Project Experience Report

(After Action)

Freight Shield

ENSE 477

Team Name: Freight Shield

Project Reviewed: SSE Capstone

Date of Review: Started on March 24th, Finished on April 5th

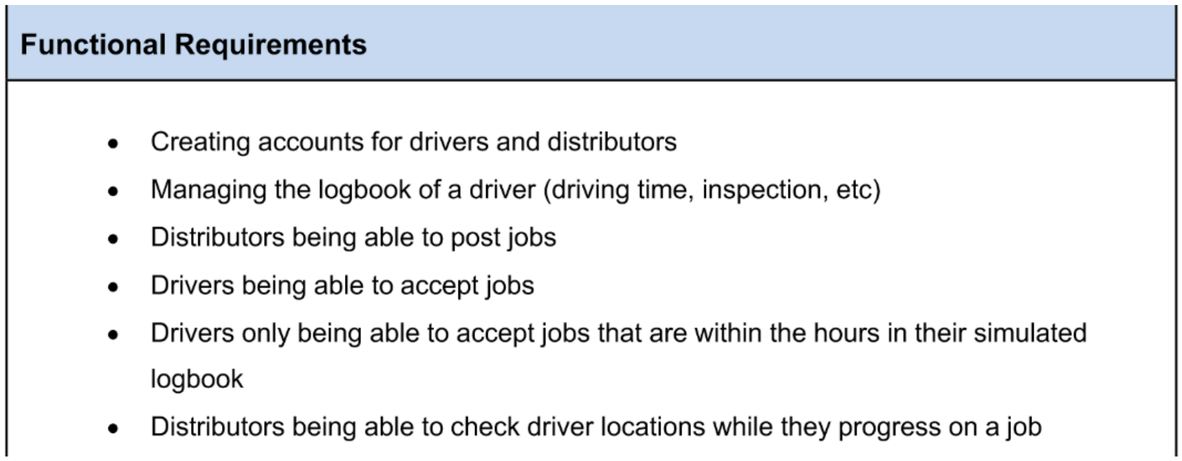
Group Participants:

Name:	Focus:
Mohammed Al-Harbi	Mobile Application
Ramanpreet Singh	Web App Development (frontend)
Amandip Padda	Web Application (backend) & Bug fixes
Alok Paranjape	Documentation & Planning

What was expected to happen?

For our expectations for the project, the purpose came from the group members working in the trucking industry. There are competing factors with safety and efficiency in industry. On one hand, that, like any larger economic system, there’s a push towards efficiency, serving as many people as possible. On the other hand, there are regulations to protect workers, (which here means working too many hours and driving while exhausted), and keeping detailed records to ensure compliance. The idea for Freight Shield came from sharing that information with software, and hopefully, helping people to make better, more informed decisions.

As part of the documentation for project initialization, we needed to make a requirements document earlier in October. This included some direct, functional requirements about our expectations for the project.



Functional Requirements

- Creating accounts for drivers and distributors
- Managing the logbook of a driver (driving time, inspection, etc)
- Distributors being able to post jobs
- Drivers being able to accept jobs
- Drivers only being able to accept jobs that are within the hours in their simulated logbook
- Distributors being able to check driver locations while they progress on a job

The requirements document breaks down the project into several, basic expectations:

- First, in order to share information, people need to be able to create authenticated accounts with our software, and, due to the distinct user groups and features for each, they need to be able to do so separately and go to the appropriate section of the software.
- Second, since most of the important information (pickup/dropoff locations, drivable hours, locations, etc) is kept in the existing logbook format, that drivers are already legally obligated to use, the next requirement is about the software emulating those documents.
- Third, in order for the information to be shared, drivers and distributors need to be able to interact, in this case, through job postings, which were planned out in

the third and fourth requirements, for each user group, respectively.

- Finally, the shared information has to be used to influence decision making. This is covered by the last two requirements.

Essentially, the requirements are structured as, (1)“who’s going to use the software”, (2)“what information is being recorded and shared”, (3&4) “how the user groups will interact and share it”, and finally (5&6) “the improvements that happen due to the software”.

From the initial requirements, we had a specific vision for the target audience in the trucking industry. Due to the safety requirements and information coming from regulations on individual drivers, and the decision making coming from the companies that manage them (renamed to carriers rather than distributors, to prevent confusion), it was clear that those groups would be our primary audience from the project. After this, we also came to the conclusion that the shipping assignments would have to come from somewhere, so we expanded the scope to include shippers as a user group. As well, when considering the sensitivity of the information being handled, we also realized the site would some kind of non-technical administration to manage accounts and verify new users. Overall, the audience started out very specific, but after considering the breadth of interactions surrounding the initial idea, we opted to widen the scope.

The initial timeline for the project was very underdeveloped. Based on the project charter, we didn’t have much in terms of a schedule, with the most concrete goal as being the one MVP with user features by the end of the fall term, and the rest being course deadlines. There’s another section later on what we could have done differently, and, reviewing the lack of milestones, they certainly qualify for it.

In terms of the people involved, there wasn’t much out of the ordinary. We predicted the people involved everyone on the team, our supervisor, and the customers at Light Speed and 1005042 Alberta Ltd that we were able to contact during project initialization.

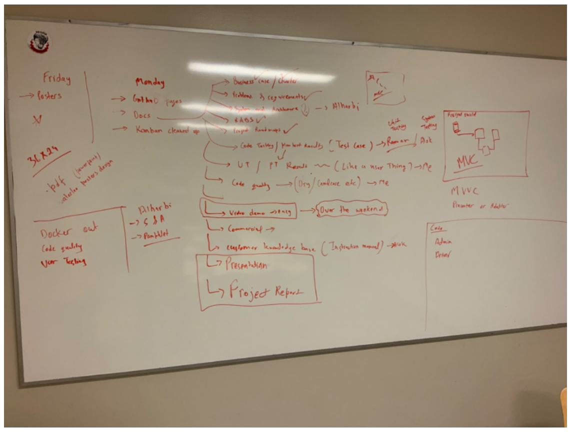
Overall Project Milestones	**Dates**
Project Proposal	25 September 2023
SCRUM 1	16 October 2023
SSE Program Chair Chat	30 October 2023
SCRUM 2	6 November 2023

There were also many envisioned barriers. This included scope creep, getting stuck on the initial objectives, the learning curve with the new technologies involved, and not being able to meet the needs of users compared to alternatives.

What actually occurred?

We used a divide and conquer approach in our project development. As part of project initialization, we separated the project into various user stories,features and documents, and then proceed to independently develop the features week to week. Team members worked on different sections of the project, especially towards latter half of development. Each person picked a section of the project to focus on. In particular, some group members weren’t as experienced with using MERN, which meant that time was better spent on other sections of the project, but there were also some disadvantages, as brought up in the later sections.

Over the term, we had a few different meetings with our advisor, Prof. Sharma. We mostly had questions about deploying our application on a cloud server. While he hadn't specifically supervised a team with a project similar to ours, he still had some general information about difficulties we might encounter during capstone.

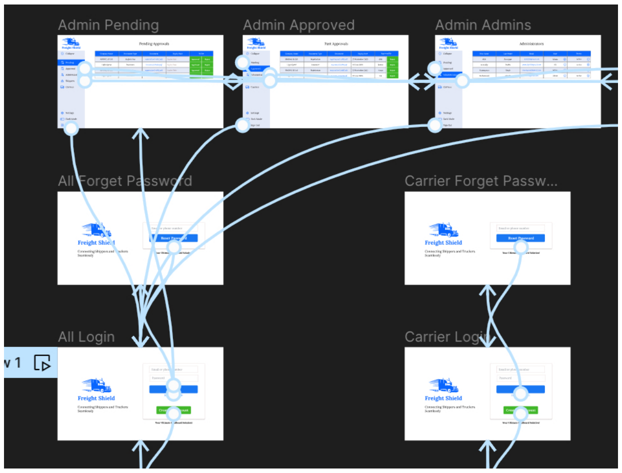


We through user testing with several employees from 1005042 Alberta Ltd and Light Speed. During project initialization, we made a survey about features that would be important. Later on, we did a few different sessions of user testing. The user testing was partially guided, and partially unstructured. We made use of Windows quick assist to give them access to a computer with an in-development version of the application and went through some basic tasks in order to gather feedback.

As a consequence of the User Story Board, a lot of the development process was objective based. In between the scrums, we'd pick out a different part of the project, usually an item already on the Kanban board, and focus in on developing it within a week or so.

At the start, there were decisions about the language and tech stack. We opted to go with MERN, partially due to previous experiences of some of the team members, but also because products similar to our envisioned solution (such as Uber) also use the MERN stack.

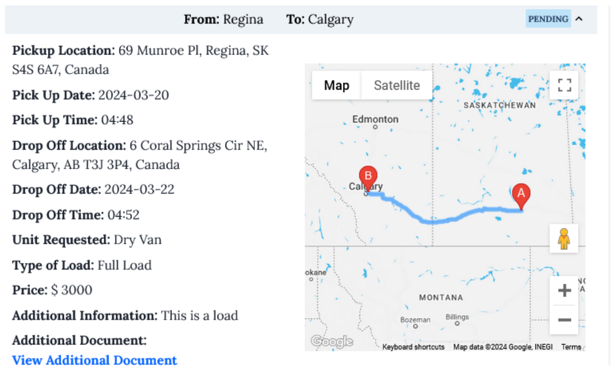
The next important part of the early development process was the prototyping, which came down to a number of low and high fidelity prototypes. It started with whiteboard drawings and sketches, but evolved into more functional mock-ups with specific pages and routing done with Figma.



As well, a significant amount of effort went into installing and setting up a connection to a MongoDB database. Because a lot of the project involved sharing information between different users, getting a good idea of how those users and that information would be stored was important to clarify. It was also important that the data based system we were using was open source, and that there was a lot of examples and documentation to learn from. While there were some ideas about using an SQL style database, a NoSQL one was easier to work with.

The mobile application was created using Expo, which was mostly caused difficulties throughout development, from the lack of features and documentation.

Another part of development was using a mapping system, which was needed for user tracking and route calculation, key features of our project. We opted to use several of Google's APIs and we found they were extremely well documented and supported. One of the features expected to be difficult to implement was the route calculation, but it was significantly easier than expected due to our choice of libraries.



From: Regina **To:** Calgary PENDING

Pickup Location: 69 Munroe Pl, Regina, SK S4S 6A7, Canada

Pick Up Date: 2024-03-20

Pick Up Time: 04:48

Drop Off Location: 6 Coral Springs Cir NE, Calgary, AB T3J 3P4, Canada

Drop Off Date: 2024-03-22

Drop Off Time: 04:52

Unit Requested: Dry Van

Type of Load: Full Load

Price: $ 3000

Additional Information: This is a load

Additional Document:
View Additional Document

Due to the sensitivity of the data being shared, it was important to be able to authenticate users properly, especially with regards to people who might manage the site and onboard others. We spent a significant amount of time working with JSON Web tokens for authenication. One advantage from this decision was that it made routing different user types more streamlined. There didn't need to be separate login pages, and they routed properly to each user's homepage, which an earlier complaint in some of the scrums and user testing.

One of the later pieces of development was remaking the notification system between users. The latest version works in real time between applications and shares data in a similar format to the rest of the site, without the status and locations. This wasn't the case before, and pages needed to be reloaded between users, and were just plain just. It took a lot of effort to get pages to properly update and track notifications live, but ultimately it was successful.

What went well and why?

We learned a lot about using MERN, and EXPO, in spite of the number of challenges we faced during development.

An interesting group dynamic was that some of the members the team preferred to work late into the night, as opposed to between daytime classes. This meant that there was a handoff between different team members, and resulted in more continuous development throughout the day. For example, when trying to develop a new feature, team members might run into new issues, create a set of questions, and then share those questions with others in order to talk to instructors and other students during the day.

While it’s not exactly a design decision, it was good that people with experience in the trucking industry were on the team and took the lead. Because the project was based on the specific conditions of industry, having that context when designing, developing and generally working on the project was important. We also wouldn’t have been able to connect with those customers without those industry connections. A large part of the development that was successful towards achieving our goals came from our customer feedback, as well as the direction from team experience.

What can be improved and how?

Even without hindsight, we could have improved our scheduling and goals. The deadlines we set were vague, and didn’t help as much with direction, especially with the divide and conquer method. While there were objectives, they weren’t properly established and written out, especially with the earlier reports. All of this lead to a lack of focus and meant that time was used waiting for connected features to work independently. If we were doing something like this again, we would definitely include more rigorous scheduling and duties.

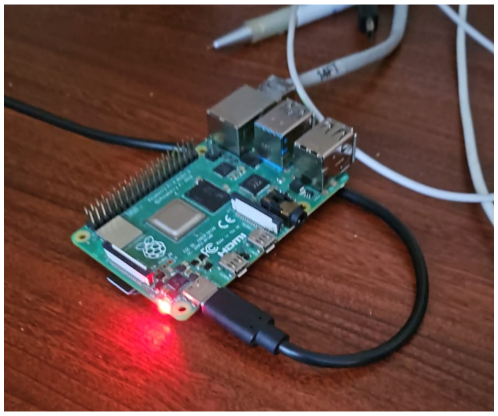
Looking back on the project, there are a lot of different decisions that, if made differently, probably would have saved time and probably allowed us to go further with development.

While there were some benefits to the divide-and-conquer approach, there were also disadvantages. In particular for this project, a lot of our requirements relied on interaction between the mobile application and the web client. For instance, Drivers being able to fill logbooks and then send them to the other user types. Because the Drivers were a separate (mobile) application, a lot of the work on the web client was done in isolation, and needed to be reworked once the mobile application was developed.

Taking this further, we could have set up the code for each user type into separate code bases right from the start of development. While the final product does include an ostensibly separate code for drivers, as a mobile application, the other three user types and all part of the same code base. As part of the waiting for separate features, a significant time sink was trying to get driver functional enough to work with each of the other user types. At the same time, changing some of the code for the connected code, such as carriers, meant needing to change other parts of the code for shippers and administrators. With the way the code was connected, we had the disadvantages of both connected and separated code bases, but the advantages of neither approach.

As well, there was another possible way we could have focused on the main communication between drivers and the other user types. There are other potential demographics in the industry that could have been designed for that are more communication focused. For instance, owner operators are a large part of industry, and within our current system, would require accounts in order to handle communication with prospective carrier companies, and to manage their own driving. On a similar level, brokers are a demographic in industry who specifically focus on negotiating agreements between the various other user demographics. By making features for more communication-centric roles in industry, we might have been able to get to the main aspects of our problem more quickly.

Another possible mistake was the initial attempts with using EC2 on Amazon Web services. Uploading the first set of files and running basic tests somehow ran out the free trial, and caused a small financial cost. We eventually settled on loading the website on a Raspberry pi, and only paying for the domain name, which was a difficult process unto itself. While we weren't able to test how easy it would have been to use Amazon Web Services, it probably would have been easier than what we did instead.



In terms of advice we might give to future teams based on our experience with this project, the most important thing would be repeating about scheduling. A lot of our issues with this project came from the lack of task scheduling and clear duties. We didn’t meet up as much as we could have, and that meant that a lot of time wasn’t spent as efficiently as it could have been.

Another piece of advice for future teams would probably be to try and solve the main problem first, rather than building up other features. For this project, the main idea was sharing of information from drivers to the other parties. However, in order to make a product that could work in industry, we had to build up a lot of other features, entire other sections of the site, in order to finally reach our main objective. This meant that testing the, ostensibly, most crucial section of the product was held off until the last few weeks Instead of doing all that, it probably would have been better to start from the drivers and information sharing, with the other parties represented by facimiles or stubs, and work out from there.