اونيۏرسيتي مليسيا قهڠ السلطان عبد الله
UNIVERSITI MALAYSIA PAHANG
AL-SULTAN ABDULLAH

GROUP PROJECT 2023/2024

BCI1093 DATA STRUCTURE & ALGORITHMS

LECTURER: SITI SALWANI BINTI YAACOB

GROUP: 10

PROJECT TITLE: REMOTE ASSET MONITORING AND MAINTENANCE SYSTEM

MATRIC NO	NAME	SECTION
SD22048	CHONG JIN JYE	01G
SD22015	LIM JON VINCE	02G
SD22059	LOW ANN GIE	01G
SD22061	ANG MEI YING	01G

TABLE OF CONTENTS

No	Content	Page
1	Case Study 1: Mudah.my	3-4
2	Case Study 2: Samsung SmartThings	5-6
3	Case Study 3: IoT in Supply Chain Management Systems	7
4	Screenshot Of Program	8-18
5	Coding	19-34
6	Task Distribution	35
7	References	36

CASE STUDY 1: Mudah.my

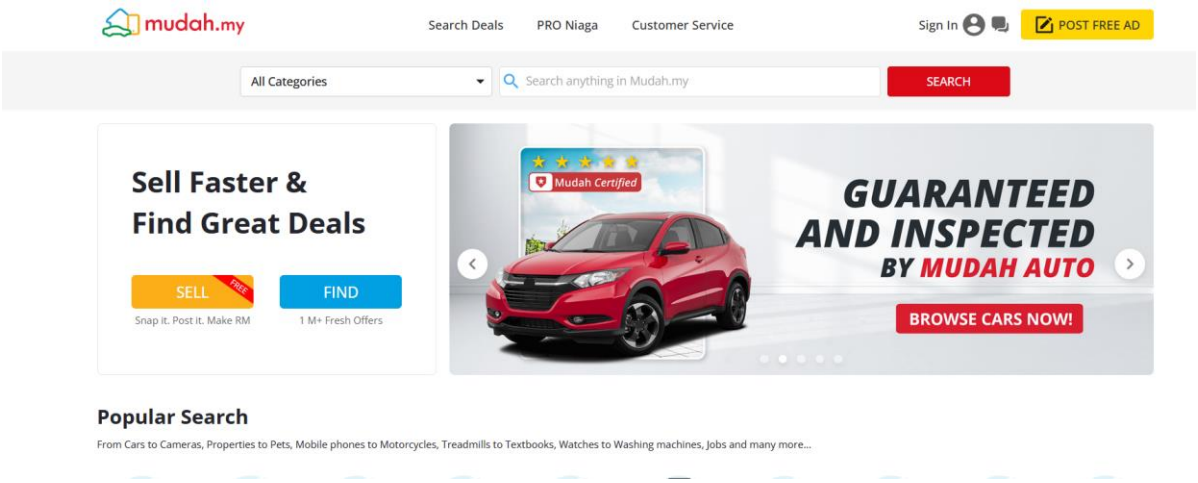


Figure 1: Mudah.my platform

Mudah.my is a Malaysian online platform that facilitates buying and selling across a wide range of products and services, including vehicles. With the integration of remote asset monitoring and maintenance capabilities for automobiles listed on their platform, Mudah.my hopes to improve the experience of purchasing and owning a car. The intention is to give purchasers additional information about the state of the cars they are interested in and to give sellers a platform with cutting-edge monitoring and maintenance services that add value.

Mudah.my is making things better for people who wish to buy or sell cars online. A very smart system that tracks the cars up for sale is being introduced. This system checks and shares real-time data about the health and performance of the car via smart devices. Thus, if someone is trying to sell an automobile, they might demonstrate the vehicle's performance to the prospective buyer. This increases everyone's confidence in their decision-making. It functions similarly to enhancing the safety and integrity of car transactions on Mudah.my.

The advantages of Mudah.my:

1. It offers a wide range of listings, which makes it an easy way for consumers to find a variety of goods and services such as electronics, cars, property and more.
2. The platform usually has an easy-to-use interface that makes it simple for buyers and sellers to browse listings, publish ads and interact.
3. Mudah.my is focused on the Malaysian market, making it a suitable platform for local buyers and sellers to connect and transact.

The disadvantages of Mudah.my:

1. In popular categories, there may be more competitors for visibility due to the large number of listings for comparable goods and services which might affect smaller sellers.
2. As with any online marketplace, there may be concerns related to the trustworthiness of sellers and the safety of transactions. Users should exercise caution and verify the legitimacy of listings.
3. The quality and accuracy of product or service listings can vary. Users may encounter incomplete or misleading information, requiring them to perform due diligence before making transactions.

CASE STUDY 2: Samsung SmartThings

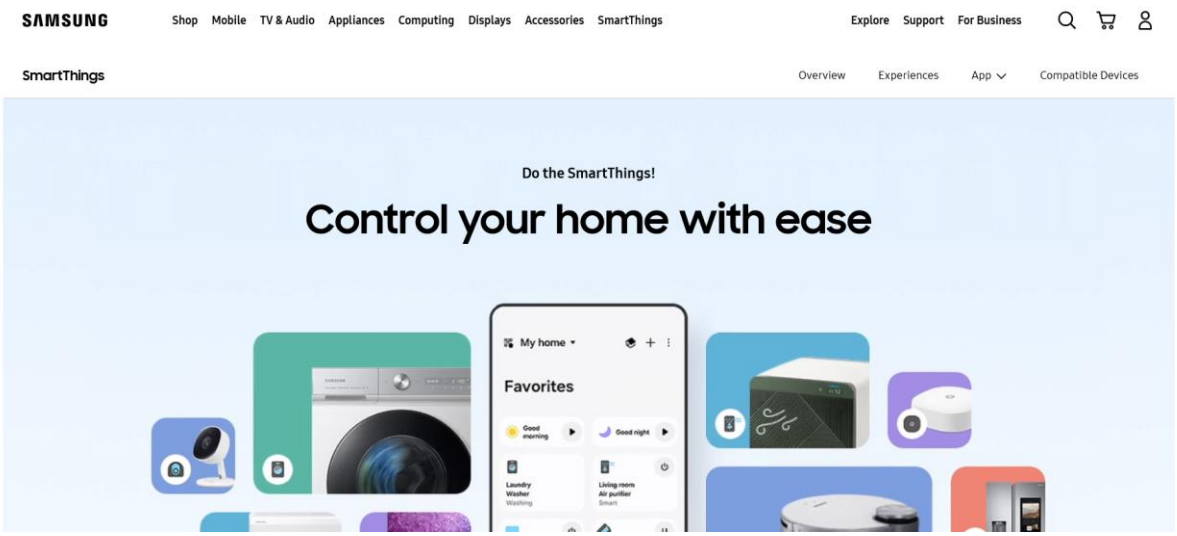


Figure 2: Samsung SmartThings platform

Samsung SmartThings is an innovative smart home platform that has gained prominence in the realm of home automation. It was created by Samsung Electronics and acts as a centralized hub for tying together and managing a range of smart devices in the house. Launched with the goal of enhancing convenience, efficiency and connectivity, SmartThings has become a key player in the growing ecosystem of Internet of Things (IoT) devices. Because of its adaptability, users may easily integrate and control a wide range of smart products including as security cameras, smart lights, smart refrigerators and smart speakers and sound systems.

With Samsung SmartThings, buyers can monitor and manage their smart home appliances using an advanced and intuitive interface. Through a specialised mobile app, users can access SmartThings and monitor the condition of their devices, receive warnings and even automate routines for a more intelligent and efficient living environment. SmartThings' open architecture allows it to work with a large variety of third-party devices, giving customers the freedom to design a personalised and networked smart home ecosystem.

The advantages of Samsung SmartThings:

1. Unified control is provided by the Samsung SmartThings. This streamlines the process of controlling several devices from a single platform for landlords and improves administration as more devices are integrated.
2. Potential vandalism, theft and breaches can be greatly decreased by integrating with sensors, cameras, smart locks and other security devices. Building security can also be improved by integrating occupancy sensors, surveillance cameras, window and door sensors.
3. Samsung SmartThings enables users to create automated routines and scenarios, triggering specific actions based on conditions or events. This streamlines daily duties and activities, increasing ease and efficiency.

The disadvantages of Samsung SmartThings:

1. Samsung SmartThings heavily relies on a stable internet connection. During network outages, users may experience disruptions in service, affecting the platform's performance.
2. As with many other smart home systems, security and privacy issues with data can come up. Concerns exist regarding technology's vulnerability to hackers as well. In orders to safeguard their smart home network, users should be aware of the data that SmartThings gathers and stores and adopt the appropriate security measures.
3. Some users have reported occasional connectivity issues, leading to disruptions in remote control and monitoring. These issues can impact the reliability of the system.

CASE STUDY 3: IoT in Supply Chain Management Systems

There are three stages of the supply chain from farm to fork using the implementation of IoT. At the agricultural end, IoT has produce maximum production. It can keep the environmentally friendly. Water distribution, soil management and meteorological information access have all been made simple by IoT technologies like cloud computing and sensors. There are some examples of using IoT in agriculture. The quantity and quality of the grains in soils can be monitored by using IoT Soil Moisture Sensors. Besides, analysing and controlling crop quality by using multispectral images. At the storage end, the real time temperature tracking sensors can improve food safety. It can also use to monitor the moisture, humidity, pH and odour of food. At the delivery end, the delivery and shipping process is another component of the supply chain where food goods require special attention. This critical process can be streamlined in real time and monitored remotely by using an IoT-powered transportation logistics tool. For instance, the movement of commodities may be tracked using radio-frequency identification (RFID) tags and QR codes.

The advantages of using IoT in supply chain management systems:

1. It can track the real-time goods and location. IoT-based systems can help to reduce delivery times and predict the customers waiting times.
2. Using IoT can reduce costs such as operating, transportation and product handling costs.
3. It can help to monitor the storage condition and prevent goods from spoiling.

The disadvantages of using IoT in supply chain management systems:

1. To successfully utilize IoT devices, an organization must have stable internet access with sufficient speed.
2. The massive amount of data generated by IoT devices can be a target for cyberattacks. If not adequately secured, sensitive information such as product details, shipment schedules and customer data could be compromised.
3. It has issues in data storage. When the IoT in the supply chain works properly, it gathers an adequate amount of data collected through the sensors. The organization requires enough server power to maintain the process smoothly.

SCREENSHOTS OF PROGRAM

Main Menu (Remote Asset Monitoring and Maintenance System)

```
Remote Asset Monitoring and Maintenance System
Enter your car details for monitoring and maintenance services
-----------------------------------------------


E - Enter Information
U - Update by Car Plate Number
D - Delete Record
L - Display Records
F - Search
R - Repair and Maintenance
S - Sort Records
I - Info
X - Exit

Enter choice:
```

Display Initial Information

```
Enter choice: L
```

No.	Client Name	Client IC	Phone Number	Car Plate	Car Model	Car Make	Car Price	Car Mileage
1	Elly	021110102645	0124561028	JNW2785	Accord	Toyota	RM20125.00	42355 km
2	John	021110102545	01145626128	WWK1203	Civic	Honda	RM25000.00	17905 km
3	Johnny	800312104235	0106548525	CEV8927	CR-V	Honda	RM91800.00	111197 km
4	Aaron	740723103120	0167003245	BNN8116	Vios	Toyota	RM40800.00	110000 km
5	Cindy	920403109823	0121456382	AAU4730	Altis	Toyota	RM59800.00	80000 km
6	Kelvin	840218103452	0163241235	ANE5441	Myvi	Perodua	RM59900.00	4999 km
7	William	970520109420	01174568912	BRJ8241	Axia	Perodua	RM38600.00	5200 km
8	Kelly	940109105241	0123571231	JUE4427	Saga	Proton	RM23300.00	80000 km
9	Alicia	761218103941	0162335684	PQF2688	HR-V	Honda	RM86900.00	100000 km
10	Rachel	980322104528	0108462357	MAS2475	Camry	Toyota	RM30000.00	102000 km

Insert At The End

→ Input:

```
Enter choice: E

------------------------------------------------------------

Enter Client Name: Arif
Enter Client IC: 970530101223
Enter Phone Number: 0123622301
Enter Car Plate: WKK2016
Enter Car Model: Mustang
Enter Car Make: Ford
Enter Car Price: 461271
Enter Car Mileage: 28100
```

→ Output:

```
Enter choice: L

No.   Client Name       Client IC       Phone Number    Car Plate       Car Model       Car Make        Car Price       Car Mileage
1       Elly            021110102645    0124561028      JNW2785         Accord          Toyota          RM20125.00      42355 km
2       John            021110102545    01145626128     WWK1203         Civic           Honda           RM25000.00      17905 km
3       Johnny          800312104235    0106548525      CEV8927         CR-V            Honda           RM91800.00      111197 km
4       Aaron           740723103120    0167003245      BNN8116         Vios            Toyota          RM40800.00      110000 km
5       Cindy           920403109823    0121456382      AAU4730         Altis           Toyota          RM59800.00      80000 km
6       Kelvin          840218103452    0163241235      ANE5441         Myvi            Perodua         RM59900.00      4999 km
7       William         970520109420    01174568912     BRJ8241         Axia            Perodua         RM38600.00      5200 km
8       Kelly           940109105241    0123571231      JUE4427         Saga            Proton          RM23300.00      80000 km
9       Alicia          761218103941    0162335684      PQF2688         HR-V            Honda           RM86900.00      100000 km
10      Rachel          980322104528    0108462357      MAS2475         Camry           Toyota          RM30000.00      102000 km
11      Arif            970530101223    0123622301      WKK2016         Mustang         Ford            RM461271.00     28100 km
```

Stack - push

→ Update (alter phone number from 0106548525 to 0123521202 where car plate is CEV8927)

```
Enter choice: U

Enter the car plate that you want to update: CEV8927

Car Plate Found !!!

Enter Client Name: Johnny
Enter Client IC: 800312104235
Enter Phone Number: 0123521202
Enter Car Plate: CEV8927
Enter Car Model: CR-V
Enter Car Make: Honda
Enter Car Price: RM91800.00
Enter Car Mileage: 111197 km
```

→ Display updated output

```
Enter choice: L
```

No.	Client Name	Client IC	Phone Number	Car Plate	Car Model	Car Make	Car Price	Car Mileage
1	Johnny	800312104235	0123521202	CEV8927	CR-V	Honda	RM91800.00	111197 km
2	Elly	021110102645	0124561028	JNW2785	Accord	Toyota	RM20125.00	42355 km
3	John	021110102545	01145626128	WWK1203	Civic	Honda	RM25000.00	17905 km
4	Aaron	740723103120	0167003245	BNN8116	Vios	Toyota	RM40800.00	110000 km
5	Cindy	920403109823	0121456382	AAU4730	Altis	Toyota	RM59800.00	80000 km
6	Kelvin	840218103452	0163241235	ANE5441	Myvi	Perodua	RM59900.00	4999 km
7	William	970520109420	01174568912	BRJ8241	Axia	Perodua	RM38600.00	5200 km
8	Kelly	940109105241	0123571231	JUE4427	Saga	Proton	RM23300.00	80000 km
9	Alicia	761218103941	0162335684	PQF2688	HR-V	Honda	RM86900.00	100000 km
10	Rachel	980322104528	0108462357	MAS2475	Camry	Toyota	RM30000.00	102000 km

Delete At The Middle

→ Insert a car plate number to be delete

```
Enter choice: D

D - Delete by Car Plate

Q - Delete at beginning

Enter choice: D

Enter Car Plate Number to delete: PQF2688
```

→ Output (client with car plate PQF2688 is deleted):

```
No.   Client Name       Client IC       Phone Number    Car Plate       Car Model       Car Make        Car Price       Car Mileage
1       Elly            021110102645    0124561028      JNW2785         Accord          Toyota          RM20125.00      42355 km
2       John            021110102545    01145626128     WWK1203         Civic           Honda           RM25000.00      17905 km
3       Johnny          800312104235    0106548525      CEV8927         CR-V            Honda           RM91800.00      111197 km
4       Aaron           740723103120    0167003245      BNN8116         Vios            Toyota          RM40800.00      110000 km
5       Cindy           920403109823    0121456382      AAU4730         Altis           Toyota          RM59800.00      80000 km
6       Kelvin          840218103452    0163241235      ANE5441         Myvi            Perodua         RM59900.00      4999 km
7       William         970520109420    01174568912     BRJ8241         Axia            Perodua         RM38600.00      5200 km
8       Kelly           940109105241    0123571231      JUE4427         Saga            Proton          RM23300.00      80000 km
9       Rachel          980322104528    0108462357      MAS2475         Camry           Toyota          RM30000.00      102000 km
```

Stack - Pop

Delete At The Beginning

→ Input:

```
Enter choice: D

D - Delete by Car Plate

Q - Delete at beginning

Enter choice: Q
```

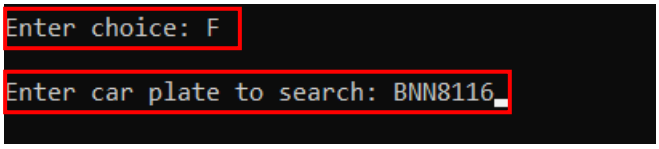
→ Output (The client Elly which is located at number one is deleted):

No.	Client Name	Client IC	Phone Number	Car Plate	Car Model	Car Make	Car Price	Car Mileage
1	John	021110102545	01145626128	WWK1203	Civic	Honda	RM25000.00	17905 km
2	Johnny	800312104235	0106548525	CEV8927	CR-V	Honda	RM91800.00	111197 km
3	Aaron	740723103120	0167003245	BNN8116	Vios	Toyota	RM40800.00	110000 km
4	Cindy	920403109823	0121456382	AAU4730	Altis	Toyota	RM59800.00	80000 km
5	Kelvin	840218103452	0163241235	ANE5441	Myvi	Perodua	RM59900.00	4999 km
6	William	970520109420	01174568912	BRJ8241	Axia	Perodua	RM38600.00	5200 km
7	Kelly	940109105241	0123571231	JUE4427	Saga	Proton	RM23300.00	80000 km
8	Alicia	761218103941	0162335684	PQF2688	HR-V	Honda	RM86900.00	100000 km
9	Rachel	980322104528	0108462357	MAS2475	Camry	Toyota	RM30000.00	102000 km

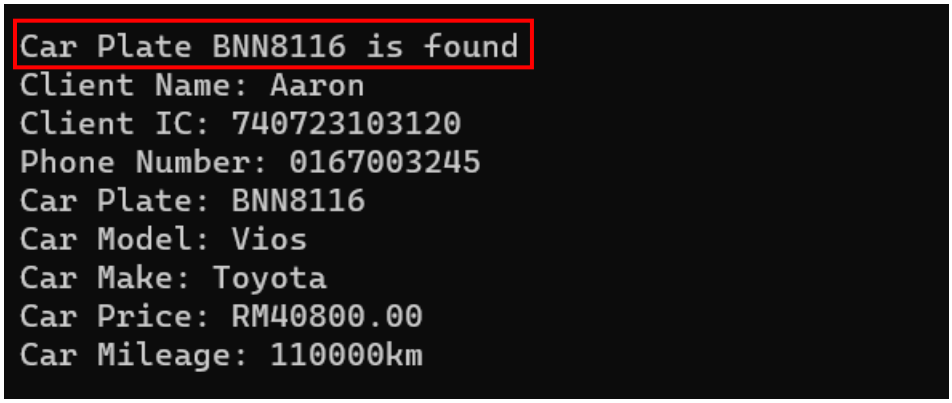
Linear Sequential Search

1. Search with correct input

→ Input:

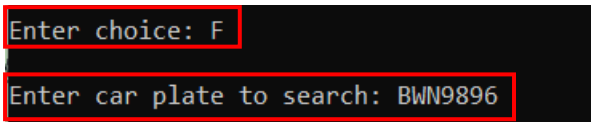


→ Output:

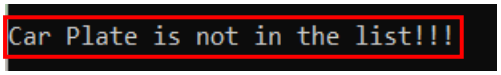


2.Search by wrong input

→ Input:



→ Output:



Bubble Sort

1. Sort by category car price

→ Input:

```
Enter choice: S
Sort by:
1 - Car Price
2 - Car Mileage
Enter choice: 1
```

→ Output:

```
The records have been sorted by car price.
```

No.	Client Name	Client IC	Phone Number	Car Plate	Car Model	Car Make	Car Price	Car Mileage
1	Elly	021110102645	0124561028	JNW2785	Accord	Toyota	RM20125.00	42355 km
2	Kelly	940109105241	0123571231	JUE4427	Saga	Proton	RM23300.00	80000 km
3	John	021110102545	01145626128	WWK1203	Civic	Honda	RM25000.00	17905 km
4	Rachel	980322104528	0108462357	MAS2475	Camry	Toyota	RM30000.00	102000 km
5	William	970520109420	01174568912	BRJ8241	Axia	Perodua	RM38600.00	5200 km
6	Aaron	740723103120	0167003245	BNN8116	Vios	Toyota	RM40800.00	110000 km
7	Cindy	920403109823	0121456382	AAU4730	Altis	Toyota	RM59800.00	80000 km
8	Kelvin	840218103452	0163241235	ANE5441	Myvi	Perodua	RM59900.00	4999 km
9	Alicia	761218103941	0162335684	PQF2688	HR-V	Honda	RM86900.00	100000 km
10	Johnny	800312104235	0106548525	CEV8927	CR-V	Honda	RM91800.00	111197 km

2. Sort by category car mileage

→ Input:

```
Enter choice: S
Sort by:
1 - Car Price
2 - Car Mileage
Enter choice: 2
```

→ Output:

```
The records have been sorted by car mileage.
```

No.	Client Name	Client IC	Phone Number	Car Plate	Car Model	Car Make	Car Price	Car Mileage
1	Kelvin	840218103452	0163241235	ANE5441	Myvi	Perodua	RM59900.00	4999 km
2	William	970520109420	01174568912	BRJ8241	Axia	Perodua	RM38600.00	5200 km
3	John	021110102545	01145626128	WWK1203	Civic	Honda	RM25000.00	17905 km
4	Elly	021110102645	0124561028	JNW2785	Accord	Toyota	RM20125.00	42355 km
5	Cindy	920403109823	0121456382	AAU4730	Altis	Toyota	RM59800.00	80000 km
6	Kelly	940109105241	0123571231	JUE4427	Saga	Proton	RM23300.00	80000 km
7	Alicia	761218103941	0162335684	PQF2688	HR-V	Honda	RM86900.00	100000 km
8	Rachel	980322104528	0108462357	MAS2475	Camry	Toyota	RM30000.00	102000 km
9	Aaron	740723103120	0167003245	BNN8116	Vios	Toyota	RM40800.00	110000 km
10	Johnny	800312104235	0106548525	CEV8927	CR-V	Honda	RM91800.00	111197 km

Repair and Maintenance

1. Enqueue is implemented without Dequeue

→ Input:

```
Enter choice: R

Enter car plate to register repair and maintenance: BNN8116

Enter date of service (in DD-MM-YYYY): 30-05-2023

Enter repair details: Break
The repair and maintenance registration is successful!

Do you want to register repair and maintenance for another car? (Y/N):Y

Enter car plate to register repair and maintenance: AAU4730

Enter date of service (in DD-MM-YYYY): 15-09-2023

Enter repair details: Tyre
The repair and maintenance registration is successful!

Do you want to register repair and maintenance for another car? (Y/N):Y

Enter car plate to register repair and maintenance: WWK1203

Enter date of service (in DD-MM-YYYY): 18-12-2023

Enter repair details: Lightbulb
The repair and maintenance registration is successful!

Do you want to register repair and maintenance for another car? (Y/N):N

Do you want to dequeue repair and maintenance from the list of car? (Y/N):N
```

→ Output:

```
No.     Client Name     Car Plate       Car Make        Car Model       Date of Service Repair Details
1.      Aaron           BNN8116         Toyota          Vios            30-05-2023      Break
2.      Cindy           AAU4730         Toyota          Altis           15-09-2023      Tyre
3.      John            WWK1203         Honda           Civic           18-12-2023      Lightbulb
```

2. Enqueue and Dequeue are implemented

→Input:

```
Enter choice: R

Enter car plate to register repair and maintenance: BNN8116

Enter date of service (in DD-MM-YYYY): 30-05-2023

Enter repair details: Break
The repair and maintenance registration is successful!

Do you want to register repair and maintenance for another car? (Y/N):Y

Enter car plate to register repair and maintenance: AAU4730

Enter date of service (in DD-MM-YYYY): 15-09-2023

Enter repair details: Tyre
The repair and maintenance registration is successful!

Do you want to register repair and maintenance for another car? (Y/N):Y

Enter car plate to register repair and maintenance: WWK1203

Enter date of service (in DD-MM-YYYY): 18-12-2023

Enter repair details: Lightbulb
The repair and maintenance registration is successful!

Do you want to register repair and maintenance for another car? (Y/N):n

Do you want to dequeue repair and maintenance from the list of car? (Y/N):y

Dequeue successful!

Do you want to register repair and maintenance for another car? (Y/N):N

Do you want to dequeue repair and maintenance from the list of car? (Y/N):N
```

→ Output:

```
No.     Client Name     Car Plate       Car Make        Car Model       Date of Service Repair Details
1.      Cindy           AAU4730         Toyota          Altis           15-09-2023      Tyre
2.      John            WWK1203         Honda           Civic           18-12-2023      Lightbulb
```

Info

→Input:

```
Enter choice: I

Enter Insurance Provider: YZ Insurance Company
Enter Policy Number: A123456789
Enter Coverage Amount: 50000
Enter Driver Name: Jane
Enter Driver License: DL123456789
Enter Driver Age: 35
```

→ Output:

```
Details:

Insurance Provider: YZ Insurance Company
Policy Number: A123456789
Coverage Amount: RM50000.00
Driver Name: Jane
Driver License: DL123456789
Driver Age: 35
```

CODING

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <ctype.h>

// for struct use
struct Car
{
    char carmodel[20];
    char carmake[20];
    char carplate[20];
    float carprice;
    int carmileage;
};
struct Client
{
    char clientname[30];
    char clientIC[20];
    char phoneNum[20];
};
struct Asset
{
    struct Car car;
    struct Client client;
    struct Asset *ptrnext;
};

// for sort use
struct ArrayStore
{
    struct Asset *array;
    struct RepairCar *array2;
    int size;
};

//repair and maintenance struct
struct RepairCar {
      char rpClientname[30];
      char rpCarplate [20];
      char rpCarmake [20];
      char rpCarmodel [20];
      char Date_of_Service [20];
      char Repair_Details [30];
};


struct InsuranceInfo {
    char insuranceProvider[30];
    char policyNumber[20];
    float coverageAmount;
};


struct DriverInfo {
```

```
    char driverName[30];
    char driverLicense[20];
    int age;
};

struct driver {

    struct InsuranceInfo insurance_info;
    struct DriverInfo driver_info;

};

// for linked list use
#define TRUE 1
#define FALSE 0
void default_asset(char clientname[30], char clientIC[20], char
phoneNum[20], char carplate[20], char carmodel[20], char carmake[20],
float carprice, int carmileage);
void update_bycarplate();
void new_number();
void new_number1();
void list_number();
void delete_record();
void delete_bycarplate();
void delete_beginning();
int dequeueRepair(int);
void search_bycarplate();
void linkedlist_to_array(struct Asset *head, struct ArrayStore *store);
void bubble_sort_price(struct ArrayStore *store);
void bubble_sort_mileage(struct ArrayStore *store);
void display_array(struct ArrayStore *store);
void list_sort();
void repairRegister();
void Info();

struct Asset *headptr, *newptr, *currentptr, *previousptr;
struct RepairCar repair[20];
struct InsuranceInfo insurance();

void display(struct driver insurance, struct driver driverInfo);

int main()
{
    headptr = NULL;

    default_asset("Elly", "021110102645", "0124561028", "JNW2785",
"Accord", "Toyota", 20125, 42355);
    default_asset("John", "021110102545", "01145626128", "WWK1203",
"Civic", "Honda", 25000, 17905);
    default_asset("Johnny","800312104235","0106548525","CEV8927","CR-
V","Honda",91800,111197);

default_asset("Aaron","740723103120","0167003245","BNN8116","Vios","Toy
ota",40800,110000);

default_asset("Cindy","920403109823","0121456382","AAU4730","Altis","To
yota",59800,80000);
```

```
default_asset("Kelvin","840218103452","0163241235","ANE5441","Myvi","Perodua",59900,4999);

default_asset("William","970520109420","01174568912","BRJ8241","Axia","Perodua",38600,5200);

default_asset("Kelly","940109105241","0123571231","JUE4427","Saga","Proton",23300,80000);
    default_asset("Alicia","761218103941","0162335684","PQF2688","HR-V","Honda",86900,100000);

default_asset("Rachel","980322104528","0108462357","MAS2475","Camry","Toyota",30000,102000);

    printf("Remote Asset Monitoring and Maintenance System\n");
    printf("Enter your car details for monitoring and maintenance services\n");
    printf("-----------------------------------------------\n");

    char ch;
    int choice=TRUE;


    while(choice==TRUE)

    {
        printf("\n\nE - Enter Information");
        printf("\nU - Update by Car Plate Number");
        printf("\nD - Delete Record");
        printf("\nL - Display Records");
        printf("\nF - Search");
        printf("\nR - Repair and Maintenance");
        printf("\nS - Sort Records");
        printf("\nI - Info");
        printf("\nX - Exit\n");
        printf("\nEnter choice: ");
        scanf(" %c",&ch);
        ch=toupper(ch);
        switch(ch)
        {
            case 'E':new_number();break;
            case 'U':update_bycarplate();break;
            case 'D':delete_record();break;
            case 'L':list_number();break;
            case 'F':search_bycarplate();break;
            case 'R':repairRegister();break;
            case 'S':list_sort(); break;
            case 'I':Info(); break;
            case 'X':choice=FALSE; break;

            default: printf("\nEnter only one from the above");
        }
    }
    return 0;

}
```

```
//setting default asset data
void default_asset(char clientname[30], char clientIC[20], char
phoneNum[20], char carplate[20], char carmodel[20], char carmake[20],
float carprice, int carmileage){

    newptr = (struct Asset *)malloc(sizeof(struct Asset));

    strcpy(newptr->client.clientname, clientname);
    strcpy(newptr->client.clientIC, clientIC);
    strcpy(newptr->client.phoneNum, phoneNum);
    strcpy(newptr->car.carplate, carplate);
    strcpy(newptr->car.carmodel, carmodel);
    strcpy(newptr->car.carmake, carmake);
    newptr->car.carprice = carprice;
    newptr->car.carmileage = carmileage;

    if (headptr==NULL)
    {
        headptr=newptr;
        newptr->ptrnext=NULL;
    }

    else
    {
        currentptr=headptr;

        while(currentptr->ptrnext !=NULL)
        {
            currentptr=currentptr->ptrnext;
        }

        currentptr->ptrnext=newptr;
        newptr->ptrnext=NULL;
    }


}

void new_number() // insert at the end
{
    newptr=(struct Asset *)malloc(sizeof(struct Asset));

    printf("\n-------------------------------------------------------
-----\n");
    fflush(stdin);
    printf("\nEnter Client Name: ");
    scanf("%[^\n]s", newptr->client.clientname);
    printf("Enter Client IC: ");
    scanf("%s", newptr->client.clientIC);
    printf("Enter Phone Number: ");
    scanf("%s", newptr->client.phoneNum);
    printf("Enter Car Plate: ");
    scanf("%s", newptr->car.carplate);
    printf("Enter Car Model: ");
    scanf("%s", newptr->car.carmodel);
    printf("Enter Car Make: ");
    scanf("%s", newptr->car.carmake);
    printf("Enter Car Price: ");
```

```
    scanf("%f", &newptr->car.carprice);
    fflush(stdin);
    printf("Enter Car Mileage: ");
    scanf("%d", &newptr->car.carmileage);
    fflush(stdin);


    if (headptr==NULL)
    {
        headptr=newptr;
        newptr->ptrnext=NULL;
    }

    else
    {
        currentptr=headptr;

        while(currentptr->ptrnext !=NULL)
        {
            currentptr=currentptr->ptrnext;
        }

        currentptr->ptrnext=newptr;
        newptr->ptrnext=NULL;
    }


}

//update car struct data
void update_bycarplate(){
      char input[50];
    int i;

    if (headptr == (struct Asset *)NULL){

        printf("\nThe list is empty");
        printf("\nPress Enter to continue.");
    }
    else{
        printf("\nEnter the car plate that you want to update: ");
        scanf("%s", input);

        currentptr = headptr;
        while (currentptr-> ptrnext != (struct Asset*)NULL){

            if (strcmp(currentptr->car.carplate, input) == 0){
                break;
            }
            else{

                previousptr = currentptr;
                currentptr = currentptr->ptrnext;}}

        if (strcmp(currentptr->car.carplate, input)==0)
        {
            if(currentptr==headptr){
```

```
                headptr=currentptr->ptrnext;
                free(currentptr);}
            else{
                previousptr->ptrnext=currentptr->ptrnext;
                free(currentptr);
            }
            printf("\nCar Plate Found !!!\n");

            new_number1();

        }
        else{
            printf("\nCar Plate is not in the list!");
            printf("\nPress Enter to continue.\n");

        }
    }
    getch();

}

void new_number1() // push
{
    newptr=(struct Asset *)malloc(sizeof (struct Asset)); // pointer to
a new
    //      memory allocation

    if (headptr==NULL)//node is empty?
    {headptr=newptr; //first pointer point to first node
        newptr->ptrnext=NULL; //first node pointer point to null
    }

    else
    {
        newptr->ptrnext=headptr;// new node pointer point to previous
first node
        headptr=newptr; // head point to new node,new node become first
node

    }

    fflush(stdin);
    printf("\nEnter Client Name: ");
    scanf("%[^\n]s", newptr->client.clientname);
    printf("Enter Client IC: ");
    scanf("%s", newptr->client.clientIC);
    printf("Enter Phone Number: ");
    scanf("%s", newptr->client.phoneNum);
    printf("Enter Car Plate: ");
    scanf("%s", newptr->car.carplate);
    printf("Enter Car Model: ");
    scanf("%s", newptr->car.carmodel);
    printf("Enter Car Make: ");
    scanf("%s", newptr->car.carmake);
    printf("Enter Car Price: ");
    scanf("%f", &newptr->car.carprice);
    fflush(stdin);
    printf("Enter Car Mileage: ");
```

```
    scanf("%d", &newptr->car.carmileage);
    fflush(stdin);
}

void list_number()
{
    if (headptr==(struct Asset*)NULL) //empty list
    {
        printf("\nEmpty list");
        return;
    }


    printf("\n");
    printf("\nNo.   Client Name\tClient IC\tPhone Number\tCar
Plate\tCar Model\tCar Make\tCar Price\tCar Mileage\n");
    currentptr=headptr;
    struct ArrayStore finalarray;

    linkedlist_to_array(headptr, &finalarray);

    display_array(&finalarray);

    free(finalarray.array);

}
//delete the record
void delete_record(){

    char ch;
    printf("\nD - Delete by Car Plate\n");
    printf("\nQ - Delete at beginning\n");
    printf("\nEnter choice: ");
    scanf(" %c",&ch);
    ch=toupper(ch);
    switch(ch){

        case 'D': delete_bycarplate();break;
        case 'Q': delete_beginning();break;

        default: printf("\nEnter only one from the above");}

}
//delete the record by car plate
void delete_bycarplate()
{
    char input[10];

    if (headptr==NULL)//node is empty?
    {
        printf("\n\nThe list is empty. Cannot delete!!!\n");//the list
is empty
    }

    else
    {
        printf("\nEnter Car Plate Number to delete: ");
        scanf("%s",input);
```

```
        currentptr=headptr;

        while(currentptr->ptrnext!=NULL)
        {

            if (strcmp(currentptr->car.carplate, input) == 0) //found
the location
            {
                break;
            }
            else
            {
                previousptr=currentptr;//save the previous current
address
                currentptr=currentptr->ptrnext; //point to next node
            }

        }

        if (strcmp(currentptr->car.carplate, input) == 0)
        {
            if (currentptr==headptr) //number is the first and only
node in list
            {
                headptr=currentptr ->ptrnext; //head point to NULL
                free(currentptr);
            }
            else //delete at the middle of link list
            {
                previousptr->ptrnext=currentptr->ptrnext;//previous
node point to next node
                free(currentptr);//destroy  node, free the memory.
            }
        }

        else
            printf("\nNumber to be deleted is not in the list!!! ");


    }

}

//delete the beginning record
void delete_beginning(){

      if (headptr==NULL){
    printf("\n\nThe list is empty. Cannot delete!!!\n");}

      else{
       currentptr=headptr;//current contain the address of first node
       headptr=headptr->ptrnext;;//head point to second previous node
       free(currentptr);//destroy first node, free the memory.
       }
}

//dequeue the beginning record of repair struct
```

```
int dequeueRepair(int count){
      int r;
    if (count==0)//node is empty?
    {
        printf("\n\nThe list is empty. Cannot delete!!!\n");//the list
is empty
    }

    else
    {
      for(r=0;r<count;r++){
            repair[r]=repair[r+1];
            }
            count=count-1;
            printf("\nDequeue successful!\n");
    }
    return count;
}

//searching record by car plate
void search_bycarplate(){

    char searchValue[20];

    if (headptr==NULL){
        printf("\n\nThe list is empty!!!\n");}
    else{
      printf("\nEnter car plate to search: ");
      scanf("%s",&searchValue);

    currentptr=headptr;

    while(currentptr->ptrnext!=NULL){

        if (strcmp(currentptr->car.carplate, searchValue)==0){

            printf("\nCar Plate %s is found\n", searchValue);
            fflush(stdin);
            printf("Client Name: %s\n", currentptr->client.clientname);
            printf("Client IC: %s\n", currentptr->client.clientIC);
            printf("Phone Number: %s\n", currentptr->client.phoneNum);
            printf("Car Plate: %s\n", currentptr->car.carplate);
            printf("Car Model: %s\n", currentptr->car.carmodel);
            printf("Car Make: %s\n", currentptr->car.carmake);
            printf("Car Price: %.2f\n", currentptr->car.carprice);
            fflush(stdin);
            printf("Car Mileage: %d\n", currentptr->car.carmileage);
            fflush(stdin);

            return;}
        else{
          currentptr=currentptr->ptrnext;}
            }

    if(strcmp(currentptr->car.carplate, searchValue)==0){

        printf("\nCar Plate %s is found", searchValue);
        fflush(stdin);
```

```
        printf("Client Name: %s\n", currentptr->client.clientname);
        printf("Client IC: %s\n", currentptr->client.clientIC);
        printf("Phone Number: %s\n", currentptr->client.phoneNum);
        printf("Car Plate: %s\n", currentptr->car.carplate);
        printf("Car Model: %s\n", currentptr->car.carmodel);
        printf("Car Make: %s\n", currentptr->car.carmake);
        printf("Car Price: %.2f\n", currentptr->car.carprice);
        fflush(stdin);
        printf("Car Mileage: %d\n", currentptr->car.carmileage);
        fflush(stdin);}

    else{
      printf("\nCar Plate is not in the list!!! ");
      }
        }
}

//convert linked list to array
void linkedlist_to_array(struct Asset *head, struct ArrayStore *store)
{
    int count = 0;
    int i;
    struct Asset *current = head;

    // Count the number of nodes in the linked list
    while (current != NULL)
    {
        count++;
        current = current->ptrnext;
    }

    // Allocate memory for the array of structures in ArrayStore
    store->array = (struct Asset *)malloc(count * sizeof(struct
Asset));
    if (store->array == NULL)
    {
        printf("Memory allocation error\n");
        exit(EXIT_FAILURE);
    }

    // Transfer data from the linked list to the ArrayStore
    current = head;
    for (i = 0; i < count; i++)
    {
        store->array[i] = *current;
        current = current->ptrnext;
    }

    // Set the size in ArrayStore
    store->size = count;
}

//bubble sorting by price
void bubble_sort_price(struct ArrayStore *store)
{
    int i;
    int originalSize = store->size;
```

```
    while (store->size > 1)
    {
        store->size--;

        for (i = 0; i < store->size; i++)
        {
            if (store->array[i].car.carprice > store->array[i +
1].car.carprice)
            {
                // Swap entire structures
                struct Asset temp;
                temp = store->array[i];
                store->array[i] = store->array[i + 1];
                store->array[i + 1] = temp;
            }
        }
    }
    store->size = originalSize;
}

//bubble sorting by mileage
void bubble_sort_mileage(struct ArrayStore *store)
{
    int i;
    int originalSize = store->size;

    while (store->size > 1)
    {
        store->size--;

        for (i = 0; i < store->size; i++)
        {
            if (store->array[i].car.carmileage > store->array[i +
1].car.carmileage)
            {
                // Swap entire structures
                struct Asset temp;
                temp = store->array[i];
                store->array[i] = store->array[i + 1];
                store->array[i + 1] = temp;
            }
        }
    }
    store->size = originalSize;
}


void display_array(struct ArrayStore *store)
{
      int i;
    if (store->size == 0)
    {
        printf("\nEmpty array\n");
        return;
    }


    // Display the array contents
```

```
    for (i = 0; i < store->size; i++)
    {
        printf("\n%d\t%-12s\t%-12s\t%-12s\t%-10s\t%-10s\t%-
10s\tRM%.2f\t%d km\n",
               i + 1,
store->array[i].client.clientname,store->array[i].client.clientIC,store
->array[i].client.phoneNum, store->array[i].car.carplate,
               store->array[i].car.carmodel,
               store->array[i].car.carmake,
store->array[i].car.carprice, store->array[i].car.carmileage);
    }
}

void list_sort()
{
    if (headptr==(struct Asset*)NULL) //empty list
    {
        printf("\nEmpty list");
        return;
    }


    currentptr=headptr;

    struct ArrayStore finalarray;
    linkedlist_to_array(headptr, &finalarray);

    int input;
    printf("Sort by:\n"
           "1 - Car Price\n"
           "2 - Car Mileage\n"
           "Enter choice: ");
    scanf("%d",&input);
    fflush(stdin);

    if(input == 1)
    {
        bubble_sort_price(&finalarray);
        printf("The records have been sorted by car price.");
    }
    else if (input == 2)
    {
        bubble_sort_mileage(&finalarray);
        printf("The records have been sorted by car mileage.");
    }
    else
    {
        printf("Invalid input.");
        return;

    }


    printf("\n");
    printf("\nNo.   Client Name\tClient IC\tPhone Number\tCar
Plate\tCar Model\tCar Make\tCar Price\tCar Mileage\n");

    display_array(&finalarray);
```

```
    free(finalarray.array);
}

//register repair and maintenance
void repairRegister(){
      char repairRegister[20];
      char dequeueRegister[20];
      int i;
      int count=0;
      char repairDate [20];
      char repairDetails [30];
      char ans;

    if (headptr==NULL){
        printf("\n\nThe list is empty!!!\n");
            }

    else{
    label1:{
    fflush(stdin);
      printf("\nEnter car plate to register repair and maintenance: ");
      scanf("%s",&repairRegister);
      printf("\nEnter date of service (in DD-MM-YYYY): ");
      scanf("%s",&repairDate);
      printf("\nEnter repair details: ");
      scanf("%s",&repairDetails);


    currentptr=headptr;

      while(currentptr->ptrnext!=NULL){

            if (strcmp(currentptr->car.carplate, repairRegister)==0){
                  printf("The repair and maintenance registration is
successful!\n");

      strcpy(repair[i].rpClientname,currentptr->client.clientname);
                  strcpy(repair[i].rpCarplate,currentptr->car.carplate);
                  strcpy(repair[i].rpCarmake,currentptr->car.carmake);
                  strcpy(repair[i].rpCarmodel,currentptr->car.carmodel);
                  strcpy(repair[i].Date_of_Service,repairDate);
                  strcpy(repair[i].Repair_Details,repairDetails);
                  fflush(stdin);
                  i=i+1;
                  count=count+1;
                  goto label2;
                  }
            else{
            currentptr=currentptr->ptrnext;
                  }
            }

        if (strcmp(currentptr->car.carplate, repairRegister)==0){
                  printf("The repair and maintenance registration is
successful!\n");

      strcpy(repair[i].rpClientname,currentptr->client.clientname);
```

```
                strcpy(repair[i].rpCarplate,currentptr->car.carplate);
                strcpy(repair[i].rpCarmake,currentptr->car.carmake);
                strcpy(repair[i].rpCarmodel,currentptr->car.carmodel);
                strcpy(repair[i].Date_of_Service,repairDate);
                strcpy(repair[i].Repair_Details,repairDetails);
                fflush(stdin);
                i=i+1;
                count=count+1;
                goto label2;
                }
        else{
            currentptr=currentptr->ptrnext;}


      }
      if(strcmp(currentptr->car.carplate, repairRegister)!=0){
            printf("\nCar Plate is not in the list!!! ");
      }
      }
            label2:{
                  fflush(stdin);
            printf("\nDo you want to register repair and maintenance for
another car? (Y/N):");
            scanf("%c",&ans);
            switch(ans){
                  case 'Y':
                        case 'y':goto label1;break;
                  case 'N':
                        case 'n':goto label3;break;
                        default: printf("Invalid value!");goto
label2;break;
                  }
            }

      label3:{
            fflush(stdin);
      printf("\nDo you want to dequeue repair and maintenance from the
list of car? (Y/N):");
      scanf("%c",&ans);
      switch(ans){
            case 'Y':
                  case 'y':count=dequeueRepair(count);goto label2;break;
            case 'N':
                  case 'n':goto label4;break;
                  default: printf("Invalid value!");goto label3;break;
      }
      }

      label4:{
      printf("\n");
    printf("\nNo.\tClient Name\tCar Plate\tCar Make\tCar Model\tDate of
Service\tRepair Details\n");
      for (i=0;i<count;i++){
            printf("%d.\t",(i+1));
            printf("%s\t\t", repair[i].rpClientname);
        printf("%s\t\t", repair[i].rpCarplate);
        printf("%s\t\t", repair[i].rpCarmake);
        printf("%s\t\t", repair[i].rpCarmodel);
```

```
        printf("%s\t", repair[i].Date_of_Service);
        printf("%s\n", repair[i].Repair_Details);
      }
      fflush(stdin);
      return;
      }
}



struct driver getinsurance(){

    struct driver i;

    fflush(stdin);
    printf("\nEnter Insurance Provider: ");
    scanf(" %[^\n]s", i.insurance_info.insuranceProvider);

    printf("Enter Policy Number: ");
    scanf("%s", &i.insurance_info.policyNumber);

    printf("Enter Coverage Amount: ");
    scanf("%f", &i.insurance_info.coverageAmount);


    return i;

}

struct driver getDriverInfo() {

    struct driver d;

    fflush(stdin);
    printf("Enter Driver Name: ");
    scanf(" %[^\n]s", d.driver_info.driverName);

    printf("Enter Driver License: ");
    scanf("%s", &d.driver_info.driverLicense);

    printf("Enter Driver Age: ");
    scanf("%d", &d.driver_info.age);

    return d;
}

void Info(){

    struct driver insurance = getinsurance();
    struct driver driverInfo = getDriverInfo();

    fflush(stdin);
    printf("\nDetails:\n");
    display(insurance, driverInfo);
}
```

```
void display(struct driver insurance, struct driver driverInfo) {
    fflush(stdin);
    printf("\nInsurance Provider: %s\n",
insurance.insurance_info.insuranceProvider);
    printf("Policy Number: %s\n",
insurance.insurance_info.policyNumber);
    printf("Coverage Amount: RM%.2f\n",
insurance.insurance_info.coverageAmount);
    printf("Driver Name: %s\n", driverInfo.driver_info.driverName);
    printf("Driver License: %s\n",
driverInfo.driver_info.driverLicense);
    printf("Driver Age: %d\n", driverInfo.driver_info.age);
}
```

TASK DISTRIBUTION

Name	Student ID	Task
Chong Jin Jye	SD22048	- Build Car, Client, Asset and ArrayStore structure - Develop the Enqueue operation to insert the information using linked list. - Implement sorting using bubble sort and add options for selecting the category to be sorted. - Implement the delete at middle for selected car plate by using linked list - Changing linked list to array - Responsible for overall project coordination
Lim Jon Vince	SD22015	- Build the RepairCar structure - Develop the Enqueue and Dequeue for part of repair and maintenance - Implement the goto statement in repair and maintenance - Identify any bugs or issues and help to fix it
Low Ann Gie	SD22059	- Write for case study - Prepare the initial information data - Implement the Push operation in update part and Pop operation in delete part using linked list - Develop the searching using Linear Sequential Search
Ang Mei Ying	SD22061	

REFERENCES

1. *Connect your home with Samsung SmartThings: Samsung United Kingdom*. Samsung us. (2024, January 12). https://www.samsung.com/us/smartthings/

2. Gerard. (2021, June 21). *Advantages and disadvantages of the smartthings hub*. Smart Homes School - home automation training courses & books. https://smarthomesschool.com/advantages-and-disadvantages-of-the-smartthings-hub/

3. India, S. (2023, August 14). *Creating efficient food supply chains with IOT-based transportation management software*. SAP India News Center. https://news.sap.com/india/2021/11/efficient-food-supply-chain/

4. *Malaysia's largest marketplace - buy & sell your new and preloved items*. Mudah.my - Malaysia's largest marketplace. (n.d.). https://www.mudah.my/

5. Yuen, M.-C., Choi, Y.-H., Ng, T.-H., Poon, C.-T., & Fung, K.-C. (2020). Development of IOT based fresh food delivery tracking system using GPS. *Fuzzy Systems and Data Mining VI*. https://doi.org/10.3233/faia200718