C:\Users\mannan\AppData\Roaming\Skype\abdulmannan8989\media_messaging\media_cache_v2\^6E7543C0B0B395011F4681B5D18878523F6BF10E15C191B788^pimgpsh_fullsize_distr.jpg

**MEI-2310**

**INTRODUCTION TO INDUSTRIAL INFORMATICS**

**DATED: 8-12-2015**

**GROUP # 8**

**Shahbaz Ali 256327**

**Abdul Mannan 256344**

**Khurshid Ali 256353**

**Introduction:**

In this assignment we have to design client-server system based on automation system which replicate the functionality of the Factory simulator, and work in conjunction with the simulator. The main task is to show the effective communication between them.

**Information Integration:**

The integration of the information is done by using different techniques, tools and models:

* **Techniques:**

1. UML
2. Node JS
3. HTML
4. JQuery
5. Gantt Chart

* **Tools:**

1. Project Libre.
2. Visual Paradigm
3. NotePad++

* **Model:**

To design a system that replicate the functionality of the Factory simulator and work

in conjunction with the simulator.

**Unified Process Life Cycle Approach:**

The project is developed using Unified Process life cycle approach. As Unified Process life cycle includes the four phases which consists of iteration. Iteration is division of main projects to mini projects. The four main phases of Unified Process life Cycle is given as:

* Inception phase is the smallest phase of the project in which scope and vision of the system is defined.
* Elaboration defines the main architecture and core resources needed to complete the project.
* Construction consists of development and implementation of the project.
* Transition consists of making the system operational.

This project is divided into five iterations in which different set of tasks to be completed which are as follow:

**First Iteration:**

In first iteration the tasks to be completed are as follow:

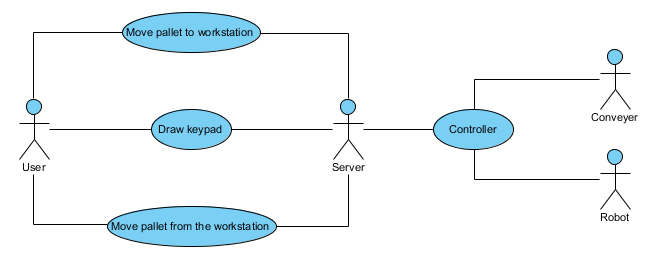
1. **System Vision:** The purpose of this system is to control the FASTory Assembly line using the web services. The task is to draw keypad of the mobile. The assembly consists of different work stations connected through conveyor line. It works stage wise, in which every stage is dependent on the task completion of the previous stage.

Our main task is divided into two parts:

* To create server. Backend programing (communication between server and controller of the assembly line).
* Front end application (by which client hits the server for any task).

1. **Use Case Diagram:**

Use case diagram is defined as the graphical representation of the interactions among the different elements of the system. Here in this Use case diagram is made by using the Visual Paradigm. The Use case diagram for the system is given below:

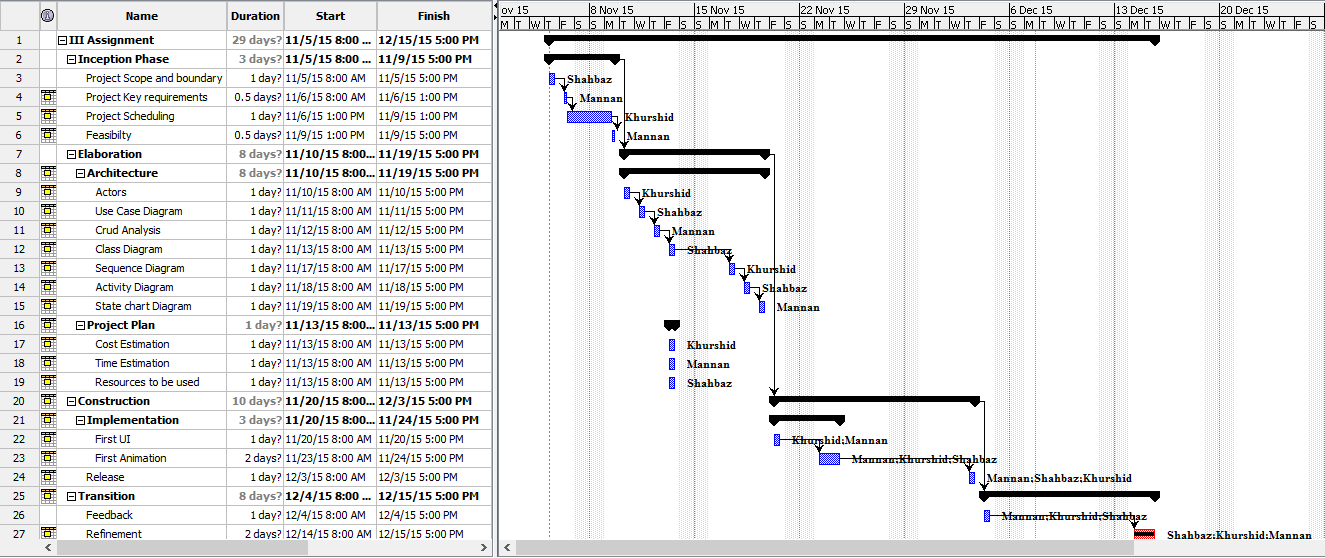
****

**Fig 1. Use Case Diagram of the System**

User/Clients are the actors in the system which order to achieve the specific goal. Here in this case User/Clients are named as User, Server, Conveyor and Robot. Use cases are the activities which are carried out by the system. Here in this case use cases are move pallet to work station, Draw keypad and Move pallet from the workstation.

1. **Work Break-Down Structure (WBS):**

The work break down structure of the system is made by using the Project Libre tool. The work break down structure describes the division of project into different task and how much time it requires to complete the specific task. Each individual of the group has assigned with specific task to be completed which is given as:



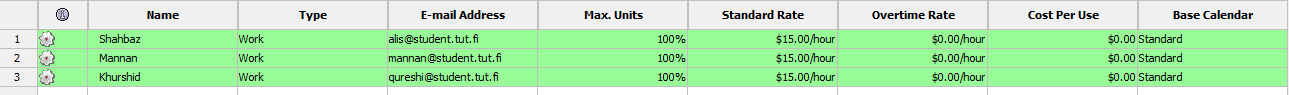
**Fig 2. Work Breakdown Structure**

1. **Benefits/Challenges and Cost:**

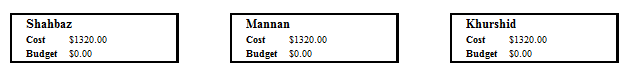
The challenges in completing the task is that we need to have visual contact with the factory line while deciding which API to hit in which order to perform the required task. This is very time consuming and can be managed if the FASTory line is accessible to use 24/7. Other than that a big challenge will be making a good enough interface for the user to get an idea of what is happening at the assembly line because the main purpose of this project is the make this line accessible to use from anywhere.

The main advantage of implementing this system is mobilizing the control of the FASTory assembly. If such an interface exist you can work on this assembly from anywhere in the world from where you can excess the system online.

The cost analysis of making this project can be very less compared to the improved usability of the system hence making it cost efficient. The cost of the project is calculated with the help of Project Libre by assigning hourly wage to the individual developer which is given as:



**Fig 3. Standard rate per hour**

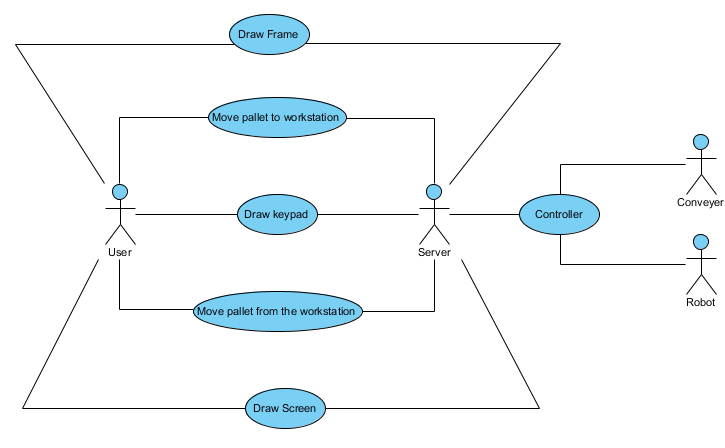


**Fig 4. Total Cost of the Project**

**Second Iteration:**

1. **Refinement of Previous Diagram:**

In the first iteration the use case diagram was made. In this iteration that Use Case Diagram refined which is given as:

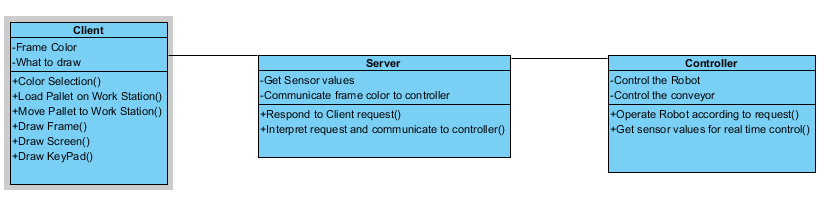


**Fig 5. Refined Use Case Diagram of the System**

User/Clients are the actors in the system which order to achieve the specific goal. Here in this case User/Clients are named as User, Server, Conveyor and Robot. Use cases are the activities which are carried out by the system. Here in this case use cases are move pallet to work station, Draw frame, Draw keypad, Draw Screen and Move pallet from the workstation.

1. **Class Diagram:**

Class diagram explains the structure of the system by describing the system’s class, attributes, operations and relationship among them. The Class diagram of the system is modelled by using the tool Visual Paradigm. The Class diagram is given as:



**Fig 6. Class Diagram of the System**

Attributes of the client are frame color and what to draw and operation are color selection, load pallet on work station, move pallet to work station, draw frame, draw screen and draw keypad.

Attributes of server are get sensor values and communicate frame color to controller. While the operations are respond to Client request and interpret the request.

Attribute of controller are control the robot and control the conveyor while the operation are operate robot according to request and get sensor values.

1. **Crud Analysis:**

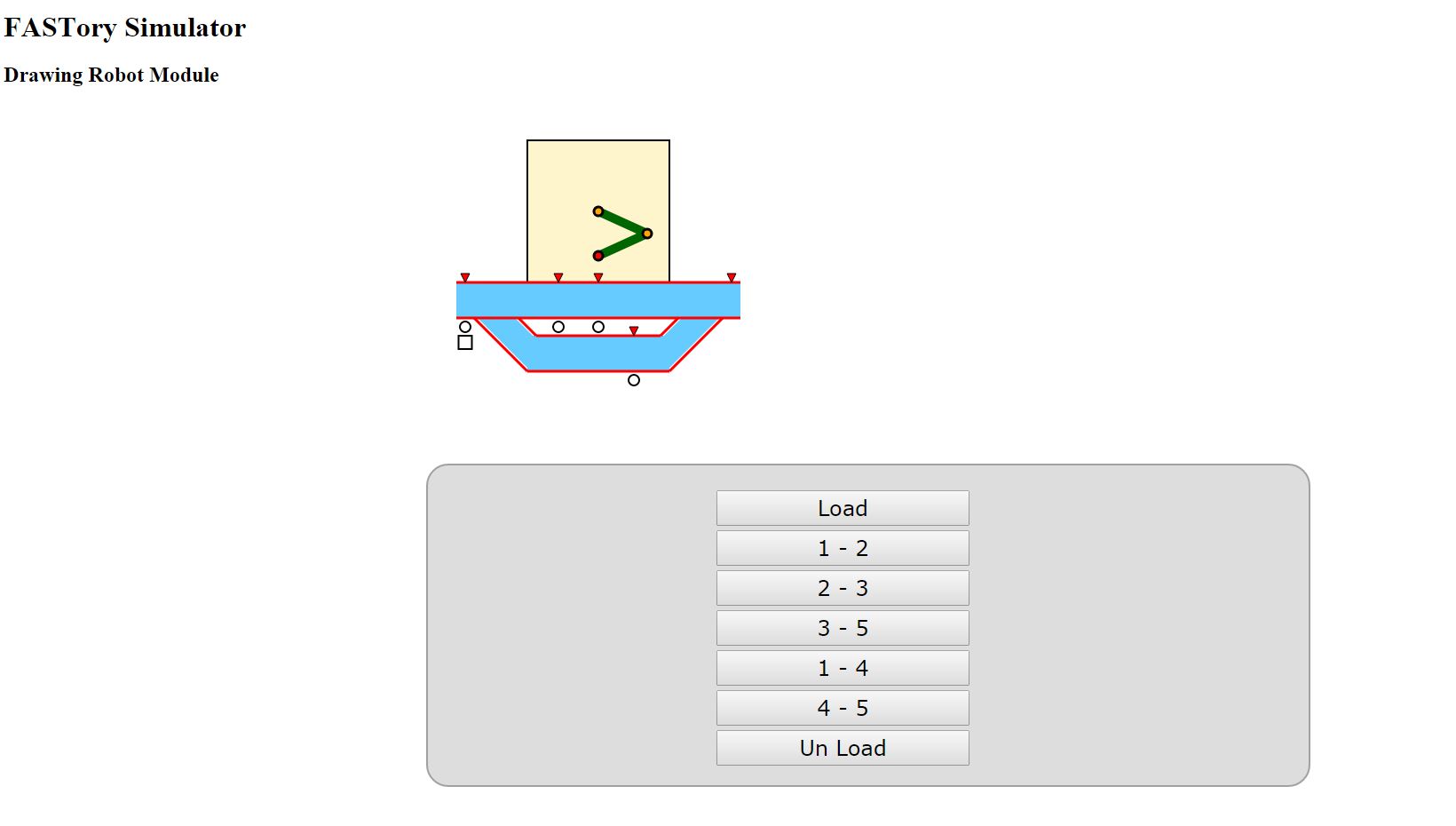
Crud Matrix (create, read, update, delete) is matrix which relates process and data or process and resources. When there is link exist between process and data or process and resources it shows whether the process perform create, update, read or delete operation on the data and the resources. The Crud analysis of the system is given below:

**Use Case location Matrix: show where use cases are performed.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Location** | | | |
| **Use Cases** | **Client** | **Server** | **Conveyor** | **Robot** |
| Move Pallet to workstation | ✔ | ✔ | ✔ |  |
| Move Pallet away from workstation | ✔ | ✔ | ✔ |  |
| Draw Frame | ✔ | ✔ |  | ✔ |
| Draw Screen | ✔ | ✔ |  | ✔ |
| Draw Keypad | ✔ | ✔ |  | ✔ |

**CRUD Analysis: Use Case Domain Class Matrix highlights access requirements**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Operation** | **Create** | **Read** | **Update** | **Delete** |
| 1 | Move Pallet to Workstation | Loaded |  |  |  |
| 2 | Colour Selection | Selected |  |  |  |
| 3 | Draw Frame | Drawn |  |  |  |
| 4 | Draw Screen | Drawn |  |  |  |
| 5 | Draw Keypad | Drawn |  |  |  |
| 6 | Respond to Client Request |  | Responded |  |  |
| 7 | Interpret and communicate to controller | Done |  |  |  |
| 8 | Operate Robot according to request |  |  | Done |  |
| 9 | Get Sensor values |  | Acquired |  |  |
| 10 | Move Pallet away from Workstation |  |  | Moved |  |

1. **First UI:**

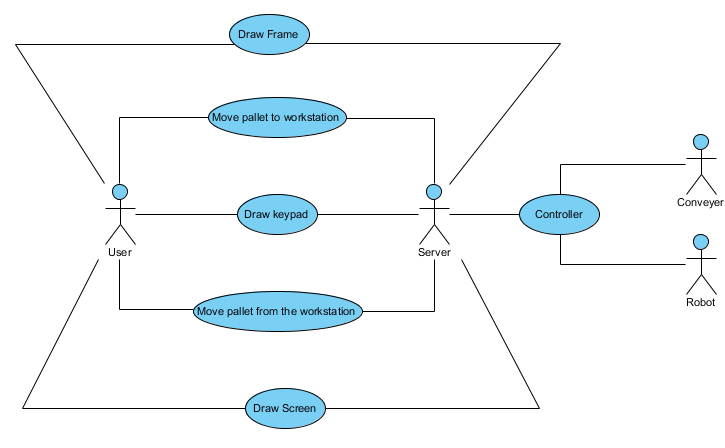
**Fig 7. First UI of the System**

To make first UI we opened the FASTory Simulator and used the inspect element feature to get the required HTML of work station 5. Then we followed the same procedure to take out the control panel of work station. Then we edited the HTML and its styling to suit our use.

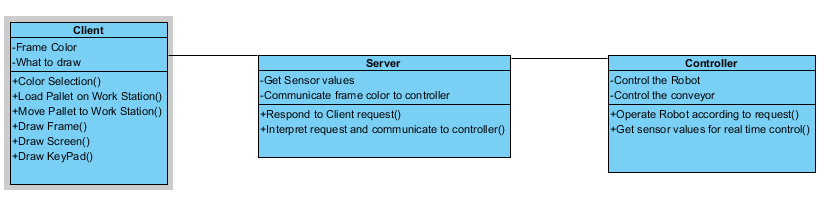
**Third Iteration:**

1. **Previous Diagram:**

There is no change made in the previous diagrams which were modelled during the second iteration which are use case and class diagrams.



**Fig 8. Use Case Diagram same as that of Second Iteration**

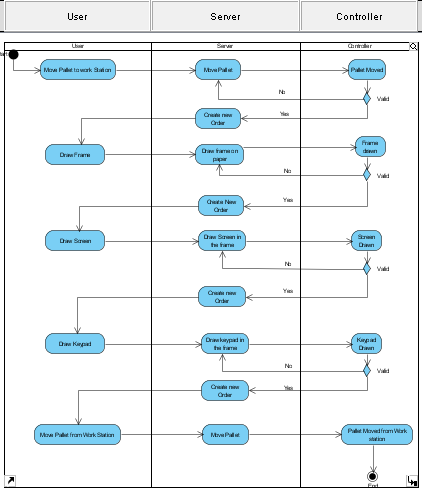
****

**Fig 9. Class Diagram same as that of Second Iteration**

1. **Activity, Sequence Diagrams and State Chart:**

**Activity Diagram:**

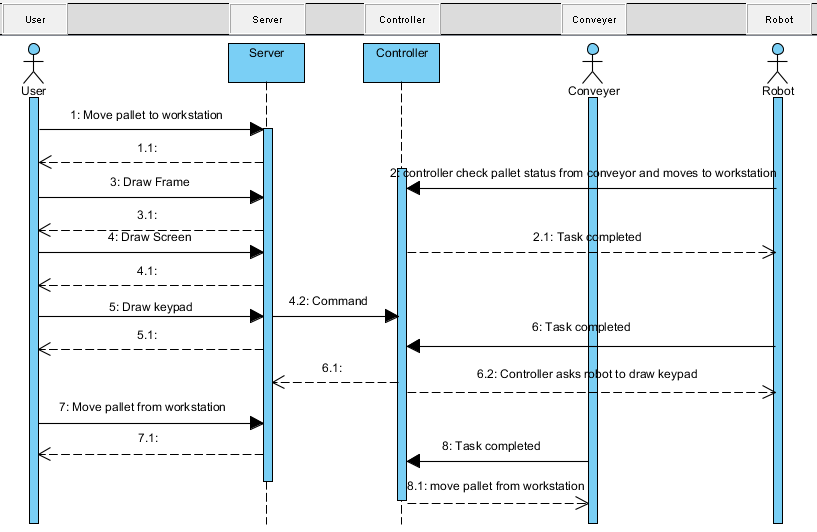
Activity Diagram explains the workflows of the process and flow of the activities for use case scenarios. It also form the basis of system sequence diagram. The activity diagram of the system is given as:



**Fig 10. Activity Diagram of the System**

**Sequence Diagram:**

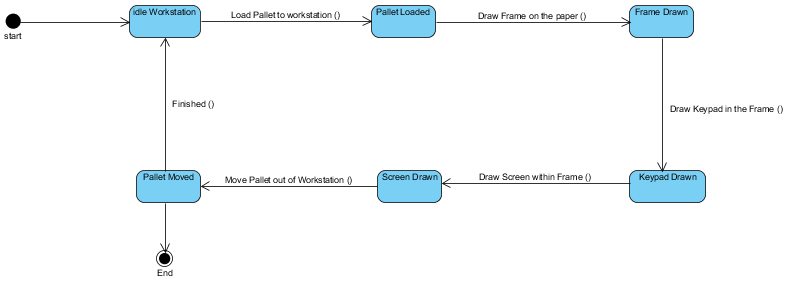
System Sequence diagram explains flow of information and identifies the relation between actors and system. It is message oriented and actor interacts with system through input output. It uses object notations. The sequence diagram of the system is given below.



**Fig 11. Sequence Diagram of the system**

**State Chart Diagram:**

A state in a state chart similar to status condition. It is developed for complex problem domain classes. It is composed of ovals representing status of object and arrows representing transitions.



**Fig 12. State Chart Diagram of the system**

1. **Web Server Implementation:**

For web server implementation we have used server.js file given by you. In this file we have integrated the HTML file made by us. After integration our HTML file can be accessed through server.

**Code:**

//Imporing express, creating express object

var express = require('express');

app = express();

//Defining a route from localhost:8080/ to ./client/III Assignment.html

app.get('/', function(req, res){

res.sendFile(\_\_dirname + '/III Assignment.html');

});

//Defining resource filter from /js/ to ./client/js/

app.use('/js', express.static(\_\_dirname +'/js'))

//Running server on port 8080. On started executin logging function.

//A typical example of async function call.

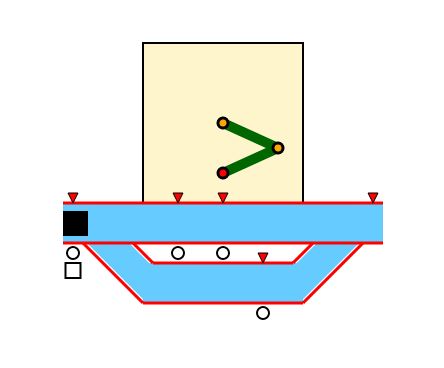
app.listen(8080, function(){

console.log('Started on port 8080...');

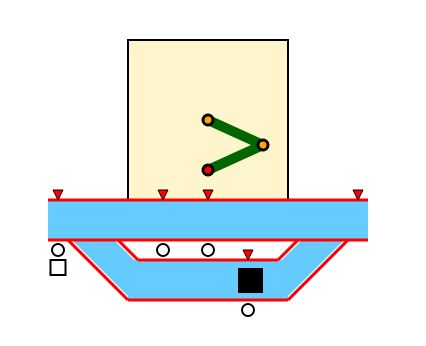
});

1. **First Animation**

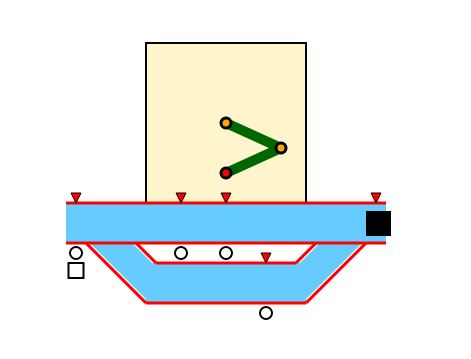
We have used d3.js provided by you and used its pallet animation on our HTML SVG. We have used the x-y positions of the pallet to animate it on our SVG,

****

**Fig 13. Pallet loaded at 1**

****

**Fig 14. Pallet moved from 1-4**

****

**Fig 15. Pallet moved from 4-5**

**Fourth Iteration:**

1. **Connection of Web Server with the Simulator:**

We used our own server to subscribe to the given events of the simulator. We have used our system’s IP as an argument of this subscription. Our system has to be on the same network as that of the simulator for it to respond to our server.

**Code:**

client.post("http://130.230.141.228:3000/RTU/CNV5/events/Z1\_Changed/notifs", args, function(data,response) {

console.log(JSON.parse(data));

});

client.post("http://130.230.141.228:3000/RTU/CNV5/events/Z2\_Changed/notifs", args, function(data,response) {

console.log(JSON.parse(data));

});

client.post("http://130.230.141.228:3000/RTU/CNV5/events/Z3\_Changed/notifs", args, function(data,response) {

console.log(JSON.parse(data));

});

client.post("http://130.230.141.228:3000/RTU/CNV5/events/Z4\_Changed/notifs", args, function(data,response) {

console.log(JSON.parse(data));

});

client.post("http://130.230.141.228:3000/RTU/CNV5/events/Z5\_Changed/notifs", args, function(data,response) {

console.log(JSON.parse(data));

});

1. **First Version of Monitoring Web Application:**

Initially we just logged our server whether it is connected to server or not. For this we have used below mentioned code:

Code:

io.on('connection', function(socket){

console.log('a user connected');

socket.on('disconnect', function(){

console.log('user disconnected');

});

});

**Fifth Iteration (Final)**

In this iteration we have used the event subscription calls of the simulator to ping our server from the simulator when the pallet appears on our subscribed workstation. We have used this feature to trigger our created animation to match with the server. In correspondence to that we have made direct API calls from our animation functions to the simulator service APIs so that if the pallet is moved on our created UI it also moves on the simulator. The overall result was that both systems were working simultaneously with each other. Which we have implemented successfully.

**The API call structure from our Client:**

function add\_pallet1(){

if(!loadPallet){

var svg = d3.select('svg'); // get root svg object to add pallets in it

pallet = svg.append('rect') // create a pallet and assign to the pallet

.attr("x" , config.positions[0].x) // NB! as you can see prev line does not have semicolon

.attr("y" , config.positions[0].y) // NB! and in current lines you are using .attr()

.attr("width" , config.pallet.size) // NB! which means that the operatiosn will be done on

.attr("height" , config.pallet.size); // NB! the same object.

loadPallet = true;

myPallets[0] = pallet;

x.open('POST', 'http://130.230.141.228:3000/RTU/CNV4/services/TransZone45', true);

x.send();

};

**};**

**Learning Outcomes and Conclusion**

This project built our understanding of the network communication happening between the online websites and their backend. We were able to create our own user interface by taking help from the simulators HTML code. Then we created a nodeJS server and used it to host that UI. In the final iteration we have used subscription feature of FASTory simulator to make external call to our server from the simulator.

With technical part of the assignment also we have learned how to use UML and create use case, activity and sequence diagrams using visual paradigm. We learned to create Gantt chart for project scheduling using Project Libre. Working on this project was a great experience and made us learn a lot of different valuable things.

**Problems Faced:**

The main problem faced was excessing the FASTory simulator. We had to be on the university network for FASTory to communicate back to our server.

Also the FASTory simulator has a lot of glitches. Sometimes it didn’t reset when we wanted it to you. The pallet stop responding sometimes on its own.

The whole scope of the project was very farfetched from out previous degrees so we had a lot of background learning to do on our own.