Levi Amen

Quentin Covert

Mark Hernandez

Cameron Johnson

Collin Victor



CSCE 361

Dr. Cohen

**Team Dynamics**



## Final Team Roles:

Early on we attempted to define team member roles based purely on our discussions of past experiences, however as time went on several of those positions changed significantly. Most notably, Collin and Mark gained more defined specialities, and Quentin took over Git repo management from Cameron.

Cameron: Team Lead and Head Developer

Collin: Database Guru/Developer

Levi: GUI Guru/Developer

Mark: Watchman Guru/Developer

Quentin: GitHub/Slack Manager/Developer

## Communication Changes:

We originally planned to use Blackboard for communication. However, we found that Blackboard communication methods didn’t fit our needs. We instead used a free communication software called Slack for scheduling meetings and discussing our concerns throughout the project. Slack allowed us to create multiple channels: #general, #github, #meeting, and #random. Minor questions were discussed over in the #general channel; it was used to conduct remote meetings as well. The #meeting channel was used to schedule remote or physical meetings. All push notifications to our repository were directed to the #github channel. #random was exactly what it sounds like: pure water-cooler nonsense. When we were writing non-code documents, like this document, we used Google Drive to actually write the document and Slack to discuss what was being written.

## Team Scheduling:

While we originally planned to meet for at least 4 hours a week throughout the semester and to conduct regular code reviews, as mentioned in our original team dynamics document. Unfortunately, as the semester quickly got under way many of the teammates began to get bogged down in other activities making meeting these goals impossible. This was especially true since three team members were busy 10 hours each weekday due to work, school and Design studio. Meetings began to be done ad hoc and scheduled on the fly through Slack. Code reviews were never scheduled over the course of the semester.

## Reflection:

In retrospect, there are things our team could have done better to see a more successful product in the end. It was quite unfortunate a majority of the team members faced busy schedules, but looking back there are things which could have been handled better to aid in getting the project done.

Firstly, the team also didn’t practice good communication between team members over Slack to help remedy the disjointness in the team. Communication is key to a successful team and while team members did communicate as a whole there needed to more communication than is normally called for due to the dynamics of the team. Meetings needed to be more regular to help keep everyone up to date on progress in the project.

Next, and most importantly, the importance of good and complete specification documents became painfully obvious in the end. As a team, different modules were recognized and dealt out to various members. In the team’s specifications sheet, particularly in the class diagrams, there was no connection between these modules, meaning it wasn’t clear how one module used another or how they were related. This made integration a nightmare which took many hours of frustration and development to get basic functionality working. This was compounded by incomplete sequence diagrams resulting in the team having to work on program flow while integrating the system. Also, two key modules (the remote and sync modules) were very ill defined in the specification sheet. This made integration exponentially more difficult as much of these modules functionality had yet to be fully realized or implemented till that point.

In conclusion, the SyncOrSwim team was far from perfect during this project. However, this was not directly due to any failure on any team member's part. This project was a learning experience for every member in the team, and ultimately many lessons were learned the hard way. Now each of the members are walking away with a much better idea of what to do next time they work on a Software Engineering project. Here’s a brief list the team compiled:

1. Verify that the connections between separate modules of the project are well understood and the relations between each one is well investigated. This will greatly ease system integration and testing and avoid painful headaches.
2. Schedule regular, set, team meetings. Team meeting may be boring, but they serve as a key mechanism for feedback between the full team. Members might catch important details about the project during the meeting they didn’t realize before that they didn’t fully understand. Also, meeting are a fantastic way of keeping other members accountable for keeping up with their work in the project.
3. DO THE PAPERWORK! Nothing is more dull than writing up requirement and specification documents, but they are the most important part of the project. It will save the team an infinite amount of frustration if these key documents are written to completion with full exploration and understanding of the project. Also, make sure each team member has reviewed the specification and requirement documents so that everyone is on the same page with what is to be accomplished when the coding starts.

Unsurprisingly, all of these realizations were taught during the Software Engineering class. However, for most people it’s one thing to learn something and a whole different beast to see it in practice. Now the SyncOrSwim team members have experienced what it means to be in a Software Engineering project, each member has come to fully appreciate the principles of Software Engineering in practice.