

---

# Cargo Load Planner

---

Implementation Details Document

Project Guide: Prof. N.S. Narayanaswamy

Author: Vikram Manjare

IITM

AUGUST 24, 2021

# Index

1	Overview	2
1.1	Motivation	2
1.2	Concepts	2
1.3	Assumptions	2
1.3.1	Common assumptions	2
1.3.2	Special assumptions	2
1.4	Software Requirements	3
2	MySQL Database	4
2.1	How to import fleetdb schema using MySQL workbench?	4
2.2	How to import data using CSV files?	4
2.3	Tables	4
2.4	Triggers	4
2.5	Keys	5
2.5.1	Primary Keys	5
2.5.2	Foreign Keys (relevant)	5
3	Backend Java Application	6
3.1	How to import and setup project in Eclipse IDE?	6
3.2	Packages and Classes	6
3.3	Input	7
3.4	Output	7
3.4.1	Report in Console	7
3.4.2	JSON Output File	8
4	Frontend Visualization	10
4.1	Screenshots (Sample data - how step by step visualization can be displayed?)	10
5	References	14

# 1 Overview

## 1.1 Motivation

- Are you shipping a lot of air? Are you sure everything will fit? Are you tired of wasting your time when drawing and calculating stuffing plans? Do you still use pen and paper or are you just guessing if the cargo fits the container or truck?
- Our goal is to design a sophisticated but also easy to use cargo loading optimization software, that helps to plan the way cargo is loaded into containers or trucks.

## 1.2 Concepts

Vehicle	<ul style="list-style-type: none"><li>▪ A vehicle transports items from place to place</li></ul>
Cargo	<ul style="list-style-type: none"><li>▪ Goods carried in a vehicle</li><li>▪ Cargo boxes are placed inside a container</li></ul>
Container	<ul style="list-style-type: none"><li>▪ Container for cargos, volumetric capacity is limited by their metal frame</li><li>▪ A vehicle has a container</li></ul>
Order	<ul style="list-style-type: none"><li>▪ Order is a group of cargo boxes with same source and destination</li><li>▪ Order is basic unit meaning all cargo boxes in an order should be in one vehicle</li></ul>

## 1.3 Assumptions

- Note that not all the constraints specified in the assumptions noted here are implemented yet

### 1.3.1 Common assumptions

- Cargoes/boxes are cuboids with same/different sizes.
- Boxes must be arranged completely within the container and parallel to its side walls.
- Boxes have properties like stackable/non-stackable, floor-only, rotatable/non-rotatable.
- Priority rules based on packing sequence, e.g. boxes with large size are firstly arranged before the medium and small size boxes respectively.
- Boxes on the top are supported by the box beneath.

### 1.3.2 Special assumptions

- Long distance orders will be loaded first and unloaded last, similarly short distance orders will be loaded last and unloaded first.
  - ◆ This will be handled using **cargoes\_position\_priority** in the **orders** table.
  - ◆ Higher priority → Short distance order → Load last (at back - Door is at back)
  - ◆ Lower priority → Long distance order → Load first (at front)

- ◆ This may affect the volume efficiency of the container, still considering the cost of unloading other orders in between this should be given first priority.
- ◆ **Note: Not implemented yet**
- The orders can be and should be loaded optimally into the container using the algorithm only at Source.
  - ◆ The algorithm can be used to load/optimize new orders at intermediate points but it will not be economically feasible as it may require unloading of all orders first and then loading all again along with new orders.

## 1.4 Software Requirements

- Tested on OS: Windows 8.1
- MySQL DB Community Edition - <https://dev.mysql.com/downloads/mysql/>
  - ◆ MySQL Workbench - <https://dev.mysql.com/downloads/workbench/> is also recommended for doing SQL operations using GUI.
- Java JDK (latest/Java SE 15+) - <https://www.oracle.com/in/java/technologies/javase-downloads.html>
- Eclipse IDE with JPA software installed
  - ◆ Eclipse IDE for Java Developers download - <https://www.eclipse.org/downloads/>
  - ◆ To install JPA in IDE follow the steps:
    - Eclipse IDE Menu → Help → Install new software → Search JPA by selecting work with: All Available Sites → Install all Dali JPA software options under Web tools platform (WTP)
- The other Java libraries used in the project are included in **lib** folder inside the project itself (referencing them is explained in [Backend Java application](#) section).

## 2 MySQL Database

### 2.1 How to import fleetdb schema using MySQL workbench?

- MySQL workbench menu → Server → Data import → Import from Self-Contained File → Select “**fleetdb schema Dump20210824.sql**” file which is in **db** folder inside project folder.
- You should see **fleetdb** database created in Navigator (left pane of workbench)

### 2.2 How to import data using CSV files?

- In the same **db** folder inside project folder, there are 3 sample data (.CSV) files which need to be imported in MySQL
  - ◆ In Navigator pane, expand fleetdb → expand Tables
  - ◆ Right click vehicles table → Table data import wizard → select **Vehicles – EBFT.csv** (under **db** folder) in File path → Next → Use existing table (**fleetdb.vehicles**) → Next → Next (no need to change anything) ... → Finish
  - ◆ Now repeat the same procedure for Orders (**Orders – EBFT.csv** ↔ **fleetdb.orders**) first and then for Cargoes (**Cargoes – EBFT.csv** ↔ **fleetdb.cargoes**).
  - ◆ **Note: The order of importing table matters due to constraints and triggers, import in same the order as specified above**
- You should see data imported in the tables

### 2.3 Tables

- Only 3 tables in the fleetdb database are relevant for the CLP project
  - ◆ fleetdb.vehicles
  - ◆ fleetdb.orders
  - ◆ fleetdb.cargoes
- You can see/browse detailed schema for those tables using workbench, columns are self-explanatory from the column names

### 2.4 Triggers

- 3 Triggers are defined on Cargoes table

cargoes_BEFORE_INSERT	Auto-set stack_weight (default is same as weight of box) and volume for the box when it's being added to the table
cargoes_AFTER_INSERT, cargoes_AFTER_DELETE	Auto-update cargoes_count, cargoes_total_weight and cargoes_total_volume in orders table when a box is added/removed to/from the Cargoes table

## 2.5 Keys

### 2.5.1 Primary Keys

Table	Key column(s)
fleetdb.vehicles	vehicle_id
fleetdb.orders	order_id
fleetdb.cargoes	order_id + box_id

### 2.5.2 Foreign Keys (relevant)

Key column	Refers to
fleetdb.orders.vehicle_id	fleetdb.vehicles.vehicle_id
fleetdb.cargoes.order_id	fleetdb.orders.order_id

## 3 Backend Java Application

### 3.1 How to import and setup project in Eclipse IDE?

- Download complete zip and extract into any folder
- **Make sure that you have installed JPA** as described in [Software Requirements](#) then follow the steps given below
- Eclipse IDE Menu → File → Import → General → Existing Projects into Workspace → Select project folder → Mark search for nested projects option → Finish.
- It's unlikely but if you get any missing libraries error, select Eclipse IDE Menu → Project → Properties → Java Build Path → Libraries → Make sure all 8 libraries under **lib** and **lib/asciitable** folders are referenced under **Modulepath**, otherwise reference them by “**Add Jars...**” button by selecting Modulepath and adding them from respective folders (lib and lib/asciitable) → Similarly check if 8 libraries under **lib/jpa** folder are referenced under **Classpath**, if not then add them all in similar way under Classpath
- Now open file src/main/java/META-INF/persistence.xml (select source view)
  - ◆ In the <Properties> tag make necessary changes to the values of the following properties according to how you have configured MySQL DB
    - javax.persistence.jdbc.url – URL of fleetdb database – Default value is jdbc:mysql://localhost:3306/fleetdb
    - javax.persistence.jdbc.user – User name for MySQL server
    - javax.persistence.jdbc.password – Corresponding user password
- You are all set to execute the program now and should be able to run it without any errors.
- Note that there can be other unknown errors you may face depending upon OS/IDE that you are using, all these are well tested on Windows 8.1/Eclipse for Java Developers IDE, you can see debug/trace in console and rectify accordingly.

### 3.2 Packages and Classes

	Default package	
1	CLPBasicMain	▪ Main class file
	clpBasic.eclipselink.entities	
2	Cargo	▪ JPA class entity corresponding to Cargoes table in the DB
3	CargoPK	<ul style="list-style-type: none"><li>▪ Required JPA Embeddable entity corresponding to composite primary key (order_id + box_id) of Cargoes table</li><li>▪ Used in Cargo entity</li></ul>
4	Container	▪ JPA class entity corresponding to Vehicles table in the DB

5	<b>Order</b>	<ul style="list-style-type: none"> <li>▪ JPA class entity corresponding to Orders table in the DB</li> </ul>
	<b>clpBasic.eclipselink.services</b>	
6	<b>DBQueryHandler</b>	<ul style="list-style-type: none"> <li>▪ Defines various methods for fetching input from the DB</li> </ul>
	<b>clpBasic.Utilities</b>	
7	<b>CargoStaticMethods</b>	<ul style="list-style-type: none"> <li>▪ Some static utility methods that are not used as of now</li> </ul>
8	<b>DataGenerator</b>	<ul style="list-style-type: none"> <li>▪ Class for random test data generation</li> </ul>
9	<b>Display</b>	<ul style="list-style-type: none"> <li>▪ Utility class for displaying debug trace properly in console</li> </ul>
10	<b>JSONHandler</b>	<ul style="list-style-type: none"> <li>▪ Class that defines methods for creating output JSON file for visualization purpose</li> </ul>
11	<b>StaticMemory</b>	<ul style="list-style-type: none"> <li>▪ Some static contents that are not used as of now</li> </ul>
	<b>EB_AFIT</b>	
		<ul style="list-style-type: none"> <li>▪ Original algorithm designer - (BALTACIOGLU, 2001)</li> <li>▪ Referenced C# repository: <a href="https://github.com/davidmchapman/3DContainerPacking">https://github.com/davidmchapman/3DContainerPacking</a></li> </ul>
12	<b>AlgorithmPackingResult</b>	<ul style="list-style-type: none"> <li>▪ Class for holding details of output of CLP algorithm, used as part of ContainerPackingResult</li> </ul>
13	<b>AlgorithmType</b>	<ul style="list-style-type: none"> <li>▪ Class for enumeration of CLP algorithms, currently only EB_AFIT is the algorithm</li> </ul>
14	<b>ContainerPackingResult</b>	<ul style="list-style-type: none"> <li>▪ Class for holding details of container and its AlgorithmPackingResults, used in PackingServices</li> </ul>
15	<b>EB_AFIT</b>	<ul style="list-style-type: none"> <li>▪ Java implementation of Air Force Bin Packing Algorithm Paper</li> </ul>
16	<b>IPackingAlgorithm</b>	<ul style="list-style-type: none"> <li>▪ Interface for a CLP algorithm</li> </ul>
17	<b>PackingServices</b>	<ul style="list-style-type: none"> <li>▪ Class defining packing service, which will call CLP algorithm</li> </ul>

### 3.3 Input

- Input is fetched from 3 relevant (vehicles, orders, cargoes) MySQL DB tables (DB schema i.e. input format is described earlier)

### 3.4 Output

#### 3.4.1 Report in Console

- Program report (same format as used by EB\_AFIT algorithm) is shown in the console window (self-explanatory)



- Following is sample console input (for sample data)

*** REPORT ***	
ELAPSED TIME	Almost 0 sec
TOTAL NUMBER OF ITERATIONS DONE	0
BEST SOLUTION FOUND AT ITERATION	1 OF VARIANT 1
TOTAL NUMBER OF BOXES	9
PACKED NUMBER OF BOXES	6
TOTAL VOLUME OF ALL BOXES	1063296
CONTAINER VOLUME	838656
BEST SOLUTION'S VOLUME UTILIZATION	838656 OUT OF 838656
PERCENTAGE OF CONTAINER VOLUME USED	100.00
PERCENTAGE OF PACKED BOXES (VOLUME)	78.87
WHILE CONTAINER ORIENTATION (X,Y,Z)	(104,96,84)

L	W	H	ID	X	Y	Z	L1	W1	H1
70	104	24	1-1	0	0	0	104	24	70
14	104	48	1-5	0	0	70	104	48	14
70	104	24	1-2	0	24	0	104	24	70
70	104	24	1-3	0	48	0	104	24	70
14	104	48	1-6	0	48	70	104	48	14
70	104	24	1-4	0	72	0	104	24	70

- Second table displays list of packed boxes with their details
  - ◆ L, W, H are original dimensions of box
  - ◆ ID is the complete box ID (order\_id + box\_id)
  - ◆ X, Y, Z are coordinates of packing location of box wrt to container (container's origin is at left bottom corner, similarly coordinates are positions for lower left corner of box)
  - ◆ L1, W1, H1 are packing dimensions representing packing orientation of box, may differ from original dimensions

### 3.4.2 JSON Output File

- Program also outputs JSON file containing packed boxes in order

- Following is sample JSON file (for sample data)

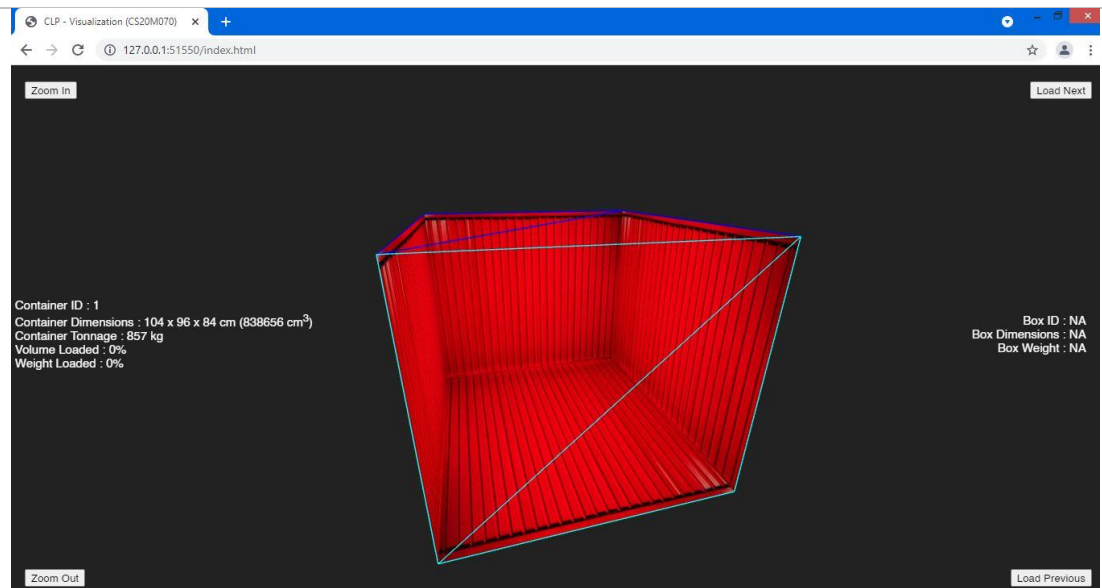
```
{
  "container_id": 1,
  "container_length": 104,
  "container_width": 96,
  "container_height": 84,
  "container_tonnage": 857000,
  "data": [{
    "id": "1-1",
    "length": 104,
    "width": 24,
    "height": 70,
    "weight": 10000,
    "x": 0,
    "y": 0,
    "z": 0,
    "color": "navy"
  }, {
    "id": "1-5",
    "length": 104,
    "width": 48,
    "height": 14,
    "weight": 16000,
    "x": 0,
    "y": 0,
    "z": 70,
    "color": "navy"
  }, {
    "id": "1-2",
    "length": 104,
    "width": 24,
    "height": 70,
    "weight": 10000,
    "x": 0,
    "y": 24,
    "z": 0,
    "color": "navy"
  }, {
    "id": "1-3",
    "length": 104,
    "width": 24,
    "height": 70,
    "weight": 10000,
    "x": 0,
    "y": 72,
    "z": 0,
    "color": "navy"
  }, {
    "id": "1-6",
    "length": 104,
    "width": 48,
    "height": 14,
    "weight": 16000,
    "x": 0,
    "y": 48,
    "z": 70,
    "color": "navy"
  }, {
    "id": "1-4",
    "length": 104,
    "width": 24,
    "height": 70,
    "weight": 10000,
    "x": 0,
    "y": 72,
    "z": 0,
    "color": "navy"
  }
  ]
}
```

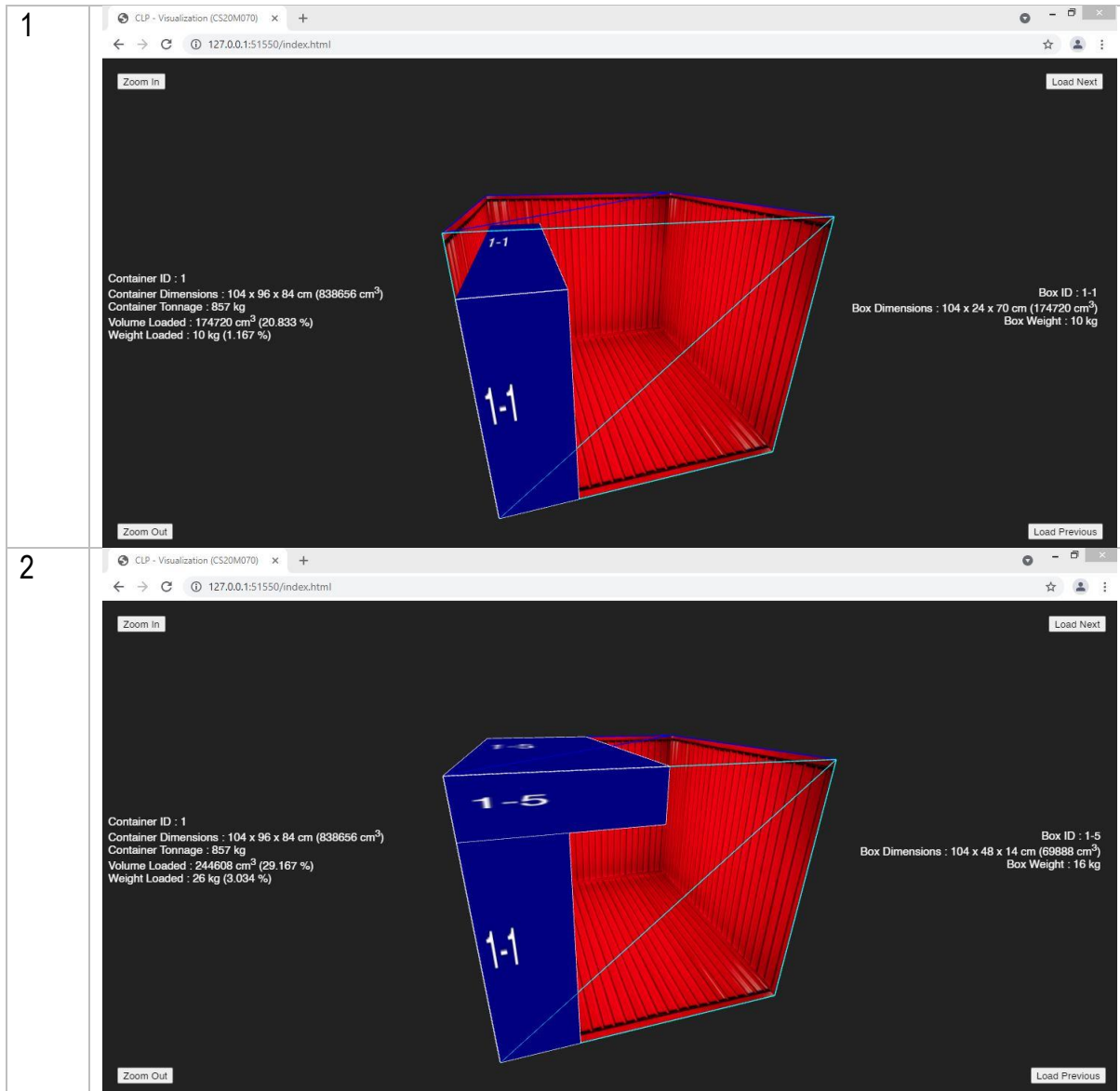
- Data format is as displayed (self-explanatory), this file is used for visualization purpose
  - Note that packed (not original) dimensions are written to the JSON file

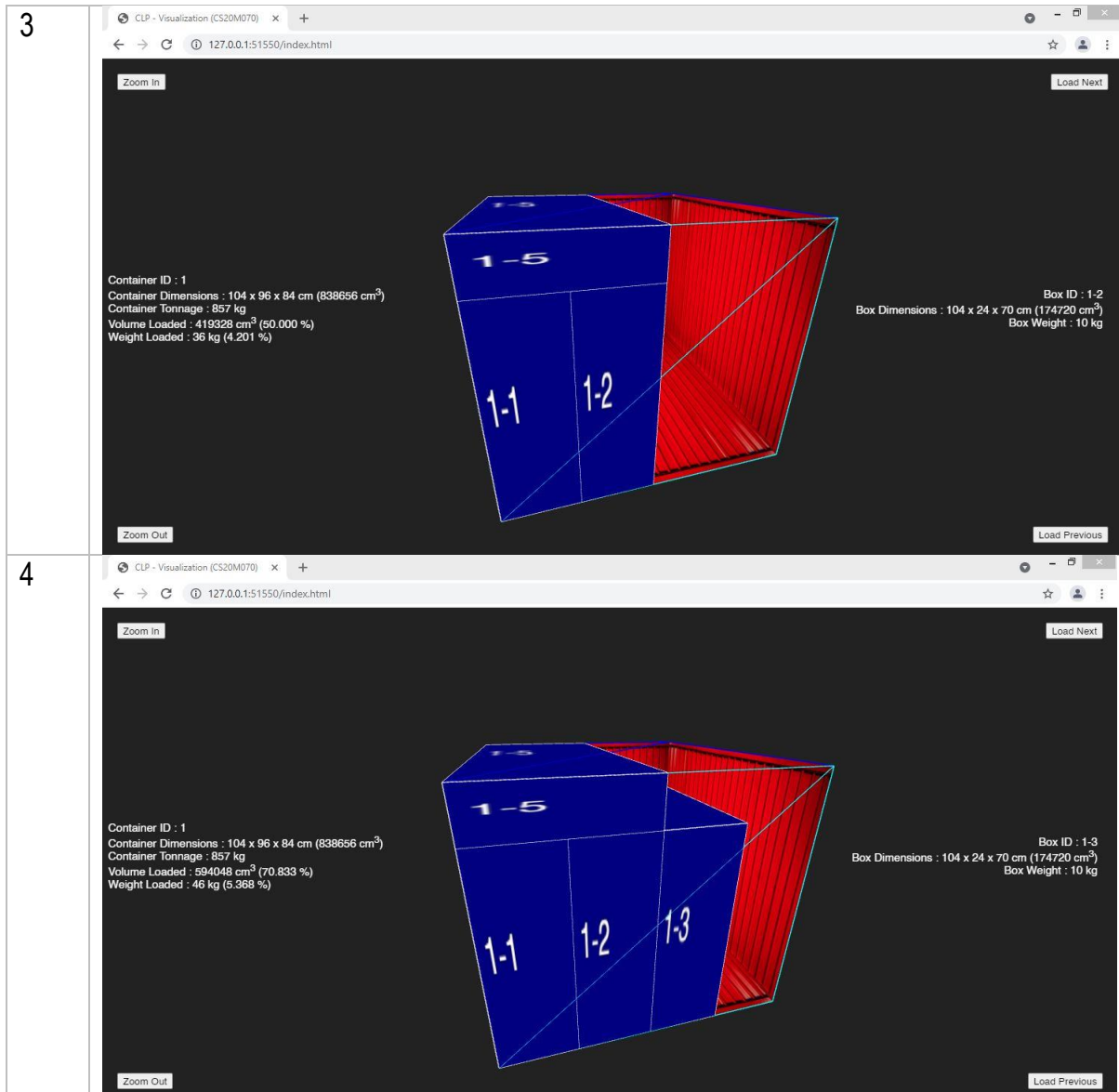
## 4 Frontend Visualization

- Three.JS, a cross-browser JavaScript library and application programming interface which is used to create and display animated 3D computer graphics in a web browser using WebGL, is used for Frontend 3D visualization.
- It uses JSON file produced by Backend Java application and displays 3D view of container and boxes in the web browser window.
- Note that JSON file need to be present in the same folder where the 3js code is.
  - ◆ JSON file can be directly written in 3js folder by setting up **JSONHandler.DirectoryPath** in the Java application code
- Also, we need to manually set **JSONFilePath** variable to the required JSON file in the **index.js** file.
- Finally using JSON file requires localhost server running.
  - ◆ Code editor Brackets - <http://brackets.io/> is used for testing purpose.
- **Note that JS code is not included here in the repository.**

### 4.1 Screenshots (Sample data - how step by step visualization can be displayed?)

Step	Screenshot
0	





5

CLP - Visualization (CS20M070)

127.0.0.1:51550/index.html

Zoom In

Load Next

Container ID : 1

Container Dimensions : 104 x 96 x 84 cm (838656 cm<sup>3</sup>)

Container Tonnage : 857 kg

Volume Loaded : 663936 cm<sup>3</sup> (79.167 %)

Weight Loaded : 62 kg (7.235 %)

1-5

1-6

1-1

1-2

1-3

Box ID : 1-6

Box Dimensions : 104 x 48 x 14 cm (69888 cm<sup>3</sup>)

Box Weight : 16 kg

Zoom Out

Load Previous

6

CLP - Visualization (CS20M070)

127.0.0.1:51550/index.html

Zoom In

Load Next

Container ID : 1

Container Dimensions : 104 x 96 x 84 cm (838656 cm<sup>3</sup>)

Container Tonnage : 857 kg

Volume Loaded : 838656 cm<sup>3</sup> (100.000 %)

Weight Loaded : 72 kg (8.401 %)

1-5

1-6

1-1

1-2

1-3

1-4

Box ID : 1-4

Box Dimensions : 104 x 24 x 70 cm (174720 cm<sup>3</sup>)

Box Weight : 10 kg

Zoom Out

Load Previous

13 | [CLP](#)

## 5 References

BALTACIOGLU, E. (2001). *THE DISTRIBUTER'S THREE-DIMENSIONAL PALLET-PACKING PROBLEM: A HUMAN INTELLIGENCE-BASED HEURISTIC APPROACH*. Ohio: AIR FORCE INSTITUTE OF TECHNOLOGY, Wright-Patterson Air Force Base, Ohio.