DISTRIBUTED STORAGE SYSTEM

Supervisors :
Mrs. Laura Thiele
Prof. Mr. Sascha Bosse
Mr. Matthias Pohl

Abhijith Remesh - 221424
Baizil Mulakkampilly - 221544

Table of Contents

Problem Definition

- Manufacturing Companies now tend to automate and digitalize their enterprise and production processes.

- Digital integration of production facilities enables to get new information and useful knowledge about their processes in real time.

- Development of a software system for the management of inventory levels which follows a distributed storage architecture to enable high flexibility and scalability to the inventory management system.

- Warehouse components to be represented as corresponding software components which can be integrated into the software system.

Requirements

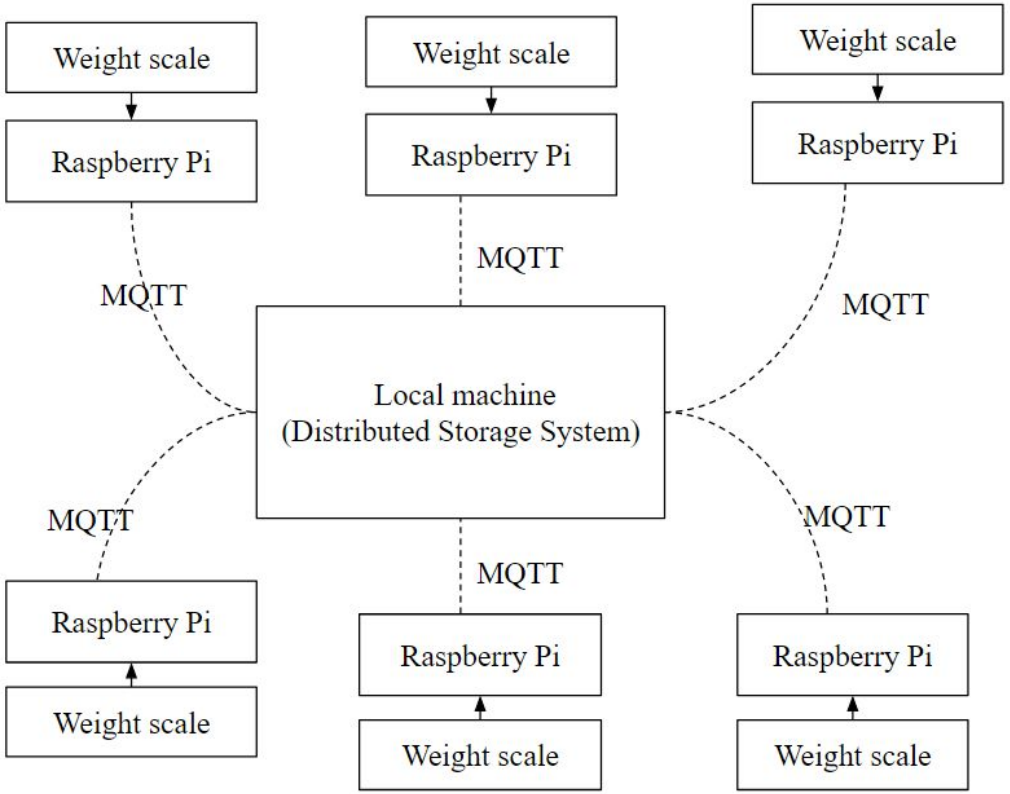
- Interfacing the scaling device with raspberry pi and parsing the data in specific data formats.
- Development of database containing the feature information of warehouse components
- Developing the logic to identify the counts of warehouse components accurately based on weight comparison.
- Setting up real time MQTT communication between the Raspberry Pi and the local machine.
- Implementing the dashboard to view the inventory levels of each warehouse component.
- Developing the user interface facility to add new warehouse component, select a component and delete an existing component.
- Development of a database for storing the inventory log information.
- Storing the inventory log information into the database as per user input.
- Utility to delete and download the inventory log information on user input.
- Several approaches to identify article weight mismatch.

Hardware and Software utilities

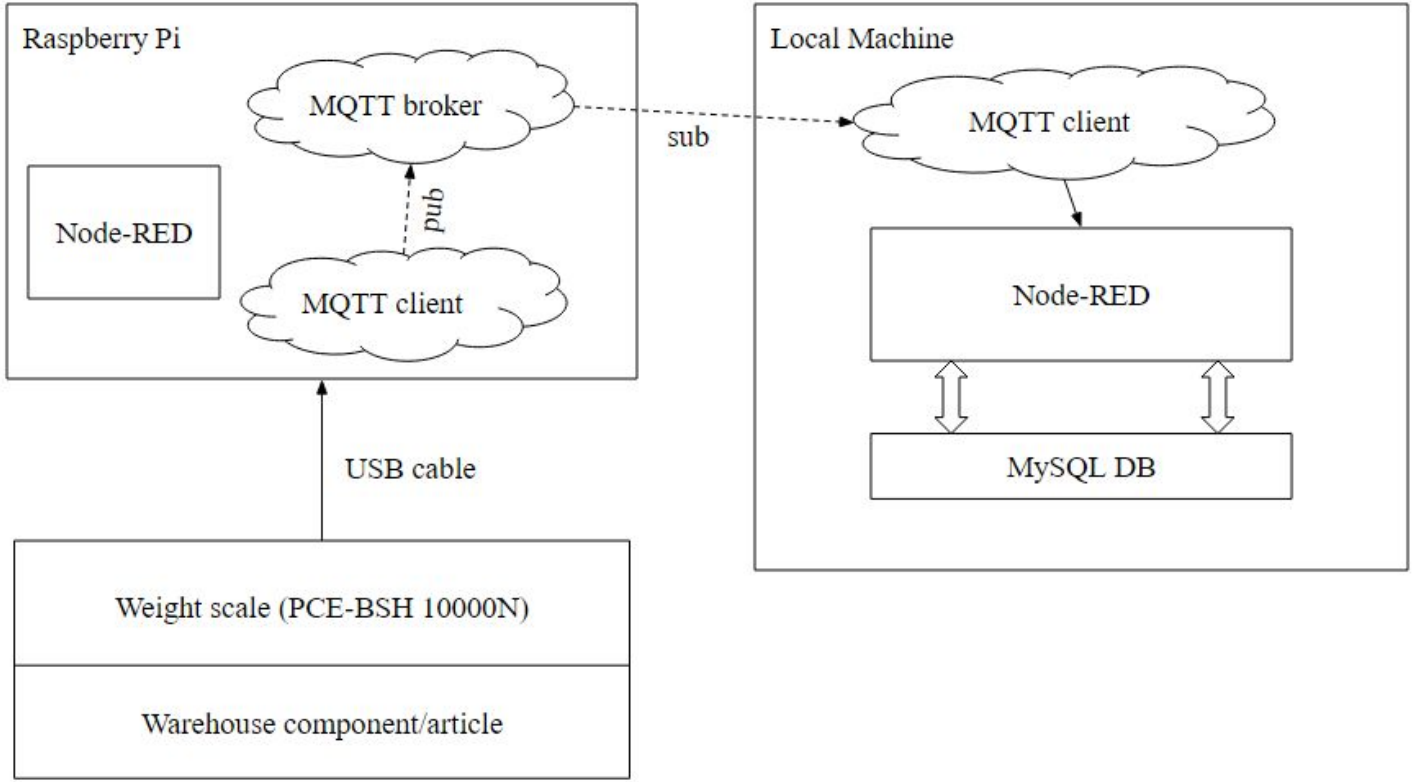
Node-RED

MYSQL

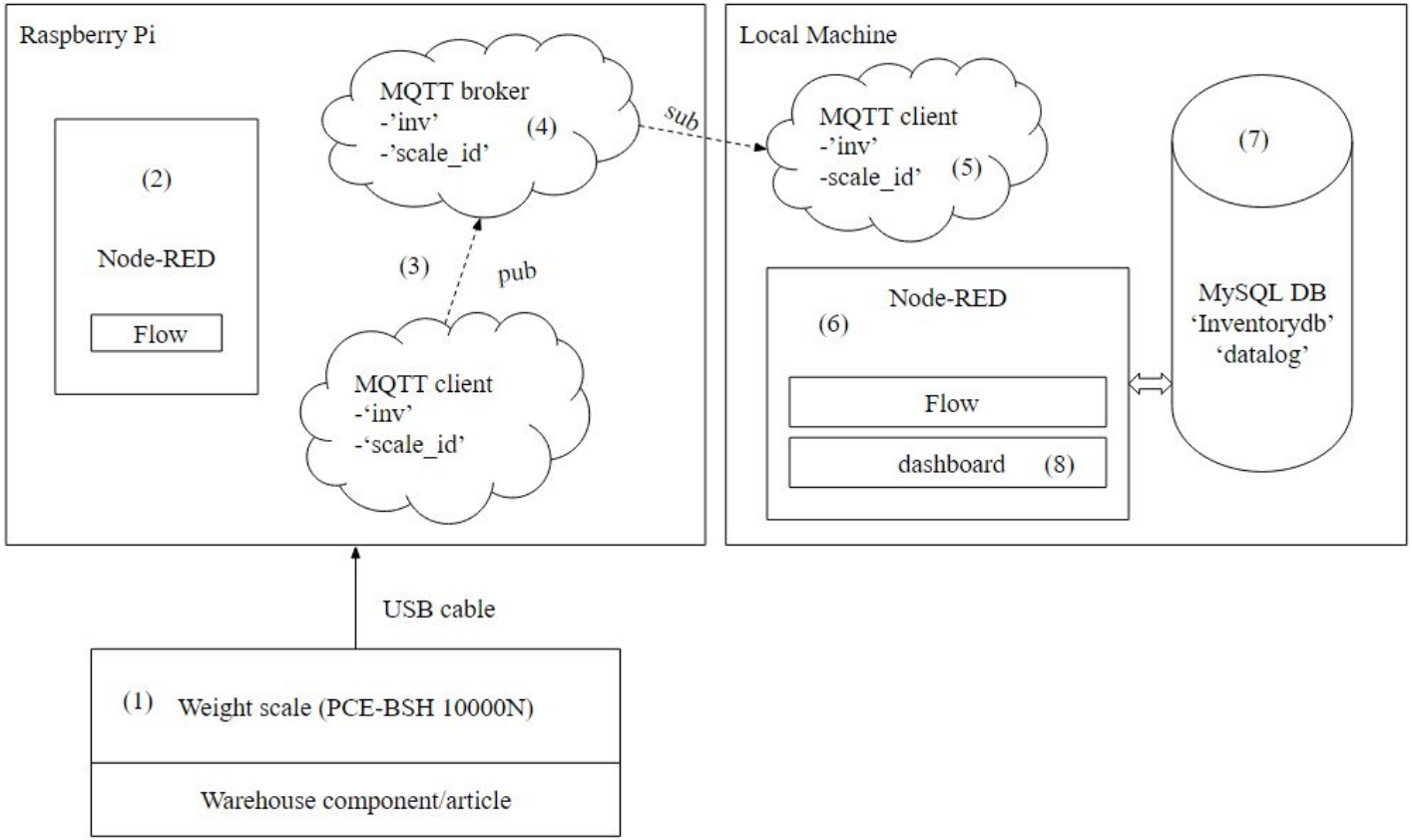
Architecture - High Level (Abstract)



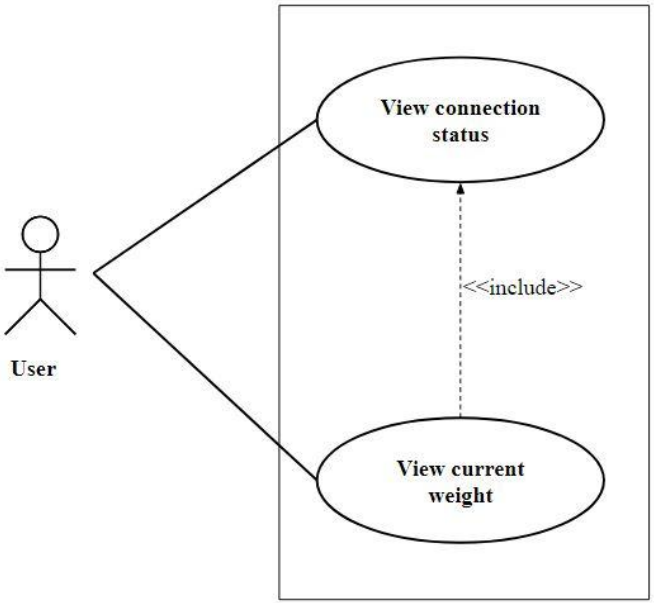
Architecture - Detail level



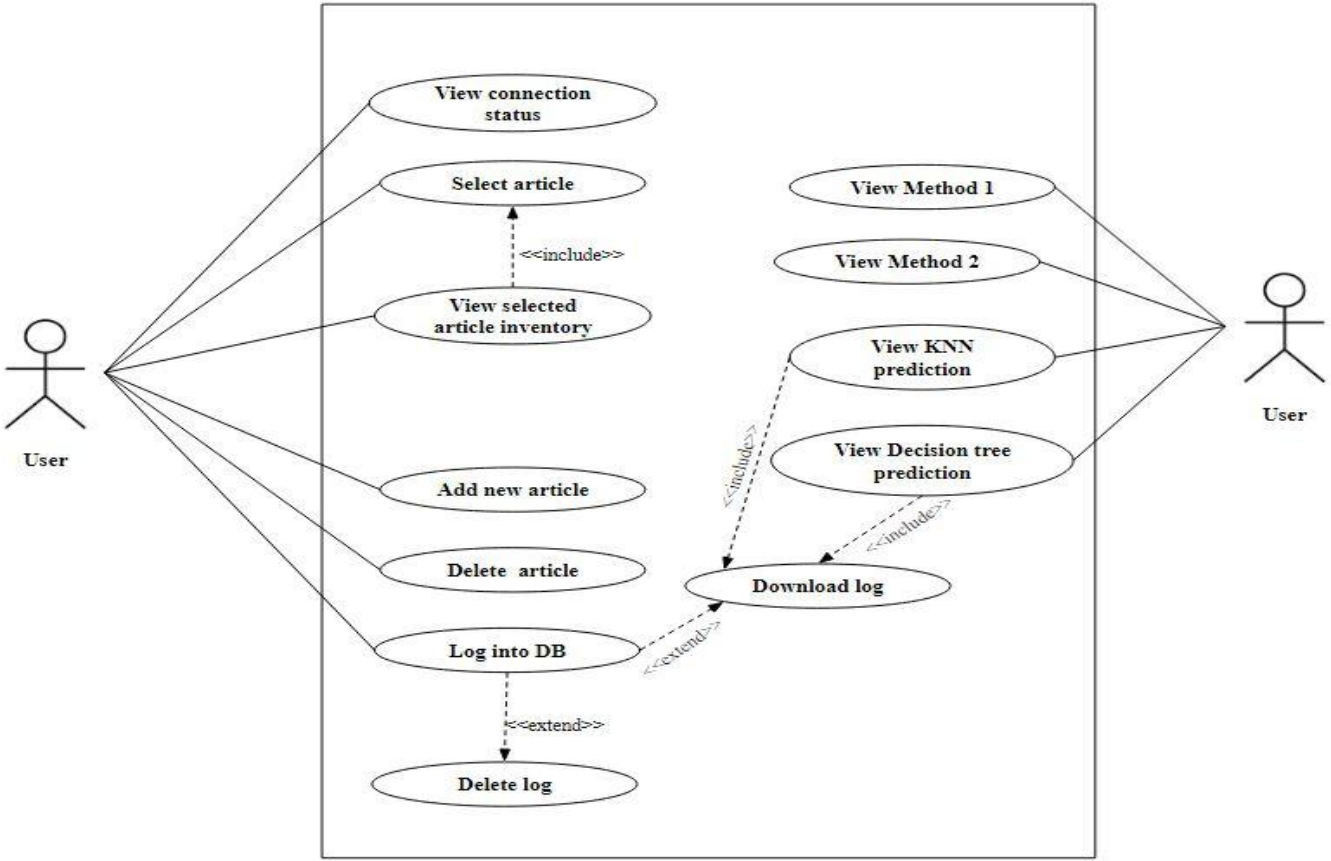
Architecture - Communication flow



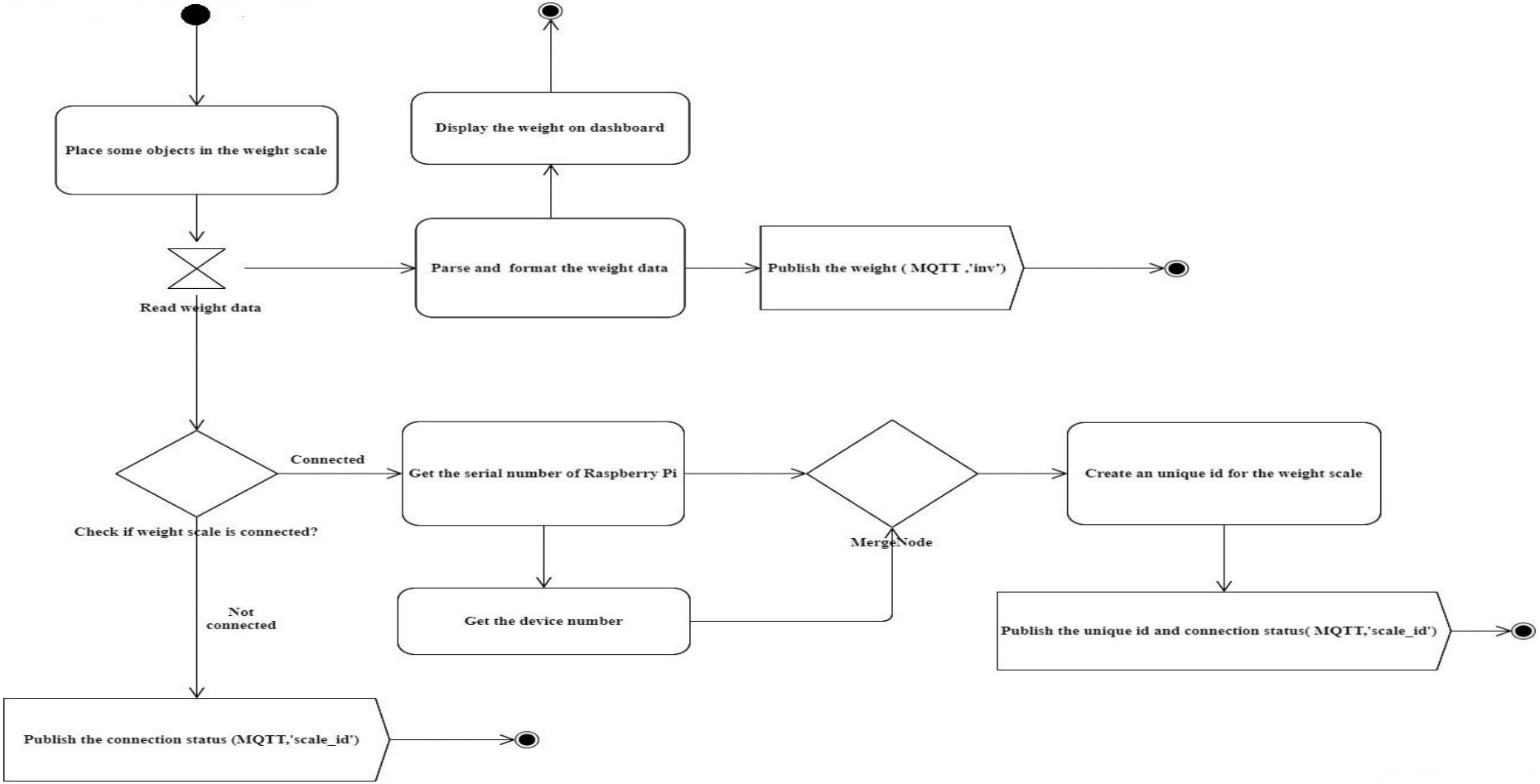
Design - Use Case diagram 1



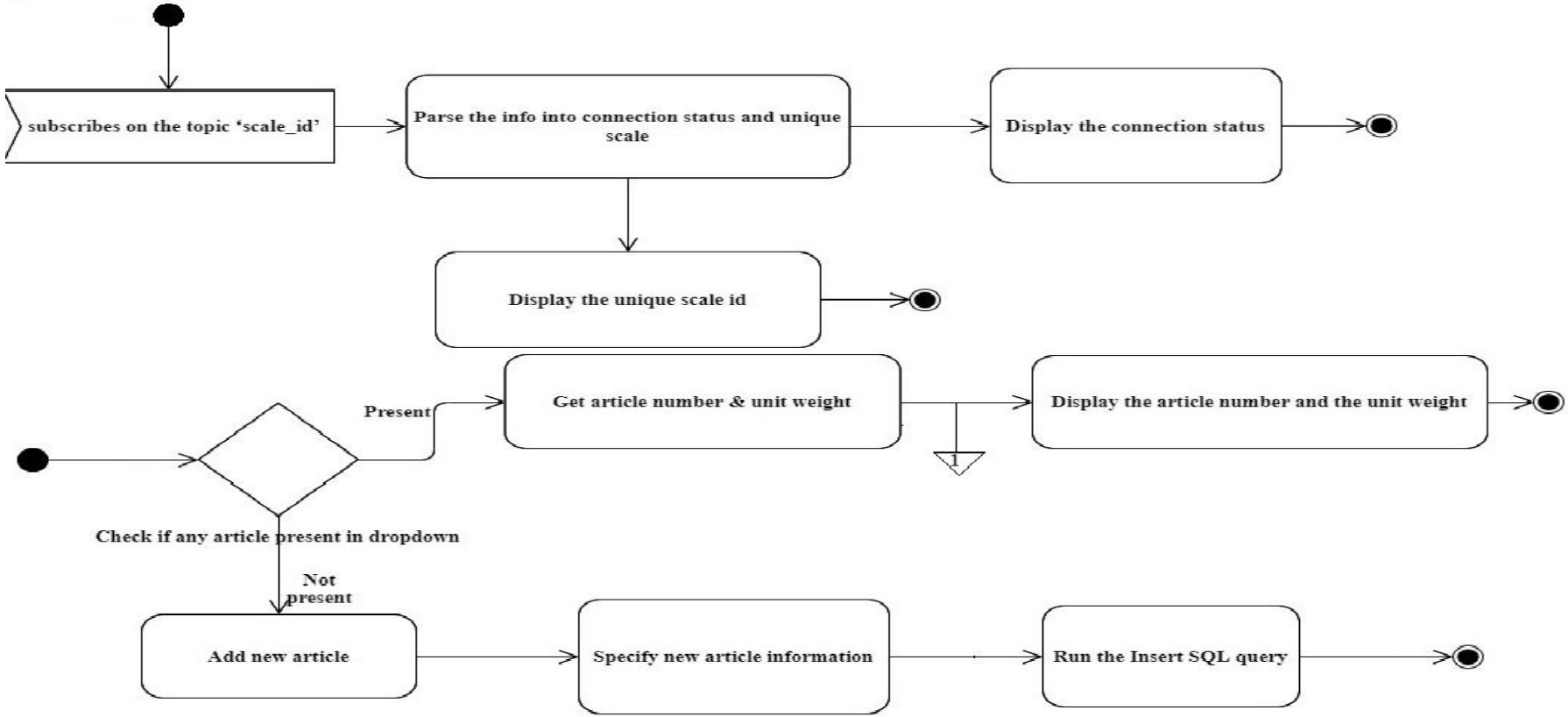
Design - Use Case diagram 2



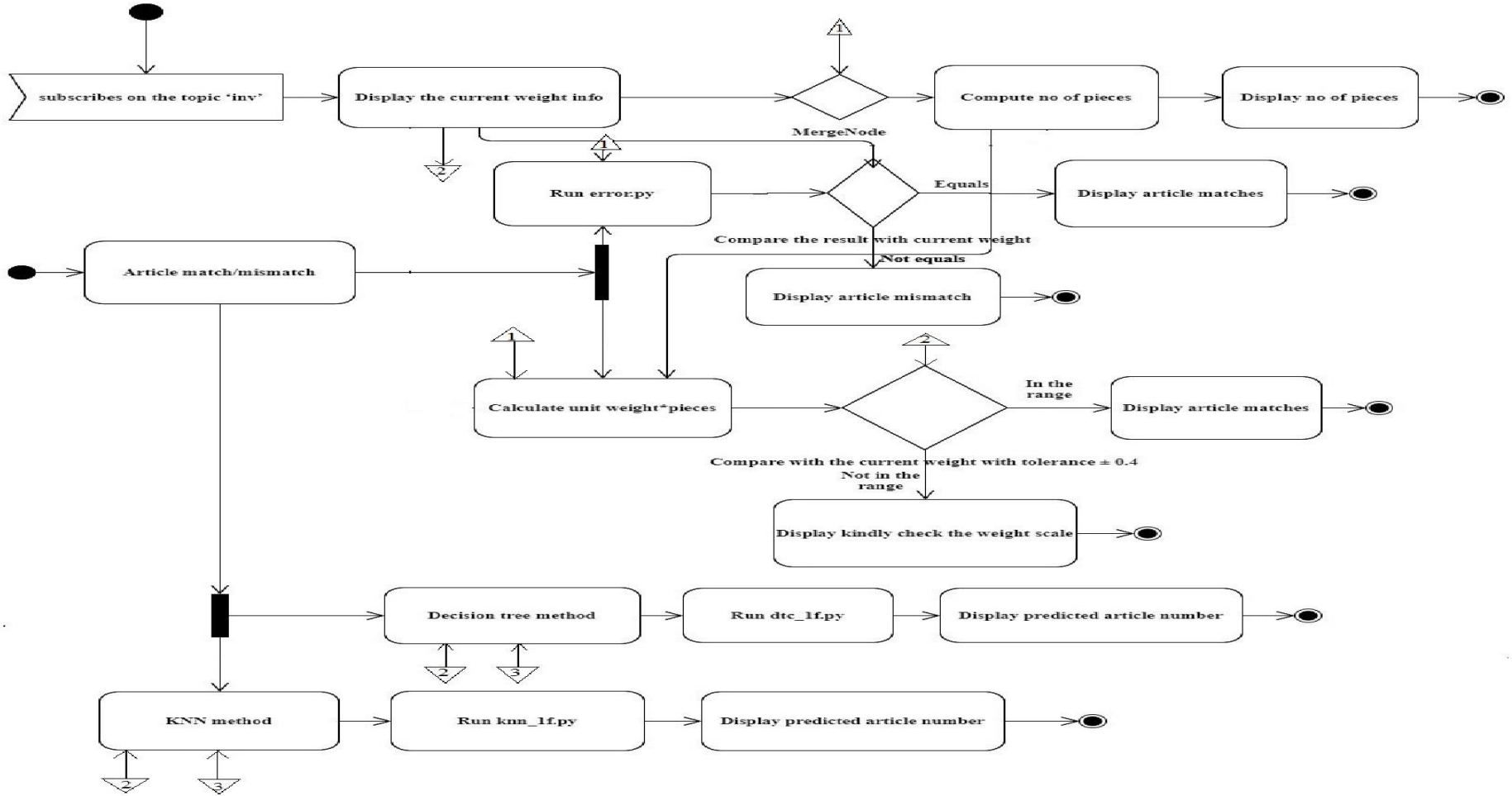
Design - Activity diagram 1



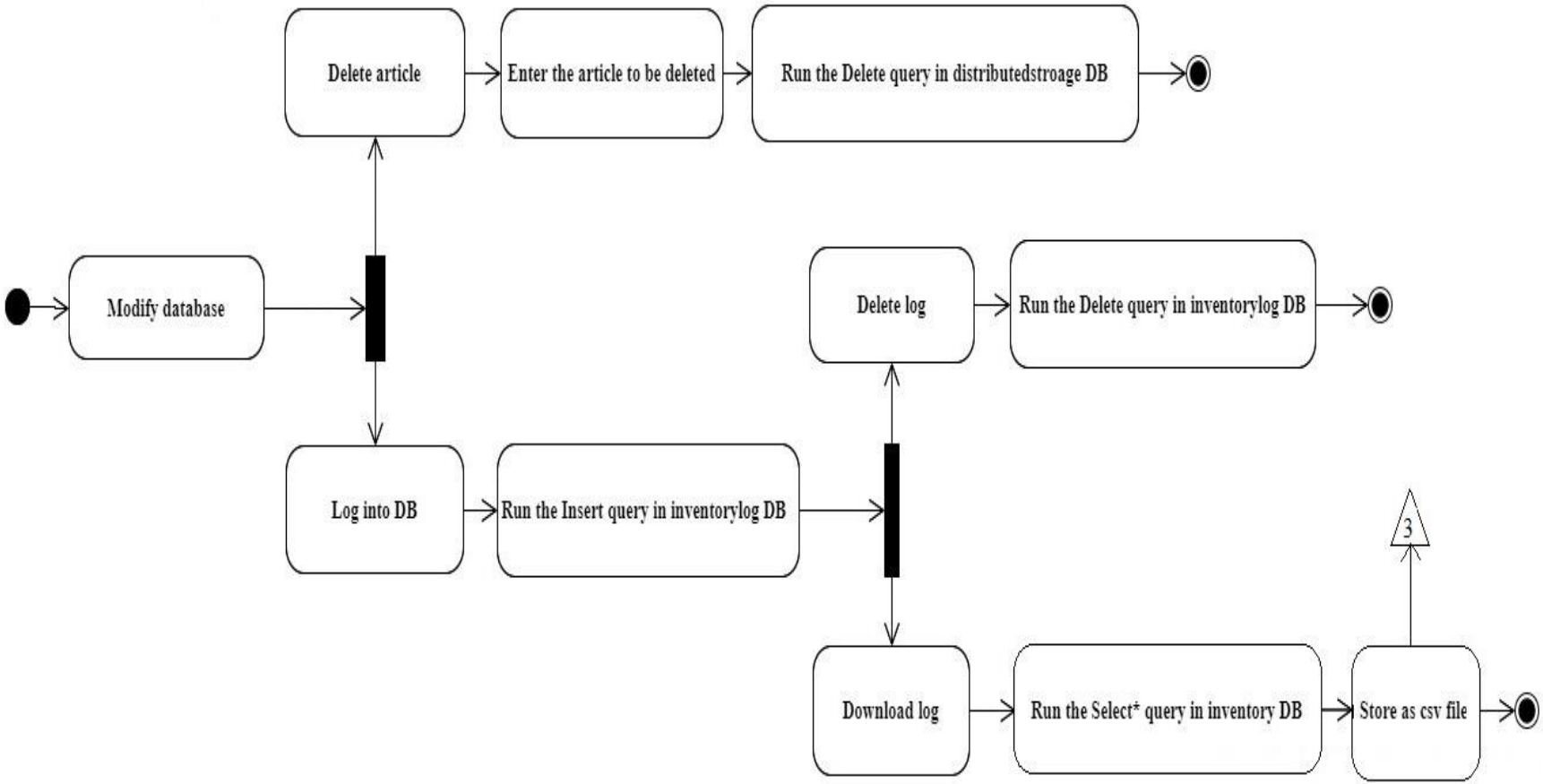
Design - Activity diagram 2.1



Design - Activity diagram 2.2



Design - Activity diagram 2.3



Implementation - MySQL databases

id	article_number	article_name	unit_weight
1	731308	Waschenklammer	4g
2	708763	Tintenpatrone	1.4g
3	700555	Screws	3g

Database: distributedstorage
table ‘inventorydb’

```
CREATE DATABASE `distributedstorage`;

CREATE TABLE `distributedstorage`.`inventorydb` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `article_number` INT NOT NULL,
  `article_name` VARCHAR(45) NOT NULL,
  `unit_weight` FLOAT NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE` (`id` ASC) VISIBLE,
  UNIQUE INDEX `article_number_UNIQUE` (`article_number` ASC) VISIBLE,
  UNIQUE INDEX `article_name_UNIQUE` (`article_name` ASC) VISIBLE);

INSERT INTO `distributedstorage`.`inventorydb` (`id`,`article_number`,`article_name`,`unit_weight`)
VALUES ('1','731308','wascheklammer', 4);
INSERT INTO `distributedstorage`.`inventorydb` (`id`,`article_number`,`article_name`,`unit_weight`)
VALUES ('2','708763','tintenpatrone', 1.4);
INSERT INTO `distributedstorage`.`inventorydb` (`id`,`article_number`,`article_name`,`unit_weight`)
VALUES ('3','700555','screw', 3);
```

id	article_number	article_name	unit_weight
1	731308	wascheklammer	4
2	708763	tintenpatrone	1.4
3	700555	screw	3

- to store the master data of warehouse components
- can also add new article details through the dashboard.

Implementation - MySQL databases

time	scale_id	article_number	unit_weight	obtained_weight	count

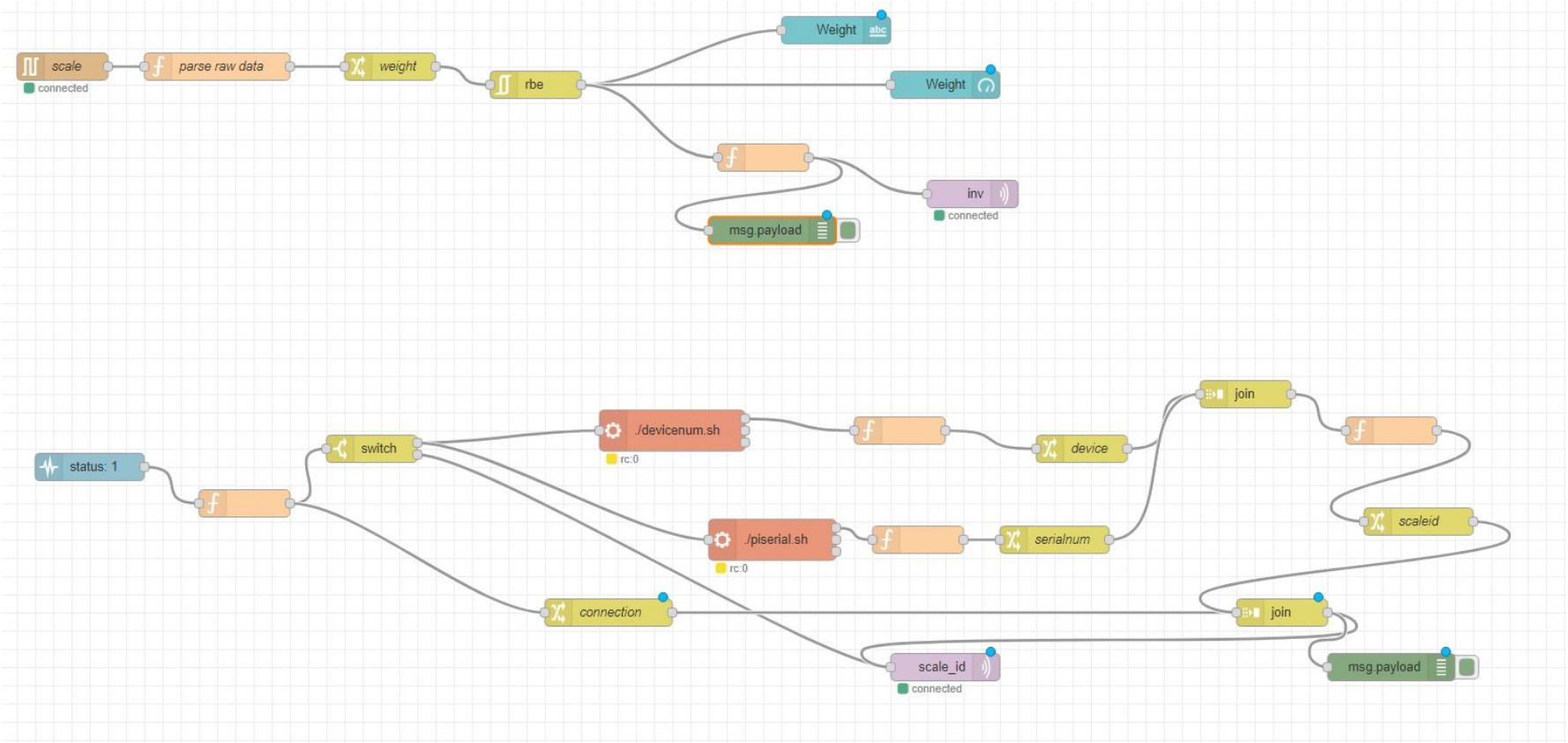
```
CREATE DATABASE `inventorylog`;

CREATE TABLE `inventorylog`. `datalog` (
  `time` varchar(45) NOT NULL,
  `scale_id` varchar(45) NOT NULL,
  `article_number` int(11) NOT NULL,
  `unit_weight` float NOT NULL,
  `obtained_weight` float DEFAULT NULL,
  `count` int(11) DEFAULT NULL
) ;
```

time	scale_id	article_number	unit_weight	obtained_weight	count
30-11-2019 23:27:58	00000000e718436c-007	731308	4	0	0
30-11-2019 23:28:03	00000000e718436c-007	731308	4	4	1
30-11-2019 23:28:07	00000000e718436c-007	731308	4	8	2
30-11-2019 23:28:11	00000000e718436c-007	731308	4	12	3
30-11-2019 23:28:15	00000000e718436c-007	731308	4	16.2	4
30-11-2019 23:28:26	00000000e718436c-007	731308	4	20.2	5
30-11-2019 23:28:30	00000000e718436c-007	731308	4	24.2	6
30-11-2019 23:28:34	00000000e718436c-007	731308	4	28.2	7
30-11-2019 23:28:39	00000000e718436c-007	731308	4	32.2	8
30-11-2019 23:28:50	00000000e718436c-007	731308	4	36.2	9
30-11-2019 23:28:55	00000000e718436c-007	731308	4	40.2	10

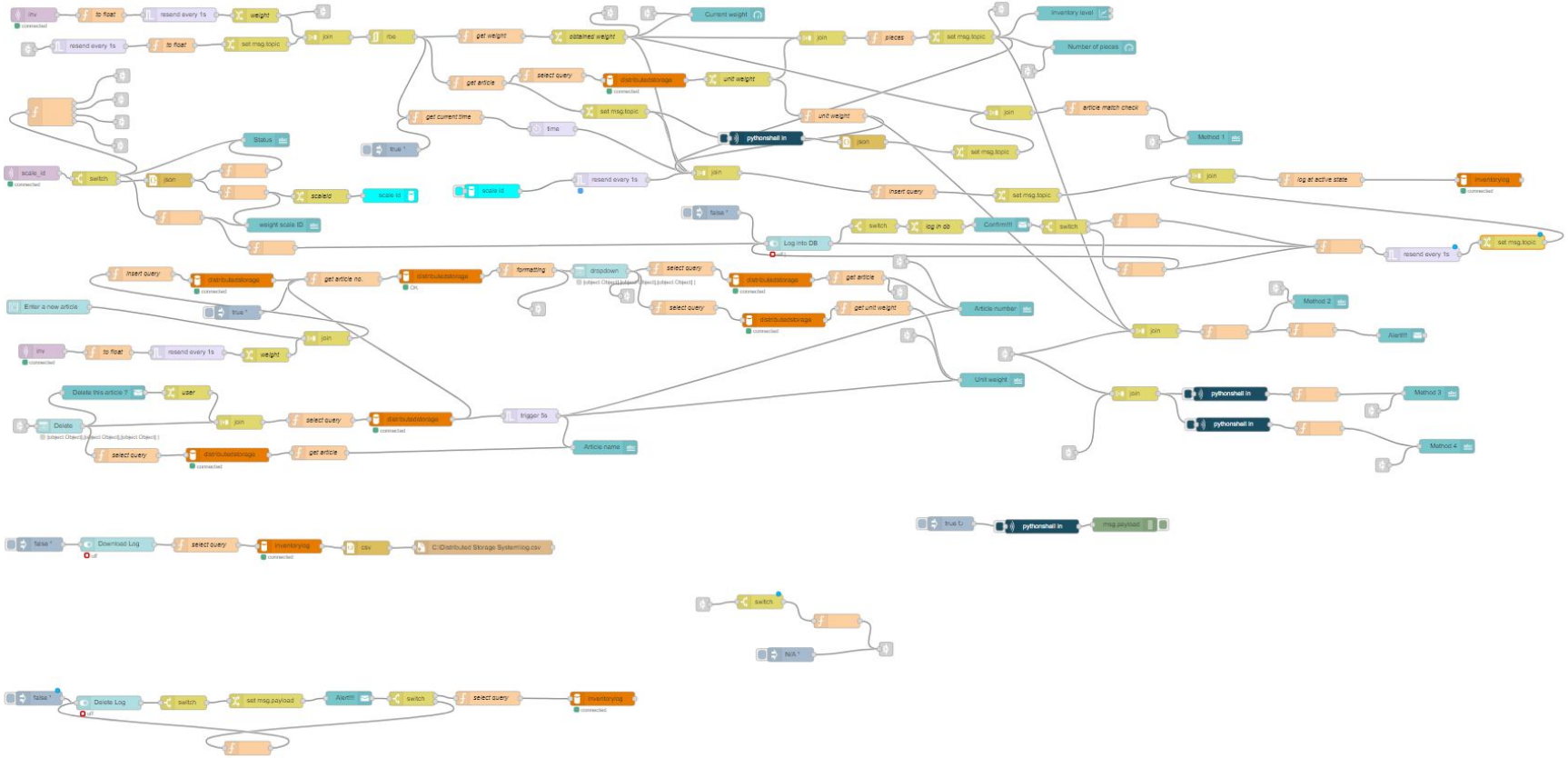
Database: inventorylog
table: datalog

- to store the timestamped log information of all the warehouse articles being kept on the weight scale on user demand - transactional data.
- can also download, delete the log information on user demand.

Implementation - Node-RED flow at Raspberry Pi



Implementation - Node-RED flow at local machine



Testing - Error detection techniques

- Method I
- Method II
- K Nearest Neighbour
- Decision Tree classifier

Testing - Error detection techniques - Method I & II

Method 1

- checks if the obtained weight detected in real time from the weight scale lies within the permissible set of estimated weight values for that specific article.

- Gets the unit weight and computes the permissible range of weight values based on unit weight
  - If unit weight of a article = x g
  - Obtained weight values = multiples of x ± t. (t is tolerance)
  - Permissible range = [x ± (t-0), x ± (t-1), x ± (t-2), x ± (t-….t), x ± (t-t), 2x ± t]

Method II

- takes into account the obtained weight (ow) in real time from the Raspberry Pi, the number of pieces information (p) and the unit weight (uw) of the selected article.
- ow == (p * uw) ± t
  - If Equation does not satisfy, then Article mismatch
  - If Equation satisfies, then Article match.

Testing - Error detection techniques - KNN & Decision tree classifier 1

- Both the models predicts the article number when an article is placed on the weight scale.
- log.csv as the source file used for training.

time	scale_id	article_number	unit_weight	obtained_weight	count

- The features like time, scale_id, unit_weight, count dropped being irrelevant.
- The feature ‘obtained_weight’ as the only input to the models.
- The feature ‘article_number’ as the class label.

obtained_weight	article_number

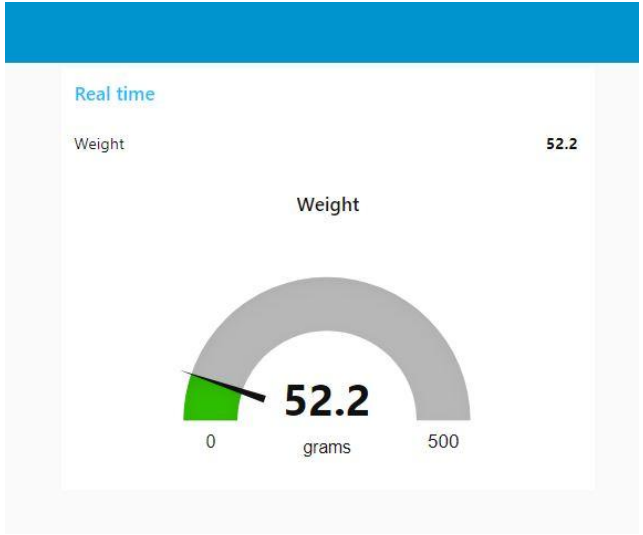
- 75 percent of this data for training and 25 percent for testing both the models.
- Training of the models scheduled weekly (every 168 hours).
- The trained models predicts the article number whenever the obtained weight parameter is passed as the input to the model ie, whenever an article is placed on the weight scale.

Deployment - weight scale reading

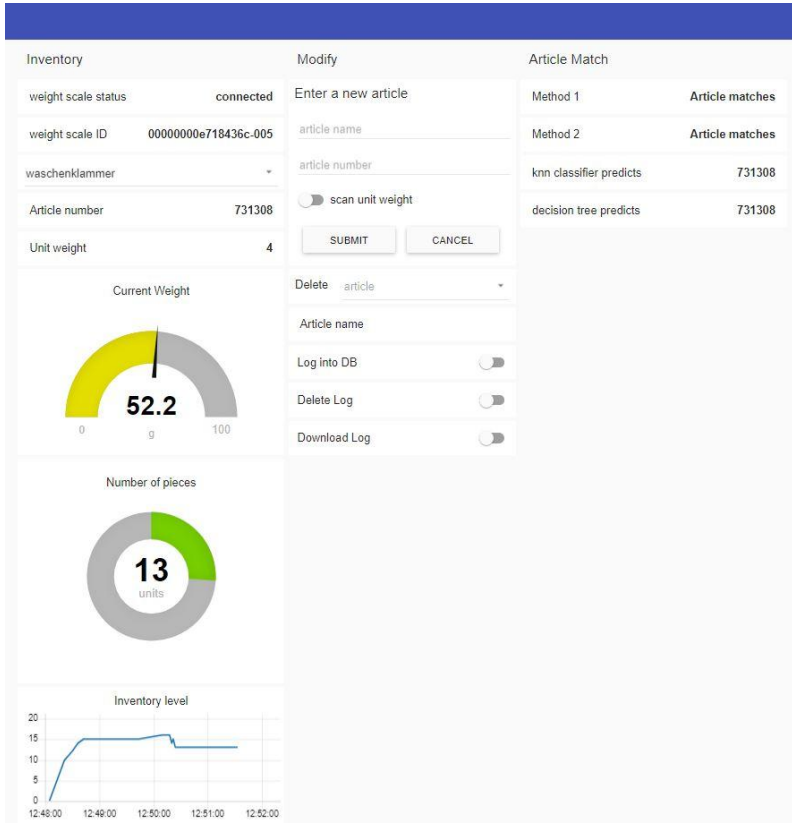
- Article - waschenklammer placed on the weight scale
  - 13 pieces
  - unit weight: 4g
  - Article number: 731308

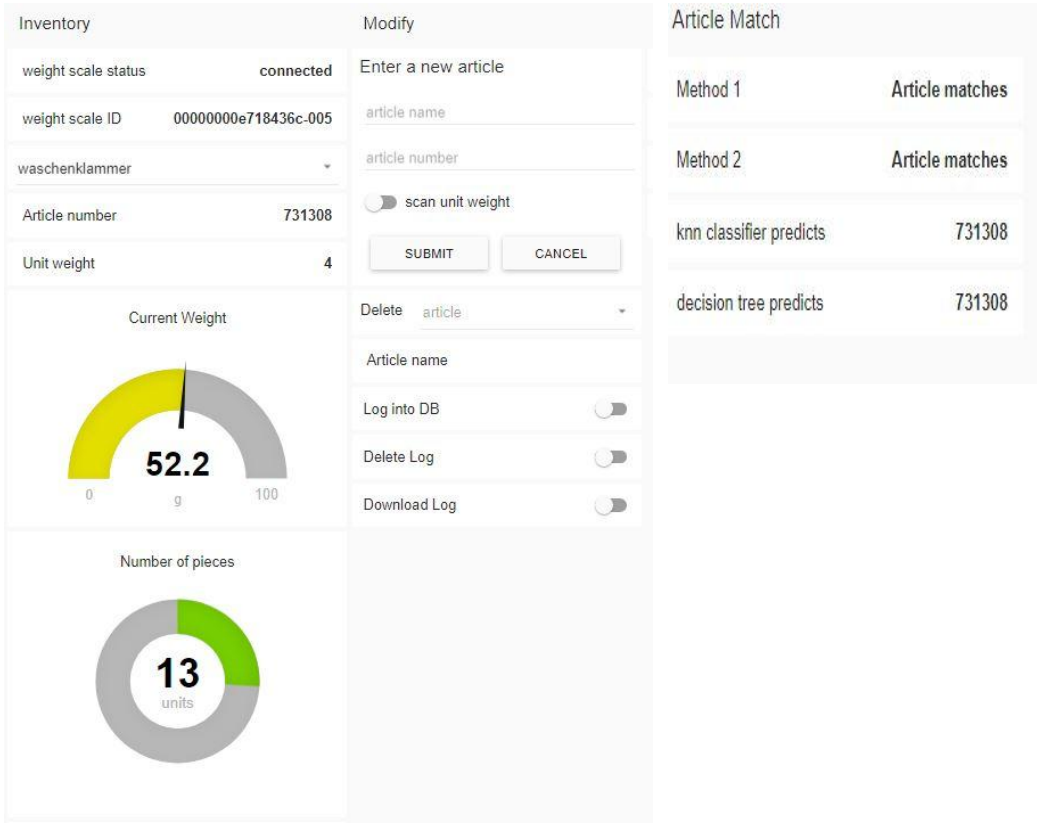
- Weight scale showing ‘52.2g’

Deployment - Node-RED dashboard at Raspberry Pi



Deployment - Node-RED dashboard at local machine





Prospects - Scalability approaches I

1 weight scale - 1 raspberry Pi

- weight scale + raspberry pi can be identified using scale id.

- for each new pi, a new duplicate flow must be deployed at both pi and client end.
  - for each new flow, the MQTT node must be specified with it’s respective ip address.
  - Each flow has it’s own dashboard.

- n weight scales > n raspberry pi > n flows at pi end > n flows at client end> n client dashboards > 1 distributed storage system

Prospects - Scalability approaches II

4 weight scale - 1 raspberry Pi

- For each weight scales connected to pi, pi randomly allocates device numbers to it like **e718436c-005, e718436c-006, e718436c-00X.**
- Can identify from which pi, the readings are being received but cannot identify from which connected weight scales, it is being sent.
- Since a pi can have four weight scales connected to it, a single pi will have four node red flows for each of the weight scale
  - for each flow, the “serial in” node must be specified with corresponding serial port.
  - for all the four flows, the “MQTT” node should be having the same ip address.
  - A single flow at the client end for one pi with four weight scales sending weight readings.
  - One single dashboard at the client end.
- For every new pi, the third step is to be repeated.
- 4n weight scales > n raspberry pi > 4 flows at each (1..n) pi > n flows at client end > n dashboards > 1 distributed storage system.

Note - Since there is only one dashboard at the client end for one raspberry pi which is connected to four weight scales, it may lead to synchronisation issues.

Thank you for your attention