

College of Science and Technology

School of Science and Technology

# COMP30231: Service-Centric & Cloud Comp 2022-23

*By*

*T0267278: Kai Hei Wong*

1. Self-evaluation
2. **Functional description**

In my program, it starts with a validation form for user to login to the client, while starting the server simultaneously, the user can login once the server recognizes a client with correct username and password, the user can also register a new account if he/she is use the program for the first time by entering the username, password and an ID generated by the web server. After that, user can login to the main page, by clicking the “Update” button, user can propose a new trip by input their location, user ID, interested trip, date and a trip ID generated by web server. Then the user needs to input their username for validation and press “OK” button to propose a new trip. Those inputted data are stored into local server and MYSQL database simultaneously for fault tolerance of the connection between the client and the web server.

The second function of the application is finding a trip buddy. In the main page, user can press “Find Trip Buddy” button to share their trip story and ask query for a trip. User can input the location where they are travelled, their ID and review of their trip, and click the “Share” button to share their review and impressions. Those inputted information are also stored into local server and MYSQL database simultaneously. Besides, user can also input the location that they want to travel and press “Ask” button to show the review about the location from other users.

Besides, user can press the “Check weather” button to see the weather forecast, in this page, user can check the temperature, weather and wind level on the upcoming week. By press the “Load” button, the console will display the result call from the web server. User can also use the combo box to choose the date and shows the weather on that day.

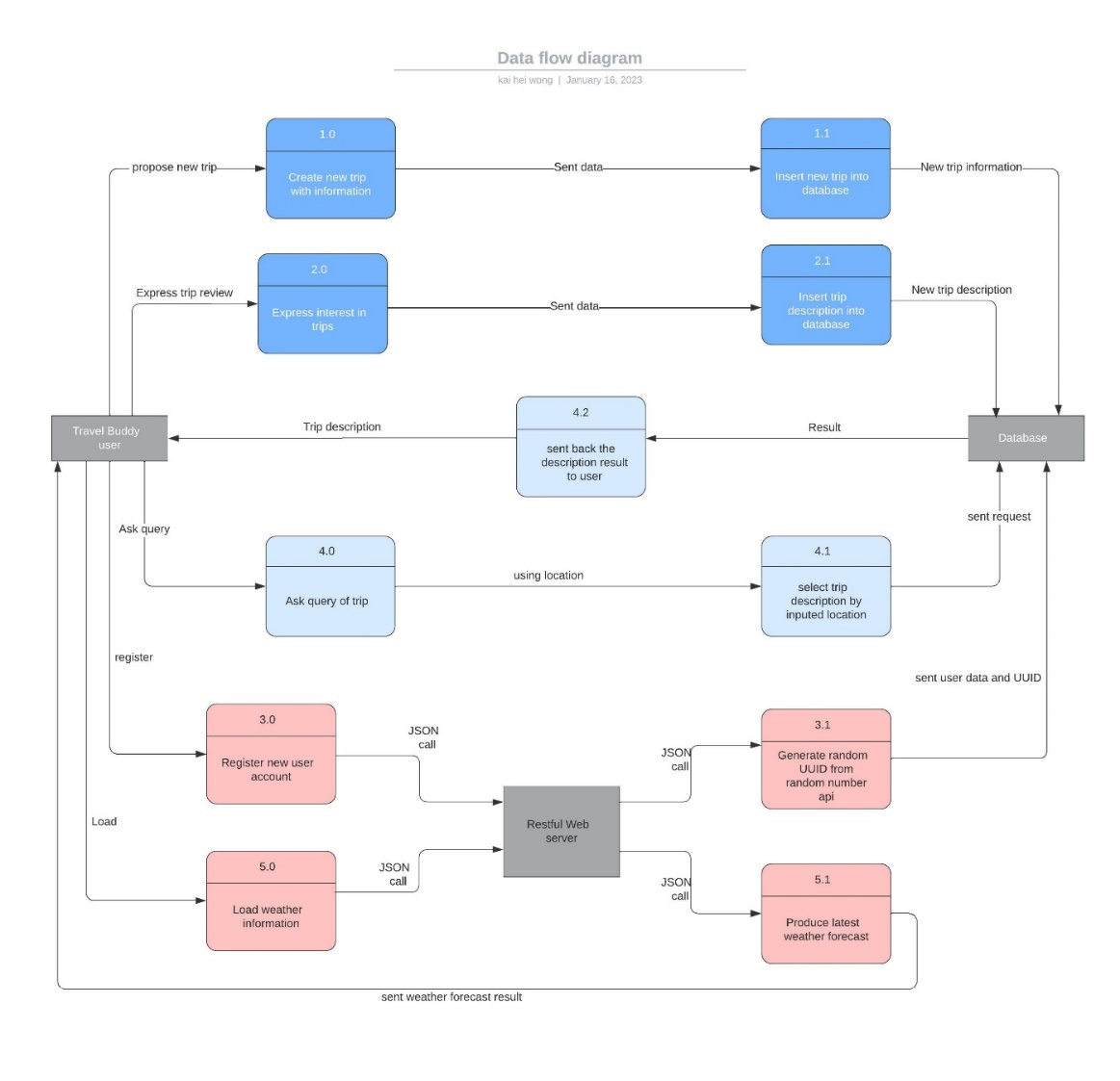
1. **Major design decisions**

In my application, the client side and web server are developed by Java. As Java contains various of libraries that can be utilized for GUI. Also, most of the lab materials are using Java language. It is more convenient for me to use Java to develop the client side. The user client connects to the local socket server and webserver simultaneously. The local server works as a local storage of user data such as username, password, ID and trip information. The reason to build a local server is prevent user cannot assess the data when the connection between client and webserver is disconnected or interrupt by error. Thus, the local server prevents fault tolerance and data loss.

The webserver is implemented in Tomcat server by using Microsoft Azure Virtual Machine. As most of the RESTful services are run by virtual machine in real life situation, using it to develop the webserver will have better demonstration.

The database is developed by Oracle MySQL, it is connected to the client side rather than webserver. Since my webserver was developed in NTU virtual machine and the database is developed in my laptop, they cannot connect each other physically. During the development, I face some error when developing webserver in my local machine. Thus, I use virtual machine to develop the webserver for better performance.

1. DFD diagram



1. **System analysis and design for cloud migration with QoS considerations**

A distributed cloud infrastructure is a network of multiple interconnected clouds that work together to provide a wide range of services to users. The deployment of a technical discussion on this type of infrastructure can bring many benefits, such as increased scalability, reliability, and availability. However, it also poses several challenges that must be addressed to ensure the quality of service (QoS) vital for Service-Oriented Architecture (SOA).

One of the main challenges of deploying a technical discussion on a distributed cloud infrastructure is ensuring the consistency and reliability of data across all the clouds. In a distributed system, there is no global clock, different activities can be synchronized. Clock synchronization aims to ensure that all nodes have the same interior clock, or the system is in sync with another exterior clock.

In a distributed system, each node possesses its local time by using local clocks and their time values may be different from each node. Thus, there is no global clock in the system that allows for the synchronization of various activities in the distributed environment. This can be achieved by implementing Cristian’s Algorithm. It is a clock synchronization algorithm used to synchronize time with a time server by client processes. The algorithm operates with low-latency networks, where Round Trip Time is shortened compared to accuracy when redundancy-prone distributed systems do not go hand in hand with this algorithm.

Another challenge is providing the required level of security to protect the data and communication between the clouds. This can be achieved by implementing security mechanisms such as encryption, authentication, and access control. There are some solutions below:

**An obsession with automation**: Automation does not only orchestrate and manage distributed environments efficiently but is also essential for gleaning meaningful security insights within the complexity of distributed environments.

**Dynamic baselining:** It means that your tools continuously and automatically reassess what forms normal, legitimate behavior, and what represents an anomaly that should be investigated.

**Dynamic firewall configurations:** Firewall rules should be configured prudently and updated automatically to keep pace with distributed environments while still helping to prevent intruders.

**Integration**: In distributed environments, security needs to be merged into every stage of the process that is used to deliver and manage applications for those environments. Developers must adopt security when designing and writing code, test engineers must comprehend security issues when testing it, and operations engineers must keep security considerations at the fore of their minds when deploying and redeploying services.

**Multi-layered security**: Security strategy also needs to comprise multiple approaches of different types. It’s not enough just to do runtime security, or just rely on firewalls.

High variability in bandwidth can lead to unpredictable and variable latency, which can negatively impact the performance and responsiveness of the application and services. There are a few solutions:

**Traffic Shaping**: Traffic shaping is a technique that is used to ensure that bandwidth is distributed fairly among different components of the infrastructure and application. This can be done by implementing Quality of Service (QoS) policies that prioritize different types of traffic and ensure that critical traffic is given priority over less important traffic.

**Load Balancing**: Load balancing is a technique that is used to distribute traffic and workload among multiple resources. This can help to reduce congestion and improve the performance of the application. Load balancing can be achieved using various methods such as round robin, least connections, and IP hash.

Interoperability is enabled specifically through the uniform application of design principles and standards. It establishes an environment where services produced by various projects at distinct times can be constantly merged and form a variety of composition configurations to utilize business tasks automatically.

Inherent interoperability symbolizes an essential purpose of service orientation that demonstrates a foundation for the realization of other strategic objectives and advantages.

Another challenge is dealing with the dynamic nature of the infrastructure, which can lead to changes in the network topology, the number of clouds, and the load on the system. To overcome this, it is necessary to implement dynamic resource allocation and management mechanisms that can adapt to changes in the environment. For example, load-balancing algorithms can be used to distribute the workload among the clouds and ensure that the system remains stable and responsive.

By using JMeter to test the latency of my web server, the latency time will increase with the number of users ascendingly. This phenomenon explains that when a vast number of users use the same service simultaneously, the response time will rise due to the crowding of data. There are a few solutions:

**Edge computing:** Edge computing is a distributed computing paradigm that brings computation and data storage closer to the location where it is needed, to improve response time and save bandwidth.

一張含有 文字, 螢幕擷取畫面, 監視器, 電腦 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 電腦, 室內 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 監視器, 電腦 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 電腦, 監視器 的圖片

自動產生的描述

Finally, it is necessary to ensure that the QoS requirements of the system are met. This can be achieved by monitoring the system and taking appropriate actions when the QoS thresholds are not met. For example, by implementing a monitoring system that can detect and diagnose performance issues, and by taking appropriate actions such as adding or removing resources or adjusting the configuration of the system.

In conclusion, deploying a technical discussion on a distributed cloud infrastructure can bring many benefits, but it also poses several challenges that must be addressed to ensure the quality of service vital for Service-Oriented Architecture. These challenges can be overcome by implementing appropriate data management, security, resource allocation, and monitoring mechanisms. By addressing these challenges, organizations can ensure that their systems are reliable, available, and performant, and can provide a high level of service to their customers.

1. **Analysis of Big Data scenarios and ways of mitigating them through cloud computing(**1,000 words**)**

Big Data is a term used to describe the large volume of structured and unstructured data that is generated and collected by organizations. This data can be used to gain insights and make better decisions, but it also poses several challenges for the infrastructure and software that support the applications that use it.

1. **Volume**

One of the main infrastructure implications of Big Data is the need for increased storage and processing capacity. As the volume of data grows, it becomes increasingly difficult to store and process it using traditional infrastructure. This can be addressed by using cloud computing deployment models that provide on-demand access to scalable storage and processing resources.

For example, using a public cloud deployment model, such as Microsoft Azure, allows organizations to scale their storage and processing resources as needed, without the need to invest in and maintain their own infrastructure.

In my application, user account data and trip information are stored in MySQL database and local server simultaneously. Normally MySQL has a 64TB storage volume, and the local server storage volume is depending on user devices, normally from 64GB to 2TB. While many users log in to the application, a vast amount of user and trip information will be stored in the MySQL database. Using relational database management systems(RDBMS) is more reasonable.

Volume is a critical aspect of big data in the travel industry. Travel Buddies application generates and collects vast amounts of data every day, such as user information, ID, customer reviews, weather forecasts, and social media posts. A restful travel buddy API service needs to be able to handle and process large volumes of data in order to provide accurate and personalized recommendations to users.

1. Veracity

       Veracity refers to the truthfulness or reliability of the data. Big data must not only be massive in size but also must be reliable to achieve value in its analysis of it. In a restful API, veracity is crucial as it impacts the accuracy and reliability of the API's output.

A restful API that utilizes big data needs to handle and process data with varying degrees of quality and accuracy. Since big data can come from a variety of sources, and not all data is accurate or reliable. For example, user-generated content such as reviews of trips or social media posts may contain inaccuracies or biases. Additionally, data can be incomplete or contain errors. If a restful API does not handle and process data with varying degrees of veracity precisely, it may provide inaccurate or unreliable output to the users.

To address this challenge, the restful API can provide data quality metrics to its users, such as rating scores for the comment or description, and allow the users to set their thresholds for data quality. This way, the users can make an informed decision on how to use the data provided by the API and ensure the veracity of the data.

In my application, the weather information is received from 7Timer through JSON, which is reliable as it is established by a true weather station rather than using random values to imitate a weather forecast.

1. **Visualization**

Visualization refers to the ability to represent data in a graphical or visual format. In a restful API, visualization plays an important role in making big data accessible and understandable for users. Visualization techniques, such as charts, graphs, and maps, are used to present data informatively and visually appealing.

In my application, the weather forecast is presented with a maximum and minimum value of temperature, wind level, and weather. It is informative but lacks maps or graphs to demonstrate the weather forecast from a different location. Thus, it would be better to add a map to the client GUI for a better understanding of the weather with various locations for users.

Another implication of Big Data is the need for improved data analytics and visualization tools. As the volume of data grows, it becomes increasingly difficult to analyze and make sense of it using traditional tools. This can be addressed by using cloud computing deployment models that provide built-in analytics and visualization tools. For example, using a cloud-based data warehouse such as Amazon Redshift or Google BigQuery allows organizations to easily analyze and visualize large amounts of data in real time.

A core task for any Big Data processing system is to transform its immense scale into something easily comprehended and actionable. For human consumption, one of the best methods for this is converting it into graphical formats.

A good visualization can enhance the user experience and improve the overall performance of the API by making it easier for users to extract insights and make decisions based on the data.

1. Velocity

Velocity refers to the speed at which data is generated and collected. In a restful API, velocity is an important aspect to consider as it impacts the API's ability to provide real-time, up-to-date information to users.

The speed at which the data is generated and processed to meet the demands and challenges that lie in the path of growth and development. Big data is often available in real time. Compared to small data, big data is produced more continually. Two kinds of velocity related to big data are the frequency of generation and the frequency of handling, recording, and publishing.

To address this challenge, a restful API can use various techniques to ensure high-speed data processing and analysis. One of the techniques is stream processing, which involves processing data as the data is generated rather than storing it first and then processing it later. Another technique is parallel processing, which involves using multiple processors or servers to process data simultaneously. Additionally, a restful API can use in-memory computing to process data faster by storing data in memory rather than on disk.

Furthermore, a restful API can also implement data caching and pre-fetching techniques to reduce the latency and improve the performance of the API. One of the solutions is using Message Queue, which allows for the asynchronous exchange of messages between applications and services. It acts as a buffer that stores messages until they can be processed, allowing for the decoupling of the sender and receiver.

Message queues are often used in distributed systems, where multiple applications and services need to communicate bi-directionally. They can provide several benefits such as allowing for asynchronous communication, which means that the sender and receiver do not have to be online simultaneously. It enhances the stability of message transportation and lets users receive a message once they are online, which achieves the goal of velocity.

In conclusion, Big Data poses several challenges for the infrastructure and software that support the applications that use it. These challenges can be tackled by using cloud computing deployment models that provide on-demand access to scalable storage and processing resources, data management and security, and data analytics and visualization tools. By using these models, organizations can manage and analyze their large volumes of data and gain valuable insights easily and cost-effectively.

1. Reference:

Kitchin, R. and McArdle, G. (2016) What makes Big Data, Big Data? exploring the ontological ..., What makes Big Data, Big Data? Exploring the ontological characteristics of 26 datasets. SAGE journals. Available at: https://journals.sagepub.com/doi/10.1177/2053951716631130

Mohammed S.HadiPersonEnvelopeAhmed Q.LaweyEnvelopeTaisir E.H.El-GorashiEnvelopeJaafar M.H.ElmirghaniEnvelope et al. (2018) Big Data Analytics for Wireless and Wired Network Design: A survey, Computer Networks. Elsevier. Available at: https://www.sciencedirect.com/science/article/pii/S1389128618300239