



University College Dublin
An Coláiste Ollscoile, Baile Átha Cliath

UNIVERSITY COLLEGE DUBLIN

Bachelor of Science (Sri Lanka)

Software Engineering Project II 2019-2020 (COMP3006L)

Assignment: Group Assignment

Group: RAWANA

Batch 08

“Modelling a Distribution Centre” Assignment

Lecturer:

Assoc. Prof. Rem Collier

| Group Member | Student Name | UCD ID (Student Number) |
|--------------|----------------------------------|-------------------------|
| Member 1 | P.I.B. Wijekoon | 18209062 |
| Member 2 | R M De Zoysa Siriwardena | 18209115 |
| Member 3 | Iresh Maduranga Ranaweeera | 18209109 |
| Member 4 | A.A.D Piyumal Shamen Saparamadhu | 18209110 |
| Member 5 | I.H. Buddhi Ranga Perera | 18209137 |

Table of Contents

| | |
|---|----|
| Concept of Behind the Naming | 1 |
| Report Content | 2 |
| 1.0 Summary of Work Done | 3 |
| 1.1 Summarize the Workflow | 3 |
| 1.2 Project Functionalities | 3 |
| 1.3 Work Done Against the Marking Rubric | 4 |
| 1.4 Modifications in System Architecture | 5 |
| 2.0 Order Processing Strategy | 5 |
| 2.1 Explain the Order Processing Strategy You Used? | 5 |
| 2.1.1 Automated System (Fake Order Generating Service) | 5 |
| 2.1.2 Manual System (Custom Order Generating Service) | 5 |
| 2.2 How Does it Works? | 5 |
| 2.2.1 Automated System (Fake Order Generating Service) | 5 |
| 2.2.1 Manual System (Custom Order Generating Service). | 8 |
| 3.0 Postman API Endpoints | 9 |
| 4.0 Reflection | 10 |
| 4.1 Experiences Gained | 10 |
| 4.2 What did We Learn | 10 |
| 4.3 What Challenges did We have to Face | 10 |
| 4.4 What Would You Have Done Differently If You Could Have Started Over? | 11 |

Concept of Behind the Naming



Rawana was the Legendary Asura king, father of all "Asura", "Naga", "Makara" and "Deva" people who lived in Sri Lanka. Under King Rawana's rule Sri Lanka saw great advancements in science and medicine. Legend tells that King Rawana had the power of ten people since he had a more developed state of mind. King Rawana descends from the native "Hela" race also known as "Surya" the people who worshiped the sun like the "Mayan" people.

We are the sign of his throne and symbol of his believers believing that nothing is impossible.

COMP3006L: Software Engineering II

<Insert Team Name>

Team Member 1
Student # 1

Team Member 2
Student # 2

Team Member 3
Student # 3

Team Member 4
Student # 4

Summary of Work Done

Please summarize here what your team has done for the project. Using the marking rubric as a guide, try to state clearly what you have done. Do not provide too much detail on the order processing strategy you used because this will be discussed next. If you have modified the system architecture (e.g. added a component like Netflix Eureka) then state this here and include a revised system architecture diagram that illustrates how the extra components are used.

Max: 1 page + figures

Order Processing Strategy

Explain the order processing strategy you used. How does it work? What are the key steps/interactions between services? If it is not your implementation of the basic order processing strategy, explain how it differs? Provide diagrams to illustrate how your strategy works.

Max: 3 pages + figures

Reflections

Please summarize your experiences on the project: what did you learn; what challenges did you have to overcome; what would you have done differently if you could have started over.

Max: 1 page

Content to be Submitted

Please indicate the status / location of each of the following deliverables:

| | |
|--------------------------|---|
| README | <i>In the root folder of each GitLab Project – should explain how to run the project</i> |
| WALKTHROUGH VIDEO | <i>Please record a short (< 5 minute) video of the system running and upload this into the same location as this document.</i> |
| REPORT | <i><u>This document – please submit it in PDF format</u></i> |
| CODE | <i>All code must be submitted in the Gitlab group provided.</i> |

1.0 Summary of Work Done

1.1 Summarize the Workflow

Initially we thought that this system functionalities were manual because we do not have a clock API. We started the planning stage of the project the day it was announced and created a repo in GitHub.com for our version control purposes. We started to build our system with manual data processing components.

We analyzed the system according to the manual data flow and the initial starting process of the system is with an online ordering shopping cart. When GitLab access were given we had implemented the project halfway through.

After further clarification with Professor Rem, we got a clear picture of the entire system. We got to know that the system should be an automated system rather than a manual process. Allocating the AKKA clock model is the key to starting an automated system.

Then we started to change the workflow by converting manual to automated and that was not an easy task since we had to make a lot of changes to make the final output.

According to the given instructions, we started from the clock and we created orders to generate process, management process, worker process, and the simulator process combined with visualizing frontend according to the MVC Architecture with the Agile developments (Regular sprints). Please refer to the GitLab for more info about the group contributions. We did the project with the target timeframe and we tried our best to achieve that.

1.2 Project Functionalities

| | |
|-----|---|
| 1. | Clock service could be started by clicking the Play button on the Dashboard. Two features namely Pause, and Reset will be implemented in the future. |
| 2. | Get real-time information of the current clock step. |
| 3. | Get real-time information about registered services to the clock registry. |
| 4. | Automated order generation according to the percentages requested. |
| 5. | Able to customise the order generating percentages from the backend. Frontend customisation of the percentages will be implemented in the future. |
| 6. | Manual order generation interface. Any order could be generated manually for any item and quantity. |
| 7. | The orders generated manually and automatically can be viewed in the new order interface. |
| 8. | Orders could be searched via order ID |
| 9. | Supervisor can assign orders to workers manually. e.g.: Supervisor assign orders to Picker |
| 10. | Workers can self-assign orders. |
| 11. | Users (Supervisor, Picker, Packer, Shipper) can login to the system via the user login interface in the website. |
| 12. | Only username is required for login. Password is not necessary since the website is only available to use within the company by employees. |
| 13. | User gets customized interfaces according to their job roles. They are not allowed to view other user interfaces. Supervisor could access all employee interfaces. |
| 14. | Supervisor can allocate orders manually. |
| 15. | Supervisor can monitor worker progress. |
| 16. | Supervisor can generate orders manually from the system by clicking the clock start button. |

| | |
|-----|---|
| 17. | Supervisor can generate orders manually from the interface given. |
| 18. | Supervisor can view orders assigned to workers from the interface. Executing orders / Pending orders |
| 19. | Supervisor can view worker progress (Assigned orders to completed orders) |
| 20. | Workers can view all orders they are assigned to and their states. Executing orders / Pending orders / New orders. |
| 21. | Workers can view the current state of the order they are assigned to. Current state Picking/Packing/Shipping? |
| 22. | Supervisor can view all orders and their current state. e.g.: Picking state is completed or not. |
| 23. | Order details are displayed. When you click on an order Items are shown with quantity. |
| 24. | View customer details associated with order. When you click on an order customer details are displayed. |
| 25. | All endpoints are visible to the simulation service (Use postman to grab endpoints) Multiple user endpoints |
| 26. | Worker picking path visualization. When a picker accepts any order with Item and quantity it generates actions for the worker to perform in the current state and next state. Then worker actions represent as steps. The steps are visualized in images (How to go the item location) Starting point to > packing stage. |
| 27. | Supervisor can add new Items (Order Items) to the system. |
| 28. | Supervisor can add new users (Supervisor, Picker, Packer, Shipper) to the system. |
| 29. | Pickers Only can use the simulator Service. When checking endpoints of users use picker usernames ex: User Iresh & Panduka |

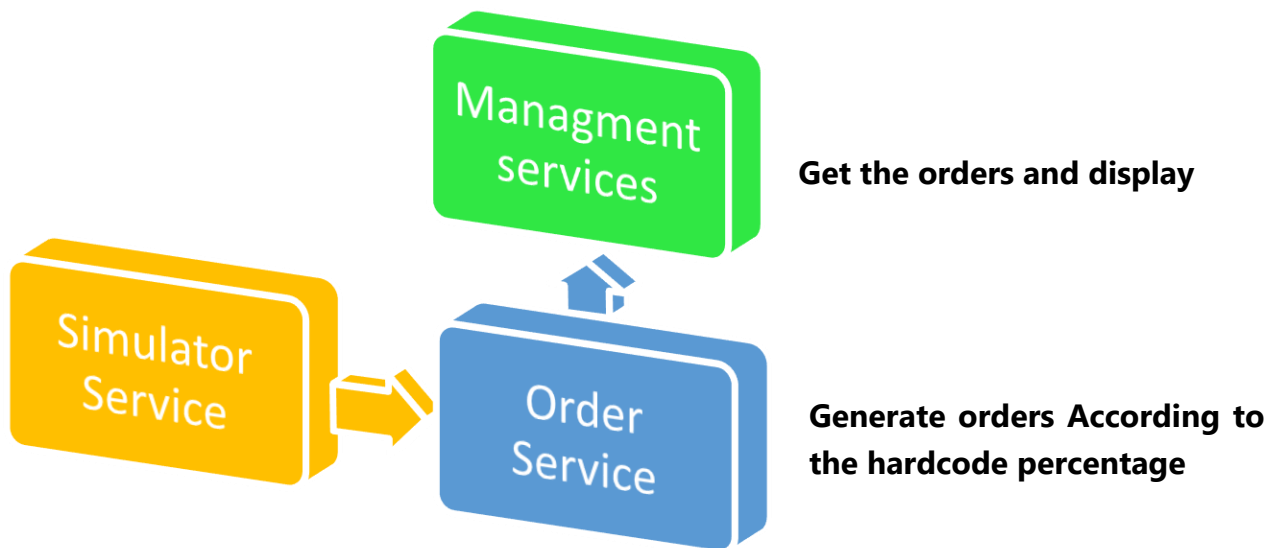
1.3 Work Done Against the Marking Rubric

| No | Service / Component | Development |
|-----|--|--|
| 01. | Simulator and Order Generating Service | Additional necessary functionalities added Ex: Multiple worker action details handling |
| 02. | Worker Management Service | A more complex order processing strategy |
| 03. | Graphical Interface | Allows visualisation of the state of ALL services*** Work play button |
| 04. | Technical | We added additional technology Microservices / Rest controllers |
| 05. | Project Management | Excellent quality documentation submitted. Good reflection on lessons learned. Appropriate use of version control. |

We believe that we developed our project up to this state to the very best of our ability with the given timeframe. We only used the provided tutorials provided by Professor Rem and from online resources without the help of a third party. All are the things developed here is from our knowledge and researching.

1.4 Modifications in System Architecture

Both Automated + Manual system added, Any work that has been done automatically could be done manually as well.



2.0 Order Processing Strategy

2.1 Explain the Order Processing Strategy You Used?

We have used automated system + manual system

2.1.1 Automated System (Fake Order Generating Service)

When the clock service is run it automatically start generating steps. Generated step count added to calls the order generated service and start generating orders according to the percentages given.

2.1.2 Manual System (Custom Order Generating Service)

This system is created for generating orders manually from a shopping cart interface.

2.2 How Does it Works?

2.2.1 Automated System (Fake Order Generating Service)

When we start the clock service it automatically starts the engine of the system.

It means it start the clock server.

Then it generates step counts as follows

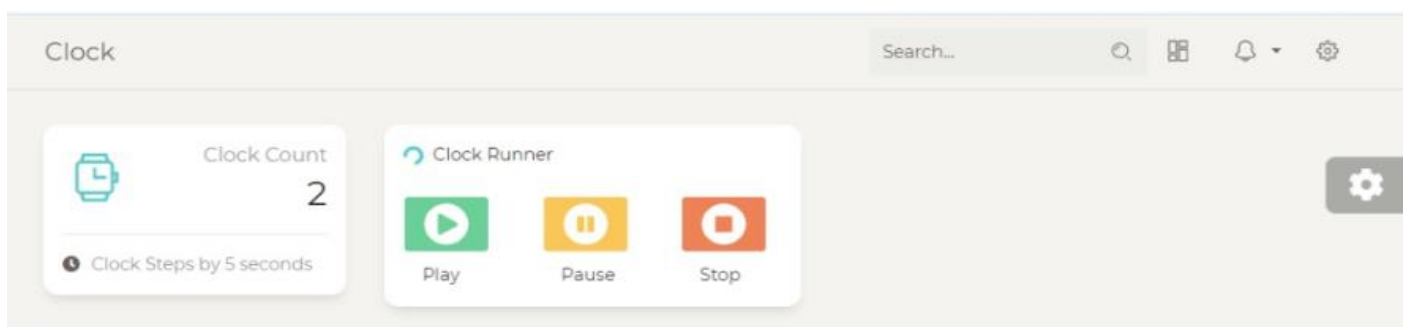
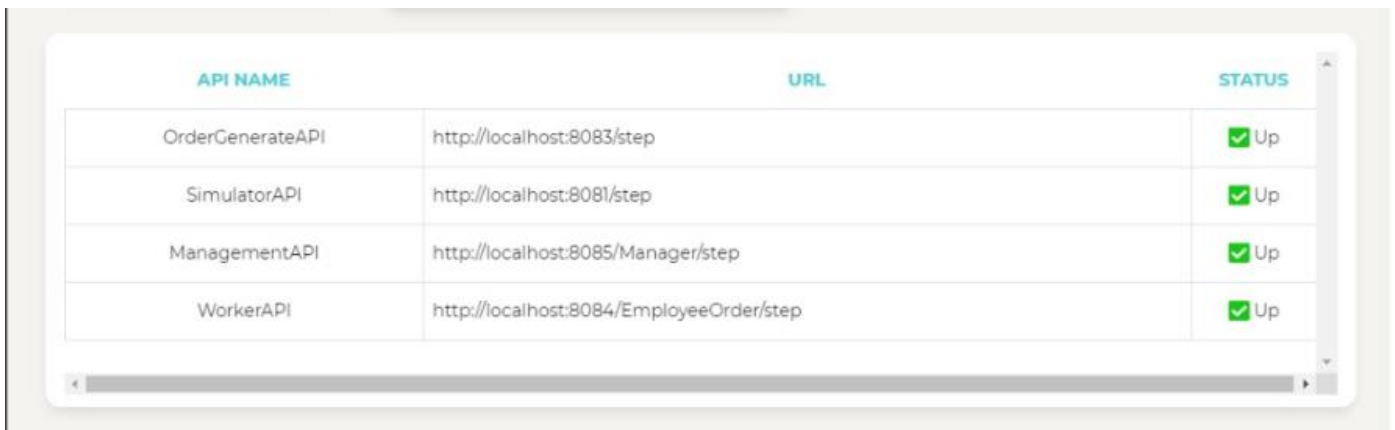


Figure 1: Step Count by Using Clock

We can register the all servicers using the Clock Registry service.



| API NAME | URL | STATUS |
|------------------|--|--------|
| OrderGenerateAPI | http://localhost:8083/step | ✓ Up |
| SimulatorAPI | http://localhost:8081/step | ✓ Up |
| ManagementAPI | http://localhost:8085/Manager/step | ✓ Up |
| WorkerAPI | http://localhost:8084/EmployeeOrder/step | ✓ Up |

Figure 2: Services Registration Process

Among the registered services it shows the OrderGenaratedAPI which means that the order generate API already registered to system.

This means that there is a connection between OrderGenaratedAPI to clock service.

Then we get the step count which generates by the clock service every 5seconds.

First it gets all item details mentioned in the stock and generate orders according to items.

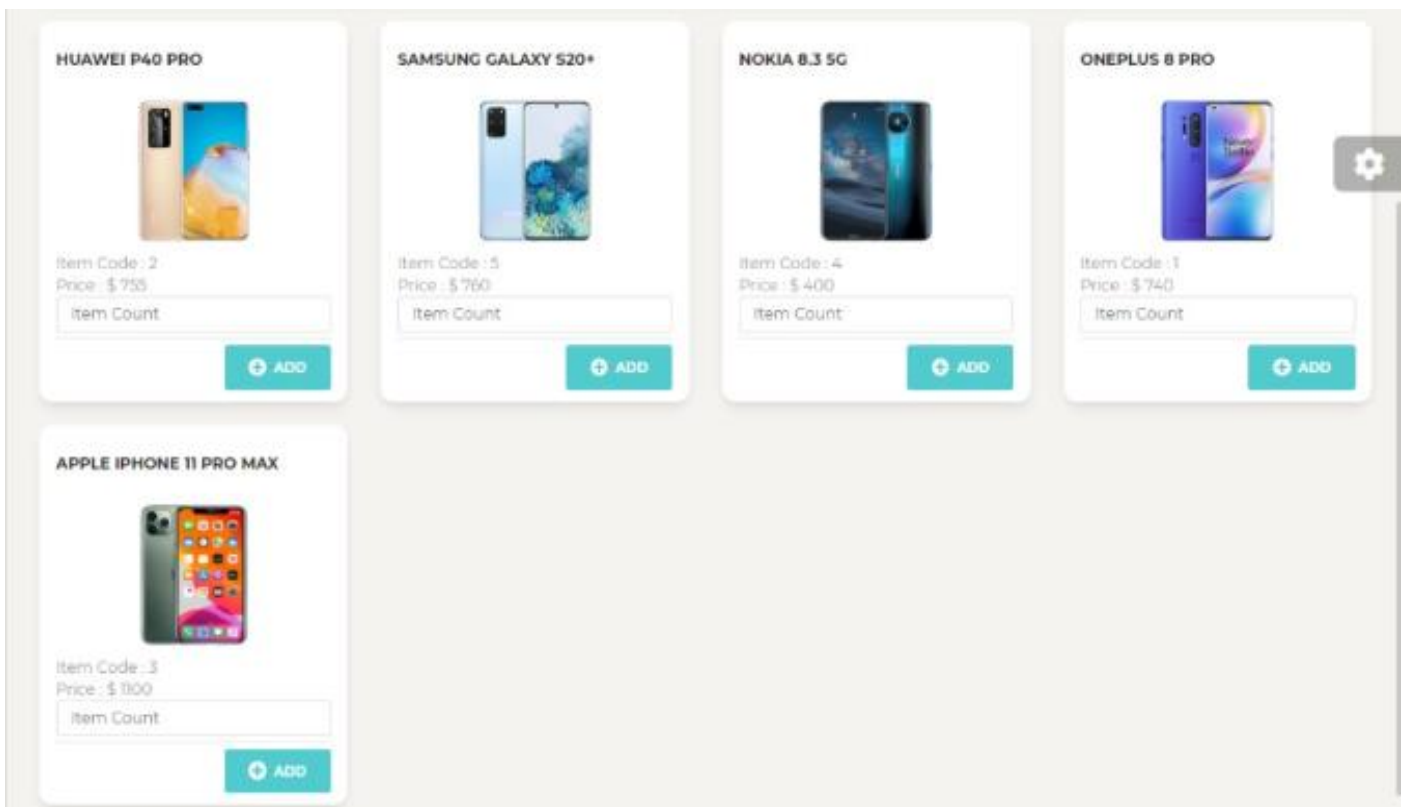


Figure 3: Product Store for Manual Orders

It starts to generate orders from the backend. We generate the orders according to the percentages given in the instructions. If you need to change the order percentage, you can change it from the backend.

The order items are generated in random order.

Generated items are saved in a data table called Orders.

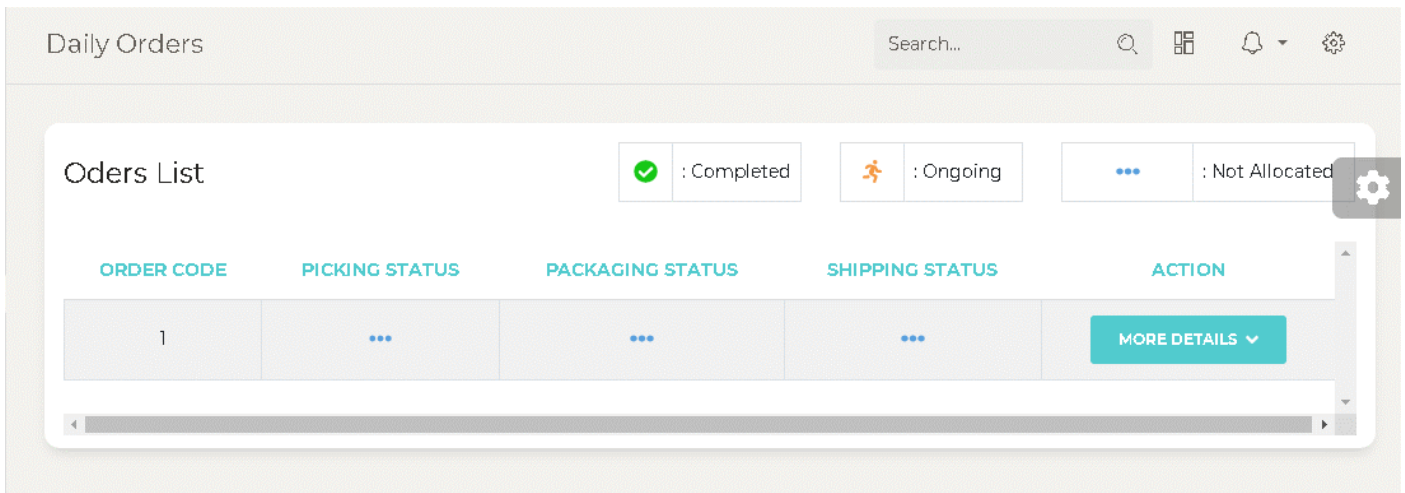


Figure 4: Daily Orders Delated Listed Table with Status of Each Process

Orders can be assigned to users automatically according to the user's weight.

We have two users in picker position to whom no orders are assigned yet.

After clicking the clock play button, it generates new orders and assign orders to unassigned workers..

When assigning orders, the systems check for 3 parameters in a user.

| | |
|----|--|
| 1. | First the system checks whether the worker has the required weight limit |
| 2. | The system checks for workers who does not have orders assigned. |
| 3. | The system selects the workers who has the least amount of orders completed. |

If a worker fulfills these conditions, then order automatically gets assigned to worker (Picker worker).

We have implemented the automated system to do order assigning tasks automatically as well as manually.

User can login according to his job roll, Picker/Packer/Shipper/Supervisor (Admin facility granted).

- First Picker log in to system by using picker username.
- Then he can see what the newly created orders against allocated to his account by clicking orders for picker.
- He can see available orders in order pool.
- When going through the Assign orders there is a sub section called Assigned orders.
- By clicking the resume, the order, there is a subsection called Full order details.
- There are customer details in one side and Item details in the order.

By clicking the get route detail picker can get map that split to steps with the instructions that needs to be followed.

When giving the routing steps, we assume that all pickers starts at the location "a1.0" and packing stations are at locations "a1.5" and "a2.5". The first route is given by the simulator to the closest item location in the order. After that other routes to other items in order are given by the simulator. Finally, the simulator gives the shortest route to a one of the two packing stations from the picked final item location. Each action is submitted by the worker service to simulator service.

By visiting those steps, he can pick every item one by one.

After collecting items and delivered to the packing location, he can complete the order by clicking the confirm order complete.

In packing stage packing user must log as a packer and can collect orders by accepting the orders. This step is fully run by the packer, he can pull orders and accept them. Like the picker he can complete the order by clicking confirm order complete.

Then it is available for the shipping stage. The shipper can log into the system and get new orders. Then the shipper can deliver the order to the customer and complete the order.

This marks the end of the order.

2.2.1 Manual System (Custom Order Generating Service).

Orders are can be generated in manually in the interface given and Order assigning and order processing as mentioned above.

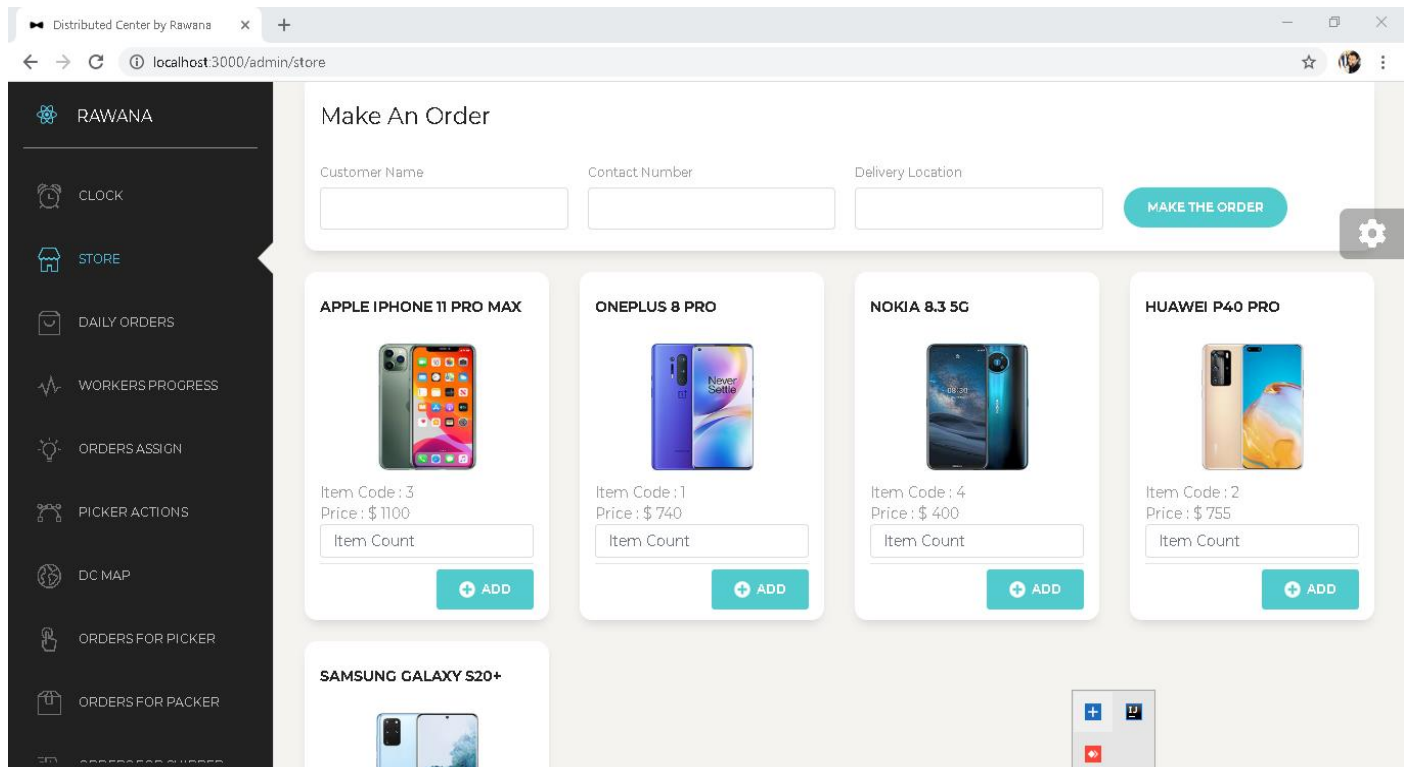


Figure 5: Store for Manual Orders

3.0 Postman API Endpoints

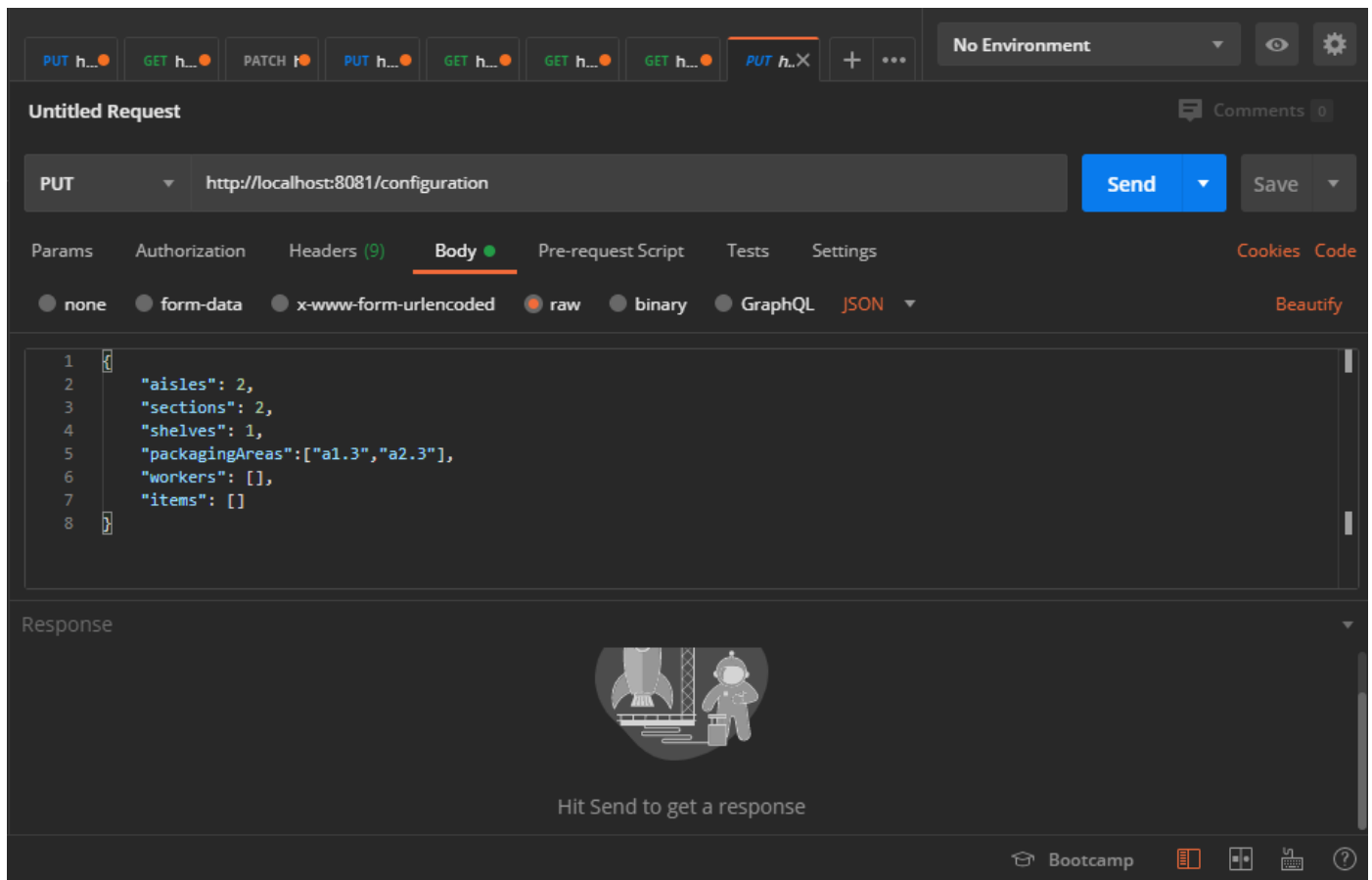


Figure 6: Postman Requests & Responses

We have slightly modified the original JSON file in order to use it with our system. Every JSON response can be viewed in the video attached.

4.0 Reflection

4.1 Experiences Gained

- Manage time as a team in a very tight schedule
- Working as a team remotely during the entire development stage of the project by communication through the internet. (We didn't have physical contact but communicated by VOIP, virtual meetings, and remote desktop sessions).
- Good understanding what is Actor programming and handling distributed systems.
- Good understand about Microservices
- Web Frontend development.

4.2 What did We Learn

- Microservices Framework - Spring Boot
- React web programming
- Knowledge about REST API
- Version control

4.3 What Challenges did We have to Face

The hardest challenge that we could not overcome was using Actor programming in the worker service.

[Automatically Assign Orders to Users from Clock Service AKKA Automatically.](#)

We were not able to use AKKA framework to pass messages between the worker services to management service and simulation service.

We couldn't assign orders according to the clock, so we just push our instruction using spring Boots

We tried our level best to find tutorials and do research and understand the concepts, but we were unable to get a complete understanding of the concepts of AKKA.

We were not able to find anyone who knew about the AKKA framework in Sri Lanka to get some help to implement it to the worker service.

[Travelling Salesman Implemented by Dijkstra's](#)

In simulation process we created graphs using Dijkstra's Algorithm, and not the Travelling Salesman Problem Algorithm. But we have implemented the Dijkstra's Algorithm to collect multiple items from multiple locations.

Worker start Initially stage as 1.0 and ending packaging locations 1.5 / 2.5

[We Didn't Have Time to Visualize the Path Graphically And Animation](#)

We could only implement the visualization by generating pictures of the steps taken by the picker.

[Auto Order generating Algorithm Implementation](#)

In the beginning we used to generate items via a percentage generator. It can generate items by given percentage and can be customized from the backend. But there were some hidden errors and the orders generated did not satisfy the given percentages. Then we used a simple Order Generating Algorithm Function to gain the requested percentages.

[Clock Jar Execution](#)

When creating Jar files from the clock service we couldn't execute the Jar files it displayed an error saying that it couldn't find the main class. That was because we had to add Apache Maven Shades Plugin to pom file.

Java Project Folder Path

There is a folder difference between when running a project inside a clone downloaded from GitLab and a zip downloaded from GitLab.

Front end Issues

Backend data represent issues with React.

We used React because it is most famous and widely used tool for web development in the world. But when we applied it into this project, we have many issues with React since it is less flexible with the system we are developing.

Difficulty in handling Tabular data.

Difficulty in handling React grid view.

We couldn't get visualize paths.

When using AKKA

We were not able to fully understand actor programming and AKKA framework.

Database Architecture

Initially we planned of using one database for each microservice. Then we planned to host the system in the cloud. But there is an issue since we need static IP's, but we can only use dynamic IP's. Therefore, we taught of using a single database (Monolithic) for all the microservices without allocating a different database for each microservice. This was due to the complexity when developing microservices with different databases and make our development process easier.

Memory Leakage React

When mounting and unmounting frontend data with backend data with related components it is not work perfectly always and we couldn't handle that issues very well. It means component life cycle was not managed well. But Front-end work well in syntaxy this is an error.

4.4 What Would You Have Done Differently If You Could Have Started Over?

- We should learn what is Actor Programming very clearly.
- We would implement the system to be fully automatic without the semi-automatic system that we developed.
- Play Push Stop buttons will be further developed and integrated with AKKA.
- Can Visualize and track worker movements.
- Assign and manage the time schedule correctly without rushing.
- Better testing environment with updated modules and data bases
- Remote login to distributed databases.
- For the visualization use languages that are more flexible like Python.
- Using latest frontend frameworks like Angular etc.
- Order generation mechanism should control via advanced mechanisms we can improve this further.
- Visualizing workers and Items via map can be modified.
- Online shopping cart be integrated and can be used as a practical DC model by implementing the manual order processing.
- Using a Map API, we can track order status until it returns to the customer.
- Docker containerization
- Security can implement in frontend with HTTPS using SSL certificate.
- Using Amazon Web Server Hosting.