test distributed scaling.

Here is a technical discussion on deploying the application on a distributed cloud infrastructure with improvements:

**General Method:**

To increase scalability, availability, and fault tolerance, the application can be installed on a distributed cloud infrastructure such as Azure.

- Docker can be used to containerise the Node.js application server, making it simple to scale and deploy on cloud virtual machines.

-I used a MongoDB Atlas, is a platform on which the MongoDB database can be hosted.

- For high availability, the application can be spread across several Availability Zones.

- Traffic can be divided among several application instances by an application load balancer.

- The number of application instances can be automatically scaled up or down based on load using Auto Scaling Groups.

Scalability

- To handle more traffic, the application can grow horizontally by adding additional application server instances behind the load balancer.

- MongoDB Atlas offers auto-scaling features to expand storage and processing capacity as required.

If I use Docker containers, application instances can be deployed and scaled more quickly.

- Auto Scaling Groups enable the configuration of dynamic scaling policies for instance scaling according to metrics such as CPU utilisation.

**Accessibility and Reliability**

- Spreading the application over several AZs increases availability in the event that one AZ experiences a failure.

- Only healthy instances will receive traffic due to the load balancer's ability to identify unhealthy ones.

MongoDB Atlas offers multi-AZ replication to ensure high database availability.

**Observation and Record-Keeping**

- Application performance monitoring and logging could be done with CloudWatch. The application has the ability to publish custom metrics to CloudWatch.

**Increasing QoS**

Redis/Memcached can be used to implement caching to speed up common request responses and lighten the load on the database.

- To prevent an overload on application instances, request throttling can help limit excessive incoming requests.

- To improve request response times, implement request queues with SQS for long-running task background processing.

In conclusion, compared to on-premises deployment, utilising managed cloud services can improve the application's scalability, availability, and fault tolerance. The system is resilient and scalable, capable of withstanding high traffic volumes and failures thanks to distributed databases, load balancing, auto scaling, and containers.

**Section D- Analysis of Big Data**

We will start by examining the volume. Data size and volume are correlated. Using this explanation, in my project, there are multiple ways that data is stored in large volumes.

Every user establishes a profile with personal information about themselves, including past and present travels, trips they've shown interest in, username etc. If the amount of profile data with thousands of users would be in the GB/TB range. For example, if my webpage had 10,000 users, each of their profiles could come up to 50KB per user.

With each user planning multiple trips, this trip data accumulates to hundreds of gigabytes as users browse and search through the specifics of scheduled trips locations and details and the ability to make and propose trips.

To scale and analyse the specific numbers and volume of data, we can use an azure data storage web app which gives us the opportunity to look at the incites of the website and its users and help us to collect the volume of data.

The velocity of the project. Seasonal travel patterns and daily usage cycles can have a substantial impact on the rate at which new data is generated in the application. Holiday seasons see significant spikes in travel planning activity, which can have an impact on the application's velocity. Weekends typically see more activity than weekdays as well. Even during normal days, the application will be more active around evening times compared to other times of the day, due to less people being at work and having more free time. Now and days, so many of us use the internet, these numbers are only rising. According to the next web[[1]](#footnote-6402) says 57% of the world’s population uses the internet and with that number only rising, our numbers will only grow bigger.

#### This leads to wildly fluctuating velocities of data generation. The number of new travel reservations made during the busiest holiday seasons can exceed 50 per second. Non-peak weekdays, on the other hand, might see 10x lower velocities, such as 5 trips/sec. As was previously mentioned, autoscaling servers, databases, caches, and other components to accommodate variable velocities is made simple by a Cloud platforms like Azure.

The third area we will be looking at is about the variety of big data application and how it relates to my travel friend application. The various forms of unstructured, semi-structured, and structured data that are being stored are what give data variety. Variety has to do with the kinds of data.

This programme creates and keeps data in a variety of formats. Structured data consists of information arranged in tabular relational formats, such as travel dates, locations, and so forth. The structured Mongoose database would house the essential trip booking data. Semi-structured data user profiles include JSON documents for expressed trips and other information in addition to some structured fields for passwords and usernames. Compared to simple tables, profiles require more flexibility. Structured data is the most common way of storing data as in big data analysis[[2]](#footnote-19619) “In our survey, a whopping 92% of respondents report handling structured data today.”

Veracity refers to my application as Trip details entered manually can contain inadvertent mistakes in dates, locations, etc. making aggregation unreliable. For example, if users spell some trips wrong, it could affect how data is read and sent, especially if its case sensitive. that is the reason we use Structured data to increase data accuracy because it needs validation checks on ranges, formats, cross-field sanity checks, and other factors to keep erroneous data out of downstream analytics.

**Infrastructure and software implications of Big Data**

**8 V’s**

**volume**

The first implication and impact we will be looking at is the volume storage of the application. large amounts of trip data and user profiles can be handled effectively by big data solutions like Azure VM Storage. However, when the amount of data increases, storage costs rise as well. To maximise expenses, careful management is required.

**velocity**

Managing the velocity of data in terms of network and processing requirements is made possible by cloud platforms like Azure. Based on demand, autoscaling features on cloud platforms automatically modify the number of resources allotted to a given application. This is especially helpful for fluidly managing variable data velocities. When an application experiences a spike in activity during peak times, like holidays or weekends, autoscaling enables the infrastructure to dynamically allocate more network bandwidth and processing power to handle the increased load.

Despite the advantages of scalability and autoscaling, managing network and processing requirements can be difficult, particularly during periods of high activity.

There will be a notable increase in user activity on the travel planning app during the holidays. Increased user interactions, travel planning, and messaging can put a strain on network bandwidth and processing power due to this abrupt spike in data velocities. This problem is made worse by the fact that user engagement tends to increase simultaneously and concentratedly during holidays.

It is imperative to guarantee the availability of sufficient resources during peak hours to tackle this challenge. This entails optimising the distribution of processing tasks, establishing suitable triggers for autoscaling, and continuously monitoring application performance.

**variety**

When combined with analytics tools like Azure provide support for effectively handling a variety of data kinds. These tools offer the capacity to examine and extract significant insights. But, processing various kinds of data are difficult, particularly when handling unstructured, semi-structured, and structured data at the same time. processing and analytics tools are necessary to guarantee that the application can extract meaningful data from the heterogeneous dataset.

**veracity**

veracity in Processing and Storage uses structured data with validation checks is one way to handle veracity. By increasing data accuracy, downstream analytics becomes more reliable.

The challenge lies in controlling noise and errors in user-generated content.

**Cost Implications:**

Pay-as-you-go models provided by cloud services enable cost optimisation based on actual usage.

The best cloud computing deployment model that I thought of is the hybrid cloud deployment.

This model maximises security, control, and scalability by combining the benefits of public and private clouds.

Private Cloud are the primary trip planning and scheduling systems, which manage sensitive user data and necessitate strict security protocols. This guarantees that the application's essential components are directly under control, enhancing security and guaranteeing adherence to privacy laws.

Public Cloud hosts the data lake and analytics systems. Using public cloud infrastructure, the application can access processing power that is scalable. Public cloud resources are dynamic and ideal for managing varying workloads, particularly during peak hours when large-scale data processing is needed.

1. Kemp, S. (2019) *Digital trends 2019: Every single stat you need to know about the internet*, *TNW | Contributors*. Available at: <https://thenextweb.com/news/digital-trends-2019-every-single-stat-you-need-to-know-about-the-internet> (Accessed: 17 January 2024). [↑](#footnote-ref-6402)
2. Russon, P. (no date) *Big Data Analytics - Tableau*, *Big Data Analysis*. Available at: <https://origin-tableau-www.tableau.com/sites/default/files/whitepapers/tdwi_bpreport_q411_big_data_analytics_tableau.pdf> (Accessed: 17 January 2024). [↑](#footnote-ref-19619)