2020

DC MODEL

COMP3006L - SOFTWARE ENGINEERING II

TEAM WOLFPACK

Team members

18209097	Aathif M Nihas Mohomed
18209124	Edmund Tharuka Dinethra
18209090	Jayawardena H Pavith Ashen
18209114	Kotelawala Abheeth Dhamsara
18209094	Shakthi Yasindu
18209087	Dasuni Jayasinghe

Summary of Work Done

We have implemented a fully-fledged distribution center model which allows the worker in the Distribution Center to carry out the day to day task in the simplest and most efficient manner, although the system seems simple for the user the team has implemented a complex and reliable architecture through microservices with additional functionalities, technologies, and complex order processing strategies to ensure high performance and easy maintenance.

Technologies	Tools	Visualization	Additional Implementation
AKKA	VSCODE	Angular UI	Swagger UI
Angular	Postman	Graph Image	Reset Service
Swagger UI	GIT	Swagger	Netflix Eureka
JGraphT	GITLAB	Bootstrap	
	MAVEN		
	Angular CLI		

Project Approach: Agile Scrum:

- Four Sprints - 1 week per sprint.
- Code review at the end of every sprint.
- Planning and reviewing meetings every sprint.

Application Workflow:

- Any configuration with item details will be sent to the simulator Service,
- Simulator Service generates Worker, Map, Item List and Location of items,
- Order Generator service will then push the generated order to Management service,
- Management service will register the worker Actor with worker service,
- Management Service will provide order details to worker Service,
- Worker Service will act as the brain and guide direction to the worker to perform activities in the most efficient manner with complex order processing strategies.

Worker and Management Service

Worker and Management service together perform a complex order processing strategy which is compatible with any configurations. Worker Service is created using AKKA which creates and registers actors with management service.

Simulator Service and Order Generator Service

In Simulator Service the basic endpoints and additional endpoints are implemented and functioning correctly.

Additionally, implemented endpoints:

- item/{id}/location – The location of the item is found through the id
- map/{source}/{target} – the distance between source and target as a number
- map/path/{source}/{target} – returns the list of vertices of the shortest path between source and target
- map/packaging – packaging points are returned

Web GUI

Web GUI is an angular and bootstrap based web application which passes the Rest API calls to the front end. Visualizes all 5 services and provides the ability to control the whole system through functions like reset, configure and play

Additional Technical Features

Netflix Eureka

This improves the code’s deployability because it is not required to hardcode the URIs.

Eureka is Netflix's Service Discovery Server and Client. This has helped us identify the status of each connected servicer of our DC Model. This is an additional technical enhancement done to the architecture to ensure good and stable connectivity.

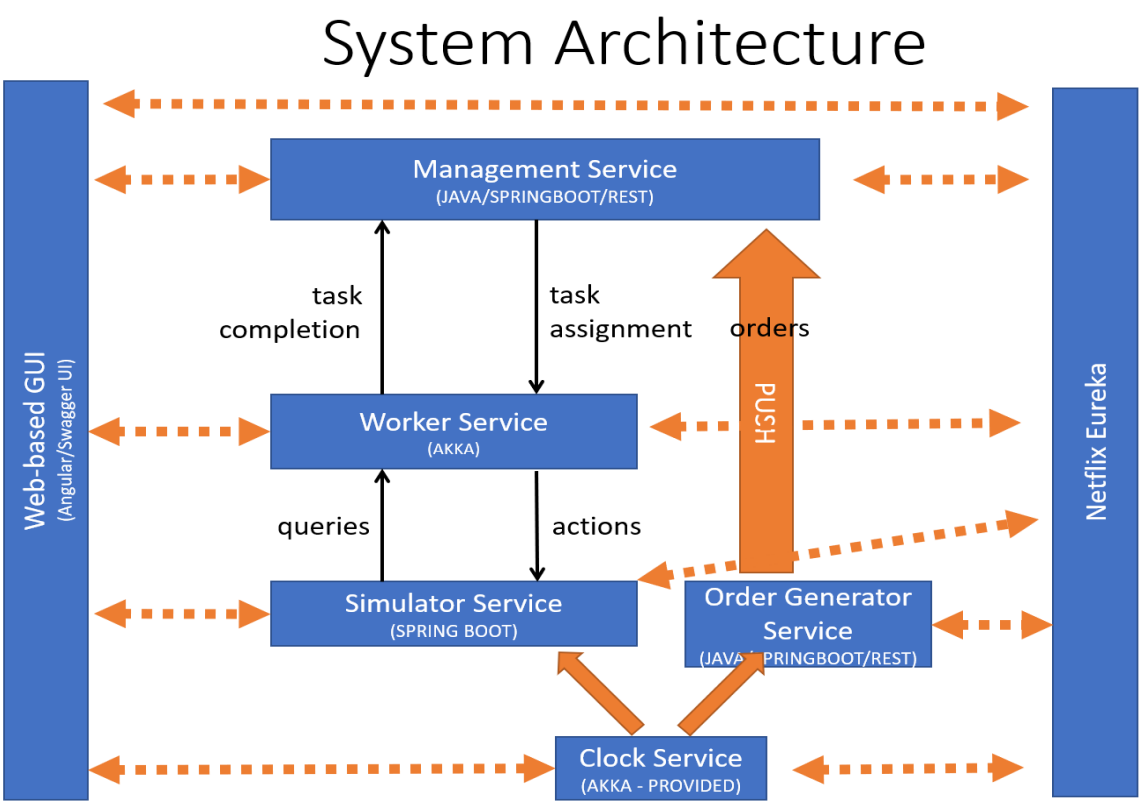
Swagger UI

Swagger UI allows the development team to visualize and interact with the API's resources without having any of the implementation logic in place.

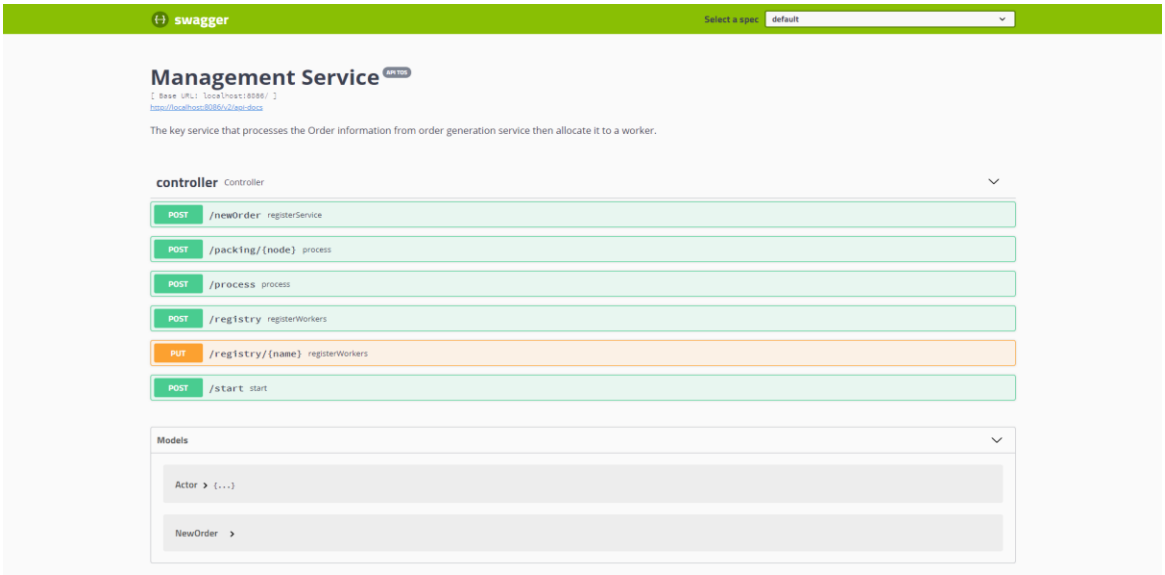
Simulator Service: http://localhost:8082/swagger-ui.html#/

Management Service : http://localhost:8086/swagger-ui.html#/

Order Generator: http://localhost:8083/swagger-ui.html#/



Enhanced Architecture Diagram



Swagger URL to check health of service

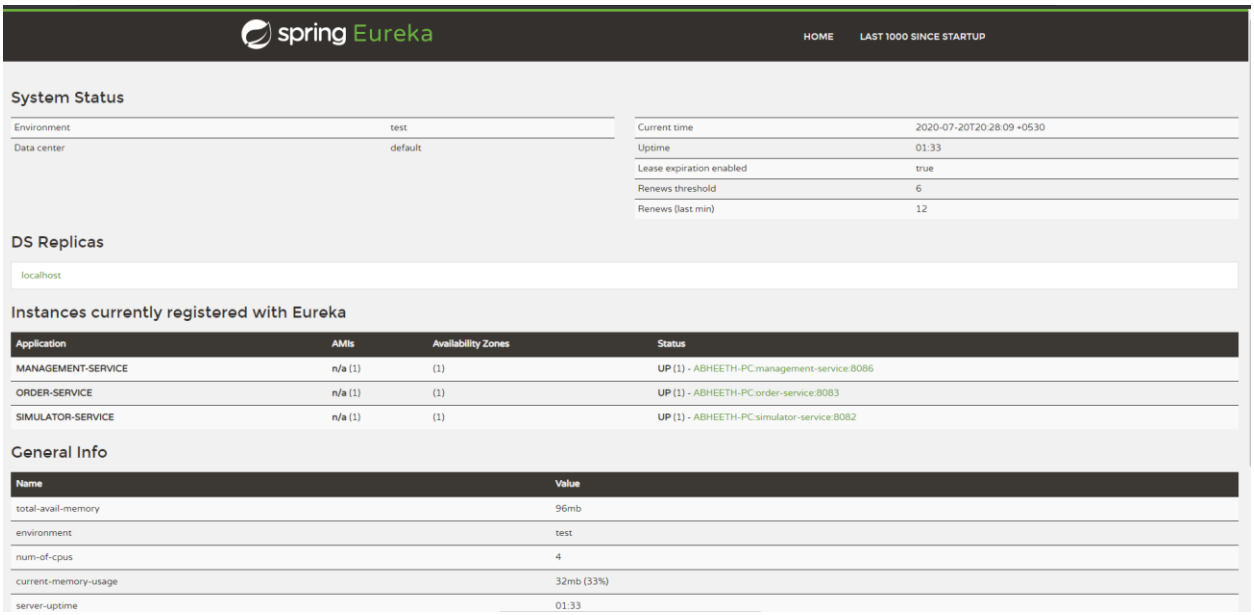


Image of Eureka functioning with the services

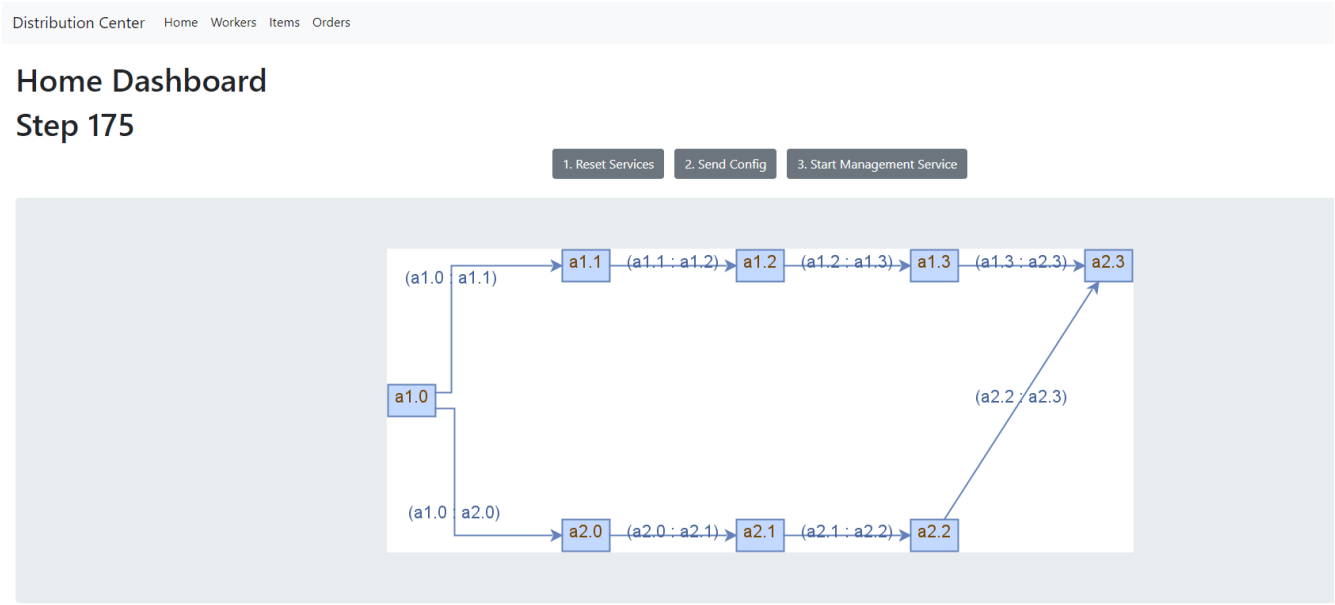


Image of the Home Dashboard with the picture of Image of the worker dashboard

Image of the item list

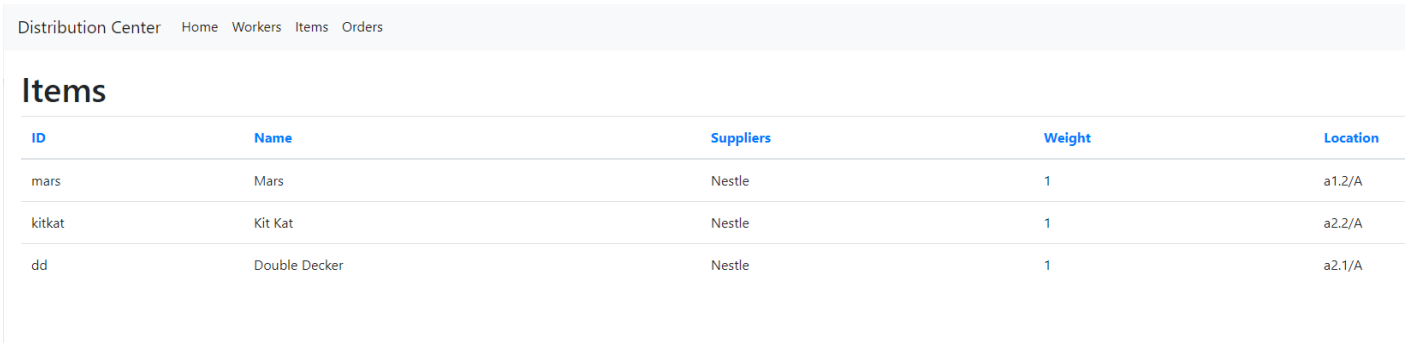


Image of the Orders

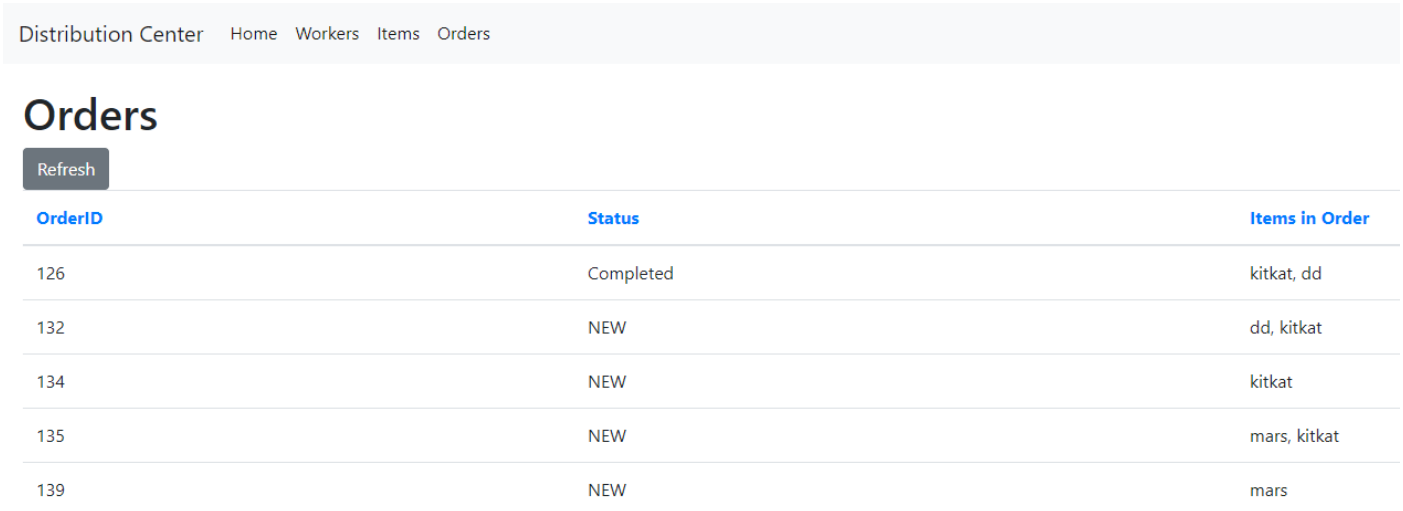


Image of the worker list

Order Processing Strategy

We have introduced complex order processing strategies that work with any configurations and allow room for. scalability in the future. This complex order processing strategy also provides worker efficiency in movement and tolerates the capacity of each and every worker.

How does this complex order processing strategy work?

The order processing strategy implemented in this project contains two additional complex processing strategies that are in addition to the basic order processing strategy. These additional strategies are built with advanced algorithms which makes the work so much easier for the worker and provides a cost-efficient activity strategy for the owner. The order processing strategies are as below:

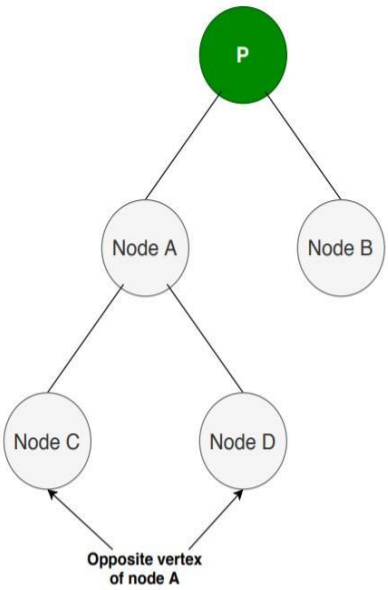
1.Complex item location allocation strategy.

2.Workers brain order processing strategy.

Complex Item location allocation strategy

After the management service is initiated the item list and item information are retrieved from the simulator service. At the initial phase of management service, the aisles and shelves are empty, and the items will be tagged “not assigned”. It is the management service’s duty to allocate items to all the aisles in the most efficient manner to the worker.

As the first task, management service will distribute the items to all the shelves. Here a complex Item allocation strategy is used with the help of advanced algorithms. The distribution and filling of items will be centered from the packaging points. A Tree data structure has been used to identify the nearest shelves to the packaging point. This is identified by the depth of the node tree where the root node is the packaging point. To understand the efficiency behind this algorithm, below diagram and visuals will be utilized



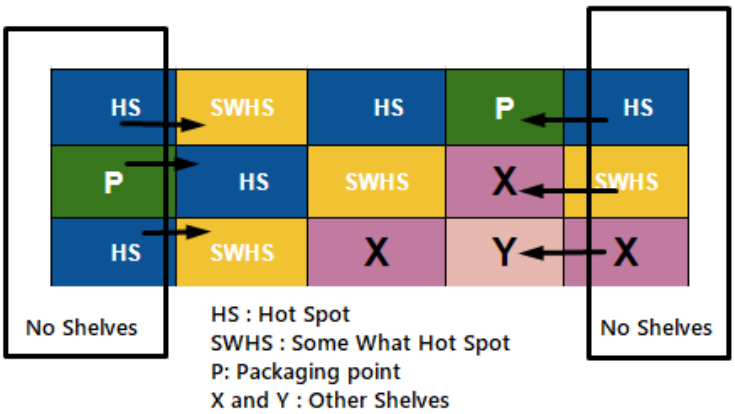


Image on the left: Visualization of the order in which the items will be allocated to

Image on the right: Tree used to take the value of a node from its depth

From packaging points “P” the items will be distributed to slots in colored in blue or called “Hotspot first (HS)”. These are the immediate child nodes for the parent packaging nodes. Which exists at a depth “1 “meaning they are the

closest nodes to a packaging point apart from packaging nodes them self. After filling the blue colored hotspots, the yellow colored “Somewhat hot spots “will be filled. These are the children nodes of the immediate child blue nodes. then X nodes will be filled and then finally the Y nodes will be filled. To obtain clarity if you look at the distance between any packaging point and Y slot, the largest number of steps will be observed.

In a scenario where we have 100 shelves but only 20 items, the items will be allocated around the packaging nodes resulting in shorter routes for workers.

In some scenarios the packaging point itself will act as the shelves and the below mentioned start and end shelves will be utilized for the worker to move around the aisles.

Workers brain order processing strategy

After the items are successfully allocated via the management service. The Worker service is initiated, and it will act as the brain of the worker. Worker Service obtains a list of workers from the simulator service and for each worker it creates an actor using AKKA. Once the actors are created, they are registered in the management service. Management service will receive a push of bulk orders from the order generator service and it will provide the next action items to the worker service. The worker service which is the brain of the worker will now run through the steps from the clock service, maps and items to instruct the worker to perform the maximum task efficiently with minimum time and steps.

- If there is only one worker to perform,

The worker service will compare the weight of the item against the weight capacity of the worker, if the worker can perform the task and is compatible the worker service will display the map and direction of the packaging point.

If the weight of the item is not compatible with the weight capacity of the worker, the worker service will explore the possibility of segregating the item into two parts and allow the worker to carry it one by one.

- If there are more than one worker to perform,

If there are around 10 worker actors registered and each of them has a unique weight capacity, the worker service will compare the weight capacity of each worker against the weight of the item and find which worker will cause minimum weight wastage. And assign the item to the worker with minimum wastage, this is a complex strategy that utilizes the maximum of workers with minimum wastage.

Ex: - If a box of candy bars is 50Kg, the worker service will scan amongst workers and find out who has the weight capacity which is close to the weight of the box of candy bars. If there are John, Alex and Ashen, John has a capacity of 60Kg, Alex has 55KG and Ashen has 52 Kg. Automatically the worker service will assign the task to ashen and ashen will be the most efficient worker with minimum wastage.

If there are two workers and an item is available which has the weight equivalent to the sum of two worker's weight capacity. then the simulator service will instruct both workers to collectively carry the item together.

The order processing strategy implemented here is not the basic strategy, basic strategy only serves the purpose by giving the orders to the workers to perform in a sequential basis. A very complex strategy which acts on its own to serve the shortest path and provides the optimum solution to the worker.

Reflections

Learnings from the project:

- We learned to work under immense pressure calmly as a team.
- Technologies and tools like GIT, Maven, Eureka, Services, Components, Spring-boot, AKKA were new, challenging and very interesting to learn.
- Coding best practices and debugging were very crucial and were only learnt through experience.
- Managing a project with a tight deadline fully remotely online.

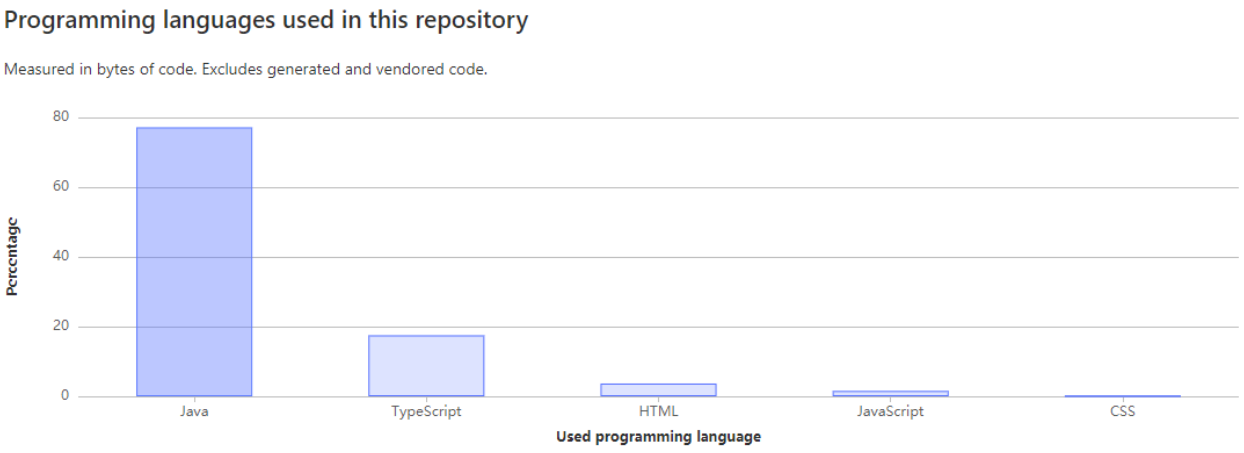
Challenges faced	What was done to over come
Learning and implementing a major scope in a minimum amount of time.	Project deadline was planned, and a daily 15 mins sync up was taken to identify the status.
Obtaining expertise helps in blocking areas.	A lot of self-learning and knowledge transfer sessions.
Commit conflicts in the same java file.	These were resolved by having a merge master in

	the team.
Educating the team on GIT and GIT Practices.	Team learnt and practiced the version control systems.
Because of limited hardware resources some members were unable to initialize all services in their local computer.	The team got together to voluntarily help in testing the code, if a buggy code was identified it was told to the developer.
Conducting daily scrum practices.	Made a hard stop rule to make meetings every day.

If we had to start over, we would:

- Make sure that the whole team understands the principles of the project very well.
- Plan more carefully.
- Focus on the security aspects of the project and provide solutions since we are working on HTTPs Client.
- Implement a Catalina server to host all the services in the same server.
- Add Additional Features to the architecture to make the system powered by AI.

Version control coverage



Commit statistics for **master** Jul 01 - Jul 22

Excluding merge commits. Limited to 2,000 commits.

- Total: **94 commits**
- Average per day: **4.3 commits**
- Authors: **6**

Gitlab Rep :https://gitlab.com/comp3006l/2020/wolfpack/dc-model

Clone link :https://gitlab.com/comp3006l/2020/wolfpack/dc-model.git

Read me : https://gitlab.com/comp3006l/2020/wolfpack/dc-model/-/blob/master/README.md

Video Link :

- https://drive.google.com/drive/folders/1_zZVnTT5Li3-GNF-mUQecwObL47-hD1L?usp=sharing
- https://www.youtube.com/watch?v=qH_AD3unf2Q&feature=youtu.be