IMPLEMENTATION

Strategy/Approach

Initially, the team intended on implementing the development of the application into a single phase with coding, testing, and deployment occurring subsequently. The implementation plan was changed early on to a more Agile approach using phased team sprints. Development was divided by functionality into their own phases. Within a particular functionality phase, the tasks were further subdivided into specific assignments. Testing also occurred simultaneously and direct feedback from this type of testing helped to fine tune the application along the way. The team eventually settled on a modified version of the phased team sprints, however. The team itself was subdivided into development groups who would work on each phase together with the advantage of multiple developers checking work as it was done.

The planned phases were per the following list:

1. Login & User Profiles
2. Admin Features (Create and register users)
3. Admin Features (Add supplies and drugs)
4. Physician Features (Plan CRUD)
5. Pharmacist and Technician Interactions
6. Advanced Interactions (System auto archive and physician clawback plan)
7. Day of Surgery (DOS) View
8. Request System (Transfer plans and plan suggestions)

Overall, the change in approach helped to reduce development time and progress could be more easily tracked with the assignments through Basecamp. Testing throughout rather than after development also helped to ensure the application code remained reliable with the addition of new functionality and features as development progressed.

Challenges

A significant challenge the team encountered was related to working with new coding languages and frameworks. The team has had some instruction on coding and web development during the course of the MBT program, and some members have even more experience than that level. However, React Native was a tool unfamiliar to all of the members. There was some base familiarity due to the JavaScript involved, but there were many more differences when applied to React Native.

Usage of mobile backend as a service (MBaaS) was another challenge. Aside from being unfamiliar with MBaaS as a concept, there was the additional learning curve of the selected service. Kinvey

was the service chosen by the team initially, but its lack of support for React Native was a significant issue. Kumulos proved to be a decently powerful alternative with React Native support and it provided the necessary backend features for the application.

Minimizing merge conflicts when combining together coded work was another challenge encountered by the group. With multiple developers and branches of work off the same code base, it was difficult at times to control merge conflicts. In order to address this issue, one team member was designated to handle all of the merges with another team member serving as an assistant. This approach proved to be directionally more successful.

The last major challenge was related to scheduling meetings with the sponsors. It was a challenge in itself to find times which worked for both the core team and sponsors due to very busy existing schedules. While several group meetings were successfully scheduled, other solutions to communicate and discuss ideas and problems were employed. Smaller group meetings versus full group meetings were favored to help ease scheduling constraints. Videos as a tool to demonstrate new functionality was also used, which had the advantage of demonstrating on a “watch when you can” basis.

Solutions

Due to the controls, communication, and change management plans in place as set forth by the project charter, the team was able to address issues as they arose quickly, both within the team and with the sponsors as required. One prominent example of this was confirming priorities and minimum viable product during development due to time constraints. The team and sponsors were able to prioritize work such that the projected end product met the expectations of the sponsors.

Earlier on during development, the change from organization of the project from waterfall to phased sprints was also a significant solution to help structure the development phase and increase productivity. It helped to focus the team and made the goals specific enough to better show progress during the semester. A related solution was designating specific team members to merge code together to the master branch in GitHub to minimize merge conflicts.

Overall, one key behavioral aspect which bolstered all other solutions was the team’s strong ability to communicate, both with regards to progress and issues early on so they could be fixed and addressed in a timely manner. This was facilitated with the scheduled weekly meetings and through messaging in Basecamp.

Lessons Learned

Communication was one of the biggest learnings as a result of this project. Initially, the team discussed with the sponsors ideas for functionality, which involved the Mockingbot prototype as well as sponsor sketches. However, there seemed to be a difference in opinion in what exactly a prototype was and what it included. Having a better understanding of the sponsors’ immediate

goals and definitions, including asking clarifying questions and making better use of additional wireframes and use cases, could have better shaped our work and approach to the solution.

Considering the development performed and the framework used, a better and more realistic estimate of expected time per phase and task could have been achieved. The team felt it underestimated the learning curve of the technology used in this project. Becoming more familiar with these technologies and allowing for more time to familiarize with the tools would have helped to make development smoother and easier.

One other lesson learned involves minimizing the scope of the project. It may have been more beneficial to better define the specifics of the deliverables. In addition, a better identification of what the minimum viable product entailed would have helped moderate the scope of the project. Finally, starting on development even earlier than allotted for this project would have helped the team deliver a more complete form of the application.

Required Follow-Up Actions

There are some features which did not fall under minimum viable product (MVP) but are still incomplete as of the final presented version. These items include:

- Edit plan functionality is incomplete
- Notifications and messages tab does not have error handling in place in the event a plan is removed or discarded
- Users cannot view themselves in the user listing
- Users cannot deactivate or revoke administrative privileges for themselves

The project sponsors have been made aware of these defects and outstanding features during the project handover.