JIRAE, Vol. 4, No. 2, October 2019, 51-56
e-ISSN 2407-7259
DOI: 10.9744/jirae.4.2.51-56

Job Management System for Cell Monitoring Services

Harry Ariyanto Putra
Department of Informatics, Petra Christian University, Surabaya, Indonesia
Information and Communication Technology, Fontys University of Applied Sciences, Eindhoven, The Netherlands
harryariyantop@gmail.com

Abstract. OWOW is a digital agency which operates in the context of software development, design, and digital marketing. One of the key services it provides is full-stack web development. OWOW separates its full-stack web development into back-end and front-end. OWOW had done some automated testing in back-end, but none in front-end. This was not ideal because the quality of web apps developed could not be easily and thoroughly ensured. OWOW believes Test Driven Development (TDD) might be the solution for the situation. TDD is a software development process where test code is written before the implementation code. Through this research, OWOW would like to start applying TDD into its full-stack web development. Research had been done to investigate how to apply TDD in OWOW's current workflow. In the research, the TDD approach was defined. Afterwards, its application on current back-end and front-end development was investigated. Furthermore, its relation to Continuous Integration was explored. The research findings were then implemented on an ongoing web app project called RentIt. This implementation had been delivered as a proof-of-concept application of TDD in OWOW's full-stack web development. It is concluded that OWOW had been introduced to apply TDD in its full-stack web development.

Keywords: Tracking, management system, azure, processes, cloud.

1. Introduction

Cells have always been mysterious. They are the smallest unit of life and each of them is a separate individual. CytoSmart has developed a device called the Omni to help reduce the task of the cell researchers. The Omni provides high-quality images of the cells cultures (snapshots) and stores them in the cloud environment, where the images will be processed with algorithm selected by the user. The result, which consists of cell images processed by the selected algorithm and its related information, then will be available to the user. All these processes run in the cloud without any means to track and observe them. The stored information was scattered in the cloud storage. The operational team needs to gather and join the scattered information in the cloud manually to make a conclusion and to understand what happened to a specific process. Not only this action was redundant, as the operational team did this for each of the process, it was also becoming more difficult as the complexity and the size of the information in the cloud grows. Another problem that the current cell processing services has is that re-doing the experiment must always start from the beginning, which makes it time-consuming and less accurate since some time has passed and the cell might not be in the condition that the researcher desired anymore.

These reasons are why the Job Management System is required. The system will provide a way to track and manage the process that is happening in the cloud and provide an ability to re-do the experiment. Well implemented management system can increase company performance, by reducing maintenance cost [5] and execution time, while using the optimum value [11].

This paper describes the Job Management System that was implemented for the cell processing services and the researches that were done to achieve the desirable outcome.

2. Research Results

There are several research works that needs to be worked on before solving the problem. Those work items are defined as sub-points following this paragraph.

2.1 Omni Workflow

The whole procedure was started by customers filling the required parameters, such as algorithm and interval of the scans, into a software application provided. Then, the Omni starts taking snapshots and send them to the cloud.

1. After all the snapshots are received, the snapshots are combined into one big image (Stitching)
2. Depending on the type of the well plates, the wells are extracted into another set of images, which only contain one well per image (Well Detection)
3. The images will be processed based on the algorithm selected by the customers.
4. The results are stored in the cloud. The customers can access the result from the specified website.
5. The experiment continues to run until the customers stop it manually.

Internally, the process of receiving the snapshots is called Receiving, the process of stitching and well detection considered as Preprocessing, and the process of running the algorithm on the images is called Processing. These processes make up one scan. One experiment consists of multiple scans.

2.2 Current Workflow Problem

Wisdom can be achieved after proper amount of knowledge obtained from meaningful information provided [1]. Only by obtaining wisdom and knowledge regarding what is the situation in the cloud, the problem can be solved correctly, as the person truly understand the problem and then the solution can be provided.

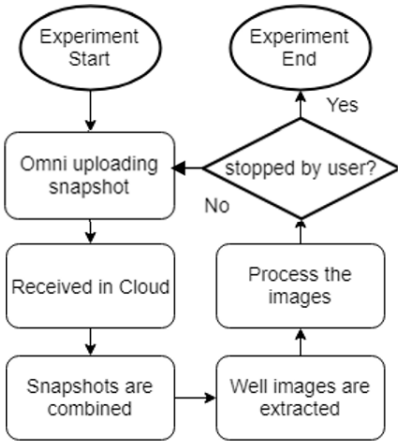

Figure 1. Current Omni Workflow

The workflow implemented was rigid and could not support features that the company wanted to introduce. The company would like to introduce the re-do (or can also be called as **reprocess**) feature, which enables the customers to re-do their past experiment, and also the area selection, which enables the customers to analyze the images based on their own custom area, instead of the whole well plate. In general, these are the problems with the current system:

1. The data of the previous scan will be overwritten if the reprocess feature is being run
2. Can't accept custom area(s), which the customer desire greatly
3. Can't change the algorithm after the experiment starts.

For the desired workflow (shown in Figure 2), the company wants to add another flow as an addition to an existing one, where the customers can select their own areas to process. The company already defined and decided to go with the desired workflow to support its newly added features, reprocess and area selection.

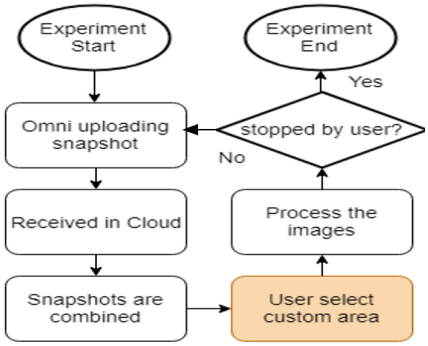

Figure 2. Desired Workflow

2.3 Job Management System

The Job Management System was defined to solve the current workflow problem. Job is the link between the required entities. There are three types of job that were used. They are scan job, area job, and experiment job.

1. Scan job is a link between the scan and the result of each scan's images that are processed based on the selected algorithm. This way, reprocess feature can be supported, since now rather than only one result, each scan can have multiples results, and the data will not be overwritten. After the reprocess feature is enabled, the customers now have the ability to change the algorithm when they reprocess their experiment.
2. Area Job is a link between scan job and the area selected by the customers using area selection features. This way, the customer can specify more than one area, on which part of the well plates they want to process.
3. Experiment Job provides a link between an experiment and each scan job its related to. Experiment job can give an overview of how scan jobs progress over time.

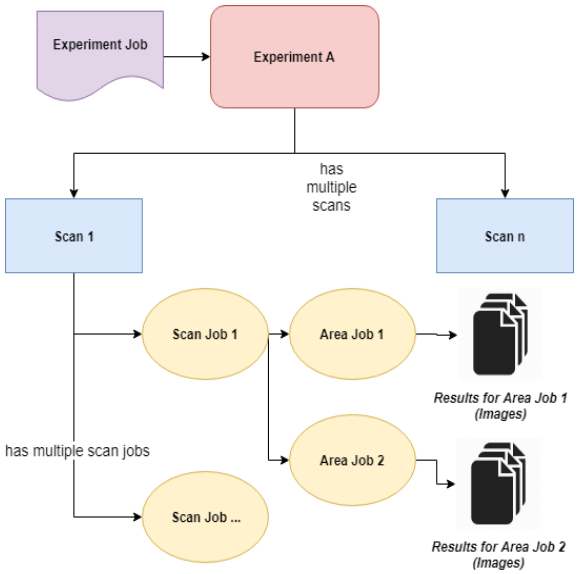

Figure 3. Relation between Job and current workflow

2.4 Tracking Scope

The scope of the tracking included the three main processes, which consist of receiving, preprocessing, and processing. These processes run on a cloud environment, where for every main process, the state of each scan and experiment are collected to a different type of databases/ storages. The status of the Omni devices itself also needs to be tracked to make sure that the devices are working properly. Besides that, Azure service bus, which serves as a messaging platform between processes, also needs to be tracked.

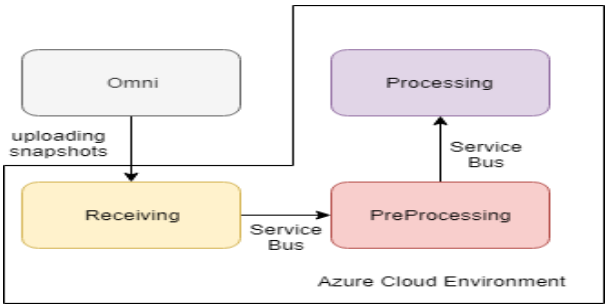

Figure 4. Tracking Scope

2.5 Application Frameworks

Back-End Framework

There are several criteria that needs to be fulfilled to decide the best framework for the system:

BE1 Must be a .Net-based framework
BE2 Already used by the company
BE3 Supports multi-platform (Windows, MacOs, and Linux)
BE4 Should be easy to maintain
BE5 Must have a high-performance rate

Based on criteria BE1, the choices are already limited to two options, which are:
a. .Net Framework
b. .Net Core

Criteria BE2 is not enough to decide which framework to be used since both are being used. When it comes to the third criteria (BE3), .Net Core has the ability to support cross-platform integration [6], whether it is for the application itself or its development environment. This means that .Net Core application can be run and build on multiple operating systems, such as MacOS and Linux, where .Net Framework couldn't. For the BE4, both of the options provide an easy way to access and maintain the packages and library needed for development. By using NuGet [10], a free and open source package manager owned by Microsoft, both of the web framework can be easily maintained and updated.

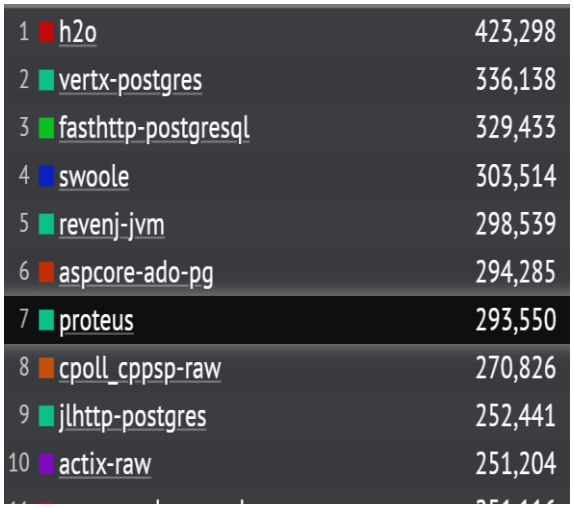

Figure 5. Top 10 Framework

Meanwhile, for the criteria BE5, .Net Core wins over .Net Framework by a large margin. On October 2018, Tech Empower (https://www.techempower.com), was running its 17th benchmark test for web frameworks. Based on a simple test, where the frameworks responded with a "Hello, World" message rendered as plaintext, .Net core, represented by 'aspcore', able to secure its seat on top 10 (Figure 12) with around 7 million response per second and with performance rate 99,7%. Meanwhile, .Net Framework, represented by 'aspnet-mono-ngx', only got the 285th place with around 5 thousand requests per second and performance rate around 0,1%.

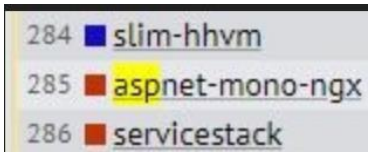

Figure 6. .NET Framework Position

As a result, the following table can be deducted (See Table 1) and .Net Core was chosen.

Table 1. Back-End Framework Comparison

	.NET CORE	.NET FRAMEWORK
BE1	✓	✓
BE2	✓	✓
BE3	✓	×
BE4	✓	✓
BE5	✓	×

Front-End Framework

There are several criteria needs to be addressed based on initial application design:
FE1 Used by the company
FE2 Maintainable
FE3 Lightweight
FE4 Has its own documentation
FE5 Component-Based Implementation

Based on FE1, the following alternatives were considered:
a. Pure JavaScript
b. Vue.js
c. React.js
d. jQuery

After conducting testing and prototyping on the alternatives mentioned above, Pure JavaScript and jQuery were not able to achieve the result needed for seamlessly displaying the necessary job information and handling the event triggered by certain activities, such as click action, between the time frame provided. In addition, these options will make the code less maintainable and hard to customize at later date (FE2). Regarding the size of the framework (FE3), results fetched from one of GitHub Repositories showed that Vue.js is the lightest one compared to other options [12]. Meanwhile, for the documentation (FE4), each option has its own documentation that can be accessed from the web, even some of them have tutorials. For the last requirement, most of the modern frameworks, such as Vue.js and React.js, support component-based implementation [9] (FE5), where the web component is reusable and can be easily integrated. The result of the research can be found in Table 2.

Table 2. Front-End Framework Comparison

	Pure JavaScript	Vue.js	React.js	jQuery
FE1	×	✓	✓	✓
FE2	×	✓	✓	×
FE3	N/A	58,8 KB	97-133 KB	82,34KB
FE4	×	✓	✓	✓
FE5	×	✓	✓	×

From Table 2, it can be observed that Pure JavaScript is not an option anymore when developing a complex application on the web. It needs help from a pre-set modern JavaScript framework so that the requirement can be fulfilled. In addition, due to its lack of maintainability and component implementation support, jQuery also is not an option. Between Vue.js and React.js, since Vue.js is the lightest, Vue.js was chosen.

2.6 Reprocessing

Re-do/Reprocessing is an ability to re-do the experiment or scans that had been done in the past. Reprocessing is important because it makes it possible to re-run a past experiment with a new algorithm using the same subject/cell cultures. This enables the customer to have another result from their past experiment. This functionality can be done by using the service bus to send a message to redo the experiment/scan to the preprocessing function. The message contain parameter needed to re-do the experiment such as the new algorithm that will be applied.

3. Implementation Results

The result of implementation is divided into 2 part based on its tracking scope, The Omni and The Cloud Environment. After the tracking system was done, it was followed by the implementation of the ability to reprocess experiments and additional features that can be added to enhance the application.

3.1 The Omni

Omni devices that are used to run the experiments need to be tracked since the whole process begins with the Omni uploading snapshots to the cloud and triggers the receiving process. The Omni produces log records that are stored on Log Analytics, each with its own timestamp and content. The record saved inside Log Analytics can be accessed by making an API calls with the required credentials to the Azure [3]. A sub-system was created to track this process, which contains the list of devices involved with an experiment and the log they generated during the receiving process.

3.2 The Cloud Environment

The cloud environment is where all the main processes of the Omni (receiving, preprocessing, and processing) happens. The communication between each process are supported in the cloud by using messaging services provided by Azure called Azure Service Bus [4]. To summarize it, for every main process, the status of the process that happens for the specified scan will be saved on a different type of databases which have their own functionality. To track what happened in the cloud environment, saved record from the databases can be utilized.

Experiment Table

For a better insight and a higher level of understanding of what is happening in the cloud, a new table which contains the list of experiments, where each experiment contains multiple scans, and each scan record contain a deeper insight, was added. The record needed was stored in Azure SQL Databases. During the implementation, it became clearer that a component-based structure for each table/item on the page would be more efficient and maintainable since the component can be re-used later. A system displaying the experiments and their details was created.

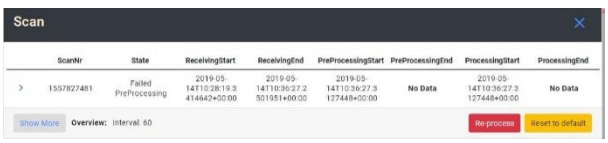

Figure 7. Experiment's detail

User Alert Table

This table's purpose is to display a list of messages or alert sent to the user regarding their experiment. The user also wants to see the scans record related to the specified alert. The reason why this table is needed is that the company wants to know whether the owner of the experiment is alerted if there is something wrong with their experimentBy utilizing the component that was made by before, alerts record can be displayed.

Service Bus Messages Table

In the cloud environment, communication between each process is handled by a messaging service provided by Azure, which is Azure Service Bus. Then, the next item to be tracked are the messages that are currently available on the queue available on the Azure Service Bus. The reason why this feature is needed because these queues serve as a way of communication between processes that are happening in the cloud. So, to know how many messages are currently waiting to be processed and also how many messages that can be retrieved is crucial for the company, since this way they can keep watch on what is the status of their communication infrastructure. After researching on how to fetch the data from the documentation of Service Bus feature provided by Azure [7], a sub-system, which contains lists of each queue available and the number of messages inside, either it is active or not, was created.

Container List Table

Most of the processing in the cloud happens in an isolated environment called container. These containers will be run when there is an image analysis process needs to be done and will be closed when there is none. Then, to make the tracking system more robust, the Container List table was made. Container List Table shows all the container that is currently running in the cloud and its detail/log. This table is needed to check if the container is running or not, and what are the logs that are being created when the container is running.

Cloud Orchestrarion Table

In the cloud environment, Log Analytics, a monitoring system provided by Azure, tracks and keep records of what is happening in the cloud environment as a whole. Retrieving the record saved by Log Analytics is very important since all of the logs generated during this process gives a first-hand insight regarding the process itself in the cloud. The record provided here covers all of the process (Receiving, Preprocessing, and Processing). But since there is a lot of records being generated at the same time and it takes a lot of time to fetch all of them, the company only wants to fetch the logs that contain error on it.

3.3 Reprocessing functionality

Reprocess is an ability to re-do an experiment that had been done in the past. By sending a customized message through API calls to Azure Service Bus [7], the functionality can be implemented. For testing purpose, Postman [2], an API Development Tools, was used to check whether the calls already has the correct parameter or not.

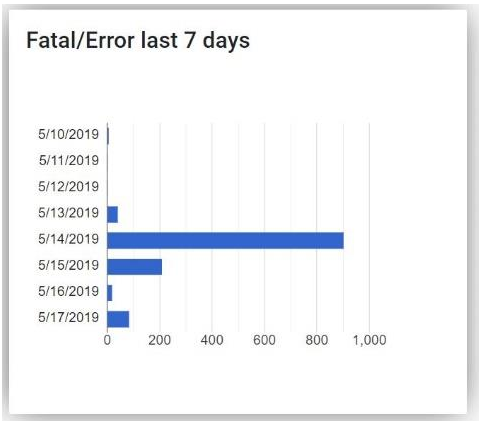

Figure 8. Error log fetched from Azure

3.4 Additional Features

During the implementation, several additional features were added to improve the application

Sort and Filtering

During the implementation, it was very hard to get a specific record. Since there was no way to group them and get only the wanted record. It was also difficult to search the record needed since the record was randomly fetched from the databases without proper ordering being set in place. So to counter the problem, a way to sort and filter the record was proposed. There were several solutions that were proposed to the company:

1. Both sort and filtering are done on the back-end and can be applied by API call
2. Sort happens on front-end side and filtering happens on back-end side

The second option was selected since sort features can be done by using Vue.js on the front-end and will reduce the time needed to fetch the data from the database.

Page Loading Speed Optimization

During implementation, whenever the application restarts or being refreshed on the web browser, the application took quite some time to fetch the data and populate the table. After doing some testing on Chrome [7], the concern was true. The reason this problem happened is that every time the page is loaded, it will fetch all the record from the database. To counter this problem, another parameter to the API call was added. This parameter decides the amount of record fetched, where only a selective amount of record will be fetched at first, and then the user can choose to show more of the record. As a result, with the same amount of requests, load time of the application decreased by 28 % (see Figure 9).

4. Conclusion

Cytosmart has developed a device called Omni, which has the ability to analyze cell cultures. Omni takes multiple images of cells and then process them according to an algorithm selected by users. Right now, the company does not have any solution to track the progress of each scan, which consists of the processes, that happens in the cloud and needs to dive into the related databases directly to know if the related processes failed or not. Besides that, to re-do the scan, the company needs to alter the databases directly and re-do the whole scan, including all its process, from scratch.

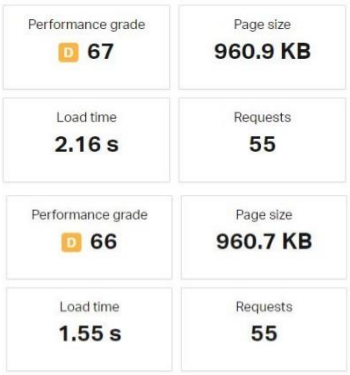

Figure 9. (top) Before optimization. (bottom) After optimization

To reduce the workloads of the operational team, a Job Management System was implemented, which was achieved by creating a web application to track the process that happens in the cloud and implements the ability to re-do the scan, failed or not.

By collecting information about the processes that happen in the cloud, technologies used by the company, and how to access the data related to the processes from the databases, the implementation of the job system and its reprocess functionality was implemented successfully. The status of the scans and their individual process were displayed. Furthermore, the logs record from the Omni devices was shown and the reprocess functionality was implemented successfully. To enhance the application, several additional features were added.

In the end, the goal of the assignment was achieved. The web application created provides a way for the company to track what happens in the cloud and reprocess a specific scan without the necessity to access the databases directly.

References

1. Ang, R. J. (2019). Use of content management systems to address nursing workflow. *International Journal of Nursing Sciences*, *6*(4), 454–459. doi: 10.1016/j.ijnss.2019.09.012
2. API Development Environment. (n.d.). Retrieved May 06, 2019, from https://www.getpostman.com/
3. Azure Log Analytics REST API. (n.d.). Retrieved May 02, 2019, from https://dev.loganalytics.io/
4. Azure Service Bus-Cloud Messaging Service | Microsoft Azure. (n.d.). Retrieved May 02, 2019, from https://azure.microsoft.com/en-us/services/service-bus/
5. Chen, C., & Tang, L. (2019). BIM-based integrated management workflow design for schedule and cost planning of building fabric maintenance. *Automation in Construction*, *107*, 102944. doi: 10.1016/j.autcon.2019.102944
6. Choose between .NET Core and .NET Framework for server apps. (2018, June 19). Retrieved April 30, 2019, from https://docs.microsoft.com/en-us/dotnet/standard/choosing-core-framework-server

7. Chrome Dev Tools | Tools for Web Developers | Google Developers. (n.d.). Retrieved May 06, 2019, from https://developers.google.com/web/tools/chrome-devtools/
8. Get started with Azure Service Bus queues. (2019, April 10). Retrieved May 03, 2019, from https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-dotnet-get-started-with-queues
9. K&C Team. (2019, February 08). Angular vs. Vue vs. jQuery vs. React vs. Ember. Retrieved April 30, 2019, from https://kruschecompany.com/blog/post/ember-jQuery-angular-react-vue-what-to-choose
10. NuGet Gallery | Home. (n.d.). Retrieved April 30, 2019, from https://www.nuget.org/
11. Serhani, M. A., El-Kassabi, H. T., Shuaib, K., Navaz, A. N., Benatallah, B., & Beheshti, A. (2020). Self-adapting cloud services orchestration for fulfilling intensive sensory data-driven IoT workflows. *Future Generation Computer Systems*, *108*, 583–597. doi: 10.1016/j.future.2020.02.066
12. Sizes of JS frameworks, just minified + minified and gzipped, (React, Angular 2, Vue, Ember). (2019, April 23). Retrieved April 30, 2019, from https://gist.github.com/Restuta/cda69e50a853aa64912d