# Restaurant Business Database

**Slot :** L53 + L54

**Course :** Data Structures and Algorithms

**Course Code :** CSE2003

**Names :**

1) Kandati Puneeth : 17BEC0254

2) Pilla Adivishnu Murthy : 18BEC0300

3) Tunuguntla Sai Venkata Saketh : 19BCE0579

4) K Shashanka Reddy : 19BCE0941

5) Battula Shanmukh Reddy : 19BCE0954

## ABSTRACT :

Many Restaurants have been facing issues of maintaining their database properly. This project provides a novel method of saving their data without any difficulty. The

administrator can add new items, employees, orders, sales record  and also can review all of them; This project also allows them to delete a particular item, profile, orders and their sales record. This app allows to search among the records. The process is made simpler by developing an android application that can be customized according to their needs. The app allows the administrator to monitor the progress his restaurant from anywhere and anytime.

The authority of the company can use this software to monitor the expenditure of his items, salaries of the workers, the number of workers employed, orders and his sales records too.

## **Aim and Objective :**

To develop a Restaurant Business Database software and android app that can monitor all his needs more easily.

## **Scope/Applicability :**

Small Restaurants can keep track of their menu items details, employee details, Orders and Records without any difficulty. This software and app can be handy to many upcoming Restaurants.

## **Introduction** :

In this project of ours, we will see how we will store the information. Here, you can create a new account, update the information of an existing account, view and manage transactions, check the details of an existing account, remove an existing account, and view the items, employee, orders and records list.
This will help to keep a check on all the requirements and will also help to maintain a proper record and data of every bit of information.

## **Literature Review** :

- In 1, 2 and 3 are the guide to make restaurant database management.
- In 4, 5 and 6 the author says the efficient methods and ways to implement restaurant.

- In 8, 10, 11 the author describes a novel method for creating item menu and employee database.

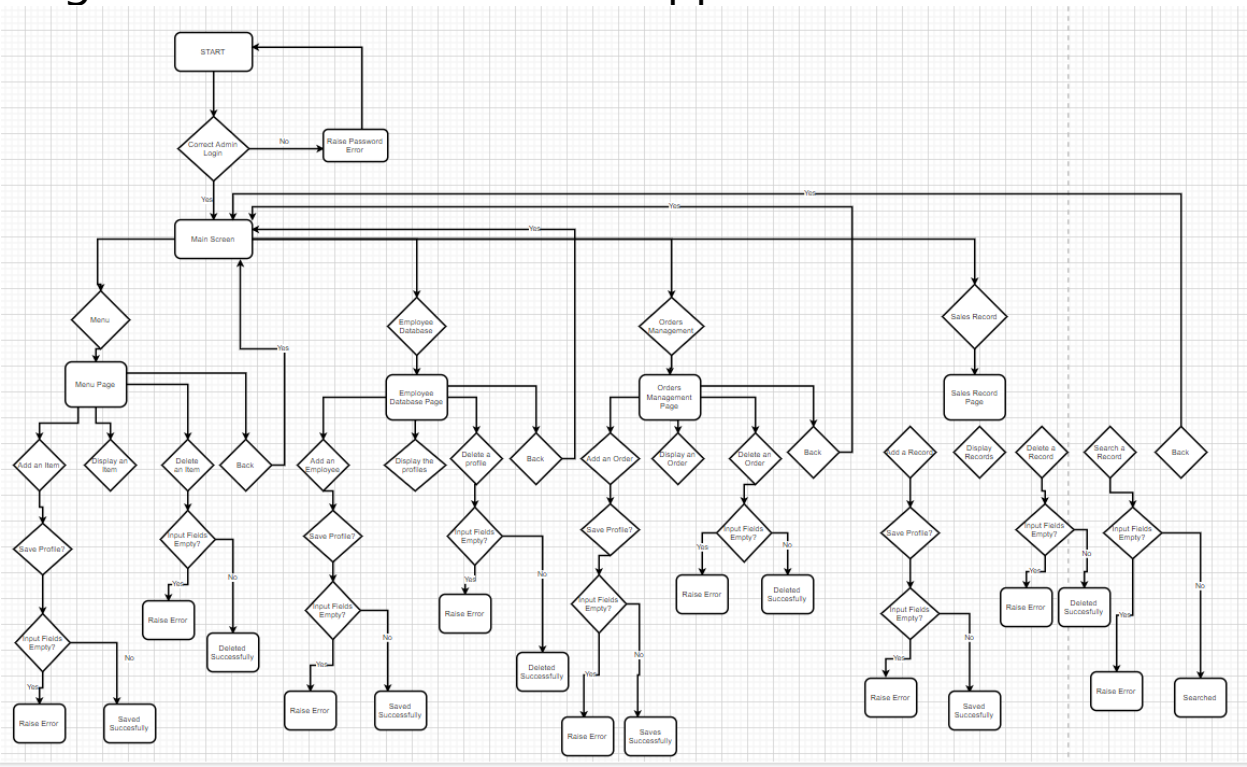
## **Implementation** :

The Restaurant Database Management system is implemented in two ways:
1. Computer Software.

2. Android app.

## **Architecture Diagram** :

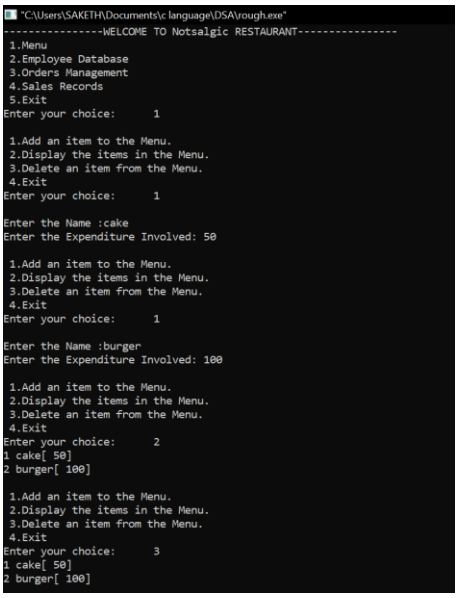
The following flow chart illustrates the architecture diagram of the android based app.

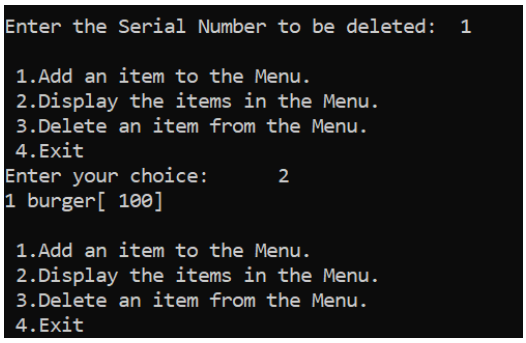


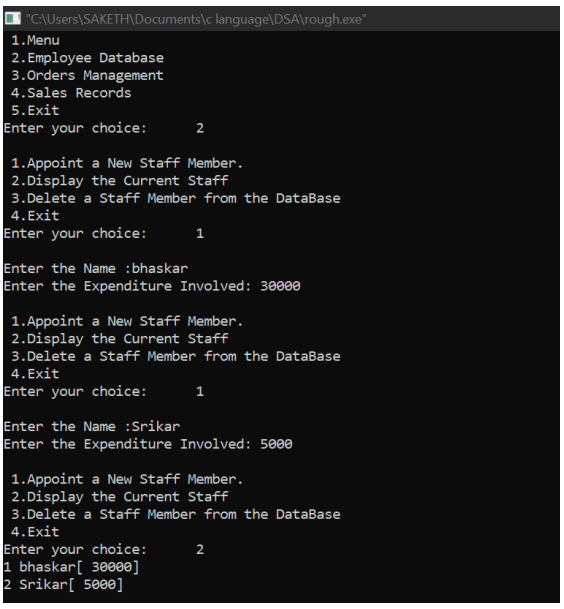
# **Module Description for Software :**

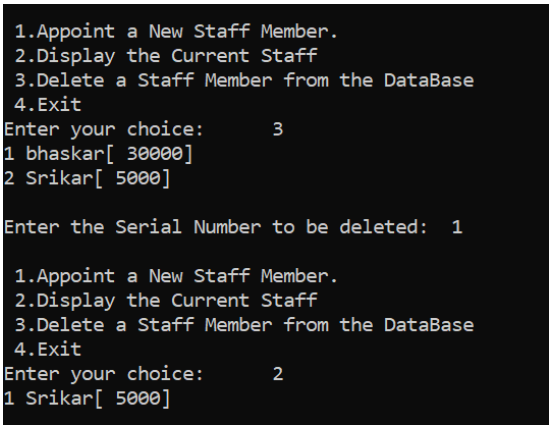
Our program performs following operations and functions

a) Item Menu and Employee Data Base



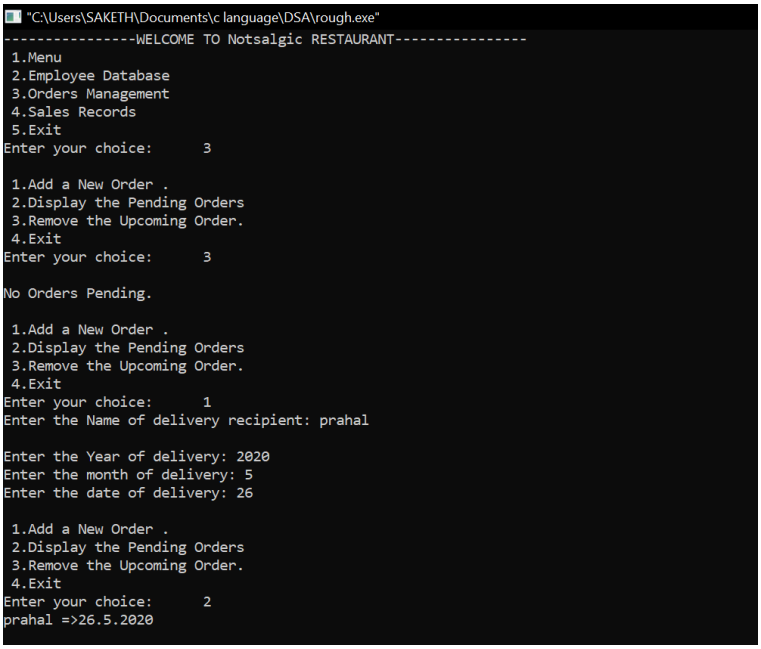




In this we have 3 functions i.e

1) Add an item : Where we can add an item
2) Display : Displays the added items
3) Delete : Deletes the selected item

b) Orders Management

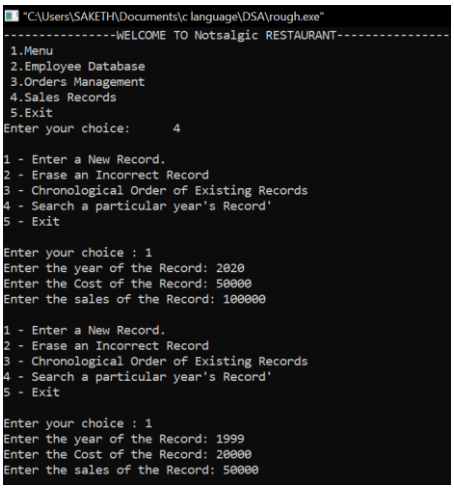


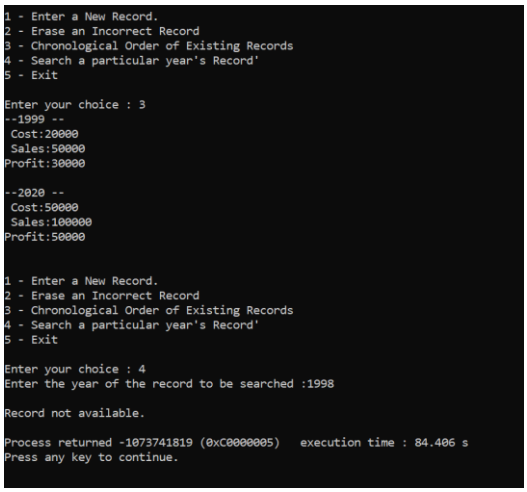

Queue data structure is used to implement Orders Management as it follows FIFO principle (First Input First Output). So the person who orders first will be served first.

The operation are :

1) Taking a new order
2) Displaying the orders
3) Removing the upcoming orders.

c) Sales Record



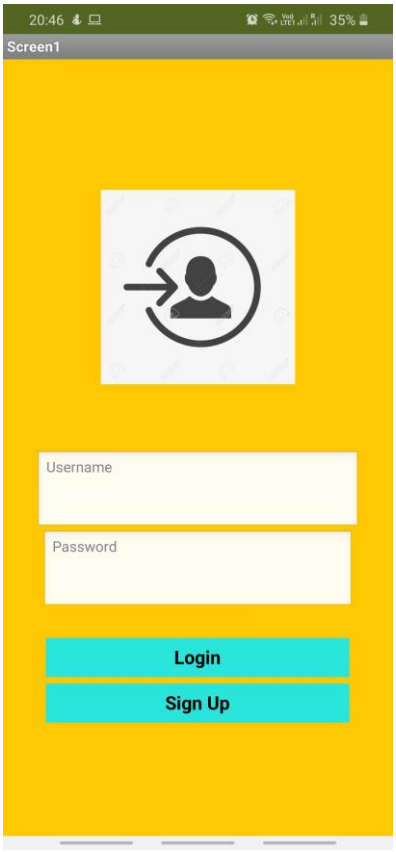



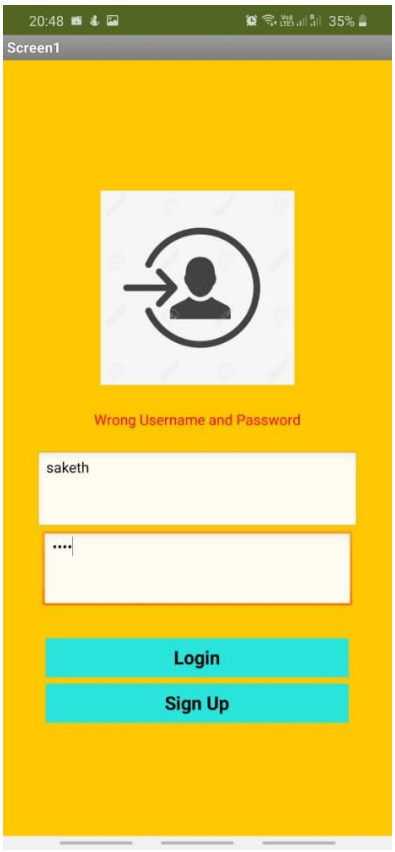

Binary Search Tree Data Structure Data Structure is used to implement Sales Record.
The operation are :

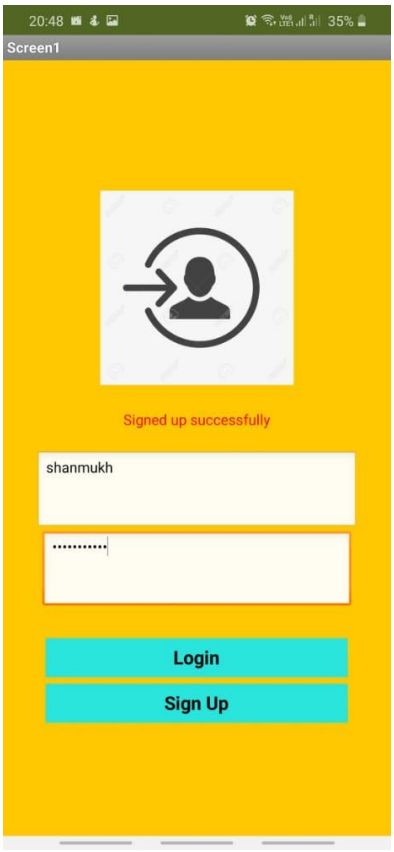
1) Enter a New Record
2) Erase a Record
3) Chronological Record
4) Search for a Record

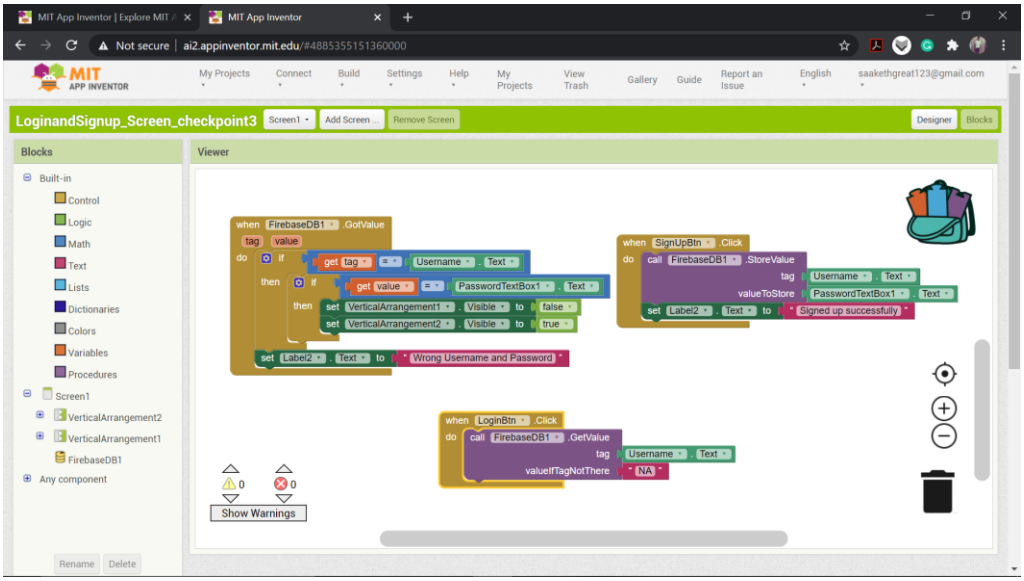
## **Module Description for App :**

1) Login Page :

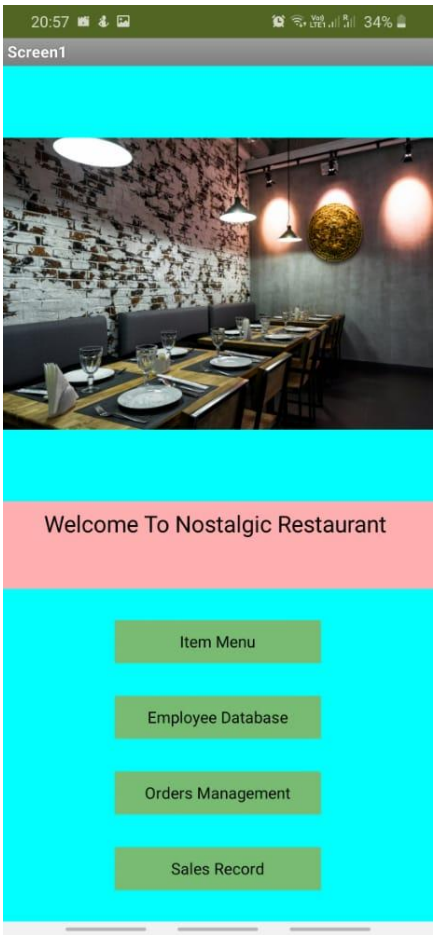


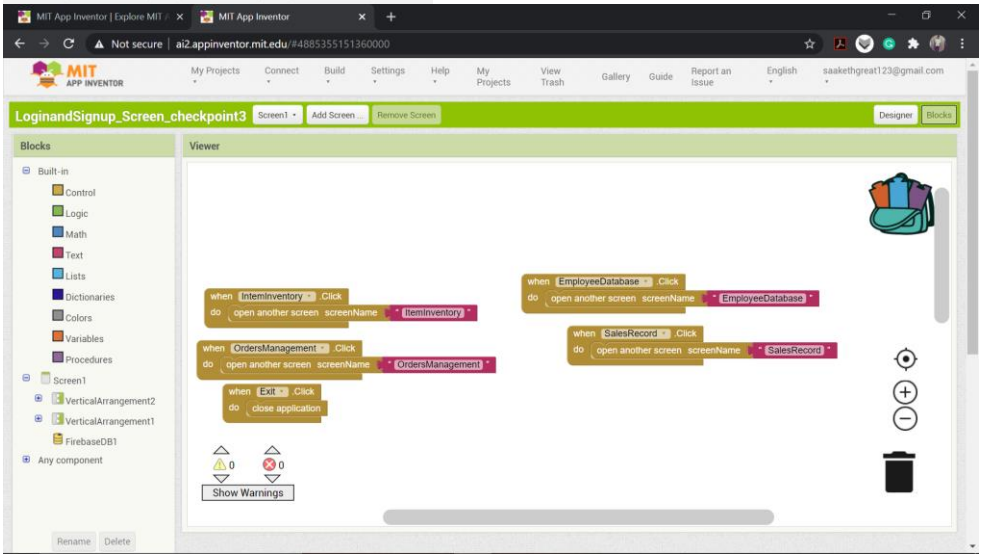



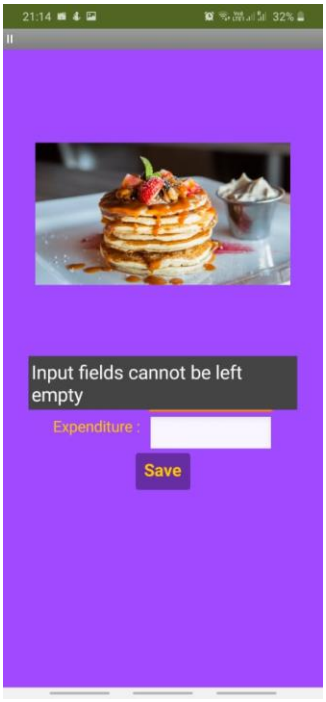

After we logged in Succesfully :

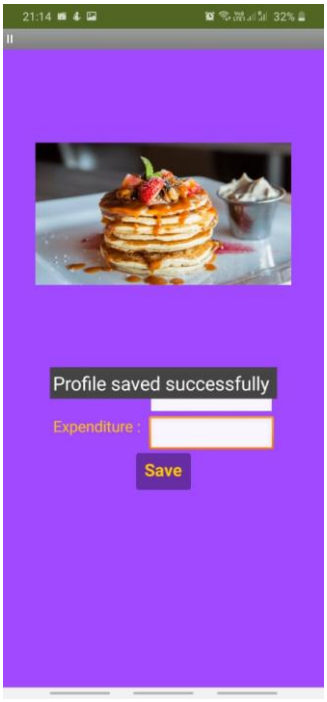


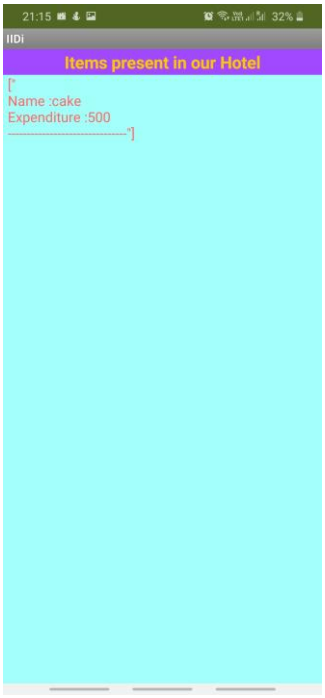



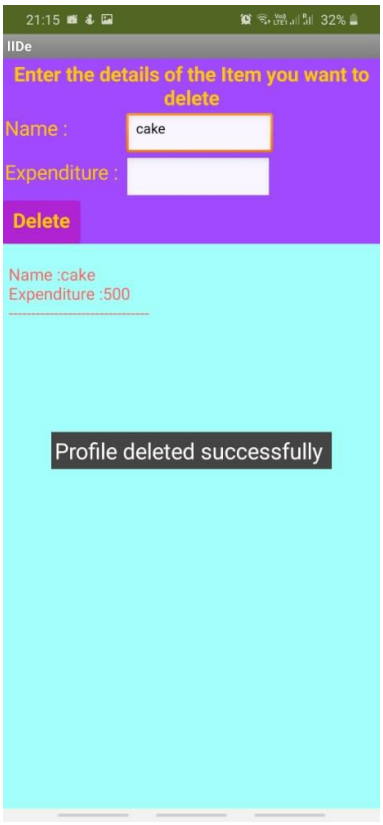

Item Menu :

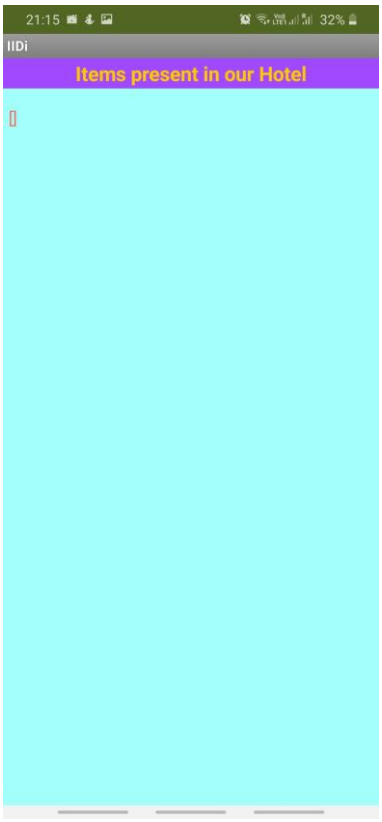
Add, Display and Deleting an Item :



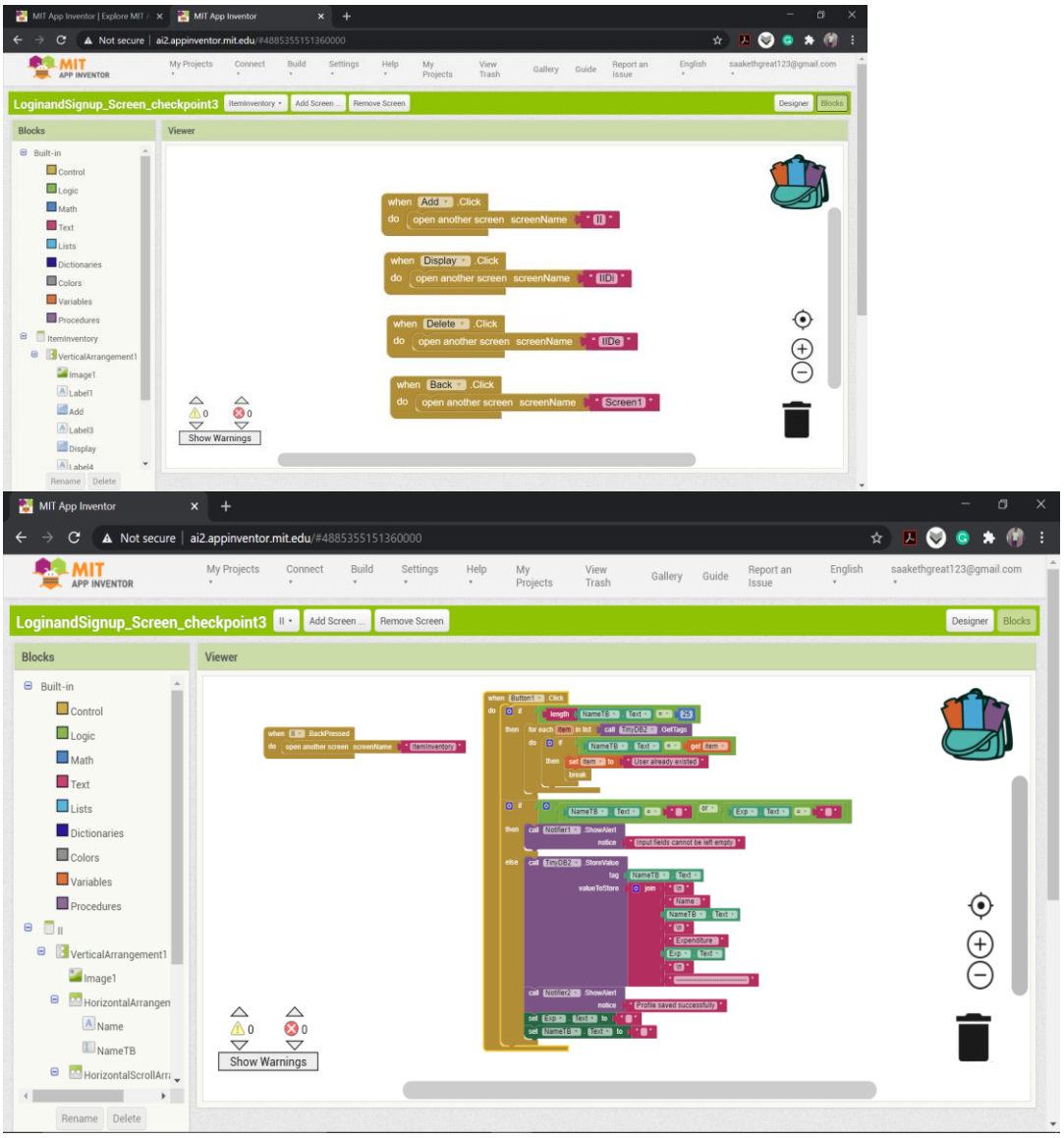



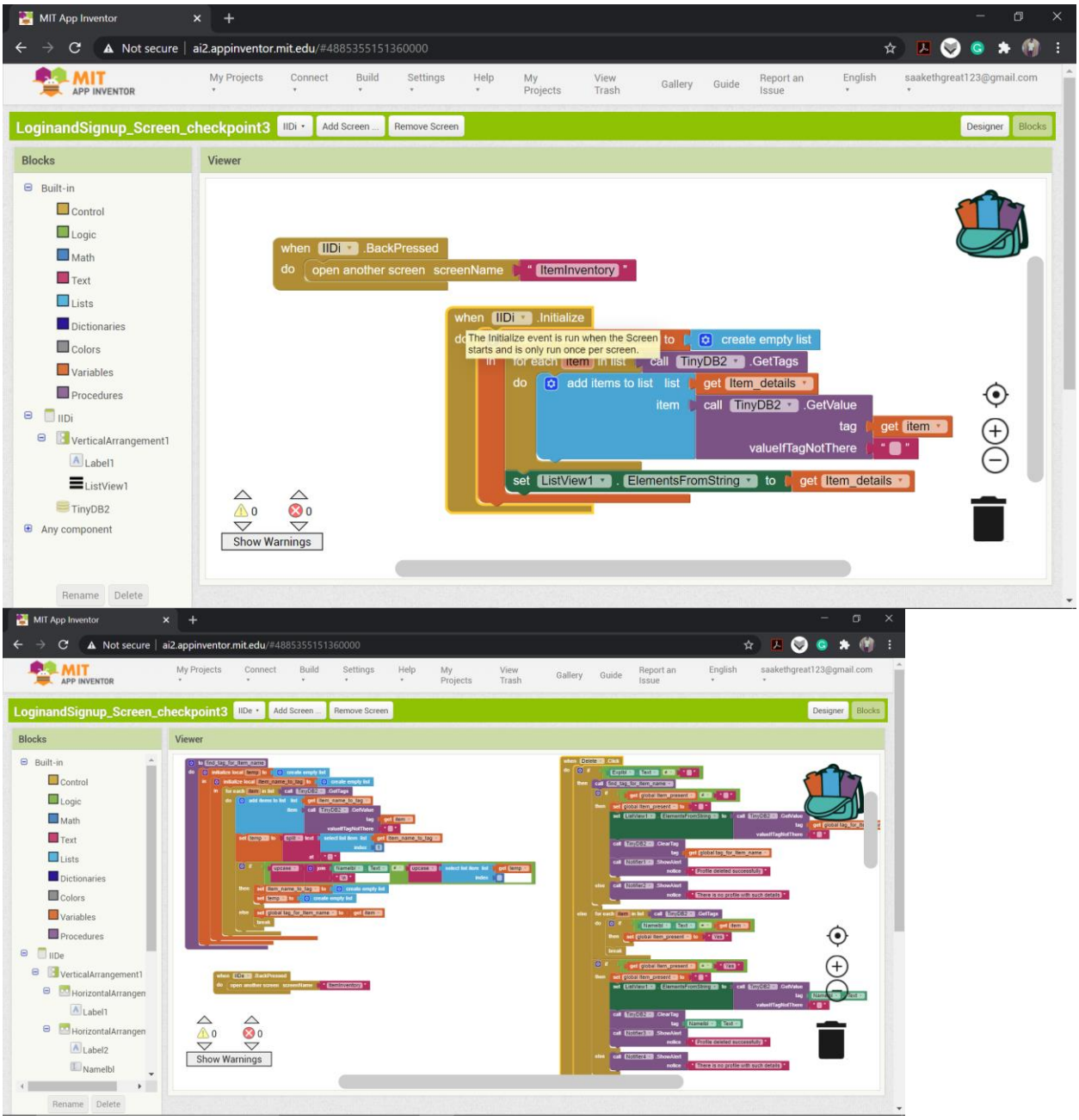



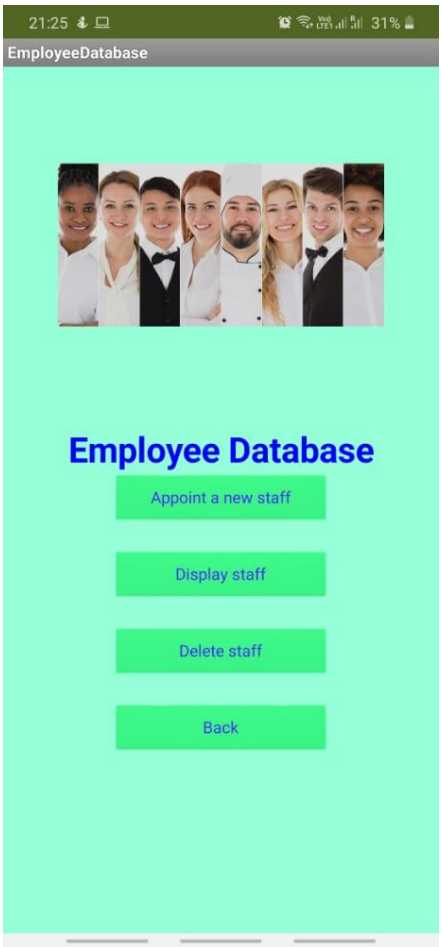



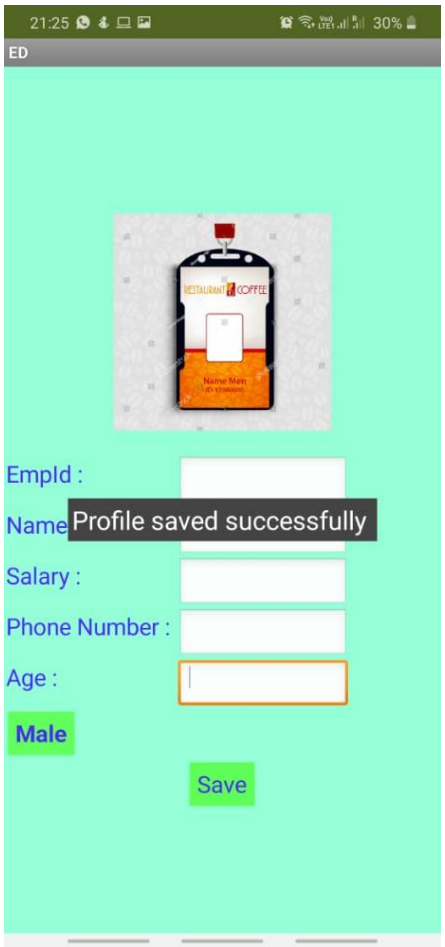

[After Deleting]


MIT App Inventor
Not secure | ai2.appinvertor.mit.edu/#4885355151360000
LoginandSignup_Screen_checkpoint3
Add Screen ...
Remove Screen
Designer
Blocks
Viewer
when Add .Click
do open another screen screenName "II"
when Display .Click
do open another screen screenName "IIDi"
when Delete .Click
do open another screen screenName "IIDe"
when Back .Click
do open another screen screenName "Screen1"
Show Warnings
Built-in
Control
Logic
Math
Text
Lists
Dictionaries
Colors
Variables
Procedures
Rename
Delete

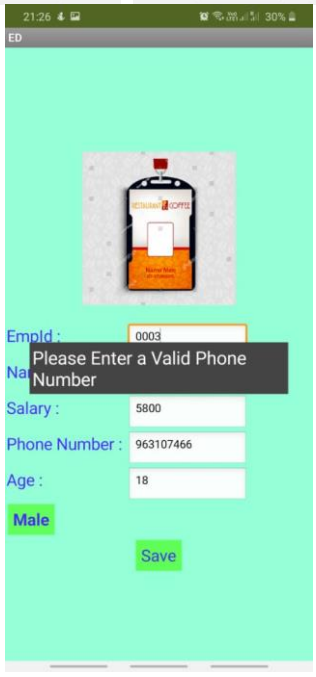



Employee Database :

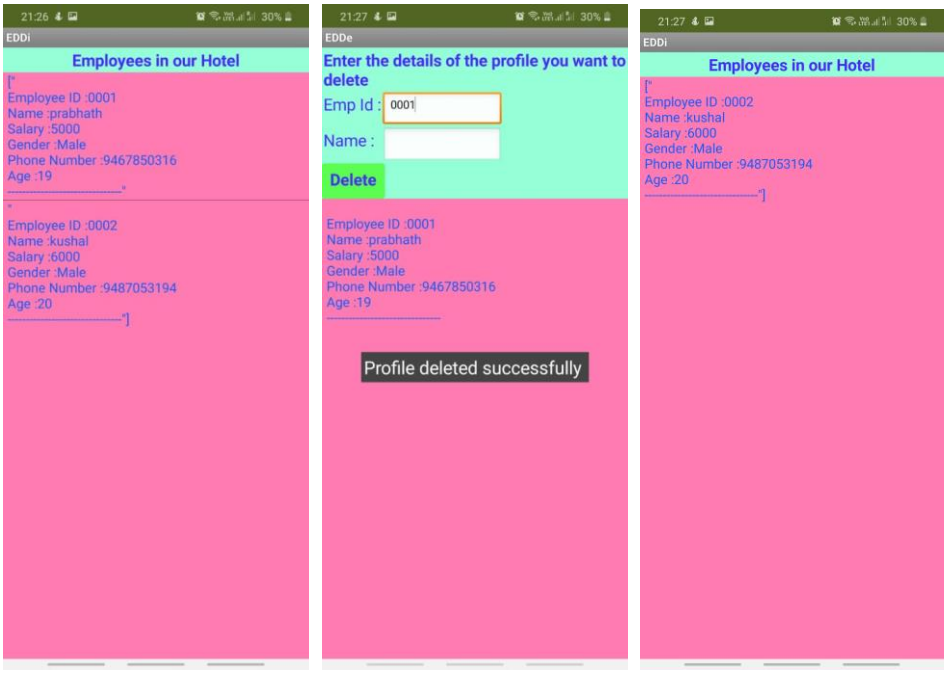

21:25
31%
EmployeeDatabase
Employee Database
Appoint a new staff
Display staff
Delete staff
Back


21:25
30%
ED
EmpId :
Name
Profile saved successfully
Salary :
Phone Number :
Age :
Male
Save

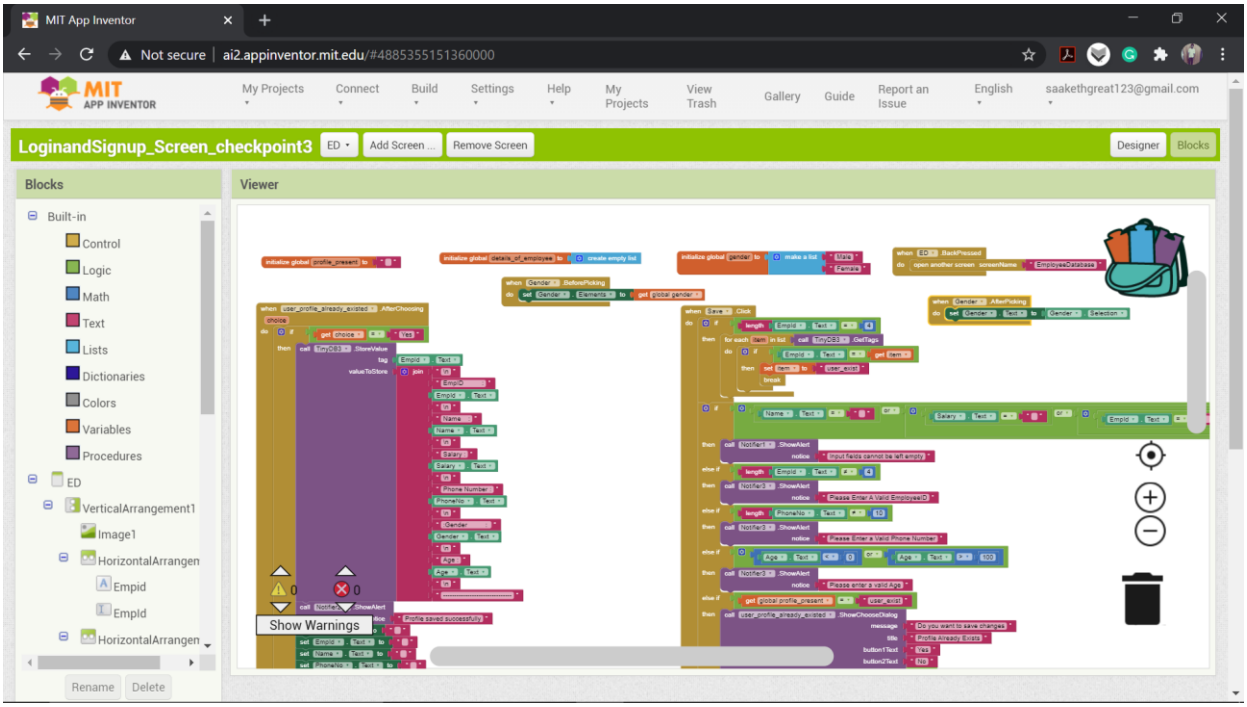

21:26
30%
ED
EmpId :
01
Input fields cannot be left empty
Salary :
Phone Number :
Age :
Male
Save

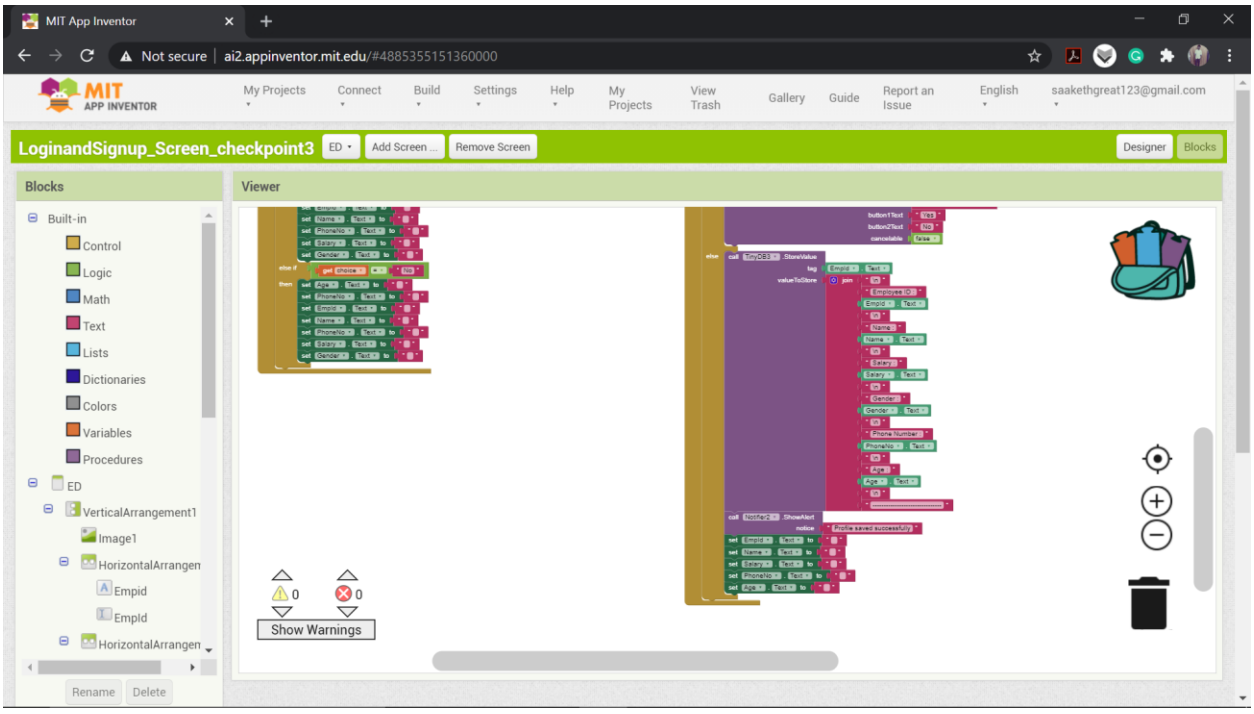

21:26
30%
ED
EmpId :
0003
Please Enter a Valid Phone Number
Salary :
5800
Phone Number :
963107466
Age :
18
Male
Save


21:26
30%
ED
EmpId :
003
Please Enter A Valid EmployeeID
Salary :
5800
Phone Number :
9631074666
Age :
18
Male
Save

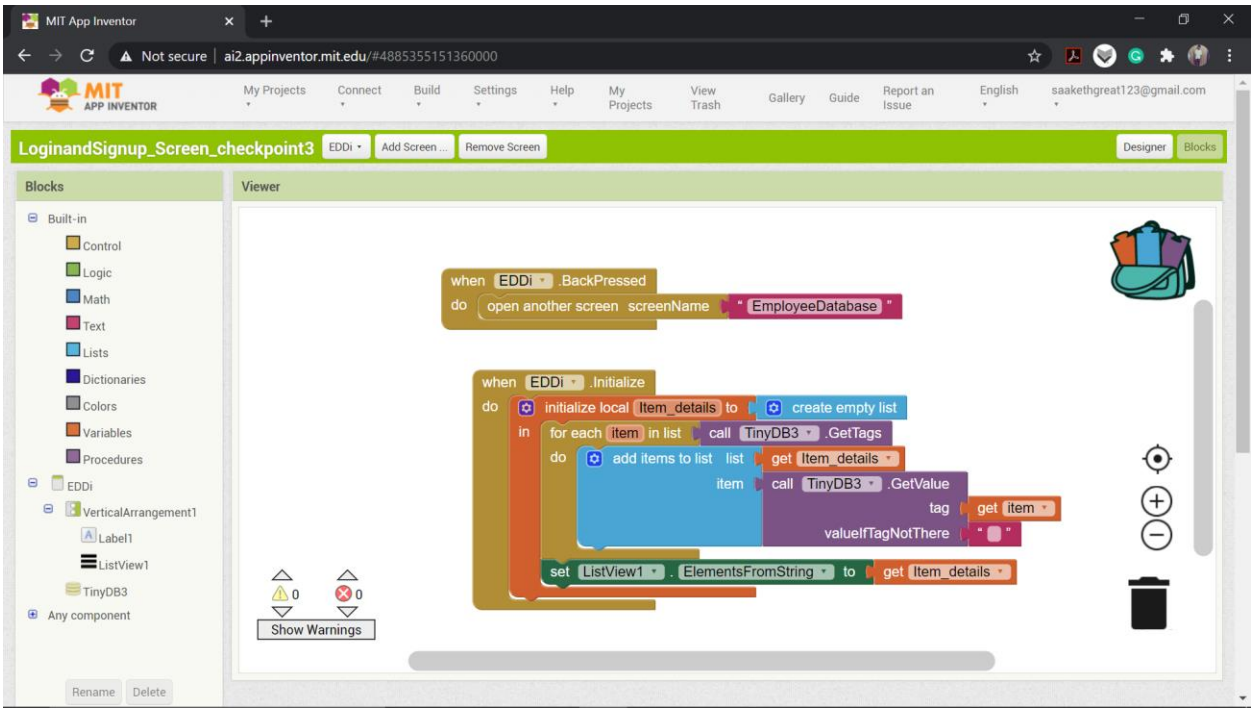

21:26
EDDi
Employees in our Hotel
["
Employee ID :0001
Name :prabhath
Salary :5000
Gender :Male
Phone Number :9467850316
Age :19
----------"
"
Employee ID :0002
Name :kushal
Salary :6000
Gender :Male
Phone Number :9487053194
Age :20
----------"]
21:27
EDDe
Enter the details of the profile you want to delete
Emp Id :
0001
Name :
Delete
Employee ID :0001
Name :prabhath
Salary :5000
Gender :Male
Phone Number :9467850316
Age :19
----------
Profile deleted successfully
21:27
EDDi
Employees in our Hotel
[
Employee ID :0002
Name :kushal
Salary :6000
Gender :Male
Phone Number :9487053194
Age :20
----------"]

For Adding :

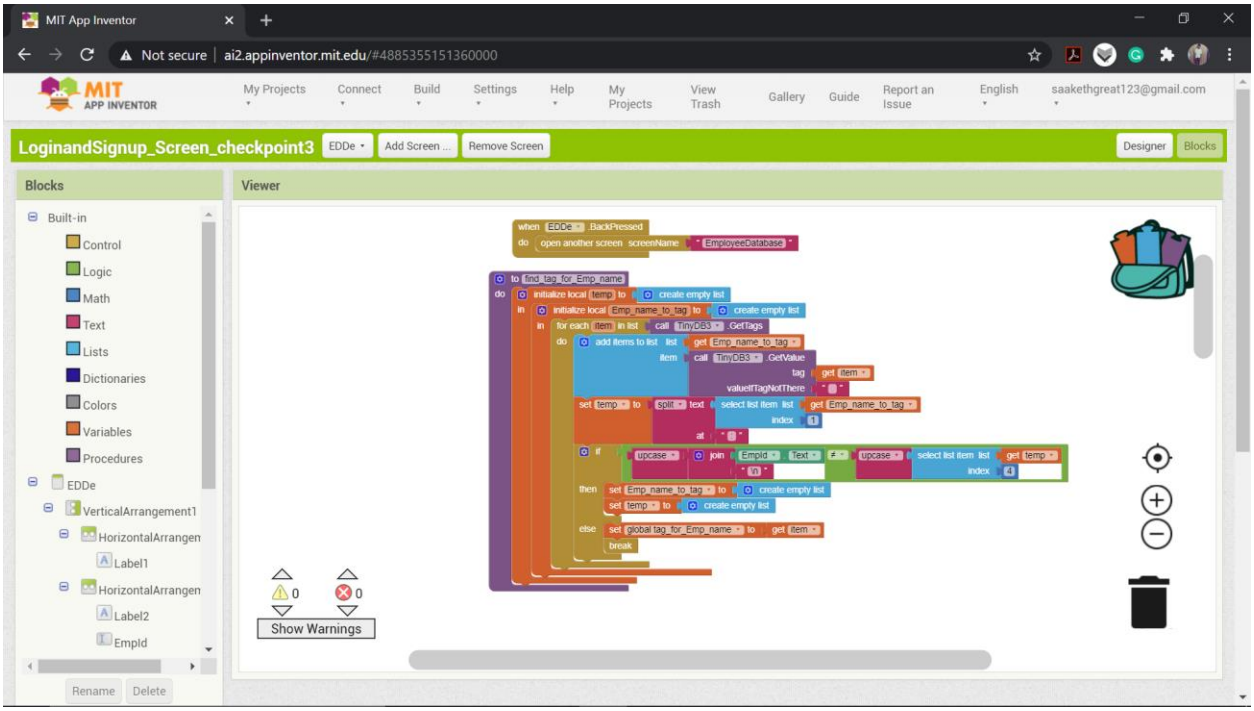


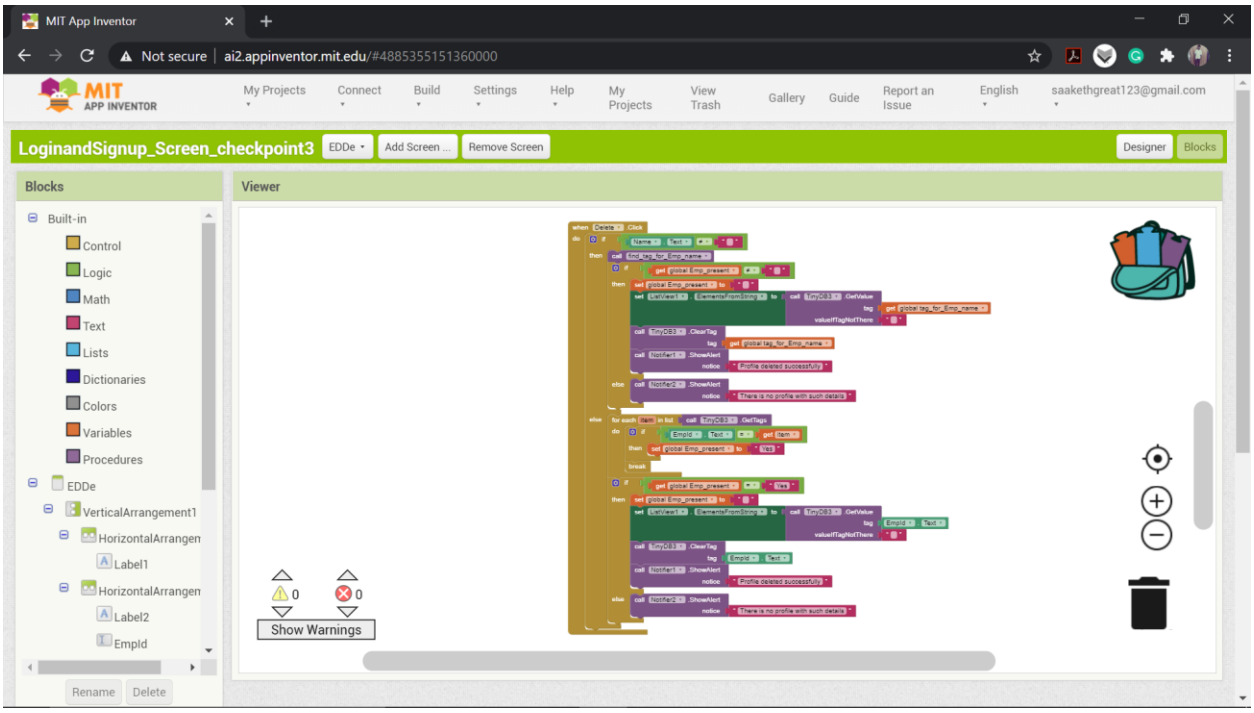




For Displaying :

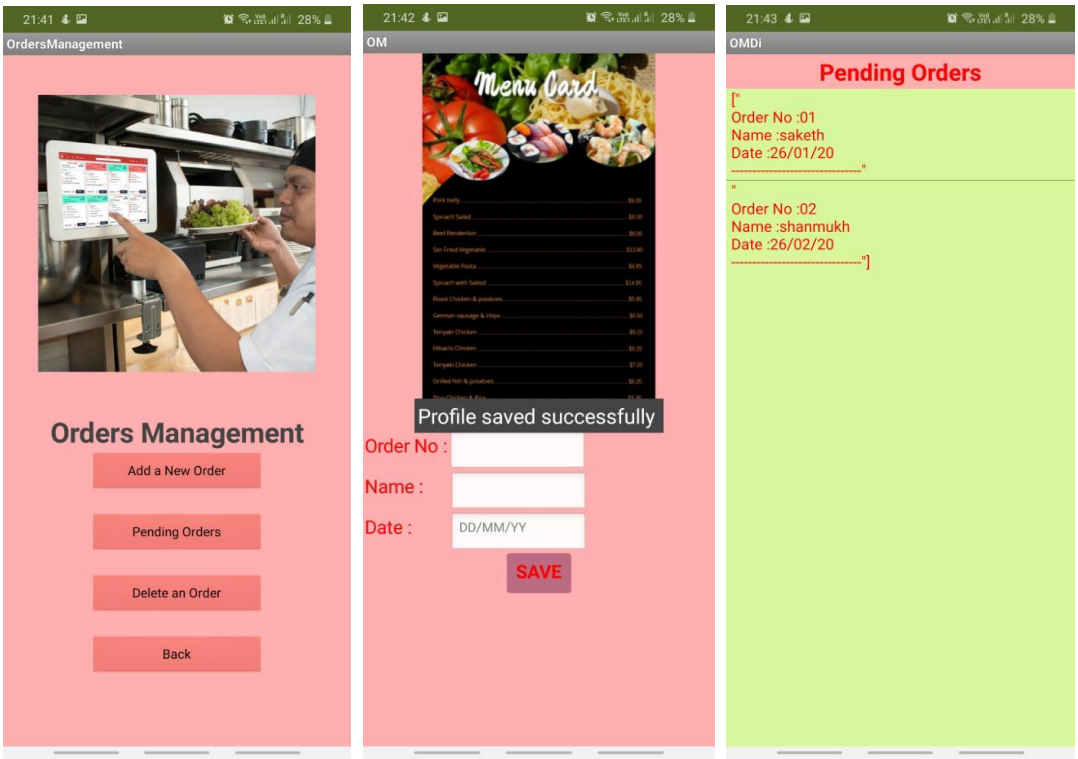



# For Deleting :

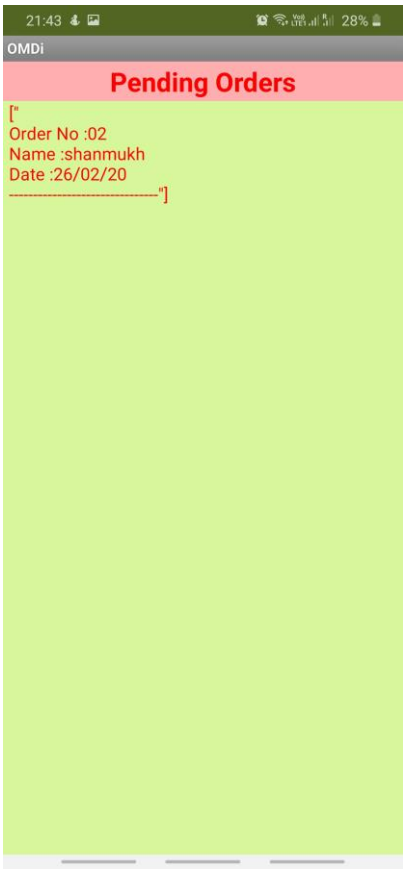




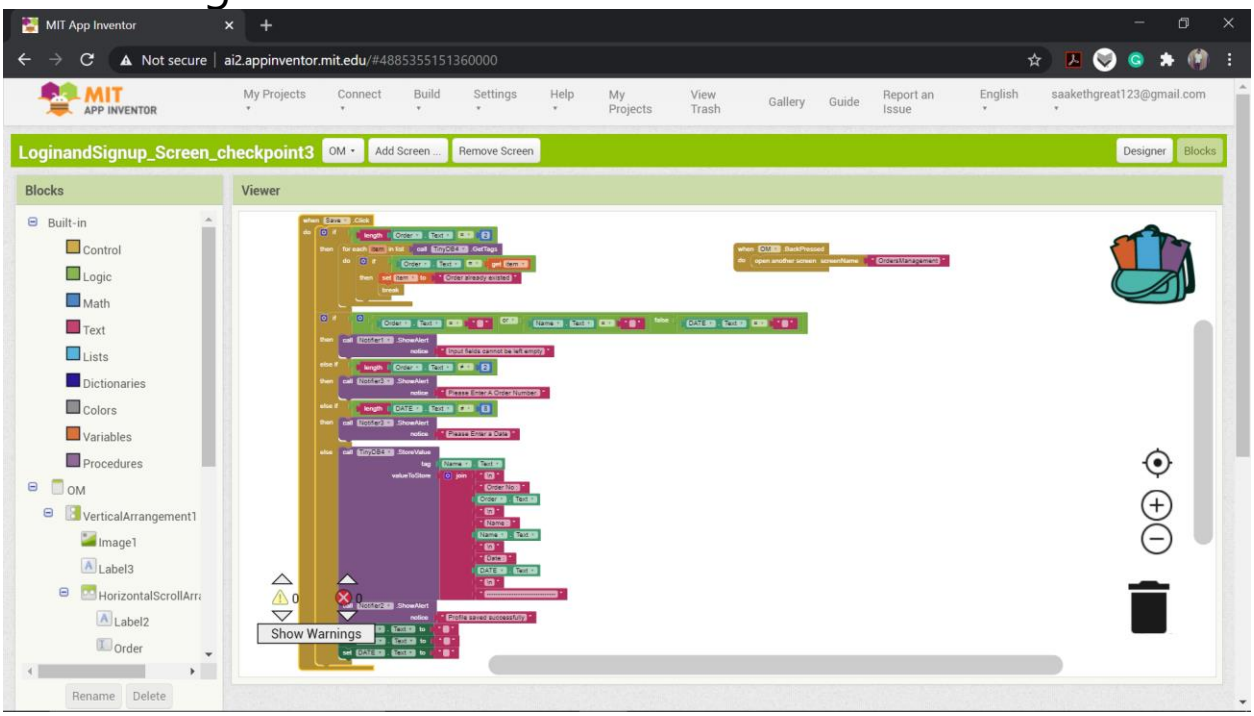

# Orders Management :



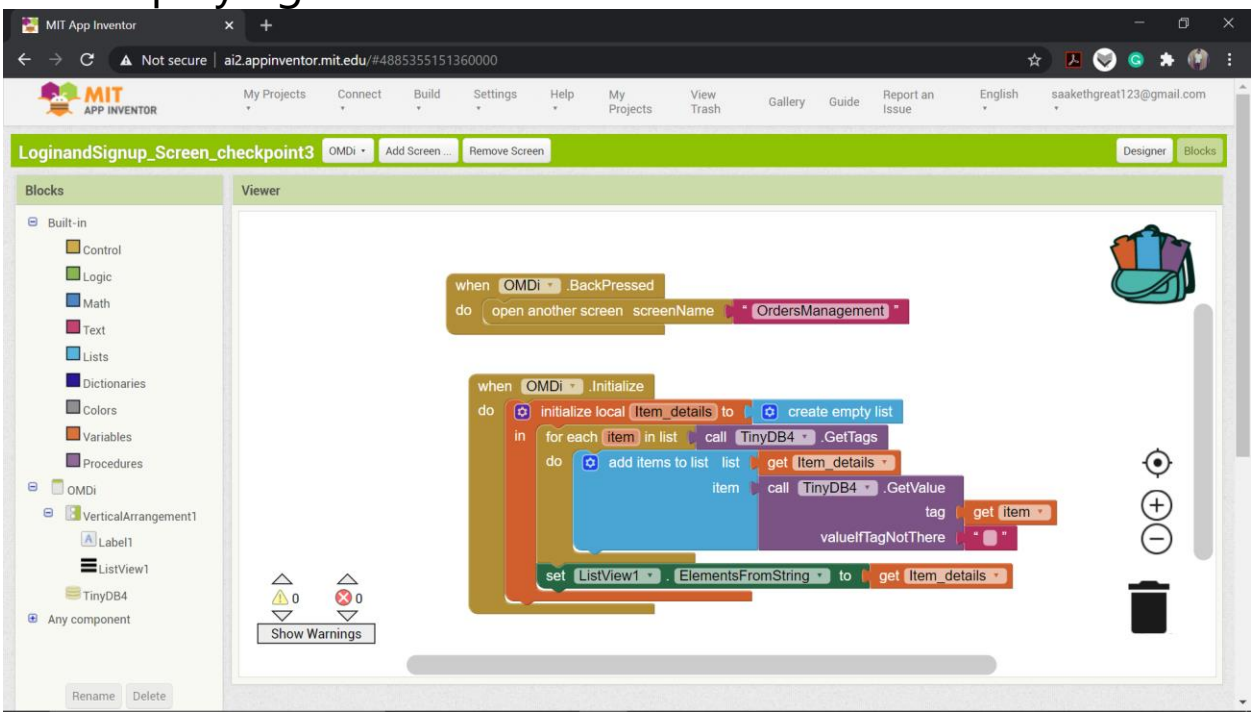

21:43
28%
OMDe
Enter the Details of Order you want to Delete
Name :
saketh
Order No :
Delete
Order No :01
Name :saketh
Date :26/01/20
Profile deleted successfully


21:43
28%
OMDi
Pending Orders
["
Order No :02
Name :shanmukh
Date :26/02/20
"]

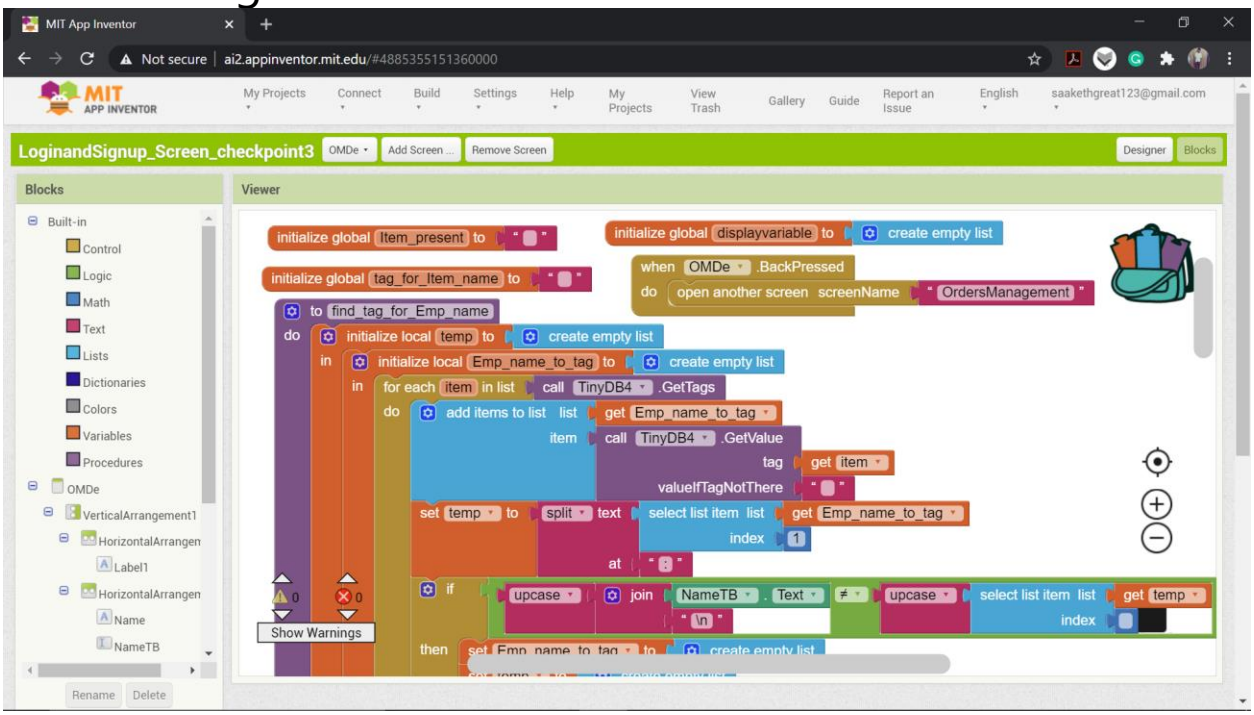
# For adding :

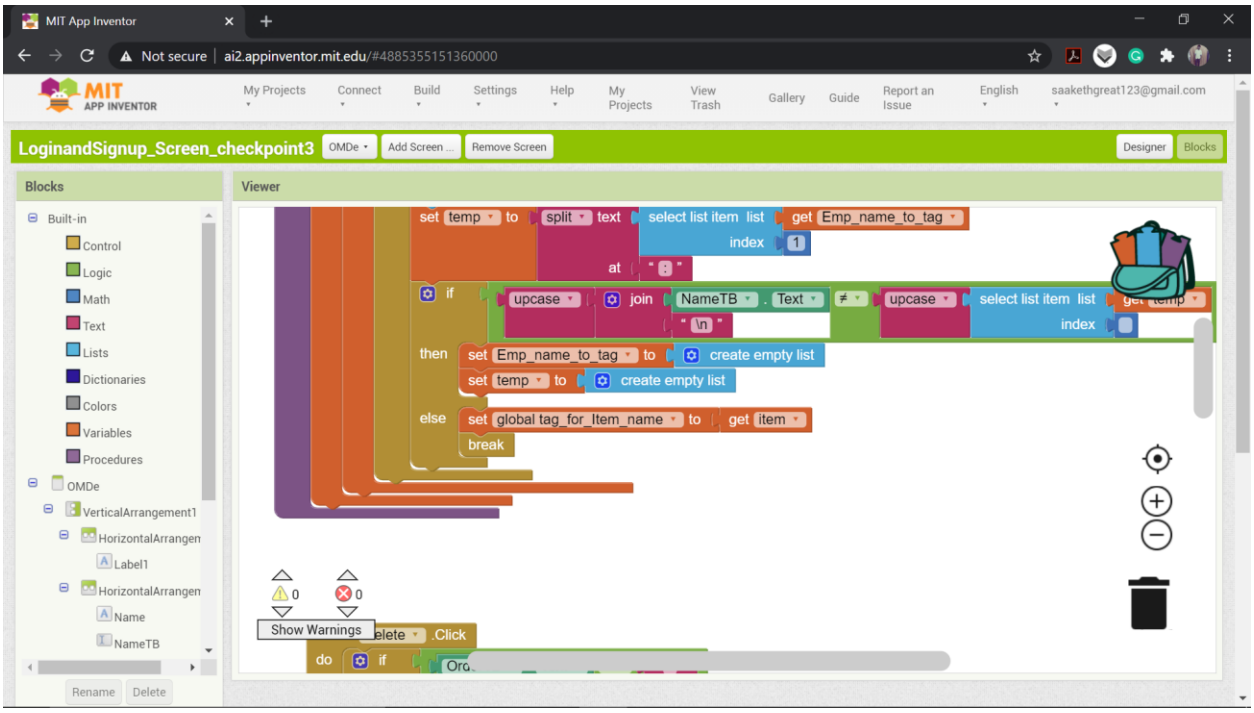

MIT App Inventor
Not secure
ai2.appinventor.mit.edu/#4885355151360000
MIT APP INVENTOR
My Projects
Connect
Build
Settings
Help
My Projects
View Trash
Gallery
Guide
Report an Issue
English
saakethgreat123@gmail.com
LoginandSignup_Screen_checkpoint3
OM
Add Screen ...
Remove Screen
Designer
Blocks
Blocks
Viewer
Built-in
Control
Logic
Math
Text
Lists
Dictionaries
Colors
Variables
Procedures
OM
VerticalArrangement1
Image1
Label3
HorizontalScrollArr
Label2
Order
Rename
Delete
Show Warnings

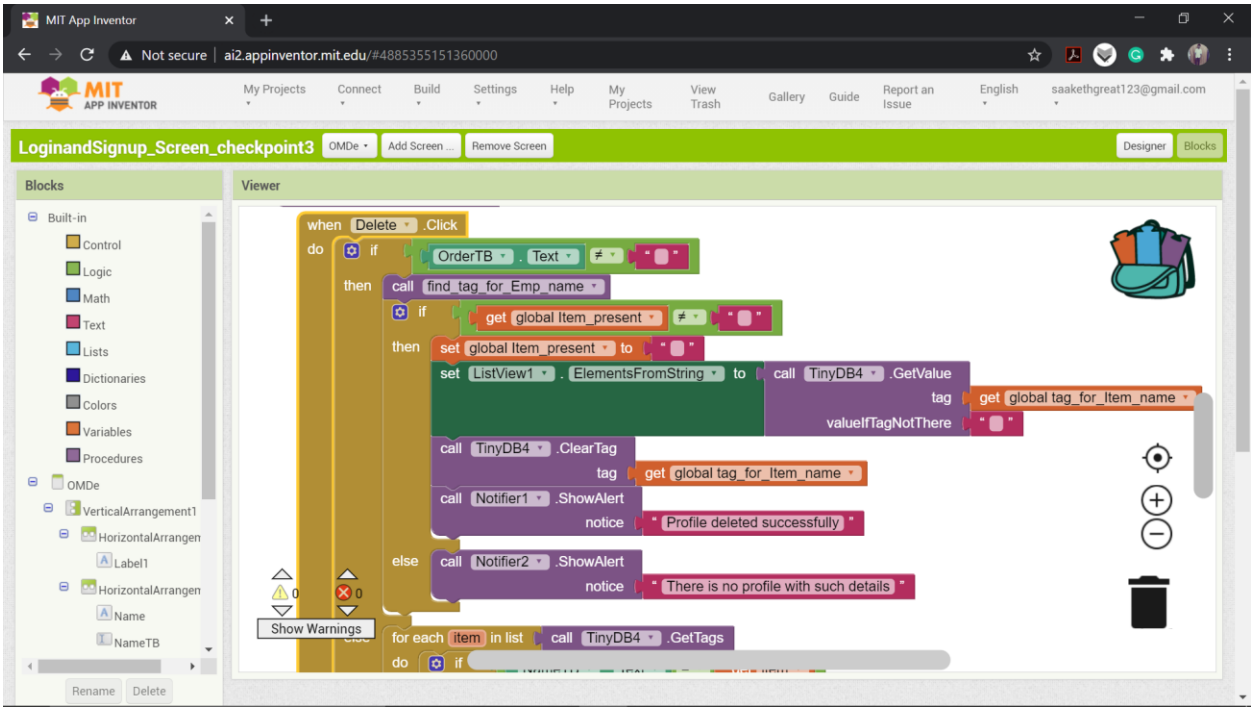
# For Displaying :

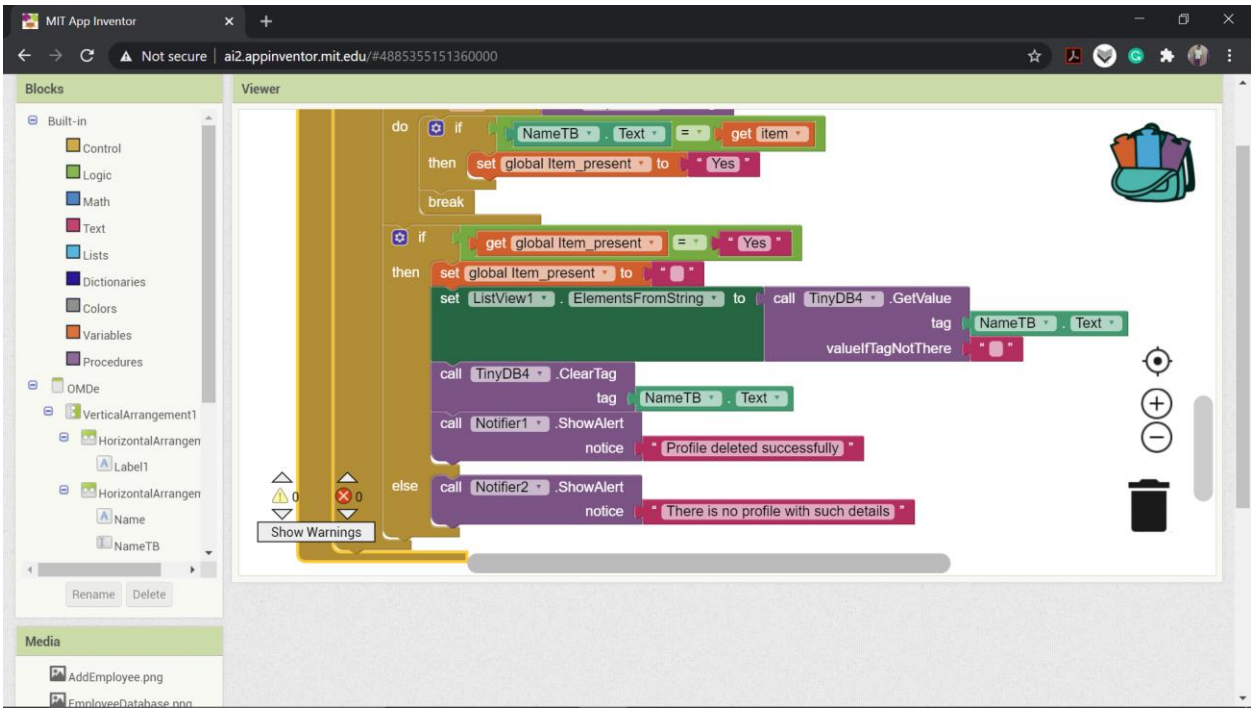



# For Deleting :



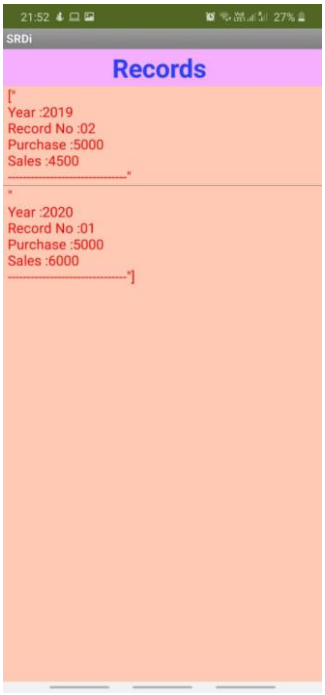

MIT App Inventor
Not secure | ai2.appinventor.mit.edu/#4885355151360000
LoginandSignup_Screen_checkpoint3
Blocks
Viewer
Show Warnings

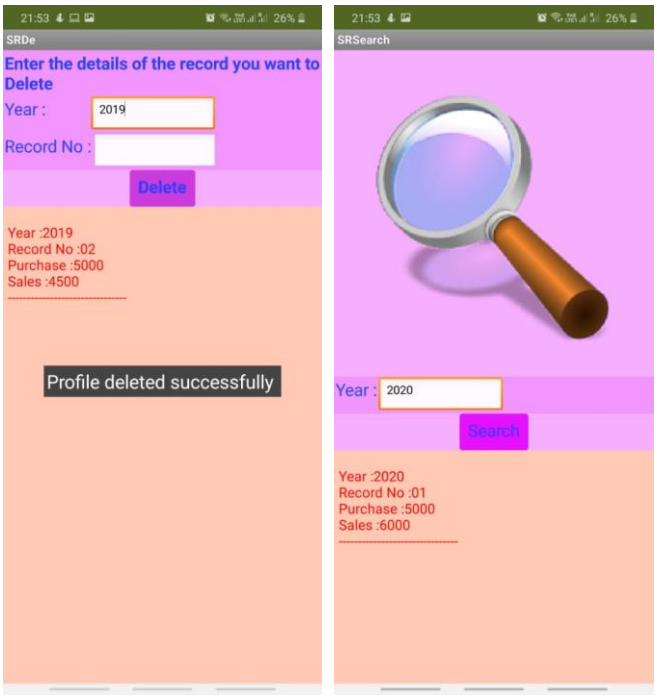

MIT App Inventor
Not secure | ai2.appinventor.mit.edu/#4885355151360000
LoginandSignup_Screen_checkpoint3
Blocks
Viewer
Show Warnings



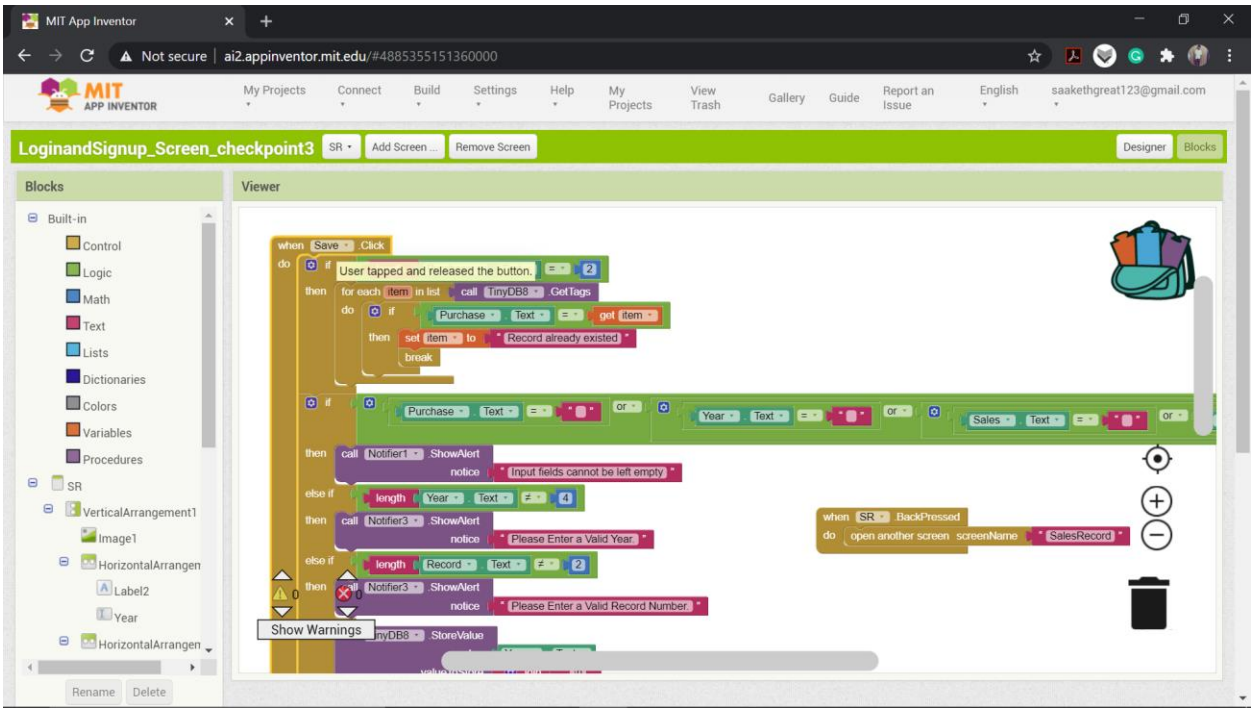

# Sales Record :

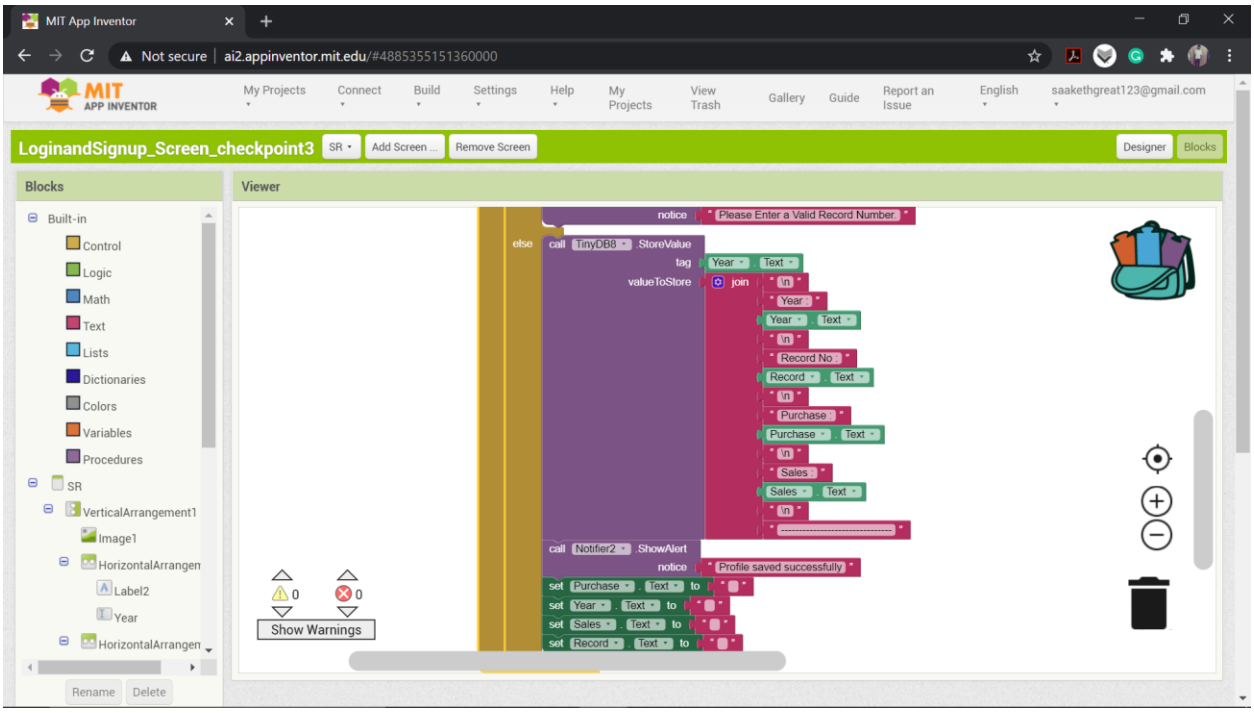




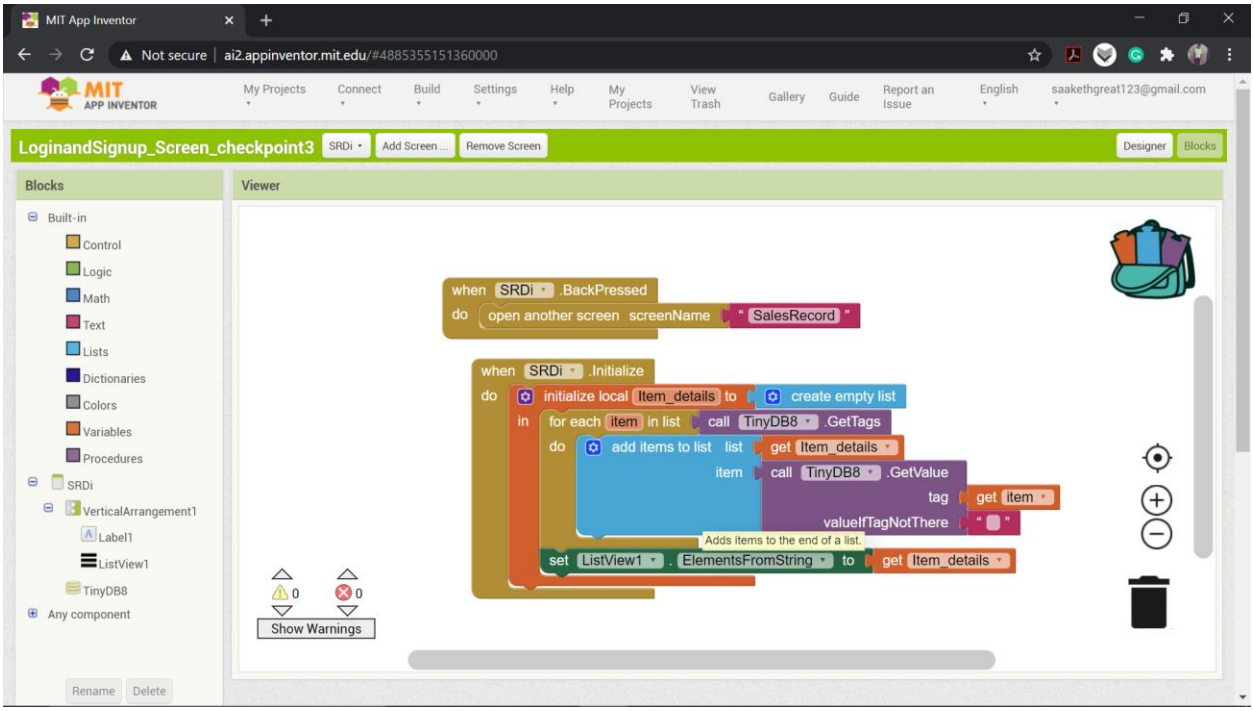

SRDe
Enter the details of the record you want to Delete
Year :
2019
Record No :
Delete
Year :2019
Record No :02
Purchase :5000
Sales :4500
Profile deleted successfully
SRSearch
Year :
2020
Search
Year :2020
Record No :01
Purchase :5000
Sales :6000

# For Adding :

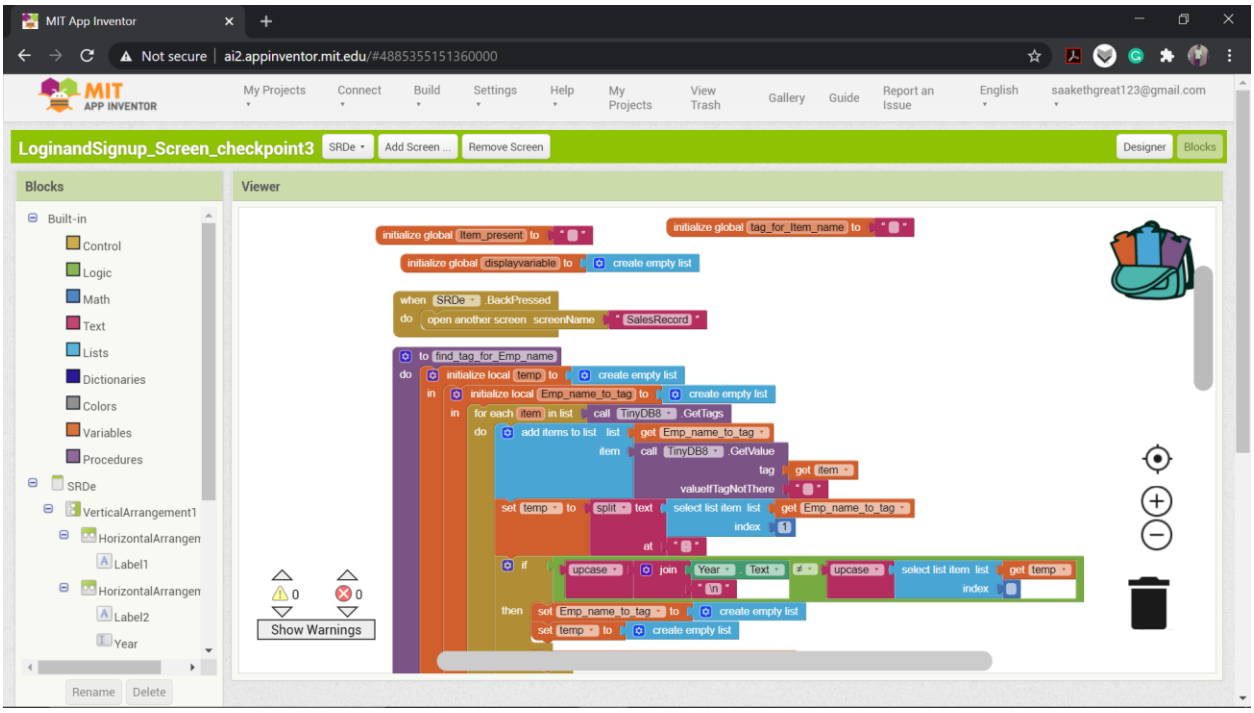

MIT App Inventor
Not secure | ai2.appinventor.mit.edu/#4885355151360000
LoginandSignup_Screen_checkpoint3
Add Screen ...
Remove Screen
Designer
Blocks
Viewer
Built-in
Control
Logic
Math
Text
Lists
Dictionaries
Colors
Variables
Procedures
SR
VerticalArrangement1
Image1
HorizontalArrangen
Label2
Year
Rename
Delete
Show Warnings
when Save .Click
User tapped and released the button.
for each item in list call TinyDB8 .GetTags
Purchase Text get item
set item to Record already existed
break
Year Text
Sales Text
call Notifier1 .ShowAlert
notice Input fields cannot be left empty
length Year Text
call Notifier3 .ShowAlert
notice Please Enter a Valid Year
length Record Text
notice Please Enter a Valid Record Number
when SR .BackPressed
do open another screen screenName SalesRecord

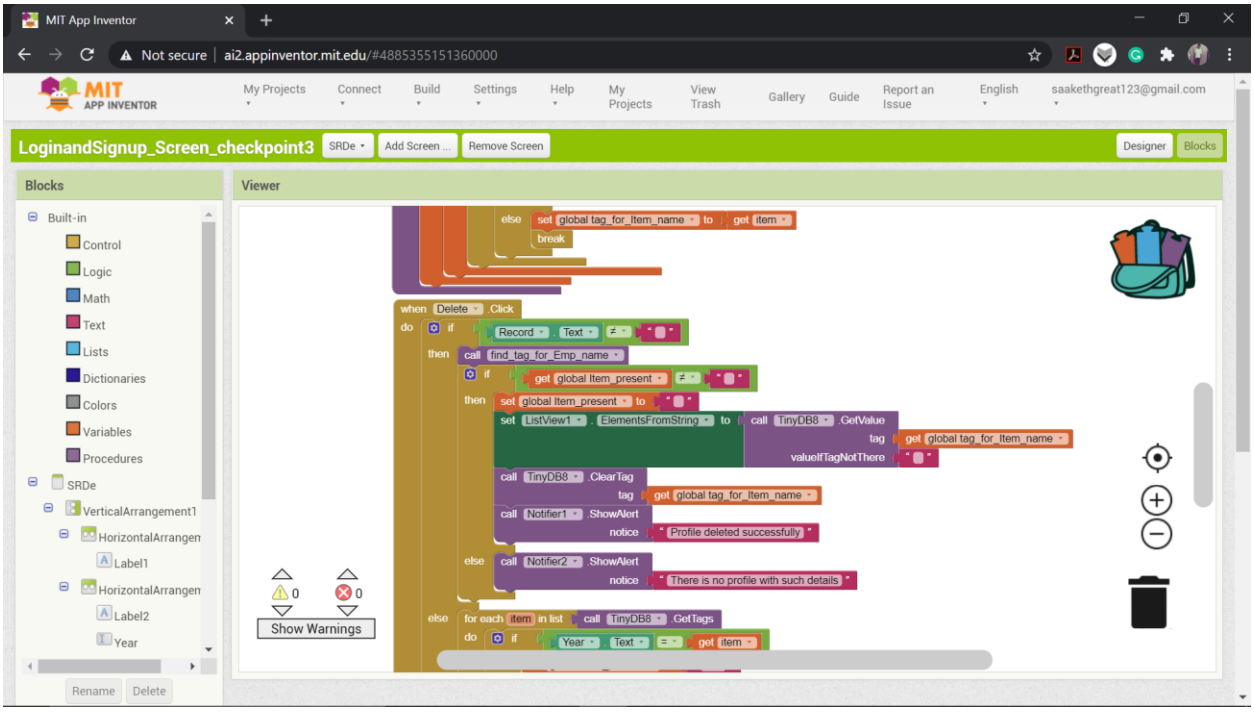


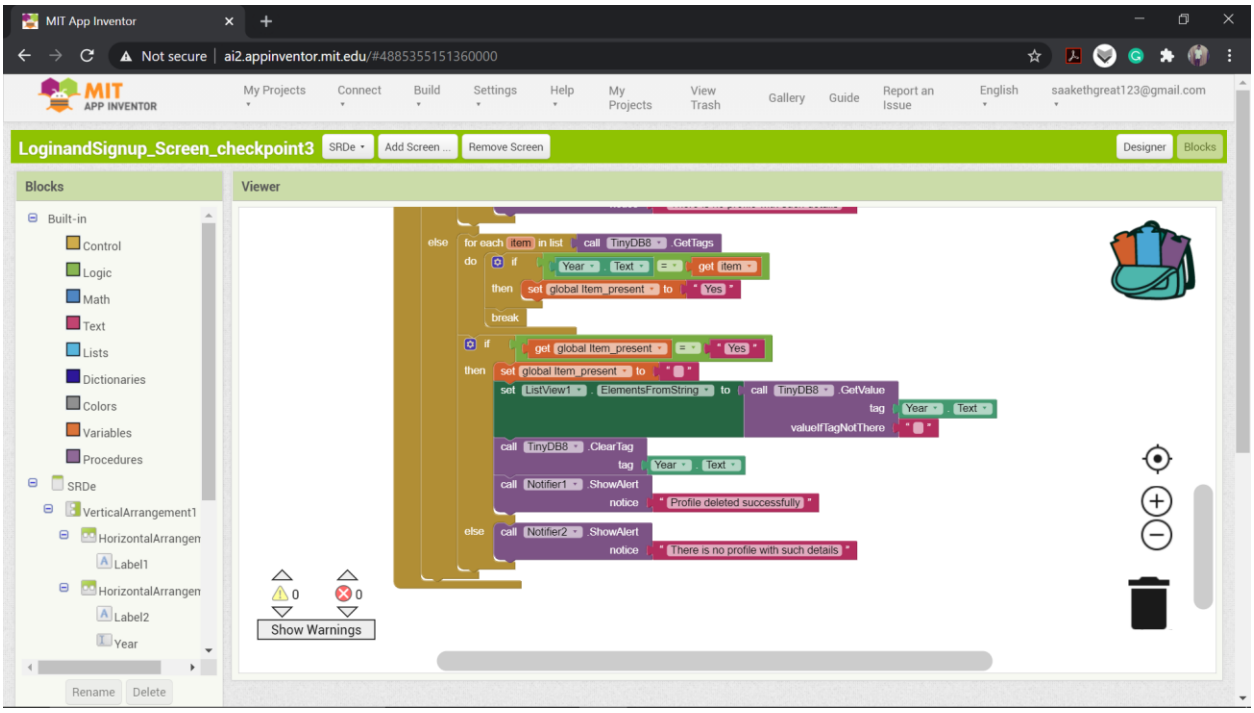

For Displaying :



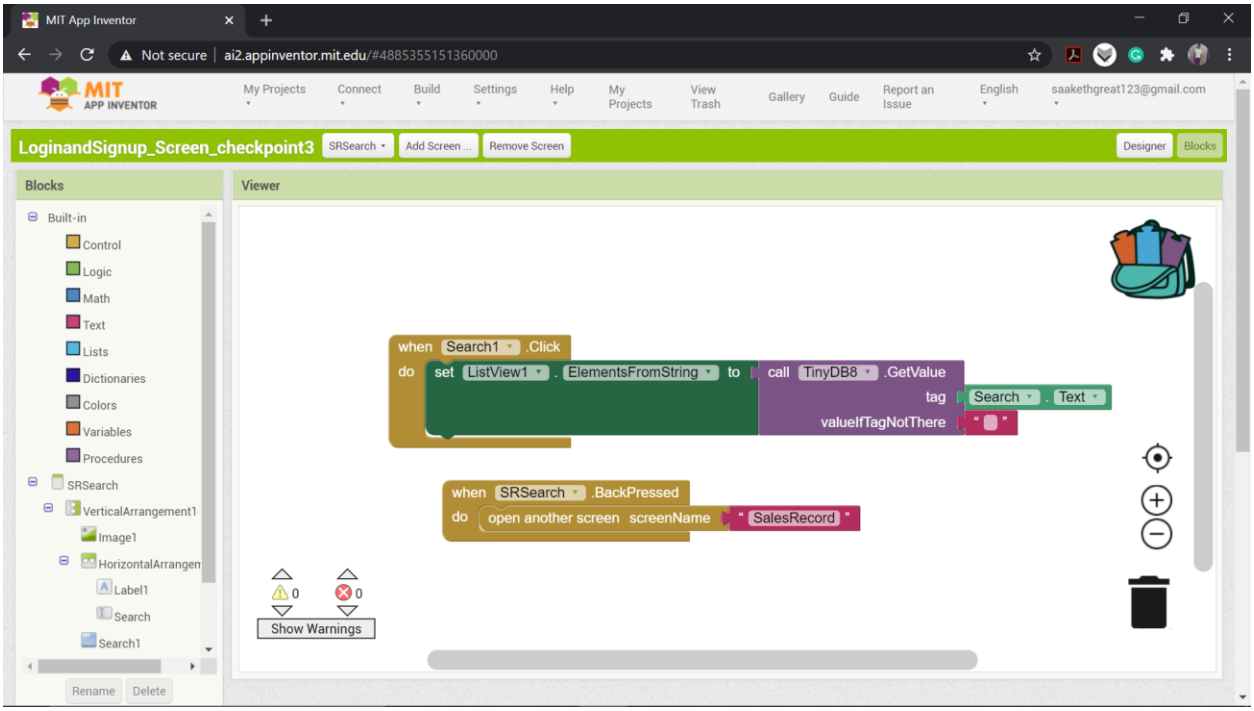

For Deleting :


MIT App Inventor
Not secure
ai2.appinventor.mit.edu/#4885355151360000
MIT APP INVENTOR
My Projects
Connect
Build
Settings
Help
My Projects
View Trash
Gallery
Guide
Report an Issue
English
saakethgreat123@gmail.com
LoginandSignup_Screen_checkpoint3
SRDe
Add Screen ...
Remove Screen
Designer
Blocks
Blocks
Viewer
Built-in
Control
Logic
Math
Text
Lists
Dictionaries
Colors
Variables
Procedures
SRDe
VerticalArrangement1
HorizontalArrangen
Label1
HorizontalArrangen
Label2
Year
Rename
Delete
Show Warnings
initialize global Item_present to
initialize global tag_for_Item_name to
initialize global displayvariable to create empty list
when SRDe .BackPressed
do open another screen screenName "SalesRecord"
to find_tag_for_Emp_name
do initialize local temp to create empty list
in initialize local Emp_name_to_tag to create empty list
in for each item in list call TinyDB8 .GetTags
do add items to list list get Emp_name_to_tag
item call TinyDB8 .GetValue
tag get item
valueIfTagNotThere
set temp to split text select list item list get Emp_name_to_tag
index 1
at
if upcase join Year . Text ≠ upcase select list item list get temp
index
\n
then set Emp_name_to_tag to create empty list
set temp to create empty list


MIT App Inventor
Not secure
ai2.appinventor.mit.edu/#4885355151360000
MIT APP INVENTOR
My Projects
Connect
Build
Settings
Help
My Projects
View Trash
Gallery
Guide
Report an Issue
English
saakethgreat123@gmail.com
LoginandSignup_Screen_checkpoint3
SRDe
Add Screen ...
Remove Screen
Designer
Blocks
Blocks
Viewer
Built-in
Control
Logic
Math
Text
Lists
Dictionaries
Colors
Variables
Procedures
SRDe
VerticalArrangement1
HorizontalArrangen
Label1
HorizontalArrangen
Label2
Year
Rename
Delete
Show Warnings
else set global tag_for_Item_name to get item
break
when Delete .Click
do if Record Text ≠
then call find_tag_for_Emp_name
if get global Item_present ≠
then set global Item_present to
set ListView1 . ElementsFromString to call TinyDB8 .GetValue
tag get global tag_for_Item_name
valueIfTagNotThere
call TinyDB8 .ClearTag
tag get global tag_for_Item_name
call Notifier1 .ShowAlert
notice "Profile deleted successfully"
else call Notifier2 .ShowAlert
notice "There is no profile with such details"
else for each item in list call TinyDB8 .GetTags
do if Year . Text = get item




# For Searching :



## **Conclusion** :

This software will try to provide a rich interface so that novice users can access easily. This application will have information like item name, expenditure, employee name, salary, id number, etc.

Our project is only a humble venture to satisfy the needs of the owner of the Restaurant. Several user-friendly coding has also adopted. This package shall prove to be a powerful package in meeting all the requirements of the Restaurant.

The objective of software planning is to provide a frame work that enables the authority to make reasonable estimates made within a limited time frame at the beginning of the project.

## **References** :


- https://medium.com/@LimeTrayTech/restaurant-management-system-all-components-explained-9cd0e2c446fe
- https://scholarworks.gvsu.edu/cgi/viewcontent.cgi?article=1222&context=cistechlib
- https://www.vertabelo.com/blog/a-restaurant-delivery-data-model/

- https://limetray.com/blog/restaurant-management-system/
- https://www.geeksforgeeks.org/project-idea-smart-restaurants/
- http://www.enggroom.com/c.aspx
- https://www.slideshare.net/harshmathur18/harsh-mathur-final-year-project-report-on-restaurant-billing-system
- https://www.irjet.net/archives/V4/i3/IRJET-V4I3720.pdf
- https://studentnet.cs.manchester.ac.uk/resources/library/thesis_abstracts/MSc13/FullText/Tan-ChinLoong-fulltext.pdf
- http://eceweb1.rutgers.edu/~marsic/books/SE/projects/Restaurant/2014-g4-report3.pdf
- https://sites.google.com/site/ignoubcafinalyearprojects/project-report/restaurent-management-system-project-report


## **Course Code** :

```
#include<stdlib.h>

#include<stdio.h>

#include<string.h>

struct btnode
```

```
{
    int year;
    int cost;
    int sales;
    int profit;
    struct btnode *l;
    struct btnode *r;
}*root = NULL, *temp = NULL, *t2, *t1;
void delete1(struct btnode *t);
void insert();
void delete2();
void inorder(struct btnode *t);
void create();
void search(struct btnode *t);
void search1(struct btnode *t,int data);
int smallest(struct btnode *t);
int largest(struct btnode *t);
void particular_search(struct btnode *t,int yr)
{
    if(t==NULL)
    printf("\nRecord not available.\n");
```

```
        if(t->year==yr)
        {
            printf("--%d --\n ", t->year);
            printf("Cost:%d \n ", t->cost);
            printf("Sales:%d \n", t->sales);
            printf("Profit:%d \n", t->profit);
            printf("\n");
        }
    if(t->year>yr)
    {
        particular_search(t->l,yr);
    }

    else if(t->year<yr)
    {
        particular_search(t->r,yr);
    }

}
int flag = 1;
```

```
void sales_record()
{
    int ch=0;

    while(ch!=5)
    {
        printf("\n1 - Enter a New Record.\n");
        printf("2 - Erase an Incorrect Record\n");
        printf("3 - Chronological Order of Existing Records\n");
        printf("4 - Search a particular year's Record'\n");
        printf("5 - Exit\n");
        printf("\nEnter your choice : ");
        scanf("%d", &ch);
        int yr;
        switch (ch)
        {
            case 1:
                insert();
                break;
            case 2:
                delete2();
```

```
            break;
        case 3:
            inorder(root);
            break;
        case 4:
            printf("Enter the year of the record to be searched :");
            scanf("%d",&yr);
            particular_search(root,yr);
            break;
        case 5:
            continue;
            break;

        default :
            printf("Wrong choice, Please enter correct choice.");
            break;
      }
   }
}
void insert()
{
```

```
    create();
    if (root == NULL)
        root = temp;
    else
        search(root);
}
void create()
{

    temp = (struct btnode *)malloc(1*sizeof(struct btnode));
    printf("Enter the year of the Record: ");
    scanf("%d", &temp->year);
    printf("Enter the Cost of the Record: ");
    scanf("%d", &temp->cost);
    printf("Enter the sales of the Record: ");
    scanf("%d", &temp->sales);
    temp->profit=temp->sales-temp->cost;
    temp->l = temp->r = NULL;
}
void search(struct btnode *t)
{
```

```
    if ((temp->year > t->year) && (t->r != NULL))
        search(t->r);
    else if ((temp->year > t->year) && (t->r == NULL))
        t->r = temp;
    else if ((temp->year < t->year) && (t->l != NULL))
        search(t->l);
    else if ((temp->year < t->year) && (t->l == NULL))
        t->l = temp;
}
void inorder(struct btnode *t)
{
    if (root == NULL)
    {
        printf("No records to display");
        return;
    }
    if (t->l != NULL)
        inorder(t->l);
        printf("--%d --\n ", t->year);
        printf("Cost:%d \n ", t->cost);
        printf("Sales:%d \n", t->sales);
```

```
        printf("Profit:%d \n", t->profit);
        printf("\n");
    if(t->r != NULL){
            inorder(t->r);

    }

}
/* To check for the deleted node */
void delete2()
{
    int data;
    if (root == NULL)
    {
        printf("No records to delete");
        return;
    }
    printf("Enter the year to be deleted : ");
    scanf("%d", &data);
    t1 = root;
    t2 = root;
```

```
    search1(root, data);
}
/* Search for the appropriate position to insert the new node */
void search1(struct btnode *t, int data)
{
    if ((data>t->year))
    {
        t1 = t;
        search1(t->r, data);
    }
    else if ((data < t->year))
    {
        t1 = t;
        search1(t->l, data);
    }
    else if ((data==t->year))
    {
        delete1(t);
    }
}
/* To delete a node */
```

```
void delete1(struct btnode *t)
{
    int k;
 /* To delete leaf node */
    if ((t->l == NULL) && (t->r == NULL))
    {
        if (t1->l == t)
        {
            t1->l = NULL;
        }
        else
        {
        t1->r = NULL;
        }
        t = NULL;
        free(t);
        return;
    }
 /* To delete node having one left hand child */
    else if ((t->r == NULL))
    {
```

```
        if (t1 == t)
        {
            root = t->l;
            t1 = root;
        }
        else if (t1->l == t)
        {
            t1->l = t->l;
        }
        else
        {
            t1->r = t->l;
        }
        t = NULL;
        free(t);
        return;
    }
/* To delete node having right hand child */
    else if (t->l == NULL)
    {
        if (t1 == t)
```

```
        {
            root = t->r;
            t1 = root;
        }
        else if (t1->r == t)
        {
          t1->r = t->r;
        }

        else{
            t1->l = t->r;
            t == NULL;
            free(t);
            return;
        }

    }
/* To delete node having two child */
    else if ((t->l != NULL) && (t->r != NULL))
    {
        t2 = root;
```

```
        if (t->r != NULL)
        {
            k = smallest(t->r);
            flag = 1;
        }
        else
        {
            k =largest(t->l);
            flag = 2;
        }
        search1(root, k);
        t->year = k;
    }
}

int smallest(struct btnode *t)
{
    t2 = t;
    if (t->l != NULL)
    {
        t2 = t;
```

```
        return(smallest(t->l));
    }
    else
    return (t->year);
}

int largest(struct btnode *t)
{
    if (t->r != NULL)
    {
        t2 = t;
        return(largest(t->r));
    }
    else
        return(t->year);
}

struct node *create1(struct node *start);
void display(struct node *start);
struct node *delete_func(struct node *start);
void enqueue_order();
```

```
void display_order();
void dequeue_order();
void check(struct order *temp);
struct node
{
    int expenditure;
    char name[100];
    struct node *next;
};
struct order
{
    char name[100];
    int month;
    int year;
    int date;
};

struct order queue[100];
int rear=-1,front=-1;
struct node *item_start=NULL;
struct node *employee_start=NULL;
```

```
void order_manager()
{
        int choice=0;
        while(choice!=4)
        {
            printf("\n 1.Add a New Order .\n");
            printf(" 2.Display the Pending Orders\n");
            printf(" 3.Remove the Upcoming Order.\n");
            printf(" 4.Exit\n");
            printf("Enter your choice:\t");
            scanf("%d",&choice);
            switch(choice)
            {
                case 1:
                    enqueue_order();
                    break;
                case 2:
                    display_order();
                    break;
                case 3:
                    dequeue_order();
```

```
                break;
            case 4:
                continue;
                break;
            default:
                printf("\n Wrong Choice:\n");
                break;
        }
    }
}

void assign(int i,struct order temp)
{
    int j;
    for (j = rear + 1; j > i; j--)
    {
        queue[j] = queue[j - 1];
    }
    strcpy(queue[rear].name,temp.name);
    queue[i].year = temp.year;
    queue[i].month = temp.month;
```

```
    queue[i].date = temp.date;
}
void enqueue_order()
{
    struct order temper;
    printf("Enter the Name of delivery recipient: ");
    scanf("%s",&temper.name);
    printf("\nEnter the Year of delivery: ");
    scanf("%d",&temper.year);
    printf("Enter the month of delivery: ");
    scanf("%d",&temper.month);
    printf("Enter the date of delivery: ");
    scanf("%d",&temper.date);
    if (rear >=99)
    {
        printf("\nThe Hotel takes in only 100 orders at a time.\n");
        return;
    }
    if ((front == -1) && (rear == -1))
    {
        front++;
```

```
        rear++;
        strcpy(queue[rear].name,temper.name);
        queue[rear].year=temper.year;
        queue[rear].month=temper.month;
        queue[rear].date=temper.date;
        return;
    }
    else
    {
        check1(temp);
        rear++;
    }

};
void check1(struct order temp)
{
    int i,j;
    for (i = 0; i <= rear; i++)
    {
        if (temp.year < queue[i].year)
        {
```

```
            assign(i,temp);
            return;
        }
        else if( temp.year==queue[i].year)
        {
            if (temp.month < queue[i].month)
            {
                assign(i,temp);
                return;
            }
            else if(temp.month==queue[i].month)
            {
                if(temp.date<=queue[i].date)
                {
                    assign(i,temp);
                    return;
                }
            }

        }
    }
```

```
strcpy(queue[rear].name,temp.name);
queue[i].year = temp.year;
queue[i].month = temp.month;
queue[i].date = temp.date;
}
void display_order()
{
    int temperory=front;
    if ((front == -1) && (rear == -1)||(front==rear && front!=0))
    {
        printf("\nNo orders Yet.");
        return;
    }
    for (temperory; temperory <= rear; temperory++)
    {
        printf("%s
=>%d.%d.%d\n",queue[temperory].name,queue[temperory].date,queue[te
mperory].month,queue[temperory].year);
    }
}
void dequeue_order()
```

```
{
    if((rear==-1 && front==-1) || (rear<=front))
    {
        printf("\nNo Orders Pending.\n");
    }
    else
    {
        front++;
    }

}
void employee_menu()
{
    int choice=0;
    while(choice!=4)
    {
        printf("\n 1.Appoint a New Staff Member.\n");
        printf(" 2.Display the Current Staff\n");
        printf(" 3.Delete a Staff Member from the DataBase\n");
        printf(" 4.Exit\n");
        printf("Enter your choice:\t");
```

```
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                employee_start=create1(employee_start);
                break;
            case 2:
                display(employee_start);
                break;
            case 3:
                employee_start=delete_func(employee_start);
                break;
            case 4:
                continue;
                break;
            default:
                printf("\n Wrong Choice:\n");
                break;
        }
    }
}
```

```
void item_menu()
{
int choice=0;
    while(choice!=4)
    {
        printf("\n 1.Add an item to the Menu.\n");
        printf(" 2.Display the items in the Menu.\n");
        printf(" 3.Delete an item from the Menu.\n");
        printf(" 4.Exit\n");
        printf("Enter your choice:\t");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                item_start=create1(item_start);
                break;
            case 2:
                display(item_start);
                break;
            case 3:
                item_start=delete_func(item_start);
```

```
            break;
        case 4:
            continue;
            break;
        default:
            printf("\n Wrong Choice:\n");
            break;
      }


   }
}
int main()
{
   printf("----------------WELCOME TO Notsalgic RESTAURANT----------------");
   int choice=0;
   while(1)
   {
      printf("\n 1.Menu\n");
      printf(" 2.Employee Database\n");
      printf(" 3.Orders Management\n");
```

```
printf(" 4.Sales Records\n");
printf(" 5.Exit\n");
printf("Enter your choice:\t");
scanf("%d",&choice);
switch(choice)
{
    case 1:
        item_menu();
        break;
    case 2:
        employee_menu();
        break;
    case 3:
        order_manager();
        break;
    case 4:
        sales_record();
        break;
    case 5:
        exit(0);
        break;
```

```
            default:
                printf("\n Wrong Choice:\n");
                break;
        }
    }
    return 0;
}
struct node *create1(struct node *start)
{
    struct node *temp,*ptr;
    int count=0;
    temp=(struct node *)malloc(sizeof(struct node));
    if(temp==NULL)
    {
        printf("\nOut of Memory Space:\n");
        exit(0);
    }
    printf("\nEnter the Name :");
    scanf("%s",&temp->name);
    printf("Enter the Expenditure Involved: ");
    scanf("%d",&temp->expenditure);
```

```
    temp->next=NULL;
    if(start==NULL)
    {
        start=temp;
    }
    else
    {
        ptr=start;
        while(ptr->next!=NULL)
        {
            ptr=ptr->next;
        }
        ptr->next=temp;
    }
    return(start);
}
void display(struct node *start)
{
    struct node *ptr;
    int count=1;
    if(start==NULL)
```

```
    {
        printf("\nList is empty.\n");
        return;
    }
    else
    {
        ptr=start;
        while(ptr!=NULL)
        {
            printf("%d %s[ %d]\n",count,ptr->name,ptr->expenditure );
            ptr=ptr->next ;
            count++;
        }
    }
}
struct node *delete_func(struct node *start)
{
    int i,pos;
    struct node *temp,*ptr;
    if(start==NULL)
    {
```

```
    printf("\nThe List is Empty:\n");
}
else
{
    display(start);
    printf("\nEnter the Serial Number to be deleted:\t");
    scanf("%d",&pos);
    if(pos==1)
    {
        ptr=start;
        start=start->next ;
        free(ptr);
    }
    else
    {
        ptr=start;
        for(i=2;i<=pos;i++)
        {
            temp=ptr;
            ptr=ptr->next ;
            if(ptr==NULL)
```

```
            {
                printf("\nSerial Number not Found:\n");
                return(start);
            }
        }
        temp->next =ptr->next ;
        free(ptr);
    }
  }
  return(start);
}
```