**Final Report**

11 December 2019

American University

**TABLE OF CONTENTS**

**Client 3**

**Team Members 3**

**Review of Milestones: 4**

**Plan vs. Reality 4**

**Major Changes 6**

**Reflections on project - Ambitious or not? 8**

# Client

Our client for this project is Kristof Aldenderfer, Adjunct Professor for Physics and Director of the Design and Build Lab (DaBL) at American University. He can be contacted at kristof@american.edu or 202-885-6431.

# Team Members

|  |  |  |
| --- | --- | --- |
| **NAME** | **ROLE** | **EMAIL** |
| Myles Russell | Team member | mr7085a@student.american.edu |
| Abdul Tarawally | Team member | at4667a@student.american.edu |
| Isabella Sims | Team member | is3923a@student.american.edu |
| Jiachen Yao | Team member | jy3695a@student.american.edu |
| Li Jie Su | Team member | ls8613@student.american.edu |

# Review of Milestones

# *What did you plan, and what did you ultimately build?*

The requirement of our client was to create a queueing and scheduling system for the DaBL. Based on client requirements, our team initially planned to create a separate website with following functions: login, signup, and machines reservation. After meeting with the client, we assessed that DaBL user data contains personal student data which is not compliant with AU policy since the intention is to store it outside of AU networks. Therefore, in contrast to the original plan, we decided to append a reservation page to the existing DaBL website.

Additionally, we decided to replace the login function by requesting user email addresses for input into the DaBL system. Following all these changes, the final version of the product became a web application that implemented scheduling functions using the *Wordpress*, *php*, and *Google Calendar APIs*. Furthermore, in order to make our client works easier, our team also delivered an electron application for DaBL administrators to add new users and modify existing users.

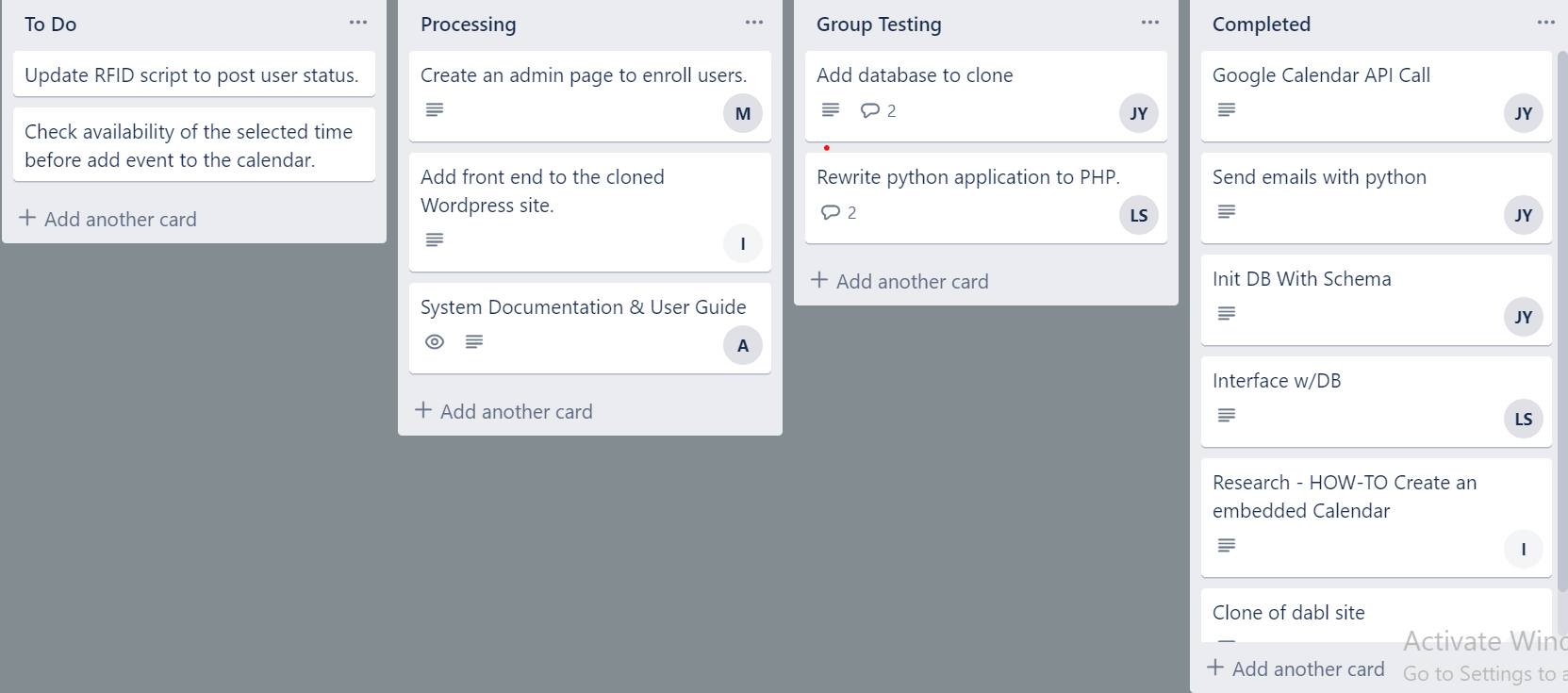
***How did you track against the scheduled milestones, and how did these change over time?***

The team started with the *Tetra* web application to track all assigned tasks to team members. After the first sprint, the team determined that *Tetra* was not user-friendly and made the decision to shift planning to the *Trello* application in order to use the Kanban board-style utilities.

*Trello* has been a useful tool throughout this project. The tool was employed to delegate and define tasks under the following four categories:

* **Todo**: Tasks that have been created but not assigned or started
* **Processing**: Tasks that have been assigned, started, but not completed
* **Group Testing**: Tasks that have been assigned into tests on groups of items, rather than individual ones.
* **Completed**: Tasks that have been assigned, completed, and checked by every team member. Tasks are moved here once the group has reached a consensus that the code or idea is ready to be merged into our current working version

An example of the format of each ticket and category are pictured below:



*Trello* also enables version control and provided structure to team weekly standup meetings. Project changes occurred more with respect to refining requirements than experiencing major shifts in what the group intended to develop. More specifically, the project requirements that did evolve, did so in such a way as to make the project more manageable and sustainable overall. Moreover, project requirements were mutually designed through several conversations with the customer resulting in client input and concession throughout the process.

***Reflect on major changes against your plans - what conclusions did your group reach about missed milestones, or any intended features that you were unable to deliver?***

Central challenges that spawned major changes in our proposal typically centered on networking, data, and infrastructure problem sets. The group encountered several issues from the outset that challenged our understanding of the DaBL infrastructure. Many of these issues were debated heavily in the group’s initial meetings aimed at deciding on a project. Inevitably, some of these issues were realized throughout the project. The group was able to architect a system that, though complicated, remained true to its original form. Notably however, the requirements increased during the project instead of changing direction.

The first challenge occurred with the interaction between the DaBL local database and the AWS instance where the web application is hosted - a problem that still exists at the end of the project. This portion was experienced towards the end of the project during system testing as we attempted to deploy the system into the client’s environment. The AWS instance is unable to open a specified port for access, thus inhibiting the ability to move onto the next phase of testing and implementation.

At this juncture, the group simply had to accept that the administrative application would be employed locally in order to meet project timelines. Reflecting on this issue, it indicated that testing and implementation would have identified the issue sooner and possibly allowed it to be resolved by the time the project was due.

The second challenge to progression was *Wordpress* front end and backend communication. The team spent a great deal of time integrating the front end and backend into the AWS system. This was likely due to inexperience with the *Wordpress* application, but pushed group efforts until the last minute with debugging issues. In this case, overconfidence of an implementation locally did not translate quickly or easily into the target environment. To remedy this, the group worked closely together to identify and debug issues, as well as partnered with the client to educate themselves on the infrastructure and areas that may be causing the problem.

This leads into the final challenge: DaBL database content did not easily translate into how the application envisioned using the data on the website. More specifically, once the group begin integrated testing with the intention of deploying, it became apparent that the database did not have email addresses listed for each user, limiting the ability to query users even if the sanitized AWS database was readily created from the DaBL local database. It was at this point that I think the group realized the web application could only be a *proof-of-concept* as more user data would have to be collected, let alone a plan in place to sanitize and push data from the local database to the AWS instance.

Conclusively, the biggest concession the group encountered was that the project would not likely be deployable by the project deadline. The team was able to deliver a proof-a-concept to the client despite several of these issues that will ideally pave the way to actual implementation at a later date.

***Were any of them too ambitious, or not ambitious enough?***

In terms of planning, our team members underestimated the learning overhead of programming in *php* (specifically integrating it into *Wordpress*) as well as interacting with backend infrastructure.

In order to run *php* backend scripts, we were required to obtain access to the server from our client. A workaround the team utilized was to use a *Wordpress* plugin. In order accomplish this, the team located the *Wordpress* server folder, location, and its content. This proved difficult because after uploading files, security issues with internal server error 500 (which is not defined in order to prevent hacking) prohibited us from going further. This was ultimately unsuccessful.

In the end, we utilized HTTP requests to run our *php* scripts. This is not to say that programming in *php* was too ambitious, but rather that our assumption that system integration would be an easy process was unfounded. The team ultimately learned that the configuration aspect of this project was much more difficult than the code itself.

Our estimation of time to complete this project was overly ambitious as well. Because we did not foresee most of the issues we encountered, we spent too much time developing individually when we should have been focusing on integration. Once the team realized the difficulty was greater than predicted, we immediately began to meet daily in order to solve the problem. It is fortunate we discovered the issues early enough to produce a *proof-of-concept* but unfortunate that we could not realistically achieve what we set out to do given AU Office of Information Technology restrictions, as well as server side limitations.