SPUG - Smart Purchasing and Guidance

Group 8: Meghna Suresh, Deepak Nadagouda, and Koushik Ragavendran

Service Computing Department, IAAS, University of Stuttgart
st169606@stud.uni-stuttgart.de, st170609@stud.uni-stuttgart.de,
st169478@stud.uni-stuttgart.de

Abstract. Commercial buildings have become an essential part of modern city culture. Embedding IoT technology in places such as malls upgrades user experience and offers to provide proficiency. SPUG is devoted to increase the comfort, hygiene and safety, and also the efficiency of the store. This system, placed in a smart commercial space aims at providing comfort to shoppers by implementing an automated trolley movement in the interior of a store, navigating on the shortest path computed to all the sections that a shopper needs to go based on the user defined shopping list from their very phones. The trolley communicates with the central server and hence can also access the "shopper-density" in a section, which facilitates adequately to promote social distancing in the current scenario. This feature addresses health concern of the occupants in the store. The trolley also reads the products with a smart tag reader and bills them, with the store database to support it. With the billing done so fast, shoppers are essentially relieved from straining themselves from standing in long queues, and avoid coming in contact with others for hygienic reasons.

Keywords: Smart Shopping · Smart Trolley · Navigation · Quick Billing · Social Distancing · IoT · Smart Commercial Building

1 System Introduction

The most common problem people face while shopping is to locate the section in which the product is, and to find the direction to that section, and after reaching there, and finding out that the product is sold out is frustrating. Another issue is waiting in a long queue to checkout even for a handful of items.

With the SPUG system, the above mentioned problems can be solved and it provides added benefits to the users and store owners. The customers can get to know the availability of items in advance and know the directions to the product sections. The smart cart, also called as SPUG, guides users to the product sections by taking a less crowded path. This prevents getting in contact with more people, as per the current restrictions due to the pandemic. The store owners can monitor the number of products available on shelf and refill them if they are empty.

The SPUG system provides added comfort to the users by making sure some of the social distancing norms are maintained.

2 System Analysis

The requirements to the SPUG system can be broadly classified into two domains:

2.1 Requirements as a user:

i. As a user, I want that the system should find directions to the desired product sections and I should get to know the price and availability of the products beforehand, so that I can shop faster and reduce the shopping time.
ii. As a user, I want the system to avoid the crowded sections of the super market and take an alternative less crowded path, so that I can maintain social distancing and avoid coming in contact with a lot of people.
iii. As a user, I want the system to move on its own without the user intervention, so that I can prevent the contact with cart and thus preventing germ contamination.
iv. As a user, I want to avoid the long waits in queues for the customers to bill the products by providing an alternative to bill the products with the mobile application, so that I can reduce the overall shopping time and come in contact with lesser people.
v. As a user, I want to monitor the products purchased during a specific timeline, so that I can keep track of the calorie content of the products purchased.

2.2 Requirements as a Supermarket Owner:

i. As a supermarket owner, I want to monitor the amount of products on shelf, so that I can refill it when the stocks are empty.
ii. As a supermarket owner, I want to monitor the crowd density in the product sections, so that I get to know the most bought products.
iii. As a supermarket owner, I want customers to shop faster, so that I can provide service to more customers within a certain time period.

3 System Architecture Design

As shown in **Fig. 1.**, the architecture of the SPUG system can be described in the four layers:

i. **Physical layer:** The physical layer constitutes of the sensor and the actuator devices. The sensors are used to input the environment data to the upper layer, the ubiquitous layer. The objective of using sensors in the SPUG system is
   – to collect data about the path - whether a node is reached,
   – to detect the presence of obstacles which are not other smart carts (like moving people around the store), and,
   – to read the items added to cart.

Actuators are used here to navigate through the store, leading to the desired product location, and to indicate that the destination has been reached. The actuators receive instructions from the ubiquitous layers and realise the navigation and guidance.

ii. **Ubiquitous layer:** The ubiquitous layer consists of the data processing sequence that converts the raw data from the sensors in the physical layer to context data required by the reasoning layer. There is a database stores static data about the different items and their locations, and also the dynamic data like the location of other SPUG carts, the number of items of each kind that is still available. This database is available for all the upper layers, and has key importance in providing data for decision making. The ubiquitous layer also has the execution sequence that drives the actuators, based on the decisions from the upper layers, selecting the type of actuator and the actuation that needs to be done.

iii. **Reasoning layer:** The reasoning layer consists of the logic controller and the AI planner. The reasoning layer is responsible for making informed decisions, based on the context received from the ubiquitous layer and the inputs from the user interface. It fetches information from the database and passes the initial state - current context of the SPUG and the goal state - destination that the SPUG has to reach. The AI planner develops a plan to be followed and realised by the actuation to realise the goal.

iv. **User layer:** The uppermost layer is the user layer, which is responsible to present data to users in a relatable fashion. This encompasses of user interfaces, for a store customer to connect to a SPUG system. The reasoning results fetch data from the database, provide the statistics of available products, and provide an option to customers whether they actually want to purchase the product. A system monitoring interface presents the current statistics of the store and enables an administrator or a store owner make right business decisions.

The component diagram of SPUG is shown in figure **Fig. 2.**

4 System Implementation

The GitHub link to the implementation for SPUG is as follows:
https://github.com/koushik1703/SmartPurchasingUndGuidance.git

The SPUG system is mainly divided into the following components:

i. Centralised server
ii. Mobile application
iii. Track for the Smart Trolley
iv. Smart Trolley - We will consider the Smart Trolley as SPUG for further references
v. AI planner

Each of the subsystem is explained in detail below:

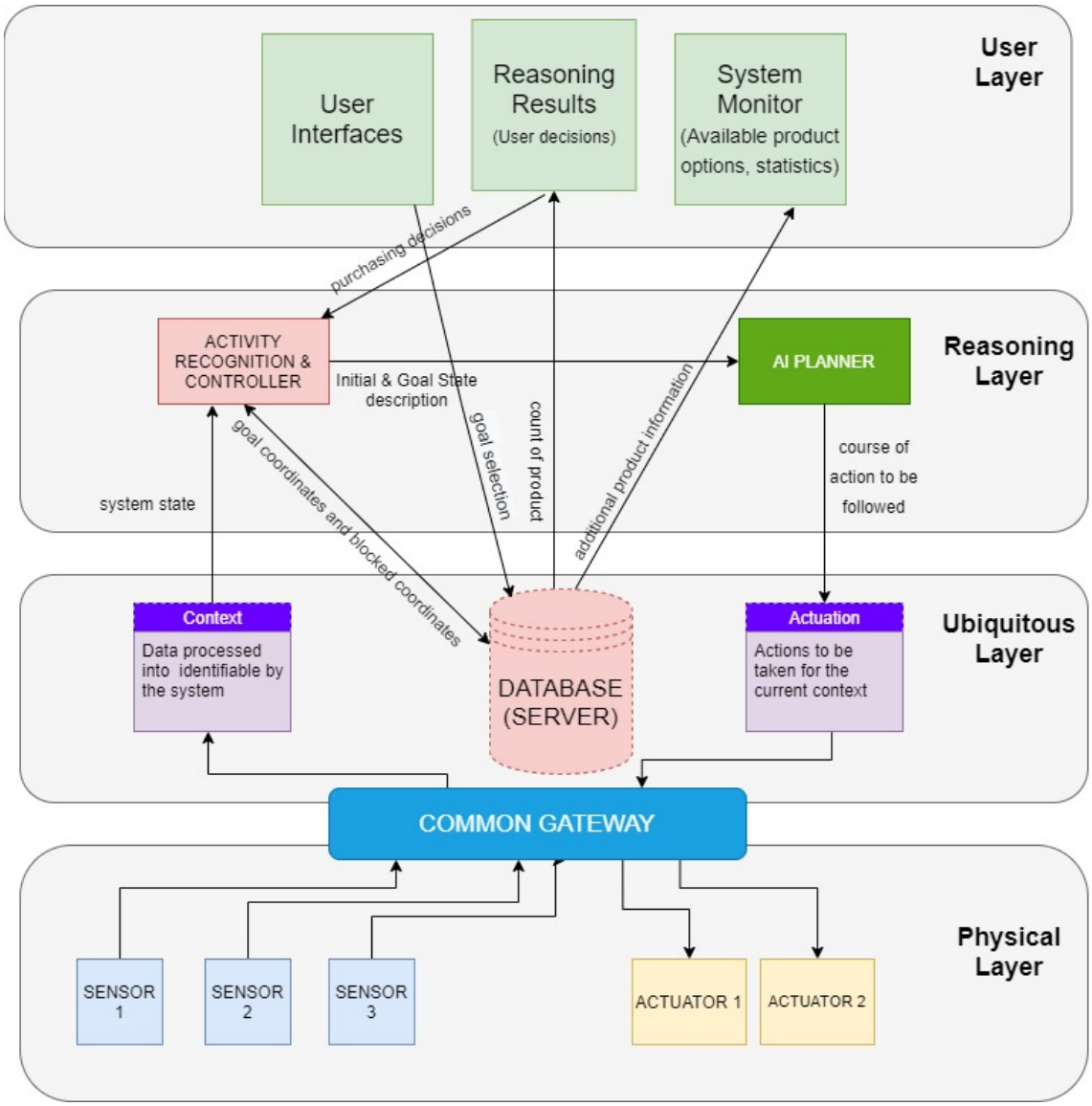

Fig. 1. Holistic System Architecture of SPUG

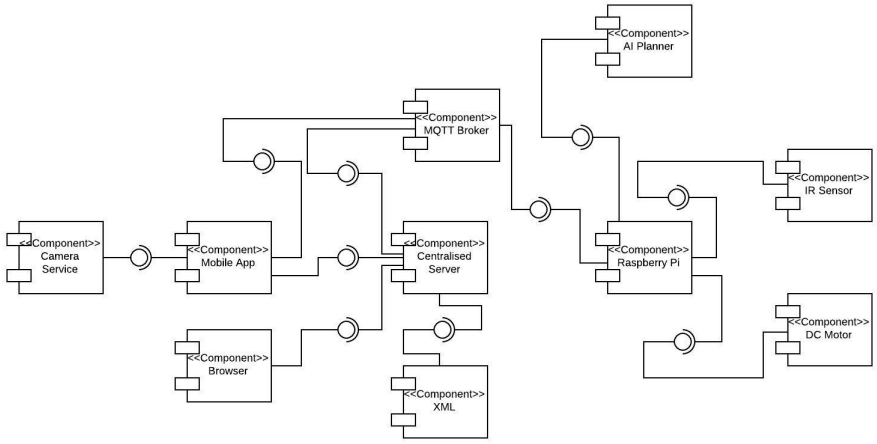

Fig. 2. Component Diagram of SPUG

4.1 Centralised Server

The central server is the main database for the SPUG system and is hosted in the store that uses SPUG system. The database is maintained in the form of an XML file which can be updated and retrieved by the server. The server uses HTTP and MQTT communication to communicate with android app and the SPUG.

The request for a cart from android device is in the form of a HTTP request. The server assigns the android device to a cart by updating the XML file. Each SPUG element has an attribute -android device ID and the device identity is added to it. If the requested SPUG has already been assigned, then it cannot be assigned to the device requesting it.

The count of each product in the store is stored in an element as count attribute in XML file. The android device can get the count of each product by making an HTTP request. The count of the product is queried in the XML file for the product requested and reply for the HTTP request contains the JSON object with the requested product's name and its count.

The coordinates of the product to be purchased is stored as attribute in XML file for each product element. The name of the product to buy is sent as JSON message through MQTT communication to the server. With the product name, the server queries the coordinate for the product requested and then these coordinates are sent to cart to reach to the product location.

As previously explained, the mapping for product and it's location coordinates is stored in XML file. The server can receive MQTT message from cart when the cart reaches the product location. The server queries the product name with the coordinates received from the XML file. Then it sends a JSON object with product name to device through MQTT message. Once the android device

updates that the product has been purchased to the server, the server sends a message to the cart that it can continue to the next coordinates through MQTT communication. If the product has been purchased, a message is received by the server from the android device and then the count of the product is updated in the XML file. If the product has not been purchased message is sent then count of product is not updated. In this way the current count of product is maintained.

To complete the session the server receives a HTTP request to "unassign" a cart from the android device. With the device identity the server queries the cart in XML file and marks it as unassigned. This is done so that the when another device requests the cart it can process for next session. In this way, it is ensured that a device cannot request a cart that has already been already assigned.

4.2 Mobile Application

Mobile application acts as an interface for the user to communicate with the SPUG system. Once a user enters the store, the user can request for a cart from the server with the cart number. The application receives response from the server whether it can be assigned or not. If the cart can be assigned to this device, the user is prompted to enter the items to be bought. If not, an error message is displayed in same view.

On successful assignment of cart to device, products to buy is to be entered. The user can enter a product and add it as shown in figure **Fig. 4.**. Once user has completed entering the wish list he can click done to enter next view where he can see the count of each product that he wishes to buy.

Once wish list is entered the android app sends a HTTP request to the server to get the count of products that is entered. On receiving response for the message, the product name and its count is displayed on the app. The user can prompt to next view to start purchasing.

When the go buy button is pressed, the next view is displayed as shown in figure **Fig. 5.** and list of product to buy is sent as message to server as JSON object through MQTT communication. On cart reaching a product location QR scanner view is displayed as shown in figure **Fig. 6.**. The user can scan the QR code that is labelled on the product.

If user confirms the purchase the item is added to the items bought view. The cost of the product is also listed in QR code and added to this view as shown in figure **Fig. 7.**. The user get update when cart reaches a product through MQTT communication. Thus, the user can scan product when location is reached and added to purchase list. On end of session the device HTTP request the server to unassign the cart from the device.

The product purchased and its details is stored in device locally in Object Oriented Model. The product details can be used to query the history on the device. The details can be filtered by date.

When the user clicks on show button, the list of products which is stored locally iterated and product purchased during this period is filtered with time

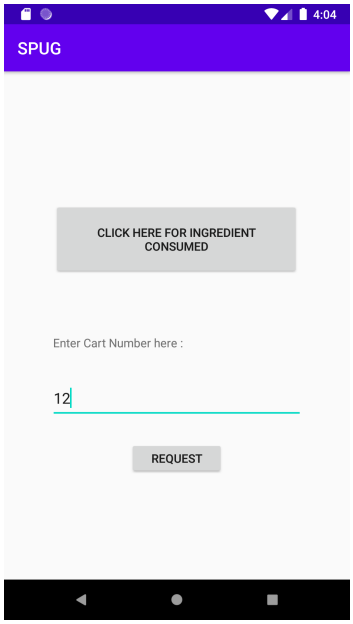

Fig. 3. Application view to request cart

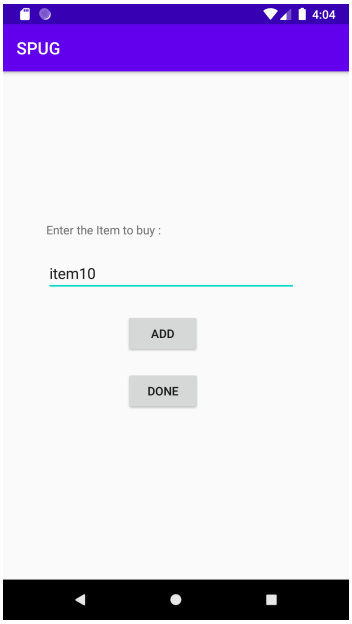

Fig. 4. Application view to enter the products

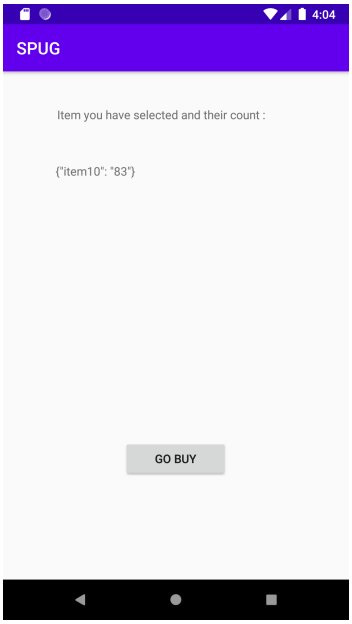

Fig. 5. Application to see the count of products

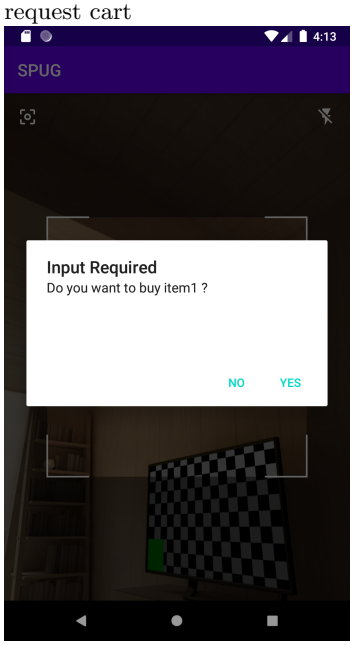

Fig. 6. Application view to scan the products

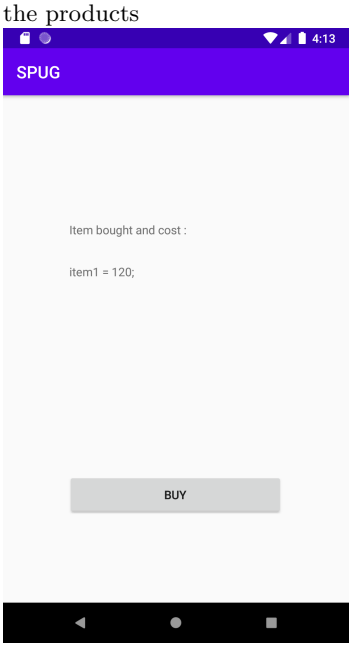

Fig. 7. Application view to see the final bill

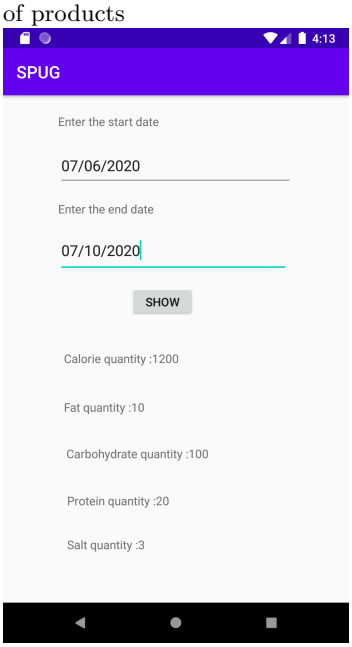

Fig. 8. View to see the list of products purchased

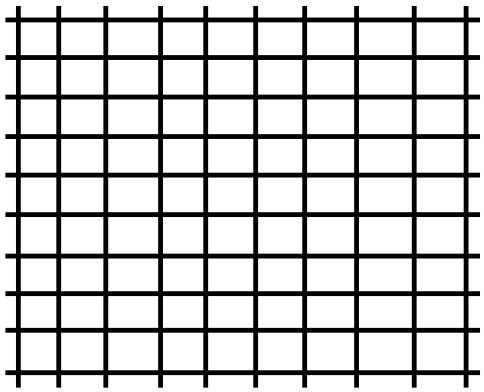

Fig. 9. SPUG Track

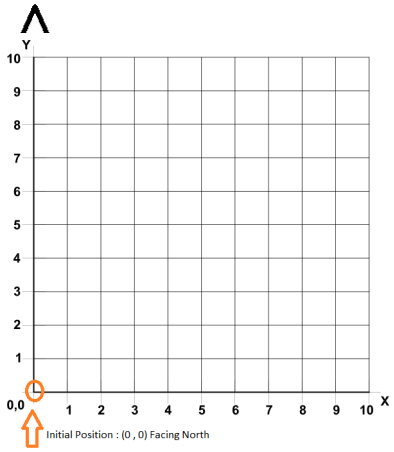

Fig. 10. Initial position of SPUG

detail stored on the product object and its details is summed up and displayed as seen in figure **Fig. 8.**.

4.3 Track

The super market hosting the SPUG system should have the track that allows the movement of the SPUG and helps it to calculate the coordinates locally. The track will be similar as shown in figure **Fig. 9.**. The following assumptions are considered:

i. The products are located at the nodes.
ii. At all the nodes SPUG's three IR Sensors detect the track, even at the boundaries of the track. We will consider this point as a junction.
iii. Directions are considered to be absolute.
iv. Initially SPUG starts at origin and is facing North as shown in figure **Fig. 10.**.
v. Any other obstacles such as humans can be present on track.
vi. At a given point some of the nodes may be occupied by other SPUGs'.

4.4 SPUG

SPUG is the cart that autonomously moves to the product location that the user desires to buy. It calculates the shortest path to all the desired products by avoiding the other SPUG present. SPUG consists of the following components:

i. Infrared sensors: It has 3 IR sensors in the front which are parallelly placed to each other to sense the track. The sensor cluster is perpendicular to the track as shown in figure **Fig. 11.**.

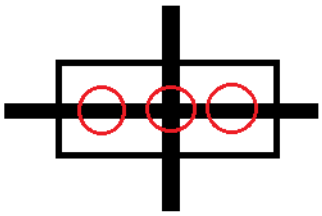

Fig. 11. IR Sensors Placement

ii. Ultrasonic sensor: It has a ultrasonic sensor in the front to check if there are any obstacles present on the path.
iii. DC Motors: It has four independently operated DC motors, which allows the cart to move in all the four directions.
iv. LED's: The LED's guide the user to move the cart according to the shortest path calculated by the AI Planner in case of Motor failure. LEDs' in the frot glow orange in case of an obstacle in the path.
v. Buzzer: Its used to indicate the use when the car reaches the product location.
vi. Raspberry PI: It is the main controller which collects the data from sensors and controls the actuators. It communicates with the server to receive the product locations and blocking points and sends data to server regarding the current position in the map and if the product location is reached. It runs the AI planner to determine the shortest path to the products and back to the origin.

Below is the detailed explanation of the sub-aspects of SPUG:

Movement of SPUG: SPUG receives the coordinates of the Products that the user has desired to buy. The product location coordinates are sent as JSON messages through MQTT Communication established between the Server and the specific SPUG. Once product locations are received it waits for the Blocking points. Blocking points are the locations which are occupied by the other SPUGs'. Blocking points are considered to be static in our scenario. The messages are sent in the same way as before.

For example, the product location points and the blocking points updated on a Cartesian coordinates map is shown in figure **Fig. 12.** for the following example:
Product Location Points: [(0, 2), (2, 4), (6, 2), (3, 1), (5, 5)]
Blocking Points: [(0 ,1), (1, 4), (4, 4), (4, 5)]

Afterwards the shortest path to reach all the product locations and back to the origin point is calculated. The shortest path is calculated according to the

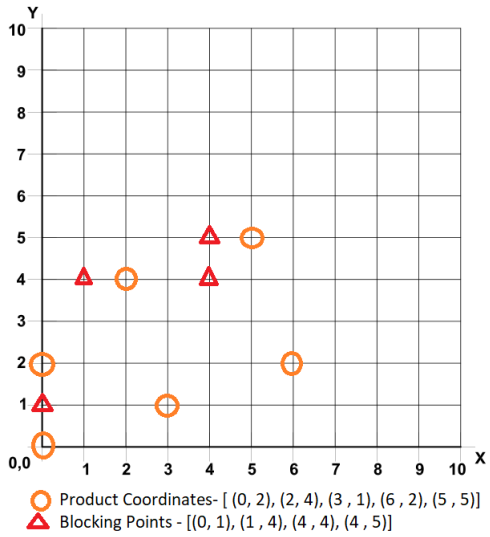

Fig. 12. Product location and Blocking Points

"Travelling Salesman Problem" by considering the **"Manhattan"** distance between the product location points. According to the travelling salesman problem a shortest possible route that visits each city and back to the origin city has to be found given a list of cities and the distances between each pair of cities.

The product location coordinates are ordered according to the shortest path calculated as show in figure 6. The origin and the first product coordinate as per the sequence and the blocking points are given to the AI Planner. AI planner considers the origin as initial point and the first product location as destination point and finds a possible solution to reach the destination by avoiding the blocking points. The SPUG reads the solution from the AI planner and moves accordingly to reach the product location as shown in figure **Fig. 13.**. Similarly, a path is calculated between a pair of points as per the shortest path sequence by the AI Planner as shown in figure **Fig. 14.** to figure **Fig. 18.**.

Updating the location: The location of the SPUG is calculated locally on the Raspberry PI. The location is updated with respect to the track. As mentioned earlier the SPUG is initially at origin facing Northwards. Based on the directions received from the AI planner it will go to the next coordinate. The next coordinate is identified upon sensing the junction(the point at which the three IR Sensors detect the track). Based on the direction it has travelled the coordinates are updated according to the below mentioned pseudo-code.

Once the destination given by the AI planner is reached it requests for the next destination till a Product location is reached. Once the SPUG reaches the

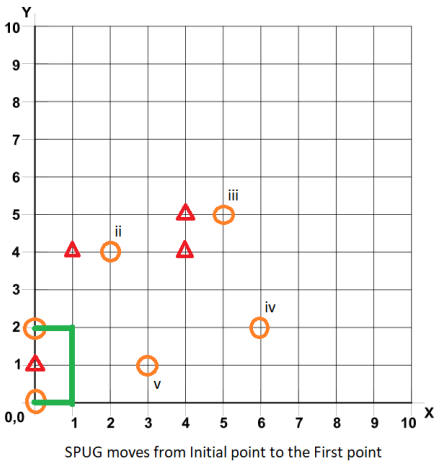

Fig. 13. Move - 1

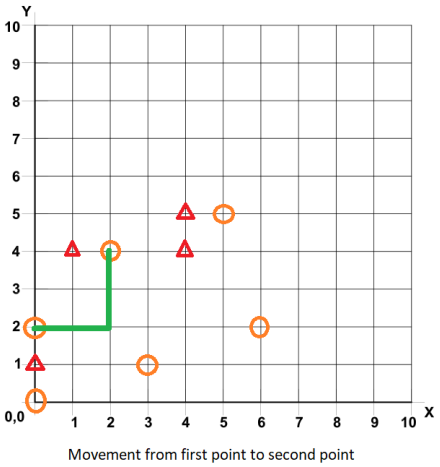

Fig. 14. Move - 2

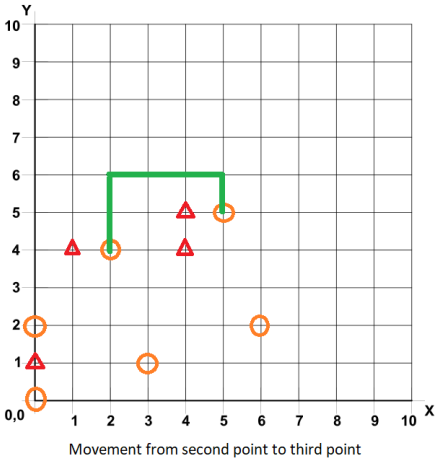

Fig. 15. Move - 3

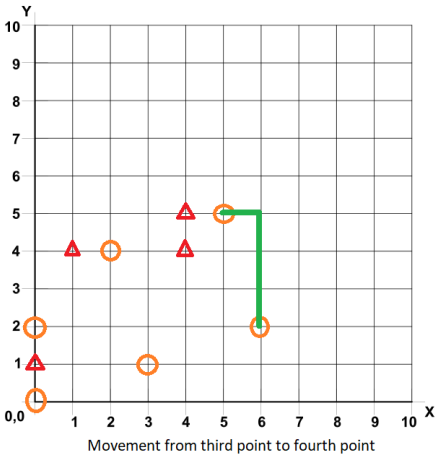

Fig. 16. Move - 4

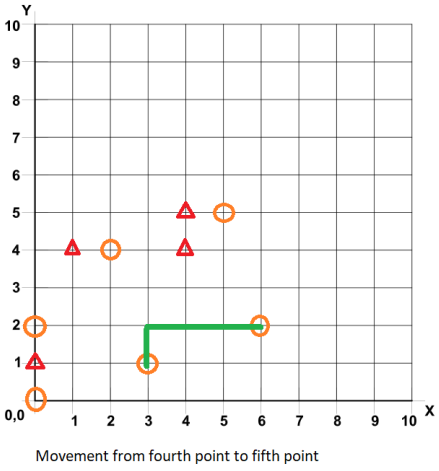

Fig. 17. Move - 5

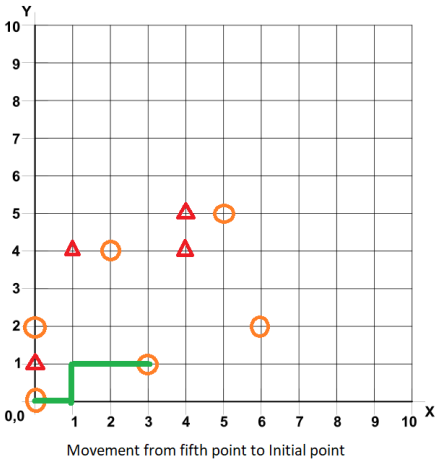

Fig. 18. Move - 6

product location a MQTT message will be sent to the server indicating that the product location is reached and a Buzzer will beep for 1 second indicating user that the product location is reached. User gets a notification in the mobile application to scan the QR code of the product and confirm if he wants to buy the product. If yes, the product will be added to the cart and a message will be sent to the SPUG to move the next product location. If no, SPUG just moves on to the next location. SPUG waits till it gets the response from the user.

Algorithm 1 Location Updating Algorithm

```
if Initial Direction is equal to North" then
    if Target Direction is equal to "North" then
        Move Straight
        Increment Y Coordinate
    else if Target Direction is equal to "South" then
        Move Reverse
        Decrement Y Coordinate
    else if Target Direction is equal to "East" then
        Turn Right
        Move Straight
        Increment X Coordinate
        Set Initial Direction to East
    else if Target Direction is equal to "West" then
        Turn Left
        Move Straight
        Decrement X Coordinate
        Set Initial Direction to West
else if Initial Direction is equal to "East" then
    if Target Direction is equal to "North" then
        Turn Left
        Move Straight
        Increment Y Coordinate
        Set Initial Direction to North
    else if Target Direction is equal to "South" then
        Turn Right
        Move Straight
        Decrement Y Coordinate
        Set Initial Direction to South
    else if Target Direction is equal to "East" then
        Move Straight
        Increment X Coordinate
    else if Target Direction is equal to "West" then
        Move Reverse
        Decrement X Coordinate
In the same way the algorithm extends to the other two directions.
```

Updating the blocking points: While the SPUG is at a particular coordinate a point occupy message is sent to the server as a MQTT message. Server receives this message and sets that particular coordinate as occupied. Once it moves away from the occupied point SPUG send a point "un-occupy" message to the server.

Server frees the coordinate which was occupied before. In this was the server gets to know the free coordinates and the location of all the SPUGs' in the map.

Obstacle detection: If there are any obstacles on path SPUG can detect the distance to the obstacles in the front. If the distance to the obstacle is below a threshold it stops moving and front two LED's glow Orange indicating the user that the cart stopped due to the obstacle. If the obstacle is removed then the LED's turn off and the SPUG starts moving back to the location it was scheduled to go.

4.5 AI Planner

The Artificial Intelligence Planner is used to determine the shortest path that needs to be followed by the SPUG on the track. The AI Planner of each SPUG ensures that the SPUG does not encounter points that are blocked by other SPUGs.

The initial point, the destination points and the blocked points are received by the SPUG from the central server through MQTT. The SPUG generates a domain file and a respective problem file with the help of python scripts. The script defines the initial state, which represents the current point the SPUG is at and the goal state is defined by the product location coordinate points where the SPUG should relocate to.

Once the PDDL files are generated, they are passed to the online AI planner (`http://editor.planning.domains/`) as a JSON message through HTTP. The online PDDL editor uses Serialized Iterated Width technique and then uses the Breadth First Spanned Forest algorithm as a fail-safe technique.

The response received is then written into a local text file on the raspberry pi. This plan has the detailed instructions on how the SPUG should move. This is read by the actuation sequence in the SPUG, which directly controls the actuation.

4.6 Sequence diagram

Sequence diagram for the overall flow of the implementation is below in figure **Fig. 19** .

4.7 Deployment diagram

Deployment diagram for the overall flow of the implementation is below in figure **Fig. 20.** .

5 Results

The store admin can view the number of items on shelf and the number of cars being used with the web view from server as shown in figure **Fig. 21.** and figure **Fig. 22**.

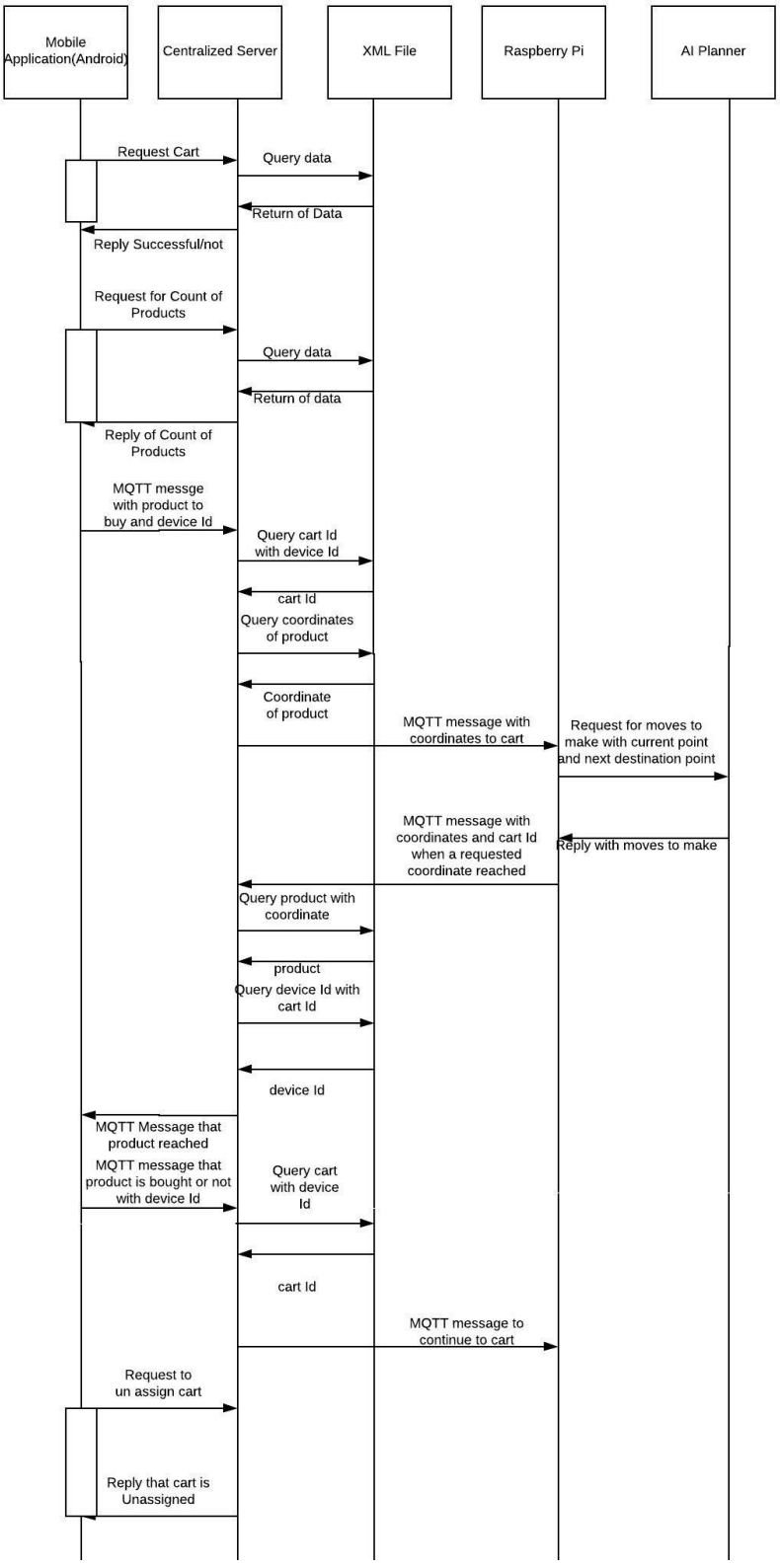

Fig. 19. Sequence Diagram

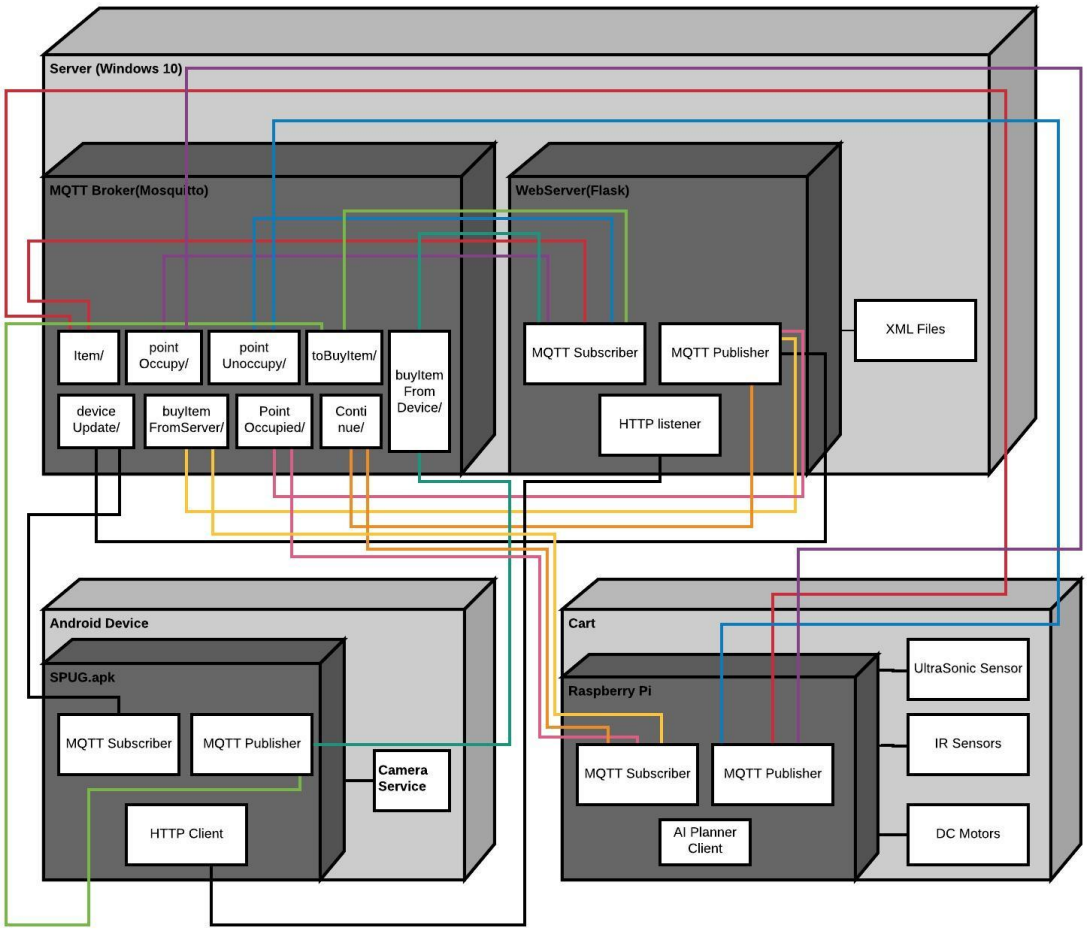

Fig. 20. Deployment diagram

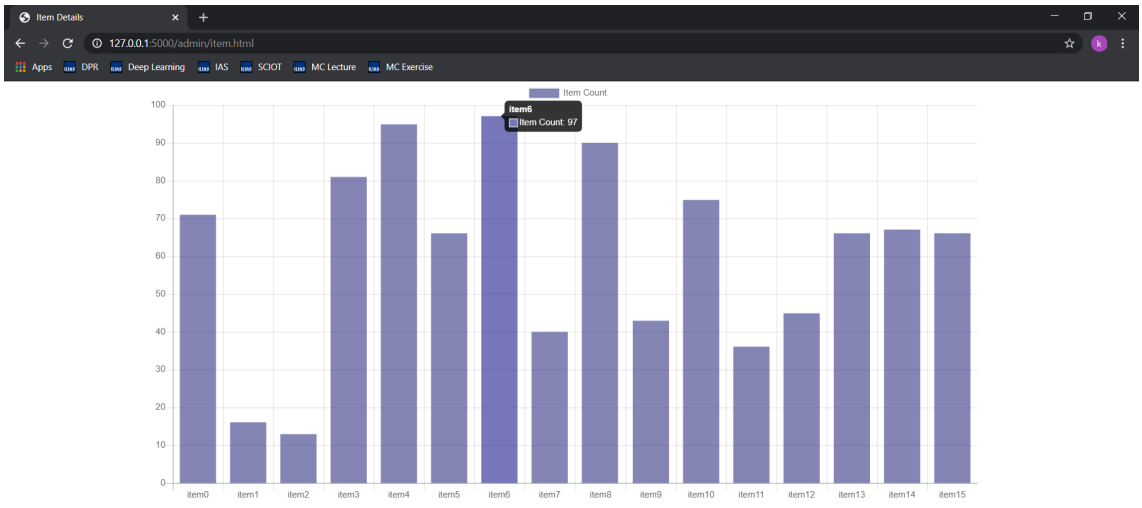

Fig. 21. Number of items on shelf

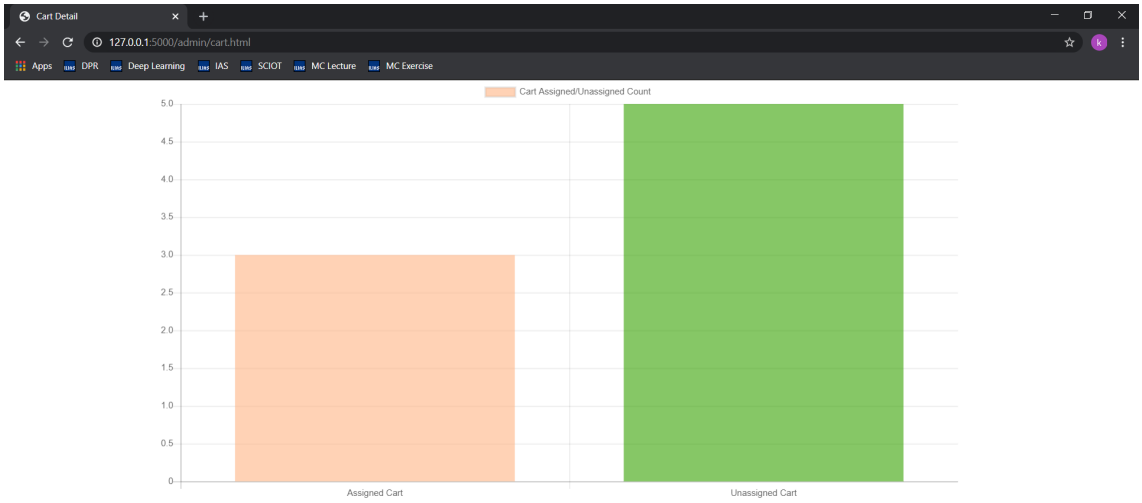

Fig. 22. Number of occupied and available carts

The customer can see the number of products purchased during the specific period and get the energy contents for those products through the mobile application as shown in figure **Fig. 23**.

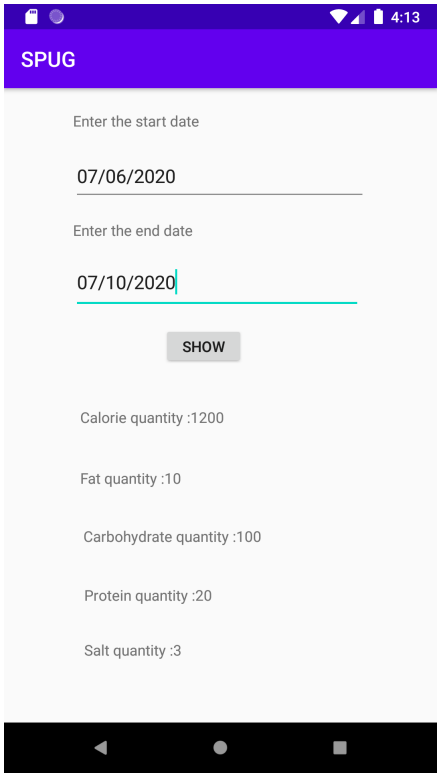

Fig. 23. Energy content of the products purchased

6 Discussion and Conclusions

Smart shopping is one of the biggest trends in the current world. We tried to implement some of the features to support this cause, however there are lot of new things that can be added to increase the comfort for users and enrich their shopping experience.

The current system has some limitations and needs lot of improvements to reach a high level of user experience, for example we had to considered a single track for the Shopping carts. This is difficult when there are lots of carts. The blocking points are considered to be stationary for a simplified approach, in the real world scenario this will not be the same. The mobile application

needs some improvements in terms of selecting the cart and payment methods. Also, currently there is no provision to purchase an item that was not earlier mentioned on the list. These were some of the limitations in the system that we implemented.

With the advent of social distancing norms everywhere the thought of shopping physically in a giant super market filled with lots of people is frightening. By implementing SPUG, we desire to bring better user experience on shopping by adhering to social distancing norms and increasing the comfort, hygiene and efficiency for the users using smart buildings, IoT, AI and connected devices.

References

7 Bibliography

1. A. Kumar, A. Gupta, S. Balamurugan, S. Balaji and R. Marimuthu, "Smart Shopping Cart," 2017 International conference on Microelectronic Devices, Circuits and Systems (ICMDCS), Vellore, 2017, pp. 1-4.

2. X. Liu, H. Zhang, J. Fang, G. Guan and Y. Huang, "Intelligent shopping cart with quick payment based on dynamic target tracking," 2016 4th International Conference on Cloud Computing and Intelligence Systems (CCIS), Beijing, 2016, pp. 88-93.

3. P. Chandrasekar and T. Sangeetha, "Smart shopping cart with automatic billing system through RFID and ZigBee" International Conference on Information Communication and Embedded Systems (ICICES2014), Chennai, 2014, pp. 1-4.
4. @inproceedings muise-icaps16demo-pd, title = Planning.Domains, author = Christian Muise, booktitle = The 26th International Conference on Automated Planning and Scheduling - Demonstrations, year = 2016
(github link:`https://github.com/LAPKT-dev/LAPKT-public/tree/master/planners/siw\_plus-then-bfs\_f-ffparser`)