Camosun College Capstone Project

Reporting and Dashboard Service Improvement

Final Report

Prepared for

Katie Tanigawa, Ben Leather, and Jonas Bambi

Prepared by

Samara Flueck
Sunghwan Park

Last Updated:	August 13, 2021
Document Version:	1.0.0

Memorandum

To: Katie Tanigawa, Instructor, ENGL 273

Cc: Ben Leather and Jonas Bambi, Instructors, ICS 290

From: Samara Flueck and Sunghwan Park

Date: August 13th, 2021

Re: BC Government OCIO - DIO Reporting and Dashboard Service Improvement - Final Report

We are pleased to submit our final report for the BC Government OCIO - Digital Investment Office's Reporting and Dashboard Service Improvement project.

Our sponsors have signed off on the project and have declared it to be an astounding success. We successfully designed a solution supported by a prototype application that replaces the current manual reporting process with a digital web application that supports automation and data integrity. This project has already been approved for funding for continued development into a fully operational application.

We have provided a high-level view of what we accomplished during this project as well as our recommendations for future development and priority features to be implemented. Links to additional documentation have also been provided where relevant.

Please review our final report and respond with any feedback you may have.

Encl: Final Report (1 copy)

Executive Summary

The BC Government OCIO - Digital Investment Office (DIO) sponsored this project to provide a unique educational experience to students by solving a real-world business problem of replacing a manual reporting process with a semi-automated one. This was a challenging but fulfilling project where we were able to gather requirements from users and then design a solution based on that research. We also were tasked with developing a prototype to demonstrate our solution, so this required strong technical abilities from all team members in addition to the business analysis and design skills required for the solution design. Team RDSI was able to complete this project not only to our sponsor's satisfaction, but also to a degree that allowed it to be approved for continued development.

Our application design is clean and simple so that even non-technical users can learn our application with minimal to no specialized training. It supports data integrity through the use of data validation and purpose-built submission forms. Our solution also provides an API that can be used to integrate the data collected into other BC Government systems. We hosted our prototype publicly on the BCDevExchange's OpenShift environment so that interested parties could see what our solution would look like when implemented.

Our team faced many challenges, but we were able to overcome and learn from them. The personal and professional growth each of us have seen in this project allows us to make informed recommendations on the future of this project and the ICS program at Camosun College.

Table of Contents

Project Description

Background

The Digital Investment Office (DIO) is a part of the BC Government and reports to the Office of the Chief Information Officer. The DIO manages capital investment funding for projects that aim to improve digital services for British Columbians. Ministries that are awarded funding must report on their projects performance to the DIO quarterly.

Purpose

The current reporting process consists of manually filling out excel spreadsheets and submitting them to the DIO via email. This manual process is time consuming and many of the fields are subjective or open to interpretation. This results in data captured that is difficult to aggregate and analyze due to data inconsistencies. Reviewers also spend a significant amount of time cross referencing documents for simple changes.

Hypothesis

Developing a modern web application that uses a digital form to capture reporting data to replace the current process will improve workflows and data quality. Manual tasks can be automated and report fields can be pre-populated with historical data to further reduce manual effort. Data validation can be used to assist the user completing a report and facilitate quality data collection. The data collected can then be stored in a central repository such as a database where it can later be surfaced to produce reports and analytics for decision makers.

Project Goals and Objectives

The overarching goal of the Capstone 2021 project team was to design a modern web application as a tool for the DIO to use that could replace spreadsheets as the primary project reporting method.

Objectives include:

- Apply modern application development methodology based on AGILE principles.
- Create a modern web application that is intuitive and easy to use.
- Store project and reporting information in a central repository such as a database.
- Host the solution in the BC Dev Exchange's OpenShift container environment.
- Provide handover documentation detailing how to operationalize the application.

List of Deliverables

The key deliverables for this project focus on designing a solution that uses a modern web application to replace the current manual quarterly reporting process. Development of a proof-of-concept prototype that can be handed over to another team for continued development should accompany the solution. Therefore, the team was expected to produce two key deliverables at the end of the project:

- Develop a functional prototype as a proof-of-concept.
- Author handover documentation for the next development phase .

Functional Prototype

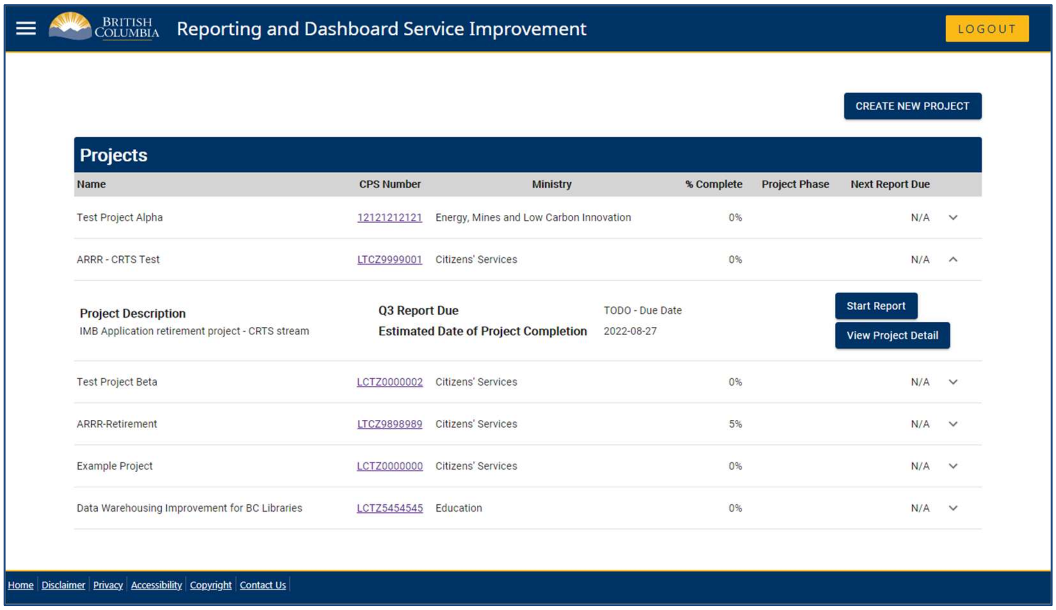
The primary purpose of the functional prototype was to demonstrate how our solution could be implemented in the BCDevExchange's OpenShift environment. The secondary purpose of the functional prototype was to be able to demonstrate our solution to stakeholders and decision makers to gain funding to continue this project. After our implementation phase ended, our sponsor demonstrated our prototype to decision makers. The executives were pleased with our work and the project was approved for continued development with the goal of creating a fully operational tool for the DIO to change how reporting processes are handled.

Our prototype focused on implementing three foundational user journeys:

- A Submitter creates a new project in the system.
- A Submitter completes and submits a quarterly report to the DIO.
- A Finance Analyst reviews a submitted quarterly report.

A public deployment of our prototype can be found at the following link:

https://rdsi-client.adccd1-prod.apps.silver.devops.gov.bc.ca/



Projects

Name	CPS Number	Ministry	% Complete	Project Phase	Next Report Due
Test Project Alpha	12121212121	Energy, Mines and Low Carbon Innovation	0%		N/A
ARRR - CRTS Test	LTCZ9999001	Citizens' Services	0%		N/A
Test Project Beta	LCTZ0000002	Citizens' Services	0%		N/A
ARRR-Retirement	LTCZ9898989	Citizens' Services	5%		N/A
Example Project	LCTZ0000000	Citizens' Services	0%		N/A
Data Warehousing Improvement for BC Libraries	LCTZ5454545	Education	0%		N/A

Project Description
IMB Application retirement project - CRTS stream

Q3 Report Due TODO - Due Date
Estimated Date of Project Completion 2022-08-27

Figure 1: Screen from our application prototype

Handover Documentation

Since this project was created with the intention of it being continued after Capstone, documentation to aid in knowledge transfer was a high priority for the sponsor. We documented our user research through the creation of personas and user journeys. Using that research, we were then able to draft schematics and design our application. We kept an architectural decision log to track our core technology choices and provided a feature list to describe what we've identified and what we've implemented in our prototype. After our development phase ended, we produced installation and deployment guides that detail how to install and deploy our prototype.

Our documentation can be found by visiting our GitHub repository:

https://github.com/bcgov/citz-dst-capstone-2021

Development Environment

The Reporting and Dashboard Service Improvement application has three tiers; the front-end web server, the back-end API server, and the database server. For our development environment, we ran local versions of our front-end web server and our back-end API server. We decided to use MongoDB Cloud Atlas to keep our test data consistent and reduce time spent troubleshooting local container deployments.

Front-end web server

- Runtime environment - NodeJS v`12.22.4`
- Programming Language - Typescript v4.3.2
- UI Library - React v17.0.2

Back-end API server

- Runtime environment - NodeJS v`12.22.4`
- Programming Language - Typescript v4.3.2
- Web application framework - express v4.17.1
- Object Document Mapping - mongoose v8.5.1

Database server

- MongoDB Cloud Atlas [1]

Sunghwan was responsible for setting up the initial development environment for the team and provided instructions on how to install that environment locally. He was also our full stack developer and designed our API and data models while assisting with state management of forms in the front-end. Samara focused on the application design and was primarily involved with the front-end development.

Production Environment

In our production environment, the three tiers of the application are deployed to OpenShift 4 (OCP4) clusters through the use of GitHub Actions [2] and pipeline-cli [3] whenever a new Pull Request is created. Refer to our Deployment Process [4] documentation for more details on deploying our application to OCP4.

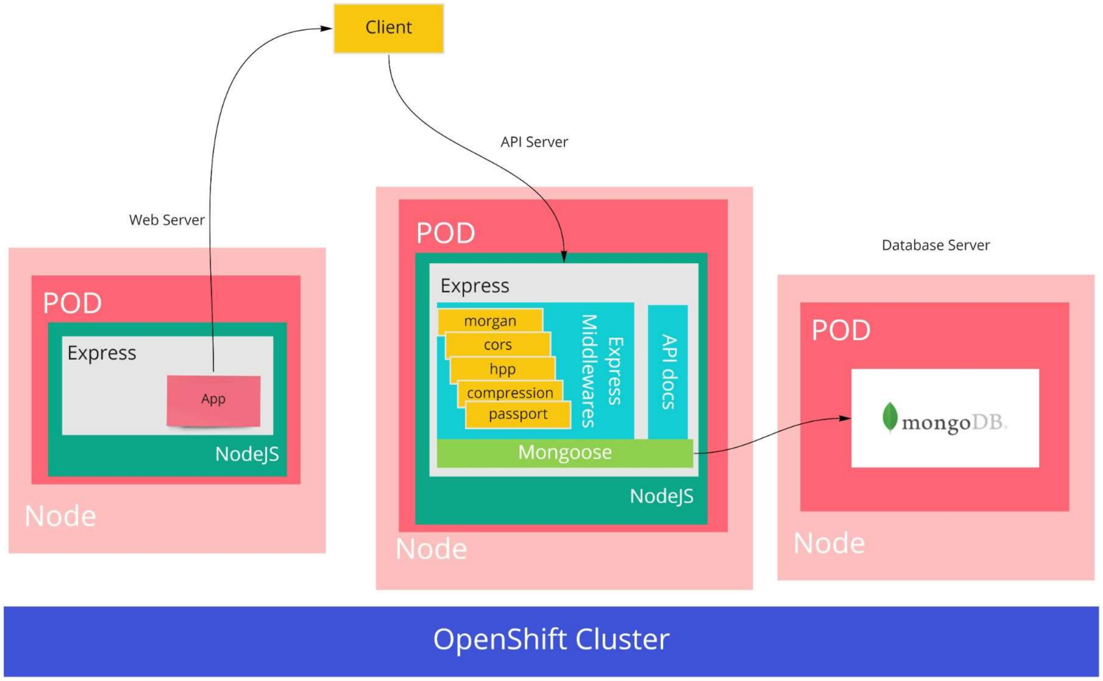


Figure 2: Solution architecture of our production environment

Infrastructure

BCDevExchange OpenShift 4 (OCP4) clusters

Deployed Containers

API / Web server - Red Hat Enterprise Linux Server 8.4, NodeJS 14.16.0

Database

Red Hat Enterprise Linux Server 7.9, MongoDB 3.6

Sunghwan

Sunghwan took on the roles of DevOps operator and back-end developer. Although he has plenty of experience in single page applications (SPAs) and back-end development, learning the React library and documenting APIs with swagger was challenging. Additionally, OpenShift was different from the traditional deployment methods he was used to and even AWS. With several training sessions, he was able to manage CI/CD processes using GitHub Actions and OpenShift CLI to deploy our prototype as we developed it. Another technical challenge was the NoSQL database, MongoDB, since his experience with non-relational databases was limited. However, it was a great learning experience to design a NoSQL database. He exercised analyzing of data consumption and query patterns and designing document structures. Using ODM instead of writing legacy queries improved productivity significantly. Overall, he got to experience what full-stack developers do.

Samara

Samara took on the roles of team lead, primary contact for the team, UX/UI designer, and front-end developer. This was a challenging project that also required strong business analysis and communication skills, especially with the COVID-19 restrictions in place. She was responsible for conducting the interviews with stakeholders and had to learn to quickly adapt to extract technical information from non-technical stakeholders in adverse interviewing conditions. Questions the team prepared would often become irrelevant or answered in other questions, so being able to formulate new questions on the fly to keep the stakeholder engaged was essential. She also learned how to document that user research through the use of personas and user journeys. She used this research to create wireframes that were used to design the front-end application. She learned that some of her designs were more complex to implement than she initially thought, so understanding the technologies used when creating designs is important to the success of the team and to avoid unexpected setbacks.

Tying It All Together

Cloud platform

At Camosun, we learned how AWS enables us to focus on business logic apart from management tasks such as security, deployment, network, and resource allocations. Although our infrastructure is the OpenShift platform, the concept of a containerized environment and CI/CD pipelines are similar to AWS's. Once we could connect similar themes, the detailed procedures of OpenShift caused little confusion with the team.

Modern web programming

Typescript is a typed superset version of javascript. Even though we haven't learned it in our Camosun classes, the team did not find it difficult to adopt. The concept of object-oriented programming we learned from our early Java courses combined with our experience using JavaScript were key to this success. Designing the MongoDB database would have been challenging if we hadn't learned the basics prior to this project. Using the ODM tool, mongoose, and its middleware functions made our tasks much simpler to complete.

Advanced React features

We learned the basics of React with a glimpse of hook functions. However, right when we jumped to implement the front-end, we encountered complex requirements that required us to use several types of hooks and even the Redux store. We can find these concepts in modern web frameworks like Angular and Vue. Therefore, if the ICS program provides an advanced web programming class, students may encounter less of a learning curve when implementing a web application in their Capstone projects.

Follow-Up Requirements or Recommendations

This project aimed to develop a prototype to demonstrate how a web application can be used to replace the current manual process with an automated one designed to reduce human errors. Therefore, we have identified many indispensable features that need to be done. Refer to the complete list of features [9] and recommendations [10] for more details.

Executive Dashboard

The dashboard should provide statistics to facilitate strategic decisions based on each project's performance and investment. Therefore, its components are primarily charts and graphs to visualize the costs and outputs.

Realtime Notifications

The application should allow users to view changes instantly. It is a fundamental feature that most users expect from modern online applications.

Query Functionality

In a prototype environment, this feature was not a concern. As the data set is getting bigger, searching and filtering functions would become the key points to maintain usability and performance.

User Management

The system administrator should be able to approve user registrations and update profiles. Managing user and project role mapping is an important part of data security.

Database Management

The data is the life-force of the application. Therefore, before launching it officially, every past and ongoing project and report should be imported. Also, the periodic and automatic data backup and exercising the restore plan is a crucial part of the application maintenance.

Single Sign-On

The user information is the core of data. Integration with IDIR authentication would be consistent with current BC Government practices and avoid the need to develop a new user registration system.

Change History

Auditors should be able to track changes in sensitive data.

Access Control

Role-Based Access Control should be implemented both on the front-end and back-end sides of our application.

Conclusion

Team RDSI was successful in delivering a solution to replace the DIO's manual reporting process with a web application within the time limits of the project. Our sponsor is pleased not only with the quality of work produced, but also that our documentation and the prototype we delivered were able to gain further funding for this project. This was a challenging and fulfilling endeavor and while we are sad to see it come to an end, we are excited to see where this project leads in the future. We thank for your time in reading this report.

References List

[1] “Managed mongodb hosting: Database-as-a-service,” *MongoDB.* [Online]. Available: https://www.mongodb.com/cloud/atlas. [Accessed: Aug 13, 2021].

[2] “Features: github actions,” *GitHub*. [Online]. Available: https://github.com/features/actions. [Accessed: Aug 13, 2021].

[3] C. Varjão, P. Simonian, S. Han, R. Stens, and A. Wenzowski, “Pipeline-cli,” *GitHub,* Dec 29, 2020. [Online]. Available: https://github.com/BCDevOps/pipeline-cli. [Accessed: Aug 13, 2021].

[4] Sunghwan Park, “Deployment Process”, *GitHub,* Aug 12, 2021. [Online]. Available: https://github.com/bcgov/citz-dst-capstone-2021/wiki/Deployment-Process. [Accessed: Aug 13, 2021].

[5] S. Park, S. Flueck, P. Sivanand, and R. Kobenter, “citz-dst-capstone-2021,” *GitHub*, Aug 13, 2021. [Online]. Available: https://github.com/bcgov/citz-dst-capstone-2021. [Accessed: Aug 13, 2021].

[6] “An Online Whiteboard & Visual Collaboration Platform for Teamwork: Miro,” *Miro.* [Online]. Available: https://miro.com/. [Accessed: Aug 13, 2021].

[7] “Project Management in GitHub,” *ZenHub*. [Online]. Available: https://www.zenhub.com/. [Accessed: Aug 13, 2021].

[8] S. Bathgate, J. Leach, Y. Freeman, I. Watts, B. Li, J. Pye, A. Jun, D. Hereñú, L. Gonis, and J. Kel, “Platform Services OCP Project Registry,” *GitHub*, Aug 12, 2021. [Online]. Available: https://github.com/bcgov/platform-services-registry. [Accessed: Aug 13, 2021].

[9] Sunghwan Park, “RDSI Features,” *GitHub,* Aug 11, 2021. [Online]. Available: https://github.com/bcgov/citz-dst-capstone-2021/wiki/Features. [Accessed: Aug 13, 2021].

[10] Sunghwan Park, “Limitations and Recommendations,” *GitHub,* Aug 11, 2021. [Online]. Available: https://github.com/bcgov/citz-dst-capstone-2021/wiki/Limitations-and-Recommendations. [Accessed: Aug 13, 2021].