Enviro DB

Legacy Document

|  |  |
| --- | --- |
| Team Members | Alec Wassill, Bryan Martinez, Cameron McNaught, Josh Lynn |
| Client | James Ravenhill |
| Instructors | Dave Watson, John O’Loughlin |
| Due Date | April 15, 2016 |

Contents

[Preface 2](#_Toc448147425)

[Lessons Learned - Alec 2](#_Toc448147426)

[Technical Concepts 2](#_Toc448147427)

[Managerial Concepts 2](#_Toc448147428)

[Recommendations for Future Projects 2](#_Toc448147429)

[Lessons Learned - Bryan 3](#_Toc448147430)

[Technical Concepts 3](#_Toc448147431)

[Managerial Concepts 3](#_Toc448147432)

[Recommendations for Future Projects 3](#_Toc448147433)

[Lessons Learned - Cameron 4](#_Toc448147434)

[Technical Concepts 4](#_Toc448147435)

[Managerial Concepts 4](#_Toc448147436)

[Recommendations for Future Projects 4](#_Toc448147437)

[Lessons Learned - Josh 6](#_Toc448147438)

[Technical Concepts 6](#_Toc448147439)

[Managerial Concepts 6](#_Toc448147440)

[Recommendations for Future Projects 6](#_Toc448147441)

[Future Enhancements 6](#_Toc448147442)

# Preface

In this document each member of the Recursive Delegation Capstone team will discuss the technical and managerial concepts learned for their Environmental Data Logging project and future projects. Finally there will be a collection of future enhancements that the team has thought of for the Environmental Data Logging application.

# Lessons Learned - Alec

## Technical Concepts

I learned a great deal of powerful technologies while working on this project such as Git, Spring Boot, Gradle, AngularJS and Angular Material. It took quite a period of time to get the work going at a speed I was satisfied with because my team and I needed to learn these new technologies. However, I am extremely grateful to learn these technologies as they are invaluable on a portfolio and used frequently in the web development industry, especially Git and AngularJS. Learning these technologies builds me a strong foundation to continue to grow as a web developer and programmer.

Not everything was went smoothly however. AngularJS and Angular Material are newer technologies, they are still in development; with features being added and bugs being fixed. We did have to come up with workarounds or research alternatives to accomplish tasks.

It was hard to plan and work on a project where the technologies and practices used evolve constantly. However once we wrapped our heads around the new technologies, we were able to crank out fairly sturdy application.

## Managerial Concepts

Working with a hardworking, opinionated team was a blessing. Some decisions did take additional time to be made, but in the end, I believe we made the right choices to the best of our knowledge. Working in a team really made me consider other methods of completing tasks, programming, organizational and logical. As for struggles, completing tasks could have gone quicker due to having to consult the rest of the team on their opinions. Not having an assigned manager does slow things down, but the end result is generally what all members agree is optimal after debating thoroughly. I do believe having weekly reports was a fantastic way of pacing our work and ensuring that our tasks got completed. They also gave us a great perspective on what we have accomplished, and what still needed to be addressed.

## Recommendations for Future Projects

I would recommend learning and playing with the technologies you plan on using for the project. Most of January and part of February was spent understanding and becoming familiar with our AngularJS framework, with the bulk of the project being cranked out in March once we wrapped our heads around it.

I would have liked to meet with the client more often to further understand the workflow and their needs for the application, as well as user interface preferences.

# Lessons Learned - Bryan

## Technical Concepts

The time difference between doing something the first time versus doing something the second time is exponential. At the start of this project, things like Git, Gradle and JPA all took a long time to learn the first time around. A lot of research went into learning how each tool operated and the proper ways that each tool can be used. But the second time you do something, it takes 10 seconds to create something that it took 5 minutes before.

Another thing I have learned is that to be careful when learning newer frameworks and technologies. When dealing with the unknown, there is very little resources available and that you must take everything with a grain of salt because the solution to a question may be outdated because of the rapidly changing frameworks. A lot of the time, we must create our own ways to tackle unique problems to situations that maybe no one has faced given really certain circumstances.

It is hard to plan out a project when technologies change so fast. Working on a two month old project, practices and methods that were considered best practice could already be outdated. At some point in time, as a team we must stop and determine how far we go with newer technologies and draw a line to say that our product will stop at this version. A good example of this is the difference between two Angular versions. It is the same framework, but the difference between one version and another can change an entire project.

## Managerial Concepts

I have learned that working with a group of people can be a double edged sword. There are so many things to gain from working in a team and so much more progress can be made by working with different people of different specialties. On the other side of the coin, working with a group of people can very inefficient. A lot of the time you spend more time talking about things rather than actually coding/working and often times it is important to have someone to keep you on track. With a project of this scope, I have learned that it is simply not possible to know the exact progress of each individuals work, having a schedule and specific deadlines really help towards organization. I have also learned that having weekly progress updates can be really useful if used right. It helps the team get a clearer picture of what each person did and what they are working towards.

## Recommendations for Future Projects

My recommendation for future projects is to start learning the frameworks you may use beforehand. The earlier you start learning a certain framework, the faster you can code at a later date. Another recommendation I would make is to spend more time testing the functionality of the program and to meet with the client more often to see if the product you are creating meets their workflow.

# Lessons Learned - Cameron

## Technical Concepts

Coming into this project I had a limited view on what I wanted the application to look like. But in order to get the application looking and functioning in the manner I wanted I needed to learn some new technologies. These include: AngularJS and LESS.

AngularJS was perhaps the biggest tool to learn, as the syntax is different between a Javascript framework and Java (which I was more accustomed to). As well learning the lifecycle of AngularJS components (factory, service, module, etc.) and how to properly use dependency injection is my coding took a while to wrap my head around.

LESS is a CSS preprocessing language that allows me to break up my LESS code into separate files, but still have only one master CSS file to include in the app. The syntax and change in the thought process behind writing CSS were much different than what I had known prior to this project and has only increased since day one.

## Managerial Concepts

The biggest managerial concept I learned through this project was that each member of team has a different skillset and, to produce the best project, each member should work and focus on what they excel at. In the case of our project: I focused on the client-side app and documentation, Alec focused on the communication between our 2 apps, Josh focused on the more complex concepts used in our server-side app and Bryan concentrated on the workflow of the app and the workflow of the data.

As well I saw the need for a project leader (which we didn’t have too much) as there were many instances throughout the past 8 months that we would fight/debate on concepts, practices and the direction of the app; having a project manager/leader that would make the deciding decision would have helped a lot. One aspect of our project that was a good lesson, but tough in this situation was our “lack” of a client. While we officially had a client, having very little input from him in regards to what the application should accomplish or how the workflow should take place was a burden on our project. I feel that if our client invested in our project, more than what he did, we wouldn’t have had as frustrating of a time working on it.

## Recommendations for Future Projects

Some things I would like to recommend to future students working on their Capstone projects include: picking the right team, divide the work accordingly, learn your frameworks prior to implementation and bring a new technology to your project.

Picking the right team is crucial to the success of your project. When I was deciding who I wanted in my group I had 2 main characteristic traits I was looking for in potential team members: willingness to learn and personality. Was the person I was considering someone who wanted and would learn a new tool or mindset to building an application? Was this a person I could handle working with for 8 months and not want to punch them in the face at the end of the project? Remember the group you go with will be your team members and classmates for 8 months. While you might want to be in a group with your friends, I encourage you not to make that a first priority in picking a team. I’ve seen too many strong programmers decide to be in the same team as their friends and now at the end of the project are greatly regretting their decision.

Divide the work accordingly amongst your team members. I had mentioned this earlier, but having each member work on what they excel at rather that everyone try to do everything is essential. Though I already mentioned how my group divided work in the implementation of our project, I’ll provide an example here how we divided work on our document. Because I enjoy technical writing and am good at formatting documents and making them look professional I was tasked to managing our documentation. This means I did not write and of the content, that was not my job. The other members of my team wrote the content and drew the diagrams then as they finished their section I would get it and incorporate their work into the master document. Through this we all learned the material covered, learned how best to program our application and overall it was less stressful on us.

Bring a new technology to your project. Take the concepts you learn in school, but don’t let what you learn dictate what you have to use in you project. In the case our project we used tools like AngularJS, Swagger, Spring Boot, Redis, Code-First and Bower. All these tools might be mentioned in school, but none are taught. Each tool allows you to enhance the functionality and ease at which you can produce a working application. That being said if you choose to implement any new tool spend at least 2-3 months learning it prior. You can learn it as you develop, but at the end once you know the tool better you won’t like some of the workarounds you came up with earlier in the project. And just to summarize this, if you do learn a new technology, out of what I listed, then learn Java Spring and a JavaScript framework. These 2 alone will set your project apart from others and allow you to really challenge yourself.

# Lessons Learned - Josh

## Technical Concepts

We jumped right into the deep end without knowing how to swim and somehow survived. We almost drowned multiple times trying to learn what we were taught in class but in a different framework. In the long run, it made us much stronger and a much better ‘swimmer’.

We did make some mistakes based on not knowing what we were doing. Making a hasty change after a 5-minute conversation created a few sleepless nights for a few of us, then realizing we made the wrong decision and needed to go back to the way it was before.

## Managerial Concepts

It is hard to manage a team when there is no clear leader. We deliberated on all issues and heard all voices, but because no one was able to make the final decision based on everyone’s views, things went slower than ideal. We could have accomplished much more with a clear leader.

The waterfall method is extremely time consuming compared to an agile environment. Halfway through we switched to more of a hybrid waterfall/agile and things worked much better. Our team was strong enough to make the proper decisions on what was the next piece that was most important to work on.

## Recommendations for Future Projects

I would recommend having a ‘mock’ project on the side to learn a new feature, like proof of concept. Our app grew in size and it made testing out a new theory very difficult without breaking everything

# Future Enhancements

* A notification/message feed system to notify admin users that new data has been imported to the system and requires attention (being assigned to a project)
* Have a few different styles of report templates available
* Password recovery
* Cleaning up the methods of importing data would improve performance
* Upgrading the JPA to allow more advanced database queries as well as improve performance
* Incorporating a calendar tool to track employee days off mixed with project timelines
* Multiple themes
* Implement multiple institutions
* Break the server into micro services with an API gateway accessing them all
* Implement OAuth
* Implement a real-time messaging system
* Implement more reports
* Create a dashboard that the user can customize to what they want to see when they log on
* Add local data to the browser. So if the user accidently leaves a page before saving it, their data will not be lost