IndustriaSync Hub

Session 2023 - 2027

Submitted by:

Umme Aymen 2023-CS-112

Supervised by:

Mam Maida Shahid

Course:

CS-102 Programming Fundamentals

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Table of Contents:

Table of Figures:

❖ About IndustriaSync Hub:

IndustriaSync Hub revolutionizes production plant management, serving as an indispensable tool for managers overseeing operations, supervisors coordinating tasks, and operators executing processes. This dynamic system optimizes workflows, providing real-time insights and fostering seamless collaboration between departments.

➢ Contribution towards CS:

IndustriaSync Hub demonstrates the key features of Computer Science, such as real-time data analysis, collaborative features and regulatory compliance to industrial workflows. It provides an innovative and efficient solution for managing production plant operations.

➢ Why IndustriaSync Hub:

IndustriaSync Hub is essential for efficient production plant management. It provides real-time insights, promotes collaboration and ensures regulatory compliances. It is a crucial tool for business aiming for optimal performance in their operations.

➢ What to expect from IndustriaSync Hub:

With IndustriaSync Hub, users can count on getting up-to-date information. It ensures smooth team work and creates a cooperative environment. This system ensures efficient decision-making, heightened productivity and adherence to industry standards.

❖ User Types on IndustriaSync Hub:

IndustriaSync Hub provides different access for different type of users. The users are divided into three categories, Administrator (Admin), Manager and Customer. Each user type have access to different type of commands related to their need and requirement. User can login and verify their type by inputting the issued username and password for authentification.

The hierarchy and functionality of user types is as under: -

i. Administrator:

The administrator in this production plant management system is central to user management, system configuration, access control, security, compliance, and system monitoring. Responsibilities include overseeing employee data, configuring and maintaining the system, managing access, ensuring security, and facilitating integration with other systems. The administrator's role is pivotal in maintaining system efficiency, monitoring performance, and generating essential production reports.

ii. Manager:

The manager in this production management system plays a vital role in planning and scheduling production, overseeing resource allocation, managing materials and inventory, and ensuring adherence to security protocols. Additionally, the manager is responsible for generating comprehensive reports, overseeing budgeting and cost control for various production departments and resources. Overall, the manager's role spans strategic planning, operational efficiency, and financial management within the production plant.

iii. Customer:

In this customer-centric production management system, users enjoy streamlined functionalities, including efficient order placement and tracking, feedback provision, simplified account management, informative notifications, and a straightforward process for returns and refunds. A dedicated support system with FAQs further enhances the overall customer experience, ensuring a seamless and satisfying interaction with the system.

❖ Functional Requirements of IndustriaSync Hub:

Some of the functional requirements expected from IndustriaSync Hub are as under:

User Type	***Required Function to be Performed***	***Result of Action Performed***
Administrator	User Management	Add, view, delete, search and update employee records
	System Configuration	Configure Production Speed, Alert Threshold and view Feedback
	Access Control	Access to managers and customers at any level
	System Monitoring	Add, view, delete, search and update workers accounts.
	Security and Compliance	A tabular form of security instructions is shown View, add and delete Instructions
	Generate Production Report	Production Report is shown

Manager	Production Planning and Scheduling	Develop Production plans and create schedules.
	Resource Allocation	Assign tasks and allocate resources on the basis of available resources
	Material Handling	A tabular form of available raw materials is shown
	View Security Manual	A tabular form of security instructions is shown
	Generate Sales Report	Generate report on the basis of sales
	Inventory Management	Monitor inventory level and manage stock
	Budgeting and Cost Control	Develop and manage budgets
Customer	Order Placement and Tracking	A tabular form of menu is shown View Menu and customize menu
	Provide Feedback	Provide feedback on received products
	Account Management	Register account
	Notifications	View, delete, update, search and add notifications
	Returns and Refunds	Initiate and view return requests
	Support and FAQs	View Order Gantt and place orders

❖ Wireframe of IndustriaSync Hub:

The following is the wireframe of IndustriaSync Hub displayed in command line interface:

Basic (For all types of Users):

- **Startup Interface:**

Startup interface is as follows:

Figure 1: Startup Interface

- **Sign In Menu:**

Sign in interface is as follows:

Figure 2: Sign In Menu

- **Sign Up Menu:**

Sign up is as follows:

Figure 3: Sign Up Menu

- **Main Menu Interface:**

Main menu interface is as follows:

Figure 4: Main Menu

- **Administrator Menu:**

Administrator menu is as follows:

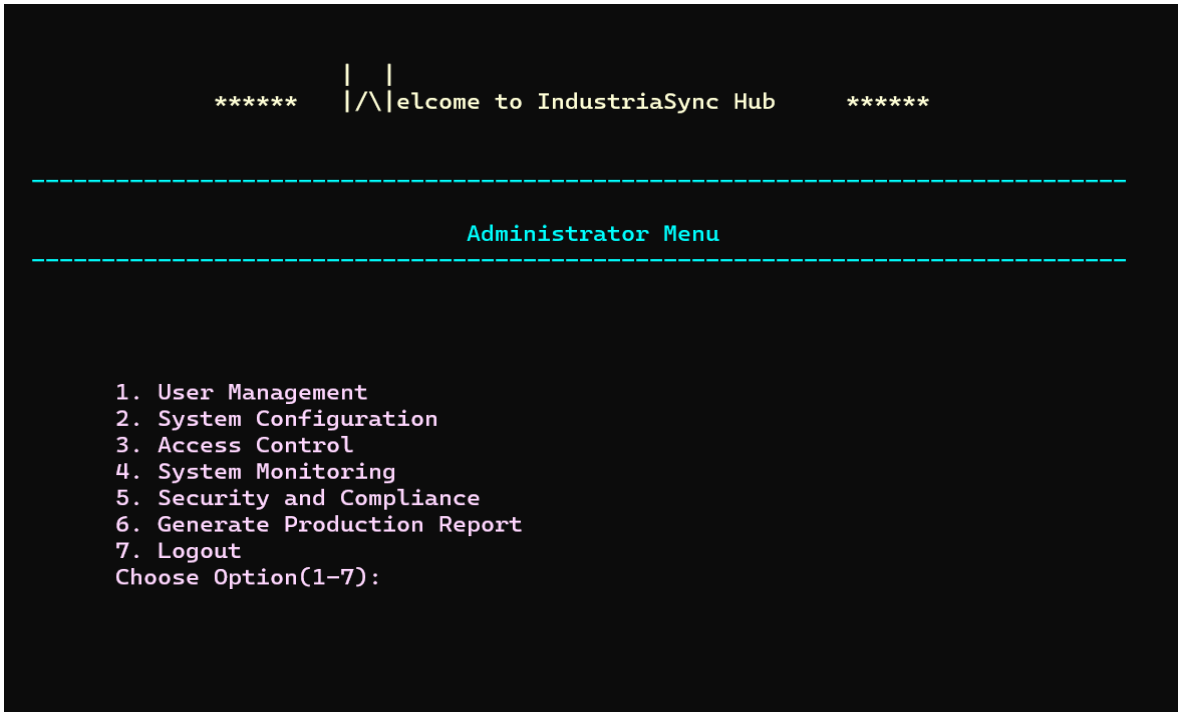


Figure 5: Administrator Menu

- **User Management Menu:**

User Management menu is as follows:

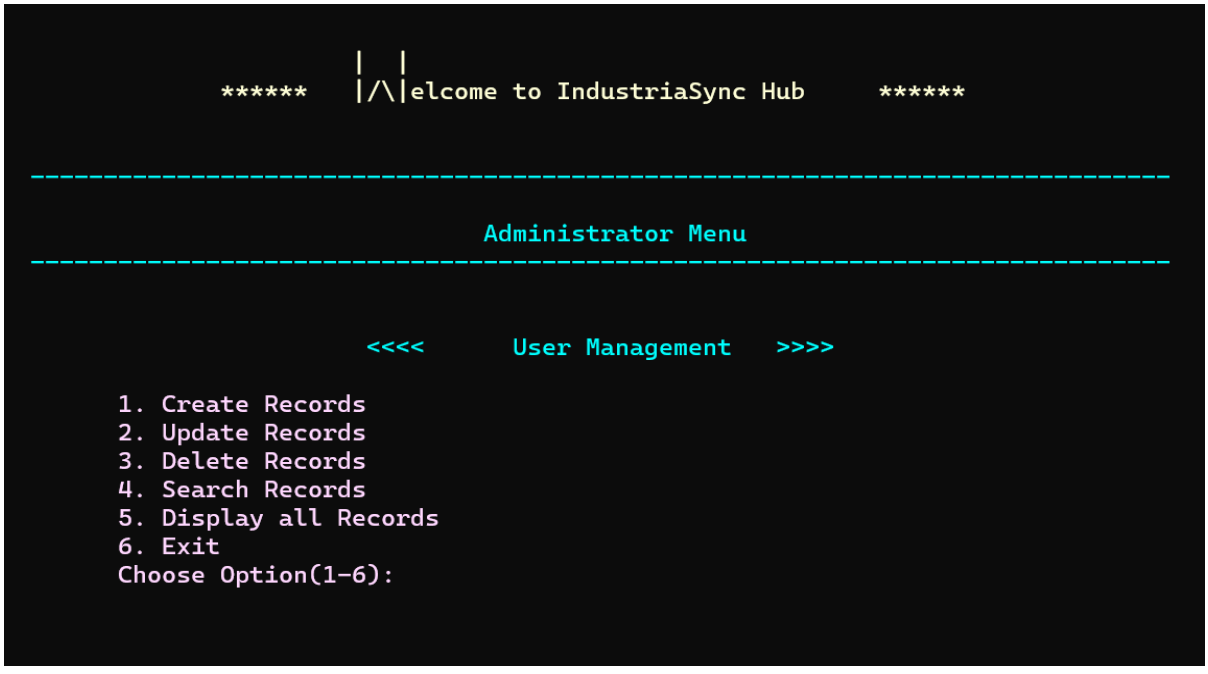


Figure 6: User Management Menu

System Configuration Menu:

System Management menu is as follows:

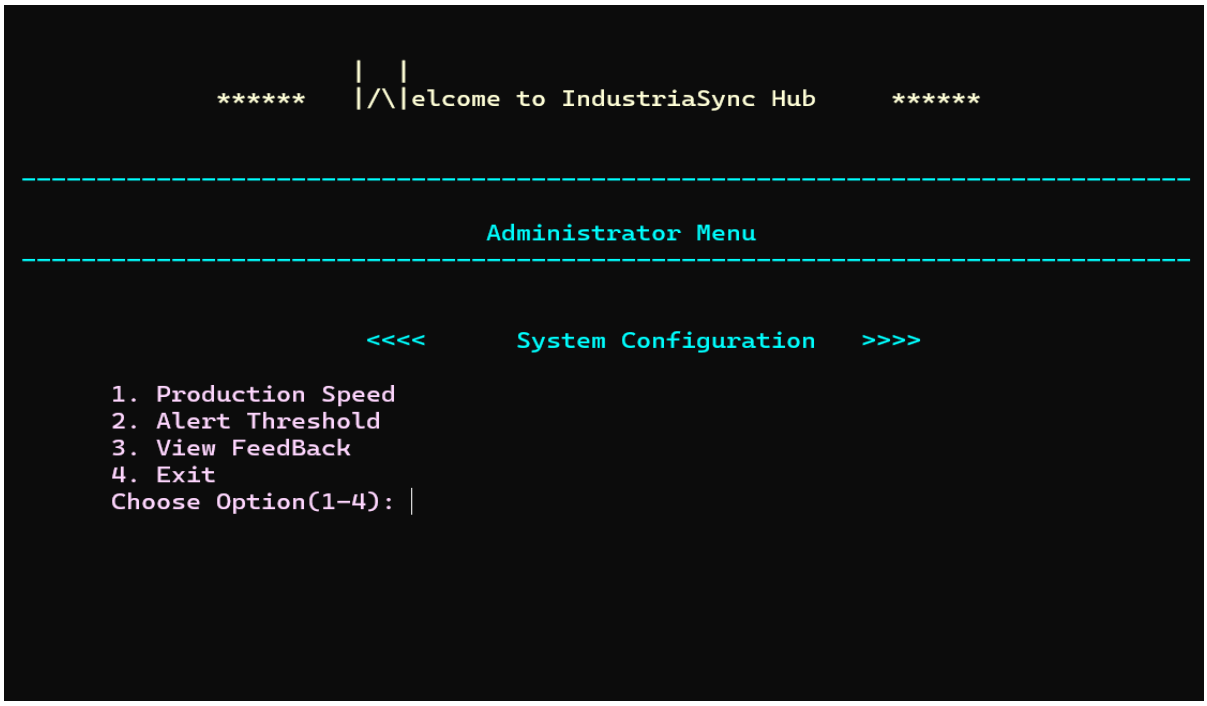


Figure 7: System Configuration Menu

- **Access Control Menu:**

Access Control Menu is as follows:

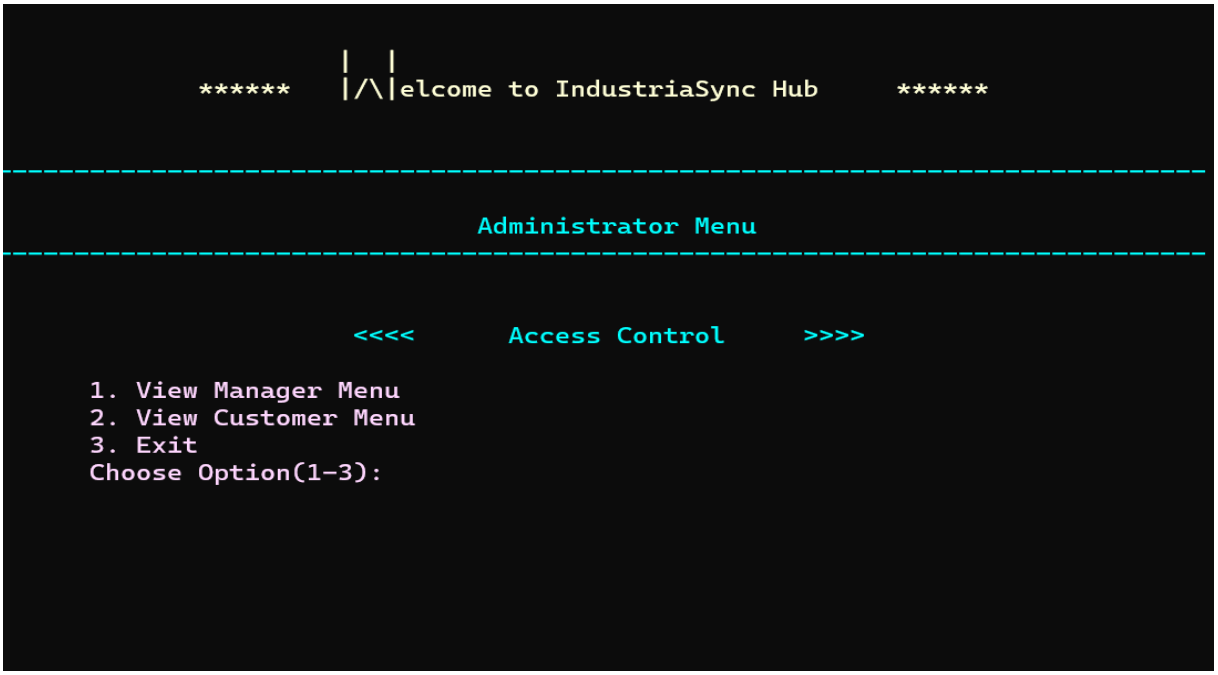


Figure 8: Access Control Menu

- **System Monitoring Menu:**

System Monitoring menu is as follows:

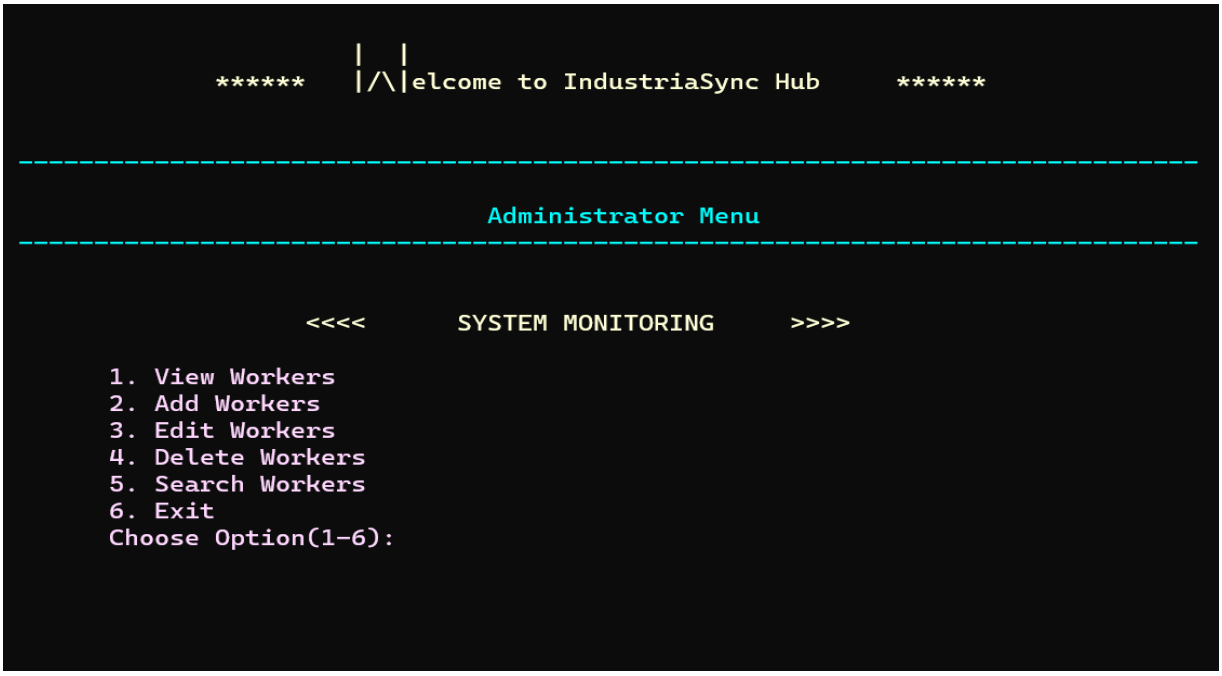
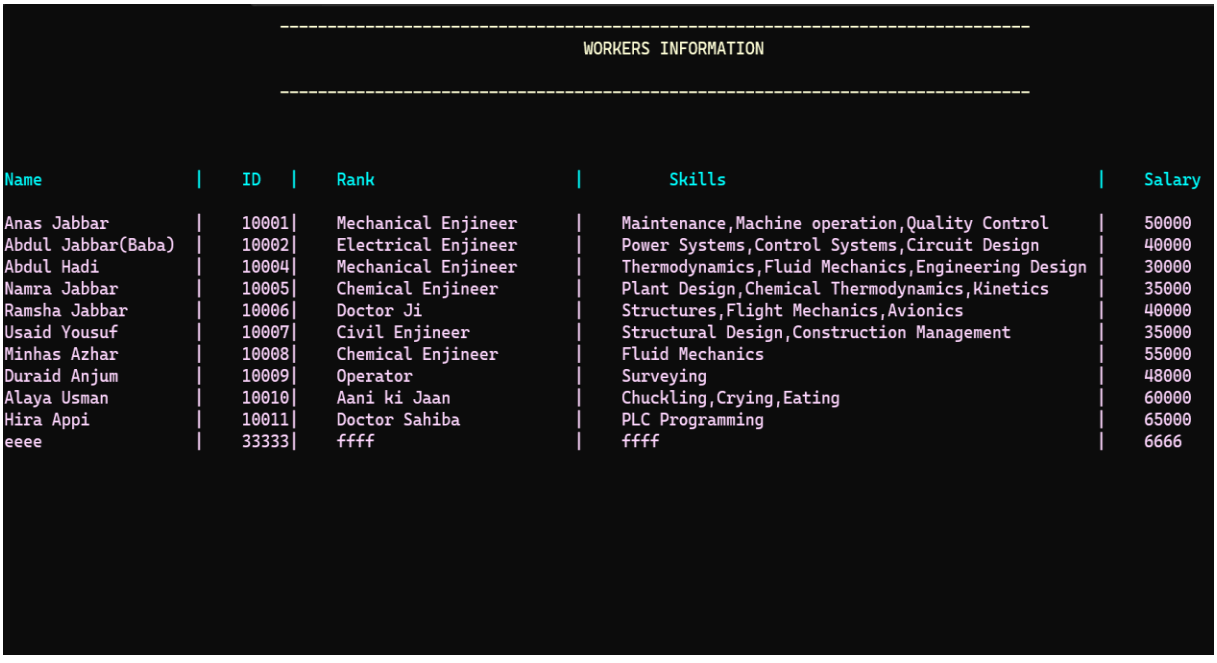


Figure 9: System Monitoring Menu

- **View Workers:**

Workers are viewed is as follows:



Name	ID	Rank	Skills	Salary
Anas Jabbar	10001	Mechanical Enjineer	Maintenance,Machine operation,Quality Control	50000
Abdul Jabbar(Baba)	10002	Electrical Enjineer	Power Systems,Control Systems,Circuit Design	40000
Abdul Hadi	10004	Mechanical Enjineer	Thermodynamics,Fluid Mechanics,Engineering Design	30000
Namra Jabbar	10005	Chemical Enjineer	Plant Design,Chemical Thermodynamics,Kinetics	35000
Ramsha Jabbar	10006	Doctor Ji	Structures,Flight Mechanics,Avionics	40000
Usaid Yousuf	10007	Civil Enjineer	Structural Design,Construction Management	35000
Minhas Azhar	10008	Chemical Enjineer	Fluid Mechanics	55000
Duraid Anjum	10009	Operator	Surveying	48000
Alaya Usman	10010	Aani ki Jaan	Chuckling,Crying,Eating	60000
Hira Appi	10011	Doctor Sahiba	PLC Programming	65000
eeee	33333	ffff	ffff	6666

Figure 10: View Workers

- **Security and Compliance:**

Security and Compliance menu is as follows:

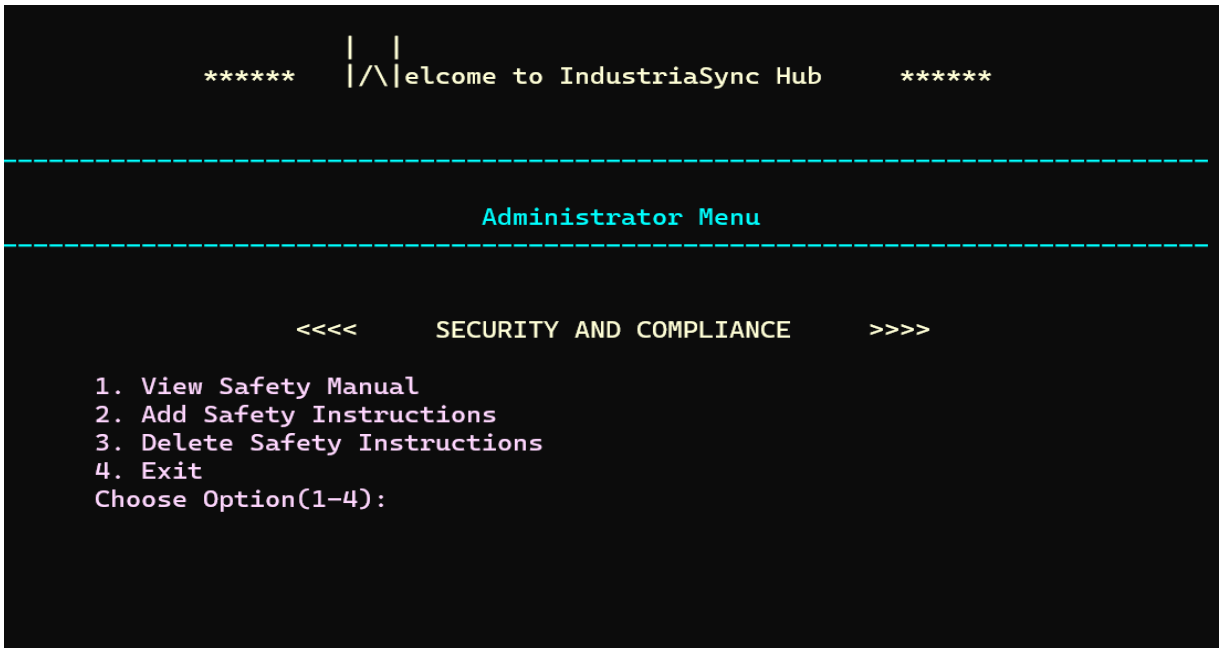


Figure 11: System Monitoring Menu

- **Generate Production Report:**

Production Report has been calculated on the basis of sales:

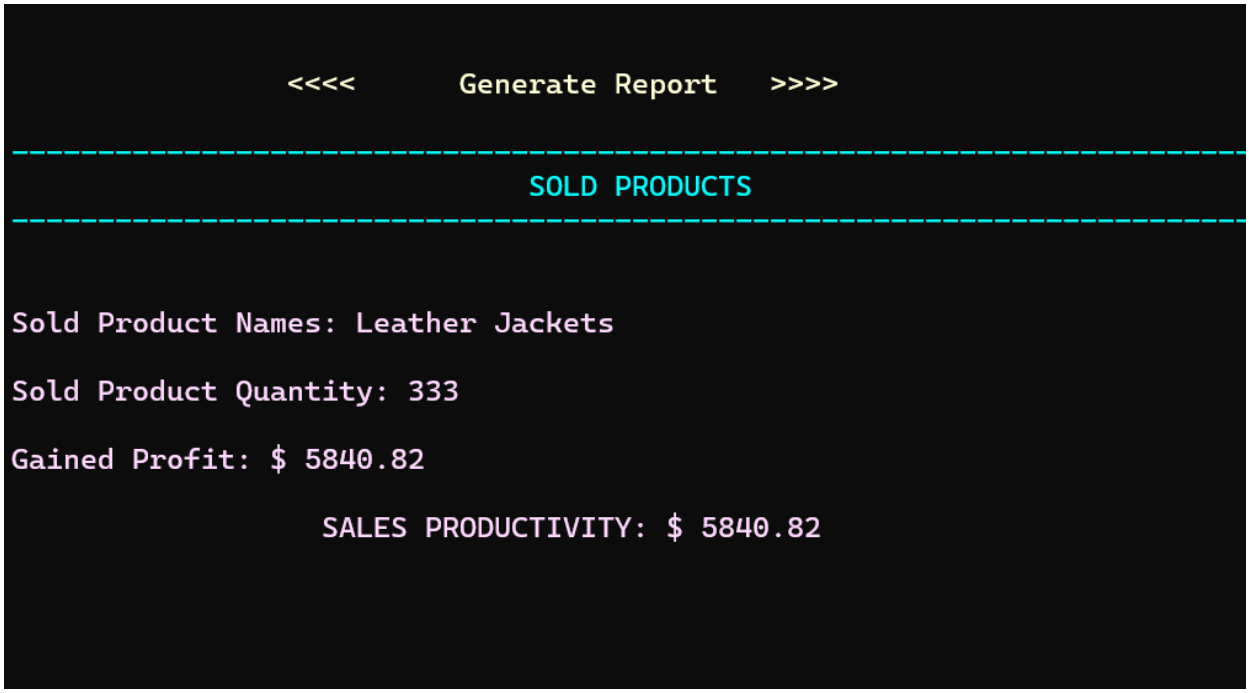


Figure 12: Production Report

- **Material Handling Menu:**

Material Handling menu is as follows:

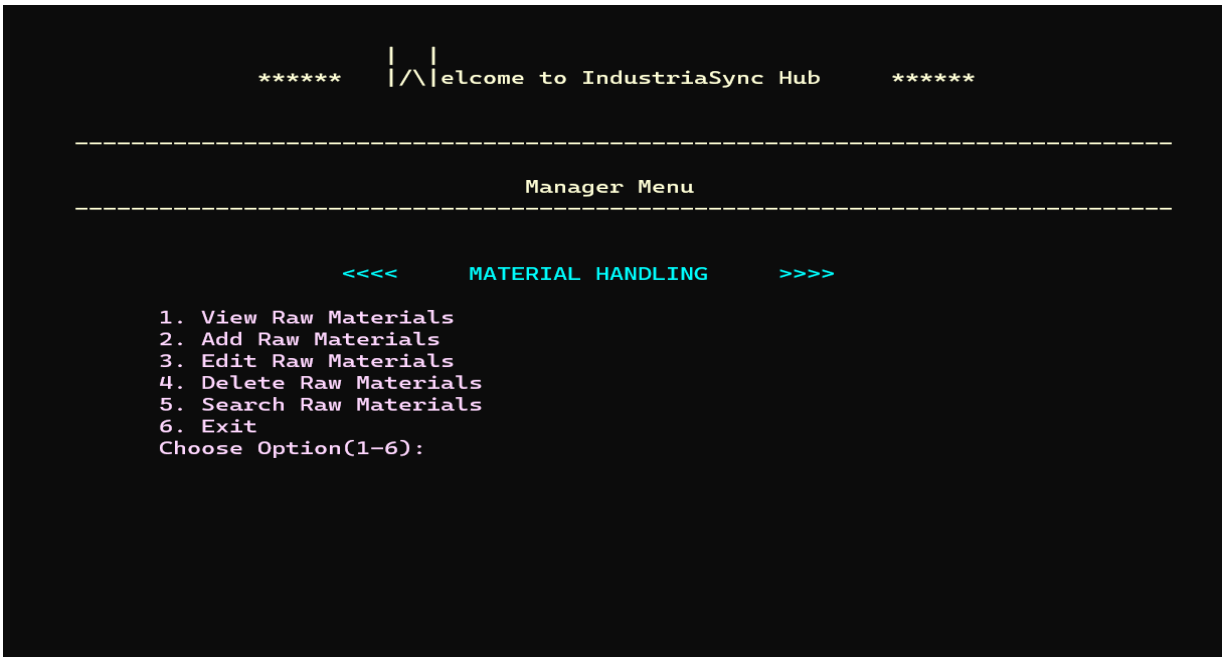
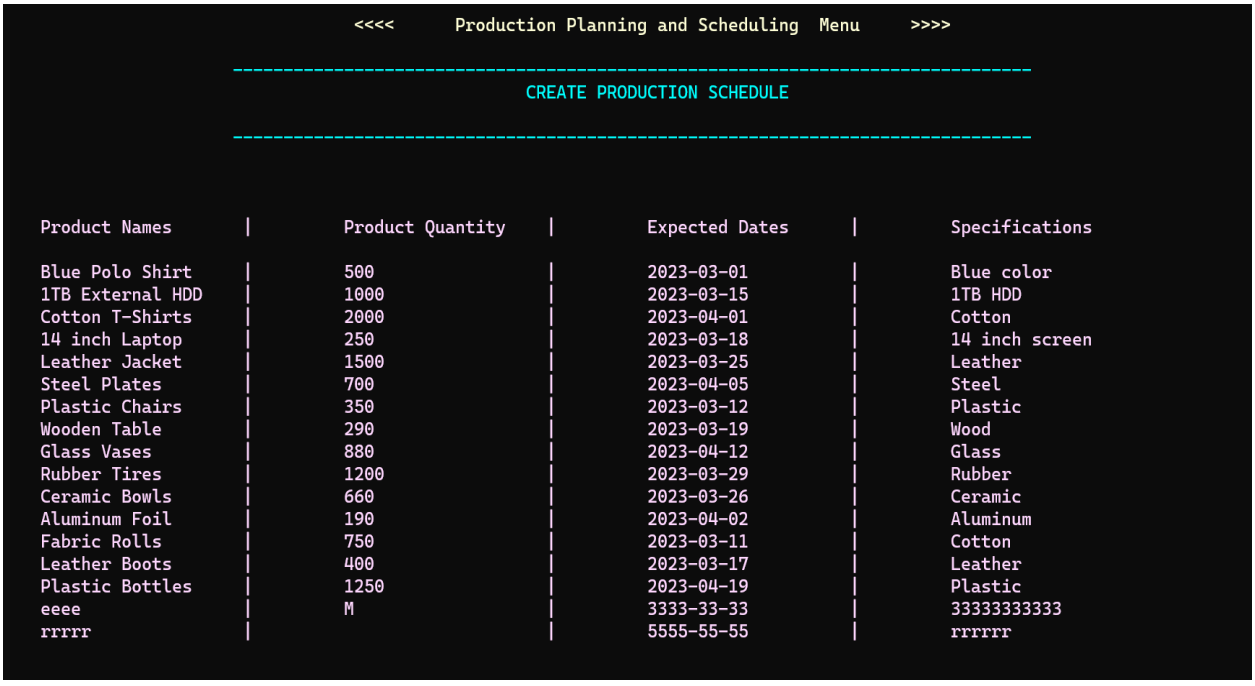


Figure 12: Manager Menu

- **View Production Schedule:**

Production Schedule is as follows:



Product Names	Product Quantity	Expected Dates	Specifications
Blue Polo Shirt	500	2023-03-01	Blue color
1TB External HDD	1000	2023-03-15	1TB HDD
Cotton T-Shirts	2000	2023-04-01	Cotton
14 inch Laptop	250	2023-03-18	14 inch screen
Leather Jacket	1500	2023-03-25	Leather
Steel Plates	700	2023-04-05	Steel
Plastic Chairs	350	2023-03-12	Plastic
Wooden Table	290	2023-03-19	Wood
Glass Vases	880	2023-04-12	Glass
Rubber Tires	1200	2023-03-29	Rubber
Ceramic Bowls	660	2023-03-26	Ceramic
Aluminum Foil	190	2023-04-02	Aluminum
Fabric Rolls	750	2023-03-11	Cotton
Leather Boots	400	2023-03-17	Leather
Plastic Bottles	1250	2023-04-19	Plastic
eeee	M	3333-33-33	33333333333
rrrrr		5555-55-55	rrrrrr

Figure 13: View Production Schedule

- **View Available Inventory:**

Inventory can be added, deleted, updated, viewed and search by ID:

INVENTORY INFORMATION

ID	Type	Names	Specification	Function
1	Lathe	LTH01	12 in max diameter, 24 in max length	Turning and facing operations
2	Milling Machine	MIL01	50 x 100 cm table, 800W motor	Milling, drilling, cutting
3	Drill Press	DRP01	20 mm max drill diameter, 500-1500 RPM	Drilling holes
4	Grinder	GRD01	6 in grinding wheel	Grinding and deburring
5	CNC Machine	CNC01	3-axis, 24 x 36 x 12 in max	Milling, routing, cutting
6	Bandsaw	BSW01	14 in throat depth	Cutting curves and straight cuts
8	Planer	PLN01	20 in max width, 1/2 HP motor	Smoothing and leveling wood surfaces
9	Wood Lathe	WLT01	12 in swing over bed, variable speed	Shaping and turning wood
10	Table Saw	TSW01	10 in blade diameter, 5000 RPM max speed	Ripping and crosscutting wood

Figure 14: Inventory Information

- **Unallocated Resources:**

Resources are allocated on the basis of available operators, inventory and products.

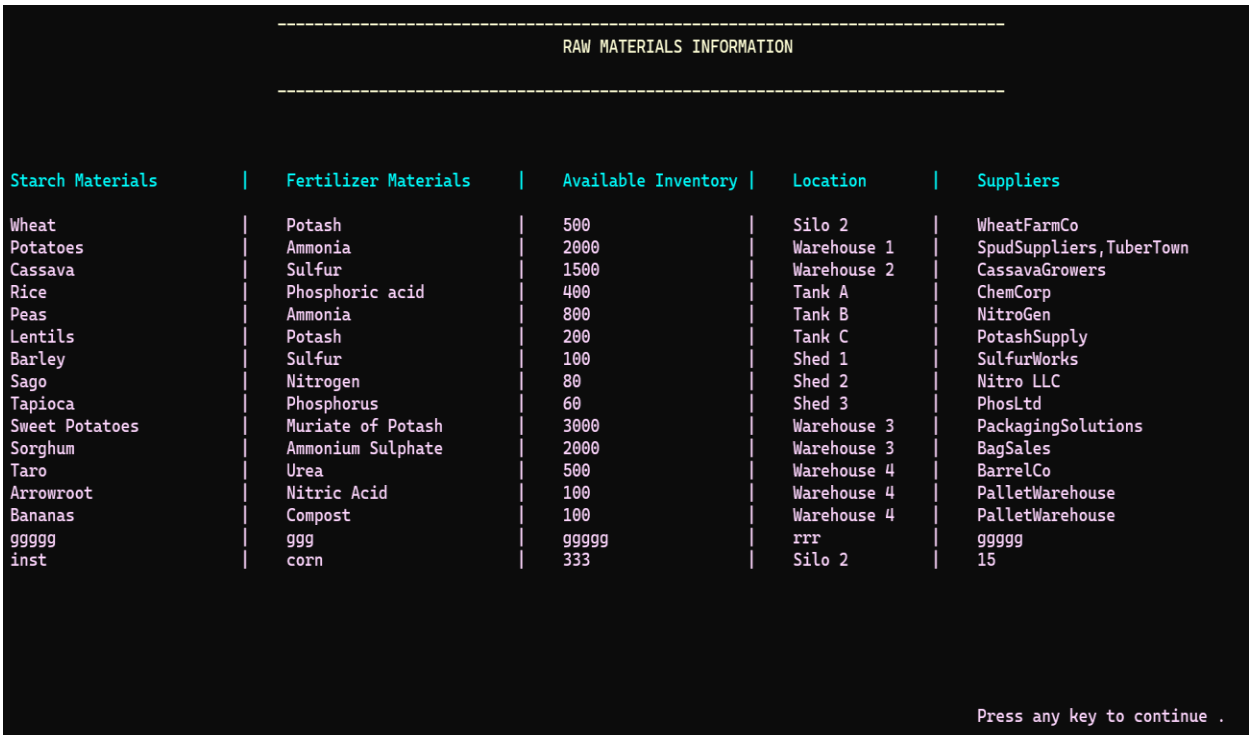
UNALLOCATED RESOURCES

Unallocated Product	Unallocated Quantity	Unallocated Date	Unallocated Operator	Unallocated Machines
Blue Polo Shirt	500	2023-03-01	Anas Jabbar	Lathe
1TB External HDD	1000	2023-03-15	Abdul Jabbar(Baba)	Milling Machine
Cotton T-Shirts	2000	2023-04-01	Abdul Hadi	Drill Press
14 inch Laptop	250	2023-03-18	Namra Jabbar	Grinder
Leather Jacket	1500	2023-03-25	Ramsha Jabbar	CNC Machine
Steel Plates	700	2023-04-05	Usaid Yousuf	Bandsaw
Plastic Chairs	350	2023-03-12	Minhas Azhar	Planer
Wooden Table	290	2023-03-19	Duraid Anjum	Wood Lathe
Glass Vases	880	2023-04-12	Alaya Usman	Table Saw
Rubber Tires	1200	2023-03-29	Hira Appi	
Ceramic Bowls	660	2023-03-26	eeee	
Aluminum Foil	190	2023-04-02		
Fabric Rolls	750	2023-03-11		
Leather Boots	400	2023-03-17		
Plastic Bottles	1250	2023-04-19		
eeee	M	3333-33-33		
rrrrr		5555-55-55		

Figure 15: Unallocated Resources

- **View Raw Materials:**

Raw material information is as follows:



RAW MATERIALS INFORMATION

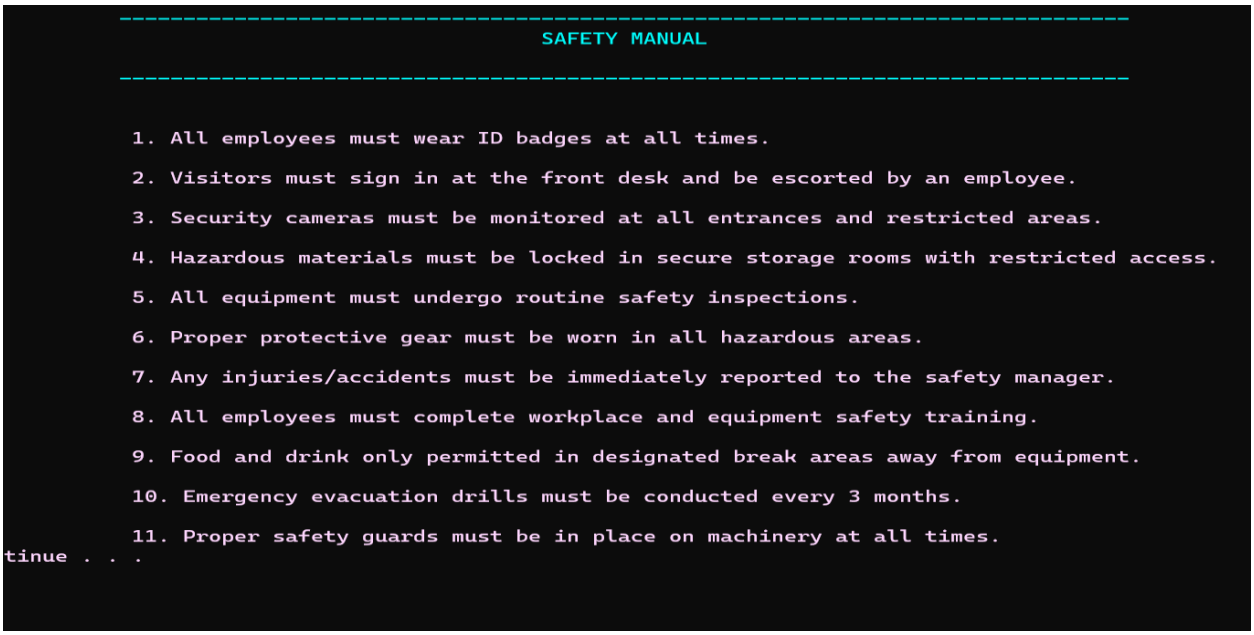
Starch Materials	Fertilizer Materials	Available Inventory	Location	Suppliers
Wheat	Potash	500	Silo 2	WheatFarmCo
Potatoes	Ammonia	2000	Warehouse 1	SpudSuppliers,TuberTown
Cassava	Sulfur	1500	Warehouse 2	CassavaGrowers
Rice	Phosphoric acid	400	Tank A	ChemCorp
Peas	Ammonia	800	Tank B	NitroGen
Lentils	Potash	200	Tank C	PotashSupply
Barley	Sulfur	100	Shed 1	SulfurWorks
Sago	Nitrogen	80	Shed 2	Nitro LLC
Tapioca	Phosphorus	60	Shed 3	PhosLtd
Sweet Potatoes	Muriate of Potash	3000	Warehouse 3	PackagingSolutions
Sorghum	Ammonium Sulphate	2000	Warehouse 3	BagSales
Taro	Urea	500	Warehouse 4	BarrelCo
Arrowroot	Nitric Acid	100	Warehouse 4	PalletWarehouse
Bananas	Compost	100	Warehouse 4	PalletWarehouse
ggggg	ggg	ggggg	rrr	ggggg
inst	corn	333	Silo 2	15

Press any key to continue .

Figure 16: Raw Material Information

- **Security Instruction Manual:**

Security instruction manual is as follows:



SAFETY MANUAL

1. All employees must wear ID badges at all times.
2. Visitors must sign in at the front desk and be escorted by an employee.
3. Security cameras must be monitored at all entrances and restricted areas.
4. Hazardous materials must be locked in secure storage rooms with restricted access.
5. All equipment must undergo routine safety inspections.
6. Proper protective gear must be worn in all hazardous areas.
7. Any injuries/accidents must be immediately reported to the safety manager.
8. All employees must complete workplace and equipment safety training.
9. Food and drink only permitted in designated break areas away from equipment.
10. Emergency evacuation drills must be conducted every 3 months.
11. Proper safety guards must be in place on machinery at all times.

tinue . . .

Figure 17: Instruction Manual

- **Manager Menu:**

Manager menu is as follows:

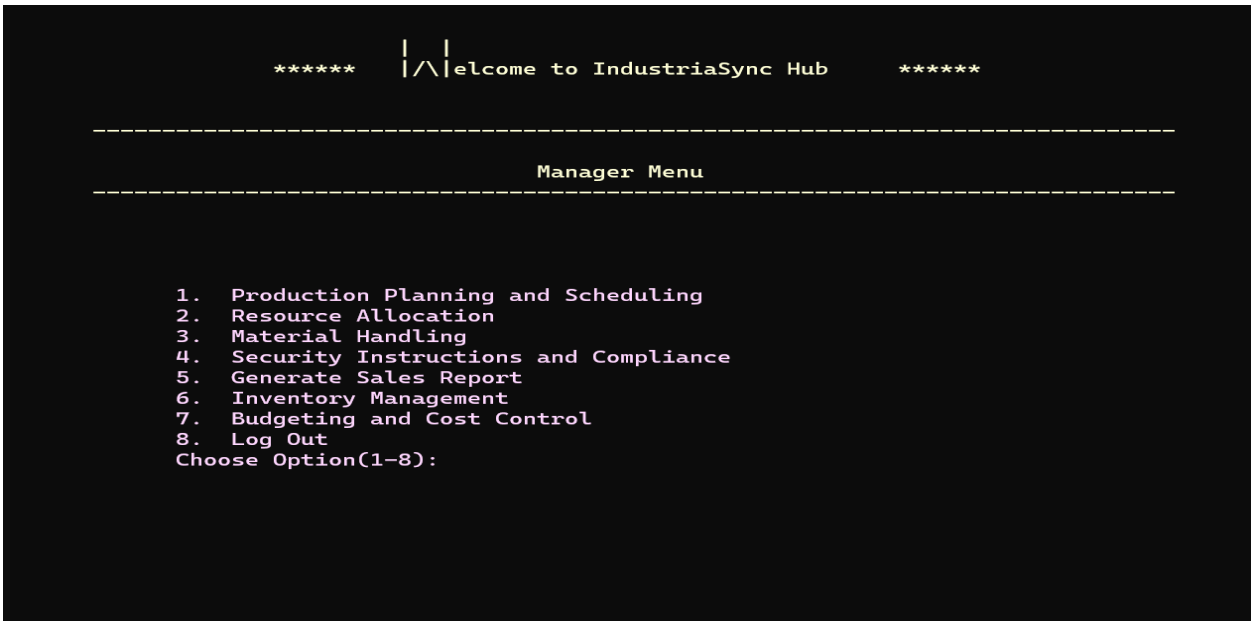


Figure 18: Manager Menu

- **Production Planning and Scheduling:**

Production planning is done on the basis of production orders and Production schedule:

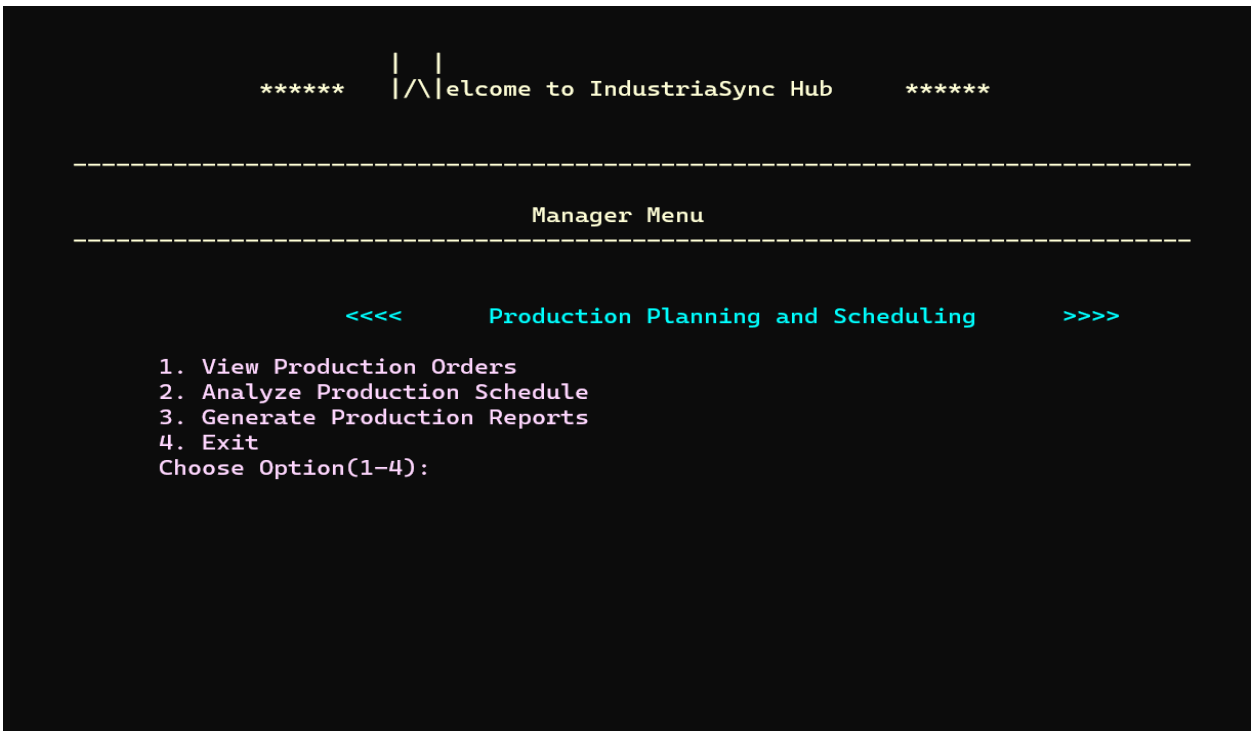


Figure 19: Production Planning and Scheduling

- **Resource Allocation Menu:**

Resources are allocated on the basis of available inventory, available workers and placed orders:

Figure 20: Resource Allocation

- **Customer Menu:**

Customer Menu is as follows:

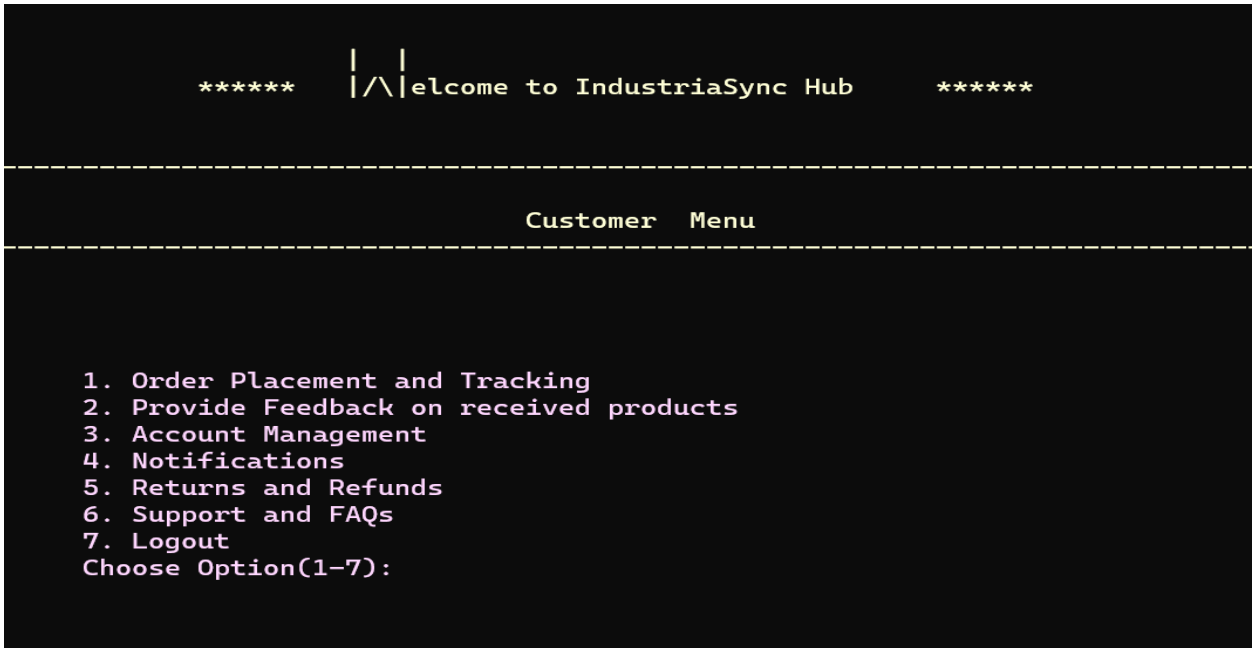


Figure 21: Customer Menu

- **Production Orders:**

Orders are being placed by customers with their professional information:

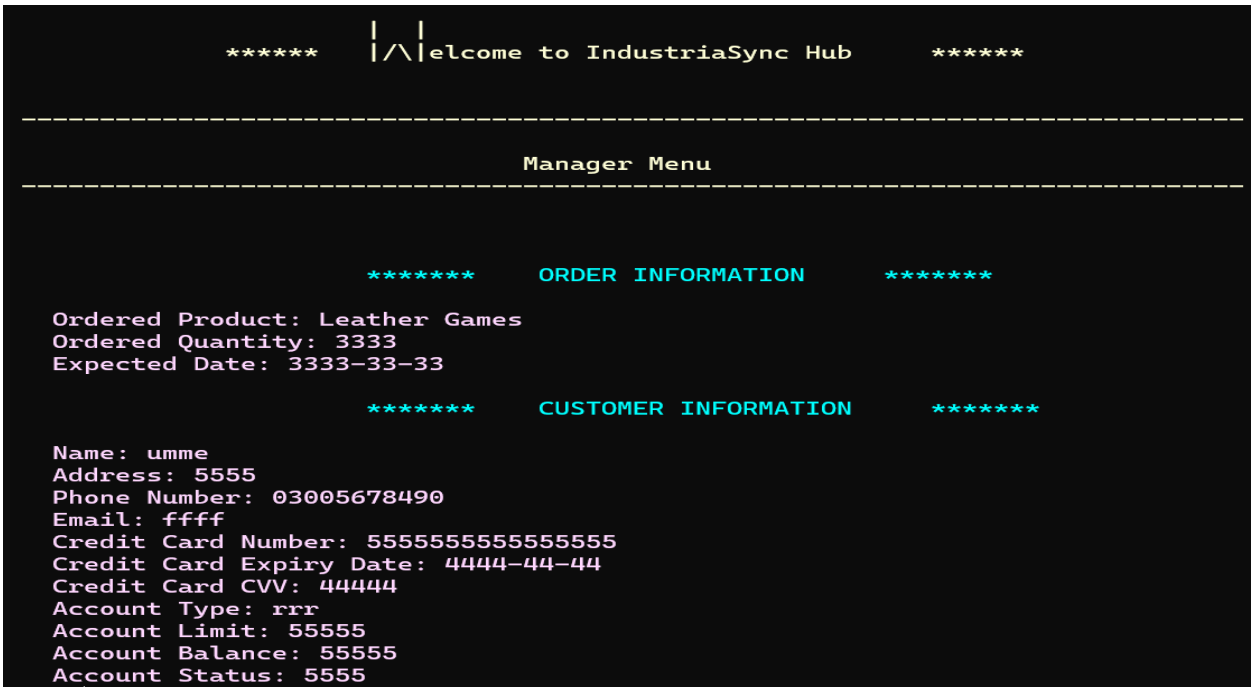


Figure 22: Order Information

- **Budget and Cost Control:**

Budget is calculated on the basis of allocated salaries to employees and prices allocated to raw materials and inventory:

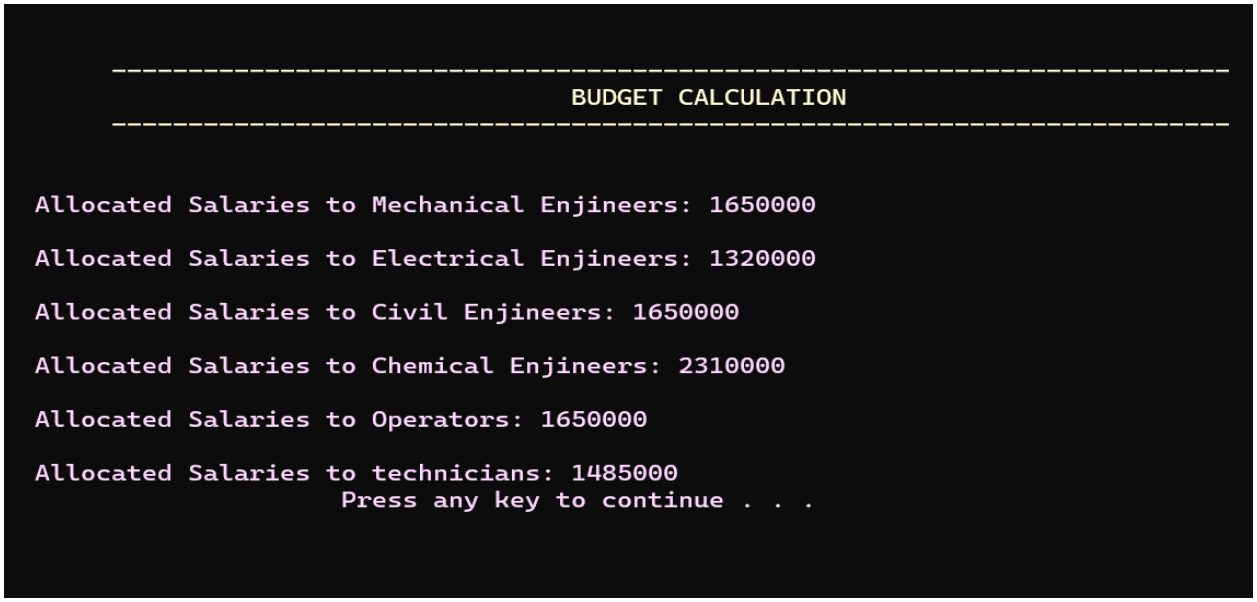


Figure 23: Budget and Cost Control

- **Allocated Resources:**

Allocated resources are shown below:

Figure 24: Allocated Resources

- **Place Orders:**

Orders can be placed by customers:

Figure 25: Place Orders

- **Returns and Refunds:**

Customer can initiate return requests and also viewed them:

Figure 26: Returns and Refunds

- **Notification Menu:**

Customer can add notification, update, delete and search them:

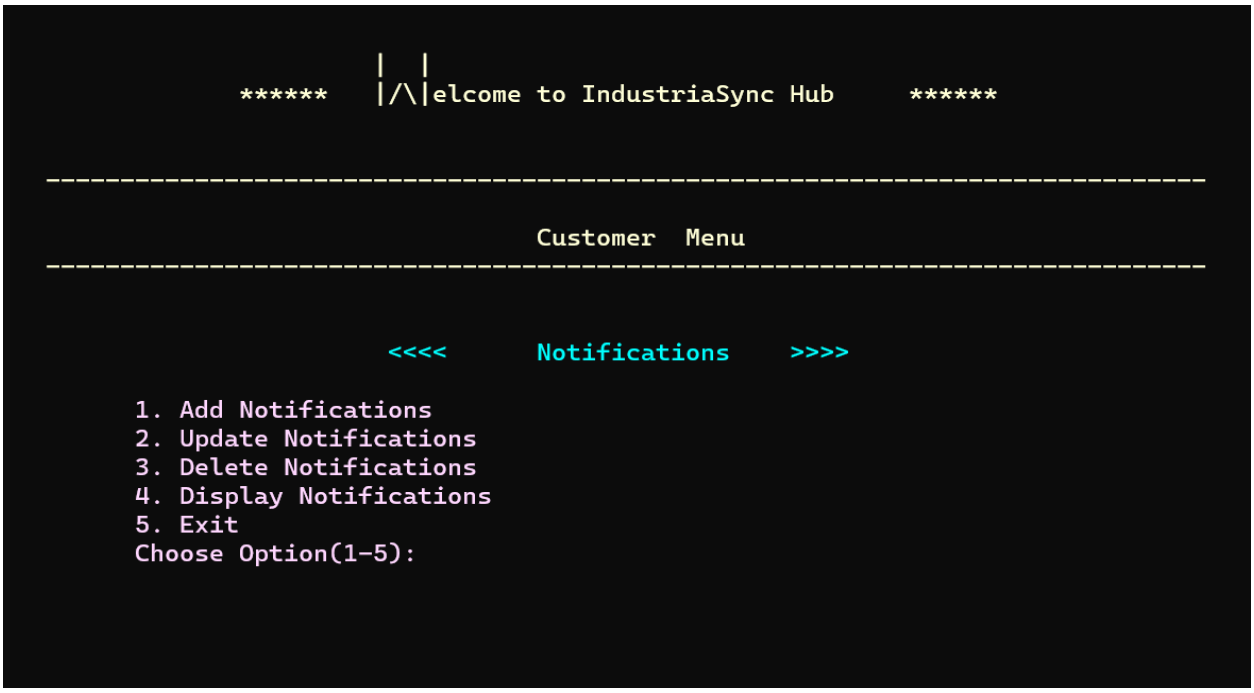


Figure 27: Notification Menu

- **Provide Feedback:**

Customer can provide feedback which is shown to administrator:

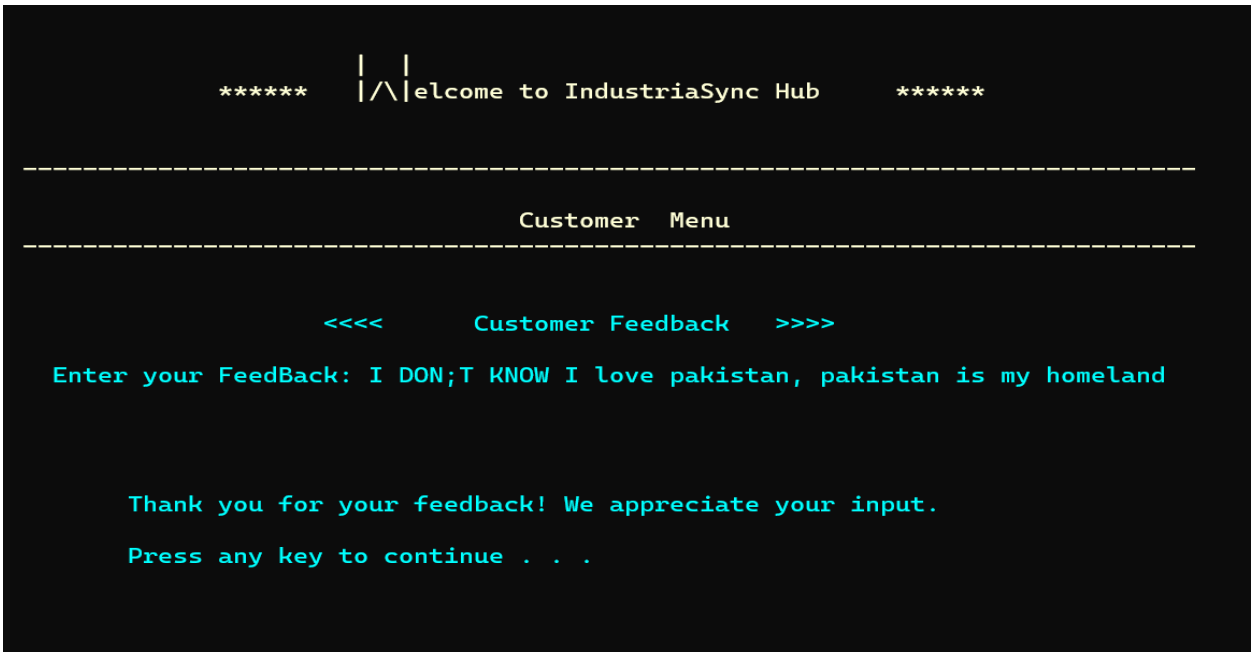


Figure 28: Provide Feedback

- **Customize Order:**

Customer can customize orders:

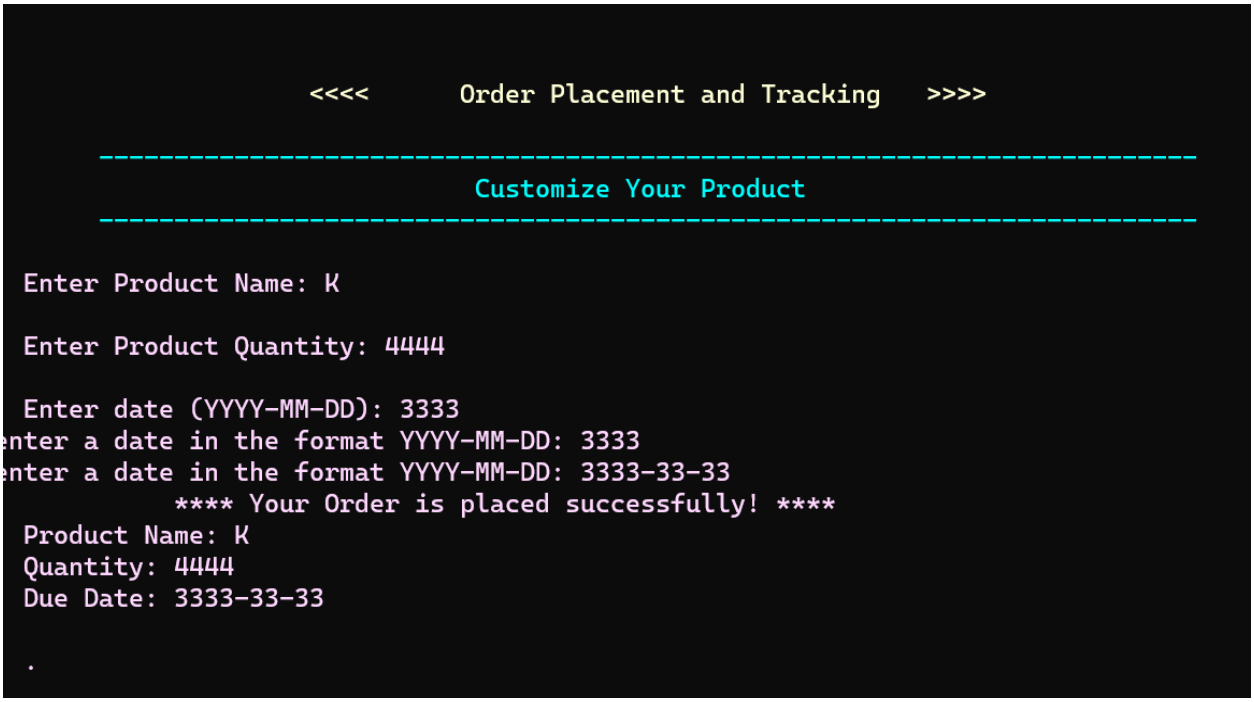


Figure 29: Customize Order

- **Available Products:**

Orders can also be placed on the basis of available products:

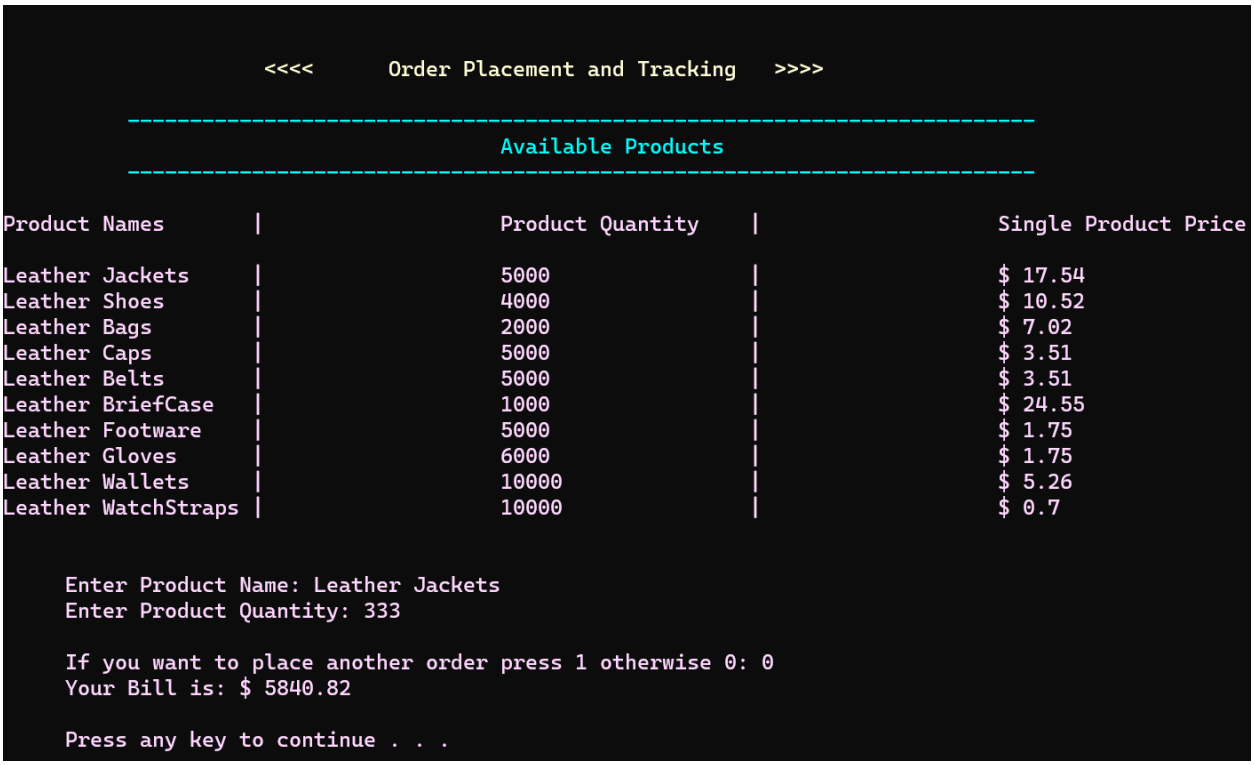
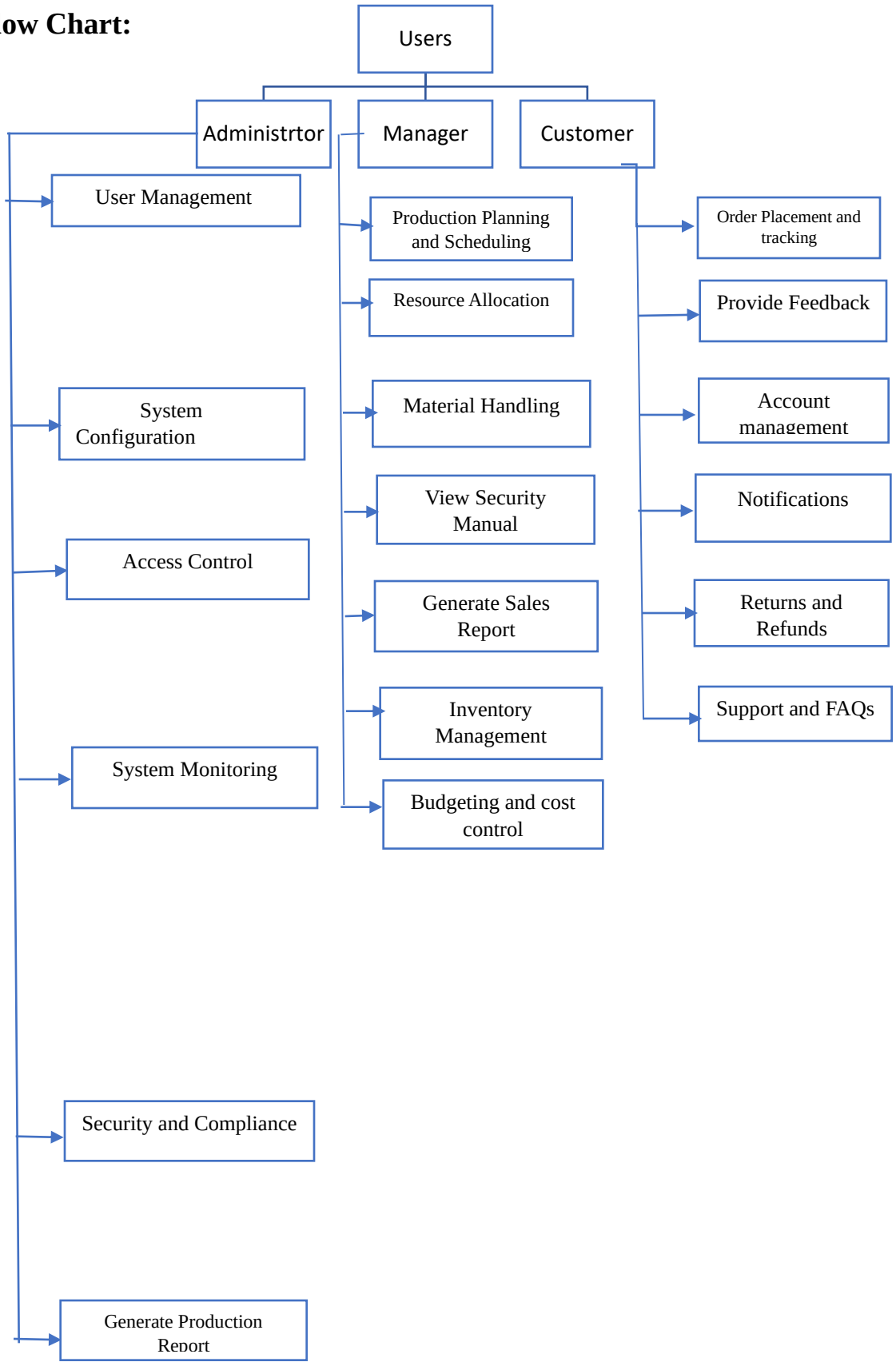


Figure 30: Generate Bill

❖ Flow Chart:


Users
Administrtor
Manager
Customer
User Management
System Configuration
Access Control
System Monitoring
Security and Compliance
Generate Production Report
Production Planning and Scheduling
Resource Allocation
Material Handling
View Security Manual
Generate Sales Report
Inventory Management
Budgeting and cost control
Order Placement and tracking
Provide Feedback
Account management
Notifications
Returns and Refunds
Support and FAQs

❖ Function Prototypes:

IndustriaSync Hub is built by keeping Single Responsibility of Functions in view. Each function is tried best to be independent of other function. Following are the Function Prototypes Used:

- #include<iostream>
- #include<windows.h>
- #include<string>
- #include<fstream>//library for filehandling
- #include<conio.h>
- #include<limits>
- using namespace std;
- void loadInventory(int &ordernumbers,string machines[][5]);
- void saveInventory(int &ordernumbers,string machines[][5]);
- void loadRawMaterials(int &numRawmat,string raw_materials[][5]);
- void saveRawMaterials(int &numRawmat,string raw_materials[][5]);
- void loadWorkers(int &numWorkers,string op_data[][5]);
- void saveWorkers(int &numWorkers,string op_data[][5]);
- void loadOrders(int &ordercount,string orders[][4]);
- void saveOrders(int &ordercount,string orders[][4]);
- string getField(string record, int field);
- bool validity_checker(string num);
- bool ID_validation(string num);
- bool CreditCardNumber_Validations(string num);
- bool Name_Validations(string name);
- bool Salary_Validations(string num);
- bool Contact_Validations(string num);
- void loginsystem(int x1,int y1,string username[],string password[],string userrole[],int usercount,int totalusers,int x,string &feedback,string empname[],int empid[],int empID,int maxrow,string product[],string quantity[],string Alert[],string productNames[],double singleproductPrices[],double productQuantity[],string date[],string Notii[],int p,string orderNumbers[],string productN[],string reasons[],bool processed[],string customerInformation[][11],int MAX_CUSTOMERS,int customerCount,int CUSTOMER_FIELDS,string fields[],string orders[][4],string fieldsOrders[],int MAX_ORDERS,string machines[][5],int totalMac,int &ordernumbers,string op_data[][5],int NUM_OPERATORS,int &numWorkers,int

&allocatednum,string raw_materials[][5],int MAX_Rawmat,int &numRawmat,int &bill,double productPrices[], string saledName[], double saledQuantity[],int &count,string Securitymanual[],int instructions,int &countInstrut,int &mac,int &civil,int &chem,int &oper,int &tech,int &elec,int &lathe,int &MillingMac,int &Drill,int &Bandsaw,int &Grinder,int &Planar,int &corn,int &sago,int &PhosphateRock,int &Ammonia,int &Potash,int &Nitrogen,string num,int &ordercount);

- void mainmenu(int x1,int y1,string username[],string password[],string userrole[],int usercount,int totalusers,int x,string role,string &feedback,string empname[],int empid[],int empID,int maxrow,string product[],string quantity[],string Alert[],string productNames[],double singleproductPrices[],double productQuantity[],string date[],string Notii[],int p,string orderNumbers[],string productN[],string reasons[],bool processed[],string customerInformation[][11],int MAX_CUSTOMERS,int &customerCount,int CUSTOMER_FIELDS,string fields[],string orders[][4],string fieldsOrders[],int MAX_ORDERS,string machines[][5],int totalMac,int &ordernumbers,string op_data[][5],int NUM_OPERATORS,int &numWorkers,int &allocatednum,string raw_materials[][5],int MAX_Rawmat,int &numRawmat, int &bill,double productPrices[], string saledName[], double saledQuantity[],int &count,string Securitymanual[],int instructions,int &countInstrut,int &mac,int &civil,int &chem,int &oper,int &tech,int &elec,int &lathe,int &MillingMac,int &Drill,int &Bandsaw,int &Grinder,int &Planar,int &corn,int &sago,int &PhosphateRock,int &Ammonia,int &Potash,int &Nitrogen,string num,int &ordercount);
- void accesscontrol(int xA,int yA,int x1,int y1,string username[],string password[],string userrole[],int usercount,int totalusers,int x,string &feedback,string empname[],int empid[],int empID,int maxrow,string product[],string quantity[],string Alert[],string productNames[],double singleproductPrices[],double productQuantity[],string date[],string Notii[],int p,string orderNumbers[],string productN[],string reasons[],bool processed[],string customerInformation[][11],int MAX_CUSTOMERS,int customerCount,int CUSTOMER_FIELDS,string fields[],string orders[][4],string fieldsOrders[],int MAX_ORDERS,string machines[][5],int totalMac,int &ordernumbers,string

op_data[][5],int NUM_OPERATORS,int &numWorkers,int &allocatednum,string raw_materials[][5],int MAX_Rawmat,int &numRawmat, int &bill,double productPrices[], string saledName[], double saledQuantity[],int &count,string Securitymanual[],int instructions,int &countInstrut,int &mac,int &civil,int &chem,int &oper,int &tech,int &elec,int &lathe,int &MillingMac,int &Drill,int &Bandsaw,int &Grinder,int &Planar,int &corn,int &sago,int &PhosphateRock,int &Ammonia,int &Potash,int &Nitrogen,string num,int &ordercount);

- void gotoxy(int x,int y);
- void setTextColor(int colorCode);
- void setBackgroundColor(int colorCode);
- void resetColors();
- void title();
- void logintitle();
- string logininterface(int x1,int y1);
- void mainmenutitle();
- void menu(int xA,int yA);
- void administrator();
- void customer();
- string custMenu(int xA,int yA);
- string AdmMenu(int xA,int yA);
- void usermanagement(int xA,int yA,string empname[],int empid[],int empID,int maxrow,string num);
- string usermanagementmenu(int xA,int yA);
- void usermanagementtitle();
- void systemconfiguration(int xA,int yA,string product[],string quantity[],string Alert[],string &feedback,int maxrow,string date[]);
- string systemconfigurationmenu(int xA,int yA);
- void systemconfigurationtitle();
- string accesscontrolmenu(int xA,int yA);
- void accesscontroltitle();
- void accountmanagement(string customerInformation[][11],int MAX_CUSTOMERS,int &customerCount,int CUSTOMER_FIELDS,string fields[],string num);
- string systemmonitoringmenu(int xA,int yA);
- void systemmonitoringtitle();
- void systemmonitoring(int xA,int yA,string op_data[][5],int NUM_OPERATORS,int &numWorkers,string num);
- void viewWorkers(string op_data[][5],int NUM_OPERATORS);

- void addWorkers(string op_data[][5],int NUM_OPERATORS,int &numWorkers,string num);
- void deleteWorkers(string op_data[][5],int NUM_OPERATORS,int &numWorkers,string num);
- void updateWorkers(string op_data[][5],int NUM_OPERATORS,string num,int &numWorkers);
- void searchWorkers(string op_data[][5],int NUM_OPERATORS,string num);
- void OrderPlacementandTracking(int xA,int yA,string productNames[],double singleproductPrices[],double productQuantity[],string product[],string quantity[],string date[],int maxrow,int &bill,double productPrices[], string saledName[], double saledQuantity[],int &count,string num);
- void OrderPlacementandTrackingTitle();
- string OrderPlacementandTrackingmenu(int xA,int yA);
- void viewmenu(string productNames[], double singleproductPrices[], double productQuantity[], int maxrow, int &bill, double productPrices[], string saledName[], double saledQuantity[], int &count,string num);
- void customize(string product[], string quantity[], int maxrow, string date[],string num);
- void providefeedback(string &feedback);
- void Notifications(int xA,int yA,string Notii[],int p,string num);
- void notificationtitle();
- string notificationmenu(int xA,int yA);
- void ReturnsandRefunds(int xA,int yA,string orderNumbers[],string productN[],string reasons[],bool processed[],int maxrow,string num);
- string ReturnsandRefundsmenu(int xA,int yA);
- void ReturnsandRefundstitle();
- string SupportandFAQsmenu(int xA,int yA);
- void Addorders(string orders[][4], string fieldsOrders[], int MAX_ORDERS,string num,int &ordercount);
- void OrderGantt(string orders[][4],string fieldsOrders[],int MAX_ORDERS);
- void SupportandFAQstitle();
- void Addrecord(string empname[],int empid[],int maxrow,string num);
- void UpdateRecord(int maxrow,int &empID,int empid[],string empname[]);
- void DeleteRecord(int &empID,int maxrow,int empid[],string empname[]);

- void searchrecord(int &empID,int maxrow,int empid[],string empname[]);
- void listrecord(int maxrow,int empid[],string empname[]);
- void productionSpeed(int maxrow,string product[],string quantity[],string Alert[],string date[]);
- void alertThreshold(int maxrow,string Alert[]);
- void viewfeedback(string &feedback);
- void addnote(int p,string Notii[]);
- void viewnote(int p,string Notii[]);
- void updatenote(int p,string Notii[],string num);
- void deletenote(int p,string Notii[],string num);
- void initiatereturn(bool processed[],string orderNumbers[],string productN[],string reasons[],int p,string num);
- void viewreturnrequests(bool processed[],string orderNumbers[],string productN[],string reasons[],int p);
- bool isPositiveInteger(const string& str);
- bool isValidDate(const string& date);
- string managermenu(int xA,int yA);
- string operatormenu(int xA,int yA);
- void managers();
- void productionPlanning_Scheduling(int xA,int yA,int maxrow,string product[],string quantity[],string date[],string customerInformation[][11],int MAX_CUSTOMERS,int CUSTOMER_FIELDS,int &customerCount,string fields[],string orders[][4],string fieldsOrders[],int MAX_ORDERS,int &bill,double productPrices[],string saledName[],double saledQuantity[],int &count,string productNames[],double productQuantity[]);
- void productionPlanning_Schedulingtitle();
- string productionschedulingmenu(int xA,int yA);
- void SupportandFAQs(int xA,int yA,string orders[][4],string fieldsOrders[],int MAX_ORDERS,string num,int &ordercount);
- void ViewProduction(int maxrow,string product[],string quantity[],string date[],string customerInformation[][11],int MAX_CUSTOMERS,int CUSTOMER_FIELDS,int &customerCount,string fields[]);
- void ModifyProduction(string orders[][4],string fieldsOrders[],int MAX_ORDERS);
- string InventoryManagementmenu(int xA,int yA);
- void InventoryManagement(int xA,int yA,string machines[][5],int totalMac,int &ordernumbers);
- void InventoryManagementtitle();

- void viewInventory(string machines[][5],int totalMac);
- void deleteInventory(string machines[][5],int totalMac,int &ordernumbers);
- void addInventory(string machines[][5],int totalMac,int &ordernumbers);
- void editInventory(string machines[][5],int totalMac,int &ordernumbers);
- void searchInventory(string machines[][5],int totalMac);
- void ResourceAllocation(int xA,int yA,string machines[][5],int totalMac,string op_data[][5],int NUM_OPERATORS,int maxrow,string product[],string quantity[],string date[],string orders[][4],int MAX_ORDERS,int &allocatednum);
- void resourceallocationtitle();
- string resourceallocationmenu(int xA,int yA);
- void Placedorders(int maxrow,string product[],string quantity[],string date[]);
- void unallocatedresources(string product[],string quantity[],string date[],string orders[][4],int MAX_ORDERS,string op_data[][5],int NUM_OPERATORS,string machines[][5],int totalMac,int &allocatednum);
- void allocatedresources(int maxrow,string product[],string quantity[],string date[],string orders[][4],int MAX_ORDERS,string op_data[][5],int NUM_OPERATORS,string machines[][5],int totalMac,int &allocatednum);
- void addRawmaterials(string raw_materials[][5],int MAX_Rawmat,int &numRawmat);
- void deleteRawmaterials(string raw_materials[][5],int MAX_Rawmat,int &numRawmat);
- void editRawmaterials(string raw_materials[][5],int MAX_Rawmat,int &numRawmat);
- void searchRawmaterials(string raw_materials[][5],int MAX_Rawmat);
- void viewRawMaterials(string raw_materials[][5],int MAX_Rawmat);
- void MaterialHandling(int xA,int yA,string raw_materials[][5],int MAX_Rawmat,int &numRawmat);
- void MaterialHandlingtitle();
- string MaterialHandlingmenu(int xA,int yA);
- void GenerateReport(int &bill,double productPrices[],string saledName[],double saledQuantity[],int maxrow,int &count,string productNames[],double productQuantity[]);

- void securityandcomplianceManual(int xA,int yA,string Securitymanual[],int instructions,int &countInstrut,string num);
- string securityandcompliancemenu(int xA,int yA);
- void addmanual(string Securitymanual[],int instructions,int &countInstrut);
- void viewmanual(string Securitymanual[],int instructions,int &countInstrut);
- void securityandcompliancetitle();
- void deleteinstructions(string Securitymanual[],int instructions,int &countInstrut,string num);
- void GenerateProductionReports(int &bill,double productPrices[],string saledName[],double saledQuantity[],int maxrow,int &count,string productNames[],double productQuantity[]);
- void BudgetingandCostControl(int xA,int yA,int &mac,int &civil,int &chem,int &oper,int &tech,int &elec,int &lathe,int &MillingMac,int &Drill,int &Bandsaw,int &Grinder,int &Planar,int &corn,int &sago,int &PhosphateRock,int &Ammonia,int &Potash,int &Nitrogen,string num);
- void budgetcalculationWorkers(int &mac,int &civil,int &chem,int &oper,int &tech,int &elec);
- void InputdataWorkers(int &mac,int &civil,int &chem,int &oper,int &tech,int &elec,string num);
- void InputdataRaw(int &corn,int &sago,int &PhosphateRock,int &Ammonia,int &Potash,int &Nitrogen,string num);
- void budgetcalculationRaw(int &corn,int &sago,int &PhosphateRock,int &Ammonia,int &Potash,int &Nitrogen);
- void InputdataInventory(int &lathe,int &MillingMac,int &Drill,int &Bandsaw,int &Grinder,int &Planar,string num);
- void budgetcalculationInventory(int &lathe,int &MillingMac,int &Drill,int &Bandsaw,int &Grinder,int &Planar);
- string BudgetingandCostControlmenu(int xA,int yA);
- void BudgetingandCostControltitle();

❖ Weakness:

In IndustriaSync Hub, switching between double (for decimal numbers) and int (for whole numbers) can cause problems because converting from double to int may lose some decimal parts. This means you might not get the exact number you expect.

❖ Future Directions:

I want to implement Navigation system for choosing option and making the application more user-friendly by reducing the amount of typed inputs required. Also, I Aim to use GUI to make CrossFit Studios market available as current demand of market is GUI based applications. I want to add graphs but due to shortage of time, it would not happen.

Student Reg. No. : **Student Name.**

	A-Extensive Evidence	**B-Convincing Evidence**	**C-Limited Evidence**	**D-No Evidence**
Documentation Formatting **Grade:**	All the documentation meets all the criteria.	Documentation is well formatted but some of the criteria is not fulfilled.	Documentation is required a lot of improvement.	Documentation is not Available
Documentation Formatting Criteria: In **Binder**, **Title** Page, **Header**-Footers, Font **Style**, Font **Size** all are all consistence and according to given **guidelines**. Project **Poster** is professionally design and well presented				
Documentation Contents **Grade:**	Documentation includes all of the criteria.	Documentation meet more than 80% of the criteria given.	Documentation meet more than 50% of the criteria.	When the documentation meet less than 50% of the criteria.
Documentation Contents Criteria: **Title** Page - **Table** of Contents - Project **Abstract** - **Functional** Requirements - **Wire** Frames –**Data Flow** Diagram-**Data** Structure (Arrays)-**Function** Headers and Description -Project **Code.** - **Weakness** in the Project and **Future** Directions. - **Conclusion** and What your **Learn** from the Project and Course and What is your **Future** Planning.				
Project Complexity **Grade:**	Project has at least 2 user's types and each user has at least 5 functionalities.	Project complexity meet 80% criteria given in extensive evidence	Project complexity meet 50% criteria given in extensive evidence	Project complexity meet less than 50% criteria given in extensive evidence
Code Style **Grade:**	All Code style criteria is followed	All code style criteria followed but some improvements required	lot of improvements required in coding style.	**Did not follow** code style,
Code Style Criteria: Consistent code style. Code is well indented. Variable and Function names are well defined. White Spaces are well used. Comments are added.				
Code Documentation Mapping **Grade:**	Code and documentation is synchronized.	Code and documentation does not synchronized at **some** places	Code and documentation does not synchronized at **many** places	Code and documentation **does not** synchronized.
Data Structure (Arrays) **Grade:**	Data structure is sufficient for the project requirements	Data Structure is sufficient but require improvement to meet project requirements.	Data structure is not sufficient and need a lot of improvement	Data Structure is not properly identified and declared.
Modularity **Grade:**	Meet all Modularity criteria	Meet all Modularity criteria but at some places it is missing	Do not sufficiently meet the modularity criteria.	No modularity or very minimum modularity.
Modularity criteria: Functions are defined for each major feature. Functions are independent (identify from parameter list and return types).				
Validations **Grade:**	Validations on all number type inputs are applied	Validations are applied but at some places it is missing.	Validations are missing at lot of places	No Validations are used
File Handling **Grade:**	Separate files for separate data. Data in csv format	File handing require some improvements	File handing require a lot of improvements	Not implemented
Aesthetics of the User Interface **Grade:**	UI is presentable. Proper coloring, Headers and clear screen is done	UI require some improvements	UI require a lot of improvements	Not implemented
Presentation and Demo **Grade:**	Presentation and Demo was 100% working	Presentation and Demo require some improvements	Presentation and Demo require a lot of improvements	Presentation was not ok and Demo was not working
Student Understanding with the Code. **Grade:**	Student has complete understanding how the code is working and knows the concept.	Student has good understand but some place he does not know the concepts	Student has a very little understand and lack the major concepts.	Student does not have any level of understanding of the code.

Checked by:	
Comments:	

Code:

```
#include<iostream>
#include<windows.h>
#include<string>
#include<fstream>
#include<conio.h>
#include<limits>
using namespace std;
void loadInventory(int &ordernumbers,string machines[][5]);
void saveInventory(int &ordernumbers,string machines[][5]);
void loadRawMaterials(int &numRawmat,string raw_materials[][5]);
void saveRawMaterials(int &numRawmat,string raw_materials[][5]);
void loadWorkers(int &numWorkers,string op_data[][5]);
void saveWorkers(int &numWorkers,string op_data[][5]);
void loadOrders(int &ordercount,string orders[][4]);
void saveOrders(int &ordercount,string orders[][4]);
string getField(string record, int field);
bool validity_checker(string num);
bool ID_validation(string num);
bool CreditCardNumber_Validations(string num);
bool Name_Validations(string name);
bool Salary_Validations(string num);
bool Contact_Validations(string num);
void loginsystem(int x1,int y1,string username[],string password[],string userrole[],int
usercount,int totalusers,int x,string &feedback,string empname[],int empid[],int empID,int
maxrow,string product[],string quantity[],string Alert[],string productNames[],double
singleproductPrices[],double productQuantity[],string date[],string Notii[],int p,string
orderNumbers[],string productN[],string reasons[],bool processed[],string
customerInformation[][11],int MAX_CUSTOMERS,int customerCount,int
CUSTOMER_FIELDS,string fields[],string orders[][4],string fieldsOrders[],int
MAX_ORDERS,string machines[][5],int totalMac,int &ordernumbers,string op_data[][5],int
```

NUM_OPERATORS,int &numWorkers,int &allocatednum,string raw_materials[][5],int MAX_Rawmat,int &numRawmat,int &bill,double productPrices[], string saledName[], double saledQuantity[],int &count,string Securitymanual[],int instructions,int &countInstrut,int &mac,int &civil,int &chem,int &oper,int &tech,int &elec,int &lathe,int &MillingMac,int &Drill,int &Bandsaw,int &Grinder,int &Planar,int &corn,int &sago,int &PhosphateRock,int &Ammonia,int &Potash,int &Nitrogen,string num,int &ordercount);

void mainmenu(int x1,int y1,string username[],string password[],string userrole[],int usercount,int totalusers,int x,string role,string &feedback,string empname[],int empid[],int empID,int maxrow,string product[],string quantity[],string Alert[],string productNames[],double singleproductPrices[],double productQuantity[],string date[],string Notii[],int p,string orderNumbers[],string productN[],string reasons[],bool processed[],string customerInformation[][11],int MAX_CUSTOMERS,int &customerCount,int CUSTOMER_FIELDS,string fields[],string orders[][4],string fieldsOrders[],int MAX_ORDERS,string machines[][5],int totalMac,int &ordernumbers,string op_data[][5],int NUM_OPERATORS,int &numWorkers,int &allocatednum,string raw_materials[][5],int MAX_Rawmat,int &numRawmat, int &bill,double productPrices[], string saledName[], double saledQuantity[],int &count,string Securitymanual[],int instructions,int &countInstrut,int &mac,int &civil,int &chem,int &oper,int &tech,int &elec,int &lathe,int &MillingMac,int &Drill,int &Bandsaw,int &Grinder,int &Planar,int &corn,int &sago,int &PhosphateRock,int &Ammonia,int &Potash,int &Nitrogen,string num,int &ordercount);

void accesscontrol(int xA,int yA,int x1,int y1,string username[],string password[],string userrole[],int usercount,int totalusers,int x,string &feedback,string empname[],int empid[],int empID,int maxrow,string product[],string quantity[],string Alert[],string productNames[],double singleproductPrices[],double productQuantity[],string date[],string Notii[],int p,string orderNumbers[],string productN[],string reasons[],bool processed[],string customerInformation[][11],int MAX_CUSTOMERS,int customerCount,int CUSTOMER_FIELDS,string fields[],string orders[][4],string fieldsOrders[],int MAX_ORDERS,string machines[][5],int totalMac,int &ordernumbers,string op_data[][5],int NUM_OPERATORS,int &numWorkers,int &allocatednum,string raw_materials[][5],int MAX_Rawmat,int &numRawmat, int &bill,double productPrices[], string saledName[], double saledQuantity[],int &count,string Securitymanual[],int instructions,int &countInstrut,int &mac,int &civil,int &chem,int &oper,int &tech,int &elec,int &lathe,int &MillingMac,int &Drill,int &Bandsaw,int &Grinder,int &Planar,int &corn,int &sago,int &PhosphateRock,int &Ammonia,int &Potash,int &Nitrogen,string num,int &ordercount);

void gotoxy(int x,int y);

void setTextColor(int colorCode);

void setBackgroundColor(int colorCode);

void resetColors();

void title();

```
void logintitle();

string logininterface(int x1,int y1);

void mainmenutitle();

void menu(int xA,int yA);

void administrator();

void customer();

string custMenu(int xA,int yA);

string AdmMenu(int xA,int yA);

void usermanagement(int xA,int yA,string empname[],int empid[],int empID,int maxrow,string
num);

string usermanagementmenu(int xA,int yA);

void usermanagementtitle();

void systemconfiguration(int xA,int yA,string product[],string quantity[],string Alert[],string
&feedback,int maxrow,string date[]);

string systemconfigurationmenu(int xA,int yA);

void systemconfigurationtitle();

string accesscontrolmenu(int xA,int yA);

void accesscontroltitle();

void accountmanagement(string customerInformation[][11],int MAX_CUSTOMERS,int
&customerCount,int CUSTOMER_FIELDS,string fields[],string num);

 string systemmonitoringmenu(int xA,int yA);

void systemmonitoringtitle();

void systemmonitoring(int xA,int yA,string op_data[][5],int NUM_OPERATORS,int
&numWorkers,string num);

void viewWorkers(string op_data[][5],int NUM_OPERATORS);

void addWorkers(string op_data[][5],int NUM_OPERATORS,int &numWorkers,string num);

void deleteWorkers(string op_data[][5],int NUM_OPERATORS,int &numWorkers,string num);

void updateWorkers(string op_data[][5],int NUM_OPERATORS,string num,int &numWorkers);

void searchWorkers(string op_data[][5],int NUM_OPERATORS,string num);
```

void OrderPlacementandTracking(int xA,int yA,string productNames[],double singleproductPrices[],double productQuantity[],string product[],string quantity[],string date[],int maxrow,int &bill,double productPrices[], string saledName[], double saledQuantity[],int &count,string num);

void OrderPlacementandTrackingTitle();

string OrderPlacementandTrackingmenu(int xA,int yA);

void viewmenu(string productNames[], double singleproductPrices[], double productQuantity[], int maxrow, int &bill, double productPrices[], string saledName[], double saledQuantity[], int &count,string num);

void customize(string product[], string quantity[], int maxrow, string date[],string num);

void providefeedback(string &feedback);

void Notifications(int xA,int yA,string Notii[],int p,string num);

void notificationtitle();

string notificationmenu(int xA,int yA);

void ReturnsandRefunds(int xA,int yA,string orderNumbers[],string productN[],string reasons[],bool processed[],int maxrow,string num);

string ReturnsandRefundsmenu(int xA,int yA);

void ReturnsandRefundstitle();

string SupportandFAQsmenu(int xA,int yA);

void Addorders(string orders[][4], string fieldsOrders[], int MAX_ORDERS,string num,int &ordercount);

void OrderGantt(string orders[][4],string fieldsOrders[],int MAX_ORDERS);

void SupportandFAQstitle();

void Addrecord(string empname[],int empid[],int maxrow,string num);

void UpdateRecord(int maxrow,int &empID,int empid[],string empname[]);

void DeleteRecord(int &empID,int maxrow,int empid[],string empname[]);

void searchrecord(int &empID,int maxrow,int empid[],string empname[]);

void listrecord(int maxrow,int empid[],string empname[]);

void productionSpeed(int maxrow,string product[],string quantity[],string Alert[],string date[]);

void alertThreshold(int maxrow,string Alert[]);

void viewfeedback(string &feedback);

```
void addnote(int p,string Notii[]);

void viewnote(int p,string Notii[]);

void updatenote(int p,string Notii[],string num);

void deletenote(int p,string Notii[],string num);

void initiatereturn(bool processed[],string orderNumbers[],string productN[],string reasons[],int p,string num);

void viewreturnrequests(bool processed[],string orderNumbers[],string productN[],string reasons[],int p);

bool isPositiveInteger(const string& str);

bool isValidDate(const string& date);

string managermenu(int xA,int yA);

string operatormenu(int xA,int yA);

void managers();

void productionPlanning_Scheduling(int xA,int yA,int maxrow,string product[],string quantity[],string date[],string customerInformation[][11],int MAX_CUSTOMERS,int CUSTOMER_FIELDS,int &customerCount,string fields[],string orders[][4],string fieldsOrders[],int MAX_ORDERS,int &bill,double productPrices[],string saledName[],double saledQuantity[],int &count,string productNames[],double productQuantity[]);

void productionPlanning_Schedulingtitle();

string productionschedulingmenu(int xA,int yA);

void SupportandFAQs(int xA,int yA,string orders[][4],string fieldsOrders[],int MAX_ORDERS,string num,int &ordercount);

void ViewProduction(int maxrow,string product[],string quantity[],string date[],string customerInformation[][11],int MAX_CUSTOMERS,int CUSTOMER_FIELDS,int &customerCount,string fields[]);

void ModifyProduction(string orders[][4],string fieldsOrders[],int MAX_ORDERS);

string InventoryManagementmenu(int xA,int yA);

void InventoryManagement(int xA,int yA,string machines[][5],int totalMac,int &ordernumbers);

void InventoryManagementtitle();

void viewInventory(string machines[][5],int totalMac);

void deleteInventory(string machines[][5],int totalMac,int &ordernumbers);
```

void addInventory(string machines[][5],int totalMac,int &ordernumbers);

void editInventory(string machines[][5],int totalMac,int &ordernumbers);

void searchInventory(string machines[][5],int totalMac);

void ResourceAllocation(int xA,int yA,string machines[][5],int totalMac,string op_data[][5],int NUM_OPERATORS,int maxrow,string product[],string quantity[],string date[],string orders[][4],int MAX_ORDERS,int &allocatednum);

void resourceallocationtitle();

string resourceallocationmenu(int xA,int yA);

void Placedorders(int maxrow,string product[],string quantity[],string date[]);

void unallocatedresources(string product[],string quantity[],string date[],string orders[][4],int MAX_ORDERS,string op_data[][5],int NUM_OPERATORS,string machines[][5],int totalMac,int &allocatednum);

void allocatedresources(int maxrow,string product[],string quantity[],string date[],string orders[][4],int MAX_ORDERS,string op_data[][5],int NUM_OPERATORS,string machines[][5],int totalMac,int &allocatednum);

void addRawmaterials(string raw_materials[][5],int MAX_Rawmat,int &numRawmat);

void deleteRawmaterials(string raw_materials[][5],int MAX_Rawmat,int &numRawmat);

void editRawmaterials(string raw_materials[][5],int MAX_Rawmat,int &numRawmat);

void searchRawmaterials(string raw_materials[][5],int MAX_Rawmat);

void viewRawMaterials(string raw_materials[][5],int MAX_Rawmat);

void MaterialHandling(int xA,int yA,string raw_materials[][5],int MAX_Rawmat,int &numRawmat);

void MaterialHandlingtitle();

string MaterialHandlingmenu(int xA,int yA);

void GenerateReport(int &bill,double productPrices[],string saledName[],double saledQuantity[],int maxrow,int &count,string productNames[],double productQuantity[]);

void securityandcomplianceManual(int xA,int yA,string Securitymanual[],int instructions,int &countInstrut,string num);

string securityandcompliancemenu(int xA,int yA);

void addmanual(string Securitymanual[],int instructions,int &countInstrut);

void viewmanual(string Securitymanual[],int instructions,int &countInstrut);

```
void securityandcompliancetitle();

void deleteinstructions(string Securitymanual[],int instructions,int &countInstrut,string num);

void GenerateProductionReports(int &bill,double productPrices[],string saledName[],double
saledQuantity[],int maxrow,int &count,string productNames[],double productQuantity[]);

void BudgetingandCostControl(int xA,int yA,int &mac,int &civil,int &chem,int &oper,int
&tech,int &elec,int &lathe,int &MillingMac,int &Drill,int &Bandsaw,int &Grinder,int
&Planar,int &corn,int &sago,int &PhosphateRock,int &Ammonia,int &Potash,int
&Nitrogen,string num);

void budgetcalculationWorkers(int &mac,int &civil,int &chem,int &oper,int &tech,int &elec);

void InputdataWorkers(int &mac,int &civil,int &chem,int &oper,int &tech,int &elec,string
num);

void InputdataRaw(int &corn,int &sago,int &PhosphateRock,int &Ammonia,int &Potash,int
&Nitrogen,string num);

void budgetcalculationRaw(int &corn,int &sago,int &PhosphateRock,int &Ammonia,int
&Potash,int &Nitrogen);

void InputdataInventory(int &lathe,int &MillingMac,int &Drill,int &Bandsaw,int &Grinder,int
&Planar,string num);

void budgetcalculationInventory(int &lathe,int &MillingMac,int &Drill,int &Bandsaw,int
&Grinder,int &Planar);

string BudgetingandCostControlmenu(int xA,int yA);

void BudgetingandCostControltitle();

main()

{

   string num;

   int x1=30,y1=16;

   const int totalusers=100;

   int usercount=0;

   int x=0;

   string username[totalusers]={};

   string password[totalusers]={};

   string userrole[totalusers]={};
```

```
    string feedback={};
    const int maxrow=100;
    string empname[maxrow]={};
    int empid[maxrow]={0};
    int empID=0;
    string product[maxrow]={};
    string quantity[maxrow]={};
    string Alert[maxrow]={};
    string date[maxrow]={};
    string productNames[maxrow] = {"Leather Jackets", "Leather Shoes", "Leather Bags", "Leather Caps", "Leather Belts","Leather BriefCase","Leather Footware","Leather Gloves","Leather Wallets","Leather WatchStraps"};
    double singleproductPrices[maxrow] = {17.54,10.52,7.02,3.51,3.51,24.55,1.75,1.75,5.26,0.70};
    double productQuantity[maxrow]={5000,4000,2000,5000,5000,1000,5000,6000,10000,10000};
    double productPrices[maxrow]={};
    string saledName[maxrow]={};
    double saledQuantity[maxrow]={0};
    int bill=0;
    const int p=10;
    string Notii[p]={};
    string orderNumbers[maxrow]={};
    string productN[maxrow]={};
    string reasons[maxrow]={};
    bool processed[maxrow]={false};
    const int MAX_CUSTOMERS = 100; // Maximum number of customers, adjust as needed
    const int CUSTOMER_FIELDS = 11; // Number of customer information fields
    string customerInformation[MAX_CUSTOMERS][CUSTOMER_FIELDS];
    int customerCount = 0;
```

```
    string fields[11]={"Name: ","Address: ","Phone Number: ","Email: ","Credit Card Number: ","Credit Card Expiry Date: ","Credit Card CVV: ","Account Type: ","Account Limit: ","Account Balance: ","Account Status: "};

     const int MAX_ORDERS = 100;

     int ordercount=0;

    string fieldsOrders[4]={"Product Name","Product Quantity","Expected Date","Specifications"};

    string orders[MAX_ORDERS][4];

  loadOrders(ordercount,orders);//loading orders from orders file

    string Macfields[5]={"Machine ID","Machine Type","Machine Name","Machine Specification","Working"};

    const int totalMac=100;

    int ordernumbers=0;

    string machines[totalMac][5];

    loadInventory(ordernumbers,machines);//loading inventory from machines file

    int numWorkers=0;

    string op_array[5]={"Operator Name","Operator ID","Operator Ranks","Operator Skills","Operator Salary"};

    const int NUM_OPERATORS = 100;

    string op_data[NUM_OPERATORS][5];

    loadWorkers(numWorkers,op_data);//loading workers from workers file

 int allocatednum=0;

 const int MAX_Rawmat=100;

    int numRawmat=0;

    string raw_Mat[5]={"Starch Raw","Fert Raw","Inventory Quantity","Location","Suppliers"};

    string raw_materials[MAX_Rawmat][5];

     loadRawMaterials(numRawmat,raw_materials);//loading raw materials from raw materials file

     int count=0;

        int countInstrut=11;
```

```
    const int instructions=20;
    string Securitymanual[instructions]={"All employees must wear ID badges at all times.",
                        "Visitors must sign in at the front desk and be escorted by an employee.",
                        "Security cameras must be monitored at all entrances and restricted areas.",
                         "Hazardous materials must be locked in secure storage rooms with restricted
access.",
                         "All equipment must undergo routine safety inspections.",
                          "Proper protective gear must be worn in all hazardous areas.",
                           "Any injuries/accidents must be immediately reported to the safety
manager.",
                          "All employees must complete workplace and equipment safety training.",
                         "Food and drink only permitted in designated break areas away from
equipment.",
                         "Emergency evacuation drills must be conducted every 3 months.",
                        "Proper safety guards must be in place on machinery at all times."
                        };
int mac,elec,chem,tech,oper,civil=0;
int lathe,MillingMac,Drill,Bandsaw,Grinder,Planar=0;
int corn,sago,PhosphateRock,Ammonia,Potash,Nitrogen=0;
loginsystem(x1,y1,username,password,userrole,usercount,totalusers,x,feedback,empname,empid,
empID,maxrow,product,quantity,Alert,productNames,singleproductPrices,productQuantity,date,
Notii,p,orderNumbers,productN,reasons,processed,customerInformation,MAX_CUSTOMERS,c
ustomerCount,CUSTOMER_FIELDS,fields,orders,fieldsOrders,MAX_ORDERS,machines,total
Mac,ordernumbers,op_data,NUM_OPERATORS,numWorkers,allocatednum,raw_materials,MA
X_Rawmat,numRawmat,bill,productPrices,saledName,saledQuantity,count,Securitymanual,instr
uctions,countInstrut,mac,civil,chem,oper,tech,elec,lathe,MillingMac,Drill,Bandsaw,Grinder,Plan
ar,corn,sago,PhosphateRock,Ammonia,Potash,Nitrogen,num,ordercount);
}
string getField(string record, int field)
{
int commaCount = 1;
string item;
```

```
for (int x = 0; x < record.length(); x++)
{
if (record[x] == ';')
{
commaCount = commaCount + 1;
}
else if (commaCount == field)
{
item = item + record[x];
}
}
return item;
}
void loadInventory(int &ordernumbers,string machines[][5])
{
  fstream file;
  string row;
  file.open("machines.txt", ios::in);
  ordernumbers=0;
  while(!file.eof())
  {
   getline(file, row);
   machines[ordernumbers][0]=getField(row,1);
   machines[ordernumbers][1]=getField(row,2);
   machines[ordernumbers][2]=getField(row,3);
   machines[ordernumbers][3]=getField(row,4);
   machines[ordernumbers][4]=getField(row,5);
   ordernumbers++;
```

```
   }
}
void saveInventory(int &ordernumbers,string machines[][5])
{
    string table="";
    for (int i = 0; i < ordernumbers; i++)
    {
        string row=machines[i][0] + ";" + machines[i][1] + ";" + machines[i][2] + ";" +
machines[i][3] + ";" + machines[i][4];
        table = table + row;
        if (i + 1 != ordernumbers) table = table + "\n";
    }
    fstream file;
    file.open("machines.txt", ios::out);
    file << table;
    file.close();

}
void loadRawMaterials(int &numRawmat,string raw_materials[][5])
{
     fstream file;
   string row;
   file.open("RawMaterials.txt", ios::in);
    numRawmat=0;
   while(!file.eof())
   {
    getline(file, row);
    raw_materials[numRawmat][0]=getField(row,1);
    raw_materials[numRawmat][1]=getField(row,2);
```

```
    raw_materials[numRawmat][2]=getField(row,3);
    raw_materials[numRawmat][3]=getField(row,4);
    raw_materials[numRawmat][4]=getField(row,5);
    numRawmat++;
}
}
void saveRawMaterials(int &numRawmat,string raw_materials[][5])
{
    string table="";
    for (int i = 0; i <  numRawmat; i++)
    {
        string row=raw_materials[i][0] + ";" + raw_materials[i][1] + ";" + raw_materials[i][2] + ";"
+ raw_materials[i][3] + ";" + raw_materials[i][4];
        table = table + row;
        if (i + 1 !=  numRawmat) table = table + "\n";
    }
    fstream file;
    file.open("RawMaterials.txt", ios::out);
    file << table;
    file.close();
}
void loadWorkers(int &numWorkers,string op_data[][5])
{
   fstream file;
   string row;
   file.open("Workers.txt", ios::in);
   numWorkers=0;
   while(!file.eof())
   {
```

```
    getline(file, row);
    op_data[numWorkers][0]=getField(row,1);
    op_data[numWorkers][1]=getField(row,2);
    op_data[numWorkers][2]=getField(row,3);
    op_data[numWorkers][3]=getField(row,4);
    op_data[numWorkers][4]=getField(row,5);
     numWorkers++;
   }


}
void saveWorkers(int &numWorkers,string op_data[][5])
{
    string table="";
    for (int i = 0; i < numWorkers; i++)
    {
        string row=op_data[i][0] + ";" + op_data[i][1] + ";" + op_data[i][2] + ";" + op_data[i][3] +
";" + op_data[i][4];
        table = table + row;
        if (i + 1 != numWorkers) table = table + "\n";
    }
    ofstream file("Workers.txt", ios::out);  // Open the file directly in the constructor


    if (!file.is_open())
    {
        cout << "Error: File not found." << endl;
    }
    else
    {
        file << table;
```

```
        file.close();
    }
}
void loadOrders(int &ordercount,string orders[][4])
{
   fstream file;
   string row;
   file.open("Orders.txt", ios::in);
   ordercount=0;
   while(!file.eof())
   {
    getline(file, row);
    orders[ordercount][0]=getField(row,1);
    orders[ordercount][1]=getField(row,2);
    orders[ordercount][2]=getField(row,3);
    orders[ordercount][3]=getField(row,4);
     ordercount++;
   }

}
void saveOrders(int &ordercount,string orders[][4])
{
    string table="";
    for (int i = 0; i < ordercount; i++)
    {
        string row=orders[i][0] + ";" + orders[i][1] + ";" + orders[i][2] + ";" + orders[i][3];
        table = table + row;
        if (i + 1 != ordercount) table = table + "\n";
```

```
    }
    ofstream file("Orders.txt", ios::out);  // Open the file directly in the constructor

    if (!file.is_open())
    {
        cout << "Error: File not found." << endl;
    }
    else
    {
        file << table;
        file.close();
    }
}

bool validity_checker(string num)
{
    int count_1 = 0, count_2 = 0;
    for (int i = 0; num[i] != '\0'; i++)
    {
        count_1++;
    }
    for (int j = 0; num[j] != '\0'; j++)
    {
        if (num[j] >= '0' && num[j] <= '9')
        {
            count_2++;
        }
    }
```

```
    if (count_1 == count_2)
    {
        return true;
    }
    else
    {
         return false;
    }
}
bool Name_Validations(string name)
{
    for(int i = 0; i < name.length(); ++i)
    {
        if(!isalpha(name[i]))
        {
            return false;
        }
    }

    return true;
}
bool CreditCardNumber_Validations(string num)
{
     int count_1=0,count_2=0;
    for(int i=0 ;num[i]!= '\0'; i++)
    {
        count_1++;
    }
```

```
    for(int j=0;num[j]!='\0';j++)
    {
        if(num[j]>='0' && num[j]<= '9')
         {count_2++;}
    }
    if ((count_1==count_2)&&(count_1==16))
    {
        return true;
    }
    else
    {
        return false;

    }
}
 bool ID_validation(string num)
{
     int count_1=0,count_2=0;
    for(int i=0 ;num[i]!= '\0'; i++)
    {
        count_1++;
    }
    for(int j=0;num[j]!='\0';j++)
    {
        if(num[j]>='0' && num[j]<= '9')
         {count_2++;}
    }
    if ((count_1==count_2)&&(count_1==5))
```

```
    {
        return true;
    }
    else
    {
        return false;

    }
}
  bool Contact_Validations(string num)
{
     int count_1=0,count_2=0;
    for(int i=0 ;num[i]!= '\0'; i++)
    {

        count_1++;

    }
    for(int j=0;num[j]!='\0';j++)
    {
        if(num[j]>='0' && num[j]<= '9')
         {count_2++;}
    }
    if ((count_1==count_2)&&(count_1==11) && (num[0]=='0') && (num[1]=='3'))
    {
        return true;
    }
    else
```

```
    {
        return false;

    }
}

bool Salary_Validations(string num)
{
     int count_1=0,count_2=0;
    for(int i=0 ;num[i]!= '\0'; i++)
    {
        count_1++;
    }
    for(int j=0;num[j]!='\0';j++)
    {
        if(num[j]>='0' && num[j]<= '9')
         {count_2++;}
    }
    if ((count_1==count_2)&&(count_1<=6))
    {
        return true;
    }
    else
    {
        return false;

    }
}
```

```
bool isPositiveInteger(const string& str)
{
    for (char ch : str)
    {
        if (!isdigit(ch) || ch == '0')
        {
            return false;
        }
    }
    return true;
}
bool isValidDate(const string& date)
{
    if (date.length() != 10)
    {
        return false;
    }

    for (int i = 0; i < 10; ++i)
     {
        if (i == 4 || i == 7)
        {
            if (date[i] != '-')
            {
                return false;
            }
        }
        else if (!isdigit(date[i]))
```

```
        {
            return false;
        }
    }
    return true;
}

void setTextColor(int colorCode)
{
  std::cout << "\033[38;5;" << colorCode << "m";
}
void setBackgroundColor(int colorCode)
{
  std::cout << "\033[48;5;" << colorCode << "m";
}
void resetColors()
{
  std::cout << "\033[0m";
}

void loginsystem(int x1,int y1,string username[],string password[],string userrole[],int
usercount,int totalusers,int x,string &feedback,string empname[],int empid[],int empID,int
maxrow,string product[],string quantity[],string Alert[],string productNames[],double
singleproductPrices[],double productQuantity[],string date[],string Notii[],int p,string
orderNumbers[],string productN[],string reasons[],bool processed[],string
customerInformation[][11],int MAX_CUSTOMERS,int customerCount,int
CUSTOMER_FIELDS,string fields[],string orders[][4],string fieldsOrders[],int
MAX_ORDERS,string machines[][5],int totalMac,int &ordernumbers,string op_data[][5],int
NUM_OPERATORS,int &numWorkers,int &allocatednum,string  raw_materials[][5],int
MAX_Rawmat,int &numRawmat, int &bill,double productPrices[], string saledName[], double
saledQuantity[],int &count,string Securitymanual[],int instructions,int &countInstrut,int
&mac,int &civil,int &chem,int &oper,int &tech,int &elec,int &lathe,int &MillingMac,int
```

```
&Drill,int &Bandsaw,int &Grinder,int &Planar,int &corn,int &sago,int &PhosphateRock,int
&Ammonia,int &Potash,int &Nitrogen,string num,int &ordercount)
   {
   while(true)
   {
   system("cls");
   title();
   logintitle();
   string optmain=logininterface(x1,y1);
   if(optmain=="1")
   {
      while (true)
     {
      string user,role,pass;
      system("cls");
      title();
      setTextColor(51);
      cout << "                                                                                " << endl;
      cout << "                                    To access the system, Please LOG IN...." << endl;
      cout << "                          ------------------------------------------------------------------------
" << endl;
      cout << "                                                                                 " <<
endl;
      cout << "                          ------------------------------------------------------------------------
" << endl;
      cout << "                                                                                 " <<
endl;
      gotoxy(50, 14);
      cout << "<<<<<<<    SIGN IN MENU     >>>>>>>>";
      setTextColor(225);
```

```
gotoxy(30, 17);
cout << "Enter Username: ";
getline(cin, user);
gotoxy(30, 18);
cout << "Enter Password: ";
getline(cin, pass);
gotoxy(30, 19);
cout << "Sign In as (Administrator, Manager, Customer): ";
getline(cin, role);
gotoxy(30, 21);
setTextColor(86);
 if(usercount==0)
 {
 setTextColor(86);
 gotoxy(40,21);
 cout<<"-------------------------------------------------"<<endl;
 gotoxy(40,22);
 cout<<"                 No Data Found               "<<endl;
 gotoxy(40,23);
 cout<<"-------------------------------------------------"<<endl;
 system("pause");
 break;
 }
   bool found = false;


for (int i = 0; i < usercount; ++i)
{
   if (!username[i].empty() && user== username[i] && pass== password[i] && role==
userrole[i])
```

```
    {
        found = true;
        x=i;
        break;
    }
}
if (found)
{
    setTextColor(86);
    gotoxy(35, 22);
    cout << "Congratulations! Your account has been logged IN successfully..." << endl;
    system("pause");

mainmenu(x1,y1,username,password,userrole,usercount,totalusers,x,userrole[x],feedback,empname,empid,empID,maxrow,product,quantity,Alert,productNames,singleproductPrices,productQuantity,date,Notii,p,orderNumbers,productN,reasons,processed,customerInformation,MAX_CUSTOMERS,customerCount,CUSTOMER_FIELDS,fields,orders,fieldsOrders,MAX_ORDERS,machines,totalMac,ordernumbers,op_data,NUM_OPERATORS,numWorkers,allocatednum,raw_materials,MAX_Rawmat,numRawmat,bill,productPrices,saledName,saledQuantity,count,Securitymanual,instructions,countInstrut,mac,civil,chem,oper,tech,elec,lathe,MillingMac,Drill,Bandsaw,Grinder,Planar,corn,sago,PhosphateRock,Ammonia,Potash,Nitrogen,num,ordercount);
}
else
{
    setTextColor(86);
    gotoxy(35, 21);
    cout << "Invalid Credentials! Please Try Again..." << endl;
    system("pause");
}


    }
```

```
    }

    else if(optmain=="2")
    {

        bool correctinput = false;
          while (correctinput == false)
          {
              system("cls");
              title();
              setTextColor(51);
              cout << "                                                                                " <<
endl;
              cout << "                                        To access the system, Please LOG IN...." <<
endl;
              cout << "                          ------------------------------------------------------------------------
--                          " << endl;
              cout << "                                                                                " <<
endl;
              cout << "                          ------------------------------------------------------------------------
--             " << endl;
              cout << "                                                                                  "
<< endl;
              gotoxy(50, 14);
              cout << "<<<<<<<    SIGN UP MENU     >>>>>>>>" << endl;
              setTextColor(207);
              string newUser, newPassword, newUserRole;
               setTextColor(225);
              gotoxy(30, 17);
              cout << "Select Username: ";
```

```
            getline(cin,newUser);
            gotoxy(30, 18);
            cout << "Select Password: ";
            getline(cin,newPassword);
            gotoxy(30, 19);
            cout << "Sign Up as (Administrator, Manager, Customer): ";
            getline(cin,newUserRole);
            if (newUserRole != "Administrator" && newUserRole != "administrator" &&
newUserRole != "Manager" && newUserRole != "manager" && newUserRole != "Customer"
&& newUserRole != "customer")
            {
                setTextColor(86);
               gotoxy(35, 21);
               cout << "Invalid User Role Entered! Please choose a valid role..." << endl;
               system("pause");
               continue;
            }
            else
            {
               username[usercount] = newUser;
               password[usercount] = newPassword;
               userrole[usercount] = newUserRole;
               x=usercount;
               usercount++;
               setTextColor(86);
               gotoxy(35, 21);
               cout << "Congratulations! Your account has been Signed UP successfully..." <<
endl;
               system("pause");
```

```
                break; // Break the inner loop to restart the entire process
            }
        }
    }
else if(optmain=="3")
{
    while(true)
    {
        setTextColor(226);
        cout << "                                                    " << endl;
        cout << "                          Are you sure you want to exit!" << endl;
        cout << endl;
        cout << "                      YES          or          NO" << endl;
        string choice;
        cout << "                     Your Choice: ";
        getline(cin, choice);
        if (choice == "YES" || choice == "yes")
        {
            system("cls");
            gotoxy(50, 0);
            setTextColor(230);
            cout << "Exiting IndustriaSync Hub...GoodBye!" << endl;
            exit(0);
        }
        else if (choice == "NO" || choice == "no")
        {
            break; // Restart the loop to allow choosing options again
        }
```

```
            else
            {
                setTextColor(86);
                cout << "                          Invalid Choice. Please Try Again..." << endl;
                cout << "                                                                " << endl;
                system("pause");
                continue; // Restart the loop to allow choosing options again
            }

        }
    }
        else if(optmain!="1"&&optmain!="2"&&optmain!="3")
        {
            setTextColor(86);
            gotoxy(x1 + 5, y1 + 5);
            cout << "                                                                " << endl;
            cout << "                        Invalid Input! Please Try Again..." << endl;
            cout << "                                                                " << endl;
            system("pause");
            continue; // Restart the loop to allow choosing options again
        }
    }
    }
void gotoxy(int x, int y)
{
  COORD coordinates;
  coordinates.X = x;
  coordinates.Y = y;
```

```
  SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coordinates);
}
void title()
{
  setBackgroundColor(0);
  setTextColor(230);
cout<<"                                                                                "<<endl;
cout<<"              _____          _         _       _____               _   _      _
"<<endl;
cout<<"             |_   _|        | |       | |     (_)       / ____|             | | | |    | |           "<<endl;
cout<<"               | | _ __   __| |_   _ ___| |_ _ __ _  __ _ | (___  _   _ _ __   ___| |__| |_   _| |__
"<<endl;
cout<<"       *******  | | '_ \\ / _` | | | / __| __| '__| |/ _` |  \\___ \\| | | | '_ \\ / __|  __  | | | | '_ \\
*******       "<<endl;
cout<<"              _| |_| | | | (_| | |_| \\__ \\ |_| |  | | (_| |  ____) | |_| | | | | (__| |  | | |_| | |_) |
"<<endl;
cout<<"             |_____|_| |_|\\__,_|\\__,_|___/\\__|_|  |_|\\__,_| |_____/ \\___, |_| |_|\\___|_|
|_|\\__,_|_.__/          "<<endl;
cout<<"                                                  __/ |                         "<<endl;
cout<<"                                                 |___/                          "<<endl;

   resetColors();

}
void logintitle()
{
setBackgroundColor(0);
 setTextColor(51);
cout<<"                                                                                "<<endl;
cout<<"                                  To access the system, Please LOG IN...."<<endl;
```

```
cout<<"                    ------------------------------------------------------------------------
"<<endl;
cout<<"                                                  Login Menu
"<<endl;
cout<<"                    ------------------------------------------------------------------------
"<<endl;
cout<<"                                                                                                "<<endl;
 resetColors();
}
string logininterface(int x1,int y1)
{
   setTextColor(225);
   string option;
   string array[3]={"Sign In", "Sign Up", "Exit"};
      {
         for(int idx=0;idx<3;idx++)
         {
            gotoxy(x1,y1);
            cout<<idx+1<<". "<<array[idx];
            y1=y1+1;
         }

      }
      gotoxy(x1,y1);
      cout<<"Choose Option(1-3): ";
      getline(cin,option);
      return option;
}
void mainmenutitle()
```

```
{
 setTextColor(230);
cout<<"                                    | |                                          "<<endl;
cout<<"                                ******   |/\\|elcome to IndustriaSync Hub     ******
"<<endl;
setTextColor(51);
cout<<"                          ------------------------------------------------------------------------
"<<endl;
cout<<"                                                                                   "<<endl;
cout<<"                          ----------------------<<<<<<<    MAIN MENU     >>>>>>>>----------------
-----        "<<endl;
}
void mainmenu(int x1,int y1,string username[],string password[],string userrole[],int
usercount,int totalusers,int x,string role,string &feedback,string empname[],int empid[],int
empID,int maxrow,string product[],string quantity[],string Alert[],string productNames[],double
singleproductPrices[],double productQuantity[],string date[],string Notii[],int p,string
orderNumbers[],string productN[],string reasons[],bool processed[],string
customerInformation[][11],int MAX_CUSTOMERS,int &customerCount,int
CUSTOMER_FIELDS,string fields[],string orders[][4],string fieldsOrders[],int
MAX_ORDERS,string machines[][5],int totalMac,int &ordernumbers,string op_data[][5],int
NUM_OPERATORS,int &numWorkers,int &allocatednum,string  raw_materials[][5],int
MAX_Rawmat,int &numRawmat,int &bill,double productPrices[], string saledName[], double
saledQuantity[],int &count,string Securitymanual[],int instructions,int &countInstrut,int
&mac,int &civil,int &chem,int &oper,int &tech,int &elec,int &lathe,int &MillingMac,int
&Drill,int &Bandsaw,int &Grinder,int &Planar,int &corn,int &sago,int &PhosphateRock,int
&Ammonia,int &Potash,int &Nitrogen,string num,int &ordercount)
{
    int xA = 34, yA = 14;
    system("cls");
    title();
    mainmenutitle();
    menu(xA,yA);
    cout<<"                                                  "<<endl;
    gotoxy(35,20);
```

```
cout<<"       ******  You have been logged in as "<<role<<"   *******"<<endl;
system("pause");
if(role=="Administrator"||role=="administrator")
{
  while(true)
  {
  administrator();
  string opt=AdmMenu(xA,yA);
   if(opt=="1")
   {
   usermanagement(xA,yA,empname,empid,empID,maxrow,num);
   continue;
   }
   else if(opt=="2")
   {
   systemconfiguration(xA,yA,product,quantity,Alert,feedback,maxrow,date);
   continue;
   }
   else if(opt=="3")
   {

accesscontrol(xA,yA,x1,y1,username,password,userrole,usercount,totalusers,x,feedback,empna
me,empid,empID,maxrow,product,quantity,Alert,productNames,singleproductPrices,productQua
ntity,date,Notii,p,orderNumbers,productN,reasons,processed,customerInformation,MAX_CUST
OMERS,customerCount,CUSTOMER_FIELDS,fields,orders,fieldsOrders,MAX_ORDERS,mac
hines,totalMac,ordernumbers,op_data,NUM_OPERATORS,numWorkers,allocatednum,raw_mat
erials,MAX_Rawmat,numRawmat,bill,productPrices,saledName,saledQuantity,count,Securityma
nual,instructions,countInstrut,mac,civil,chem,oper,tech,elec,lathe,MillingMac,Drill,Bandsaw,Gri
nder,Planar,corn,sago,PhosphateRock,Ammonia,Potash,Nitrogen,num,ordercount);
   continue;
   }
```

```
else if(opt=="4")
{
systemmonitoring(xA,yA,op_data,NUM_OPERATORS,numWorkers,num);
continue;
}
else if(opt=="5")
{
securityandcomplianceManual(xA,yA,Securitymanual,instructions,countInstrut,num);
continue;
}
else if(opt=="6")
{

GenerateProductionReports(bill,productPrices,saledName,saledQuantity,maxrow,count,productN
ames,productQuantity);
continue;
}
else if(opt=="7")
{

loginsystem(x1,y1,username,password,userrole,usercount,totalusers,x,feedback,empname,empid,
empID,maxrow,product,quantity,Alert,productNames,singleproductPrices,productQuantity,date,
Notii,p,orderNumbers,productN,reasons,processed,customerInformation,MAX_CUSTOMERS,c
ustomerCount,CUSTOMER_FIELDS,fields,orders,fieldsOrders,MAX_ORDERS,machines,total
Mac,ordernumbers,op_data,NUM_OPERATORS,numWorkers,allocatednum,raw_materials,MA
X_Rawmat,numRawmat,bill,productPrices,saledName,saledQuantity,count,Securitymanual,instr
uctions,countInstrut,mac,civil,chem,oper,tech,elec,lathe,MillingMac,Drill,Bandsaw,Grinder,Plan
ar,corn,sago,PhosphateRock,Ammonia,Potash,Nitrogen,num,ordercount);
break;
}
else
{
```

```
        cout<<"                    Invalid Choice. Please Try Again..."<<endl;
        cout<<"                     "<<endl;
        system("pause");
        continue;
        }
       }

    }
     else if(role=="Manager"||role=="manager")
     {
        while(true)
        {
        managers();
        string opt=managermenu(xA,yA);
        if(opt=="1")
        {

productionPlanning_Scheduling(xA,yA,maxrow,product,quantity,date,customerInformation,MAX_CUSTOMERS,CUSTOMER_FIELDS,customerCount,fields,orders,fieldsOrders,MAX_ORDERS,bill,productPrices,saledName,saledQuantity,count,productNames,productQuantity);
        continue;
        }
        else if(opt=="2")
        {

ResourceAllocation(xA,yA,machines,totalMac,op_data,NUM_OPERATORS,maxrow,product,quantity,date,orders,MAX_ORDERS,allocatednum);
        continue;
        }
        else if(opt=="3")
```

```
{
MaterialHandling(xA,yA,raw_materials,MAX_Rawmat,numRawmat);
continue;
}
else if(opt=="4")
{
viewmanual(Securitymanual,instructions,countInstrut);
continue;
}
else if(opt=="5")
{

GenerateReport(bill,productPrices,saledName,saledQuantity,maxrow,count,productNames,produ
ctQuantity);
continue;
}
else if(opt=="6")
{
InventoryManagement(xA,yA,machines,totalMac,ordernumbers);
continue;
}
else if(opt=="7")
{

BudgetingandCostControl(xA,yA,mac,civil,chem,oper,tech,elec,lathe,MillingMac,Drill,Bandsaw
,Grinder,Planar,corn,sago,PhosphateRock,Ammonia,Potash,Nitrogen,num);
continue;
}
else if(opt=="8")
{
```

```
loginsystem(x1,y1,username,password,userrole,usercount,totalusers,x,feedback,empname,empid,
empID,maxrow,product,quantity,Alert,productNames,singleproductPrices,productQuantity,date,
Notii,p,orderNumbers,productN,reasons,processed,customerInformation,MAX_CUSTOMERS,c
ustomerCount,CUSTOMER_FIELDS,fields,orders,fieldsOrders,MAX_ORDERS,machines,total
Mac,ordernumbers,op_data,NUM_OPERATORS,numWorkers,allocatednum,raw_materials,MA
X_Rawmat,numRawmat,bill,productPrices,saledName,saledQuantity,count,Securitymanual,instr
uctions,countInstrut,mac,civil,chem,oper,tech,elec,lathe,MillingMac,Drill,Bandsaw,Grinder,Plan
ar,corn,sago,PhosphateRock,Ammonia,Potash,Nitrogen,num,ordercount);
      break;
      }
      else
      {
      cout<<"                    Invalid Choice. Please Try Again..."<<endl;
      cout<<"                      "<<endl;
      system("pause");
      continue;
      }
     }
   }

    else if(role=="Customer"||role=="customer")
    {
     while(true)
     {
     customer();
     string opt=custMenu(xA,yA);
    if(opt=="1")
     {
```

```
OrderPlacementandTracking(xA,yA,productNames,singleproductPrices,productQuantity,product
,quantity,date,maxrow,bill,productPrices,saledName,saledQuantity,count,num);
    continue;
    }
    else if(opt=="2")
    {
    providefeedback(feedback);
    continue;
    }
    else if(opt=="3")
    {

accountmanagement(customerInformation,MAX_CUSTOMERS,customerCount,CUSTOMER_
FIELDS,fields,num);
    continue;
    }
    else if(opt=="4")
    {
    Notifications(xA,yA,Notii,p,num);
    continue;
    }
    else if(opt=="5")
    {
    ReturnsandRefunds(xA,yA,orderNumbers,productN,reasons,processed,maxrow,num);
    continue;
    }
    else if(opt=="6")
    {
    SupportandFAQs(xA,yA,orders,fieldsOrders,MAX_ORDERS,num,ordercount);
```

```
      continue;
       }
      else if(opt=="7")
       {

loginsystem(x1,y1,username,password,userrole,usercount,totalusers,x,feedback,empname,empid,
empID,maxrow,product,quantity,Alert,productNames,singleproductPrices,productQuantity,date,
Notii,p,orderNumbers,productN,reasons,processed,customerInformation,MAX_CUSTOMERS,c
ustomerCount,CUSTOMER_FIELDS,fields,orders,fieldsOrders,MAX_ORDERS,machines,total
Mac,ordernumbers,op_data,NUM_OPERATORS,numWorkers,allocatednum,raw_materials,MA
X_Rawmat,numRawmat,bill,productPrices,saledName,saledQuantity,count,Securitymanual,instr
uctions,countInstrut,mac,civil,chem,oper,tech,elec,lathe,MillingMac,Drill,Bandsaw,Grinder,Plan
ar,corn,sago,PhosphateRock,Ammonia,Potash,Nitrogen,num,ordercount);


       break;
       }
       else
       {
       cout<<"                    Invalid Choice. Please Try Again..."<<endl;
       cout<<"                      "<<endl;
       system("pause");
       continue;
       }
      }


   }
}
void menu(int xA,int yA)
{

    setTextColor(225);
```

```
    string array[3]={"Administrator", "Managers","Customers"};
        {
            for(int idx=0;idx<3;idx++)
            {
                gotoxy(xA,yA+2);
                cout<<idx+1<<". "<<array[idx];
                yA=yA+1;
            }

        }
         resetColors();

}
void administrator()
{
    system("cls");
    setTextColor(230);
    cout<<"                                                                    "<<endl;
    cout<<"                                                                    "<<endl;
    cout<<"                                  | |                              "<<endl;
    cout<<"                                ******   |/\\|elcome to IndustriaSync Hub     ******
"<<endl;
    cout<<"                                                                    "<<endl;
    setTextColor(51);
    cout<<"                        ------------------------------------------------------------------------
"<<endl;
    cout<<"                                              Administrator Menu
"<<endl;
    cout<<"                        ------------------------------------------------------------------------
"<<endl;
```

```
   cout<<"                                                                    "<<endl;
}
string AdmMenu(int xA,int yA)
   {
       setTextColor(225);
      string option;
   string array[7]={"User Management", "System Configuration", "Access Control", "System Monitoring", "Security and Compliance","Generate Production Report","Logout"};
      {
         for(int idx=0;idx<7;idx++)
         {
            gotoxy(xA,yA);
            cout<<idx+1<<". "<<array[idx];
            yA=yA+1;
         }

      }
      gotoxy(xA,yA);
      cout<<"Choose Option(1-7): ";
      getline(cin,option);
      return option;
}
   void systemconfiguration(int xA,int yA,string product[],string quantity[],string Alert[],string &feedback,int maxrow,string date[])
   {

       while(true)
       {
       administrator();
```

```
 systemconfigurationtitle();
 string opt=systemconfigurationmenu(xA,yA);
 if(opt=="1")
 {
 productionSpeed(maxrow,product,quantity,Alert,date);
continue;
}
else if(opt=="2")
{
alertThreshold(maxrow,Alert);
continue;
}
else if(opt=="3")
{
viewfeedback(feedback);
continue;
}

else if(opt=="4")
{
   break;

}
else
{
cout<<"                    Invalid Choice. Please Try Again..."<<endl;
cout<<"                     "<<endl;
system("pause");
```

```
        continue;
        }
         }

    }
    void productionSpeed(int maxrow,string product[],string quantity[],string Alert[],string date[])
    {
     system("cls");
     systemconfigurationtitle();
      setTextColor(51);
        gotoxy(35,5);
        cout<<" ----------------------------------------------------------------"<<endl;
        gotoxy(35,6);
        cout<<"                      Current Record(s)                    "<<endl;
        gotoxy(35,7);
        cout<<" ----------------------------------------------------------------"<<endl;
        cout<<"                                                           "<<endl;
        int counter=0;
        for(int x=0;x<maxrow;x++)
        {
            if(!product[x].empty())
            {
            setTextColor(225);
             counter++;
             gotoxy(30,9);
             cout<<"Ordered Product: "<<product[x]<<endl;
             gotoxy(30,10);
             cout<<"Ordered Quantity: "<<quantity[x]<<endl;
```

```
        gotoxy(30,11);
        cout<<"Expected Date: "<<date[x]<<endl;
         setTextColor(230);
        gotoxy(35,13);
        cout<<"-------------------------------------------------"<<endl;
        gotoxy(35,14);
        cout<<"       <<<<<< Production speed increases  >>>>>>   "<<endl;
        gotoxy(35,15);
        cout<<"-------------------------------------------------"<<endl;
        Alert[x]=1;
       }

    }

    if(counter==0)
    {
     setTextColor(86);
    gotoxy(35,13);
    cout << "------------------------------" << endl;
    gotoxy(35,14);
    cout << "       No Record found!        " << endl;
    gotoxy(35,15);
    cout << "------------------------------ " << endl;
       gotoxy(35,17);
       cout<<"Production Speed Remains same!";
    }
     gotoxy(30,18);
     system("pause");
```

```
}
void alertThreshold(int maxrow,string Alert[])
{
 system("cls");
 systemconfigurationtitle();
  setTextColor(51);
 gotoxy(35,5);
cout<<" ----------------------------------------------------------------"<<endl;
gotoxy(35,6);
cout<<"                      Alert Threshold                     "<<endl;
gotoxy(35,7);
cout<<" ----------------------------------------------------------------"<<endl;
cout<<"                                                          "<<endl;
 int counter=0;
  for(int x=0;x<maxrow;x++)
    {
       if(Alert[x]!="\0")
       {
          counter++;
          gotoxy(35,9);
           setTextColor(31);
          cout << "ALERT: Value exceeds the alert threshold!" << endl;
}
    }
    if(counter==0)
    {
      gotoxy(35,9);
```

```
            setTextColor(86);
           cout << "Value is within the acceptable range." << endl;
        }

        gotoxy(35,13);
        system("pause");
    }
     void viewfeedback(string &feedback)
   {
    system("cls");
     systemconfigurationtitle();
      setTextColor(51);
      gotoxy(35,5);
    cout<<" ------------------------------------------------------------"<<endl;
    gotoxy(35,6);
    cout<<"                    Customer FeedBack                    "<<endl;
    gotoxy(35,7);
    cout<<" ------------------------------------------------------------"<<endl;
    cout<<"                                                         "<<endl;

     if(!feedback.empty())
     cout<<feedback;
     else
     {
    setTextColor(86);
     gotoxy(35,15);
     cout<<"No FeedBack Yet!"<<endl;
     }
```

```
    gotoxy(30,16);
    system("pause");
  }
```

```
void accesscontrol(int xA,int yA,int x1,int y1,string username[],string password[],string
userrole[],int usercount,int totalusers,int x,string &feedback,string empname[],int empid[],int
empID,int maxrow,string product[],string quantity[],string Alert[],string productNames[],double
singleproductPrices[],double productQuantity[],string date[],string Notii[],int p,string
orderNumbers[],string productN[],string reasons[],bool processed[],string
customerInformation[][11],int MAX_CUSTOMERS,int customerCount,int
CUSTOMER_FIELDS,string fields[],string orders[][4],string fieldsOrders[],int
MAX_ORDERS,string machines[][5],int totalMac,int &ordernumbers,string op_data[][5],int
NUM_OPERATORS,int &numWorkers,int &allocatednum,string  raw_materials[][5],int
MAX_Rawmat,int &numRawmat, int &bill,double productPrices[], string saledName[], double
saledQuantity[],int &count,string Securitymanual[],int instructions,int &countInstrut,int
&mac,int &civil,int &chem,int &oper,int &tech,int &elec,int &lathe,int &MillingMac,int
&Drill,int &Bandsaw,int &Grinder,int &Planar,int &corn,int &sago,int &PhosphateRock,int
&Ammonia,int &Potash,int &Nitrogen,string num,int &ordercount)
{
    while(true)
 {
   administrator();
   accesscontroltitle();
   string opt=accesscontrolmenu(xA,yA);
       if(opt=="1")
       {
       system("cls");
       accesscontroltitle();
        while(true)
       {
       managers();
       string opt=managermenu(xA,yA);
       if(opt=="1")
```

```
{

productionPlanning_Scheduling(xA,yA,maxrow,product,quantity,date,customerInformation,MAX_CUSTOMERS,CUSTOMER_FIELDS,customerCount,fields,orders,fieldsOrders,MAX_ORDERS,bill,productPrices,saledName,saledQuantity,count,productNames,productQuantity);

continue;

}

else if(opt=="2")

{

ResourceAllocation(xA,yA,machines,totalMac,op_data,NUM_OPERATORS,maxrow,product,quantity,date,orders,MAX_ORDERS,allocatednum);

continue;

}

else if(opt=="3")

{

MaterialHandling(xA,yA,raw_materials,MAX_Rawmat,numRawmat);

continue;

}

else if(opt=="4")

{

viewmanual(Securitymanual,instructions,countInstrut);

continue;

}

else if(opt=="5")

{

GenerateReport(bill,productPrices,saledName,saledQuantity,maxrow,count,productNames,productQuantity);

continue;

}
```

```
else if(opt=="6")
{
InventoryManagement(xA,yA,machines,totalMac,ordernumbers);
continue;
}
else if(opt=="7")
{

BudgetingandCostControl(xA,yA,mac,civil,chem,oper,tech,elec,lathe,MillingMac,Drill,Bandsaw
,Grinder,Planar,corn,sago,PhosphateRock,Ammonia,Potash,Nitrogen,num);
continue;
}
else if(opt=="8")
{

loginsystem(x1,y1,username,password,userrole,usercount,totalusers,x,feedback,empname,empid,
empID,maxrow,product,quantity,Alert,productNames,singleproductPrices,productQuantity,date,
Notii,p,orderNumbers,productN,reasons,processed,customerInformation,MAX_CUSTOMERS,c
ustomerCount,CUSTOMER_FIELDS,fields,orders,fieldsOrders,MAX_ORDERS,machines,total
Mac,ordernumbers,op_data,NUM_OPERATORS,numWorkers,allocatednum,raw_materials,MA
X_Rawmat,numRawmat,bill,productPrices,saledName,saledQuantity,count,Securitymanual,instr
uctions,countInstrut,mac,civil,chem,oper,tech,elec,lathe,MillingMac,Drill,Bandsaw,Grinder,Plan
ar,corn,sago,PhosphateRock,Ammonia,Potash,Nitrogen,num,ordercount);
break;
}
else
{
cout<<"                    Invalid Choice. Please Try Again..."<<endl;
cout<<"                     "<<endl;
system("pause");
continue;
}
```

```
}
 }
 else if(opt=="2")
 {
    system("cls");
    accesscontroltitle();
while(true)
{
customer();
string opt=custMenu(xA,yA);
if(opt=="1")
 {


OrderPlacementandTracking(xA,yA,productNames,singleproductPrices,productQuantity,product
,quantity,date,maxrow,bill,productPrices,saledName,saledQuantity,count,num);
 continue;
 }
 else if(opt=="2")
 {
 providefeedback(feedback);
 continue;
 }
 else if(opt=="3")
 {


accountmanagement(customerInformation,MAX_CUSTOMERS,customerCount,CUSTOMER_
FIELDS,fields,num);
 continue;
 }
```

```
else if(opt=="4")
{
Notifications(xA,yA,Notii,p,num);
continue;
}
else if(opt=="5")
{
ReturnsandRefunds(xA,yA,orderNumbers,productN,reasons,processed,maxrow,num);
continue;
}
else if(opt=="6")
{
SupportandFAQs(xA,yA,orders,fieldsOrders,MAX_ORDERS,num,ordercount);
continue;
}
else if(opt=="7")
{

loginsystem(x1,y1,username,password,userrole,usercount,totalusers,x,feedback,empname,empid,
empID,maxrow,product,quantity,Alert,productNames,singleproductPrices,productQuantity,date,
Notii,p,orderNumbers,productN,reasons,processed,customerInformation,MAX_CUSTOMERS,c
ustomerCount,CUSTOMER_FIELDS,fields,orders,fieldsOrders,MAX_ORDERS,machines,total
Mac,ordernumbers,op_data,NUM_OPERATORS,numWorkers,allocatednum,raw_materials,MA
X_Rawmat,numRawmat,bill,productPrices,saledName,saledQuantity,count,Securitymanual,instr
uctions,countInstrut,mac,civil,chem,oper,tech,elec,lathe,MillingMac,Drill,Bandsaw,Grinder,Plan
ar,corn,sago,PhosphateRock,Ammonia,Potash,Nitrogen,num,ordercount);
break;
}
else
{
cout<<"                    Invalid Choice. Please Try Again..."<<endl;
```

```
        cout<<"                          "<<endl;
        system("pause");
        continue;
        }
       }

    }
    else if(opt=="3")
    {
       break;
    }
    else
        {
        cout<<"                        Invalid Choice. Please Try Again..."<<endl;
        cout<<"                          "<<endl;
        system("pause");
        continue;
        }
 }
}


void systemmonitoring(int xA,int yA,string op_data[][5],int NUM_OPERATORS,int
&numWorkers,string num)
{
    while(true)
    {
    administrator();
    systemmonitoringtitle();
    string opt=systemmonitoringmenu(xA,yA);
```

```
    if(opt=="1")
    {
        viewWorkers(op_data,NUM_OPERATORS);
        continue;
    }
    else if(opt=="2")
    {
      addWorkers(op_data,NUM_OPERATORS,numWorkers,num);
      continue;

    }
    else if(opt=="3")
    {
     updateWorkers(op_data,NUM_OPERATORS,num,numWorkers);
     continue;
    }
    else if(opt=="4")
    {
        deleteWorkers(op_data,NUM_OPERATORS,numWorkers,num);
        continue;

    }
    else if(opt=="5")
    {
      searchWorkers(op_data,NUM_OPERATORS,num);
      continue;
    }
    else if(opt=="6")
```

```
    {
        break;

    }
    else
    {
        cout<<"                         Invalid Choice. Please Try Again..."<<endl;
        cout<<"                          "<<endl;
        system("pause");
        continue;
    }
    }

}
 void systemmonitoringtitle()
    {
         setTextColor(230);
    cout<<"                                  <<<<     SYSTEM MONITORING     >>>>
"<<endl;
    cout<<"                                                                    "<<endl;
}
 string systemmonitoringmenu(int xA,int yA)
    {
         setTextColor(225);
        string option;
    string array[6]={"View Workers","Add Workers","Edit Workers","Delete Workers","Search
Workers","Exit"};
    for(int i=0;i<6;i++)
    {
```

```
        gotoxy(xA,yA);
        cout<<i+1<<". "<<array[i]<<endl;
        yA=yA+1;
    }
    gotoxy(xA,yA);
    cout<<"Choose Option(1-6): ";
    getline(cin,option);
    return option;
    }
void viewWorkers(string op_data[][5],int NUM_OPERATORS)
{
    system("cls");
     setTextColor(230);
    cout<<"                    ------------------------------------------------------------------------------
"<<endl;
    cout<<"                                                  WORKERS INFORMATION
"<<endl;
    cout<<endl;
    cout<<"                    ------------------------------------------------------------------------------
"<<endl;
         setTextColor(51);
        gotoxy(0,7);
        cout<<"Name";
        gotoxy(20,7);
        cout<<"|";
        gotoxy(25,7);
        cout<<"ID";
        gotoxy(30,7);
        cout<<"|";
```

```
gotoxy(35,7);
cout<<"Rank";
 gotoxy(60,7);
cout<<"|";
gotoxy(70,7);
cout<<"Skills";
gotoxy(115,7);
cout<<"|";
gotoxy(120,7);
cout<<"Salary";
int y=9;
int o=y,p=y,q=y,r=y;
 setTextColor(225);
for(int i=0;i<5;i++)
{
   for(int j=0;j<NUM_OPERATORS;j++)
   {
      if(i==0)
      {
       gotoxy(0,y);
       cout<<op_data[j][i];
        if(!op_data[j][i].empty())
       {
       gotoxy(20,y);
       cout<<"|";
       }
       y++;
      }
```

```
else if(i==1)
{
   gotoxy(25,o);
 cout<<op_data[j][i];
 if(!op_data[j][i].empty())
 {
 gotoxy(30,o);
 cout<<"|";
 }
 o++;

}
else if(i==2)
{
 gotoxy(35,p);
 cout<<op_data[j][i];
  if(!op_data[j][i].empty())
 {
 gotoxy(60,p);
 cout<<"|";
 }
 p++;
}
else if(i==3)
{
 gotoxy(65,q);
 cout<<op_data[j][i];
if(!op_data[j][i].empty())
```

```
            {
                gotoxy(115,q);
                cout<<"|";
            }
            q++;
            }
            else if(i==4)
            {
            gotoxy(120,r);
            cout<<op_data[j][i];
            r++;
            }
        }
    }
    gotoxy(30,30);
    system("pause");
}
void addWorkers(string op_data[][5],int NUM_OPERATORS,int &numWorkers,string num)
{
   system("cls");
    setTextColor(230);
   cout<<"                    ------------------------------------------------------------------------------
"<<endl;
   cout<<"                                          ADD WORKERS                    "<<endl;
   cout<<endl;
   cout<<"                    ------------------------------------------------------------------------------
"<<endl;
   string name,skills,rank;
   int id,salary;
```

```
string ID,Salary;
 setTextColor(225);
 while(true)
{
  gotoxy(30,8);
cout << "Enter Name: ";
getline(cin,name);
if(Name_Validations(name))
{
   break;
}
else
{
    setTextColor(86);
   gotoxy(30,9);
   cout << "Invalid Name Format!" << endl;
   getch();
   gotoxy(30,8);
   cout<<"                                                  ";
   gotoxy(30,9);
   cout<<"                                                  ";
}
}
 setTextColor(225);
 while(true)
{
gotoxy(30,10);
cout<<"Enter ID: ";
```

```
getline(cin,num);
if(ID_validation(num))
  {
   id=stoi(num);
   break;
   }
   else
{
   setTextColor(86);
  gotoxy(30,11);
  cout<<" Invalid Input! Please Enter ID which consists of 5 digits...";
  getch();
  gotoxy(35,10);
  cout<<"                                          ";
  gotoxy(30,11);
  cout<<"                                                          ";



}
}
 setTextColor(225);
gotoxy(30,12);
cout<<"Enter Rank: ";
getline(cin,rank);
 gotoxy(30,14);
cout<<"Enter Skills: ";
getline(cin,skills);
while(true)
```

```
{
gotoxy(30,16);
cout<<"Enter Salary: ";
cin>>num;
if(Salary_Validations(num))
  {
   salary=stoi(num);
   break;
   }
   else
{
   setTextColor(86);
  gotoxy(30,17);
  cout<<" Invalid Input!Submitted Salary is out of range...";
  getch();
  gotoxy(35,16);
  cout<<"                                          ";
  gotoxy(30,17);
  cout<<"                                                    ";
}

}
ID=to_string(id);
Salary=to_string(salary);
if(op_data[numWorkers][0].empty())
{
op_data[numWorkers][0]=name;
op_data[numWorkers][1]=ID;
```

```
    op_data[numWorkers][2]=rank;
    op_data[numWorkers][3]=skills;
    op_data[numWorkers][4]=Salary;
    numWorkers++;
    saveWorkers(numWorkers,op_data);
    }
    gotoxy(30,18);
    system("pause");
}
void deleteWorkers(string op_data[][5],int NUM_OPERATORS,int &numWorkers,string num)
{
    system("cls");
     setTextColor(51);
    cout<<"                    ------------------------------------------------------------------------------
"<<endl;
    cout<<"                                                DELETE WORKERS
"<<endl;
    cout<<endl;
    cout<<"                    ------------------------------------------------------------------------------
"<<endl;
     int id;
     string ID;
      setTextColor(225);
    while(true)
    {
    gotoxy(30,8);
    cout<<"Enter Employee ID: ";
    getline(cin,num);
    if(ID_validation(num))
```

```
    {
     id=stoi(num);
     break;
     }
     else
  {
     setTextColor(86);
    gotoxy(30,9);
    cout<<" Invalid Input! Please Enter ID which consists of 5 digits...";
    getch();
    gotoxy(30,8);
    cout<<"                                          ";
    gotoxy(30,9);
    cout<<"                                                  ";



  }
  }
  ID=to_string(id);
  int counter=0;
  for(int i=0;i<NUM_OPERATORS;i++)
  {
    if(op_data[i][1]==ID)
    {
       for(int j=i;j<numWorkers;j++)
       {
          counter++;
          op_data[j][0]=op_data[j+1][0];
```

```
            op_data[j][1]=op_data[j+1][1];
            op_data[j][2]=op_data[j+1][2];
            op_data[j][3]=op_data[j+1][3];
            op_data[j][4]=op_data[j+1][4];
        }
        numWorkers--;
         setTextColor(31);
        gotoxy(30,10);
        cout<<"Deleted Successfully";
        saveWorkers(numWorkers,op_data);
      }
   }
    if(counter==0)
      {
          setTextColor(86);
        gotoxy(30,10);
        cout<<"Employee Not Found";
      }
    gotoxy(30,12);
   system("pause");
}
void updateWorkers(string op_data[][5],int NUM_OPERATORS,string num,int &numWorkers)
{
  system("cls");
   setTextColor(51);
   cout<<"                    ------------------------------------------------------------------------------
"<<endl;
   cout<<"                                                UPDATE WORKERS
"<<endl;
```

```
    cout<<endl;
    cout<<"                    ---------------------------------------------------------------------------
"<<endl;
    string name,skills;
    int id,salary;
    string ID,rank,Salary;
     setTextColor(225);
     while(true)
    {
    gotoxy(30,8);
    cout<<"Update by Employee ID: ";
    getline(cin,num);
    if(ID_validation(num))
       {
         id=stoi(num);
         break;
         }
         else
    {
         setTextColor(86);
        gotoxy(30,9);
        cout<<" Invalid Input! Please Enter ID which consists of 5 digits...";
        getch();
        gotoxy(35,8);
        cout<<"                                          ";
        gotoxy(30,9);
        cout<<"                                                          ";
    }
    }
```

```
 setTextColor(225);
ID=to_string(id);
Salary=to_string(salary);
int counter=0;
for(int i=0;i<NUM_OPERATORS;i++)
{
   if(op_data[i][1]==ID)
   {
      counter++;
       while(true)
{
  gotoxy(30,10);
cout << "Enter Name: ";
getline(cin,name);
if(Name_Validations(name))
{
   break;
}
else
{
    setTextColor(86);
   gotoxy(30,11);
   cout << "Invalid Name Format!" << endl;
   getch();
   gotoxy(30,10);
   cout<<"                                                ";
   gotoxy(30,11);
   cout<<"                                                 ";
```

```
    }
    }
     setTextColor(225);
    gotoxy(30,12);
    cout<<"Enter Rank: ";
    getline(cin,rank);
    gotoxy(30,14);
    cout<<"Enter Skills: ";
    getline(cin,skills);
     while(true)
    {
    gotoxy(30,16);
    cout<<"Enter Salary: ";
    cin>>num;
    if(Salary_Validations(num))
       {
         salary=stoi(num);
         break;
         }
         else
    {
         setTextColor(86);
        gotoxy(30,17);
        cout<<" Invalid Input!Salary is out of range...";
        getch();
        gotoxy(35,16);
        cout<<"                                          ";
        gotoxy(30,17);
```

```
        cout<<"                                                    ";
    }
    }
    op_data[i][0]=name;
    op_data[i][1]=ID;
    op_data[i][2]=rank;
    op_data[i][3]=skills;
    op_data[i][4]=Salary;
     setTextColor(31);
    gotoxy(30,18);
    cout<<"Edited Successfully";
    saveWorkers(numWorkers,op_data);

    }
}
if(counter==0)
{
     setTextColor(31);
    gotoxy(30,10);
    cout<<"Worker Not Found";
}
gotoxy(30,18);
system("pause");
}
void searchWorkers(string op_data[][5],int NUM_OPERATORS,string num)
{
    system("cls");
     setTextColor(51);
```

```
   cout<<"                    ------------------------------------------------------------------------------
"<<endl;
   cout<<"                                                SEARCH WORKERS
"<<endl;
   cout<<endl;
   cout<<"                    ------------------------------------------------------------------------------
"<<endl;
   int id;
   string ID;
    setTextColor(225);
    while(true)
   {
   gotoxy(30,8);
   cout<<"Enter Employee ID: ";
   getline(cin,num);
   if(ID_validation(num))
     {
      id=stoi(num);
      break;
      }
      else
   {
      setTextColor(86);
     gotoxy(30,9);
     cout<<" Invalid Input! Please Enter ID which consists of 5 digits...";
     getch();
     gotoxy(35,8);
     cout<<"                                          ";
     gotoxy(30,9);
```

```
        cout<<"                                                    ";
    }
    }
    ID=to_string(id);
    int counter=0;
     setTextColor(225);
    for(int i=0;i<NUM_OPERATORS;i++)
    {
        if(op_data[i][1]==ID)
        {
            counter++;
            gotoxy(30,10);
            cout<<"Employee Name: "<<op_data[i][0]<<endl;
            gotoxy(30,12);
            cout<<"Employee ID: "<<op_data[i][1]<<endl;
            gotoxy(30,14);
            cout<<"Employee Rank : "<<op_data[i][2]<<endl;
            gotoxy(30,16);
            cout<<"Employee Skills: "<<op_data[i][3]<<endl;
            gotoxy(30,18);
            cout<<"Employee Salary: "<<op_data[i][4]<<endl;
        }
    }
    if(counter==0)
{
     setTextColor(31);
    gotoxy(30,10);
    cout<<"Employee Not Found";
```

```
}
system("pause");
}
void securityandcomplianceManual(int xA,int yA,string Securitymanual[],int instructions,int
&countInstrut,string num)
{
while(true)
   {
   administrator();
   securityandcompliancetitle();
   string opt=securityandcompliancemenu(xA,yA);
   if(opt=="1")
   {
      viewmanual(Securitymanual,instructions,countInstrut);
      continue;
   }
   else if(opt=="2")
   {
      addmanual(Securitymanual,instructions,countInstrut);
     continue;

   }
   else if(opt=="3")
   {
     deleteinstructions(Securitymanual,instructions,countInstrut,num);
    continue;



   }
```

```
   else if(opt=="4")
   {
      break;

   }
   else
   {
      cout<<"                        Invalid Choice. Please Try Again..."<<endl;
      cout<<"                          "<<endl;
      system("pause");
      continue;
   }
   }

}
 void securityandcompliancetitle()
   {
       setTextColor(230);
   cout<<"                              <<<<    SECURITY AND COMPLIANCE    >>>>
"<<endl;
   cout<<"                                                                  "<<endl;
}
 void OrderPlacementandTrackingTitle()
   {
       setTextColor(230);
   cout<<"                              <<<<   Order Placement and Tracking   >>>>
"<<endl;
   cout<<"                                                                  "<<endl;
```

```
    }
string OrderPlacementandTrackingmenu(int xA,int yA)
    {
    setTextColor(225);
    string option;
    string array[3]={"View Menu","Customize Menu","Exit"};
    for(int i=0;i<3;i++)
    {
        gotoxy(xA,yA);
        cout<<i+1<<". "<<array[i]<<endl;
        yA=yA+1;
    }
    gotoxy(xA,yA);
    cout<<"Choose Option(1-3): ";
    getline(cin,option);
    return option;
    }
    void OrderPlacementandTracking(int xA,int yA,string productNames[],double
singleproductPrices[],double productQuantity[],string product[],string quantity[],string date[],int
maxrow,int &bill,double productPrices[], string saledName[], double saledQuantity[],int
&count,string num)
{


    while(true)
        {
        customer();
        OrderPlacementandTrackingTitle();
        string opt=OrderPlacementandTrackingmenu(xA,yA);
        if(opt=="1")
```

```
        {

viewmenu(productNames,singleproductPrices,productQuantity,maxrow,bill,productPrices,saled
Name,saledQuantity,count,num);
        continue;
        }
        else if(opt=="2")
        {
        customize(product,quantity,maxrow,date,num);
        continue;
        }
        else if(opt=="3")
        {
           break;


        }
        else
        {
        cout<<"                         Invalid Choice. Please Try Again..."<<endl;
        cout<<"                          "<<endl;
        system("pause");
        continue;
        }
         }


}
void viewmenu(string productNames[], double singleproductPrices[], double productQuantity[],
int maxrow, int &bill, double productPrices[], string saledName[], double saledQuantity[], int
&count,string num)
{
```

```
 setTextColor(230);
system("cls");
gotoxy(35, 3);
cout << "          <<<<      Order Placement and Tracking   >>>>       " << endl;
cout << "                                                    " << endl;
 setTextColor(51);
gotoxy(35, 5);
cout << "-------------------------------------------------------------------------" << endl;
gotoxy(35, 6);
cout << "                          Available Products                    " << endl;
gotoxy(35, 7);
cout << "-------------------------------------------------------------------------" << endl;
gotoxy(35, 8);
cout << "                                                      " << endl;
 setTextColor(225);
gotoxy(25, 9);
cout << "Product Names";
gotoxy(45, 9);
cout << "|";
gotoxy(65, 9);
cout << "Product Quantity";
gotoxy(85, 9);
cout << "|";
gotoxy(105, 9);
cout << "Single Product Price" << endl;
int y = 11;
bool isfound = false;
for (int idx = 0; idx < 10; idx++)
```

```
    {
        gotoxy(25, y);
        cout << productNames[idx];
        gotoxy(45, y);
        cout << "|";
        gotoxy(65, y);
        cout << productQuantity[idx];
        gotoxy(85, y);
        cout << "|";
        gotoxy(105, y);
        cout << "$ " << singleproductPrices[idx];
        y = y + 1;
    }
    double saledQuant[10]={0};
    string Name[10]={};
    string anotherorder = "1";
    for (int i = 0; anotherorder == "1" && i < maxrow; i++)
    {
            gotoxy(30, 23);
            cout << "Enter Product Name: ";
            getline(cin,Name[i]);
            while(true)
            {
            gotoxy(30,24);
            cout<<"Enter Product Quantity: ";
            getline(cin,num);
            if(Salary_Validations(num))
             {
```

```
            saledQuant[i]=stod(num);
            break;
           }
         else
          {
            gotoxy(30,25);
             cout<<" Invalid Input!";
            getch();
             gotoxy(35,24);
            cout<<"                                                  ";
               gotoxy(30,25);
               cout<<"                                                    ";
          }
          }
          gotoxy(30, 26);
          cout << "If you want to place another order press 1 otherwise 0: ";
          getline(cin, anotherorder);
          if(anotherorder=="1")
          {
           gotoxy(20,23);
           cout<<"                                                  ";
           gotoxy(20,24);
           cout<<"                                                   ";
           gotoxy(30,26);
           cout<<"                                                    ";
          }
          count++;
    }
```

```
    bool found=false;
    int counter = 0;
    for (int i = 0; i < count; i++)
    {
        for (int j = 0; j < 10; j++)
        {
            if((Name[i]==productNames[j]) && (saledQuant[i] <=productQuantity[j]))
            {
                saledName[i]=Name[i];
                saledQuantity[i]=saledQuant[i];
                found=true;
                break;
            }
        }
    }
    if(found==true)
    {
     for (int i = 0; i < count; i++)
    {
        for(int j=0;j<10;j++)
        {
                if ((saledName[i] == productNames[j]) && (saledQuant[i] <= productQuantity[j]))
            {
                bill++;
                counter++;
                productPrices[i] = saledQuantity[i] * singleproductPrices[j];
                gotoxy(30, 27);
                cout << "Your Bill is: $ " << productPrices[i];
```

```
            break;
        }
    }
}
}

if (counter == 0)
{
    setTextColor(86);
    gotoxy(30, 27);
    cout << "Product not found or quantity exceeds limit." << endl;
    gotoxy(30, 28);
    cout << "NOW back to Main Menu and customize your order.";
}

gotoxy(30, 29);
system("pause");
}
void customize(string product[], string quantity[], int maxrow, string date[],string num)
{
system("cls");
 setTextColor(230);
gotoxy(35, 3);
cout << "            <<<<      Order Placement and Tracking   >>>>       " << endl;
cout << "                                        " << endl;
 setTextColor(51);
gotoxy(35, 5);
cout << "-------------------------------------------------------------------------" << endl;
```

```
    gotoxy(35, 6);
    cout << "                    Customize Your Product                    " << endl;
    gotoxy(35, 7);
    cout << "--------------------------------------------------------------------------" << endl;
    gotoxy(35, 8);
    cout << "                                                              " << endl;
     setTextColor(225);
    string prod;
    int quant;
    string duedate;
    string Quant;

    gotoxy(30, 9);
    cout << "Enter Product Name: ";
    getline(cin, prod);
     while(true)
    {
    gotoxy(30,11);
    cout<<"Enter Product Quantity: ";
    getline(cin,num);
    if(Salary_Validations(num))
       {
         quant=stoi(num);
         break;
         }
         else
    {
         setTextColor(31);
```

```
        gotoxy(30,12);
        cout<<" Invalid Input!";
        getch();
        gotoxy(35,11);
        cout<<"                                          ";
        gotoxy(30,12);
        cout<<"                                                          ";
    }
    }
      gotoxy(30,13);
      cout << "Enter date (YYYY-MM-DD): ";
      getline(cin, duedate);
      Quant=to_string(quant);
    while (!isValidDate(duedate))
    {
        cout << "Invalid date format. Please enter a date in the format YYYY-MM-DD: ";
        getline(cin, duedate);
    }
    if(prod.empty()||Quant.empty()||duedate.empty())
    {
        gotoxy(50,16);
        cout << "Please enter all the required fields." << endl;
        if(prod.empty())
        {
         gotoxy(30,9);
         cout<<"\b \b";
         cout<<" Enter Product Name: ";
          getline(cin, prod);
```

```
    }
    else if(Quant.empty())
    {
       while(true)
      {
       gotoxy(30,11);
       cout<<"Enter Product Quantity: ";
       getline(cin,num);
       if(Salary_Validations(num))
         {
           quant=stoi(num);
            break;
         }
       else
        {
        gotoxy(30,12);
        cout<<" Invalid Input! Please enter a positive integer: ";
        getch();
        gotoxy(35,11);
        cout<<"                                                ";
        gotoxy(30,12);
        cout<<"                                                                  ";
        }
        }
    }
    else if(duedate.empty())
    {
       gotoxy(30,13);
```

```
        cout<<"\b \b";
        cout<<" Enter Expected Date (YYYY-MM-DD): ";
         getline(cin, duedate);


       }
          while (!isValidDate(duedate))
   {
      gotoxy(40,14);
      cout << "Invalid date format. Please enter a date in the format YYYY-MM-DD: ";
      getline(cin, duedate);
   }


   }

   for (int i = 0; i < maxrow; i++)
   {
      if (product[i].empty()&& quantity[i].empty()&& date[i].empty())
      {
         product[i] = prod;
         quantity[i] = Quant;
         date[i] = duedate;
          setTextColor(31);
         // Provide order details to the user
         gotoxy(40,16);
         cout << "**** Your Order is placed successfully! ****" << endl;
          setTextColor(86);
          gotoxy(30,17);
         cout << "Product Name: " << product[i] << endl;
```

```
            gotoxy(30,18);
            cout << "Quantity: " << quantity[i] << endl;
            gotoxy(30,19);
            cout << "Due Date: " << date[i] << endl;

            break;
        }
    }

    cout << "                                                                    " << endl;
    system("pause");
}

string securityandcompliancemenu(int xA,int yA)
{
     setTextColor(225);
    string option;
    string array[4]={"View Safety Manual","Add Safety Instructions","Delete Safety
Instructions","Exit"};
    for(int i=0;i<4;i++)
    {
        gotoxy(xA,yA);
        cout<<i+1<<". "<<array[i]<<endl;
        yA=yA+1;
    }
    gotoxy(xA,yA);
    cout<<"Choose Option(1-4): ";
    getline(cin,option);
    return option;
```

```
}
void viewmanual(string Securitymanual[],int instructions,int &countInstrut)
{
   int y=6;
   system("cls");
    setTextColor(51);
   cout<<"                    ------------------------------------------------------------------------
"<<endl;
   cout<<"                                                SAFETY MANUAL
"<<endl;
   cout<<endl;
   cout<<"                    ------------------------------------------------------------------------
"<<endl;
    setTextColor(225);
   for(int i=0;i<countInstrut;i++)
   {
      gotoxy(30,y);
      cout<<i+1<<". "<<Securitymanual[i]<<endl;
      y=y+2;
   }
   system("pause");
}

void addmanual(string Securitymanual[],int instructions,int &countInstrut)

{
   int counter=0;
   system("cls");
    setTextColor(51);
```

```
   cout<<"                    ------------------------------------------------------------------------------
"<<endl;
   cout<<"                                                ADD SAFETY INSTRUCTIONS
"<<endl;
   cout<<endl;
   cout<<"                    ------------------------------------------------------------------------------
"<<endl;
   string ins={};
   setTextColor(225);
   gotoxy(30,8);
   cout<<"Add Safety Instruction: ";
    setTextColor(51);
   getline(cin,ins);
   for(int i=0;i<countInstrut;i++)
   {
      if(Securitymanual[i]==ins)
      {
          setTextColor(86);
         cout<<"Instruction Already Exists. Please Try Again..."<<endl;
         cout<<"                              "<<endl;
         system("pause");
         return;
      }
   }

   if(countInstrut==instructions)
   {
      cout<<"Error: Array Full. Cannot Add More Instructions."<<endl;
      cout<<"                              "<<endl;
```

```
        system("pause");
        return;
    }
    if(!ins.empty())
    {
    Securitymanual[countInstrut]=ins;
    countInstrut++;
    }
    gotoxy(30,11);
    system("pause");
}
void deleteinstructions(string Securitymanual[],int instructions,int &countInstrut,string num)
{
     setTextColor(230);
    system("cls");
    cout<<"                    ------------------------------------------------------------------------------
"<<endl;
    cout<<"                                                DELETE INSTRUCTIONS
"<<endl;
    cout<<endl;
    cout<<"                    ------------------------------------------------------------------------------
"<<endl;
    int number=0;
    while(true)
    {
        setTextColor(51);
    gotoxy(30,8);
    cout<<"Enter Instruction Number: ";
    cin>>num;
```

```
   if(validity_checker(num))
     {
      number=stoi(num);
      break;
      }
      else
   {
      setTextColor(86);
     gotoxy(30,9);
     cout<<" Invalid Input";
     getch();
     gotoxy(30,8);
     cout<<"                                                    ";
     gotoxy(30,9);
     cout<<"                                                    ";
   }

   }
   int counter=0;
   for(int i=0;i<countInstrut;i++)
   {
      if(i==number)
      {
         counter++;
         gotoxy(30,8);
         cout<<"Are you sure you want to delete instruction number "<<num<<"? (Y/N): ";
         char ch;
         cin>>ch;
```

```
            if(ch=='Y' || ch=='y')
            {
               for(int j=i;j<countInstrut;j++)
               {
                  Securitymanual[j]=Securitymanual[j+1];
               }
               countInstrut--;
               gotoxy(30,10);
               cout<<"Instruction Deleted Successfully";
            }
            else
            {
               gotoxy(30,10);
               cout<<"Instruction Not Deleted";
            }
         }
      }
       if(counter==0)
         {
            gotoxy(30,10);
            cout<<"Number Not Found";
         }
       gotoxy(30,12);
      system("pause");
}

       string systemconfigurationmenu(int xA,int yA)
      {
```

```
        setTextColor(225);
       string option;
   string array[4]={"Production Speed","Alert Threshold","View FeedBack","Exit"};
   for(int i=0;i<4;i++)
   {
      gotoxy(xA,yA);
      cout<<i+1<<". "<<array[i]<<endl;
      yA=yA+1;
   }
   gotoxy(xA,yA);
   cout<<"Choose Option(1-4): ";
   getline(cin,option);
   return option;
   }
   string  accesscontrolmenu(int xA,int yA)
{
    setTextColor(225);
    string option;
   string array[3]={"View Manager Menu","View Customer Menu","Exit"};
   for(int i=0;i<3;i++)
   {
      gotoxy(xA,yA);
      cout<<i+1<<". "<<array[i]<<endl;
      yA=yA+1;
   }
   gotoxy(xA,yA);
   cout<<"Choose Option(1-3): ";
   getline(cin,option);
```

```
    return option;


}
 void systemconfigurationtitle()
    {
        setTextColor(51);
    cout<<"                                    <<<<      System Configuration   >>>>
"<<endl;
    cout<<"                                                                        "<<endl;
    }
    void accesscontroltitle()
    {
    cout<<"                                    <<<<      Access Control     >>>>
"<<endl;
    cout<<"                                                                        "<<endl;
    }
     void providefeedback(string &feedback)
{
system("cls");
customer();
 setTextColor(51);
cout<<"                                  <<<<      Customer Feedback   >>>>
"<<endl;
cout<<"                                                                        "<<endl;


    gotoxy(30,14);
    cout<<"Enter your FeedBack: ";
    setTextColor(225);
    getline(cin,feedback);
```

```
    setTextColor(86);
    gotoxy(35,19);
    cout << "Thank you for your feedback! We appreciate your input." << endl;
    gotoxy(35,21);
    system("pause");
}
void customer()
{
    system("cls");
     setTextColor(230);
    cout<<"                                                                    "<<endl;
    cout<<"                                                                    "<<endl;
    cout<<"                                  | |                              "<<endl;
    cout<<"                               ******   |/\\|elcome to IndustriaSync Hub     ******
"<<endl;
    cout<<"                                                                    "<<endl;
    cout<<"                        ---------------------------------------------------------------------------
"<<endl;
    cout<<"                                          Customer  Menu
"<<endl;
    cout<<"                        ---------------------------------------------------------------------------
"<<endl;
    cout<<"                                                                    "<<endl;
}
string custMenu(int xA,int yA)
    {
         setTextColor(225);
        string option;
```

```
   string array[7]={"Order Placement and Tracking", "Provide Feedback on received products",
"Account Management", "Notifications", "Returns and Refunds","Support and
FAQs","Logout"};
      {
         for(int idx=0;idx<7;idx++)
         {
            gotoxy(xA,yA);
            cout<<idx+1<<". "<<array[idx];
            yA=yA+1;
         }

      }
      gotoxy(xA,yA);
      cout<<"Choose Option(1-7): ";
      getline(cin,option);
      return option;
}



void viewnote(int p,string Notii[])
{
system("cls");
customer();
 setTextColor(51);
cout<<"                                <<<<     Notification Menu    >>>>                "<<endl;
cout<<"                                                                                  "<<endl;
cout<<"                          ----------------------------------------------------------------    "<<endl;
cout<<"                                      VIEW  NOTIFICATIONS
"<<endl;
```

```
cout<<"                    ----------------------------------------------------------------     "<<endl;
cout<<"                                                                                         "<<endl;
int counter=0;
int y=18;
 setTextColor(225);
for(int x=0;x<p;x++)
 {
   if(Notii[x]!="\0")
      {
       counter++;
       gotoxy(40,y);
       cout<<x+1;
       gotoxy(60,y);
       cout<<"|";
       gotoxy(75,y);
       cout<<Notii[x];
       cout<<endl;
       y=y+2;
      }
 }
 if(counter==0)

 {          setTextColor(86);
          cout<<"                                                                  "<<endl;
          cout<<"                              ------------------------------------"<<endl;
          cout<<"                                    No Notification found! "<<endl;
          cout<<"                              ------------------------------------
"<<endl;
```

```
 }


        system("pause");


}
void updatenote(int p,string Notii[],string num)
{
system("cls");
customer();
 setTextColor(51);
cout<<"                                <<<<      Notification Menu    >>>>                 "<<endl;
cout<<"                                                                                    "<<endl;
cout<<"                          ---------------------------------------------------------------------     "<<endl;
cout<<"                                          UPDATE NOTIFICATIONS
"<<endl;
cout<<"                          ---------------------------------------------------------------------     "<<endl;
cout<<"                                                                                    "<<endl;
       int counter=0;
        int notenum;
       while (true)
{
    gotoxy(30, 20);
    std::cout << "Enter notification number: ";

    // Take input for notification number using std::getline
    std::getline(std::cin, num);

    if (validity_checker(num))
```

```
    {
        try
        {
            notenum = stoi(num);  // Convert to integer after validation
            break;  // Break out of the loop if the input is valid
        }
        catch (const std::out_of_range& e)
        {
            // Handle the case where the number is too large to fit into an int
            gotoxy(30, 21);
            std::cout << "Invalid input!";
            getch();
        }
    }
    else
    {
        // Display error message
        gotoxy(30, 21);
        cout << "Invalid input!";
        getch();

        // Clear the input area
        gotoxy(30, 20);
        cout << "\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t";
        gotoxy(30, 21);
        cout << "\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t";
    }
}
```

```
      string newnotification;
       gotoxy(30,22);
      cout<<"Enter updated notification: ";
      getline(cin,newnotification);
      bool isfound=false;
       if (notenum >= 0 && notenum < p)
       {
       Notii[notenum-1] = newnotification;
       counter++;
       isfound = true;
       }
      if(isfound)
      {
        gotoxy(35,28);
        cout<<"Notification has been updated successfully!"<<endl;
      }
      if(counter==0)

 {
          cout<<"                                                    "<<endl;
          cout<<"                                 ------------------------------------"<<endl;
          cout<<"                                       No Notification found! "<<endl;
          cout<<"                                 ----------------------------------
"<<endl;

 }

       system("pause");
```

```
}
void deletenote(int p,string Notii[],string num)
{
system("cls");
customer();
 setTextColor(51);
cout<<"                                <<<<      Notification Menu    >>>>                 "<<endl;
cout<<"                                                                                    "<<endl;
cout<<"                          ----------------------------------------------------------------------    "<<endl;
cout<<"                                          DELETE NOTIFICATIONS
"<<endl;
cout<<"                          ----------------------------------------------------------------------    "<<endl;
cout<<"                                                                                    "<<endl;
 int counter=0;
      int notenum;
       setTextColor(225);
       while (true)
{
   gotoxy(30, 20);
   cout << "Enter notification number: ";

   // Take input for notification number using std::getline
   getline(std::cin, num);
   if (validity_checker(num))
   {
      try
      {
         notenum = stoi(num);  // Convert to integer after validation
         break;  // Break out of the loop if the input is valid
```

```
        }
        catch (const std::out_of_range& e)
        {
            // Handle the case where the number is too large to fit into an int
            gotoxy(30, 21);
            cout << "Invalid input!";
            getch();
        }
    }
    else
    {
         // Display error message
        gotoxy(30, 21);
        cout << "Invalid input!";
        getch();

        // Clear the input area
        gotoxy(30, 20);
        cout << "\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t";
        gotoxy(30, 21);
        cout << "\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t";
    }
}
      bool isfound=false;
      string msg="Notification has been Deleted!";
      if (notenum >= 0 && notenum < p)
       {
       Notii[notenum-1] = msg;
```

```
        counter++;
        isfound = true;
        }
       if(isfound)
       {
         gotoxy(35,28);
         cout<<"Notification has been Deleted successfully!"<<endl;
       }
       if(counter==0)

 {
           cout<<"                                                    "<<endl;
           cout<<"                                 ------------------------------------"<<endl;
           cout<<"                                       No Notification found! "<<endl;
           cout<<"                                 ------------------------------------
"<<endl;

 }

        system("pause");
}
void addnote(int p,string Notification[])
{
system("cls");
customer();
 setTextColor(51);
cout<<"                              <<<<    Notification Menu   >>>>              "<<endl;
cout<<"                                                                "<<endl;
cout<<"                        ---------------------------------------------------------------    "<<endl;
```

```
cout<<"                                   ADD  NOTIFICATIONS                          "<<endl;
cout<<"                          ------------------------------------------------------------------     "<<endl;
cout<<"                                                                                 "<<endl;
string notification={};
int counter=0;
 setTextColor(225);
     cout<<"                      Enter Notification: ";
      setTextColor(86);
     getline(cin, notification);
     for(int i=0;i<p;i++)
     {
        counter++;
      if(Notification[i]=="\0" && notification!="\0")
      {
         Notification[i]=notification;
         break;
      }

     }
    cout<<"                                                                          "<<endl;
    cout<<"                                 Notification has been added successfully!"<<endl;
    cout<<endl;
    if(counter==0)
    {
      cout<<"                                                                        "<<endl;
       cout<<"                                 ------------------------------------"<<endl;
       cout<<"                                       No Notification found! "<<endl;
       cout<<"                                 ----------------------------------                 "<<endl;
```

```
    }
   system("pause");
}
void ReturnsandRefundstitle()
   {
    setTextColor(51);
   cout<<"                                    <<<<      Returns and Refunds   >>>>
"<<endl;
   cout<<"                                                                          "<<endl;
   }
    string ReturnsandRefundsmenu(int xA,int yA)
   {
       setTextColor(225);
      string option;
   string array[3]={"Initiate Return","View Return Requests","Exit"};
   for(int i=0;i<3;i++)
   {
      gotoxy(xA,yA);
      cout<<i+1<<". "<<array[i]<<endl;
      yA=yA+1;
   }
   gotoxy(xA,yA);
   cout<<"Choose Option(1-3): ";
   getline(cin,option);
   return option;
   }
void ReturnsandRefunds(int xA,int yA,string orderNumbers[],string productN[],string
reasons[],bool processed[],int maxrow,string num)
{
```

```
    while(true)
{
   customer();
   ReturnsandRefundstitle();
   string opt=ReturnsandRefundsmenu(xA,yA);
       if(opt=="1")
      {
      initiatereturn(processed,orderNumbers,productN,reasons,maxrow,num);
      continue;
      }
      else if(opt=="2")
      {
        viewreturnrequests(processed,orderNumbers,productN,reasons,maxrow);
        continue;
      }

      else if(opt=="3")
      {
         break;

      }
      else
      {
      cout<<"                        Invalid Choice. Please Try Again..."<<endl;
      cout<<"                          "<<endl;
      system("pause");
      continue;
      }
```

```
        }



}
void initiatereturn(bool processed[],string orderNumbers[],string productN[],string reasons[],int
p,string num)
{
system("cls");
   customer();
    setTextColor(51);
   cout << "                                    <<<<     Initiate Return     >>>>
" << endl;
   cout << "                                                                                    " << endl;
   string newProductName, newReason;
   int newOrderNumber=0;
   string num1={};
    setTextColor(225);
   for (int i = 0; i < p; ++i)
   {
      if (!processed[i])
      {
         while(true)
         {
            gotoxy(30,15);
            cout << "Enter Order Number: ";
            getline(cin, num);
            if(validity_checker(num))
            {
               newOrderNumber=stoi(num);
```

```
            if(newOrderNumber>0 && newOrderNumber<=p)
            {
            num1=to_string(newOrderNumber);
            }
            break;
        }
        else
        {
            gotoxy(30,16);
            cout << "Invalid Order Number. Please try again." << endl;
            getch();
            gotoxy(30,15);
            cout<<"                                                                    ";
            gotoxy(30,16);
            cout<<"                                                                    ";

        }
    }
     while(true)
     {
        gotoxy(30,17);
        cout << "Enter Product Name: ";
        getline(cin, newProductName);
        if(Name_Validations(newProductName))
        {
        break;
        }
        else
```

```
            {
            gotoxy(30,18);
            cout << "Invalid Name Format!" << endl;
            getch();
            gotoxy(35,17);
            cout<<"                                                                                ";
            gotoxy(35,18);
            cout<<"                                                                                ";
  }
  }
        gotoxy(30,19);
        cout << "Enter Reason for Return: ";
        getline(cin, newReason);
        if(newReason.empty() || newProductName.empty() || newOrderNumber==0)
        {
           for(int i=0;i<p;i++)
           {
              if(orderNumbers[i]==num1)
              {
                 orderNumbers[i] = "";
                 productN[i] = "";
                 reasons[i] = "";
                 processed[i] = false;
                 gotoxy(30,21);
                 cout << "Return request cancelled." << endl;
                 gotoxy(30,22);
                 system("pause");
                 return;
```

```
                }
            }
        }
        else
        {
        orderNumbers[i] = newOrderNumber;
        productN[i] = newProductName;
        reasons[i] = newReason;
        processed[i] = true;  // Marking as processed
        gotoxy(30,21);
        cout << "Return request submitted successfully." << endl;
        gotoxy(30,22);
        system("pause");
        return;  // Exit the function after processing one request.
        }
      }
   }

   cout << "Return request limit reached. Please try again later." << endl;
   system("pause");
}
void viewreturnrequests(bool processed[],string orderNumbers[],string productN[],string
reasons[],int p)
{
system("cls");
customer();
 setTextColor(51);
cout<<"                                <<<<     View Return Requests     >>>>
"<<endl;
```

```
cout<<"                                                                    "<<endl;
 setTextColor(225);
for (int i = 0; i < p; ++i)
 {
       if (processed[i])
   {
       gotoxy(30,15);
       cout << "Order Number: " << orderNumbers[i] << endl;
       gotoxy(30,16);
       cout << "Product Name: " << productN[i] << endl;
       gotoxy(30,17);
       cout << "Reason: " << reasons[i] << endl;
       gotoxy(30,18);
       cout << "Processed: " << (processed[i] ? "Yes" : "No") << endl;
       gotoxy(30,19);
       cout << "------------------------" << endl;
       processed[i]=true;
   }
 }
   system("pause");
}
void managers()
{

   system("cls");
    setTextColor(230);
   cout<<"                                                                    "<<endl;
   cout<<"                                                                   "<<endl;
```

```
   cout<<"                                    | |                                          "<<endl;
   cout<<"                                ******   |/\\|elcome to IndustriaSync Hub     ******
"<<endl;
   cout<<"                                                                                 "<<endl;
   cout<<"                          ------------------------------------------------------------------------
"<<endl;
   cout<<"                                              Manager Menu                       "<<endl;
   cout<<"                          ------------------------------------------------------------------------
"<<endl;
   cout<<"                                                                                    "<<endl;


}
string managermenu(int xA,int yA)
{
    setTextColor(225);
   string option;
   string array[8]={" Production Planning and Scheduling", " Resource Allocation"," Material
Handling"," Security Instructions and Compliance"," Generate Sales Report"," Inventory
Management"," Budgeting and Cost Control"," Log Out"};
      {
         for(int idx=0;idx<8;idx++)
         {
            gotoxy(xA,yA);
            cout<<idx+1<<". "<<array[idx];
            yA=yA+1;
         }

      }
      gotoxy(xA,yA);
      cout<<"Choose Option(1-8): ";
```

```
    getline(cin,option);
    return option;
}
void productionPlanning_Schedulingtitle()
{ setTextColor(51);
  cout<<"                          <<<<    Production Planning and Scheduling
>>>>"<<endl;
  cout<<"                                                                    "<<endl;
}
string productionschedulingmenu(int xA,int yA)
  {
     setTextColor(225);
    string option;
  string array[4]={"View Production Orders","Analyze Production Schedule","Generate
Production Reports","Exit"};
  for(int i=0;i<4;i++)
  {
    gotoxy(xA,yA);
    cout<<i+1<<". "<<array[i]<<endl;
    yA=yA+1;
  }
  gotoxy(xA,yA);
  cout<<"Choose Option(1-4): ";
  getline(cin,option);
  return option;
  }
  void productionPlanning_Scheduling(int xA,int yA,int maxrow,string product[],string
quantity[],string date[],string customerInformation[][11],int MAX_CUSTOMERS,int
CUSTOMER_FIELDS,int &customerCount,string fields[],string orders[][4],string
```

```
fieldsOrders[],int MAX_ORDERS,int &bill,double productPrices[],string saledName[],double
saledQuantity[],int &count,string productNames[],double productQuantity[])
  {
     while(true)
     {
     managers();
     productionPlanning_Schedulingtitle();
     string opt=productionschedulingmenu(xA,yA);
     if(opt=="1")
     {

ViewProduction(maxrow,product,quantity,date,customerInformation,MAX_CUSTOMERS,CUS
TOMER_FIELDS,customerCount,fields);
     continue;
     }
     else if(opt=="2")
     {
     ModifyProduction(orders,fieldsOrders,MAX_ORDERS);
     continue;
     }
     else if(opt=="3")
     {

GenerateReport(bill,productPrices,saledName,saledQuantity,maxrow,count,productNames,produ
ctQuantity);
     continue;
     }


     else if(opt=="4")
     {
```

```
        break;


    }
    else
    {
    cout<<"                    Invalid Choice. Please Try Again..."<<endl;
    cout<<"                     "<<endl;
    system("pause");
    continue;
    }
     }


  }
  void accountmanagement(string customerInformation[][11],int MAX_CUSTOMERS,int
&customerCount,int CUSTOMER_FIELDS,string fields[],string num)
{
system("cls");
 setTextColor(230);
cout<<"                          <<<<   Account Management   >>>>
"<<endl;
cout<<"                                                          "<<endl;
cout<<"                                                          "<<endl;
setTextColor(225);
if(customerCount<MAX_CUSTOMERS)
{
  while(true)
  {
  gotoxy(30,6);
  cout << "Enter Name: ";
```

```
getline(cin, customerInformation[customerCount][0]);
if(Name_Validations(customerInformation[customerCount][0]))
{
   break;
}
else
{
   gotoxy(30,7);
   cout << "Invalid Name Format!" << endl;
   getch();
   gotoxy(30,6);
   cout<<"                                                ";
   gotoxy(30,7);
   cout<<"                                                 ";
}
}
gotoxy(30,8);
cout << "Enter Address: ";
getline(cin, customerInformation[customerCount][1]);
while(true)
{
 gotoxy(30,10);
cout << "Enter Phone Number: ";
getline(cin, num);
if(Contact_Validations(num))
  {
    customerInformation[customerCount][2]=num;
    break;
```

```
        }
        else
    {
       gotoxy(30,11);
       cout<<" Invalid Input!Contact Number Must be 11 Digits";
       getch();
       gotoxy(30,10);
       cout<<"                                                                    ";
       gotoxy(30,11);
       cout<<"                                                                    ";
    }

    }
    gotoxy(30,12);
    cout << "Enter Email: ";
    getline(cin, customerInformation[customerCount][3]);
    while(true)
    {
    gotoxy(30,14);
    cout << "Enter Credit Card Number: ";
    getline(cin,num);
    if(CreditCardNumber_Validations(num))
      {
        customerInformation[customerCount][4]=num;
        break;
        }
        else
    {
```

```
    gotoxy(30,15);
    cout<<" Invalid Input! Credit Card Number Must be 16 Digits";
    getch();
    gotoxy(30,14);
    cout<<"                                                                    ";
    gotoxy(30,15);
    cout<<"                                                                    ";
  }
  }
  gotoxy(30,16);
  cout << "Enter Credit Card Expiry Date: ";
  getline(cin, customerInformation[customerCount][5]);
  while (!isValidDate(customerInformation[customerCount][5]))
  {
    gotoxy(30,17);
    cout << "Invalid date format. Please enter a date in the format YYYY-MM-DD: ";
    getline(cin, customerInformation[customerCount][5]);
  }
   while(true)
  {
  gotoxy(30,18);
  cout << "Enter Credit Card CVV: ";
  getline(cin,num);
  if(ID_validation(num))
     {
      customerInformation[customerCount][6]=num;
      break;
      }
```

```
        else
    {
       gotoxy(30,19);
       cout<<" Invalid Input!CCV Must be 5 Digits";
       getch();
       gotoxy(30,18);
       cout<<"                                                                                ";
       gotoxy(30,19);
       cout<<"                                                                                 ";
    }
    }
     gotoxy(30,20);
    cout << "Enter Account Type: ";
    getline(cin, customerInformation[customerCount][7]);
     while(true)
    {
    gotoxy(30,22);
    cout << "Enter Account Limit: ";
    getline(cin,num);
    if(Salary_Validations(num))
       {
        customerInformation[customerCount][8]=num;
        break;
       }
        else
    {
       gotoxy(30,23);
       cout<<" Invalid Input!";
```

```
    getch();
    gotoxy(30,22);
    cout<<"                                                                    ";
    gotoxy(30,23);
    cout<<"                                                                    ";
  }
  }
   while(true)
  {
  gotoxy(30,24);
  cout << "Enter Account Balance: ";
  getline(cin,num);
  if(Salary_Validations(num))
     {
      customerInformation[customerCount][9]=num;
      break;
     }
      else
  {
     gotoxy(30,25);
     cout<<" Invalid Input!";
     getch();
     gotoxy(30,24);
     cout<<"                                                                    ";
     gotoxy(30,25);
     cout<<"                                                                    ";
  }
  }
```

```
    gotoxy(30,26);
    cout << "Enter Account Status: ";
    getline(cin, customerInformation[customerCount][10]);
    for(int i=0; i<CUSTOMER_FIELDS; i++)
    {
    if(customerInformation[customerCount][i].empty())
    {
            gotoxy(30,6+(2*i));
            cout << "\b \b \b";
            cout<<" Enter "<<fields[i];
            getline(cin, customerInformation[customerCount][i]);
    }
    }
    setTextColor(86);
    for(int i=0; i<CUSTOMER_FIELDS; i++)
    {
    if(!customerInformation[customerCount][i].empty())
    {
         gotoxy(50,27);
        cout << "Customer added successfully." << endl;
    }
    }
    // Increment the customer count
    ++customerCount;
}
else
    {
        cout << "Maximum number of customers reached." << endl;
```

```
    }
    system("pause");


}
void ViewProduction(int maxrow,string product[],string quantity[],string date[],string
customerInformation[][11],int MAX_CUSTOMERS,int CUSTOMER_FIELDS,int
&customerCount,string fields[])
    {
    system("cls");
    managers();
     int counter=0;
      setTextColor(51);
         gotoxy(50,13);
    cout<<"*******    ORDER INFORMATION     *******"<<endl;
          setTextColor(225);
        for(int x=0;x<maxrow;x++)
        {

if(!product[x].empty()&&!quantity[x].empty()&&!date[x].empty()&&!customerInformation[x][
0].empty()&&!customerInformation[x][1].empty()&&!customerInformation[x][2].empty()&&!c
ustomerInformation[x][3].empty()&&!customerInformation[x][4].empty()&&!customerInformat
ion[x][5].empty()&&!customerInformation[x][6].empty()&&!customerInformation[x][7].empty(
)&&!customerInformation[x][8].empty()&&!customerInformation[x][9].empty()&&!customerIn
formation[x][10].empty())
            {
             counter++;
             gotoxy(30,15);
             cout<<"Ordered Product: "<<product[x]<<endl;
             gotoxy(30,16);
             cout<<"Ordered Quantity: "<<quantity[x]<<endl;
             gotoxy(30,17);
```

```
        cout<<"Expected Date: "<<date[x]<<endl;
         setTextColor(51);
        gotoxy(50,19);
        cout<<"*******    CUSTOMER INFORMATION     *******"<<endl;
         setTextColor(225);
        int x=30,y=21;
       for(int i=0;i<customerCount;i++)
       {
      for(int j=0;j<CUSTOMER_FIELDS;j++)
     {
       gotoxy(x,y);
       cout<<fields[j]<<customerInformation[i][j]<<endl;
       y++;
     }

       }

     }
    }
    if(counter==0)
    {
        setTextColor(86);
    gotoxy(50,19);
    cout << "------------------------------" << endl;
    gotoxy(50,20);
    cout << "       No Record found!        " << endl;
    gotoxy(50,21);
    cout << "------------------------------ " << endl;
```

```
    }
 system("pause");
}



string notificationmenu(int xA,int yA)
{
    setTextColor(225);
string option;
string array[5]={"Add Notifications", "Update Notifications", "Delete Notifications", "Display Notifications","Exit"};
    {
       for(int idx=0;idx<5;idx++)
       {
          gotoxy(xA,yA);
          cout<<idx+1<<". "<<array[idx];
          yA=yA+1;
       }

    }
    gotoxy(xA,yA);
    cout<<"Choose Option(1-5): ";
    getline(cin,option);
    return option;
}
 void notificationtitle()
{
    setTextColor(51);
cout<<"                              <<<<    Notifications   >>>>                "<<endl;
```

```
    cout<<"                                                                    "<<endl;
    }
    void Notifications(int xA,int yA,string Notii[],int p,string num)
{
    system("cls");
    while(true)
    {
    customer();
    notificationtitle();
    string opt=notificationmenu(xA,yA);
    if(opt=="1")
    {
      addnote(p,Notii);
       continue;
    }
    else if(opt=="2")
    {

      updatenote(p,Notii,num);
      continue;

    }
    else if(opt=="3")
    {

     deletenote(p,Notii,num);
     continue;
```

```
    }

    else if(opt=="4")
    {
      viewnote(p,Notii);
      continue;
    }
    else if(opt=="5")
    {
       break;

    }
    else
    {
       cout<<"                    Invalid Choice. Please Try Again..."<<endl;
       cout<<"                      "<<endl;
       system("pause");
       continue;
    }
    }



}

void SupportandFAQs(int xA,int yA,string orders[][4],string fieldsOrders[],int
MAX_ORDERS,string num,int &ordercount)
{
```

```
while(true)
 {
 customer();
 SupportandFAQstitle();
 string opt=SupportandFAQsmenu(xA,yA);
 if(opt=="1")
{
OrderGantt(orders,fieldsOrders,MAX_ORDERS);
continue;
}
else if(opt=="2")
{
Addorders(orders,fieldsOrders,MAX_ORDERS,num,ordercount);
continue;
}
else if(opt=="3")
{
break;
}
else
{
cout<<"                    Invalid Choice. Please Try Again..."<<endl;
cout<<"                     "<<endl;
system("pause");
continue;
}
 }
```

```
    }
    string SupportandFAQsmenu(int xA,int yA)
    {
        setTextColor(225);
       string option;
    string array[3]={"View Order Gantt","Place Orders","Exit"};
    for(int i=0;i<3;i++)
    {
       gotoxy(xA,yA);
       cout<<i+1<<". "<<array[i]<<endl;
       yA=yA+1;
    }
    gotoxy(xA,yA);
    cout<<"Choose Option(1-3): ";
    getline(cin,option);
    return option;
    }
    void SupportandFAQstitle()
    {
        setTextColor(51);
    cout<<"                                  <<<<     ORDER GANTT     >>>>
"<<endl;
    cout<<"                                                                  "<<endl;
}
void ModifyProduction(string orders[][4],string fieldsOrders[],int MAX_ORDERS)
{
    system("cls");
     setTextColor(230);
```

```
   cout<<"                              <<<<     Production Planning and Scheduling  Menu
>>>>                  "<<endl;
   cout<<"                                                                                  "<<endl;
    setTextColor(51);
   cout<<"                        ---------------------------------------------------------------------------
"<<endl;
   cout<<"                                          CREATE PRODUCTION SCHEDULE
"<<endl;
   cout<<endl;
   cout<<"                        ---------------------------------------------------------------------------
"<<endl;
    setTextColor(225);
       gotoxy(10,9);
       cout<<"Product Names";
       gotoxy(30,9);
       cout<<"|";
       gotoxy(40,9);
       cout<<"Product Quantity";
       gotoxy(60,9);
       cout<<"|";
       gotoxy(70,9);
       cout<<"Expected Dates"<<endl;
        gotoxy(90,9);
       cout<<"|";
       gotoxy(100,9);
       cout<<"Specifications"<<endl;
       int y=11;
       int o=y,p=y,q=y;
       for(int i=0;i<4;i++)
       {
```

```
for(int j=0;j<100;j++)
{
   if(j<100 && i==0)
   {
    gotoxy(10,y);
    cout<<orders[j][i];
     if(!orders[j][i].empty())
    {
    gotoxy(30,y);
    cout<<"|";
    }
    y++;
   }
   else if(j<100 && i==1)
   {
     gotoxy(40,o);
    cout<<orders[j][i];
    if(!orders[j][i].empty())
    {
    gotoxy(60,o);
    cout<<"|";
    }
    o++;

   }
   else if(j<100 && i==2)
   {
    gotoxy(70,p);
```

```
                cout<<orders[j][i];
                 if(!orders[j][i].empty())
                {
                gotoxy(90,p);
                cout<<"|";
                }
                p++;
               }
              else if(j<100 && i==3)
              {
               gotoxy(100,q);
               cout<<orders[j][i];
               q++;
              }
           }
        }
        system("pause");
}
void Addorders(string orders[][4], string fieldsOrders[], int MAX_ORDERS,string num,int
&ordercount)
{
    system("cls");
    customer();
    cout<<endl<<endl;
    string product,date,spec;
    double quantity=0;
     setTextColor(51);
    gotoxy(50,11);
    cout<<"<<<<<<       PLACE ORDERS      >>>>>>>>"<<endl;
```

```
 setTextColor(225);
 while(true)
{
  gotoxy(30,13);
cout<<"Enter Product Name: ";
getline(cin,product);
if(Name_Validations(product))
{
   break;
}
else
{
   gotoxy(30,14);
   cout << "Invalid Name Format!" << endl;
   getch();
   gotoxy(30,13);
   cout<<"                                                                    ";
   gotoxy(30,14);
   cout<<"                                                                    ";
}
}
while(true)
{
gotoxy(30,15);
cout<<"Enter Product Quantity: ";
getline(cin,num);
if(Salary_Validations(num))
   {
```

```
        quantity=stod(num);
        break;
       }
        else
    {
       gotoxy(30,16);
       cout<<" Invalid Input";
       gotoxy(30,15);
       cout<<"                                                                    ";
       gotoxy(30,16);
       cout<<"                                                                    ";
   }
   }
    gotoxy(30,17);
   cout<<"Enter Expected Date: ";
   getline(cin,date);
   while (!isValidDate(date))
   {
       gotoxy(30,18);
       cout << "Invalid date format. Please enter a date in the format YYYY-MM-DD: ";
       getline(cin,date);
   }
    gotoxy(30,19);
   cout<<"Enter Specifications: ";
   getline(cin,spec);
   if(orders[ordercount][0].empty())
   {
       orders[ordercount][0]=product;
```

```
        orders[ordercount][1]=quantity;
        orders[ordercount][2]=date;
        orders[ordercount][3]=spec;
        ordercount++;
        saveOrders(ordercount,orders);
    }
   system("pause");
}
void OrderGantt(string orders[][4],string fieldsOrders[],int MAX_ORDERS)
{
     system("cls");
      setTextColor(230);
    gotoxy(45,5);
    cout<<"<<<<<<      GENERATE ORDER GANTT      >>>>>>>>"<<endl;
     setTextColor(51);
      gotoxy(10,9);
        cout<<"Product Names";
        gotoxy(30,9);
        cout<<"|";
        gotoxy(40,9);
        cout<<"Product Quantity";
        gotoxy(60,9);
        cout<<"|";
        gotoxy(70,9);
        cout<<"Expected Dates"<<endl;
         gotoxy(90,9);
        cout<<"|";
        gotoxy(100,9);
```

```
cout<<"Specifications"<<endl;
 setTextColor(225);
int y=11;
int o=y,p=y,q=y;
for(int i=0;i<4;i++)
{
   for(int j=0;j<100;j++)
   {
      if(j<100 && i==0)
      {
       gotoxy(10,y);
       cout<<orders[j][i];
        if(!orders[j][i].empty())
       {
       gotoxy(30,y);
       cout<<"|";
       }
       y++;
      }
      else if(j<100 && i==1)
      {
        gotoxy(40,o);
       cout<<orders[j][i];
       if(!orders[j][i].empty())
       {
       gotoxy(60,o);
       cout<<"|";
       }
```

```
            o++;


           }
          else if(j<100 && i==2)
          {
           gotoxy(70,p);
           cout<<orders[j][i];
            if(!orders[j][i].empty())
           {
           gotoxy(90,p);
           cout<<"|";
           }
           p++;
          }
          else if(j<100 && i==3)
          {
           gotoxy(100,q);
           cout<<orders[j][i];
           q++;
          }
        }
      }
      system("pause");

}
void viewInventory(string machines[][5],int totalMac)
{
   system("cls");
```

```
    setTextColor(230);
   cout<<"                    ------------------------------------------------------------------------------
"<<endl;
   cout<<"                                                  INVENTORY INFORMATION
"<<endl;
   cout<<endl;
   cout<<"                    ------------------------------------------------------------------------------
"<<endl;
        setTextColor(51);
       gotoxy(0,7);
       cout<<"ID";
       gotoxy(10,7);
       cout<<"|";
       gotoxy(15,7);
       cout<<"Type";
       gotoxy(30,7);
       cout<<"|";
       gotoxy(35,7);
       cout<<"Names";
        gotoxy(45,7);
       cout<<"|";
       gotoxy(60,7);
       cout<<"Specification";
       gotoxy(90,7);
       cout<<"|";
       gotoxy(100,7);
       cout<<"Function";
        setTextColor(225);
       int y=9;
```

```
int o=y,p=y,q=y,r=y;
for(int i=0;i<5;i++)
{
   for(int j=0;j<totalMac;j++)
   {
      if(j<totalMac && i==0)
      {
       gotoxy(0,y);
       cout<<machines[j][i];
        if(!machines[j][i].empty())
       {
       gotoxy(10,y);
       cout<<"|";
       }
       y++;
      }
      else if(j<totalMac && i==1)
      {
        gotoxy(15,o);
       cout<<machines[j][i];
       if(!machines[j][i].empty())
       {
       gotoxy(30,o);
       cout<<"|";
       }
       o++;

      }
```

```
else if(j<totalMac && i==2)
{
 gotoxy(35,p);
 cout<<machines[j][i];
  if(!machines[j][i].empty())
 {
 gotoxy(45,p);
 cout<<"|";
 }
 p++;
}
else if(j<totalMac && i==3)
{
 gotoxy(50,q);
 cout<<machines[j][i];
if(!machines[j][i].empty())
{
     gotoxy(90,q);
     cout<<"|";
}
 q++;
}
 else if(j<totalMac && i==4)
{
 gotoxy(95,r);
 cout<<machines[j][i];
 r++;
}
```

```
            }
        }
        system("pause");
}
void addInventory(string machines[][5],int totalMac,int &ordernumbers)
{
    system("cls");
     setTextColor(230);
    cout<<"                    ------------------------------------------------------------------------------
"<<endl;
    cout<<"                                                    ADD INVENTORY
"<<endl;
    cout<<endl;
    cout<<"                    ------------------------------------------------------------------------------
"<<endl;
     setTextColor(225);
    string name,id,function,spec,type;
    gotoxy(30,8);
    cout<<"Enter Machine ID: ";
    getline(cin,id);
     gotoxy(30,10);
    cout<<"Enter Machine Type: ";
    getline(cin,type);
     gotoxy(30,12);
    cout<<"Enter Machine name: ";
    getline(cin,name);
     gotoxy(30,14);
    cout<<"Enter Specifications: ";
    getline(cin,spec);
```

```
    gotoxy(30,16);
    cout<<"Enter Function: ";
    getline(cin,function);
    if(machines[ordernumbers][0].empty())
    {
    machines[ordernumbers][0]=id;
    machines[ordernumbers][1]=type;
    machines[ordernumbers][2]=name;
    machines[ordernumbers][3]=spec;
    machines[ordernumbers][4]=function;
    ordernumbers++;
    saveInventory(ordernumbers,machines);
    }
    gotoxy(30,18);
    system("pause");
}
void deleteInventory(string machines[][5],int totalMac,int &ordernumbers)
{
    system("cls");
     setTextColor(230);
    cout<<"                    ------------------------------------------------------------------------
"<<endl;
    cout<<"                                                DELETE INVENTORY
"<<endl;
    cout<<endl;
    cout<<"                    ------------------------------------------------------------------------
"<<endl;
     setTextColor(51);
    string id;
```

```
    gotoxy(30,8);
    cout<<"Enter Machine ID: ";
    getline(cin,id);
    int counter=0;
    for(int i=0;i<totalMac;i++)
    {
       if(machines[i][0]==id)
       {
          for(int j=i;j<ordernumbers-1;j++)
          {
             counter++;
             machines[j][0]=machines[j+1][0];
             machines[j][1]=machines[j+1][1];
             machines[j][2]=machines[j+1][2];
             machines[j][3]=machines[j+1][3];
             machines[j][4]=machines[j+1][4];
          }
          ordernumbers--;
          gotoxy(30,10);
          cout<<"Deleted Successfully";
          saveInventory(ordernumbers,machines);
       }
    }
     if(counter==0)
       {
          gotoxy(30,10);
          cout<<"Machine Not Found";
       }
```

```
    gotoxy(30,12);
   system("pause");
}
void editInventory(string machines[][5],int totalMac,int &ordernumbers)
{
  system("cls");
   setTextColor(230);
   cout<<"                    ------------------------------------------------------------------------------
"<<endl;
   cout<<"                                                EDIT INVENTORY
"<<endl;
   cout<<endl;
   cout<<"                    ------------------------------------------------------------------------------
"<<endl;
   string id,name,function,spec,type;
    setTextColor(51);
   gotoxy(30,8);
   cout<<"Edit by Machine ID: ";
   getline(cin,id);
   int counter=0;
   for(int i=0;i<totalMac;i++)
   {
      if(machines[i][0]==id)
      {
         counter++;
         gotoxy(30,10);
         cout<<"Enter Machine Type: ";
         getline(cin,type);
         gotoxy(30,12);
```

```
            cout<<"Enter Machine name: ";
            getline(cin,name);
            gotoxy(30,14);
            cout<<"Enter Specifications: ";
            getline(cin,spec);
            gotoxy(30,16);
            cout<<"Enter Function: ";
            getline(cin,function);
            machines[i][0]=id;
            machines[i][1]=type;
            machines[i][2]=name;
            machines[i][3]=spec;
            machines[i][4]=function;
            gotoxy(30,18);
            cout<<"Edited Successfully";
            saveInventory(ordernumbers,machines);
        }
}
if(counter==0)
{
   gotoxy(30,10);
   cout<<"Machine Not Found";
}
gotoxy(30,18);
system("pause");
}
void searchInventory(string machines[][5],int totalMac)
{
```

```
    system("cls");
     setTextColor(230);
    cout<<"                    ------------------------------------------------------------------------------
"<<endl;
    cout<<"                                                SEARCH INVENTORY
"<<endl;
    cout<<endl;
    cout<<"                    ------------------------------------------------------------------------------
"<<endl;
    string id,name,function,spec,type;
     setTextColor(225);
    gotoxy(30,8);
    cout<<"Enter Machine ID: ";
    getline(cin,id);
    int counter=0;
    for(int i=0;i<totalMac;i++)
    {
       if(machines[i][0]==id)
       {
          counter++;
          gotoxy(30,10);
          cout<<"Machine ID: "<<machines[i][0]<<endl;
          gotoxy(30,12);
          cout<<"Machine Type: "<<machines[i][1]<<endl;
          gotoxy(30,14);
          cout<<"Machine name: "<<machines[i][2]<<endl;
          gotoxy(30,16);
          cout<<"Specifications: "<<machines[i][3]<<endl;
          gotoxy(30,18);
```

```
            cout<<"Function: "<<machines[i][4]<<endl;
        }
    }
    if(counter==0)
{
    gotoxy(30,10);
    cout<<"Machine Not Found";
}
system("pause");
}
void InventoryManagement(int xA,int yA,string machines[][5],int totalMac,int &ordernumbers)
{
    while(true)
    {
    managers();
    InventoryManagementtitle();
    string opt=InventoryManagementmenu(xA,yA);
    if(opt=="1")
    {
        viewInventory(machines,totalMac);
        continue;
    }
    else if(opt=="2")
    {
       addInventory(machines,totalMac,ordernumbers);
       continue;

    }
```

```
   else if(opt=="3")
   {

     editInventory(machines,totalMac,ordernumbers);
     continue;

   }
   else if(opt=="4")
   {
       deleteInventory(machines,totalMac,ordernumbers);
       continue;

   }
   else if(opt=="5")
   {
      searchInventory(machines,totalMac);
      continue;
   }
   else if(opt=="6")
   {
       break;

   }
   else
   {
       cout<<"                        Invalid Choice. Please Try Again..."<<endl;
       cout<<"                          "<<endl;
       system("pause");
```

```
        continue;
    }
    }

}
 void InventoryManagementtitle()
    {
        setTextColor(51);
    cout<<"                                <<<<     INVENTORY MANAGEMENT     >>>>
"<<endl;
    cout<<"                                                                    "<<endl;
}
 string InventoryManagementmenu(int xA,int yA)
    {
        setTextColor(225);
       string option;
    string array[6]={"View Inventory","Add Inventory","Edit Inventory","Delete
Inventory","Search Inventory","Exit"};
    for(int i=0;i<6;i++)
    {
       gotoxy(xA,yA);
       cout<<i+1<<". "<<array[i]<<endl;
       yA=yA+1;
    }
    gotoxy(xA,yA);
    cout<<"Choose Option(1-6): ";
    getline(cin,option);
    return option;
    }
```

```
void ResourceAllocation(int xA,int yA,string machines[][5],int totalMac,string op_data[][5],int
NUM_OPERATORS,int maxrow,string product[],string quantity[],string date[],string
orders[][4],int MAX_ORDERS,int &allocatednum)
{
   while(true)
   {
   managers();
   resourceallocationtitle();
   string opt=resourceallocationmenu(xA,yA);
   if(opt=="1")
   {
      viewInventory(machines,totalMac);
      continue;
   }
   else if(opt=="2")
   {
     viewWorkers(op_data,NUM_OPERATORS);
     continue;
   }
   else if(opt=="3")
   {

    Placedorders(maxrow,product,quantity,date);
    continue;
   }
   else if(opt=="4")
   {
```

```
allocatedresources(maxrow,product,quantity,date,orders,MAX_ORDERS,op_data,NUM_OPER
ATORS,machines,totalMac,allocatednum);
        continue;
    }
    else if(opt=="5")
    {

unallocatedresources(product,quantity,date,orders,MAX_ORDERS,op_data,NUM_OPERATOR
S,machines,totalMac,allocatednum);
       continue;
    }
    else if(opt=="6")
    {
        break;


    }
    else
    {
        cout<<"                    Invalid Choice. Please Try Again..."<<endl;
        cout<<"                      "<<endl;
        system("pause");
        continue;
    }
    }


}
void resourceallocationtitle()
    {
         setTextColor(51);
```

```
   cout<<"                              <<<<     INVENTORY MANAGEMENT     >>>>
"<<endl;
   cout<<"                                                                  "<<endl;
}
 string resourceallocationmenu(int xA,int yA)
   {
       setTextColor(225);
      string option;
   string array[6]={"View Available Inventory","View Available Workers","View PLaced
Orders","View Allocated Resources","View Unallocated Resources","Exit"};
   for(int i=0;i<6;i++)
   {
      gotoxy(xA,yA);
      cout<<i+1<<". "<<array[i]<<endl;
      yA=yA+1;
   }
   gotoxy(xA,yA);
   cout<<"Choose Option(1-6): ";
   getline(cin,option);
   return option;
   }
 void Placedorders(int maxrow,string product[],string quantity[],string date[])
 {
    setTextColor(51);
   system("cls");
 gotoxy(50,6);
   cout<<"*******    ORDER INFORMATION     *******"<<endl;
    setTextColor(225);
       int counter=0;
```

```
    for(int x=0;x<maxrow;x++)
    {
       if(!product[x].empty()&&!quantity[x].empty()&&!date[x].empty())
       {
        counter++;
        gotoxy(30,8);
        cout<<"Ordered Product: "<<product[x]<<endl;
        gotoxy(30,10);
        cout<<"Ordered Quantity: "<<quantity[x]<<endl;
        gotoxy(30,12);
        cout<<"Expected Date: "<<date[x]<<endl;
       }
    }
    if(counter==0)
    {
        setTextColor(86);
       gotoxy(30,15);
       cout<<"No Orders Placed Yet"<<endl;
    }
    system("pause");
}
void allocatedresources(int maxrow,string product[],string quantity[],string date[],string
orders[][4],int MAX_ORDERS,string op_data[][5],int NUM_OPERATORS,string
machines[][5],int totalMac,int &allocatednum)
{
  system("cls");
   setTextColor(230);
  cout<<"                    ------------------------------------------------------------------------------
"<<endl;
```

```
   cout<<"                                        ALLOCATED RESOURCES
"<<endl;
   cout<<endl;
   cout<<"                    ------------------------------------------------------------------------------
"<<endl;
    setTextColor(225);
     bool orderplaced=false;
      int counter=0;
     for(int x=0;x<MAX_ORDERS;x++)
     {
        if((product[x]==orders[x][0])&&(quantity[x]<=orders[x][1])&&(date[x]<=orders[x][2]))
        {
          orderplaced=true;
          break;
        }
     }
     if(orderplaced==true)
     {
        for(int x=0;x<maxrow;x++)
        {
           if(!product[x].empty()&&!quantity[x].empty()&&!date[x].empty())
           {
            counter++;
            allocatednum++;
            gotoxy(30,6);
            cout<<"Allocated Product: "<<product[x]<<endl;
            gotoxy(30,8);
            cout<<"Allocated Quantity: "<<quantity[x]<<endl;
            gotoxy(30,10);
```

```
            cout<<"Allocated Date: "<<date[x]<<endl;
            gotoxy(30,12);
            cout<<"Allocated Operator: "<<op_data[x][0]<<endl;
            gotoxy(30,14);
            cout<<"Allocated Machine: "<<machines[x][1]<<endl;
           }
        }
      }
      if(counter==0)
      {
         setTextColor(86);
        gotoxy(50,13);
        cout<<"No Resources Allocated Yet"<<endl;
      }
      system("pause");
}
 void unallocatedresources(string product[],string quantity[],string date[],string orders[][4],int
MAX_ORDERS,string op_data[][5],int NUM_OPERATORS,string machines[][5],int
totalMac,int &allocatednum)
 {
   system("cls");
    setTextColor(230);
   cout<<"                    ------------------------------------------------------------------------------
"<<endl;
   cout<<"                                                UNALLOCATED RESOURCES
"<<endl;
   cout<<endl;
   cout<<"                    ------------------------------------------------------------------------------
"<<endl;
 setTextColor(51);
```

```
gotoxy(0,7);
cout<<"Unallocated Product";
gotoxy(20,7);
cout<<"|";
gotoxy(25,7);
cout<<"Unallocated Quantity";
gotoxy(45,7);
cout<<"|";
gotoxy(50,7);
cout<<"Unallocated Date";
 gotoxy(70,7);
cout<<"|";
gotoxy(75,7);
cout<<"Unallocated Operator";
gotoxy(95,7);
cout<<"|";
gotoxy(100,7);
cout<<"Unallocated Machines";
 setTextColor(225);
int y=9;
int o=y,p=y,q=y,r=y;
for(int i=0;i<5;i++)
{
   for(int j=allocatednum;j<100;j++)
   {
      if(i==0)
      {
       gotoxy(0,y);
```

```
 cout<<orders[j][i];
  if(!orders[j][i].empty())
 {
 gotoxy(20,y);
 cout<<"|";
 }
 y++;
 gotoxy(75,q);
 cout<<op_data[j][i];
if(!op_data[j][i].empty())
{
     gotoxy(95,q);
     cout<<"|";
}
 q++;
}
else if(i==1)
{
   gotoxy(25,o);
 cout<<orders[j][i];
 if(!orders[j][i].empty())
 {
 gotoxy(45,o);
 cout<<"|";
 }
 o++;
 gotoxy(100,r);
 cout<<machines[j][i];
```

```
                r++;
               }
               else if(i==2)
               {
                gotoxy(50,p);
                cout<<orders[j][i];
                 if(!orders[j][i].empty())
                {
                gotoxy(70,p);
                cout<<"|";
                }
                p++;
               }


           }
       }
       system("pause");
}
void addRawmaterials(string  raw_materials[][5],int MAX_Rawmat,int &numRawmat)
{
   system("cls");
    setTextColor(230);
   cout<<"                    ---------------------------------------------------------------------------
"<<endl;
   cout<<"                                                ADD RAW MATERIALS
"<<endl;
   cout<<endl;
   cout<<"                    ---------------------------------------------------------------------------
"<<endl;
```

```
 setTextColor(225);
string fert_raw,starch_raw,inv_qty,location,suppliers;
gotoxy(30,8);
cout<<"Enter Fertilizer Raw Material: ";
getline(cin,fert_raw);
 gotoxy(30,10);
cout<<"Enter Starch Raw Material: ";
getline(cin,starch_raw);
 gotoxy(30,12);
cout<<"Enter Available Inventories: ";
getline(cin,inv_qty);
 gotoxy(30,14);
cout<<"Enter Location: ";
getline(cin,location);
gotoxy(30,16);
cout<<"Enter Suppliers: ";
getline(cin,suppliers);
if(raw_materials[numRawmat][0].empty())
{
raw_materials[numRawmat][0]=starch_raw;
raw_materials[numRawmat][1]=fert_raw;
raw_materials[numRawmat][2]=inv_qty;
raw_materials[numRawmat][3]=location;
raw_materials[numRawmat][4]=suppliers;
numRawmat++;
saveRawMaterials(numRawmat,raw_materials);
}
gotoxy(30,18);
```

```
    system("pause");
}
void deleteRawmaterials(string  raw_materials[][5],int MAX_Rawmat,int &numRawmat)
{
    system("cls");
     setTextColor(230);
    cout<<"                    ------------------------------------------------------------------------------
"<<endl;
    cout<<"                                                DELETE RAW MATERIALS
"<<endl;
    cout<<endl;
    cout<<"                    ------------------------------------------------------------------------------
"<<endl;
     setTextColor(225);
    string fert,starch;
    gotoxy(30,8);
    cout<<"Enter Fertilzer Raw Material: ";
    getline(cin,fert);
    gotoxy(30,10);
    cout<<"Enter Starch Raw Material: ";
    getline(cin,starch);
    int counter=0;
    for(int i=0;i<MAX_Rawmat;i++)
    {
        if(raw_materials[i][0]==starch && raw_materials[i][1]==fert)
        {
            for(int j=i;j<numRawmat;j++)
            {
                counter++;
```

```
            raw_materials[j][0]=raw_materials[j+1][0];
            raw_materials[j][1]=raw_materials[j+1][1];
            raw_materials[j][2]=raw_materials[j+1][2];
            raw_materials[j][3]=raw_materials[j+1][3];
            raw_materials[j][4]=raw_materials[j+1][4];
          }
          numRawmat--;
          gotoxy(30,12);
          cout<<"Deleted Successfully";
          saveRawMaterials(numRawmat,raw_materials);
       }
    }
     if(counter==0)
       {
          gotoxy(30,10);
          cout<<"Machine Not Found";
       }
     gotoxy(30,12);
    system("pause");
}
void editRawmaterials(string  raw_materials[][5],int MAX_Rawmat,int &numRawmat)
{
   system("cls");
    setTextColor(230);
    cout<<"                    ------------------------------------------------------------------------------
"<<endl;
    cout<<"                                                EDIT RAW MATERIALS
"<<endl;
    cout<<endl;
```

```
   cout<<"                    ------------------------------------------------------------------------------
"<<endl;
    setTextColor(225);
   string fert_raw,starch_raw,inv_qty,location,suppliers;
   gotoxy(30,8);
   cout<<"Edit by Material Location: ";
   getline(cin,location);
   int counter=0;
   for(int i=0;i<MAX_Rawmat;i++)
   {
      if(raw_materials[i][3]==location)
      {
         counter++;
         gotoxy(30,10);
         cout<<"Enter Fertilizer Raw Materials: ";
         getline(cin,fert_raw);
         gotoxy(30,12);
         cout<<"Enter Starch Raw Materials: ";
         getline(cin,starch_raw);
         gotoxy(30,14);
         cout<<"Enter Suppliers: ";
         getline(cin,suppliers);
         gotoxy(30,16);
         cout<<"Enter Available Inventory: ";
         getline(cin,inv_qty);
         raw_materials[i][0]=starch_raw;
         raw_materials[i][1]=fert_raw;
         raw_materials[i][2]=inv_qty;
         raw_materials[i][3]=location;
```

```
            raw_materials[i][4]=suppliers;
            gotoxy(30,18);
            cout<<"Edited Successfully";
            saveRawMaterials(numRawmat,raw_materials);
        }
}
if(counter==0)
{
    gotoxy(30,10);
    cout<<"Location Not Found";
}
gotoxy(30,18);
system("pause");
}
void searchRawmaterials(string  raw_materials[][5],int MAX_Rawmat)
{
    system("cls");
     setTextColor(230);
    cout<<"                    ------------------------------------------------------------------------------
"<<endl;
    cout<<"                                                SEARCH RAW MATERIALS
"<<endl;
    cout<<endl;
    cout<<"                    ------------------------------------------------------------------------------
"<<endl;
     setTextColor(225);
    string fert_raw,starch_raw,inv_qty,location,suppliers;
    gotoxy(30,8);
    cout<<"Enter Location: ";
```

```
   getline(cin,location);
   int counter=0;
   for(int i=0;i<MAX_Rawmat;i++)
   {
      if(raw_materials[i][3]==location)
      {
         counter++;
         gotoxy(30,10);
         cout<<"Starch Raw Material: "<<raw_materials[i][0]<<endl;
         gotoxy(30,12);
         cout<<"Fertilizer Raw Materials: "<<raw_materials[i][1]<<endl;
         gotoxy(30,14);
         cout<<"Available Inventory: "<<raw_materials[i][2]<<endl;
         gotoxy(30,16);
         cout<<"Location: "<<raw_materials[i][3]<<endl;
         gotoxy(30,18);
         cout<<"Suppliers: "<<raw_materials[i][4]<<endl;
      }
   }
   if(counter==0)
{
   gotoxy(30,10);
   cout<<"Location Not Found";
}
system("pause");
}
void MaterialHandling(int xA,int yA,string  raw_materials[][5],int MAX_Rawmat,int
&numRawmat)
{
```

```
while(true)
{
managers();
MaterialHandlingtitle();
string opt=MaterialHandlingmenu(xA,yA);
if(opt=="1")
{
    viewRawMaterials(raw_materials,MAX_Rawmat);
    continue;
}
else if(opt=="2")
{
    addRawmaterials(raw_materials,MAX_Rawmat,numRawmat);
   continue;

}
else if(opt=="3")
{
  editRawmaterials(raw_materials,MAX_Rawmat,numRawmat);
  continue;



}
else if(opt=="4")
{
    deleteRawmaterials(raw_materials,MAX_Rawmat,numRawmat);
    continue;
```

```
    }
    else if(opt=="5")
    {
        searchRawmaterials(raw_materials,MAX_Rawmat);
       continue;
    }
    else if(opt=="6")
    {
        break;

    }
    else
    {
        cout<<"                         Invalid Choice. Please Try Again..."<<endl;
        cout<<"                          "<<endl;
        system("pause");
        continue;
    }
    }

}
 void MaterialHandlingtitle()
    {
         setTextColor(51);
    cout<<"                                <<<<    MATERIAL HANDLING    >>>>
"<<endl;
    cout<<"                                                                  "<<endl;
}
 string MaterialHandlingmenu(int xA,int yA)
```

```
   {
        setTextColor(225);
       string option;
   string array[6]={"View Raw Materials","Add Raw Materials","Edit Raw Materials","Delete
Raw Materials","Search Raw Materials","Exit"};
   for(int i=0;i<6;i++)
   {
      gotoxy(xA,yA);
      cout<<i+1<<". "<<array[i]<<endl;
      yA=yA+1;
   }
   gotoxy(xA,yA);
   cout<<"Choose Option(1-6): ";
   getline(cin,option);
   return option;
   }
   void viewRawMaterials(string  raw_materials[][5],int MAX_Rawmat)
{
   system("cls");
    setTextColor(230);
   cout<<"                    ------------------------------------------------------------------------------
"<<endl;
   cout<<"                                                RAW MATERIALS INFORMATION
"<<endl;
   cout<<endl;
   cout<<"                    ------------------------------------------------------------------------------
"<<endl;
       setTextColor(51);
       gotoxy(0,7);
       cout<<"Starch Materials";
```

```
gotoxy(25,7);
cout<<"|";
gotoxy(30,7);
cout<<"Fertilizer Materials";
gotoxy(55,7);
cout<<"|";
gotoxy(60,7);
cout<<"Available Inventory";
 gotoxy(80,7);
cout<<"|";
gotoxy(85,7);
cout<<"Location";
gotoxy(100,7);
cout<<"|";
gotoxy(105,7);
cout<<"Suppliers";
 setTextColor(225);
int y=9;
int o=y,p=y,q=y,r=y;
for(int i=0;i<5;i++)
{
   for(int j=0;j<MAX_Rawmat;j++)
   {
      if(i==0)
      {
       gotoxy(0,y);
       cout<<raw_materials[j][i];
        if(!raw_materials[j][i].empty())
```

```
 {
 gotoxy(25,y);
 cout<<"|";
 }
 y++;
}
else if(i==1)
{
  gotoxy(30,o);
 cout<<raw_materials[j][i];
 if(!raw_materials[j][i].empty())
 {
 gotoxy(55,o);
 cout<<"|";
 }
 o++;

}
else if(i==2)
{
 gotoxy(60,p);
 cout<<raw_materials[j][i];
  if(!raw_materials[j][i].empty())
 {
 gotoxy(80,p);
 cout<<"|";
 }
 p++;
```

```
            }
            else if(i==3)
            {
             gotoxy(85,q);
             cout<<raw_materials[j][i];
            if(!raw_materials[j][i].empty())
            {
                gotoxy(100,q);
                cout<<"|";
            }
             q++;
            }
             else if(i==4)
            {
             gotoxy(105,r);
             cout<<raw_materials[j][i];
             r++;
            }
          }
       }
       system("pause");
}


void GenerateReport(int &bill,double productPrices[],string saledName[],double
saledQuantity[],int maxrow,int &count,string productNames[],double productQuantity[])
{
    system("cls");
     setTextColor(230);
       gotoxy(35,3);
```

```
cout<<"          <<<<     Generate Report   >>>>      "<<endl;
cout<<"                                               "<<endl;
 setTextColor(51);
gotoxy(30,5);
cout<<"-----------------------------------------------------------------------"<<endl;
gotoxy(30,6);
cout<<"                    SOLD PRODUCTS                  "<<endl;
gotoxy(30,7);
cout<<"-----------------------------------------------------------------------"<<endl;
gotoxy(30,8);
cout<<"                                                   "<<endl;
gotoxy(30,10);
int y=10;
int counter=0;
double revenue=0;
 double SALESPRODUCTIVITY;
  setTextColor(225);
if(bill!=0)
{
   for (int i = 0; i < count; i++)
{
for(int j=0;j<10;j++)
{
      if ((saledName[i] == productNames[j]) && (saledQuantity[i] <= productQuantity[j]))
   {
counter++;
gotoxy(30,y);
cout<<"Sold Product Names: "<<saledName[i]<<endl;
```

```
        gotoxy(30,y+2);
        cout<<"Sold Product Quantity: "<<saledQuantity[i]<<endl;
        gotoxy(30,y+4);
        cout<<"Gained Profit: $ "<<productPrices[i]<<endl;
        y=y+6;
        revenue=revenue+productPrices[i];
        }
        }
    }
        cout<<endl;
        setTextColor(31);
        SALESPRODUCTIVITY=(revenue/bill);
        cout<<"\t\t\t\t\t\t"<<"SALES PRODUCTIVITY: $ "<<SALESPRODUCTIVITY<<endl;
        }

        if(counter==0)
        {
           setTextColor(86);
           gotoxy(50,16);
           cout<<"No Sales Yet!";
        }
        gotoxy(30,29);
system("pause");



}
```

```
void GenerateProductionReports(int &bill,double productPrices[],string saledName[],double
saledQuantity[],int maxrow,int &count,string productNames[],double productQuantity[])
{
	system("cls");
	 setTextColor(230);
	gotoxy(35,3);
	cout<<"          <<<<      Generate Report   >>>>       "<<endl;
	cout<<"                                                  "<<endl;
	 setTextColor(51);
	gotoxy(30,5);
	cout<<"-------------------------------------------------------------------------"<<endl;
	gotoxy(30,6);
	cout<<"                        SOLD PRODUCTS                    "<<endl;
	gotoxy(30,7);
	cout<<"-------------------------------------------------------------------------"<<endl;
	gotoxy(30,8);
	cout<<"                                                    "<<endl;
	 setTextColor(225);
	gotoxy(30,10);
	int y=10;
	int counter=0;
	double revenue=0;
	 double SALESPRODUCTIVITY;
	if(bill!=0)
	{
	    for (int i = 0; i < count; i++)
   {
	for(int j=0;j<10;j++)
	{
```

```
            if ((saledName[i] == productNames[j]) && (saledQuantity[i] <= productQuantity[j]))
          {
      counter++;
      gotoxy(30,y);
      cout<<"Sold Product Names: "<<saledName[i]<<endl;
      gotoxy(30,y+2);
      cout<<"Sold Product Quantity: "<<saledQuantity[i]<<endl;
      gotoxy(30,y+4);
      cout<<"Gained Profit: $ "<<productPrices[i]<<endl;
      y=y+6;
      revenue=revenue+productPrices[i];
      }
      }
   }
      setTextColor(31);
      SALESPRODUCTIVITY=(revenue/bill);
      cout<<"\t\t\t\t\t\t"<<"SALES PRODUCTIVITY: $ "<<SALESPRODUCTIVITY<<endl;
      }

      if(counter==0)
      {
         setTextColor(31);
         gotoxy(50,16);
         cout<<"No Sales Yet!";
      }
      gotoxy(30,29);
system("pause");
```

```
}

string BudgetingandCostControlmenu(int xA,int yA)
{
    setTextColor(225);
   string option;
   string array[7]={"Input Workers Data","Input Inventory Data","Input Raw Materials Data","Budget Calculation of Workers","Budget Calculation of Inventory","Budget Calculation of Raw Materials","Exit"};
   for(int i=0;i<7;i++)
   {
      gotoxy(xA,yA);
      cout<<i+1<<". "<<array[i]<<endl;
      yA=yA+1;
   }
   gotoxy(xA,yA);
   cout<<"Choose Option(1-7): ";
   getline(cin,option);
   return option;
}
void BudgetingandCostControl(int xA,int yA,int &mac,int &civil,int &chem,int &oper,int &tech,int &elec,int &lathe,int &MillingMac,int &Drill,int &Bandsaw,int &Grinder,int &Planar,int &corn,int &sago,int &PhosphateRock,int &Ammonia,int &Potash,int &Nitrogen,string num)
{
    while(true)
      {
      managers();
      BudgetingandCostControltitle();
```

```
string opt=BudgetingandCostControlmenu(xA,yA);
if(opt=="1")
 {
 InputdataWorkers(mac,civil,chem,oper,tech,elec,num);
 continue;
 }
 else if(opt=="2")
 {
 InputdataInventory(lathe,MillingMac,Drill,Bandsaw,Grinder,Planar,num);
 continue;
 }
 else if(opt=="3")
 {
 InputdataRaw(corn,sago,PhosphateRock,Ammonia,Potash,Nitrogen,num);
 continue;
 }
 else if(opt=="4")
 {
 budgetcalculationWorkers(mac,civil,chem,oper,tech,elec);
 continue;
 }
 else if(opt=="5")
 {
 budgetcalculationInventory(lathe,MillingMac,Drill,Bandsaw,Grinder,Planar);
 continue;
 }
 else if(opt=="6")
 {
```

```
        budgetcalculationRaw(corn,sago,PhosphateRock,Ammonia,Potash,Nitrogen);
       continue;
        }
       else if(opt=="7")
        {

        break;
        }
        else
        {
        cout<<"                         Invalid Choice. Please Try Again..."<<endl;
        cout<<"                          "<<endl;
        system("pause");
        continue;
        }
       }
}
 void BudgetingandCostControltitle()
    {
         setTextColor(51);
    cout<<"                                <<<<    BUDGETING AND COST CONTROL    >>>>
"<<endl;
    cout<<"                                                                    "<<endl;
}

void InputdataWorkers(int &mac,int &civil,int &chem,int &oper,int &tech,int &elec,string num)
{
     setTextColor(230);
    system("cls");
```

```
gotoxy(35,5);
cout << "--------------------------------------------------------------------------" << endl;
gotoxy(35, 6);
cout << "                          DATA INPUT                                    " << endl;
gotoxy(35, 7);
cout << "--------------------------------------------------------------------------" << endl;
gotoxy(35, 8);
cout << "                                                                        " << endl;
 setTextColor(225);
while(true)
{
gotoxy(30,10);
cout<<"Enter Number of Mechanical Enjineers: ";
cin>>num;
if(Salary_Validations(num))
   {
     mac=stoi(num);
     break;
     }
     else
{
    gotoxy(30,11);
    cout<<" Invalid Input!";
    getch();
    gotoxy(30,10);
    cout<<"                                                                    ";
    gotoxy(30,11);
    cout<<"                                                                    ";
```

```
    }

    }
     while(true)
    {
    gotoxy(30,12);
    cout<<"Enter Number of Electrical Enjineers: ";
    cin>>num;
    if(Salary_Validations(num))
       {
         elec=stoi(num);
         break;
         }
         else
    {
        gotoxy(30,13);
        cout<<" Invalid Input!";
        getch();
        gotoxy(30,12);
        cout<<"                                                                    ";
        gotoxy(30,13);
        cout<<"                                                                     ";
    }

    }
      while(true)
    {
    gotoxy(30,14);
```

```
cout<<"Enter Number of Chemical Enjineers: ";
cin>>num;
if(Salary_Validations(num))
  {
   chem=stoi(num);
   break;
   }
   else
{
  gotoxy(30,15);
  cout<<" Invalid Input!";
  getch();
  gotoxy(30,14);
  cout<<"                                                                    ";
  gotoxy(30,15);
  cout<<"                                                                     ";
}

}
 while(true)
{
 gotoxy(30,16);
cout<<"Enter Number of Civil Enjineers: ";
cin>>num;
if(Salary_Validations(num))
  {
   civil=stoi(num);
   break;
```

```
        }
        else
    {
       gotoxy(30,17);
       cout<<" Invalid Input!";
       getch();
       gotoxy(30,16);
       cout<<"                                                                    ";
       gotoxy(30,17);
       cout<<"                                                                      ";
    }

    }
     while(true)
    {
    gotoxy(30,18);
    cout<<"Enter Number of Operators: ";
    cin>>num;
    if(Salary_Validations(num))
       {
        oper=stoi(num);
        break;
        }
        else
    {
       gotoxy(30,19);
       cout<<" Invalid Input!";
       getch();
```

```
    gotoxy(30,18);
    cout<<"                                                    ";
    gotoxy(30,19);
    cout<<"                                                     ";
  }

  }
   while(true)
  {
  gotoxy(30,20);
  cout<<"Enter Number of Technicians: ";
  cin>>num;
  if(Salary_Validations(num))
    {
     tech=stoi(num);
     break;
     }
     else
  {
    gotoxy(30,21);
    cout<<" Invalid Input!";
    getch();
    gotoxy(30,20);
    cout<<"                                                   ";
    gotoxy(30,21);
    cout<<"                                                    ";
  }
```

```
    }
     setTextColor(86);
    bool isfound=false;
    if((mac>=0 && mac <=100 ) && (elec>=0 && elec<=100) && (civil>=0 && civil <=100)
&& (chem>=0 && chem<=100) && (oper>=0 && oper <=100) && (tech>=0 && tech <=100))
     {
        isfound=true;
     }
     if(isfound==true)
     {
        gotoxy(35,22);
        {
           cout<<"Data is submitted successfully!";
        }
     }
     else
     {
       gotoxy(35,22);
       cout<<"Invalid Data entered...Please TRY AGAIN!";
     }
     gotoxy(35,23);
     system("pause");
}
void budgetcalculationWorkers(int &mac,int &civil,int &chem,int &oper,int &tech,int &elec)
{
    system("cls");
     setTextColor(230);
    gotoxy(35,5);
    cout << "-----------------------------------------------------------------------" << endl;
```

```
gotoxy(35, 6);
cout << "                          BUDGET CALCULATION                          " << endl;
gotoxy(35, 7);
cout << "----------------------------------------------------------------------" << endl;
gotoxy(35, 8);
cout << "                                                                      " << endl;
 setTextColor(225);
int mecsal,elecsal,civilsal,chemsal,opersal,techsal;
if(mac!=0 && elec!=0 && civil!=0 && oper!=0 && tech!=0 && chem!=0)
{
mecsal=mac*50000;
elecsal=elec*40000;
civilsal=civil*50000;
chemsal=chem*70000;
opersal=oper*50000;
techsal=tech*45000;
gotoxy(30,10);
cout<<"Allocated Salaries to Mechanical Enjineers: "<<mecsal;
 gotoxy(30,12);
cout<<"Allocated Salaries to Electrical Enjineers: "<<elecsal;
 gotoxy(30,14);
cout<<"Allocated Salaries to Civil Enjineers: "<<civilsal;
gotoxy(30,16);
cout<<"Allocated Salaries to Chemical Enjineers: "<<chemsal;
gotoxy(30,18);
cout<<"Allocated Salaries to Operators: "<<opersal;
gotoxy(30,20);
cout<<"Allocated Salaries to technicians: "<<techsal;
```

```
    }
    else
    {
        gotoxy(50,12);
        cout<<"YET DATA HAS NOT BEEN SUBMITTED!";
    }
    gotoxy(50,21);
    system("pause");

}

void InputdataRaw(int &corn,int &sago,int &PhosphateRock,int &Ammonia,int &Potash,int
&Nitrogen,string num)
{
    system("cls");
     setTextColor(230);
    gotoxy(35,5);
    cout << "--------------------------------------------------------------------------" << endl;
    gotoxy(35, 6);
    cout << "                          DATA INPUT                                    " << endl;
    gotoxy(35, 7);
    cout << "--------------------------------------------------------------------------" << endl;
    gotoxy(35, 8);
    cout << "                                                                        " << endl;
     setTextColor(225);
    while(true)
    {
     gotoxy(30,10);
    cout<<"Enter Number of Corn Stocks: ";
```

```
    cin>>num;
    if(Salary_Validations(num))
      {
       corn=stoi(num);
       break;
       }
       else
    {
       gotoxy(30,11);
       cout<<" Invalid Input!";
       getch();
       gotoxy(35,10);
       cout<<"                                                    ";
       gotoxy(30,11);
       cout<<"                                                     ";
    }


    }
    while(true)
    {
      gotoxy(30,12);
    cout<<"Enter Number of Sago: ";
    cin>>num;
    if(Salary_Validations(num))
      {
       sago=stoi(num);
       break;
       }
```

```
        else
    {
       gotoxy(30,13);
       cout<<" Invalid Input!";
       getch();
       gotoxy(35,12);
       cout<<"                                                                    ";
       gotoxy(30,13);
       cout<<"                                                                     ";
    }

    }
     while(true)
    {
      gotoxy(30,14);
    cout<<"Enter Number of PhosphateRock Stocks: ";
    cin>>num;
    if(Salary_Validations(num))
      {
       PhosphateRock=stoi(num);
        break;
        }
        else
    {
       gotoxy(30,15);
       cout<<" Invalid Input!";
       getch();
       gotoxy(35,14);
```

```
    cout<<"                                                    ";
    gotoxy(30,15);
    cout<<"                                                    ";
  }

  }
   while(true)
  {
    gotoxy(30,16);
  cout<<"Enter Number of Ammonia Stocks: ";
  cin>>num;
  if(validity_checker(num))
    {
     Ammonia=stoi(num);
     break;
     }
     else
  {
    gotoxy(30,17);
    cout<<" Invalid Input!";
    getch();
    gotoxy(35,16);
    cout<<"                                                    ";
    gotoxy(30,17);
    cout<<"                                                    ";
  }

  }
```

```
while(true)
{
   gotoxy(30,18);
cout<<"Enter Number of Potash Stocks: ";
cin>>num;
if(validity_checker(num))
  {
   Potash=stoi(num);
   break;
   }
   else
{
   gotoxy(30,17);
   cout<<" Invalid Input!";
   getch();
   gotoxy(35,16);
   cout<<"                                                                    ";
   gotoxy(30,17);
   cout<<"                                                                    ";
}

}
while(true)
{
   gotoxy(30,20);
cout<<"Enter Number of Nitrogen Stocks: ";
cin>>num;
if(validity_checker(num))
```

```
      {
       Nitrogen=stoi(num);
       break;
       }
       else
   {
      gotoxy(30,21);
      cout<<" Invalid Input!";
      getch();
      gotoxy(35,20);
      cout<<"                                                                    ";
      gotoxy(30,21);
      cout<<"                                                                    ";
   }
   }
   setTextColor(86);
   bool isfound=false;
   if((corn>=0 && corn <=100 ) && (sago>=0 && sago<=100) && (PhosphateRock>=0 &&
PhosphateRock <=100) && (Ammonia>=0 && Ammonia<=100) && (Potash>=0 && Potash
<=100) && (Nitrogen>=0 && Nitrogen <=100))
    {
       isfound=true;
    }
    if(isfound==true)
    {
       gotoxy(35,22);
       {
          cout<<"Data is submitted successfully!";
       }
```

```
    }
    else
    {
      gotoxy(35,22);
      cout<<"Invalid Data entered...Please TRY AGAIN!";
    }
    gotoxy(35,23);
    system("pause");
}
void InputdataInventory(int &lathe,int &MillingMac,int &Drill,int &Bandsaw,int &Grinder,int
&Planar,string num)
{
   system("cls");
    setTextColor(230);
   gotoxy(35,5);
   cout << "--------------------------------------------------------------------------" << endl;
   gotoxy(35, 6);
   cout << "                          DATA INPUT                          " << endl;
   gotoxy(35, 7);
   cout << "--------------------------------------------------------------------------" << endl;
   gotoxy(35, 8);
   cout << "                                                              " << endl;
    setTextColor(225);
    while(true)
   {
       gotoxy(30,10);
   cout<<"Enter Number of Lathe Machines: ";
   cin>>num;
   if(validity_checker(num))
```

```
    {
     lathe=stoi(num);
     break;
     }
     else
  {
    gotoxy(30,11);
    cout<<" Invalid Input!";
    getch();
    gotoxy(35,10);
    cout<<"                                                                    ";
    gotoxy(30,11);
    cout<<"                                                                    ";
  }
  }
   while(true)
  {
      gotoxy(30,12);
  cout<<"Enter Number of Milling Machines: ";
  cin>>num;
  if(validity_checker(num))
    {
     MillingMac=stoi(num);
     break;
     }
     else
  {
    gotoxy(30,13);
```

```
        cout<<" Invalid Input!";
        getch();
        gotoxy(35,12);
        cout<<"                                                    ";
        gotoxy(30,13);
        cout<<"                                                    ";
    }
    }
     while(true)
    {
         gotoxy(30,14);
    cout<<"Enter Number of Planars: ";
    cin>>num;
    if(validity_checker(num))
       {
        Planar=stoi(num);
         break;
         }
         else
    {
        gotoxy(30,15);
        cout<<" Invalid Input!";
        getch();
        gotoxy(35,14);
        cout<<"                                                   ";
        gotoxy(30,15);
        cout<<"                                                    ";
    }
```

```
    }
     while(true)
    {
       gotoxy(30,16);
    cout<<"Enter Number of Grinders: ";
    cin>>num;
    if(validity_checker(num))
      {
       Grinder=stoi(num);
       break;
       }
       else
    {
       gotoxy(30,17);
       cout<<" Invalid Input!";
       getch();
       gotoxy(35,16);
       cout<<"                                                                    ";
       gotoxy(30,17);
       cout<<"                                                                    ";
    }
    }
     while(true)
    {
       gotoxy(30,18);
    cout<<"Enter Number of BandSaws: ";
    cin>>num;
    if(validity_checker(num))
```

```
    {
     Bandsaw=stoi(num);
     break;
     }
     else
  {
    gotoxy(30,19);
    cout<<" Invalid Input!";
    getch();
    gotoxy(35,18);
    cout<<"                                                    ";
    gotoxy(30,19);
    cout<<"                                                    ";
  }
  }
   while(true)
  {
    gotoxy(30,20);
  cout<<"Enter Number of Drills: ";
  cin>>num;
  if(validity_checker(num))
    {
     Drill=stoi(num);
     break;
     }
     else
  {
    gotoxy(30,21);
```

```
        cout<<" Invalid Input!";
        getch();
        gotoxy(35,20);
        cout<<"                                                                    ";
        gotoxy(30,21);
        cout<<"                                                                    ";
    }
    }
    setTextColor(86);
    bool isfound=false;
    if((lathe>=0 && lathe <=100 ) && (MillingMac>=0 && MillingMac<=100) && (Drill>=0
&& Drill <=100) && (Bandsaw>=0 && Bandsaw<=100) && (Grinder>=0 && Grinder <=100)
&& (Planar>=0 && Planar <=100))
     {
        isfound=true;
     }
     if(isfound==true)
     {
        gotoxy(35,22);
        {
           cout<<"Data is submitted successfully!";
        }
     }
     else
     {
        gotoxy(35,22);
        cout<<"Invalid Data entered...Please TRY AGAIN!";
     }
     gotoxy(35,23);
```

```
    system("pause");
}

void budgetcalculationInventory(int &lathe,int &MillingMac,int &Drill,int &Bandsaw,int
&Grinder,int &Planar)
{
   system("cls");
    setTextColor(230);
   gotoxy(35,5);
   cout << "--------------------------------------------------------------------------" << endl;
   gotoxy(35, 6);
   cout << "                          BUDGET CALCULATION                          " << endl;
   gotoxy(35, 7);
   cout << "--------------------------------------------------------------------------" << endl;
   gotoxy(35, 8);
   cout << "                                                                      " << endl;
    setTextColor(225);
   int lathsal,milsal,drillsal,bandsal,grindsal,planarsal;
   if(lathe!=0 && MillingMac!=0 && Drill!=0 && Bandsaw!=0 && Grinder!=0 && Planar!=0)
   {
   lathsal=lathe*50000;
   drillsal=Drill*40000;
   bandsal=Bandsaw*50000;
   grindsal=Grinder*70000;
   planarsal=Planar*50000;
   milsal=MillingMac*45000;
   gotoxy(30,10);
   cout<<"Allocated Budget to Lathe Machines: "<<lathsal;
    gotoxy(30,12);
```

```
    cout<<"Allocated Budget to Milling Machines: "<<milsal;
     gotoxy(30,14);
    cout<<"Allocated Budget to Planars: "<<planarsal;
    gotoxy(30,16);
    cout<<"Allocated Budget to Grinders: "<<grindsal;
    gotoxy(30,18);
    cout<<"Allocated Budget to BandSaws: "<<bandsal;
    gotoxy(30,20);
    cout<<"Allocated Budget to Drills: "<<drillsal;
    }
    else
    {
       setTextColor(86);
       gotoxy(50,12);
       cout<<"YET DATA HAS NOT BEEN SUBMITTED!";
    }
    gotoxy(50,21);
    system("pause");


}
void budgetcalculationRaw(int &corn,int &sago,int &PhosphateRock,int &Ammonia,int
&Potash,int &Nitrogen)
{
    system("cls");
     setTextColor(230);
    gotoxy(35,5);
    cout << "--------------------------------------------------------------------------" << endl;
    gotoxy(35, 6);
    cout << "                          BUDGET CALCULATION                          " << endl;
```

```
    gotoxy(35, 7);
    cout << "--------------------------------------------------------------------------" << endl;
    gotoxy(35, 8);
    cout << "                                                          " << endl;
     setTextColor(225);
    int cornsal,sagosal,phosrocksal,ammosal,potashsal,nitsal;
    if(corn!=0 && sago!=0 && PhosphateRock!=0 && Ammonia!=0 && Potash!=0 &&
Nitrogen!=0)
    {
    cornsal=corn*5000;
    sagosal=sago*4000;
    phosrocksal=PhosphateRock*5000;
    ammosal=Ammonia*7000;
    potashsal=Potash*5000;
    nitsal=Nitrogen*4500;
    gotoxy(30,10);
    cout<<"Allocated Budget to Corn: "<<cornsal;
     gotoxy(30,12);
    cout<<"Allocated Budget to Sago: "<<sagosal;
     gotoxy(30,14);
    cout<<"Allocated Budget to Phosphate Rock: "<<phosrocksal;
    gotoxy(30,16);
    cout<<"Allocated Budget to Ammonia: "<<ammosal;
    gotoxy(30,18);
    cout<<"Allocated Budget to Potash: "<<potashsal;
    gotoxy(30,20);
    cout<<"Allocated Budget to Nitrogen: "<<nitsal;
    }
    else
```

```
    {
        setTextColor(86);
       gotoxy(50,12);
       cout<<"YET DATA HAS NOT BEEN SUBMITTED!";
    }
   gotoxy(50,21);
   system("pause");


}
void usermanagement(int xA,int yA,string empname[],int empid[],int empID,int maxrow,string
num)
{


   while(true)
   {
   administrator();
   usermanagementtitle();
   string opt=usermanagementmenu(xA,yA);
   if(opt=="1")
   {
      system("cls");
      usermanagementtitle();
      Addrecord(empname,empid,maxrow,num);
      continue;
   }
   else if(opt=="2")
   {
     while(true)
   {
```

```
    gotoxy(30,23);
  cout<<"                  Search by ID: ";
  getline(cin,num);
  if(ID_validation(num))
    {
     empID=stoi(num);
     break;
     }
     else
  {
    gotoxy(30,24);
    cout<<" Invalid Input! Please Enter ID which consists of 5 digits...";
    getch();
    gotoxy(30,23);
    cout<<"                                          ";
    gotoxy(30,24);
    cout<<"                                           ";

  }

  }
    UpdateRecord(maxrow,empID,empid,empname);
    continue;

  }
  else if(opt=="3")
  {
```

```
    while(true)
  {
     gotoxy(30,23);
  cout<<"                        Search by ID: ";
  getline(cin,num);
  if(ID_validation(num))
    {
     empID=stoi(num);
     break;
     }
     else
  {
    gotoxy(30,24);
    cout<<" Invalid Input! Please Enter ID which consists of 5 digits...";
    getch();
    gotoxy(30,23);
    cout<<"                                                          ";
    gotoxy(30,24);
    cout<<"                                                           ";

  }

  }
   DeleteRecord(empID,maxrow,empid,empname);
   continue;


  }
```

```
else if(opt=="4")
{
    while(true)
{
   gotoxy(30,23);
cout<<"                    Search by ID: ";
getline(cin,num);
if(ID_validation(num))
  {
    empID=stoi(num);
    break;
    }
    else
{
    gotoxy(30,24);
   cout<<" Invalid Input! Please Enter ID which consists of 5 digits...          ";
   getch();
   gotoxy(30,23);
   cout<<"                                                                      ";
   gotoxy(30,24);
   cout<<"                                                                     ";



}


}
   searchrecord(empID,maxrow,empid,empname);
   continue;
```

```
    }
    else if(opt=="5")
    {
      listrecord(maxrow,empid,empname);
      continue;
    }
    else if(opt=="6")
    {
       break;

    }
    else
    {
       cout<<"                    Invalid Choice. Please Try Again..."<<endl;
       cout<<"                     "<<endl;
       system("pause");
       continue;
    }
    }



}
    void Addrecord(string empname[],int empid[],int maxrow,string num)
    {
        setTextColor(225);
       int counter=0;
```

```
string EmpName={};
int EmpID={};
while(true)
{
gotoxy(35,3);
cout<<"Enter Employee ID: ";
getline(cin,num);
if(ID_validation(num))
  {
   EmpID=stoi(num);
   break;
   }
   else
{
   gotoxy(30,4);
   cout<<" Invalid Input! Please Enter ID which consists of 5 digits...";
   getch();
   gotoxy(35,3);
   cout<<"                                                    ";
   gotoxy(30,4);
   cout<<"                                                    ";

}
}
fstream file;
file.open("data.txt",ios::out);
for(int idx=0;idx<5;idx++)
{
```

```
file << EmpID;
file << "\n";
}
file.close();
while(true)
{
gotoxy(35,6);
cout << "Employee Name: ";
getline(cin,EmpName);
if(Name_Validations(EmpName))
{
   break;
}
else
{
   gotoxy(35,7);
   cout << "Invalid Name Format!" << endl;
   getch();
   gotoxy(35,6);
   cout<<"                                                ";
   gotoxy(35,7);
   cout<<"                                                  ";
}
}
bool idExists = false;
for (int i = 0; i < maxrow; i++)
{
   if (empid[i] == EmpID)
```

```
        {
            idExists = true;
            break;
        }
    }

    if (!idExists)
    {
        for (int i = 0; i < maxrow; i++)
        {
            if (empid[i]==0)
            {
                empid[i] = EmpID;
                empname[i] = EmpName;
                gotoxy(35,8);
                cout << "New Employee Account is created successfully!" << endl;
                break;
            }
        }
    }
    else
    {
        gotoxy(35,8);
        cout << "Employee ID already exists. Please enter a different ID." << endl;
    }

   gotoxy(30,10);
    system("pause");
```

```
}

   void UpdateRecord(int maxrow,int &empID,int empid[],string empname[])
   {
       setTextColor(225);
   string EmpName={};
   int EmpID=0;
   int counter = 0;

   for (int x = 0; x < maxrow; x++)
   {
      if (empid[x] == empID)
      {
          counter++;
           while(true)
   {
    gotoxy(35,25);
   cout << "Employee Name: ";
   getline(cin,EmpName);
   if(Name_Validations(EmpName))
   {
      break;
   }
   else
   {
      gotoxy(35,26);
      cout << "Invalid Name Format!" << endl;
      getch();
```

```
        gotoxy(35,25);
        cout<<"                                                                    ";
        gotoxy(35,26);
        cout<<"                                                                    ";
    }
    }
            empname[x] = EmpName;

            cout << "Update Successful!" << endl;
            // Display the updated employee information
            cout << "Employee ID: " << empid[x] << ", Updated Name: " << empname[x] << endl;

            // Optionally, break if you want to exit the loop after updating
            break;
        }
    }

    if (counter == 0)
    {
        cout << "Employee ID not found!" << endl;
    }

    system("pause");
}

    void DeleteRecord(int &empID,int maxrow,int empid[],string empname[])
    {
         setTextColor(225);
```

```
        int counter = 0;

    for (int x = 0;x < maxrow; x++)
    {
        if (empid[x] == empID)
        {
            counter++;
            empname[x] = {};
            empid[x] = {0};

            cout << "Successfully Deleted!" << endl;

            // Optionally, break if you want to exit the loop after deleting
            break;
        }
    }

    if (counter == 0)
     {
        cout << "Employee ID not found!" << endl;
    }

    system("pause");
}

    void searchrecord(int &empID,int maxrow,int empid[],string empname[])
    {
        system("cls");
```

```
 setTextColor(230);
gotoxy(35,2);
cout<<" ------------------------------------------------------------"<<endl;
gotoxy(35,3);
cout<<"                    Current Record(s)                    "<<endl;
gotoxy(35,4);
cout<<" ------------------------------------------------------------"<<endl;
cout<<"                                                         "<<endl;
 setTextColor(51);
gotoxy(25,6);
cout<<"NO.";
gotoxy(45,6);
cout<<"|";
gotoxy(65,6);
cout<<"Employee ID";
gotoxy(85,6);
cout<<"|";
gotoxy(105,6);
cout<<"Employee Name"<<endl;
 setTextColor(225);
int counter=0;
int startx=25;
int starty=8;
int x=startx;
int y=starty;
for(int i=0;i<maxrow;i++)
{
   if(empid[i]==empID)
```

```
        {
         counter++;
        gotoxy(x,y);
        cout<<counter;
        x=x+20;
        gotoxy(x,y);
        cout<<"|";
        x=x+20;
        gotoxy(x,y);
        cout<<empid[i];
        x=x+20;
        gotoxy(x,y);
        cout<<"|";
        x=x+20;
        gotoxy(x,y);
        cout<<empname[i]<<endl;
        y=y+1;
        x=startx;
    }



    }

    if(counter==0)
    {
        gotoxy(x+20,y);
        cout<<"------------------------------"<<endl;
        gotoxy(x+20,y+1);
```

```
        cout<<"       No Record found!        "<<endl;
        gotoxy(x+20,y+2);
        cout<<"------------------------------ "<<endl;

    }
     system("pause");
  }
  void listrecord(int maxrow,int empid[],string empname[])
 {
     setTextColor(230);
    system("cls");
    gotoxy(35,2);
    cout<<" ---------------------------------------------------------------"<<endl;
    gotoxy(35,3);
    cout<<"                      Current Record(s)                    "<<endl;
    gotoxy(35,4);
    cout<<" ---------------------------------------------------------------"<<endl;
    cout<<"                                                          "<<endl;
    gotoxy(25,6);
     setTextColor(51);
    cout<<"NO.";
    gotoxy(45,6);
    cout<<"|";
    gotoxy(65,6);
    cout<<"Employee ID";
    gotoxy(85,6);
    cout<<"|";
    gotoxy(105,6);
```

```
cout<<"Employee Name"<<endl;
 setTextColor(225);
int counter=0;
int startx=25;
int starty=8;
int x=startx;
int y=starty;
 for (int i = 0; i < maxrow; i++)
 {
if (empid[i]!=0)
 {
   counter++;

   gotoxy(x, y);
   cout << counter;
   x += 20;
   gotoxy(x, y);
   cout << "|";
   x += 20;
   gotoxy(x, y);
   cout << empid[i];
   x += 20;
   gotoxy(x, y);
   cout << "|";
   x += 20;
   gotoxy(x, y);
   cout << empname[i] << endl;
   y++;
```

```
            x = startx;
        }
    }


    if (counter == 0)
     {
        gotoxy(x + 20, y);
        cout << "------------------------------" << endl;
        gotoxy(x + 20, y + 1);
        cout << "       No Record found!       " << endl;
        gotoxy(x + 20, y + 2);
        cout << "------------------------------ " << endl;
    }
    system("pause");
    }
    void usermanagementtitle()
    {
         setTextColor(51);
    cout<<"                                    <<<<      User Management   >>>>
"<<endl;
    cout<<"                                                                    "<<endl;
    }
    string usermanagementmenu(int xA,int yA)
    {
         setTextColor(225);
    string option;
    string array[6]={"Create Records", "Update Records", "Delete Records", "Search
Records","Display all Records","Exit"};
        {
```

```
        for(int idx=0;idx<6;idx++)
        {
            gotoxy(xA,yA);
            cout<<idx+1<<". "<<array[idx];
            yA=yA+1;
        }
    }
    gotoxy(xA,yA);
    cout<<"Choose Option(1-6): ";
    getline(cin,option);
    return option;
}
```