|  |
| --- |
| Nottingham Trent University |
| Coursework Report |
| Service-Centric & Cloud Computing | COMP30231 |

|  |
| --- |
| Hassaan Naveed | N0898071 |



# Section A & B - Self Evaluation

I believe my implementation of my RESTful web service and client application has been overall successful, as I have completed the majority of the primary required functionality, as well as added some extra features.

Diagram

Description automatically generatedThis diagram illustrates the architectrue of my application. The User Client, which has been programmed in Java and C#, is able to send and receive data from the Web Service, which is programmed in Java. This is able to send and receive data from the RabbitMQ server, as well as the World Weather Online and Random.org API’s.

The User Client offers a clean, user friendly GUI where the user is able to select a number of buttons relating to each of the required functions within Task A. The user is unable to send any data to the server without generating a user ID first. This is due to the fact that the user ID is required when sending data, so the user should be prompted to generate an ID. When the user clicks on the generate ID button, a HTTP GET requets is sent to the web service, which is locally hosted on Apache Tomcat. The service then sends a GET request to the random.org API, requesting a random integer. This number is then sent back to the client, and is assigned as the user ID. All data transfer is done within JSON format.

Once the user has generated their ID, they are able to access the functions of sending and querying trips and intents. When a user decides to send a trip request, a new GUI window opens containing text fields to fill out information about the location, coordinates, and date. The user is unable to leave any text fields blank, as all information is essential. The user is also unable to input a date that is more than 14 days in the future, or in the past. This is because the weather forecasting service can only display a 14 day forecast, and trips cannot be made in the past. When they submit the proposal, their inputs are converted into a JSON format string, and a message ID is generated in the same way as the user ID. A HTTP PUT request is then sent to the web service, containing the JSON data in its body. When the service recieves the data, it immediately sends it to the RabbitMQ server, using the “TRAVEL\_OFFERS” exchange and the topic exchange type, through a producer class. A separate consumer class in the background is monitoring for any data that comes through the exchange with the correct topic key name. When it recieves the data that has been sent, it stores it within a list in the program. When the user wishes to query the trip offers that have been made, they are able to click on the query proposals button, which sends a HTTP GET request to the web service. For each item within the proposals list, the service makes a GET request to the world weather online API, in order to retrieve the weather data at the given location name or coordinates within that proposal. It appends that weather data to the proposal, and sends back one JSON structure containing all of the proposals, with their weather data appended to them. These are then deserialised using Gson and displayed on the client GUI in a user friendly, readable manner.

When the user wishes to send an intent to go on a trip, they are able to click on the send intent button. This brings up a new GUI window, that only contains a drop down menu. This menu will be populated with all of the message ID’s for all of the trip proposals that have been queried. The user is able to select on a message ID, and send an intent for that trip. This acts similarly to the trip proposals. A HTTP PUT request is sent to the web service with the message ID, the user ID of the owner of the trip, and the user ID of the responder within a JSON body. When this is received by the web service, It is immediately sent to the RabbiTMQ server through the “TRAVEL\_INTENTS” exchange, and then received by the consumer class, which places it within a list. When the client wishes to see if their trip proposal has had any responses, they are able to click on the query intents button. This sends a GET request to the web service, which puts all of the intents within the list into a single JSON object, which is sent to the client. This is deserialised using Gson, and displayed in a readable manner on the GUI.

In order to demonstrate the platform independence of Service Oriented Architecture, I have also decided to create an additional user client using the C# programming language. This client offers the exact same functionality as the Java client, and works in the same way as illustrated above. It is able to successfully send and receive data, to the server, including data which has been sent by the Java client.

Overall, I feel as though I have been successful in my implementation. All of the required functionality within Task A has been added. The service is RESTful, amd runs on a local Tomcat server. It allows clients to perform the five major actions, generate an ID, query for trip and intents, and publish trips and intents, as REST methods. All data communication is done in JSON, and the service connects to RabbitMQ to send and receive user messages. Within task B, the service successfully connects to and consumes both listed external APIs; World Weather Online and Random.org. The client is able to store trip proposals and intents within internal lists, and provides a user friendly GUI. The client has been created in multiple programming languages to demonstrate platform independence. Multiple, concurrent clients are able to communicate with each other whils running at the same time. The only area where my implementation is lacking is that it is unable to locally cache results from the external APIs, which could lead to unnecessary calls.

# Section C - Analysis of QoS

Cloud Computing is the on-demand delivery of any hosted services over the internet (Hayes, 2008). A cloud typically consists of a datacentre of multiple, distributed servers hosted by an organisation. This organisation can sell the storage, computing power, environments, and cloud-based applications on a pay-per-use basis (Forbes, 2014). Cloud computing has recently emerged as a dominant force in the industry, as many companies have moved to cloud based services. For example, Microsoft provides the Azure service for a variety of development tools and environments, as well as moving their Office Applications to the cloud (Microsoft, 2010). This allows for convenient methods of acquiring computing power, operating systems, and applications, that are often cheaper than local alternatives.

With the rise of Cloud Computing and Web Services, has come Service Oriented Architecture. This is a design architecture based on combining multiple web services within a single application (Lee, 2010). Quality of Service is an important factor in any SOA service. As the demand and userbase of a service increases, the service should be able to efficiently keep up with the demand and provide a reliable, fast service, and provide scalability.

In order to test the QoS of my application I decided to use JMeter. When testing a small amount of clients the application is able to perform well, and transfer data within almost instantaneous amounts of time. In an ideal scenario, this would translate to a high number of concurrent users. I put the application through a stress-test of 1000 concurrent users in order to identify whether the application performed well under heavy load. I performed two tests, a HTTP PUT request, and a HTTP GET request.

Chart, line chart

Description automatically generated

The above graph shows the stress test results for the PUT request. The graph shows that overall, there is little variation within the latency. However, the latency rises at a very drastic rate. As the number of requests increases over time, the latency on average increases at the same rate, with large deviations throughout, and at the end of the test. The throughput of the service is 361 requests per minute, and the highest latency is abover 100,000 milliseconds. This is fairly poor performance, as with a large number of concurrent user, an individual user could be waiting potentially minutes for their request to go through.

Chart, line chart

Description automatically generated

We can also see a similar same pattern in the GET request, although the average rise is not as drastic as the results for the PUT request, however, there is muich higher variation. Initially, there is a small, steady rise in latency. It then alternates between rising steadily and drasticallty. On average, it rises at a slower pace than the PUT request, and reaches a maximum latency of 7000 milliseconds.

Overall, the scalability and QoS of the application is poor. When there is a large amount of concurrent users, the amount of time it takes to complete a HTTP request is significantly long. This would result in a poor user experience as users could be suffering from long waiting times for their request to go through, and the data they have inputted to be added to the message queue.

Cloud computing could assist in solving the latency and response time issues as the web service could be hosted on a virtualised machine within the cloud. There are several models of cloud computing. Public, Private, and Hybrid (Savu, 2011). The public cloud is typically owned organisations, and provide access to a variety of services. These services can be bought by anyone, who pay for what they use. They are accessed over a network connection. The private cloud is typically created by an organisation for personal use within their own datacentre, and the organisation is responsible for the services it provides. The hybrid clouds is a combination of public and private, alolowing applications to run in the optimal locations.

Running the web service on the cloud would mean it would have access to greater computing power than a local machine, and more power could easily be purchased when required. Additionally, high volumes of data could also be accounted for, and more data would be able to be purchased as it is being used. This would greatly assist in reducing the latency of the requests, as more requests could be handled faster. Additionally, the computing could be distributed to occur in parallel, so multiple requests could be handled at the same time, further reducing response times. Finally, the service could be hosted in multiple geographical locations with ease, in order to reduce response times of users who are situated in countries that are not close to the original datacentre. With Cloud Computing, this can all be completed easily, and with a lower cost than setting up multiple, powerful datacentres arouind the world and ensuring they are future-proofed. Therefore, the hybrid or public clouds would be the most optimal solution, as computing power and storage could be purchased from organisations, and virtualised operating systems for the service could easily be set up in multiple locations.

# Section D - Analysis of Big Data Scenarios and Mitigation

Big data is a large collection of structured, semi-structured, and unstructured data that is collected by organizations and can be utilised for extracting certain information (Sagiroglu & Sinanc, 2013). It has become integral to the ideas of machine learning, predictive forecasting, product development, as well as many other applications (Buhl, et al., 2013). For example, Big Data is used in many predictive applications such as forecasting the weather based on historical data, as well as data from weather sensors. It can also be used in business, to identify what kinds of products are successful based on a variety of factors such as demographics and the time of the year.

Big Data is the idea of disparate data that contains greater variety, arriving in increasing volume and with higher velocity (Oracle, n.d.). These are the original 3 of the 8 V’s that typically characterise Big Data. In regards to my application, I will be discussing 4 of the 8 Vs. Volume, Velocity, Variety, and Value.

Volume refers to the amount of data that is generated (Sagiroglu & Sinanc, 2013). The primary factor affecting the volume of data within my application would be the number of users. As the userbase of the application continues to grow, the amount of data being sent to the web service would increase. Additionally, secondary factors would also influence the volume of data that would arrive, such as the time of the year. It would be expected that during the Spring and Summer seasons, a higher volume of data would be sent that during Autumn and Winter, as a higher amount of people would be wanting to travel during warmer climates, and so more requests may be sent to the application. An issue that arises with increasing volumes of data is the cost of storage (Katal, et al., 2013). As the volume of data generated increases, as does the required capacity to store that data. A solution to this issue would be to employ cloud computing in order to store all of the generated data on the cloud, within a virtualised environment. A public cloud would be optimal in this scenario, as initially, it would be difficult to estimate the amount of data that would be generated. Therefore with utilising a public cloud service, only the storage capacity that is used would need to be payed for. Once the growth of the application is able to be projected, a hybrid cloud could be used to provide a greater degree of control and flexibility.

Velocity refers to how quickly the data is generated and how fast it moves. It is important for data to flow as quickly as possible, so it can be analysed within a short amount of time (Katal, et al., 2013). This allows for the data to be readilty available when it is needed. This is critical in time sensetive situations, such as in a medical or business setting, in order to make the best decisions possible (Sagiroglu & Sinanc, 2013). A limitation of my application concerning data velocity is that as the volume of data that is transferred to the web service increases, the time is takes for that data to arrive and be processed will decrease (Kaisler, et al., 2013). Through the use of RabbitMQ Message Queues, data is received one-by-one through the queue, which can be a severe bottleneck when workin with large sets of data. A solution to this problem would be to utilise cloud computing to employ parallel processing of the incoming data. Multiple, virtualised machines could be used together as a distributed system in order to speed up processing times. This allows for faster, more accurate data displayed to the user. Additonally, another issue with increasing velocity is that security could be potentially compromised . With a high flow of data, malicious payloads could be hidden in order to gain unauthorised access to the system (Gewlrtz, 2018). Cloud computing can assist with this as many public clouds such as Azure and Amazon Web Services will be highly secure, and have robust protection to ensure that malicious data can be caught without significantly affecting the processing time. A final issue with data velocity would be latency. If all the data and the web services are hosted in a single geographic region of the world, a greater amount of time would be required to send that data to regions that are further away. A solution to this would be to employ cloud computing to host the data in multiple regions of the world. This way, users could connect to the server that is geographically closest to them, in order to speed up the velocity of the data. The public cloud would be an optimal solution for this, as space in preexisting datacentres could be purchased easily.

Variety refers to the diversity of the data (Katal, et al., 2013). Data can be captured and collected using a variety of methods, and come in a variety of different forms and formats. For example, within my application, data is gathered from user input, as well as external API’s such as weather forecasts. This data is structured, as it is gathered from human-generated inputs and sensors. Additionally, the application also makes use of semi-strcutured data, as all data communication is done in JSON format. With Big Data, a greater variety of data sources could also be used. For example, the application does not make use of unstructured data. The time of year could be captured, in order to identify when the most amount of data is sent to the web service, or analyzing a variety of sources such as social media to identify popular locations for travel, to use for advertisement. Cloud computing can be used in this scenario through the machine learning and data analysis tools that are offered in public services such as Azure.

Value refers to what an organisation can do with the collected data, how useful it is, and what insights can be gained from it (Kaisler, et al., 2013). This is an important factor in Big Data, as one of the primary advantages of big data is being able to identify new patterns or trends to help facilitate decisions. For example, with my application, the data that is collected can become valuable as more data is generated over the course of time. The data can be analyzed to show new trends emerging within where and when people travel. As new travel location become more popular, and others’ become less popular, the data can show these trends and effective business decisions can be made, such as advertisement and ticket pricing. One issue with data value is that as other characteristics of Big Data increase, such as the aforementioned Volume and Velocity, the value of individual data entries can decrease (Katal, et al., 2013). This can be mitigated with cloud computing as many cloud platforms such as Azure and AWS are creating more tools and services that introduce powerful data analytics, which can assist in bringing value back to individual data entries.

# References

Buhl, H. U., Röglinger, M., Moser, F. & Heidemann, J., 2013. Big Data. *Business & Information Systems Engineering,* Volume 5, p. pages65–69.

Forbes, 2014. *Where's The Rub: Cloud Computing's Hidden Costs.* [Online]   
Available at: https://www.forbes.com/sites/centurylink/2014/02/27/wheres-the-rub-cloud-computings-hidden-costs/  
[Accessed 03 02 2021].

Gewlrtz, D., 2018. *Volume, velocity, and variety: Understanding the three V's of big data.* [Online]   
Available at: https://www.zdnet.com/article/volume-velocity-and-variety-understanding-the-three-vs-of-big-data/  
[Accessed 03 02 2022].

Hayes, B., 2008. Cloud computing. *Communications of the ACM,* 51(7), pp. 9-11.

Kaisler, S., Armour, F., Espinosa, A. J. & Money, W., 2013. Big Data: Issues and Challenges Moving Forward. *2013 46th Hawaii International Conference on System Sciences,* pp. 995-1004.

Katal, A., Wazid, M. & Goudar, R. H., 2013. Big data: Issues, challenges, tools and Good practices. *2013 Sixth International Conference on Contemporary Computing (IC3),* pp. 404-409.

Lee, Y., 2010. QoS metrics for service level measurement for SOA environment. *2010 6th International Conference on Advanced Information Management and Service (IMS),* pp. 509-514.

Microsoft, 2010. *The Official Microsoft Blog.* [Online]   
Available at: https://web.archive.org/web/20140511230956/http://blogs.technet.com/b/microsoft\_blog/archive/2010/02/01/windows-azure-general-availability.aspx  
[Accessed 03 02 2022].

Oracle, n.d. *What is Big Data?.* [Online]   
Available at: https://www.oracle.com/uk/big-data/what-is-big-data/#best-practices  
[Accessed 02 03 2022].

Sagiroglu, S. & Sinanc, D., 2013. Big data: A review. *2013 International Conference on Collaboration Technologies and Systems (CTS),* pp. 42-47.

Savu, L., 2011. Cloud Computing: Deployment Models, Delivery Models, Risks and Research Challenges. *2011 International Conference on Computer and Management (CAMAN),* pp. 1-4.