Low Maintenance, Low Cost, Highly Secure, and Highly Manageable Serverless Solutions for Software Reverse Engineering

Kim Nguyen
nguyenkim@cityu.edu

Sam Chung
chungsam@cityu.edu

School of Technology & Computing
City University of Seattle
Seattle WA

Abstract

In recent years, serverless and cloud-based software has been gaining lots of attention from developers and companies. Everyone wants to adopt these futuristic technologies, individuals, or businesses alike. However, the common roadblocks in migrating to serverless or cloud-based architectures present enormous learning curves and management difficulties. As a result, many companies are hesitant to adopt these new technologies, despite their benefits. This research provides a low-maintenance, low-cost, highly secure, and highly manageable serverless solution to software reverse engineering that is reusable for different organizations. A full-stack serverless General Event Management System is provided as a Proof of Concept (POC). In addition, solutions to current serverless technologies strains, including practical technical projects and knowledge management practices, will be provided.

Keywords: Serverless, Software Reverse Engineering, Cloud Computing, Event Management System

1. INTRODUCTION

Traditionally, companies host applications locally or rent servers from other companies, which can be costly, high maintenance, less flexible, and challenging when migrating is needed. As a result, companies seek solutions that support rapid development, continuous integration, continuous deployment, and cost-effectiveness.

Serverless and cloud-based architecture came as solutions. In fact, according to a survey done on 160 businesses, "the positive impacts of adopting serverless are the event-driven architecture (51%), cost of resources (44%), speed of development (36%), the flexibility of scaling (31%), and application performance (19%)" (Esimann et al., 2021). However, businesses are reluctant to adopt these technologies (Akande, April, & Van Belle, 2013). Remarkably, many organizations have adopted serverless computing. Lenarduzzi et al. (2020) have found that maintenance costs increase after adopting serverless computing, with the leading cause being a lack of technical understanding. As a result, to have an effective serverless implementation, companies must be aware of "different management issues with cloud computing" (Akande, April, & Van Belle, 2013), such as "lack of standardization, customization, technology bottlenecks, strategy issues, etc." (Akande, April, & Van Belle, 2013, Abstract).

Several research studies have been done before to identify the challenges of serverless technologies, yet none has been able to solve those. In addition to technical difficulties, effective technical project management practices also play an essential role in successful serverless adoption. In fact, "One of the key factors influencing project success or failure is project management. Unfortunately, effective management of software projects is not in practice" (Manzil & Javed, 2007). As a result, this paper also suggests several project management software for effective collaboration as well as besides technical solutions. These software and practices can be applied to any technical or software development projects and are not limited to serverless or cloud-based software only.

Problem Statement

We address three aspects in this paper:

1. Providing low maintenance, low cost, highly secure, highly manageable solutions for businesses using serverless technology.
2. Addressing the challenges of serverless development.
3. Providing suggestions for effective technical project management and documentation.

Motivation

Software systems that are locally hosted tend to be costly and hard to maintain. There are many reasons why this situation happens, including employee rotations, missing documents, poorly written legacy code, and many more. This situation is unfortunately not specific to any organization. It is common in most organizations. On the other hand, real-time collaboration between developers can also be challenging due to the hosting method. With many factors combining, companies are motivated to adopt serverless. Based on research done on 89 different serverless applications, 47% of them adopted serverless for cost-saving purposes, while another 34% of them chose serverless to reduce developers' workloads on operational tasks. (Eismann et al., 2020).

On the project management side, it is recognized that developers don't enjoy documenting their works (Selic, 2009). As a result, technical documents are often lacking information, making it difficult for future uses. Besides writing good code, it is also essential that developers document their code well. Hence, having agile project management and knowledge tools to improve this problem is also necessary.

Accordingly, to support the findings of this research, we provide a well-architected software application that can be applied to any organization, despite the business sizes, operating systems, or business's focuses. At the same time, we want to share practical technical project and knowledge management methods that any organization can adopt. A General Event Management System is built as a proof of concept for this purpose.

Following are the four main motivations of this research:

- Encourage organizations to adopt cloud computing (Akande, April, & Belle, 2013).
- Reduce technical debts of serverless computing (Lenarduzzi et al., 2020).
- Increase ease of serverless testing and testability (Lenarduzzi et al., 2020).
- Improve technical project management and documentation (Selic, 2009).

2. BACKGROUND

Below are the explanations of the main concepts - reengineering, cloud computing, General Event Management System, low maintenance, low cost, highly secure, and highly manageable. "Reengineering is the most widely used approach to support evolutionary maintenance. It examines and alters a legacy system to reconstitute it in a new form". (Pérez-Castillo et al., 2011). There are many reasons why a system needs to be reengineered. It could be because of customers' needs, software improvements, etc. There are three stages in a reengineering process: "reverse engineering, restructuring, and forward engineering." (Pérez-Castillo et al., 2011).

According to the National Institute of Standards and Technology (NIST), "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." (Mell & Grance, 2011)

General Event Management System is a system or software where users can easily create and manage different events. Allowed actions include CRUD operation. This software is made generalized so that any organization can reuse its services.

Low maintenance refers to the ease of detecting, mitigating, and solving problems of any service, especially in emergency situations.

Low cost evaluates the cost of operation, specifically, the cost of resources and hardware.

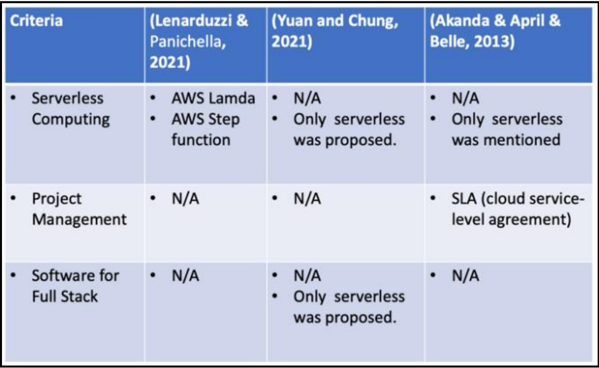
Highly secure refers to the overall security of the system and architecture, not only from the server-side but also from the client-side. Precisely, the application should come with authentication and secure data transfer.

Highly manageable should address the ease of managing the performances of serverless components. The high manageability could be through reports, analysis, logs, etc.

3. RELATED WORK

The implementation of cloud computing is not new, yet, not seasoned. Currently, Cloud Computing still has many rooms for developers to explore, leaving documentation, technical discussions, and solutions with unclear instructions or open questions. For example, to connect a Lambda Function to a database, there are many different ways. AWS official documentation alone has several suggested methods to achieve this action. Each has its pros and cons. The downsides of each solution are still being improved as things go. Thus, many benefits cloud computing can bring, as well as difficulties. Following is a summary and synthesis of three related works for this research.

As shown in Table 1, only the first research mentioned AWS Lambda and AWS Step Function in terms of serverless computing. However, no implementation was found. A similar situation applies to full-stack development, where only one research has proposed a serverless application, yet, no implementation was provided (Yuan & Chung 2021). In terms of project and knowledge management practice, none of the related works has touched these areas.



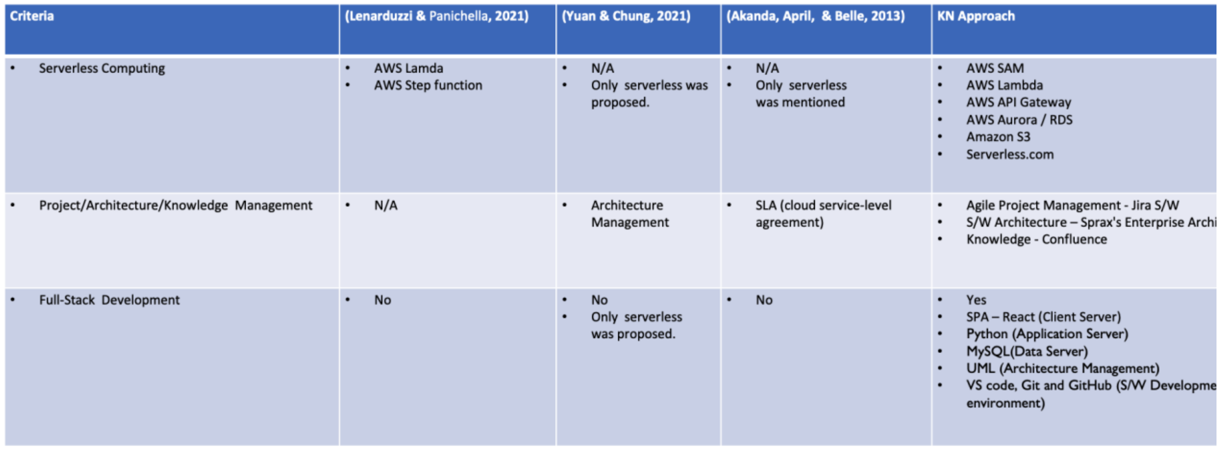
Criteria	(Lenarduzzi & Panichella, 2021)	(Yuan and Chung, 2021)	(Akanda & April & Belle, 2013)
• Serverless Computing	• AWS Lamda • AWS Step function	• N/A • Only serverless was proposed.	• N/A • Only serverless was mentioned
• Project Management	• N/A	• N/A	• SLA (cloud service-level agreement)
• Software for Full Stack	• N/A	• N/A • Only serverless was proposed.	• N/A

Table 1. Related Work Comparison

4. APPROACH

Despite mentioning some cloud-based software and serverless services, the related works mostly focused on identifying the shortcomings of serverless and cloud computing developments while suggesting the areas for further research or providing good practices rather than producing any solutions. Table 2 shows the comparisons between the related works and our approach called KN Approach.

In our approach, we address the issues of adopting serverless development, which was mentioned in the previous sections, by providing a full-stack, serverless and ready-to-use system. This solution can minimize the work and learning curves for organizations when adopting serverless. The system architecture was well designed and generalized so that any business model can easily adopt. At the same time, we propose project and knowledge management practices and software. All the technology stacks and software used in the KN approach are listed in Table 2.



Criteria	(Lenarduzzi & Panichella, 2021)	(Yuan & Chung, 2021)	(Akanda, April, & Belle, 2013)	KN Approach
• Serverless Computing	• AWS Lamda • AWS Step function	• N/A • Only serverless was proposed.	• N/A • Only serverless was mentioned	• AWS SAM • AWS Lambda • AWS API Gateway • AWS Aurora / RDS • Amazon S3 • Serverless.com
• Project/Architecture/Knowledge Management	• N/A	• Architecture Management	• SLA (cloud service-level agreement)	• Agile Project Management - Jira S/W • S/W Architecture – Sprax's Enterprise Archi • Knowledge - Confluence
• Full-Stack Development	• No	• No • Only serverless was proposed.	• No	• Yes • SPA – React (Client Server) • Python (Application Server) • MySQL(Data Server) • UML (Architecture Management) • VS code, Git and GitHub (S/W Developme environment)

Table 2. Related Work and Our Approach called KN.

Figure 1 below is the breakdown of the KN approach phases. As we reengineered an existing locally hosted system to a serverless system, we have broken the process down to five phases as below.

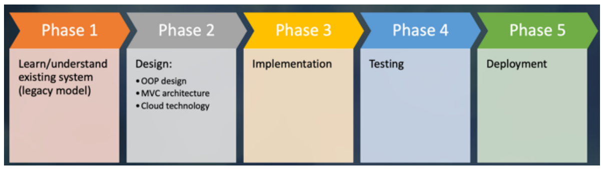
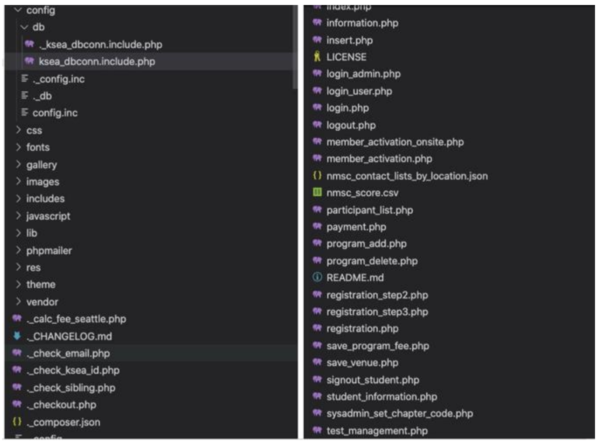


Figure 1. Five Phases of Our Approach

Phase 1 – Learn and Understand Legacy System: During this phase, a thorough understanding of the legacy system is necessary to effectively and correctly reengineer the system while maintaining the correct business logic. The legacy system we worked on is built mainly with PHP and uses Microsoft SQL for the database. The system is locally hosted and has a monolithic structure. Figure 2 shows the legacy system's source code, and Figure 3 shows the legacy system's architecture.



```
config
  db
    ._ksea_dbconn.include.php
    ksea_dbconn.include.php
  ._config.inc
  ._db
  config.inc
css
fonts
gallery
images
includes
javascript
lib
phpmailer
res
theme
vendor
._calc_fee_seattle.php
._CHANGELOG.md
._check_email.php
._check_ksea_id.php
._check_sibling.php
._checkout.php
._composer.json
```

```
index.php
information.php
insert.php
LICENSE
login_admin.php
login_user.php
login.php
logout.php
member_activation_onsite.php
member_activation.php
nmsc_contact_lists_by_location.json
nmsc_score.csv
participant_list.php
payment.php
program_add.php
program_delete.php
README.md
registration_step2.php
registration_step3.php
registration.php
save_program_fee.php
save_venue.php
signout_student.php
student_information.php
sysadmin_set_chapter_code.php
test_management.php
```

Figure 2. The Legacy System's Source Code

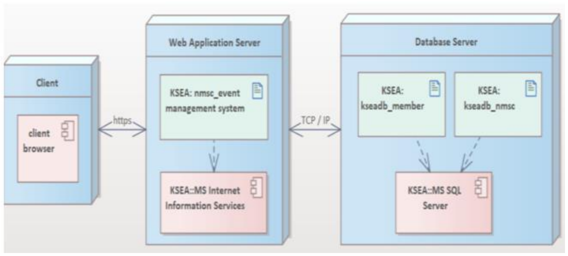


Figure 3. Legacy System (Yuan & Chung, 2021)

Phase 2 – Design: The goals for the target system are multi-tier architecture, serverless, and easy to include mobile development in the future. Figure 4 below is the architecture of the target system.

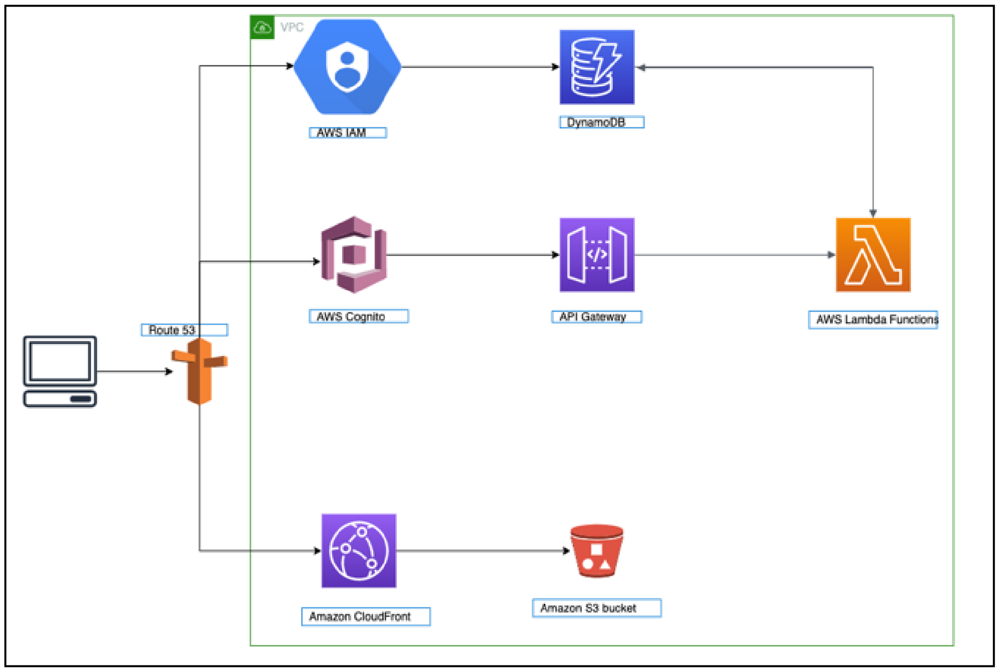
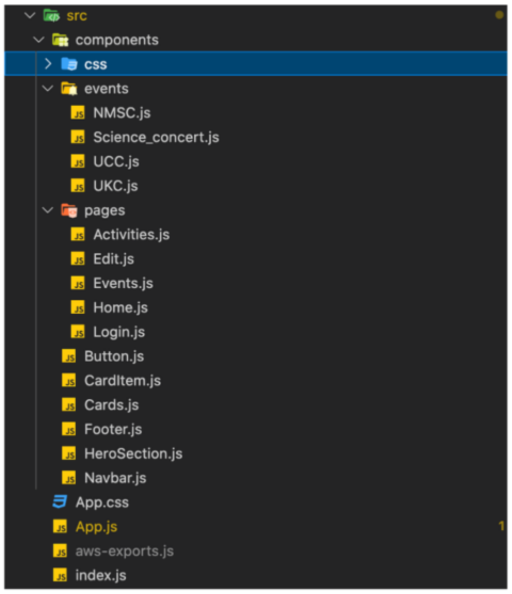


Figure 4. The Target System with Serverless Architecture.

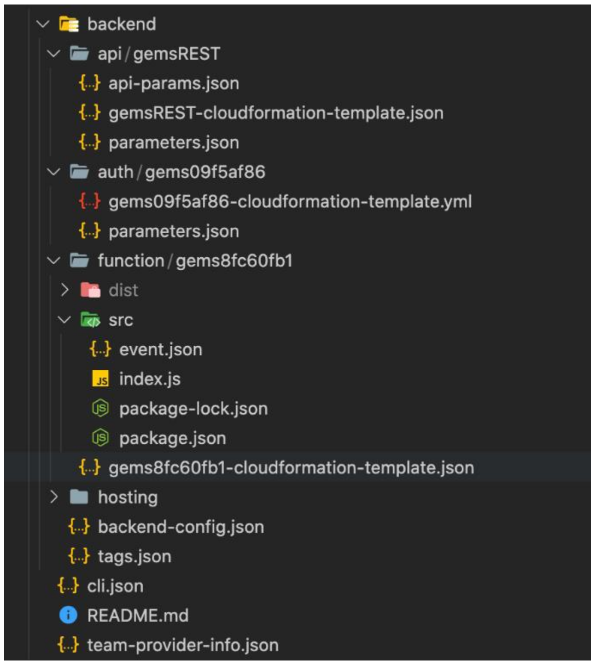
Phase 3 – Implementation: Once the learning of the legacy system and designing of the target system has been in place, it is time to implement. The target system's frontend is built with React, one of the popular web development libraries, as shown in Figure 5.



```
src
  components
    css
    events
      NMSC.js
      Science_concert.js
      UCC.js
      UKC.js
    pages
      Activities.js
      Edit.js
      Events.js
      Home.js
      Login.js
    Button.js
    CardItem.js
    Cards.js
    Footer.js
    HeroSection.js
    Navbar.js
  App.css
  App.js
  aws-exports.js
  index.js
```

Figure 5. The Target System's Frontend Source Code Structure

The backend component in Figure 6 is built using Amazon API Gateway, AWS Lambda Functions, AWS Cognito (used for authentication), and AWS Amplify. The programming languages used are Python and JavaScript (runs on NodeJS environment). The database used is Amazon DynamoDB – a NoSQL database.

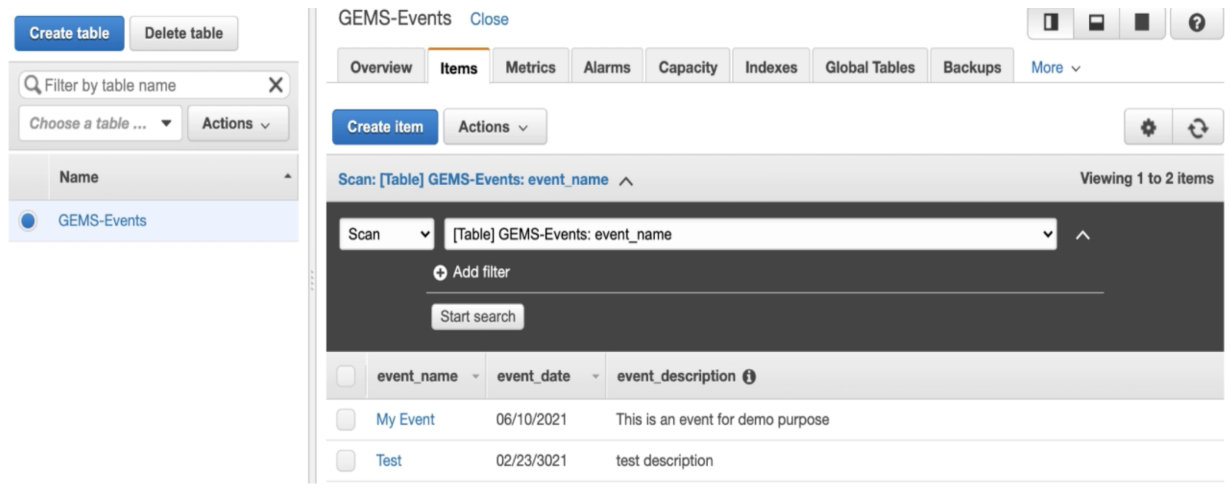


```
backend
  api / gemsREST
    api-params.json
    gemsREST-cloudformation-template.json
    parameters.json
  auth / gems09f5af86
    gems09f5af86-cloudformation-template.yml
    parameters.json
  function / gems8fc60fb1
    dist
    src
      event.json
      index.js
      package-lock.json
      package.json
    gems8fc60fb1-cloudformation-template.json
  hosting
  backend-config.json
  tags.json
cli.json
README.md
team-provider-info.json
```

Figure 6. The Target System's Backend Source Code Structure

By using DynamoDB, the target system is fully serverless. Figure 7 shows a snapshot of the database used in the demo product.

Phase 4 – Testing: Testing of the target system development is done constantly and alongside development. When changes are made to the source code or architecture design, testing is done right away. The testing is done in the Visual Studio Code environment, AWS console, and Amplify environment.



Scan: [Table] GEMS-Events: event_name

Viewing 1 to 2 items

Scan [Table] GEMS-Events: event_name

event_name	event_date	event_description
My Event	06/10/2021	This is an event for demo purpose
Test	02/23/3021	test description

Figure 7. The Target System's DynamoDB Database

Phase 5 – Deployment: AWS Amplify is used to deploy both frontend and backend, where frontend source code is hosted in a private S3 bucket, and backend functions are deployed to AWS Lambda. See Figure 8.

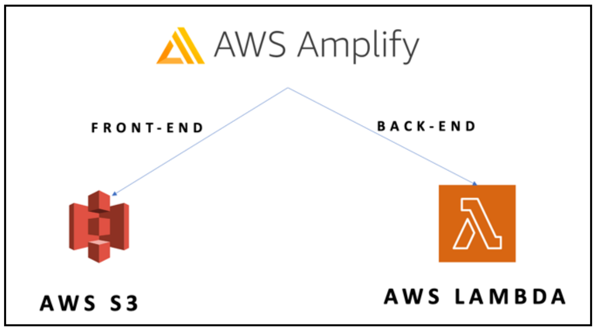
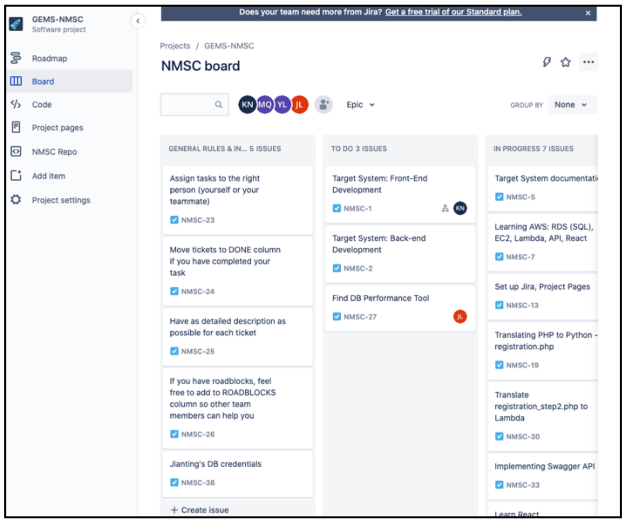


Figure 8. The Target System's Deployment Method

5. DATA COLLECTION

For the software side, data collection for this research is done by comparing the legacy system and the target system to draw out the improvements, pros, and cons of the re-designed system.

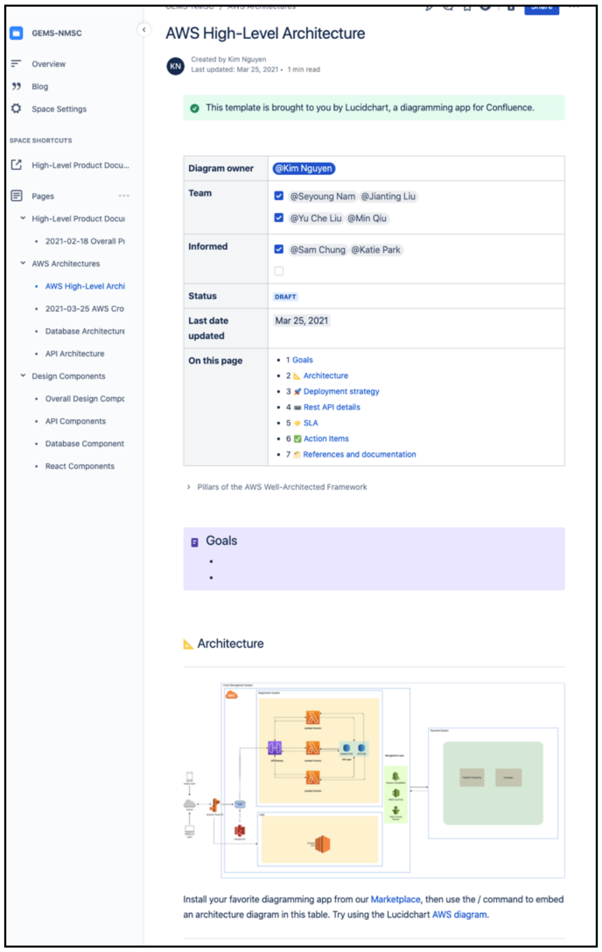
For the project management side, we collect data through user's satisfaction from using Jira and Confluence. Below are examples of Jira board and Confluence documents used during implementation. See Figures 9 and 10.



NMSC board

GENERAL RULES & IN... 5 ISSUES	TO DO 3 ISSUES	IN PROGRESS 7 ISSUES
Assign tasks to the right person (yourself or your teammate) NMSC-23	Target System: Front-End Development NMSC-1 KN	Target System documentati NMSC-5
Move tickets to DONE column if you have completed your task NMSC-24	Target System: Back-end Development NMSC-2	Learning AWS: RDS (SQL), EC2, Lambda, API, React NMSC-7
Have as detailed description as possible for each ticket NMSC-25	Find DB Performance Tool NMSC-27 JL	Set up Jira, Project Pages NMSC-13
If you have roadblocks, feel free to add to ROADBLOCKS column so other team members can help you NMSC-26		Translating PHP to Python - registration.php NMSC-19
Jianting's DB credentials NMSC-38		Translate registration_step2.php to Lambda NMSC-30
		Implementing Swagger API NMSC-33
		Learn React

Figure 9. Jira Software for Agile Project Management



AWS High-Level Architecture

KN Created by Kim Nguyen
Last updated: Mar 25, 2021 • 1 min read

This template is brought to you by Lucidchart, a diagramming app for Confluence.

Diagram owner	@Kim Nguyen
Team	☑ @Seyoung Nam @Jianting Liu ☑ @Yu Che Liu @Min Qiu
Informed	☑ @Sam Chung @Katie Park ☐
Status	DRAFT
Last date updated	Mar 25, 2021
On this page	• 1 Goals • 2 Architecture • 3 Deployment strategy • 4 Rest API details • 5 SLA • 6 Action Items • 7 References and documentation

› Pillars of the AWS Well-Architected Framework

Goals
•
•

Architecture

Install your favorite diagramming app from our Marketplace, then use the / command to embed an architecture diagram in this table. Try using the Lucidchart AWS diagram.

Figure 10. Confluence Software for Knowledge Management

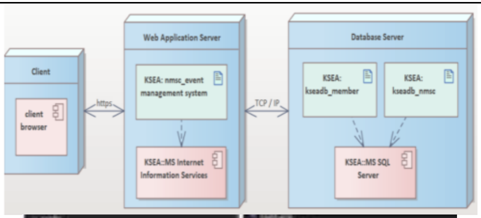
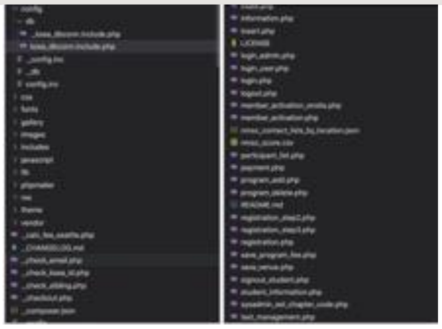
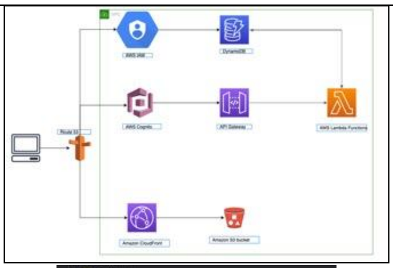
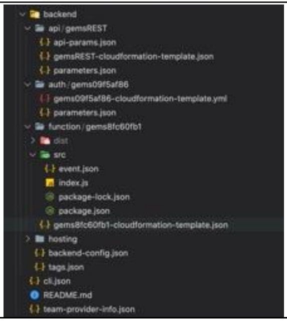
6. Data Analysis

The criteria used to analyze the data collected are listed below:

- Maintainability of software
- Cost of implementation, development, and maintenance.
- Security of software and environment.
- Manageability of Software
- Effectiveness of proposed project management and knowledge management practices.
- Scalability of software.
- Reusability of software, source code, and architecture.

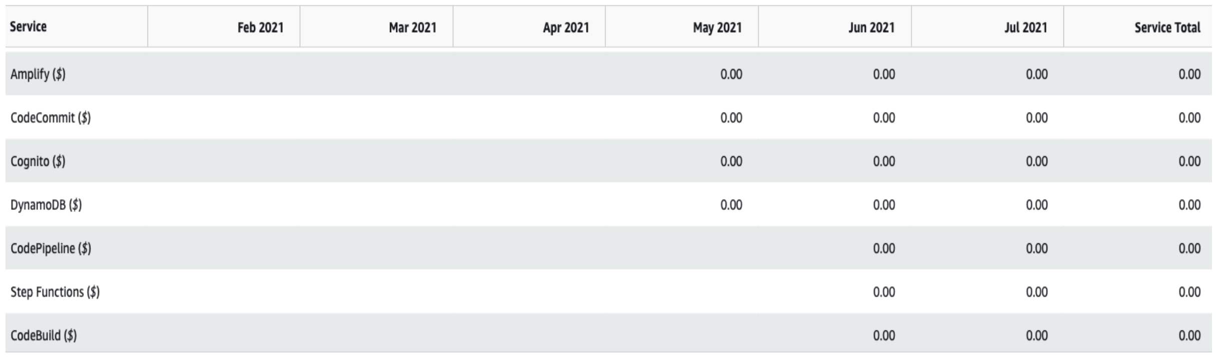
Let's look at the data analysis for each of the criteria listed above.

Low maintenance: the ease of detecting, mitigating, and solving problems in general and in case of any emergency, in particular, has increased remarkably. The reason being that the

 Client client browser https Web Application Server KSEA: nmsc_event management system KSEA::MS Internet Information Services TCP / IP Database Server KSEA: kseadb_member KSEA: kseadb_nmsc KSEA::MS SQL Server 	 AWS IAM DynamoDB Route 53 AWS Cognito API Gateway AWS Lambda Functions Amazon CloudFront Amazon S3 bucket 
Legacy System Architecture & Code structure	Target System Architecture & Code structure

```
∨ backend
  ∨ api / gemsREST
      {.} api-params.json
      {.} gemsREST-cloudformation-template.json
      {.} parameters.json
  ∨ auth / gems09f5af86
      {.} gems09f5af86-cloudformation-template.yml
      {.} parameters.json
  ∨ function / gems8fc60fb1
    > dist
    ∨ src
        {.} event.json
        index.js
        package-lock.json
        package.json
      {.} gems8fc60fb1-cloudformation-template.json
  > hosting
    {.} backend-config.json
    {.} tags.json
  {.} cli.json
  README.md
  {.} team-provider-info.json
```

Figure 11. Comparing Legacy System and Target System



Service	Feb 2021	Mar 2021	Apr 2021	May 2021	Jun 2021	Jul 2021	Service Total
Amplify ($)				0.00	0.00	0.00	0.00
CodeCommit ($)				0.00	0.00	0.00	0.00
Cognito ($)				0.00	0.00	0.00	0.00
DynamoDB ($)				0.00	0.00	0.00	0.00
CodePipeline ($)					0.00	0.00	0.00
Step Functions ($)					0.00	0.00	0.00
CodeBuild ($)					0.00	0.00	0.00

Figure 12. Operational cost of serverless services

Separation of Concerns (SoC) principle has been applied.

From Figure 11, putting the architectures and code structures of the legacy system and the target system side by side, it is clear that all the services of the target system are separated into different layers. Thus, if any problem or update happens, it is straightforward to navigate and identify the services that needed to be worked. Unlike the legacy system, where all the code are mixed together, making things hard to navigate, especially in emergencies, for example, when attacks happen.

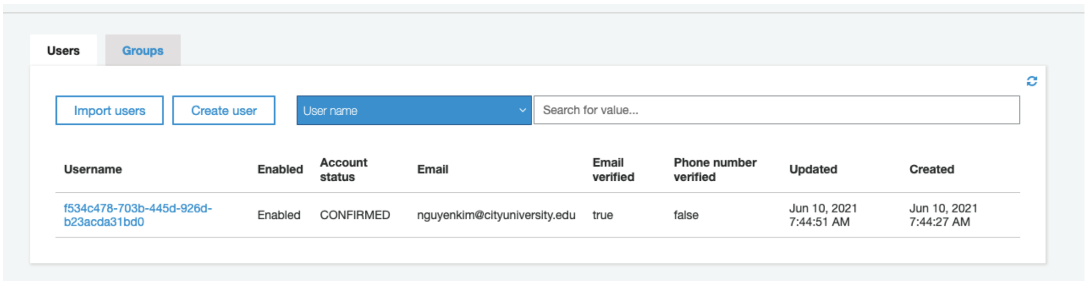
Low cost: AWS serverless computing pricing model is pay-as-you-go. If no resource is allocated, no charge should be occurred, as shown in Figure 12.

Besides not paying for unnecessary expenses, users can also eliminate hardware maintenance and management costs, thus, making applications' operations low, as mentioned in the Introduction section.

Highly secure: Besides the guaranteed security of the cloud provided by AWS, our solution uses Cognito – an authentication and access control service to ensure that only users with the right access right can connect and use the system.

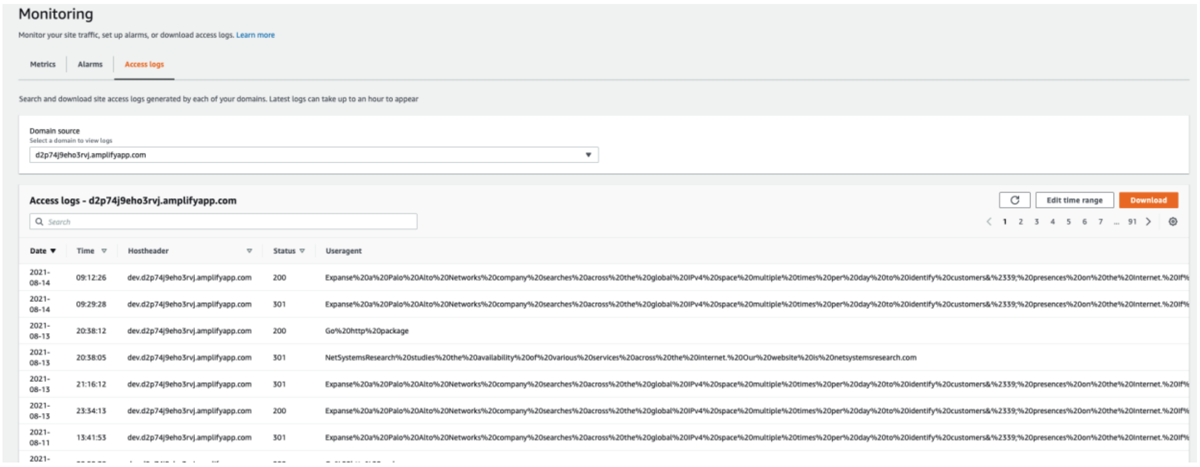
Figure 13 is the interface of the user pools. Once any user sign-up, they must verify their real email address to then be granted access to use the system.

Highly manageable: By using AWS and its serverless components, reports, analysis, alarms, metrics, and logs of services can be auto-



Username	Enabled	Account status	Email	Email verified	Phone number verified	Updated	Created
f534c478-703b-445d-926d-b23acda31bd0	Enabled	CONFIRMED	nguyenkim@cityuniversity.edu	true	false	Jun 10, 2021 7:44:51 AM	Jun 10, 2021 7:44:27 AM

Figure 13. Cognito User Pool interface



Monitoring

Monitor your site traffic, set up alarms, or download access logs. Learn more

Search and download site access logs generated by each of your domains. Latest logs can take up to an hour to appear

d2p74j9eho3rvj.amplifyapp.com

Access logs - d2p74j9eho3rvj.amplifyapp.com

Date	Time	Hostheader	Status	Useragent
2021-08-14	09:12:26	dev.d2p74j9eho3rvj.amplifyapp.com	200	Expanse%20a%20Palo%20Alto%20Networks%20company%20searches%20across%20the%20global%20IPv4%20space%20multiple%20times%20per%20day%20to%20identify%20customers&%2339;%20presences%20on%20the%20Internet.%20If%
2021-08-14	09:29:28	dev.d2p74j9eho3rvj.amplifyapp.com	301	Expanse%20a%20Palo%20Alto%20Networks%20company%20searches%20across%20the%20global%20IPv4%20space%20multiple%20times%20per%20day%20to%20identify%20customers&%2339;%20presences%20on%20the%20Internet.%20If%
2021-08-13	20:38:12	dev.d2p74j9eho3rvj.amplifyapp.com	200	Go%20http%20package
2021-08-13	20:38:05	dev.d2p74j9eho3rvj.amplifyapp.com	301	NetSystemsResearch%20studies%20the%20availability%20of%20various%20services%20across%20the%20internet.%20Our%20website%20is%20netsystemsresearch.com
2021-08-13	21:16:12	dev.d2p74j9eho3rvj.amplifyapp.com	301	Expanse%20a%20Palo%20Alto%20Networks%20company%20searches%20across%20the%20global%20IPv4%20space%20multiple%20times%20per%20day%20to%20identify%20customers&%2339;%20presences%20on%20the%20Internet.%20If%
2021-08-13	23:34:13	dev.d2p74j9eho3rvj.amplifyapp.com	200	Expanse%20a%20Palo%20Alto%20Networks%20company%20searches%20across%20the%20global%20IPv4%20space%20multiple%20times%20per%20day%20to%20identify%20customers&%2339;%20presences%20on%20the%20Internet.%20If%
2021-08-11	13:41:53	dev.d2p74j9eho3rvj.amplifyapp.com	301	Expanse%20a%20Palo%20Alto%20Networks%20company%20searches%20across%20the%20global%20IPv4%20space%20multiple%20times%20per%20day%20to%20identify%20customers&%2339;%20presences%20on%20the%20Internet.%20If%

Figure 14. Access Log of Serverless application

generated, making it much easier to manage the system from a holistic view. Figure 14 shows the access log of our application.

The effectiveness of proposed project management and knowledge management practices was increased due to Jira and Confluence software. Atlassian provides Jira task management and Confluence documentation features, where teams can keep track of each other's progress and having product documents all in one place. A source code management like GitHub can also be integrated.

Several software engineers have pointed out that Jira and Confluence brought a more positive impact on the documentation and task management of the software development process. For example, Min Qiu – a software engineer with over three years of experience, said, "Jira is helpful for our team to manage the development tasks, which means that we can see each team member's workflow and process. This can ensure that each sprint is under control and help the manager define the potential risks. During each sprint, we can put each functionality in the Confluence, so we can accelerate the feedback loop with inline comments on pages and files attached". He also shared his experience before and after using Jira and Confluence – "Before without Jira, we need to spend a lot of time assigning the task and communicating with each other that make sure what they are doing and how work is done so far. Furthermore, when we are dealing with the documentation, we need to create a shared folder to put everyone's documents there, and also not convenient to manage those things."

The scalability and reusability of software, source code, and architecture are much higher than the legacy system. As mentioned in the previous sections, the target system's architecture was designed intentionally with the plan in mind that any extra services can be added at any time. Specifically, since the application is serverless, with a backend built and run on AWS Lambda functions, if a mobile application needs to be added, companies can directly connect the Lambda functions with the mobile application without having to re-design the whole architecture.

7. FINDING

The problem this research tries to address is "How can we develop low maintenance, low cost, highly secure, and highly manageable solutions for software reverse engineering?". The answer lies in serverless computing, agile project, and knowledge management software. Specifically, the infrastructure and software include AWS serverless computing, AWS hosting and managing services, Atlassian project management, and Confluence documentation software.

As examined in the Data Analysis section, using AWS serverless' services and architecture, all the aspects of low maintenance, low cost, highly secure, and highly manageable solution are satisfied.

However, besides the benefits that cloud computing and project management software bring, some challenges remain. One of the biggest ones is the learning curve of cloud computing, which can often take months, not only for developers but also for other employees and operations of a company.

Accordingly, the findings of this research indicate that there are pros and cons to using serverless computing. Below are some pros to serverless development:

- Manageability, security, liability, maintainability, scalability, and speed of software are improved noticeably.
- Integration for future development, such as mobile development, is made more accessible.
- Cost is effective due to the pay-as-you-go model and serverless pricing.

However, some cons are remaining:

- Cloud computing requires steep learning curves.
- Lambda development comes with difficulties, especially when integrating Lambda Functions with other layers of the application.
- AWS Serverless Application Model (SAM) is a management tool that helps manage serverless services on AWS. However, it requires using the "YAML Ain't Markup Language" (YAML) files. Thus, it requires a remarkable amount of knowledge to learn since YAML is not a popular format. Especially if non-serverless services are needed. For instance, developing Infrastructure as Code (IaC) using YAML files can be harder to configure.
- Integrating different AWS services is challenging than the AWS official documentation put out to be.

8. CONCLUSION & FUTURE WORK

This research brings a more holistic view of serverless and cloud-based full-stack development to software reengineering. Low maintenance, low cost, highly secure, highly manageable serverless solutions are attractive and beneficial to companies. However, steep technical knowledge, software documentation, and other operations often come in as obstacles to obtaining such results. AWS serverless computing and Atlassian project management software are the answer to these difficulties. Despite the remaining challenges the software and architectures have, the benefits can outweigh, as long as the right approach and services are effectively and efficiently utilized, as suggested throughout this research.

However, we can still improve the provided solution to achieve higher ease of management and development. Instead of creating and managing separate services through the AWS console, the whole system's architecture can be designed and managed through only one YAML file in AWS SAM – a serverless management tool. For future work, the following can be considered to achieve such results:

- Integrate AWS SAM into the existing system of this research.
- Develop IaC using SAM YAML file.
- Integrate Code Commit to increase ease of collaboration between team members.

9. REFERENCE

Akande, A. O., April, N. A., & Van Belle, J. P. (2013, December). Management issues with cloud computing. In Proceedings of the Second International Conference on Innovative Computing and Cloud Computing (pp. 119-124). https://doi-org.proxy.cityu.edu/10.1145/2556871.2556899

Carver, J. C., Penzenstadler, B., Scheuner, J., & Staron, M. (2021). Insights for Serverless Application Engineering. IEEE Annals of the History of Computing, 38(01), 123-125. https://doi-ieeecomputersociety-org.proxy.cityu.edu/10.1109/MS.2020.3028659

Eismann, S., Scheuner, J., Van Eyk, E., Schwinger, M., Grohmann, J., Herbst, N., ... & Iosup, A. (2020). Serverless applications: Why, when, and how? IEEE Software, 38(1), 32-39. https://doi-ieeecomputersociety-org.proxy.cityu.edu/10.1109/MS.2020.3023302

Kim, W., Chung, S., & Endicott-Popovsky, B. (2014, October). Software architecture model driven reverse engineering approach to open source software development. In Proceedings of the 3rd annual conference on Research in information technology (pp. 9-14). https://doi.org/10.1145/2656434.2656440

Lenarduzzi, V., Daly, J., Martini, A., Panichella, S., & Tamburri, D. A. (2020). Toward a technical debt conceptualization for serverless computing. IEEE Software, 38(1), 40-47. https://doi.org/10.1109/MS.2020.3030786

Lethbridge, T. C., Singer, J., & Forward, A. (2003). How software engineers use documentation: The state of the practice. IEEE software, 20(6), 35-39. https://doi-ieeecomputersociety-org.proxy.cityu.edu/10.1109/52.41645

Li, J., Kulkarni, S. G., Ramakrishnan, K. K., & Li, D. (2019, December). Understanding open source serverless platforms: Design considerations and performance. In Proceedings of the 5th International Workshop on Serverless Computing (pp. 37-42). https://doi-org.proxy.cityu.edu/10.1145/3366623.3368139

Manzil, E. M., & Javed, T. (2007). Practicum in Software Project Management-an Endeavor to Effective and Pragmatic Software Project Management Education. https://doi-org.proxy.cityu.edu/10.1145/1287624.1287691

Mell, P., Grance, T. (2011). The NIST Definition of Cloud Computing. https://csrc.nist.gov/publications/detail/sp/800-145/final

Pérez-Castillo, R., de Guzman, I. G. R., Piattini, M., & Ebert, C. (2011). Reengineering technologies. IEEE software, 28(6), 13-17. https://doi-ieeecomputersociety-org.proxy.cityu.edu/10.1109/MS.2011.145

Selic, B. (2009). Agile documentation, anyone?. IEEE software, 26(6), 11-12. https://doi-ieeecomputersociety-org.proxy.cityu.edu/10.1109/MS.2009.167

Shi, Y., Meng, X., Zhao, J., Hu, X., Liu, B., & Wang, H. (2010, October). Benchmarking cloud-based data management systems. In Proceedings of the second international workshop on Cloud data management (pp. 47-54). https://doi-org.proxy.cityu.edu/10.1145/1871929.1871938.

Yuan, H. E., & Chung, S. (2021). A Case Study of Software Reengineering using Emerging Cloud Computing: A Non-profit's National Math Competition Event Management System. http://repository.cityu.edu/handle/20.500.11803/1027