**A(I)**

**Introduction to Medical Imaging Application**

Not many initiatives have been as valuable in the sphere of precision medicine as medical imaging *Giardino et al. (2017).* It eventually proved to be an indispensable pillar for driving advancements in diagnosis, treatment planning, and patient monitoring in modern healthcare. However, managing and analyzing the ever-increasing volume of medical image data has been a significant challenge *Kumar et al. (2012)*. With advancements in cloud technology, it is now possible to create scalable, secure, and efficient solutions to store, process, and analyze medical images *Comaniciu et al. (2016)*. This report presents the architecture of a cloud-native application designed to facilitate access and analysis of a comprehensive repository of medical imaging information that includes patient demographics, diagnostic findings, associated images, and metadata.

The application leverages Amazon Web Services (AWS) to deliver a cloud-based platform tailored for medical imaging. It enables healthcare professionals to upload, search, and retrieve imaging data in real-time. The platform is designed for scalability to handle expanding datasets, ensures strong data security to safeguard patient information, and provides a RESTful interface for flexible search and retrieval.

By leveraging the principles of cloud-native design, this solution uses AWS resources to meet healthcare data regulations, providing a reliable system for medical practitioners. It is designed for scalability, high availability, and fault tolerance, ensuring efficient and secure management of medical imaging data.

A(ii)

**Concepts of IaaS, PaaS, and SaaS: Definitions with Real-World Examples**

The development of a cloud-native medical imaging application leverages the foundational models of cloud computing: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). These models provide the necessary scalability, security, and efficiency to manage sensitive medical data effectively. Below are definitions and practical examples relevant to the proposed application.  
**1. Infrastructure as a Service (IaaS)**

IaaS is the bedrock for the application as it grants developers granular control over virtualized computing resources such as servers, networking, and storage such that they can manage and scale applications without owning physical hardware *Faridi et al. (2024*). A good example is the AWS Elastic Compute Cloud.

* **AWS EC2:**  With its diverse instance types, including those equipped with powerful GPUs *Ascentient (2024)*, this resource will serve as the powerhouse for computationally intensive tasks. In our case, the medaical imaging application runs on EC2 instances to process diagnostic images and manage requests for metadata and patient details. EC2 instances are auto-scaled to handle peak workloads during high data inflow.

**2. Platform as a Service (PaaS):**

In essence, PaaS abstracts away the complexities of infrastructure management, allowing developers to focus on crafting innovative applications. This streamlined approach is the powerhouse behind rapid development cycles and seamless scalability in healthcare technology *Covetus (2019)*.

* **AWS Elastic Beanstalk:** Elastic Beanstalk is a PaaS offering strictly focused on helping developers A PaaS offering from AWS that allows developers to deploy applications without worrying about the underlying infrastructure. In the context of our design, Elastic Beanstalk could be used to host the web service backend of the application, automatically managing deployment, scaling, and updates. For example, it enables seamless hosting of the API endpoints for uploading and retrieving diagnostic images and metadata.

**3. Software as a Service (SaaS): Delivering a User-Centric Experience**

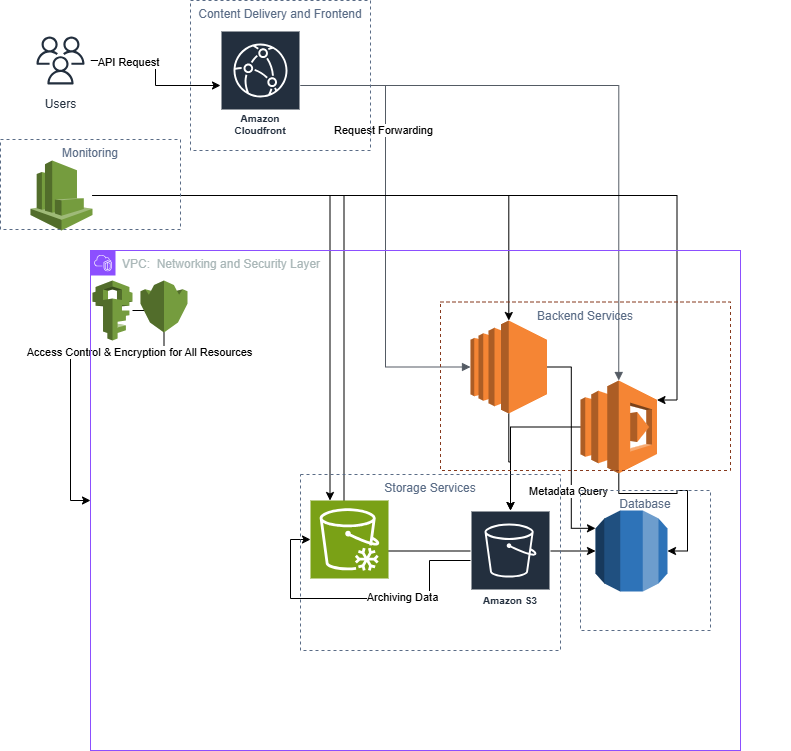
SaaS provides ready-to-use applications accessible via the internet, where users do not need to manage the underlying infrastructure or application code *Vola (2023)*.

* The medical imaging application functions as a SaaS-like web service for radiologists and doctors, enabling them to securely upload, search, and retrieve patient data and imaging files. While not a prebuilt SaaS offering, it delivers SaaS-level convenience and accessibility to its users through a secure web interface hosted on **Amazon CloudFront**

A(III)

**AWS Service Integration: Detailed Description of Selected AWS Services and Their Roles**

The design of the medical imaging application integrates a range of AWS services to ensure scalability, security, and efficiency. Each service plays a critical role in achieving the desired outcomes of scalability, security, and efficient data handling, all of which align with the foundational principles taught within the AWS Academy Cloud Foundations course.



**1. Compute:**

* **Integration:**
  + **Amazon EC2:** It serves as the backbone of backend infrastructure *Ascentient (2024)*, managing core operations such as processing user requests, running diagnostic workflows, and performing computationally demanding activities like rendering medical images and supporting AI-based analyses.
  + **AWS Lambda:** This handles event-driven tasks efficiently, including extracting metadata from uploaded images, updating the database, and sending notifications. This resource helps in optimizing resource utilization and minimizing operational costs.

This combination provides the necessary compute power for demanding workloads while ensuring cost-effectiveness by leveraging serverless functions for less resource-intensive tasks.

**2. Storage:**

* **Integration:**
  + **Amazon S3:** Serves as the primary storage repository for all diagnostic images and associated metadata, ensuring high availability, durability, and seamless data retrieval *Azeus Convene (2024)*.
  + **Amazon S3 Glacier :** Complements S3 by providing a low-cost option for archiving older or seldom-accessed data, preserving data integrity while reducing storage costs.

This dual-storage approach balances fast access to active records with cost-efficient management of historical data.

**3. Database:**

* **Integration:**
  + **Amazon RDS:** Offers a reliable and scalable relational database solution for storing structured data such as, patient demographics, diagnostic outcomes, and metadata *Techvify Software (2023)*. Its Multi-AZ deployments ensure high availability and data resilience, making it suitable for healthcare-critical environments.

RDS helps healthcare professionals to quickly query and retrieve critical patient information alongside associated imaging data, that ultimately support quicker decision-making and quality patient care.

**4. Networking and Content Delivery:**

* **Integration:**
  + **Amazon CloudFront:** Delivers images, reports, and other data to users by reducing latency *Healthcare IT News (2013)* and enhancing transfer speeds, providing a smooth user experience regardless of their location.
  + **Amazon VPC:** Creates a secure and isolated network for backend resources, protecting sensitive patient data and enhancing the overall security posture of the application.

CloudFront optimizes data delivery, while VPC provides a secure and controlled environment for all application components, enhancing data security and compliance.

**5. Security:**

* **Integration:**
  + **AWS IAM:** Implements detailed access control, allowing only authorized personnel to access and modify patient data, The App Solutions (2023).
  + **AWS KMS:** Encrypts data both at rest (within S3) and in transit, protecting sensitive information and ensuring compliance with stringent healthcare regulations such as HIPAA.

These services provide a comprehensive security framework, safeguarding patient privacy and mitigating the risk of data breaches.

**6. Monitoring:**

* **Integration:**
  + **AWS CloudWatch:** Continuously monitors the performance and health of all critical components, including EC2 instances, S3 buckets, RDS databases, and Lambda functions *Carmatec (2024)*. It also aggregates and analyzes logs, which give useful insight into application performance and identifying potential issues.

CloudWatch facilitates proactive monitoring and troubleshooting, ensuring the application remains highly available and performs optimally.

**Example Workflow:**

1. A radiologist accesses the frontend application (hosted on CloudFront) and uploads a diagnostic image.
2. The image is uploaded to Amazon S3.
3. An AWS Lambda function is triggered, which extracts the key metadata from the image and stores it securely in the Amazon RDS database.
4. Lifecycle policies automatically archive older images to Amazon S3 Glacier for cost-effective long-term storage.
5. AWS CloudWatch monitors resource utilization, logs events, and provides alerts to notify administrators of any potential issues.
6. AWS IAM and AWS KMS ensure that all data access and interactions are secure and compliant with preset and relevant regulations.

A(IV)

**Web Service Design**

The web service is designed as a **cloud-native application** with the following objectives:

1. **Accessibility**: Medical professionals can access patient data, diagnostic results, related images, and metadata through a secure web interface.
2. **Search and Filter Capabilities**: The service allows users to search and filter records based on patient IDs, diagnosis types, upload dates, and other metadata Techvify Software (2023).
3. **Efficiency**: Implements a RESTful architecture to facilitate smooth interactions between the frontend (via **Amazon CloudFront**) and backend services hosted on **Amazon EC2** and **AWS Lambda** *WJAETS (2024)*.
4. **Scalability and Security**:
   * **Amazon S3** is used for scalable storage of images.
   * **Amazon RDS** serves as the relational database for structured metadata storage.
   * **AWS IAM** and **AWS KMS** manage access control and encryption.

**REST Interface**

The REST interface provides endpoints for uploading, retrieving, searching, and managing data. Below are the key endpoints and their functionalities:

**1. Upload Diagnostic Image**

**Endpoint:** POST /uploadImage  
**Purpose:** Allows users to upload diagnostic images and related metadata.

**Request Example:**

json

Copy code

{

"patientId": "P12345",

"diagnosis": "Lung Cancer",

"imageFile": "<binary\_image\_file>",

"metadata": {

"imageType": "X-ray",

"uploadDate": "2024-12-25",

"notes": "Initial diagnosis of the patient."

}

}

**Response Example:**

json

Copy code

{

"status": "success",

"message": "Image uploaded successfully.",

"imageId": "img67890"

}

**2. Retrieve Patient Details**

**Endpoint:** GET /getPatientDetails/{patientId}  
**Purpose:** Fetches detailed information about a patient, including their diagnostic results and associated images.

**Request Example:**  
URL: /getPatientDetails/P12345

**Response Example:**

json

Copy code

{

"patientId": "P12345",

"name": "John Doe",

"age": 54,

"diagnoses": [

{

"diagnosis": "Lung Cancer",

"imageId": "img67890",

"uploadDate": "2024-12-25",

"notes": "Initial diagnosis of the patient."

}

]

}

**3. Search Diagnostic Images**

**Endpoint:** GET /searchImages  
**Purpose:** Enables users to search for images based on filters like diagnosis type, image type, or date of upload.

**Request Example:**  
URL: /searchImages?diagnosis=Lung Cancer&imageType=X-ray&uploadDate=2024-12-25

**Response Example:**

json

Copy code

{

"status": "success",

"results": [

{

"imageId": "img67890",

"patientId": "P12345",

"diagnosis": "Lung Cancer",

"imageType": "X-ray",

"uploadDate": "2024-12-25"

}

]

}

**4. Delete Diagnostic Image**

**Endpoint:** DELETE /deleteImage/{imageId}  
**Purpose:** Deletes a specific diagnostic image and its metadata.

**Request Example:**  
URL: /deleteImage/img67890

**Response Example:**

json

Copy code

{

"status": "success",

"message": "Image with ID img67890 has been deleted."

}

**5. Update Patient Information**

**Endpoint:** PUT /updatePatient/{patientId}  
**Purpose:** Updates details about a specific patient.

**Request Example:**

json

Copy code

{

"name": "Jane Doe",

"age": 55

}

**Response Example:**

json

Copy code

{

"status": "success",

"message": "Patient information updated successfully."

}

**Search and Access Options**

1. **Filters**:
   * The /searchImages endpoint allows doctors to filter records by:
     + **Diagnosis Type**: e.g., "Cancer."
     + **Image Type**: e.g., "MRI," "X-ray."
     + **Upload Date**: e.g., "2024-12-25."
2. **Direct Access**:
   * The /getPatientDetails endpoint provides direct access to all records for a specific patient.
3. **Metadata-Driven Retrieval**:
   * Metadata stored in **Amazon RDS** is used to refine searches and connect diagnostic images (from S3) with patient records.

**Security in REST API**

1. **Authentication**:
   * All endpoints require **JWT (JSON Web Tokens)** for secure user authentication.
   * Example header:  
     Authorization: Bearer <jwt\_token>
2. **Encryption**:
   * Data is encrypted at rest using **AWS KMS** and in transit using **HTTPS**.
   * Sensitive data such as patient records and imaging files are secured with strict access policies in **AWS IAM**.
3. **Access Control**:
   * Role-based access ensures that only authorized users (e.g., radiologists, doctors) can interact with specific endpoints.

B(I)

**Description of the Application**

The developed application is designed to analyze bacterial organisms based on their features using Gene Ontology (GO) annotations provided in .gaf files. The focus of the analysis is on the identification of organisms that exhibit the "competence" feature, a physiological state associated with the ability to take up exogenous genetic material. This feature is indicated by the GO term GO:0030420 and its subterms.

The application processes input .gaf files, which contain tab-separated data describing proteins and their functions *Ashburner et al. (2000)*. Specifically, the second column contains protein identifiers, and the fifth column includes associated GO terms. By analyzing occurrences of GO:0030420 and its subterms in the fifth column, the application determines the likelihood of each organism exhibiting competence.

The .gaf files used as input were:

* *Escherichia\_coli\_K-12\_ecocyc\_83333.gaf*
* *Bacillus\_subtilis\_168-224308.gaf*
* *Bacillus\_amyloliquefaciens\_FZB42-326423.gaf*
* *Bacillus\_licheniformis\_ATCC\_14580-279010.gaf*
* *Bacillus\_megaterium\_DSM\_319-592022.gaf*
* *Geobacillus\_kaustophilus\_HTA426-235909.gaf*
* *Geobacillus\_thermodenitrificans\_NG80\_2-420246.gaf*

For each organism, the application generates an output file summarizing the count of "competence" and its associated subterms, providing insights into the potential for DNA uptake within each bacterial species.

B(ii)

**Input and Output**

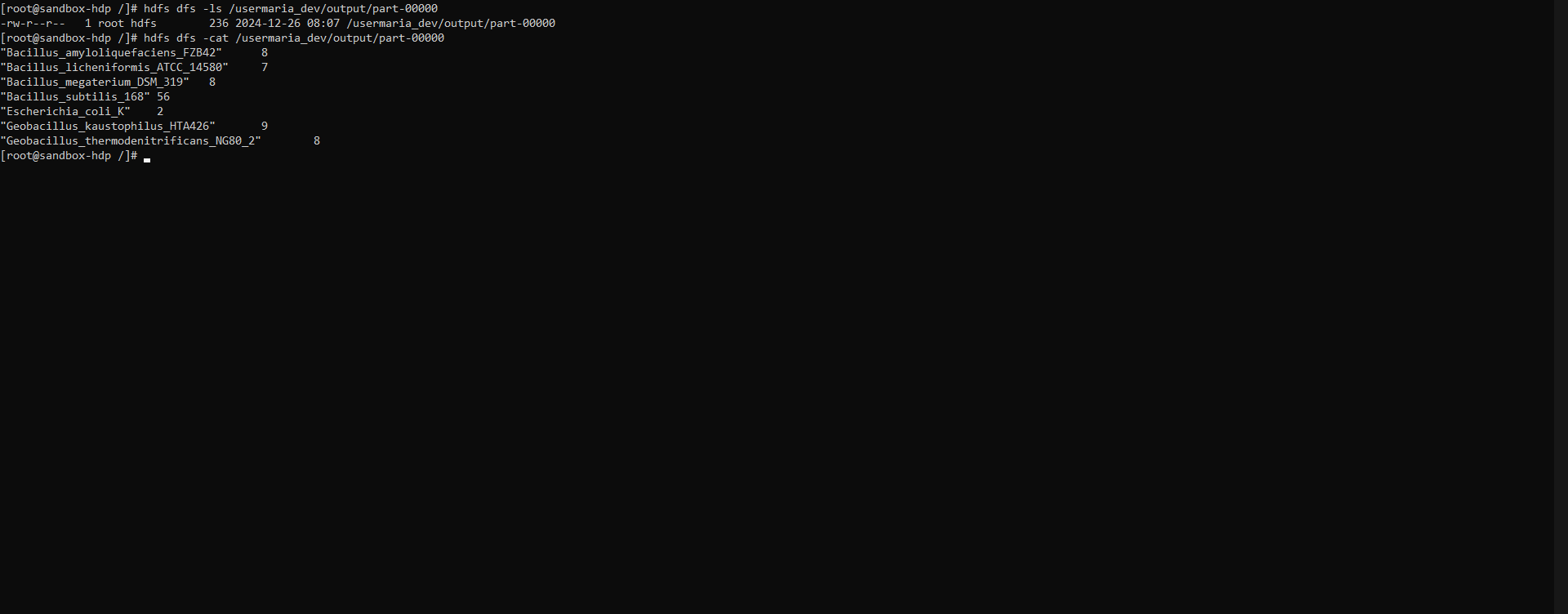
**Input**

The .gaf files used in the application contain 17 fields, with the following relevant fields:

* **Column 2**: Protein Identifier
* **Column 5**: GO Term (e.g., GO:0030420)

These files were uploaded to Hadoop's HDFS for processing.

**Output**

****

The output lists the count of GO:0030420 and its subterms for each organism, as shown below:

"Bacillus\_amyloliquefaciens\_FZB42" 8

"Bacillus\_licheniformis\_ATCC\_14580" 7

"Bacillus\_megaterium\_DSM\_319" 8

"Bacillus\_subtilis\_168" 56

"Escherichia\_coli\_K" 2

"Geobacillus\_kaustophilus\_HTA426" 9

"Geobacillus\_thermodenitrificans\_NG80\_2" 8

These results indicate the frequency of competence-related terms in the GO annotations for each organism.

B(iii)

**Methodology**

**Technology Stack**

The application was developed *Taylor (2010)* using the following technologies:

* **Hadoop**: For distributed storage and computation.
* **MapReduce**: To process large .gaf files in a scalable manner.
* **Python**: Specifically, the mrjob library for writing the MapReduce job.

**Implementation Logic**

* **Mapper**: Extracts the organism name (from the file name) and counts occurrences of GO:0030420 and its subterms in the fifth column of the .gaf files.
* **Reducer**: Aggregates the counts for each organism to produce a total count of competence-related terms.

**Execution Steps**

1. **Upload Input Files**: Input .gaf files were uploaded to HDFS under /user/maria\_dev/input using the following command:

2. hdfs dfs -put /root/input\_files/\*.gaf /user/maria\_dev/input

1. **Run the MapReduce Job**: The MapReduce job was executed using the command:

4. python /root/competence\_feature\_count.py -r hadoop hdfs:///user/maria\_dev/input/\*.gaf \

5. --output-dir hdfs:///usermaria\_dev/output \

6. --hadoop-streaming-jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming.jar

1. **View the Results**: The results were accessed in HDFS:

8. hdfs dfs -cat /usermaria\_dev/output/part-00000

B(IV)

**Findings and Evaluation**

**Findings**

* Bacillus subtilis 168 exhibited the highest occurrence of competence-related GO terms (56 occurrences), indicating it is the most likely to display the competence feature.
* Bacillus amyloliquefaciens FZB42 and Geobacillus kaustophilus HTA426 showed moderate occurrences (8 and 9, respectively), suggesting a lower likelihood of competence compared to Bacillus subtilis
* *Escherichia\_coli\_K-12* had the lowest count (2 occurrences), indicating a lower likelihood of exhibiting competence.

**Evaluation**

1. **Scalability**:
   * The use of Hadoop and MapReduce demonstrated the potential for scalability. Adding more organisms or larger .gaf files would not significantly impact performance.
2. **Accuracy**:
   * The application successfully identified and counted the relevant GO terms (GO:0030420 and its subterms) as per the task requirements.
3. **Limitations**:
   * The results depend entirely on the annotations provided in the .gaf files. Missing or incomplete data in these files could affect the analysis.
   * The application employs simple pattern matching for GO subterms and does not currently perform hierarchical analysis that considers deeper relationships within the GO ontology structure.
4. **Improvements**:
   * Future iterations could make use of an enhanced parser for .gaf files to handle unexpected formats or missing fields gracefully.
   * Additional features could include visualizing the results or integrating with external databases to enrich the analysis.

**Conclusion**

The analysis of bacterial organisms using Hadoop and MapReduce successfully identified competence-related features based on Gene Ontology (GO) terms. The application demonstrated its capability to efficiently process large .gaf files and extract meaningful insights, highlighting organisms likely to exhibit competence as a physiological state.

The findings showed that *Bacillus\_subtilis\_168* had the highest competence feature count, followed by other organisms such as *Geobacillus\_kaustophilus\_HTA426*. The application’s scalability and efficiency make it well-suited for processing even larger datasets in future applications. However, limitations related to input data quality and the scope of GO term analysis provide opportunities for further enhancement.

By leveraging distributed computing resources, this approach can scale to analyze thousands of organisms, making it a valuable tool in bioinformatics research.

**References**

1. Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T. and Harris, M.A., 2000. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1), pp.25-29.
2. Carmatec, 2024. Cloud Computing in Healthcare and Medicine: Key Trends. *Carmatec*. Available at: https://www.carmatec.com/blog/cloud-computing-in-healthcare-and-medicine-key-trends/ [Accessed 26 Dec. 2024].
3. Comaniciu, D., Engel, K., Georgescu, B. and Mansi, T., 2016. Shaping the Future through Innovations: From Medical Imaging to Precision Medicine. *arXiv preprint arXiv:1605.02029*. Available at: <https://arxiv.org/abs/1605.02029> [Accessed 26 Dec. 2024].
4. Covetus, 2019. PaaS: Introduction, its Benefits & Challenges in the Healthcare Industry. *Covetus*. Available at: https://www.covetus.com/blog/paas-introduction-its-benefits-challenges-in-the-healthcare-industry [Accessed 26 Dec. 2024].
5. Faridi, S., Sarwar, U., Ahtisham, M. and Khan, S., 2024. Cloud computing in healthcare: A comprehensive review of data storage, analysis, and security. *World Journal of Advanced Engineering Technology and Sciences*, 5(3), pp.291-300.
6. Gene Ontology Consortium, 2023. Gene Ontology Annotation File Format Guide. *Gene Ontology Consortium*. Available at: <https://geneontology.org> [Accessed 26 Dec. 2024].
7. Giardino, A., Gupta, S., Olson, E., Sepulveda, K., Lenchik, L., Ivanidze, J., Rakow-Penner, R., Patel, M., Subramaniam, R. and Ganeshan, D., 2017. Role of Imaging in the Era of Precision Medicine. *Academic Radiology*, 24(5), pp.639-649.
8. Healthcare IT News, 2013. Which Cloud Model is Right for Your Healthcare Organization? *Healthcare IT News*. Available at: https://www.healthcareitnews.com/blog/which-cloud-model-right-your-healthcare-organization [Accessed 26 Dec. 2024].
9. Johnston, C., Martin, B., Fichant, G., Polard, P. and Claverys, J.P., 2014. Bacterial transformation: distribution, shared mechanisms and divergent control. *Nature Reviews Microbiology*, 12(3), pp.181-196.
10. Kumar, V., Gu, Y., Basu, S., Berglund, A., Eschrich, S.A., Schabath, M.B., Forster, K., Aerts, H.J., Dekker, A., Fenstermacher, D., Goldgof, D.B., Hall, L.O., Lambin, P., Balagurunathan, Y., Gatenby, R.A. and Gillies, R.J., 2012. Radiomics: the process and the challenges. *Magnetic Resonance Imaging*, 30(9), pp.1234-1248.
11. Taylor, R.C., 2010. An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics. *BMC Bioinformatics*, 11(Suppl 12), p.S1.
12. Techvify Software, 2023. Healthcare Cloud Computing – A Comprehensive Guide. *Techvify Software*. Available at: https://techvify-software.com/healthcare-cloud-computing/ [Accessed 26 Dec. 2024].
13. The App Solutions, 2023. Cloud Computing in Healthcare: Advantages, Applications, & Challenges. *The App Solutions*. Available at: https://theappsolutions.com/blog/cloud/cloud-computing-healthcare/ [Accessed 26 Dec. 2024].
14. Vola, 2023. SaaS, PaaS, and IaaS in Healthcare Software. *Vola*. Available at: https://vola.app/saas-paas-and-iaas/ [Accessed 26 Dec. 2024].
15. WJAETS, 2024. Cloud computing in healthcare: A comprehensive review of data storage, analysis, and security. *World Journal of Advanced Engineering Technology and Sciences*, 5(3), pp.291-300.

**Appendix**

**A. Source Code**

# competence\_feature\_count.py

from mrjob.job import MRJob

class GAFCompetenceCount(MRJob):

def mapper(self, \_, line):

fields = line.split('\t')

if len(fields) > 4 and "GO:0030420" in fields[4]:

filename = self.options.input\_file.split('/')[-1]

organism = filename.split('-')[0]

yield organism, 1

def reducer(self, organism, counts):

yield organism, sum(counts)

if \_\_name\_\_ == "\_\_main\_\_":

GAFCompetenceCount.run()

**B. Output File Content**

"Bacillus\_amyloliquefaciens\_FZB42" 8

"Bacillus\_licheniformis\_ATCC\_14580" 7

"Bacillus\_megaterium\_DSM\_319" 8

"Bacillus\_subtilis\_168" 56

"Escherichia\_coli\_K" 2

"Geobacillus\_kaustophilus\_HTA426" 9

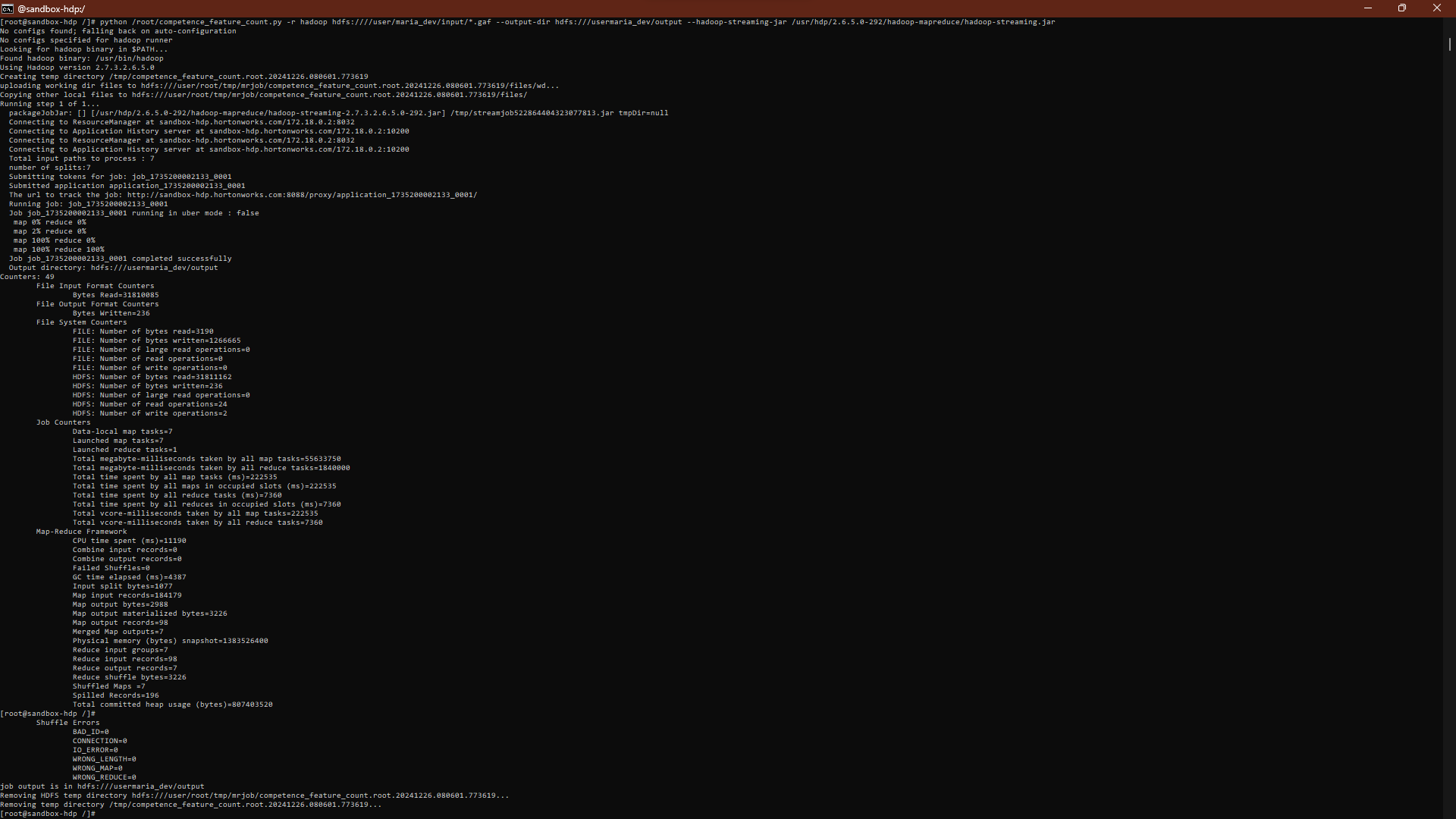
"Geobacillus\_thermodenitrificans\_NG80\_2" 8

**C. Commands Executed**

1. Upload input files:

hdfs dfs -put /root/input\_files/\*.gaf /user/maria\_dev/input

1. Run the MapReduce job:



python /root/competence\_feature\_count.py -r hadoop hdfs:///user/maria\_dev/input/\*.gaf \

--output-dir hdfs:///usermaria\_dev/output \

--hadoop-streaming-jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming.jar

1. View results:

hdfs dfs -cat /usermaria\_dev/output/part-00000