DT0223: SOFTWARE ARCHITECTURES

a.y.2019/2020

Henry Muccini

University of L’Aquila, Italy

Architecture Design Decision &

Group Decision Making Project

AI Operationalization in the Cloud for Healthcare System

With the support by Davide del Vecchio, and Karthik Vaidyanathan

Date	02.12.2019
Team ID	Team IT’S

TEAM MEMBERS		
NAME & SURNAME	MATRICULATION NUMBER	E-MAIL ADDRESS
Jayasurya Arasur-Subramanian	267359	jayasurya.arasursubramanian@student.univaq.it
Hilal Taha	267365	hilaltaha@hotmail.com
Niharika Gouda	267362	niharika.gouda@student.univaq.it

Contents

Challenges/Risk Analysis

Risk	Date the risk is identified	Date the risk is resolved	Explanation on how the risk has been managed
Architectural choice	23/11/2019	27/11/2019	We decided to design a publish/subscribe system so that we can exploit all the benefits of this architecture.
Establishing connection between each layer in architecture.	12/12/2019	12/12/2019	Layer modularity will help with the identifying specific connection points.
Data Security	13/12/2019	13/12/2019	Cloud encrypts the data and stores it.
Connection establishment between the edge device and the server	21/12/2019	06/1/2020	The connection will establish as soon as the edge device is ready to use and have the installed model
Deployment engineer traditionally don’t master the project’s code.	3/1/2020	11/1/2020	Making the data scientist involved in the deployment process of the AI model
Inaccurate predictions	9/1/2020	8/2/2020	When Evaluating the model, a threshold is defined to considered the prediction or not.

Requirements Refinement

For the requirements we added a priority factor to each requirement in order to rate them while making our design decisions.

Functional Requirement

A. The system must be able to train models using 3 methods of training: batch training, Ad-hoc training, Dynamic.

Priority: High, since our system serve all AI requirements of the company it should support all kind of training methods. Data scientist should be able to train models and cloud architecture will allow us to support all the three training methods.

B. System must constantly evaluate the accuracy of the model.

Priority: High, since our system deals with high critical data (medical data) we should pay more attention on the accuracy of the model which leads to the final analysis. The stakeholders (data engineers and data scientists) ensures that the model is accurate in providing the results.

C. Versioned models must be stored in specific repository and made it available to serving layer.

Priority: High, All previous models must be provided at all time.

D. The system must be able to predict the health status of the employee using the trained models.

Priority: High, since this is the goal of the system which is monitoring users.

Non-Functional Requirements

A. Each 500mb of data must send to the cloud.

Priority: Medium, due to the fact that we a.

B. The system must be able to withstand any kind of security threats.

Priority: High, Cloud architecture helps in ensuring the data is secure and the permission to access the data is only given to the stakeholders. The public and private key is used to access the system.

C. The cloud service must have the capability to process 10 requests within 5 minutes (uploading in parallel).

Priority: High, since we are going to host multiple projects at the same time.

D. The system should start hosting an initial set up of 10 specific projects each one will have a model that is built continuously during the working hours

Priority: Low, REST API helps in the concurrency of the system. Hence multiple users can access the server at a time.

E. System should update the models every 24hrs.

Priority: Low, since our system is updating the data continuously.

F. Edge device should sense the data of the person for every 5 minutes and sends the data to the system

Priority: High, because the data produced is needed to create the models.

Informal Description of the System and its Software Architecture

Architectural Patterns

We identified two architectures which will serve our requirement. They are

1. Cloud Architecture
2. Multilayered Architecture

Cloud Architecture

Cloud Architecture refers to the various components in terms of databases, software capabilities, applications, etc. engineered to leverage the power of the cloud resources to solve business problems. Cloud architecture defines the components as well as the relationships between them.

The various components of the Cloud Architecture are:

- On premise resources
- Cloud resources
- Software components and services
- Middleware

The entire cloud architecture is aimed at providing the users with high bandwidth, allowing users to have uninterrupted access to data and applications, on demand agile network with possibility to move quickly and efficiently between servers or even between clouds and most importantly network security.

The ever-increasing need for data processing, storage, elastic and unbounded scale of computing infrastructure has provided great thrust for shifting the data and computing operations on the cloud. Cloud computing is a cost-efficient model for service provision. The adoption of cloud computing is done because most of the services provided by the cloud are low cost and readily available. The pay- as-you- go structure of the cloud is particularly suited for our application. Moving to the cloud has reduced the cost of computing and operations due to resource sharing, virtualization, less maintenance cost, lower IT infrastructure cost, lower software cost, expertise utilization and sharing. Other than this cloud architecture can support various services like SaaS, PaaS, DaaS, IaaS. In our case we use Platform as a Service (PaaS) for providing application platform and database for storage of data.

Multi-layered Architecture

Multilayered Architecture moderates the increasing complexity of modern applications. It also makes it easier to work in a more agile manner. The Multilayered architecture consists of three layers:

- Presentation layer
- Application layer
- Data layer

The multilayered approach is good for developing web-scale, production-grade, and cloud-hosted applications are very quick and relatively risk-free. It also makes it easier to update any legacy systems. When architecture is broken up into multilayers, the changes that need to be made will be simpler and less extensive.

In our case we are building a system in which application logic is split into smaller components that spread across several servers and the system under consideration requires faster network communications, high reliability, and great performance. So, we are going to the multilayered architecture. One of the complex parts in multilayered architecture is establishing the communication between each layer. This can be overcome by the layer of Isolation concept. For more details, verify the book Software Architecture Patterns by Mark Richards.

In our case we need to establish the connection between business layer and the data layer. But we need to go through persistence layer to reach data layer. So, we put persistence layer in open according to isolation concept which gives direct access for business layer to the data layer. By putting persistence layer in open, the business layer can bypass the persistence layer and gain direct access to the data layer.

Architecture Description:

Figure.1 represents the architecture diagram of our system. The architecture of our system is made by the combination of layered and cloud architecture (for cloud we use Microsoft Azure). The layered architecture consists of three layers namely Presentation layer, Business layer and Data layer. The presentation layer consists of edge device which is used for sensing of the data and it will publish the data to the project server which is present on cloud in the business layer through pub-sub. The business layer consists of cloud architecture where training of our model takes place. The project server sends the collected data to the cloud architecture where the model is going to train, scored and evaluate.

After the model is trained it is also stored in the database. Before it get stored the model will go under version control and then it get encrypted by using Microsoft Managed Keys and then stored in database (cloud storage) which is in data layer.

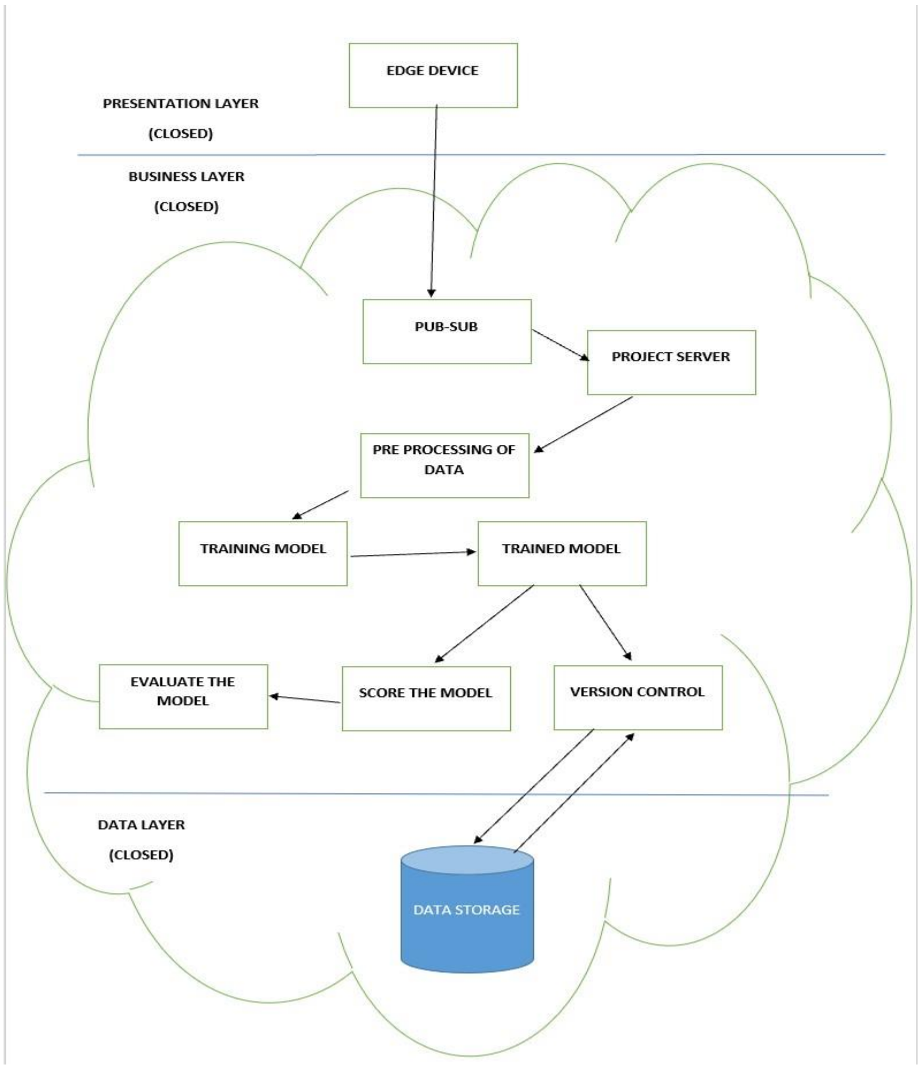

Figure 1Reference Architecture

Design Decisions

Concern	What is the software architecture that suits our system? How to store the data?
Alternatives	1. Blackboard and Webservice 2. Multi layered and Cloud Architecture 3. Layered and Microservice
Ranking criteria	1. Speed 2. Usability 3. Availability 4. Scalability 5. Reliability
Architectural decision	[2]
Description	We don’t have the frequency of the request, so we need more speed to process the requests instead of the ability to receive more request so We are going for multilayered and cloud architecture.
Rationale	We divide the system to 3-layer Presentation layer, Application layer and Data layer. Presentation layer sends the data to the application layer which will have two parts which is cloud architecture and we will have a data layer connected to the application layer.

Concern	When to send data, which is sensed from the edge device to the Project server?
Alternatives	1. Every 30 seconds 2. On status check 3. Every 5 minutes
Ranking criteria	1. Power consumption 2. Availability

	3. Convenience
Architectural decision	[3]
Description	In the case of sending the sensed data every 30 seconds this will be consuming energy really fast when we will need the edge device to work for a whole working day also the server might be busy doing computations since it’s not expected to be a very powerful computation device, in the case of the status check the end user might not be checking his status frequent enough to generate data needed for training therefore we need a middle solution between both.
Rationale	The edge device will send data every 5 minutes interval which won’t use a lot of energy and since we have a lot of end users for each project it should be enough for the training phase also it won’t depend on the usage of the device from the end user.

Concern	Size of each batch of sensed data uploaded to the cloud?
Alternatives	1. 500 GB 2. 900 GB 3. After interval of time
Ranking criteria	1. Speed 2. Efficiency 3. Storage
Architectural decision	[1]
Description	The edge device will be sensing and sending data continuously to the local storage that we will have.
Rationale	The data is critical and hence how much data is sensed by the cloud for processing is significant for the system and 500 GB is chosen because it is a good amount for training and there is no need to over load files with a bigger amount and increase failure probability, and it is easier to handle faults.

Concern	Type of database to be adopted for the system
Alternatives	1. On premise 2. Cloud database
Ranking criteria	1. Availability 2. Reliability 3. Scalability 4. Efficiency 5. Cost
Architectural decision	[2]
Description	Whenever the system finishes the training model it updates those model in the database, so we need a database which can manage large data, provides security to our data, updating the data with ease and data integrity.
Rationale	When compared to any other database, cloud database provides large bandwidth and huge storage capacity as we use. They also provide scalability on demand along with high availability and data integrity. We can also choose among hybrid, public and private cloud according to our suitability.

Concern	Which cloud service provider is most suitable for our system?
Alternatives	1. Azure 2. AWS 3. Google Cloud
Ranking criteria	1. Availability 2. Reliability 3. Failure tolerance 4. Security

	5. Deployment of the model 6. Pricing
Architectural decision	[1]
Description	One of the important scenarios in our system is training the model and storing them in the database. That training should be done without any limitation and should be flexible for pricing. Another important thing is availability the system must be up at all time to receive either data or training requests, and the reliability of subscribed and published data.
Rationale	When compared to any other cloud provider Azure provides more flexibility in pub-sub service and the pricing also compatible when compare to other providers. It doesn’t give any limitation for importing and exporting of data. And it also provides encryption of data by Microsoft standard keys. As well as it satisfies the company’s private policies.

Views and Viewpoints

Stakeholders

- End users (public)
- Data Scientist
- Data Engineer
- Developers building AI solutions

Concerns

The major concerns of our system are

- Dependency
- Energy Consumption
- Networking and communication
- Usability
- Performance
- Security
- Cost

Concern Stakeholder Traceability

	End user	Data Engineers	Data Scientist	Developer
Dependability		x	x	x
Energy Consumption	x	x	x	x
Networking & Communication		x		x
Usability	x		x	
Performance		x	x	x
Security				x
Cost	x	x		x

Static and Dynamic View

Use Case Diagram

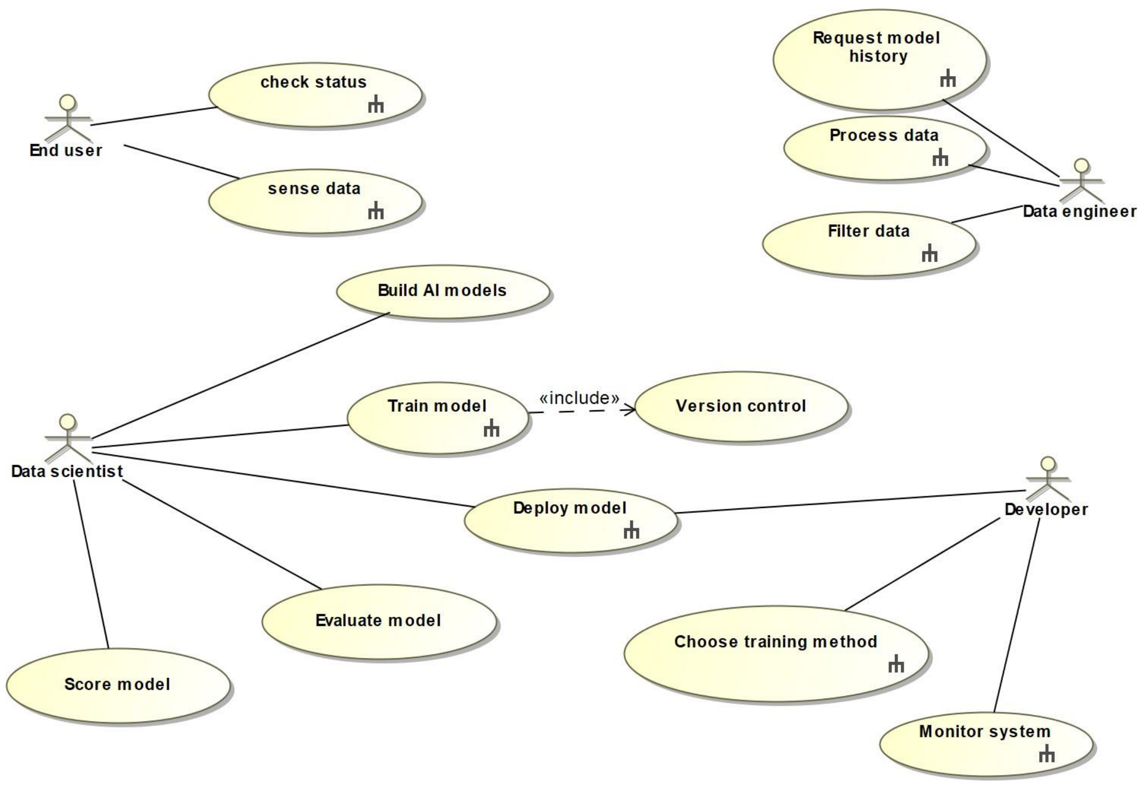

Figure 2 Use case diagram

End User(actor): Any user that we are monitoring their health status, this user will be able to check his health status and we will be constantly sensing his health indicators.

Check status: end user will be able to check his medical status at any time through the edge device

Sense Data: the edge device will be sensing the data from the user every interval of time (5 minutes) and published into the message broker.

Data Engineer(actor): Person who process the data and prepare the training model. He will request the data from the source code repository for processing and filtering of the data which are going to be trained.

Request model history: The model history of the system is requested by the data Engineers for incase to know whether to deploy the model or not.

Process data: The data are get preprocessed in this before undergoing training process which includes selection of columns which should be used for training process.

Filter data: The data are get filtered which is the empty rows are get removed before undergoing training process.

Data Scientist(actor): Person who creates AI models using methods decided by the system developer. He will train the model according to the AI model which he had decided, and he will score and evaluate the trained model.

Build AI model: The Data scientist will be able to build models for training in the cloud.

Train Model: Data scientist will invoke the training process feeding it the parameters needed.

Version Control: After training the model the resulting model will be stored in order to access whenever needed.

Developer(actor): Person who build AI powered solution consuming models. He will then deploy those models into the system, and he will monitor the performance of the system.

Chose training method: The training method which is going to be used for training the model is selected in this by developer.

Component Diagram

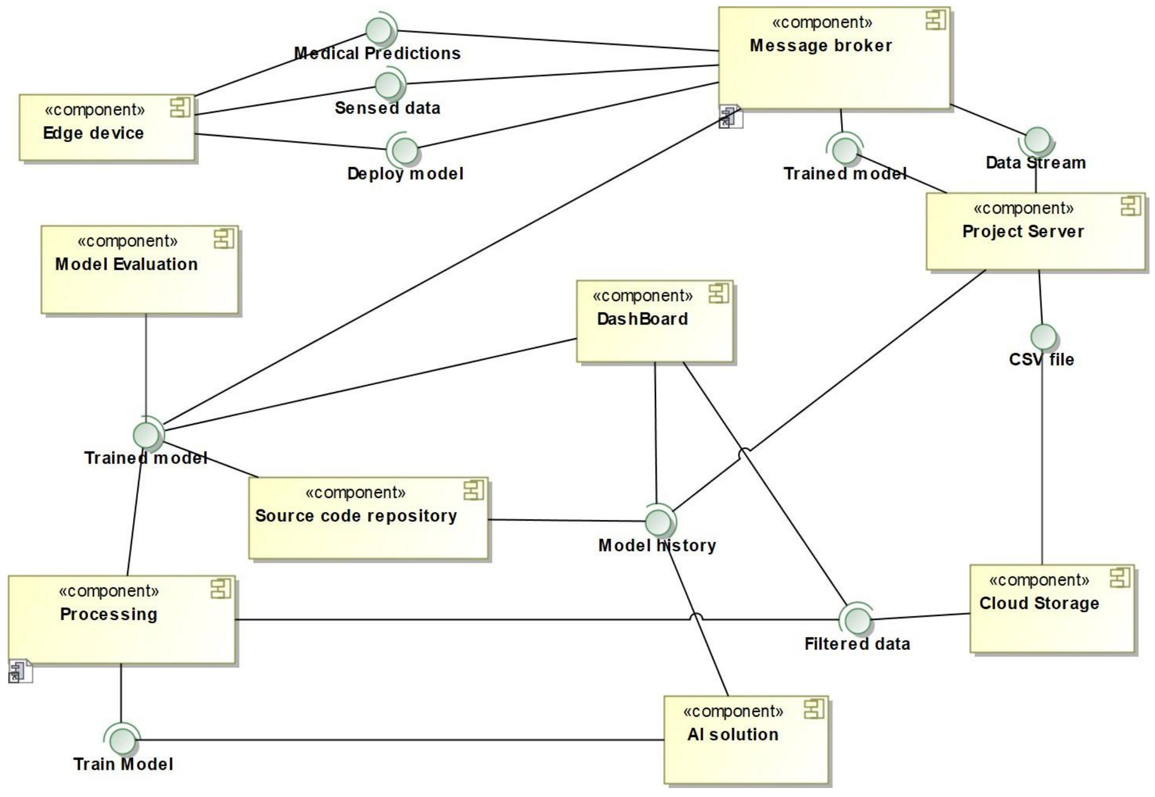

Figure 3 Component diagram

The component diagram is showing us all the software components and what they require and provide in order to accomplish the system requirements.

1. **Edge Device:** The software component that is installed on the edge device and responsible for the predictions and sensing the health status indicators of the user.
2. **Kafka Message Broker:** We have this component to manage the stream of data received from all the users.
3. **Project Server:** This is local component that will receive all the data and prepares them in a CSV file to be sent in batches to the cloud storage.
4. **Cloud Storage:** This component is where the data are Stored and get filtered before training.

5. **Processing:** Modeling of data takes place.
6. **AI Solution:** To provide the suitable model for training and it is possible to retrieve.
7. **Model evaluation:** Component used to evaluate the trained model.
8. **Source Code Repository:** To Store the model history and provide it when needed to deployment or for monitoring (cloud storage).
9. **Dashboard:** Receives information from other components to provide monitoring functionality.

Processing Component:

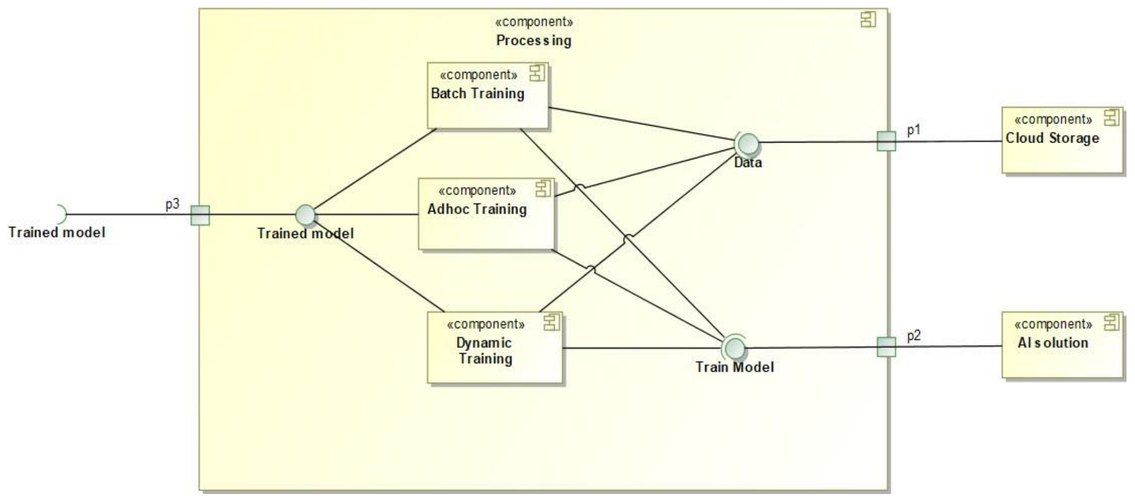

Figure 4 Component diagram of processing

Uses the training model and the filtered data to train the data and provide the training model that will be stored and deployed and reused for another training session in some cases, the training methods are Batch computing, Ad-hoc Computing and Dynamic computing, for dynamic computing the model can be entered again using the AI solution Component.

Message Broker Component:

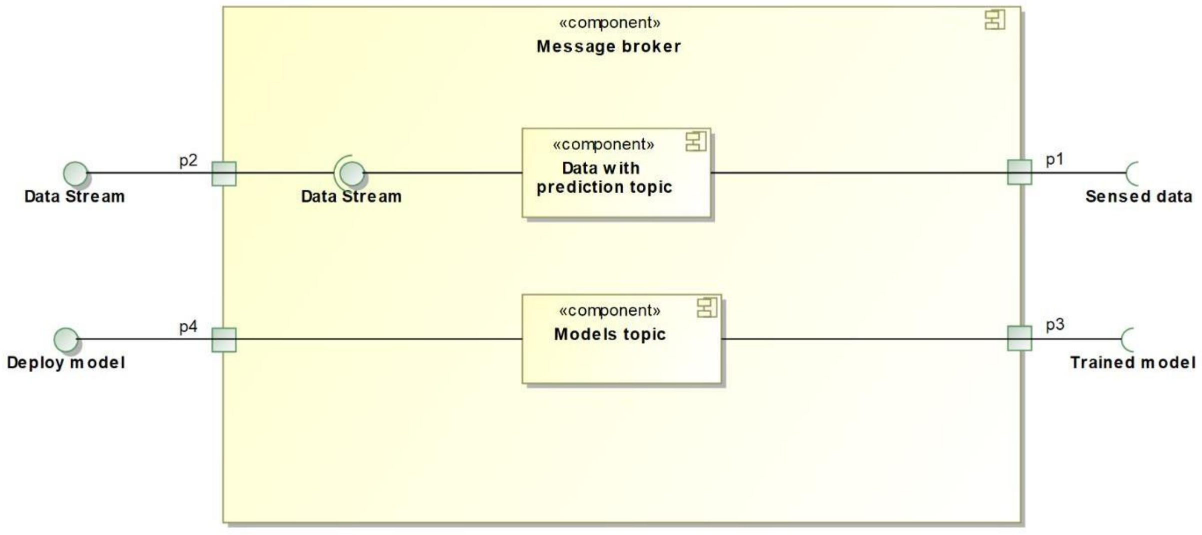

Figure 5 Component Diagram of Kafka Message Broker

The Message broker is an essential component in our system the edge device will publish the data on a topic and all the medical predictions done by the edge device about the health status of the end user will also be published, then the data is provided to the next component, also for deploying the trained model on the edge device the data published by the various previous components and then it is published to be deployed on the edge device.

Sequence Diagrams

The following sequence diagrams explain component interaction arranged in time sequence.

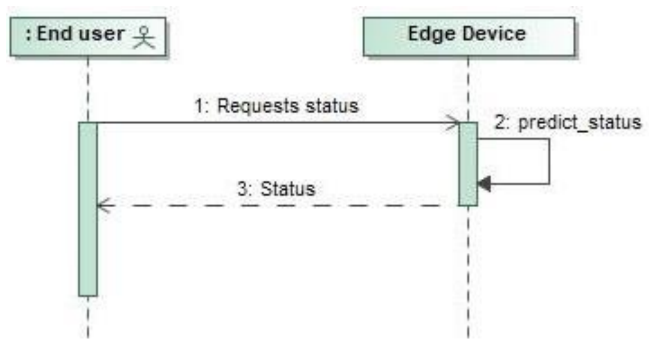

Figure 6 Check status sequence

The above sequence diagram explains the interaction takes place while checking status of health between the end user and the edge device. The end user requests the edge device for health status. The edge device predicts the status of the user’s health and displays the status. Initially when the user wants to check his medical status he requests that through the edge device. After getting a request the edge device will respond to the user with his already predicted medical status.

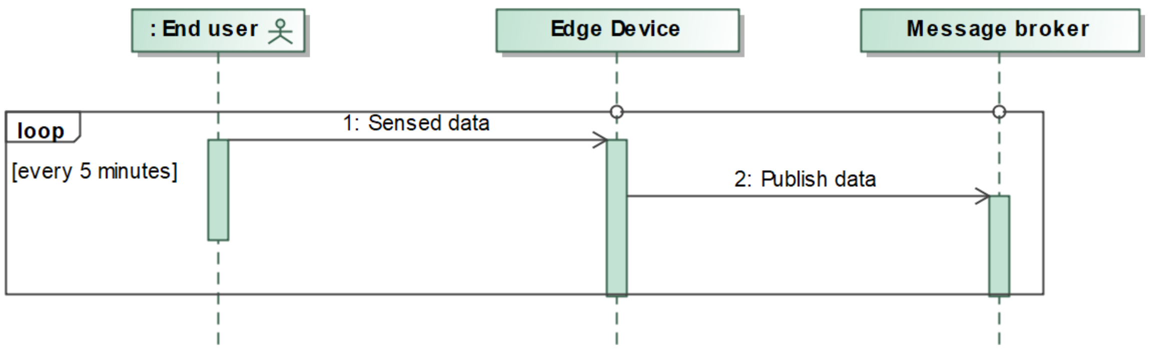

Figure 7 Sense data sequence

The above diagram explains the interaction that takes place between end user, edge device and message broker while sensing the medical data of the user. The edge device will continuously sense the health data of the user for every 5 minutes and publish those data to the message broker. From there the data are sent to the project server.

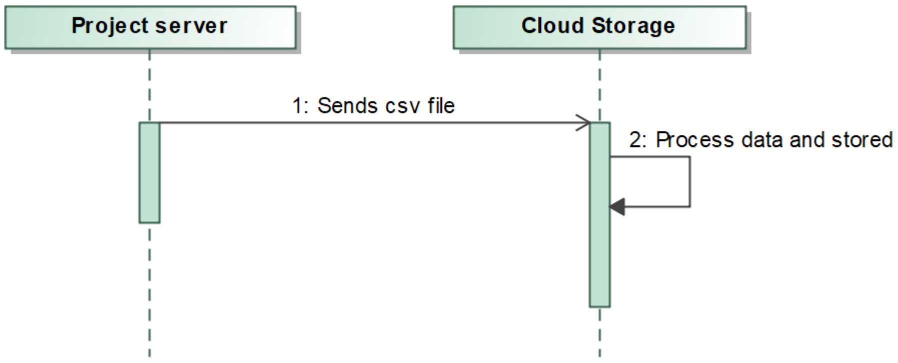

Figure 8 Process data sequence

The above diagram represents the processing of data. The Cloud Storage gets the data from the project server in the form of csv files and process those data. The processed data is then get stored in the storage.

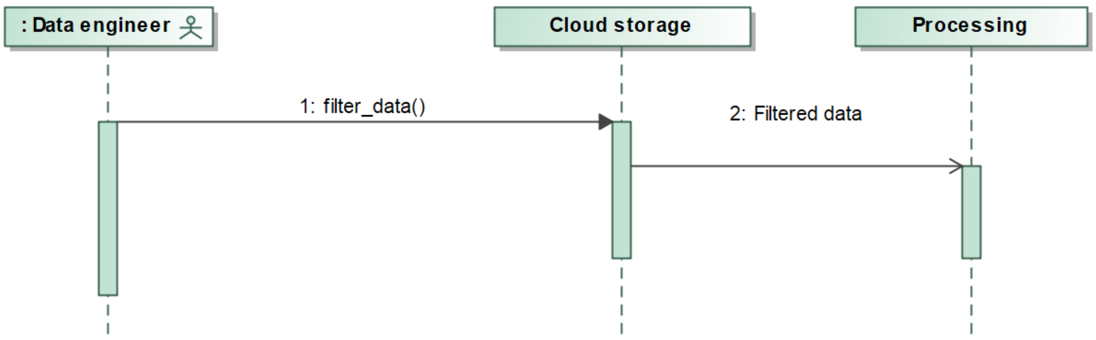

Figure 9 Filter data sequence

The above diagram represents the filtering of data which is used for training process. The data stored in the cloud storage get filtered by the data engineers and then these data are sent to the processing component where the training of model takes place.

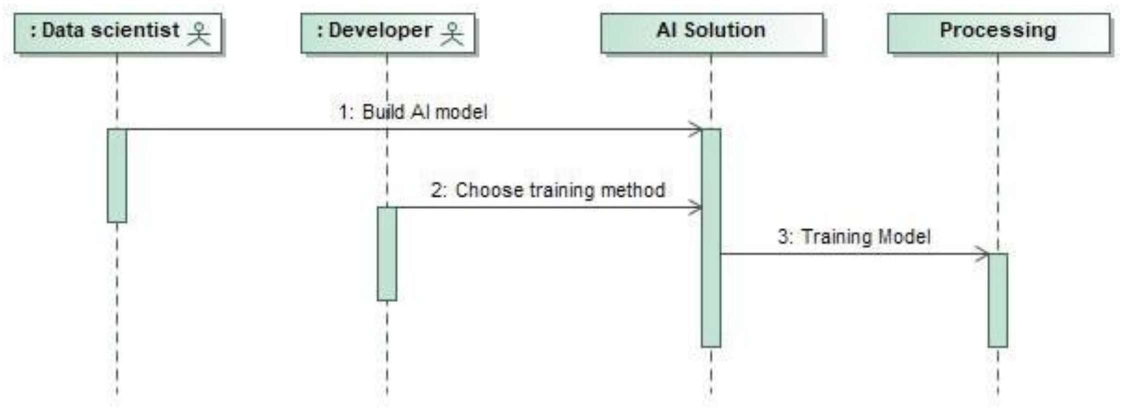

Figure 10 Choose training method sequence

The above diagram represents how the system will choose the training method and work according to it. The data scientist will build a AI model and sends it to AI solution, The Developer will choose the training method and send that to the AI solution which will combine both and make a training method and parameters for the model and send those to the processing unit where data modeling takes place.

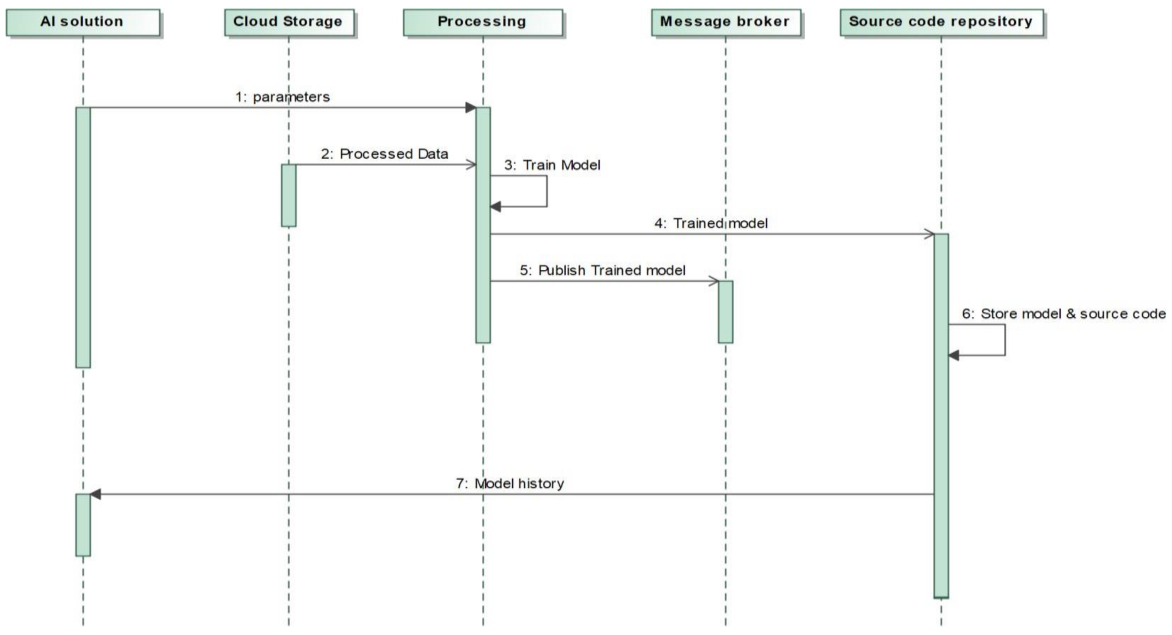

Figure 11 Train model

If the chosen training method is batch or Ad-hoc our system follows the above sequence. From AI solution and from cloud storage, Processing unit will receive parameters and processed data respectively and it trains the model. Then the trained model is published to message broker and sent to source code repository where the model and source code will be stored.

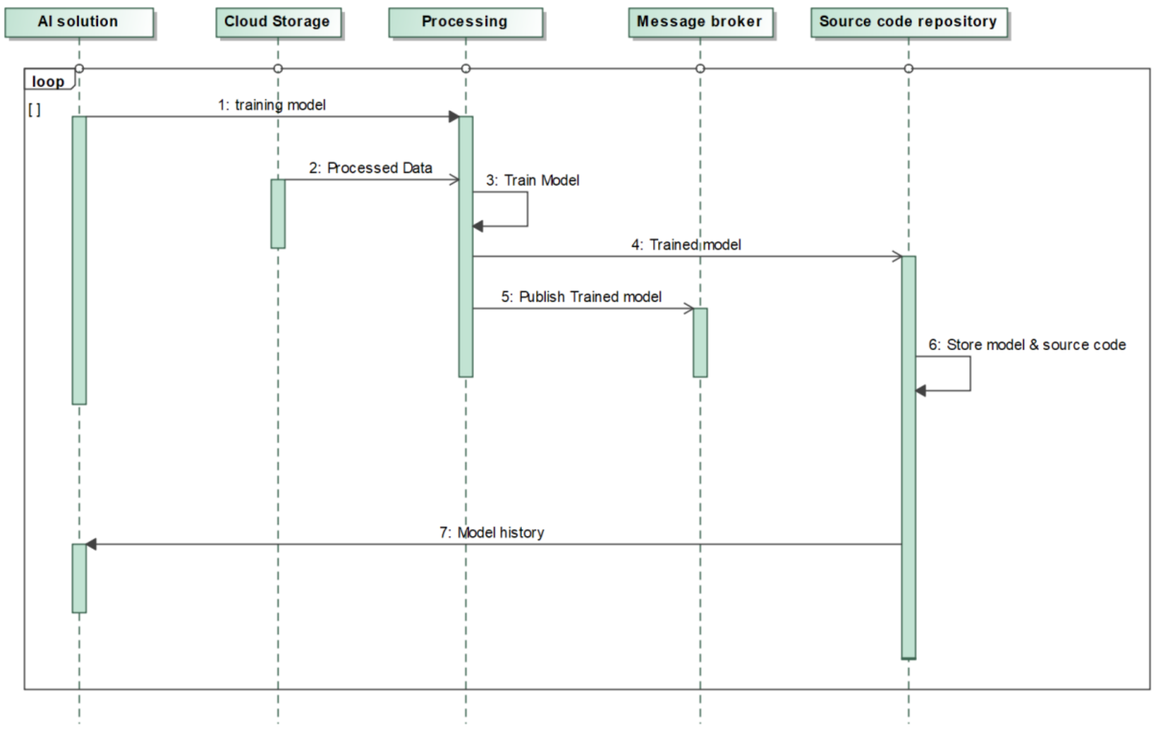

Figure 12 Dynamic model training sequence

If the chosen training model is dynamic our system follows the above sequence. The training model from AI solution is sent to the processing where training takes place and the trained model are published to message broker and to the source code repository. Then the model history is sent to the AI solution for dynamic training. This process will be in a loop.

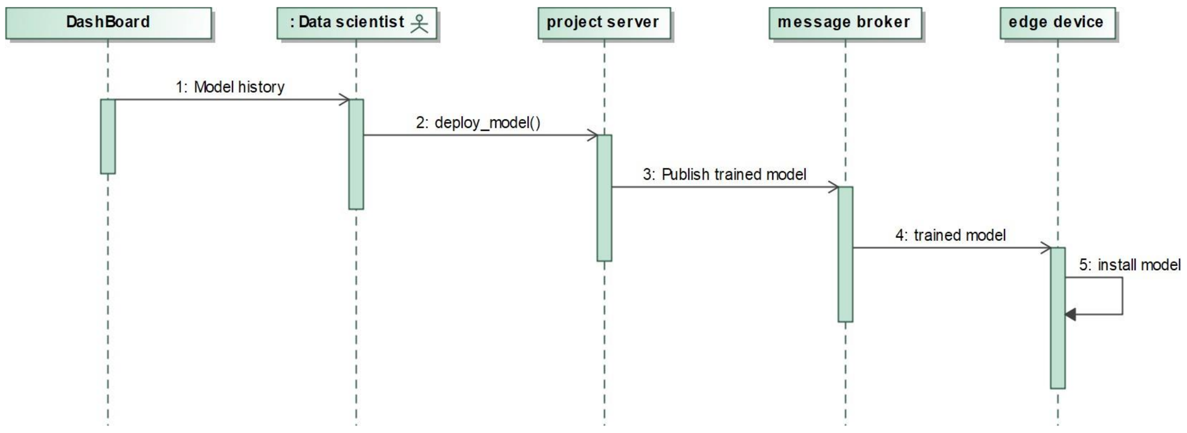

Figure 13 Deploying previously trained model sequence

The above diagram represents the deployment of previously trained model to the edge device. The dashboard sends model history to the data scientist. Data scientist decides whether to deploy the model or not. Once the model is sent to the project server, it publishes the model to message broker and message broker will deploy the model on the edge device.

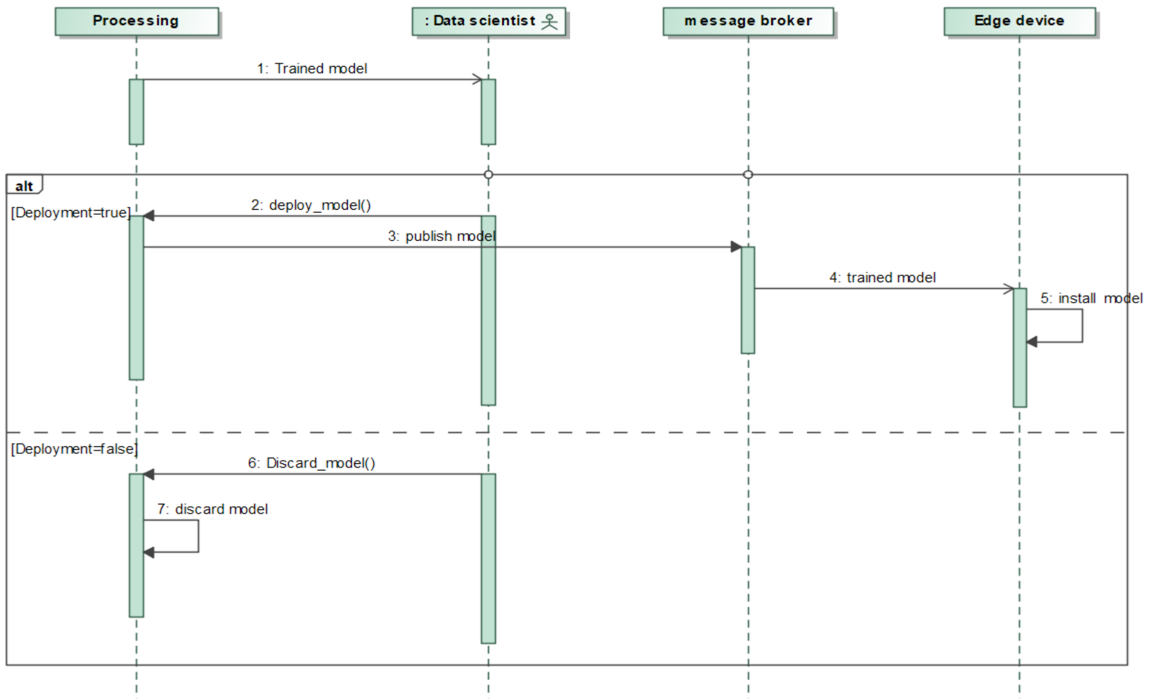

Figure 14 Deploy newest training model sequence

The above diagram represents the deploying of newly trained model to the edge device. The processing unit sends the trained model to the data scientist. Data scientist decides whether to publish the model or to discard the model based on accuracy of the model. If yes, he then, publishes the model to message broker and message broker will deploy the model to the edge device.

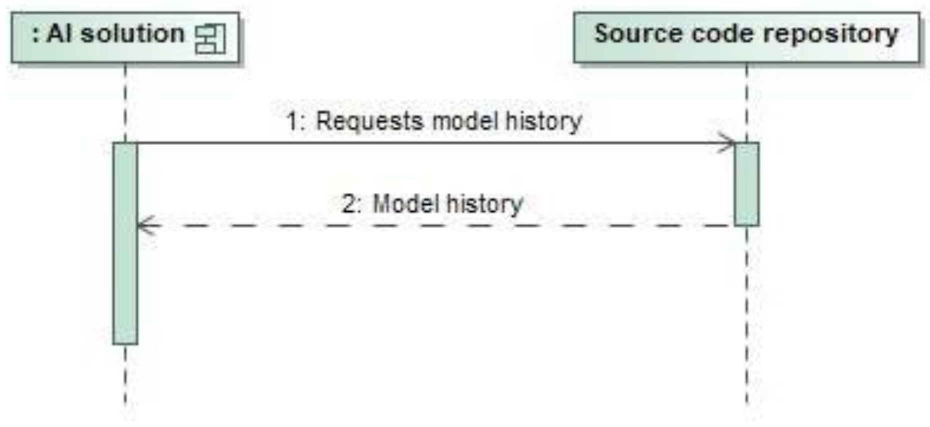

Figure 15 Request model history sequence

The above diagram represents the process of requesting the model history by AI solution from source code repository. The model history sent by the repository to the AI solution for training the model.

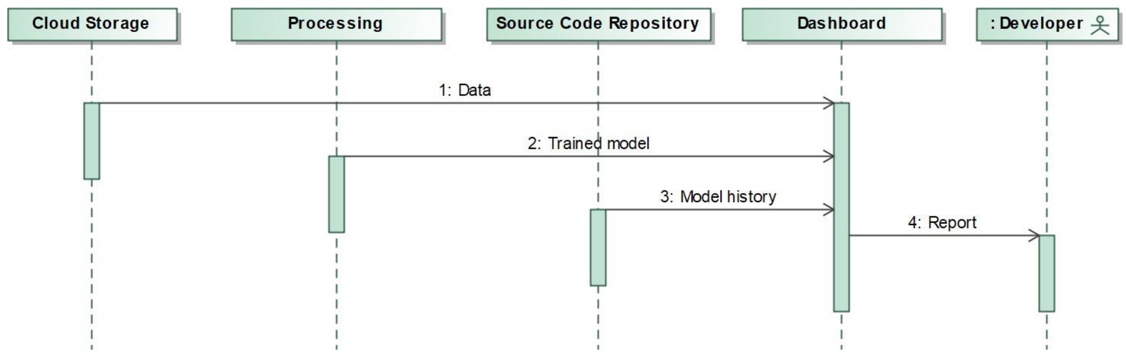

Figure 16 Monitoring of system sequence

The above diagram represents the monitoring process of the system. Cloud Storage sends the data to the dashboard. The processing unit will send trained model and source code repository will send model history to the dashboard. The dashboard generates reports for the developer. With the help of those report the developer can monitor the system.

Software architecture

Code and description at: https://github.com/niharikagouda/software-architecture-project/blob/master/Software%20architecture%20project.zip in raw form in zip.

Summary

Weighing all the pros and cons of the architecture, we have finalized the architecture which is multi-layered architecture and cloud-architecture along with pub-sub messaging system which covers all the requirements.

Architecture style/styles that can be used in our scenario

With a long discussion our team identified the collaboration of cloud architecture with layered architecture which will serve good for given scenario. The detailed explanation of why we chose this process is explained in the Architectural pattern’s topic of the document. We can use microservice also instead of cloud architecture, but we use cloud architecture for the following reasons.

1. **Fresh Software:** Immediate upgrades to new features and functionality into workers hand to make them more productive.
2. **Do more with less:** The reduction in numbers of servers, data centers, software cost and the number of staff can significantly reduce IT costs without impacting an organizations IT capability.
3. **Flexible cost:** The cost of cloud computing is flexible than traditional methods. Companies pay only for server and infrastructure capacity.
4. **Always-on availability:** The connection is always on and as long as workers have an connectivity to the local Wi-Fi network to publish the data and the workers will have the edge device that will always be able to predict the health status since the models are trained and installed directly on the device.
5. **Improved mobility:** Data and applications are available to employees no matter where they are in the world. Workers can take their work anywhere via smart phones and tablets.

6. **Improved collaboration:** Cloud applications improve collaboration by allowing groups of people to meet virtually and easily share information in real time.
7. Cloud computing is more cost effective since it depends on the usage.
8. **Flexible capacity:** Cloud is the flexible facility that can be turned up, down or off depending upon circumstances.
9. **Facilitate M&A activity:** Cloud computing accommodates faster changes so that two companies can become one much faster and more efficiently.

The reason why we haven’t chosen microservice architecture are given bellow

1. As the application grows, so does the associated code base, which can overload your development environment each time it loads the application, reducing developer productivity.
2. Because the application has been packaged in one EAR/WAR, changing the technology stack of the application becomes a difficult task. With this kind of architecture, refactoring the code base becomes difficult because it’s hard to predict how it will impact application functionality.
3. Scaling monolithic applications can be accomplished by deploying the same EAR/WAR packages in additional servers, known as horizontal scaling. Each copy of the application in additional servers will utilize the same amount of underlying resources, which is inefficient in its design.
4. **Communication between services is complex**: Since everything is now an independent service, we must carefully handle requests traveling between our modules. In one such scenario, developers may be forced to write extra code to avoid disruption. Over time, complications will arise when remote calls experience latency.
5. **More services equal more resources**: Multiple databases and transaction management can be painful.
6. **Global testing is difficult**: Testing a microservices-based application can be cumbersome. In a monolithic approach, we would just need to launch our WAR on an application server and ensure its connectivity with the underlying database. With microservices, each dependent service needs to be confirmed before testing can occur.
7. **Debugging problems can be harder**: Each service has its own set of logs to go through. Log, logs, and more logs.

8. **Deployment challengers**: The product may need coordination among multiple services, which may not be as straightforward as deploying a WAR in a container.

Pros and Cons:

Use of Rest API

The main advantage of using Rest API is it is so flexible that it can increase request by handling large loads of incoming and outbound calls. And it suits very well for our system.

Using the system in APP

By analyzing the advantage and disadvantage which are given bellow we can decide whether to use our system in mobile application are not.

Advantage of system in App

- Easy, Faster and instance access to the system

We can easily access or know our health condition instantly when we are connected to the internet.

- Using device features for gaining more data

We can make our app to utilizing various features of native device like Camera that can scan PDFs, GPS to know the location of the user.

- We can give instant notification to the user

In App we can push the notification or alert the user when he wants to prevent him from getting any disease.

- Productive improvement and cost reduction (marketing point of view).

App helps the company to expand their audience reach in a very short time by reducing marketing costs.

Disadvantage of the App and how to manage that?

- There is a high chance of security risk

Every time we need a TLS/SSL encryption for communication to protect our app from the external threads.

- Poor authorization and authentication

These vulnerabilities are established mostly on the server side. We can overcome that by ignoring device identifiers ought and not sending the out-of-band authentication tokens to the related device.

- Client-side injection

This category has consisted of a broad diverseness of input strikes against the application itself. General best practices for reduction of client-side injection drawbacks cover the input validation of the app entry points, on the server side. To avoid this, you should use parameterized queries, disable file system access for Web views, JavaScript and plugin support for Web views.

- Wrong Session handling

While session handling mechanisms are largely applied at the server side of apps, secure session management practices can be used in devices themselves. The Confidentiality & Integrity of session tokens should be preserved via TLS/SSL connections. Like authorization & authentication, device identifiers should be avoided here as well, and you should execute safe mechanisms to cancel session on lost devices.

- Side channel data leakage

This comprises of data exchange that normally maximizes the app performance. As with Weak Data Storage, you should develop your app under the premise that the device might be taken. The application should be dynamically examined in order to prove that it does not leak the data while runtime.

Model is pushed in the database that massively scores it on the relevant dataset

Advantage of scoring the model using a relevant dataset

1. Licensing costs may be reduced.
2. Easier Maintenance
3. Common connectivity and resource pooling would come in handy when implementing global caching.
4. Re-usability: Not only the platform can be reused but sometime the data can also be reused.

Disadvantage of scoring the model using a relevant dataset

1. **Security:** losing the data or breaking into the data will expose all data together to threats
2. **Data rollback** can be very difficult

Asynchronous via messaging: Implement a pub sub architecture where a stream of messages will be evaluated on the fly

We have listed the advantages of using Pub-Sub in our system bellow and we haven’t found any valid reason to omit Pub-Sub in our system.

Advantage of using pub-sub in our system

1. Eliminate Polling

Pub-Sub allow instantaneous, push-based delivery, eliminating the need for message consumer to periodically check for new information.

2. Dynamic Targeting

Pub-Sub makes discovery of service easier; Publisher will post a message to the topic and any the device which need the data will subscribe its endpoint to the topic and start receiving messages. Subscribers can change, upgrade, multiply or disappear and the system dynamically adjusts.

3. Scale Independently

Publishers and subscribers are decoupled and work independently from each other, which allows us to develop and scale them independently. We can decide to handle orders one way this month, then a different way next month.

4. Simplify Communication

The Publish Subscribe model reduces complexity by removing all the point-to-point connections with a single connection to a message topic, which will manage subscriptions to decide what messages should be delivered to which endpoints**.**

References

1. Cloud Architecture:

- https://arxiv.org/pdf/1306.4956.pdf
- https://www.researchgate.net/profile/Shehnila_Zardari/publication/228578811_Cloud_adoption_A_goal-oriented_requirements_engineering_approach/links/0c96053a2105094fee000000/Cloud-adoption-A-goal-oriented-requirements-engineering-approach.pdf

2. Advantage and disadvantage of cloud architecture.

- https://cloudacademy.com/blog/microservices-architecture-challenge-advantage-drawback/
- https://arxiv.org/pdf/1306.4956.pdf

3. Advantage and disadvantage of microservice architecture.

https://cloudacademy.com/blog/microservices-architecture-challenge-advantage-drawback/

4. Layered Architecture:

Book: Software Architecture Patte by Mark Richards

Publisher: O'Reilly Media, Inc.

Release Date: February 2015

ISBN: 9781491971437

5. Disadvantage of mobile application : https://ieeexplore.ieee.org/document/5640893

6. Advantage of mobile application: https://ieeexplore.ieee.org/document/6530464

7.Pub-Sub Services

https://ieeexplore.ieee.org/document/6782663

https://aws.amazon.com › pub-sub-messaging › benefits

8. Shared Database

https://www.quora.com/What-are-the-pros-and-cons-of-a-Shared-Database-in-Enterprise-Application-Integration