# Team Exhaustlessness Reflection

# Purpose

The purpose of this document is to provide an overview of Team Exhaustlessness’ project lifecycle, details on collaboration, challenges the team faced, and lessons learned.

# Introduction

Over the course of the semester, our team was tasked to deliver a software product that served as an inventory management system. The goal of this system was to be able to track inventory on multiple levels – corporate, managerial, and customer.

The corporate user would be able to create stores, define inventory items, assign them to designated aisles and shelves, and produce reporting on total inventory and such. The managerial user would be able to process shipments from corporate, track in stock versus overstock items, and run inventory reporting for their respective store. Lastly, the customer user would be able to generate a list of available stores, search for items and their respective shelf locations, and consequently buy items.

As a team, we aimed to create a cohesive software product that was an ecosystem for inventory management where multiple actors could receive value.

# Team organization, members, and responsibilities

|  |
| --- |
| **Team Members** |
| Minh Hang Radetsky |
| Benjamin Gavrilov |
| Julianna Bernardi |

Our team, Team Exhaustlessness, consisted of three members – Minh Hang Radetsky, Benjamin Gavrilov, and Julianna Bernardi. All three team members brought their own respective talents and expertise to the table which benefited discussions and planning sessions. In the beginning of the semester, we would hold meetings every Sunday night at 7pm which served as our weekly check point. At this time we would plan for what would need to be accomplished, if someone had obstacles in their work that they needed help with, and so on. It served as a great time to prepare for the week and divide work load, however, as the month of November rolled over, it became increasingly difficult to meet on Sundays at 7pm due to everybody’s schedule. From there we decided to have ad hoc meetings when new iterations would be released, if someone wanted to pull up a meeting we would schedule one via discord, and we became more fluid in our scheduling.

Given how small our team was, we did not have subteams. However, at times 2 group members would team up to tackle a task together. For example, Ben and Julianna worked closely together to design and implement the database. Ben and Minh would work together in some pair programming sessions to debug.

**Responsibilities**

Given how small our team was, we did not really assign formal roles, we all touched every aspect of the project. Minh took the most initiative and was the first to begin developing the front end and taking the first jabs at the use cases. Reflecting on development, Minh was very active in laying the groundwork which allowed Ben and Julianna to add more. In terms of responsibilities, all of us had a level of involvement on development with Minh taking the most initiative, however, we all gained experience in front end development, back end development, and linking both. We would assign use cases to each team member and they would be expected to deliver. Ben and Julianna would often find themselves reviewing code with Minh and asking for her help and input.

# Process

In the beginning half of the project lifecycle, prior to Iteration 2, our team had a simple but effective process. We would meet on our weekly Sunday call, we would assign work to different team members, and then we would usually re group on Thursday to get a pulse on the work. We did this for the database design, class analysis, and other parts of the planning phase. However, post Iteration 2, we often found ourselves just communicating over Discord, assigning use cases, and meeting when needed.

For some use cases, we assigned a whole use case to team members, so they would learn how to implement a whole functionality from the front end to the backend.

We did have some pair programming earlier on to get team members up to speed and to solve environment setup issues. However, pair programming in our team was on an ad hoc basis.

For requirements, Minh Hang closely communicated with the professor to clarify requirements and attended some office hours to validate the deliverable with the professor. We documented the requirements for each use case in the notion page. Each use case included sample request/response payloads and clarifications on some points.

Code releases were usually reviewed with the team prior to submission deadline.

We would test the front-end by invoking actions from the front-end and earlier on use json-server to mock up the backend for front-end development. On the backend, we primarily used “sam local invoke” to test the business logics. The testing was performed by the members who implemented the code themselves

To get through development, we primarily communicated through Discord.

# Tools

Tools we used:

* One Drive
  + We used One Drive as our main file sharing system that contained our submission documents, final reports, and finalized charts. It was very helpful to have a centralized location where we could have our final versions of our work.
* Discord
  + We used Discord to communicate our progress, express our questions and concerns. It served as our main platform for communication.
* Notion
  + Early in the semester, Minh suggested we use Notion to track our tasks and notes. It truly became a great platform where we could refer to our tasks and Minh created phenomenal “how to” guides that served to be very helpful in development later in the life cycle.
* Lucid chart
  + LucidCharts was used to create the charts we needed for the class diagrams.
* Figma
  + Figma was used to design the storyboards for the design phase of the project.
* Powerpoint
  + Powerpoint was used for more minor items such as presenting the payloads.

In terms of what tools worked best for our team, it would have to be Notion. Notion was very useful in keeping track of tasks and documentation on how to tackle certain tasks. Minh did an excellent job in maintaining Notion and documenting her notes from the AWS tutorials to assist the rest of the team when it came time for their development.

In terms of what tools did not work too well, none of them per say failed, but rather we used the correct tool for what we needed it for. For example, we did not use One Drive to keep a repository of word docs as notes, but we used Notion instead.

We think it is very important for a team to have the correct tools and establish what the infrastructure will look like prior to jumping into development and design. Without having the right tools and infrastructure, there will be an informational mess with notes everywhere and no organization whatsoever.

# Accomplishments

|  |  |
| --- | --- |
| **Use Cases Implemented** | **Description** |
| Corporate – Create Item | Allows corporate user to create an item for stores to stock |
| Corporate – Create Store | Allows corporate user to create a new store |
| Corporate – Remove Store | Allows corporate user to remove/delete an existing store |
| Corporate – Assign Item Location | Allows corporate user to assign item location in the stores, shelf and aisle |
| Corporate – Generate Store Inventory Report | Allows corporate user to generate an inventory report for any given store |
| Corporate – Generate Total Inventory Report | Allows corporate user to generate an inventory report of all total items and stores |
| Corporate – Most Expensive Item | Allows corporate user to see which item is the most expensive |
| Manager – Process Shipment | Allows manager user to process shipments to their store, accept shipments |
| Manager – Fill Shelves | Allows manager user to fill shelves in the store |
| Manager – Show Missing Items | Allows manager to see what items are available to have but are not in store currently |
| Manager – Generate Inventory Report | Allows manager user to generate inventory report for their store |
| Manager – Generate Overstock Report | Allows manager user to produce a report that highlights the over stock |
| Customer – List Stores | Allows the customer to search for stores by GPS |
| Customer – Items on Shelf in Store | Allows the customer to see items in a store and their location |
| Customer – Find Item in Store | Allows the customer to find an item by text |
| Customer – Buy Item | Allows the customer to buy an item |
| Authentication/Authorization | Authenticates corporate and manager users and authorizes them to view appropriate pages. |

## Deliverables

Upon each iteration our team produced a variety of deliverables. Please see below the deliverables we created:

Group Analysis consisted of:

* Class Diagram of the system
* Storyboards of the system UI design
* JSON payloads for each use case to be implemented

First Iteration Report consisted of:

* Our plan on how to implement authentication for next iteration
* A guide on the first set of use cases we implemented (Corp Create Store, Corp Create Item, Corp Assign Item Location, Mang. Process Shipment, Mang. Generate Inventory Report)
* Our database schema and the respective table creation SQL scripts

Second Iteration Report consisted of:

* A guide on how the authentication works for each user, a list of login ID’s, and how to log out
* A guide to the second set of use cases we implemented (Corp List Stores, Corp Remove Stores, Corp Generate Store Inventory Report, Mang. Generate Overstock report, Cust. List Stores, Cust. Find items on Shelves/Aisles, Cust. Find Items in Store by Text)
* Additional notes on how to process shipments and other minor details

Final Iteration Reports consisted of:

* All details and guides mentioned in Second Iteration Report
* A guide on the third set of use cases (Corp. Generate Total Inventory Report, Corp. Most Expensive Item, Mang. Fill Shelves, Mang. Show Missing Items, Cust. Buy Items)

Other deliverables that we wish to deliver but could not deliver:

* Implement Custom Authorizers to protect each API call to lambda function from unauthorized users. However, due to the shortage of time, we only implement front-end, session-based authentication/authorization.

# Reflection

## What worked, what didn’t work

Some of the biggest problems that our team faced were regarding scheduling conflicts, everyone’s busy schedules, and communication. Each of us had super busy work lives and personal lives that made it very difficult to find alignment in not just meeting times, but overall time allocation. Minh was a superstar in communication, but Ben and Julianna found it difficult to be on top of it at times.

We had very productive discussions regarding building class diagram and database schema design in the first few weeks of teamwork. However, the decision around approach on front-end, authentication, back-end development environment was primarily initiated by Minh Hang and got onboarded by everyone as we struggled to find time to discuss and analyze the approaches. These technical decisions happen to be working but they pose a single point of failure due to other team members not having the same level of input to do check and balance.

The team members, however, worked very well when being assigned modularized, self-encapsulated tasks like implementing a use case.

## Our biggest mistake

In regard to mistakes, there necessarily wasn’t one big downfall or one big mistake that led to failure. Instead of mistakes, we had challenges. Our team has a diverse pool of group members and our experience levels varied greatly.

We would communicate issues and roadblocks earlier and more frequently instead of trying to resolve things individually to utilize the wisdom of the group.

We would hope to have more time to understand and connect with each other to improve the team’s cohesion and dynamics.

## Changes we would make

Utilize the Notion’s scrumboard more proactively as a tool to break down tasks, track progress, document business requirements and lessons learned.

Be more proactive in communicating issues or concerns so other team members can participate and help each other.

Follow a more structured review process in terms of quality assurance and code review.

# Lessons learned

1. Alignment of Values. At times it appeared our team had a disconnect of values. At certain points throughout the project some group members would view this project differently and what they want to get out of the project.
2. Some times information gets lost in the Discord text channel so it would make sense to spin up a call to clear up the confusion and misunderstanding.
3. Developing a casual and friendly relationship with your group members makes it much easier to ask for help, work alongside each other, and helps ease anxiety around lack of technical experience.
4. Try to establish a weekly call. We did this in the first half of the semester and it was great to have those weekly checkpoints to always have the project on top of mind.
5. Sharing knowledge as a group to help each other. Document knowledge in writing as a way of self reflection and helping other teammates.