6G6Z0037 Cloud Computing Report

1CWK100

Student Name: Husnain Ahmed

Student ID: 21308666

Contents

[What is Cloud Computing? 3](#_Toc195202487)

[Analysing Security Challenges in Cloud Computing: Best Practices and Mitigations 8](#_Toc195202488)

[Cloud Infrastructure vs. Physical Infrastructure 11](#_Toc195202489)

[Your Implementation 12](#_Toc195202490)

[Discussion 15](#_Toc195202491)

[Conclusion 17](#_Toc195202492)

[References 18](#_Toc195202493)

[Appendices 20](#_Toc195202494)

# What is Cloud Computing?

Cloud Computing is where any services are delivered over the internet, all resources whether they be storage, processing power or software can be run remotely.

It removes the need for anyone, whether an individual or a business to require on site storage and processing capability. It works by “loaning out” the use of computer and server farms remotely over the internet to anyone who pays for it.

The key features of cloud computing:

* Scalability – The ability to alter the amount of resources used according to demand.
* Resource Sharing – The cloud resources can be shared amongst many users which can help save money and be more efficient with resource use.
* Metered Access – The ability to track the resource usage accurately and charge the user for the specific amount of resources they have accessed.
* Elasticity – The ability to adjust the resources based on demand, for example if an app that uses cloud services for processing power had higher user traffic at certain times of the day and lower user traffic at other times, the resources used will scale up and down accordingly.

The different types of cloud computing:

* Deployment Models:
* Public Cloud – Resources that are available to anyone without any authorisation needed.
* Private Cloud – Resources that belong to a private organisation or individual, grants greater security and control.
* Hybrid Cloud – A combination of public and private cloud resources, has the benefits of both such as the cost efficiency of public and the security of private.
* Community Cloud – Resources that are shared by multiple linked organisations with a common goal and security requirements.
* Multi Cloud – Resources that are made up of more than one service from more than one vendor. All hybrid clouds are multi clouds but not all multi clouds are hybrid clouds.
* Service Models:
* Software as a service (SaaS) – Where a service that delivers a software or application is provided, this is managed by the cloud service provider.
* This excels in mobile apps that users can access through a browser such as email services, Gmail is a good example of this.
* Infrastructure as a service (IaaS) – Where the infrastructure is provided to you as a service, there is no need to manage the servers, network or storage.
  + - This is best used to host websites; it is also used by cloud storage providers such as Dropbox or Google drive. An example of this is AWS Elastic beanstalk.
* Platform as a service (PaaS) – Where the user is provided with a software platform that is managed by the cloud service provider, used to build and deploy apps.
  + - This is mainly used by developers to run apps. Dev-ops engineers also use the shared cloud platform to manage apps.

Each type of service model excels in different circumstances:

* + SaaS is the easiest to use as almost everything is provided to the user and managed by the cloud provider, but it offers the least amount of control over the service.
  + IaaS is the most difficult to use as almost everything is managed by the user, this offers the most control and freedom at the cost of added complexity. It is best used for complicated tasks that need to be tailored to specific use cases.
  + PaaS is a good middle ground between these two, while the infrastructure is managed by the cloud service provides, the user is required to manage their applications and data. It is a best used to develop apps.

The Pros and Cons of Cloud Computing:

* Pros:
  + Reduced infrastructure costs
    - There is no need to spend money on purchasing new servers up front when you can rent the usage of one.
  + Data consolidation
    - When all your data is in the cloud, you know where it is at all times, the cloud service providers also maintain the security of your data.
  + Scalability
    - When the need arises, you can easily purchase more or discard the existing resources based on demand.
  + Enhanced Collaboration
    - Group work has never been easier than with the cloud, multiple users can all work on the same task or document at once.
  + Accessibility
    - The servers can be accessed from anywhere as long as you have an internet connection, this is great for remote workers.
* Cons:
  + Security concerns
    - There is the potential for data breaches or unauthorised access of your data, as you do not own the cloud server your data is on, you give up control to unknown third parties.
  + Migration data from on premises to cloud
    - Moving between in-house and cloud servers is usually easy, however moving between different cloud providers or going back to in-house servers is not the same. This is because of vendor lock-in and switching storage methods can be quite complex and costly.
  + Slower backups and restores
    - Moving data to and from the cloud may come with some delays because of network latency, this process will take longer than an in-house server and can be affected by the amount of people on your network at any given time.
  + Internet reliance
    - If your internet is down, you won’t be able to access your data.

Comparison of Different Cloud Providers:

* AWS:
  + The largest cloud provider, excels in mobile development, cybersecurity and networking.
  + Has lots of community support and documentation, this makes it the most reliable of the 3.
  + Can be hard to learn due to the many services offered (EC2, S3, RDS, etc.).
  + Can get expensive if not optimised.
* Azure:
  + AWS’ main rival, has the largest data centre network and is the first choice for organisations that already make use of Microsoft products like office.
  + Enterprise-focused.
  + Notable services include Compute (VM’s), Blob (Storage), Serverless (Azure functions) and Azure machine learning.
  + Suffers from the same pricing issue as AWS but can be cheaper if the user already has a Microsoft license.
* GCP:
  + The best for managing big data, machine learning and AI.

* + Easy interaction with other cloud native apps like Kubernetes.
  + Has the least services available, most notable are the Compute engine and BigQuery.
  + The cheapest of the 3 with sustained use discounts.

I have personal experience with AWS for this assignment, I benefited from the extensive documentation and online aid for learning how to use it. I believe it is the best choice for beginners.

# Analysing Security Challenges in Cloud Computing: Best Practices and Mitigations

With the increasing reliance on the cloud comes ever increasing security risks from malicious third parties. These could be hackers, rival companies or data brokers looking to sell your data. To combat this risk, we must prepare some contingencies for the common security risks.  
  
The main security issues are:

* Data Breaches:
  + Sensitive data in the cloud is a big target for cyberattacks, any breaches could result in sensitive information being made public.
* Unauthorised Access:
  + The increased accessibility of the cloud inadvertently provides a greater risk of unauthorised access through weak login details.
* Insider Threats:
  + Whether intentional or not, misuse of cloud access can compromise security.
* Shared Technology Vulnerabilities:
  + If one system is compromised, others that are connected to it may also become compromised.

Certain types of cloud deployment models are more vulnerable to some of these:

* Public Cloud:
  + Due to its shared nature, data breaches and security misconfigurations are more likely to occur.
* Private Cloud:
  + The enhanced control comes with higher costs and a requirement for in depth management to guard against threats.
* Hybrid Cloud:
  + The added complexity of managing both public and private cloud environments increases risk, data transfer between these sections is also riskier.
* Multi Cloud:
  + The usage of many different cloud providers, can lead to confusion with how to manage the differing security policies, these gaps can be exploited.

The best practices to mitigate these concerns are:

* Encryption:
  + Data should be encrypted, both in transit and at rest to ensure sensitive information is protected even if intercepted.
* Zero Trust Security:
  + This assumes that no user of device is trusted by default without continuous verification.
    - * The cloud should be divided into micro segments, limiting user access to only the select portions of the cloud that are relevant to them, moving between these should be accompanied by constant verification.
* Regular Security Audits:
  + Audits should be used to detect security vulnerabilities and detect suspicious activities or threats.
* Ensuring Regulatory Compliance:
  + Processes should be put in place to ensure that regulations such as General Data Protection Regulation (GDPR) and Health Insurance Portability and Accountability Act (HIPAA) are complied with.
* Access Control:
  + Strict control measures should be enforced such as Multi Factor Authentication (MFA), Identity and Access Management (IAM) systems and Role-Based Access Control (RBAC) to ensure users only access relevant data and services.

In addition, a well-defined contingency plan should be prepared for the event that an incident or security breach does occur. This should outline the process of detecting, containing and recovering from a security breach.

By understanding and preparing for these security issues organisations can ensure that their data and systems in the cloud are well protected.

# Cloud Infrastructure vs. Physical Infrastructure

As the CTO for the start-up company providing a social media platform, I recommend that the company make use of cloud services for our data storage and hosting needs as opposed to buying the required physical server infrastructure.

While a Physical Server may provide additional security, better control over the infrastructure and may even save money in the long run. It comes with high upfront costs not suitable for such a new company, as well as challenges with upscaling the server infrastructure when the user base grows in the future and generates more data.

A Cloud Server on the other hand is not negatively impacted by the limited starting capital our company has access to as we only have to buy the required resources we consume, and eventually when we grow our userbase to the point that we need to upscale our server we will be making more money from our app. The fact that cloud services operate on a pay as you go model only stands to benefit us.

While it is expected that our app will keep growing social media platforms are very unpredictable. Many once popular sites have faded into obscurity. The capability to upscale and downscale our resource use based on any changes that happen is very important because of this. Physical infrastructure is too much of an investment for such a risky venture.

Maintaining a physical server will also be a heavy burden on such a small company with the expected high volume of traffic, whilst cloud servers are maintained by the cloud provider, so we don’t need to worry about that and can instead focus on developing our prototype app.

Deploying our app will be much faster on the cloud as we can quickly purchase access to more servers and infrastructure or stronger processing power whenever we need while doing the same for a physical server requires it to be planned out in advance and ordered.

Cloud Providers also offer very strong security and robust infrastructure that has built in redundancy and disaster recovery abilities. This ensures that the risk of downtime is minimal, and our users will be happy with their experience.

In conclusion, while physical servers may provide better control over our data in the long-term, for our current circumstances as a start-up cloud servers are the best option.

# Your Implementation

In this section I will explain the tools and technologies I used to deploy the ‘Contacts web application’, which cloud services I use and why I did so.

The tools and technologies I used:

* Ubuntu 24.04 LTS:
  + This was the virtual machines operating system that I used, it is a Linux distribution well known for its stability and widespread community support and its compatibility with java apps.
* Java Development Kit:
  + This was used to run the java web application file that was given, it does an excellent job of compiling and executing java code from .WAR files.
* Tomcat 9:
  + This was the web server I used to deploy the .WAR file, it is well known for its security, scalability, performance, community support and the fact that it is opensource which makes it very trustworthy.
* MySQL 8:
  + This was the database management tool that I made use of, I ran MySQL Workbench 8.0 CE locally on my laptop, I followed best practices and used a cloud-based MySQL service (RDS) as opposed to installing a MySQL database locally on the same VM instance.

The cloud services I used and why:

* Amazon Web Services (AWS):
  + This was my cloud service provider of choice, it is well known for its mature ecosystem, extensive documentation and community support as well as the wide range of services it has on offer.
  + Through my usage of this I have shown industry-standard cloud infrastructure and database management practices.
* Amazon’s EC2:
  + This was my virtual machine service of choice, it is well known for its scalable processing power which allows for flexibility in choosing the relevant instance type according to the users needs.
  + I used this to host my web application, I used a basic instance type with minimal processing power as seen in Appendices A. This was sufficient for the small-scale web app which was not very resource intensive.
* AWS Aurora and RDS:
  + This was my cloud based database management tool of choice, it is well known for simplifying database management tasks such as setup, patching, backup, and recovery.
  + Using this as opposed to installing a MySQL database locally on the EC2 instance increases the potential for scalability in the future if needed.
  + I used this to create a remote database for me to connect to later on with MySQL Workbench as seen in Appendices D and E.

My appendices on page 20 and onwords show the screenshots of my deployment process. Below is an explanation of the steps I underwent to deploy the app:

* Setting up an EC2 instance:
  + I launched an EC2 instance configured with Ubuntu 24.04 LTS.
  + Appendices A.
* Deploying an Apache Tomcat 9 web server:
  + I deployed the application’s .WAR file to the web server.
  + Appendices B and C.
* Creating a RDS database:
  + I setup a MySQL database instance on RDS.
  + Appendices D and E.
* Connecting to the database with MySQL Workbench:
  + I connected to the database through MySQL Workbench.
  + Appendices F.
* Using a script to create data in the database:
  + I created tables and data in the database using a pre-created script.
  + Appendices G.
* Configuring the web app to use the database:
  + I connected the web app to the database so that it could make use of the data within it and show it on the screen.
  + Appendices H.
* Testing that the web app works as intended:
  + I tested the functioning web application for creating new data and altering existing data.
  + Appendices I.

These steps and the accompanying screenshots demonstrate the successful deployment of the ‘Contacts web application’ to AWS cloud services and the use of CRUD operations to modify the data in the database.

# Discussion

If I were to develop a similar application to the ‘Contacts web app’ using alternative web services, I would approach the development by focusing on microservices architecture so that my app has the potential for future scalability.

This is a more modern approach than a monolithic application and is especially well suited to the use of the cloud. It also excels when using serverless functions.

For the Applications infrastructure I would use:

* AWS Lightsail / AWS Elastic Beanstalk:
  + Instead of AWS EC2, Lightsail is a simplified IaaS which specialises in developing and deploying small scale apps quickly, though at the cost of scalability.
  + Another alternative to this is AWS Elastic Beanstalk, a PaaS service which removes the need to manage the infrastructure manually and could be used instead if we plan to upscale in the future.

For the data storage I would use:

* AWS S3:
  + Instead of AWS RDS to store and manage my data, though S3 is not a relational database management tool and is instead used to store objects, it should suffice for this project.
* Google’s Firebase:
  + Another alternative to RDS, this is a backend-as-a-service (BaaS) platform that provides a real time database and authentication features, though this is a NoSQL platform so I would need to approach the development of the web application slightly different than the original MySQL based app.

For the back-end logic and API I would use a mixture of both:

* AWS Lambda:
  + Instead of the provided .WAR file to replicate the CRUD operations for creating and manipulating data.
  + An individual lambda function could be created for each of the CRUD operations (Create, Read, Update, Delete).
* Amazon API Gateway:
  + This would serve as the front end for the serverless functions and would take care of things like authentication, routing and request / response transformation. It is crucial in managing API requests in the microservices architecture.

In conclusion, I would use either PaaS (Elastic Beanstalk) or IaaS (Lightsail) solutions for supporting services, S3 bucket or Firebase for data storage and serverless functions through AWS Lambda for the core functionality. This would allow me to deploy the web application through alternative methods.

# Conclusion

In conclusion, the usage of cloud computing has both its merits and detriments. This report has outlined the different types of cloud computing deployment models, the difference between private clouds, public clouds and all in between.

The different service models: SaaS, IaaS and PaaS were evaluated and their core features and most compatible situations were assessed.

The key features of cloud computing have been explained and the pros and cons of cloud computing have been weighed, as well as provided an analysis of the 3 main cloud service providers (AWS, GCP, Azure) and what they have on offer.

The security concerns that are inherent to cloud environments were analysed and some recommendations on how to best mitigate these concerns were given.

The decision between the choice of physical server usage and cloud server usage for the purposes of a small startup company was weighed in and it was decided that the cloud was a better option due to its scalability, cost effectiveness and the provision of third party maintenance.

My deployment of a simple java web application to AWS was shown, there I have explained the steps I took to deploy the app with services like Amazon’s RDS to store data and EC2 for virtual machine usage and hosting.

I made use of tools and technologies like Ubuntu, Tomcat, JDK and MySQL in this process and showed how they can be used to navigate and communicate with the cloud.

In this project, I have gained a deeper understanding of cloud computing, it’s uses and how it can be used to deploy and host apps. This report has covered cloud computing concepts, security and infrastructure choices, as well as compared cloud providers. The deployment of the web application to AWS using EC2 and RDS has demonstrated infrastructure management, service selection and other key features of cloud computing.

# References

1. Google (2024) What Is Cloud Computing? Google Cloud. [Online] [Accessed on 1sth April 2025] <https://cloud.google.com/learn/what-is-cloud-computing>
2. Redhat (2022) Types of cloud computing. www.redhat.com. [Online] [Accessed on 1st April 2025] <https://www.redhat.com/en/topics/cloud-computing/public-cloud-vs-private-cloud-and-hybrid-cloud>
3. Morefield Communications (2022) What are the Pros & Cons of Cloud Computing? Morefield. [Online] [Accessed on 1st April 2025] <https://morefield.com/blog/pros-and-cons-of-cloud-computing>
4. Coursera (2022) What’s the Difference Between AWS vs. Azure vs. Google Cloud? Coursera. [Online] [Accessed on 3rd April 2025] <https://www.coursera.org/articles/aws-vs-azure-vs-google-cloud>
5. Shaw, B. (2024) What Is Cloud Security? | CrowdStrike. Crowdstrike.com. [Online] [Accessed on 3rd April 2025] <https://www.crowdstrike.com/en-us/cybersecurity-101/cloud-security/>
6. Eventura (2024) Cloud Security: Challenges and Best Practices. Eventura. [Online] [Accessed on 3rd April 2025] <https://eventura.com/blog/cloud-it-services/cloud-security-challenges-and-best-practices/>
7. How does cloud security work? | Cloud computing security (2024) Cloudflare.com. [Online] [Accessed on 4th April 2025] <https://www.cloudflare.com/en-gb/learning/cloud/what-is-cloud-security/>
8. Amazon Lightsail, AWS Elastic Beanstalk, or Amazon EC2? - Amazon Lightsail, AWS Elastic Beanstalk, or Amazon EC2? (2025) Amazon.com. [Online] [Accessed on 5th April 2025] <https://docs.aws.amazon.com/decision-guides/latest/lightsail-elastic-beanstalk-ec2/lightsail-elastic-beanstalk-ec2.html>
9. AWS vs GCP vs Azure: Which Cloud Platform is Best for your Business - Qovery (2024) Qovery.com. [Online] [Accessed on 6th April 2025] <https://www.qovery.com/blog/aws-vs-gcp-vs-azure/>
10. What is Amazon Glacier? (n.d.) SearchAWS. [Online] [Accessed on 8th April 2025] <https://www.techtarget.com/searchaws/definition/Glacier-Amazon-Glacier>
11. Amazon S3 vs RDS: 5 Critical Differences - Learn | Hevo (2021). [Online] [Accessed on 9th April 2025] <https://hevodata.com/learn/s3-vs-rds/>
12. AWS API Gateway with Lambda: Quick Overview and Tutorial | Solo.io (2025) Solo.io. [Online] [Accessed on 10th April 2025] <https://www.solo.io/topics/aws-api-gateway/aws-api-gateway-lambda>
13. Amazon Aurora vs. Firebase Realtime Database Comparison (2024) Db-engines.com. [Online] [Accessed on 10th April 2025] <https://db-engines.com/en/system/Amazon+Aurora%3BFirebase+Realtime+Database>
14. Google Cloud (n.d.) What Is Microservices Architecture? Google Cloud. [Online] [Accessed on 10th April 2025] <https://cloud.google.com/learn/what-is-microservices-architecture>
15. Amazon S3 - Cloud Object Storage - AWS (2025) Amazon Web Services, Inc. [Online] [Accessed on 10th April 2025] <https://aws.amazon.com/s3/?nc=sn&amp>
16. Awad, J. W. (2025) AWS API Gateway (REST API) Deep Dive | By Joud W. Awad | Medium. Medium. [Online] [Accessed on 10th April 2025] <https://medium.com/@joudwawad/aws-api-gateway-deep-dive-rest-apis-5ae16a326b3a>
17. Sachinadi (2023) Introduction to AWS Elastic Beanstalk - Sachinadi - Medium. Medium. [Online] [Accessed on 10th April 2025] <https://medium.com/@sachinadi424/introduction-to-aws-elastic-beanstalk-5c7eb6798ecc>

# Appendices

Appendices A: AWS EC2 setup screenshots

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Appendices B: Tomcat web application manager screenshot

A screenshot of a computer

AI-generated content may be incorrect.

Appendices C: Contacts database without connected database screenshot

A screenshot of a computer

AI-generated content may be incorrect.

Appendices D: AWS Aurora and RDS DB setup screenshot

A screenshot of a computer

AI-generated content may be incorrect.

Appendices E: AWS Aurora and RDS DB being created screenshots

A screenshot of a computer

AI-generated content may be incorrect.

A green sign with white text

AI-generated content may be incorrect.

Appendices F: MySQL workbench DB connection screenshots

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Appendices G: MySQL workbench DB script running screenshot

A screenshot of a computer

AI-generated content may be incorrect.

Appendices H: Contacts database configuration screenshots

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Appendices I: Contacts database account creation and modification screenshots

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Appendices J: Java string processor built and packaged into jar file

A computer screen shot of a program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Appendices K: AWS Lambda setup

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

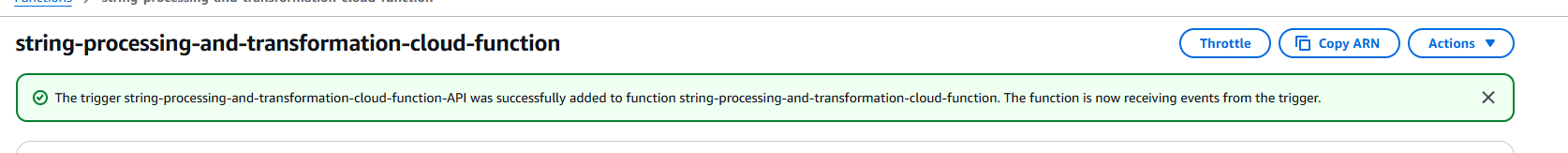
AI-generated content may be incorrect. A green and white sign

AI-generated content may be incorrect.

Appendices L: Configuring triggers and deploying API

A screenshot of a computer

AI-generated content may be incorrect.



A close-up of a computer screen

AI-generated content may be incorrect.

A screenshot of a application

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Appendices M: Testing endpoints with postman

Json:

A screenshot of a computer

AI-generated content may be incorrect.

XML:

A screenshot of a computer

AI-generated content may be incorrect.

Appendices N: .Jar file and .Java file





Appendices O: OneDrive link to code

[lambda string converter function](https://stummuac-my.sharepoint.com/:f:/g/personal/21308666_stu_mmu_ac_uk/Ei0ehtiMFhhFpwoRrPR5x60BUIMVFoYRpvX7l_LcMn9hFw?e=ptxdw2)

<https://stummuac-my.sharepoint.com/:f:/g/personal/21308666_stu_mmu_ac_uk/Ei0ehtiMFhhFpwoRrPR5x60BUIMVFoYRpvX7l_LcMn9hFw?e=ptxdw2>

Appendices P: Java code as text

package lambda\_functions;

import java.util.HashMap;

import java.util.Map;

import java.util.stream.Collectors;

import java.util.\*;

import org.json.JSONObject;

import com.amazonaws.services.lambda.runtime.Context;

import com.amazonaws.services.lambda.runtime.RequestHandler;

import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyRequestEvent;

import com.amazonaws.services.lambda.runtime.events.APIGatewayProxyResponseEvent;

import javax.xml.parsers.\*;

import javax.xml.transform.\*;

import javax.xml.transform.dom.\*;

import javax.xml.transform.stream.\*;

import org.w3c.dom.\*;

import java.io.StringWriter;

public class StringProcessingTask implements RequestHandler<APIGatewayProxyRequestEvent, APIGatewayProxyResponseEvent> {

*@Override*

public APIGatewayProxyResponseEvent handleRequest(APIGatewayProxyRequestEvent request, Context context) {

APIGatewayProxyResponseEvent responseEvent = new APIGatewayProxyResponseEvent();

try {

//parse json input into string for transformation

JSONObject inputJson = new JSONObject(request.getBody());

String text = inputJson.optString("text", "").trim();

//handle empty input

if (text.isEmpty()) {

throw new IllegalArgumentException("Please enter some json data with the value of 'text'.");

}

//process input

JSONObject result = new JSONObject();

result.put("Original : ", text);

result.put("Number of characters : ", text.length());

result.put("Number of words : ", text.split("\\s+").length);

result.put("String reversed : ", new StringBuilder(text).reverse().toString());

result.put("Words reversed : ", reverseAllWords(text));

result.put("All uppercase : ", text.toUpperCase());

result.put("All lowercase : ", text.toLowerCase());

result.put("Crazy case : ", convertToCrazyCase(text));

result.put("Camel case : ", convertToCamelCase(text));

result.put("Kebab case : ", convertToKebabCase(text));

result.put("Snake case : ", convertToSnakeCase(text));

result.put("String with vowels all removed : ", text.replaceAll("[aeiou]", ""));

result.put("Count the occurrence of every character in the string : ", countAllCharacters(text));

//check to see if user wants xml format

boolean userWantsXMLFormat = false;

Map<String, String> headers = request.getHeaders();

if (headers != null && headers.containsKey("Accept") && headers.get("Accept").contains("xml")) {

userWantsXMLFormat = true;

}

if (userWantsXMLFormat) {

responseEvent.setStatusCode(200);

responseEvent.setBody(xmlOutput(result));

responseEvent.setHeaders(getHeadersForXml());

} else {

responseEvent.setStatusCode(200);

responseEvent.setBody(result.toString());

responseEvent.setHeaders(getHeadersForJson());

}

} catch (Exception e) {

responseEvent.setStatusCode(400);

responseEvent.setBody("{\"error\": \"" + e.getMessage() + "\"}");

}

return responseEvent;

}

private String reverseAllWords(String text) {

String[] words = text.split(" ");

Collections.*reverse*(Arrays.*asList*(words));

return String.*join*(" ", words);

}

private String convertToCrazyCase(String text) {

StringBuilder crazy = new StringBuilder();

boolean upper = true;

for (char c : text.toCharArray()) {

crazy.append(upper ? Character.*toUpperCase*(c) : Character.*toLowerCase*(c));

if (Character.*isLetter*(c)) upper = !upper;

}

return crazy.toString();

}

private String convertToCamelCase(String text) {

String[] words = text.toLowerCase().split(" ");

StringBuilder sb = new StringBuilder(words[0]);

for (int i = 1; i < words.length; i++) {

sb.append(Character.*toUpperCase*(words[i].charAt(0)))

.append(words[i].substring(1));

}

return sb.toString();

}

private String convertToKebabCase(String text) {

return text.toLowerCase().replaceAll("\\s+", "-");

}

private String convertToSnakeCase(String text) {

return text.toLowerCase().replaceAll("\\s+", "\_");

}

private Map<Character, Integer> countAllCharacters(String text) {

Map<Character, Integer> charCounts = new HashMap<>();

for (char c : text.toCharArray()) {

if (!Character.*isWhitespace*(c)) {

charCounts.put(c, charCounts.getOrDefault(c, 0) + 1);

}

}

return charCounts;

}

//xml output instead of json

private String xmlOutput(JSONObject data) throws Exception {

Document doc = DocumentBuilderFactory.*newInstance*().newDocumentBuilder().newDocument();

Element root = doc.createElement("stringAnalysis");

doc.appendChild(root);

Iterator<String> keys = data.keys();

while (keys.hasNext()) {

String key = keys.next();

Element el = doc.createElement(key.replaceAll("[^a-zA-Z0-9]", ""));

if (data.get(key) instanceof JSONObject) {

JSONObject charCounts = data.getJSONObject(key);

Iterator<String> charKeys = charCounts.keys();

while (charKeys.hasNext()) {

String charKey = charKeys.next();

Element charEl = doc.createElement("char");

charEl.setAttribute("value", charKey);

charEl.setTextContent(charCounts.get(charKey).toString());

el.appendChild(charEl);

}

} else {

el.setTextContent(data.get(key).toString());

}

root.appendChild(el);

}

Transformer transformer = TransformerFactory.*newInstance*().newTransformer();

StringWriter writer = new StringWriter();

transformer.transform(new DOMSource(doc), new StreamResult(writer));

return writer.toString();

}

private Map<String, String> getHeadersForJson() {

Map<String, String> headers = new HashMap<>();

headers.put("Content-Type", "application/json");

return headers;

}

private Map<String, String> getHeadersForXml() {

Map<String, String> headers = new HashMap<>();

headers.put("Content-Type", "application/xml");

return headers;

}

}