



Uniform Store System

Software Engineering 2- CSE 222

Submitted To:

DR. Gamal Abd El-Shafy

Submitted By:

Youssef Ossama Eid 16P8178

Ahmed Bahaa Ibrahim 16P6057

Basma Magdy Mohamed 16P8187

Amr Hossam El-Deen 16P3076

Mohamed Hesham Hussain 15P3090

Topics Covered

1. Functional & Non Functional Requirements
2. Data Flow Diagram
3. Object Diagram
4. Context Diagram
5. Semantic Diagram
6. Sequence Diagram
7. State and Stimulus Diagram
8. Architecture Design
9. Module View Analysis & View Points
10. Design Decision
11. Architecture Style
12. Component Model
13. Cost Estimate
14. Process Model
15. User Guide

1.Functional & Non Functional Requirements

Functional requirements

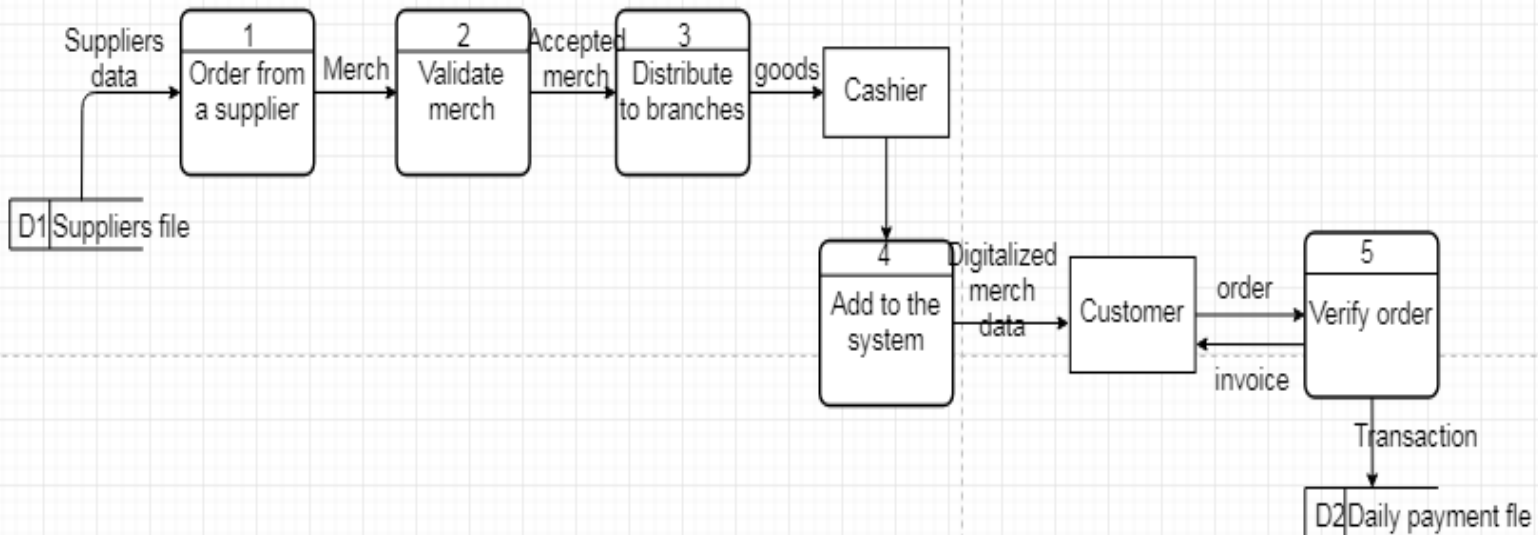
- Cashier should be able to browse store catalogue
- Select an item
- Add item to ecart
- Add new product
- Delete order
- Print receipt
- Calculate final price
- See daily profit
- Check item's price
- Choose payment method
- Store and search customer's phone number
- Return order
- Send order to supplier
- Calculate price after discounts

Non-Functional requirements

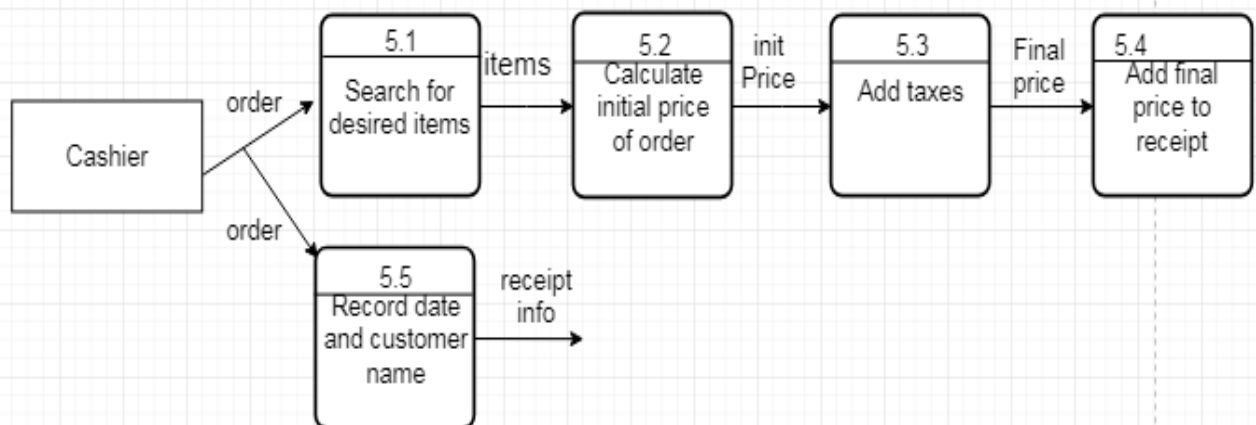
- Programmed in Java
- Made in 2 months
- Price does not exceed 5000L.E
- Memory size does not exceed 3 GBs
- conform to all applicable local laws
conform to the company standard STD0945
- Should have a fast response time and tolerate common types of faults

2.Data Flow Diagram

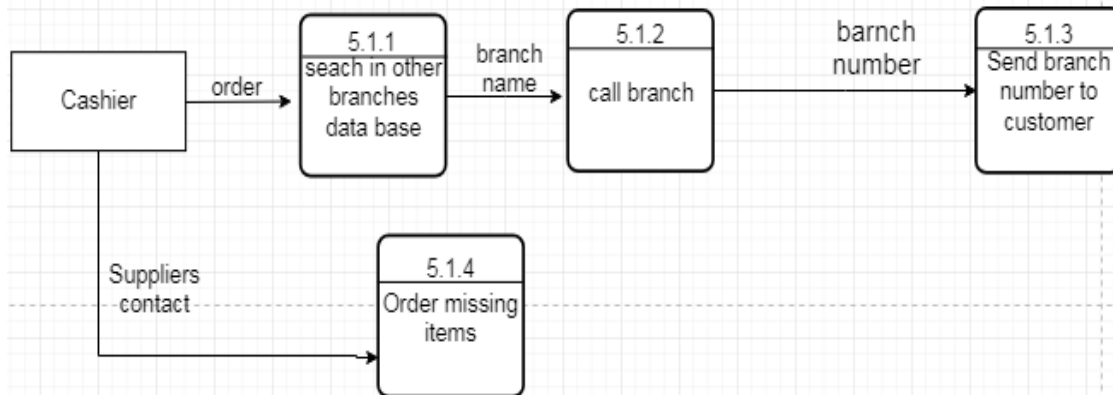
DFD Level 0



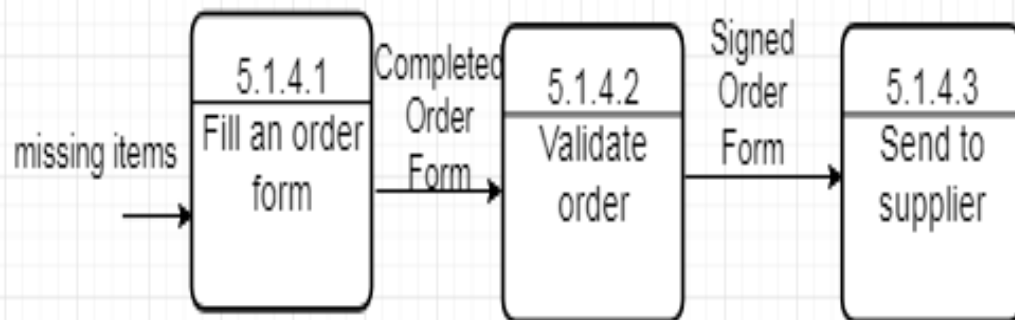
DFD Level 1 On the major process of Verifying an order



DFD Level 2 on the major process of searching for desired items in failure condition

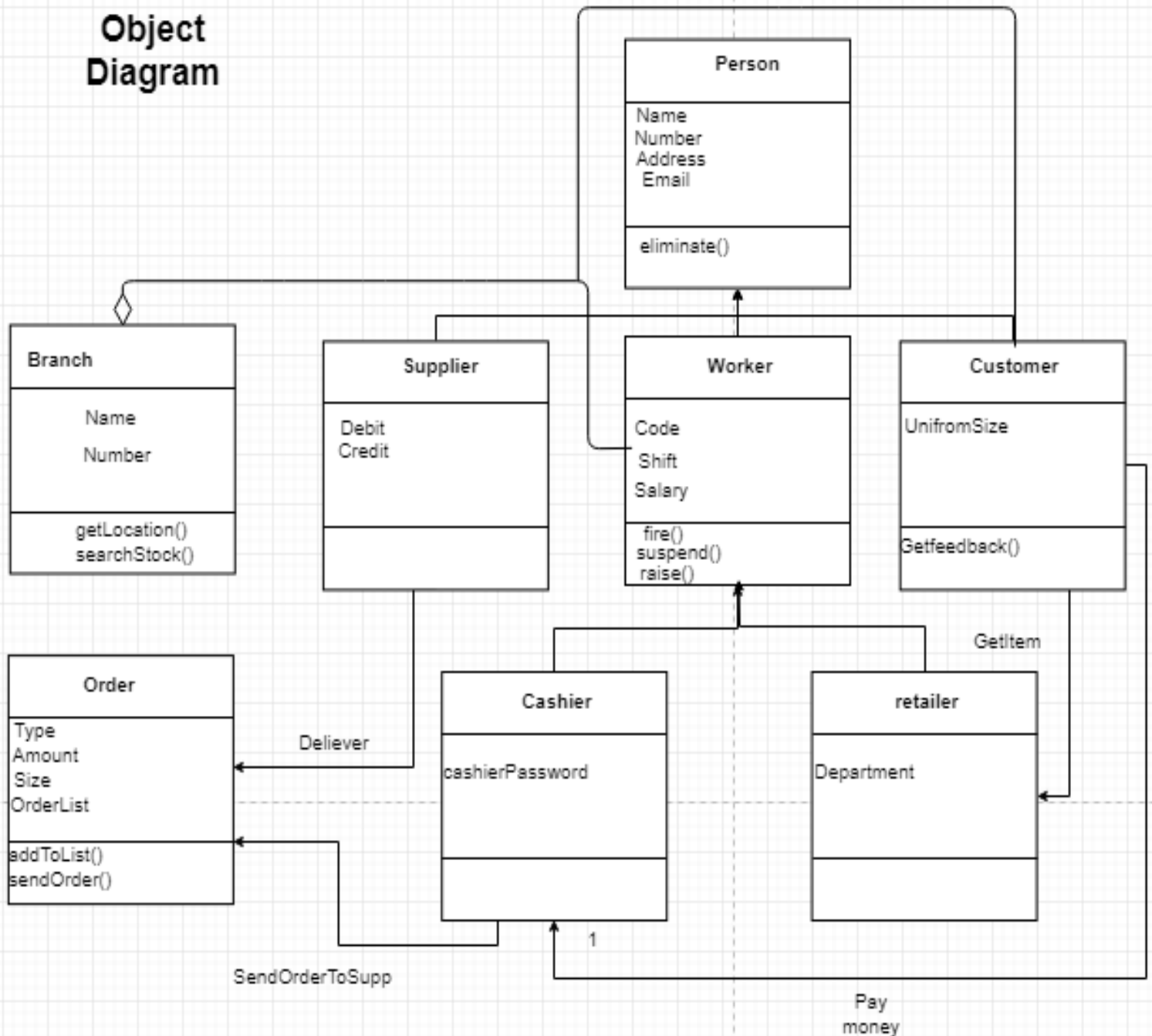


DFD level 3 on the major process of Ordering missing items

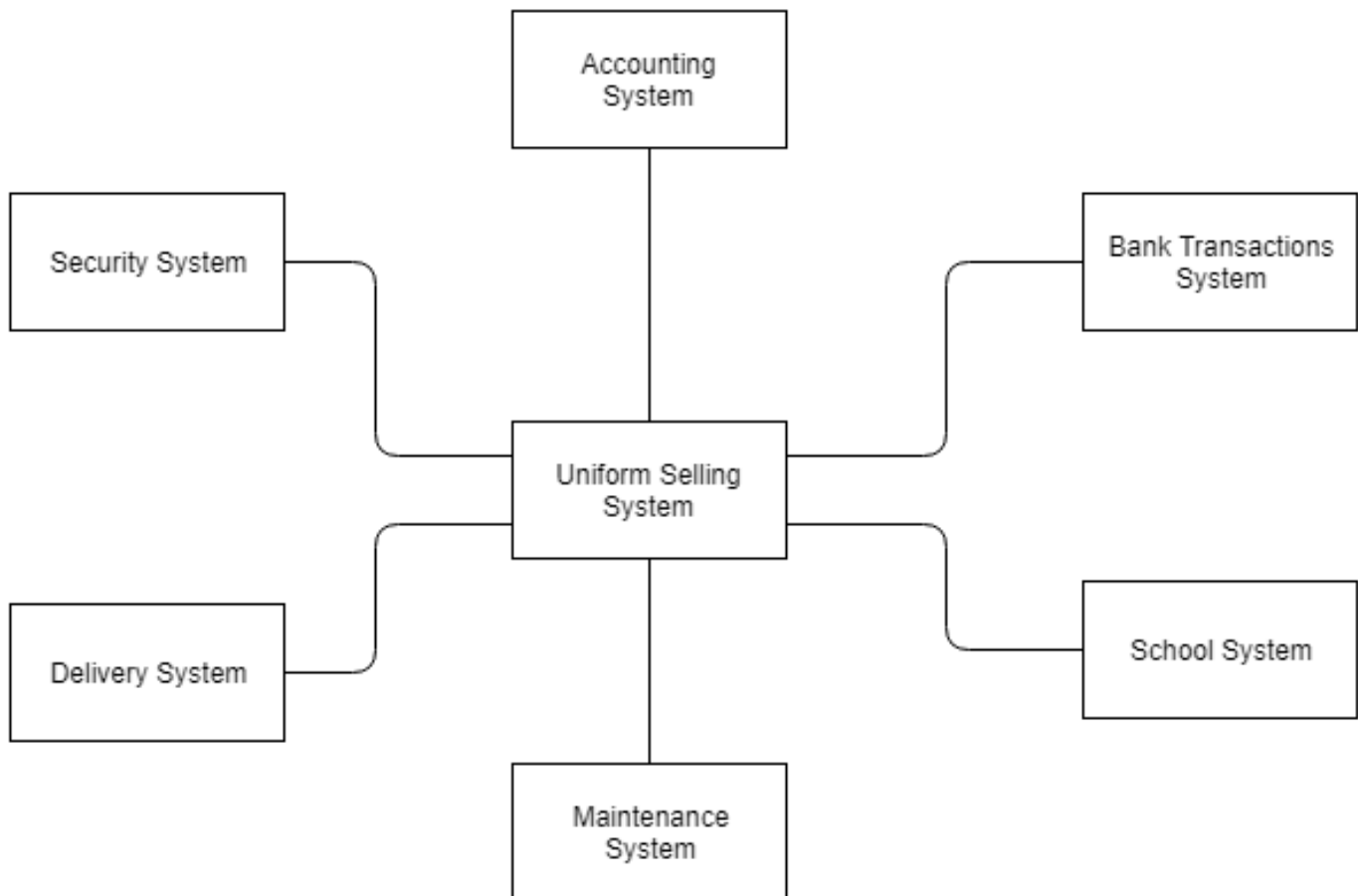


3.Object Diagram

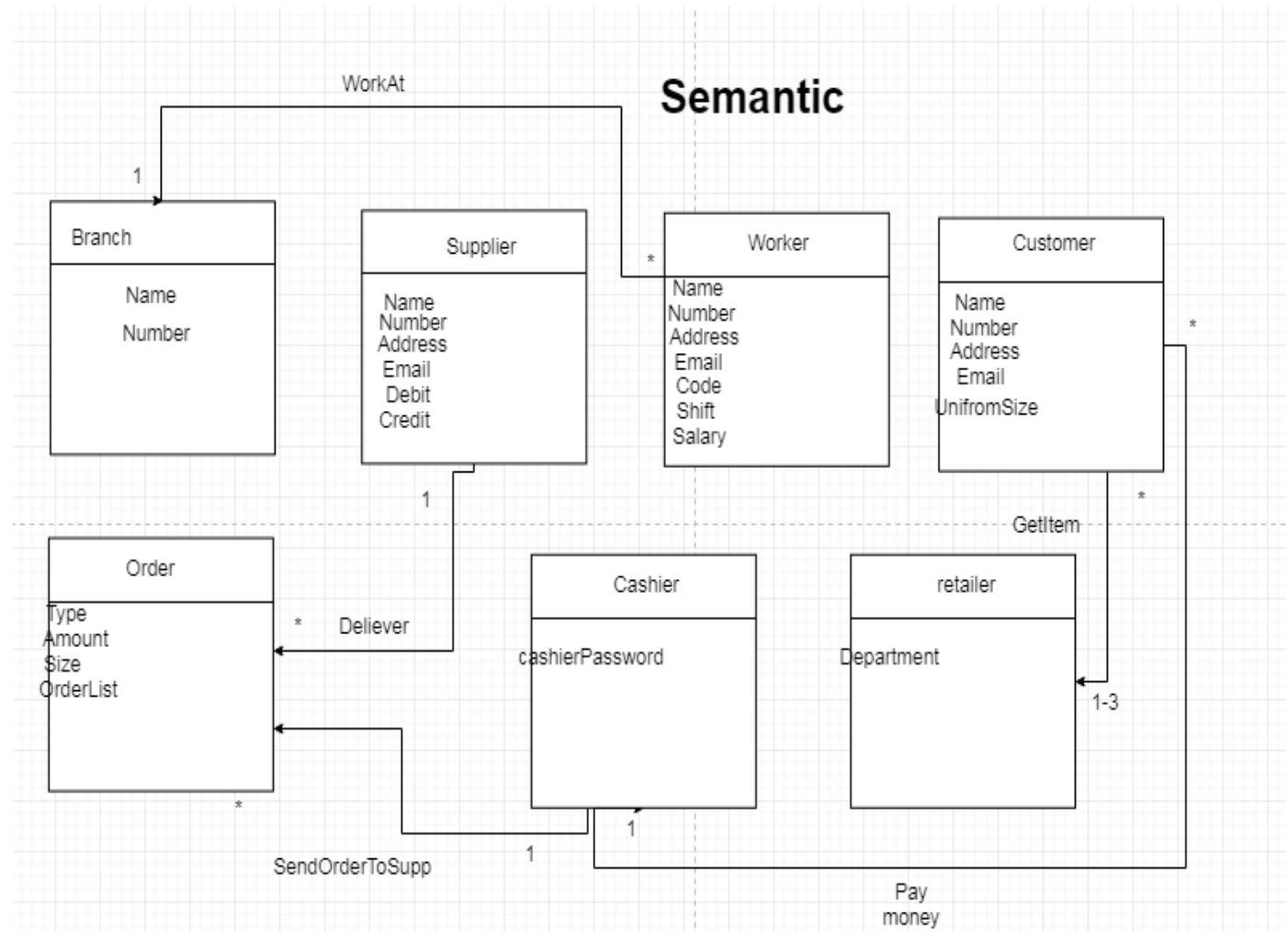
Object Diagram



4.Context Diagram



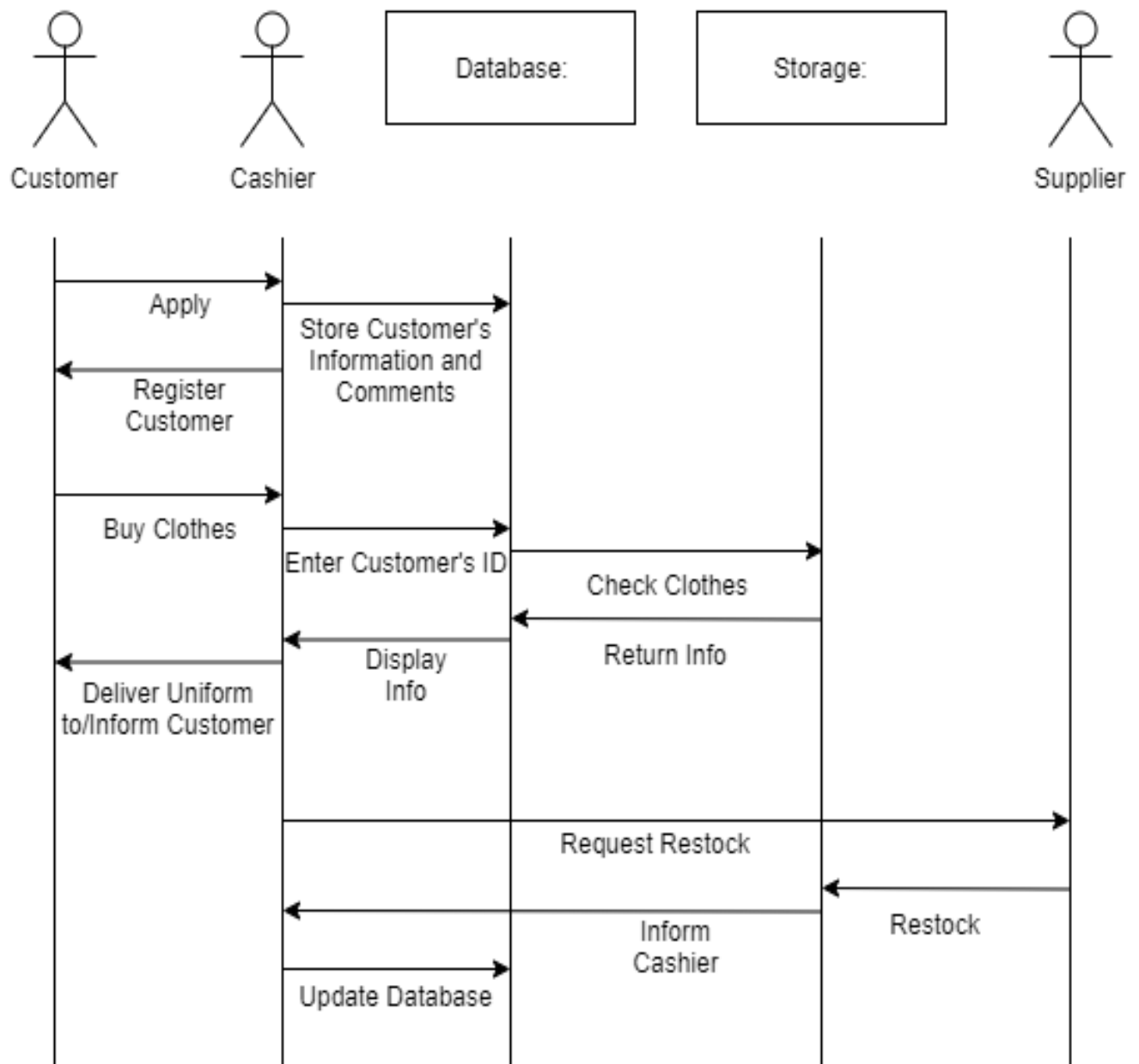
5.Semantic Diagram



Data Entries:

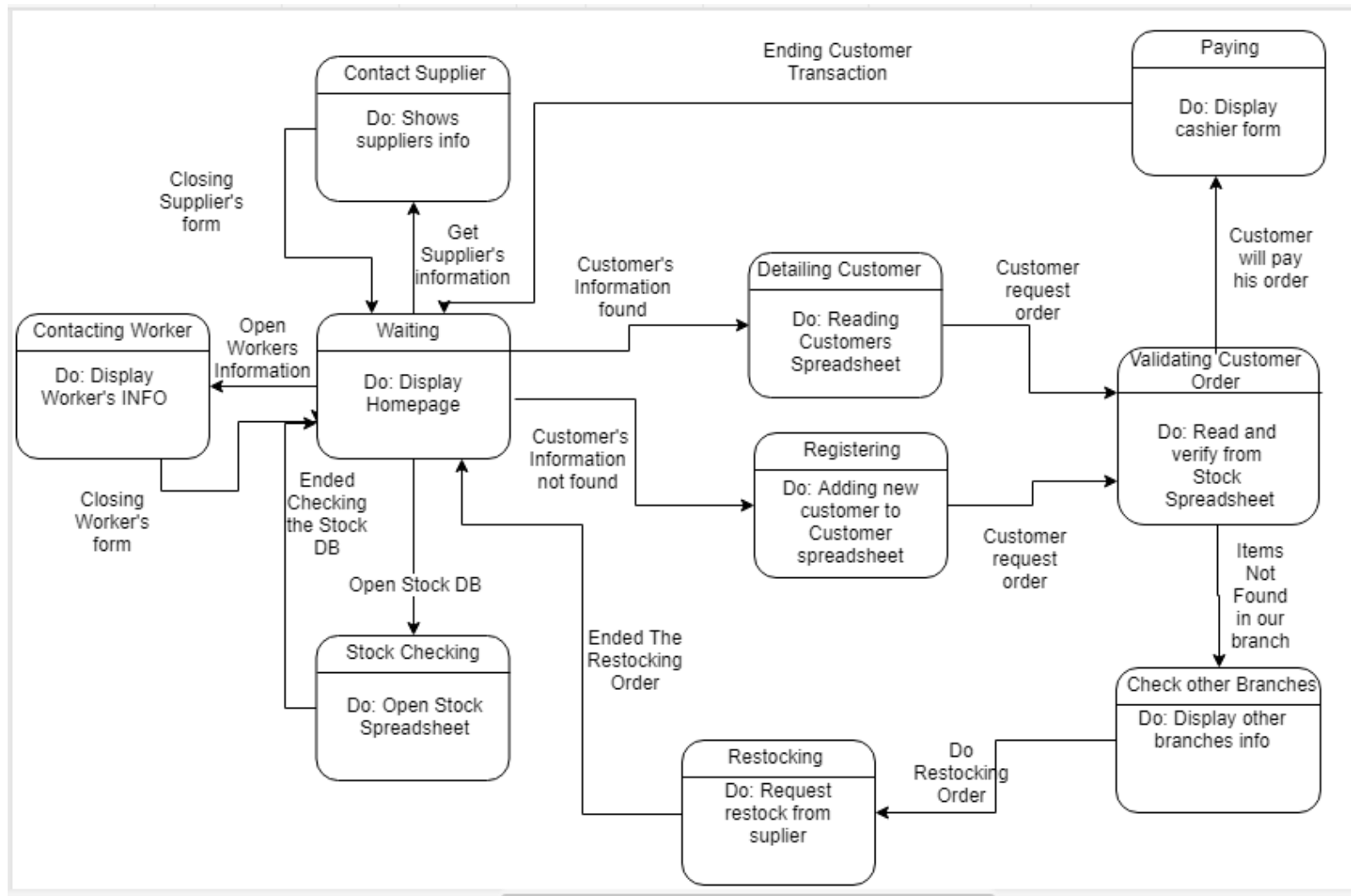
Name	Description	Type	Date
Customer	The details of customer that will be used in future orders.	Entity	30.12.2002
<u>UniformSize</u>	The size of the uniform cloth that was requested by the customer	Attribute	30.12.2002
Pay Money	A one-to-many relation between the cashier(s) and customer who will pay the cashier for the uniform	Relation	30.12.2002
Shift	The time of the worker where he should attend the workplace	Attribute	29.10.2002
Worker	A details of the worker that will be working in a branch	Entity	29.10.2002
WorkAt	A Many-to-One relation between the worker(s) and branch	Relation	29.10.2002

6. Sequence Diagram



7.State and Stimuli Diagram

Uniform System Model



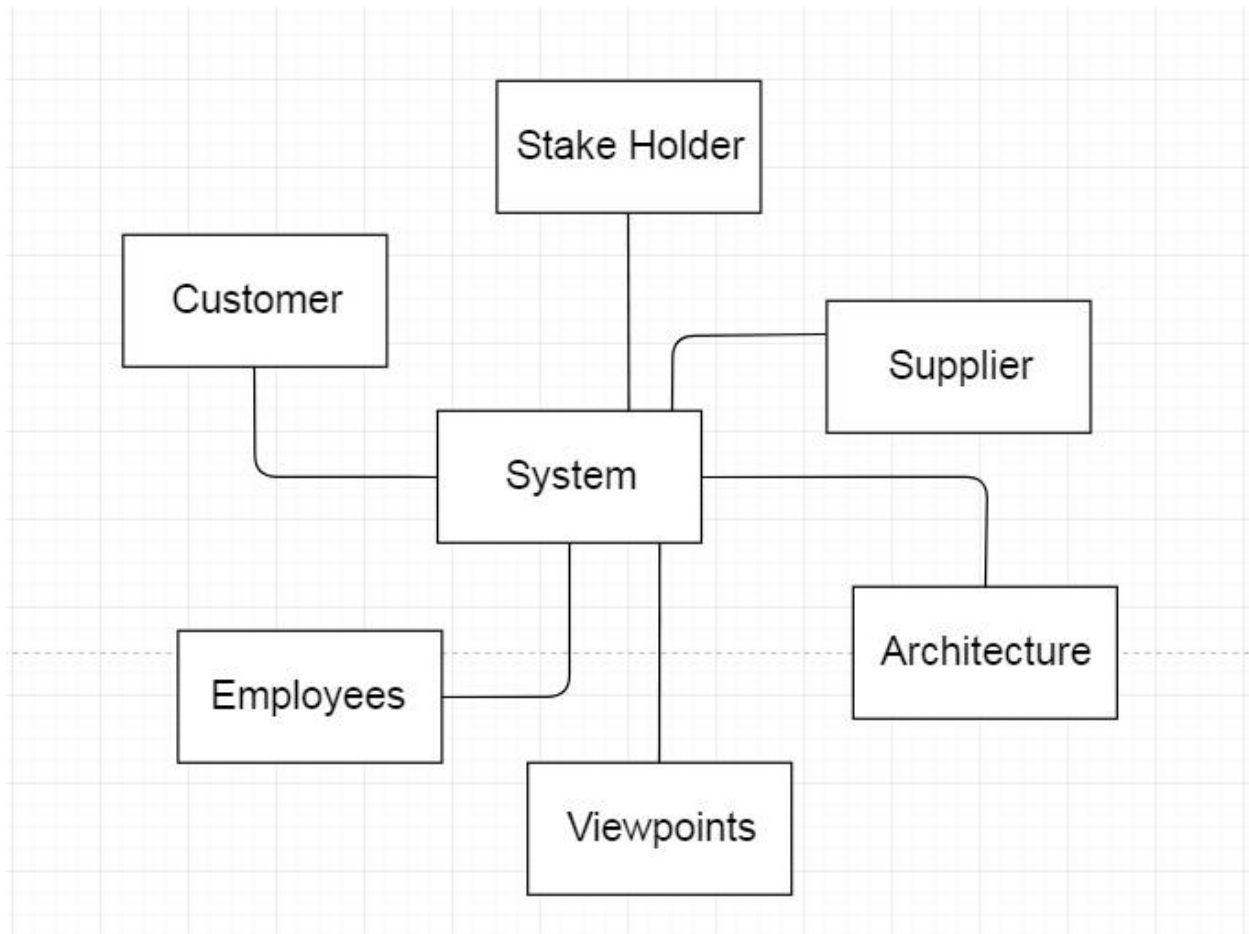
State Description

State	Description
Registering	Open Register window and add the customer's Name, No. of Children, and Comments.
Paying	Display Cashier's window to add customer's order
Stock Checking	Display The Stock State of the branch
Detailing Customer	Viewing Customer' detailed information
Check other Branches	Check other branches with the available stock
Contact Supplier	Show all information about our suppliers
Restocking	Order unavailable items from the supplier
Waiting	Display All The Controllers that the Admin can use
Contact Worker	Show all details about the Worker
Validating Customer Order	Checking order items from stock

Stimuli Description

Stimulus	Description
Registering	Cashier registers first entry Customer.
Paying	Customer approaches Cashier with an order.
Stock Checking	The Cashier wants to check the available stock
Detailing Customer	Existing Customer in our System arrives to the cashier
Check other Branches	Items required are not found in our branch.
Contact Supplier	Cashier want to check some Supplier's Data
Inform Customer.	Item found in other branches.
Restocking	Customer Order Item not found in our branch.
Waiting	Starting home page and when any procedure ends
Contact Worker	Admin want to check specific worker's Information
Validating Customer Order	Customer approach with an order to the cashier

8. Architecture Design

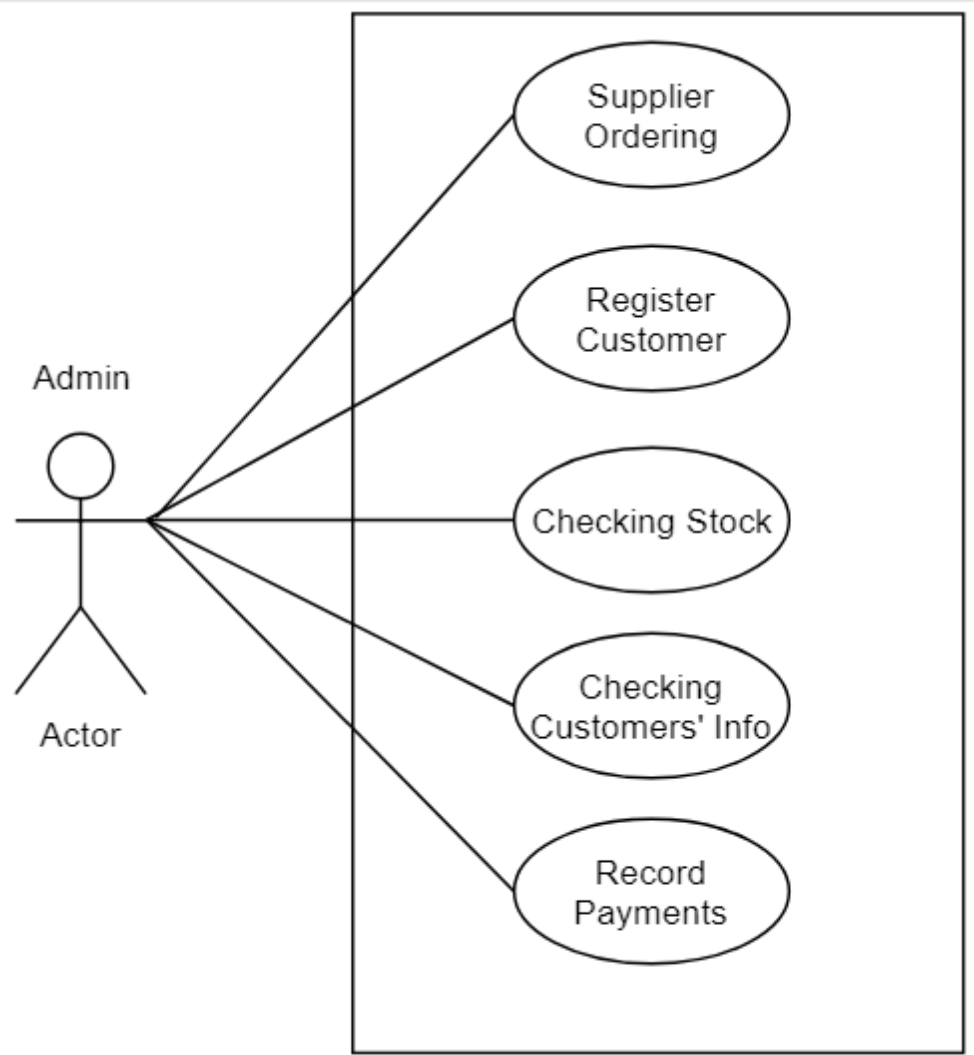


9.Module View Analysis and View Points

Scenario Viewpoint

1. The Customer arrives our shop to buy the School Uniform for his children.
2. The Customer is checked if he is registered to our system with all required information such as; mobile number, sizes of his children and their Academic years, this information help us to maintain our stock and guess the estimated amount needed of each size in each peace.
3. After that, The Customer goes to the retailer and tells him his order, the retailer sends the order to the cashier to check if the order is valid with the clothes in the stock or there is something missing.
4. The Cashier opens the Stock Database and checks if the stock fits the order or not.
5. IF the order is valid the retailer will go into the stock and brings the uniform to the customer and then the Cashier asks the Customer if he would like to pay with Visa or Cash, and then the Customer goes with the receipt to the retailer to give away the order.

6. IF the item wasn't found at all, the cashier checks the other branches' stocks Database and inform the customer with the nearest branch containing the items he needs. After That The Cashier then will open the Order Items form, he will enter the order he requires with the amounts that he needs and when he clicks "Send" Button there is an automated mail sent to the Supplier that The shop work with to send this order to that branch as soon as possible.
7. The Previous case happens in the case of items that are unique like trousers of size 3XL in Primary Stage ,these sizes are extreme so we don't make order until it ends from our stock.
8. In Normal Cases, when number of an item in our stock decreases to 5 , there is a notification sent to the cashier to warn him to order these items before they are finished from the Stock.



Implementation Viewpoint

- We used JAVA in implementation ,this is because it is so easy to work with GUI, as our main goal was to build a simple User Interface for the workers in the Shop ,and also the Developers team ,all of them are professional in JAVA Language and this make the System more efficient ,also we can handle the excel files of the stock ,and the Customers with our code and this will be easy with JAVA.
- Our Project's package(sw) consist of six GUI forms and one main class which are:
 - Branches.java
 - Cashier.java
 - Stock_orders.java
 - Suppliers.java
 - Home.java
 - Workers_info.java
 - MainClass.java

Logical Viewpoint

- ❖ Our System contains six different GUI forms and two excel sheets to support all functional requirements needed for the system.
- ❖ Home Form: It is the main Form that contains all the options of the system like a button to send order to supplier other button to open the cashier form.
- ❖ Cashier Form: It is The Form in which the cashier enter the detailed order of the customer and there is an auto calculation done after entering the items to calculate the price of the bill and then print the bill to the customer.
- ❖ Stock Orders Form: It is the Form in which we enter the list of items that are missing or decreasing in our stock, and enter the amount that we need and send it to the Supplier to supply it to our branch as soon as possible.
- ❖ Branches Form: It shows the general information of all other branches like their addresses, phone numbers , in case that the customer want to get any communication with them.
- ❖ Workers Info Form: It shows general Information about our workers like their names, salaries, phone number, address , and the work-shift time.

- ❖ Supplier Records Form: It contains all data we need about our supplier like his name , email address , his debit and credit balances as there is an option that paying after ordering stock is not a must.
- ❖ Stock Excel Sheet: It contains the detailed stock of our branch and all other branches , with the amount of items in each branch and the sizes , this is so helpful also when items are missing from our branch, we can search for this item in other branches' stock easily.
- ❖ Customers Info Excel Sheet: It contains all the information about our customers like their phone numbers, sizes of their children , and their Academic years as this also help us to make charts and graphs to know which items are on demand so we can request them from the supplier more frequently.

Process Viewpoint

- ❖ The Developers team take into their consideration to make The System with perfect runtime behavior, and also to make it with a great performance, System Availability, Concurrency and distribution, System Integrity and fault-tolerance.
- ❖ The System has very good connection between the code and the Data in the excel sheets, as it is very fast in both reading from and writing to the sheets, and this helped in increasing the Performance, The Runtime Behavior.
- ❖ The Auto Calculation Process that was integrated inside the cashier form helps the cashier to decrease the error percentage to 0% in calculation processes, and it will also save a lot of time and effort.
- ❖ The Email Auto Sending when the cashier presses the “Send” button in the Supplier Orders Form ,this also helped so much instead of the cashier opens the email and write a subject , and this all details of writing an E mail so this process helped a lot in system Integrity and reduce the risk to send the email to any wrong email address and improve the Style of the Email itself.

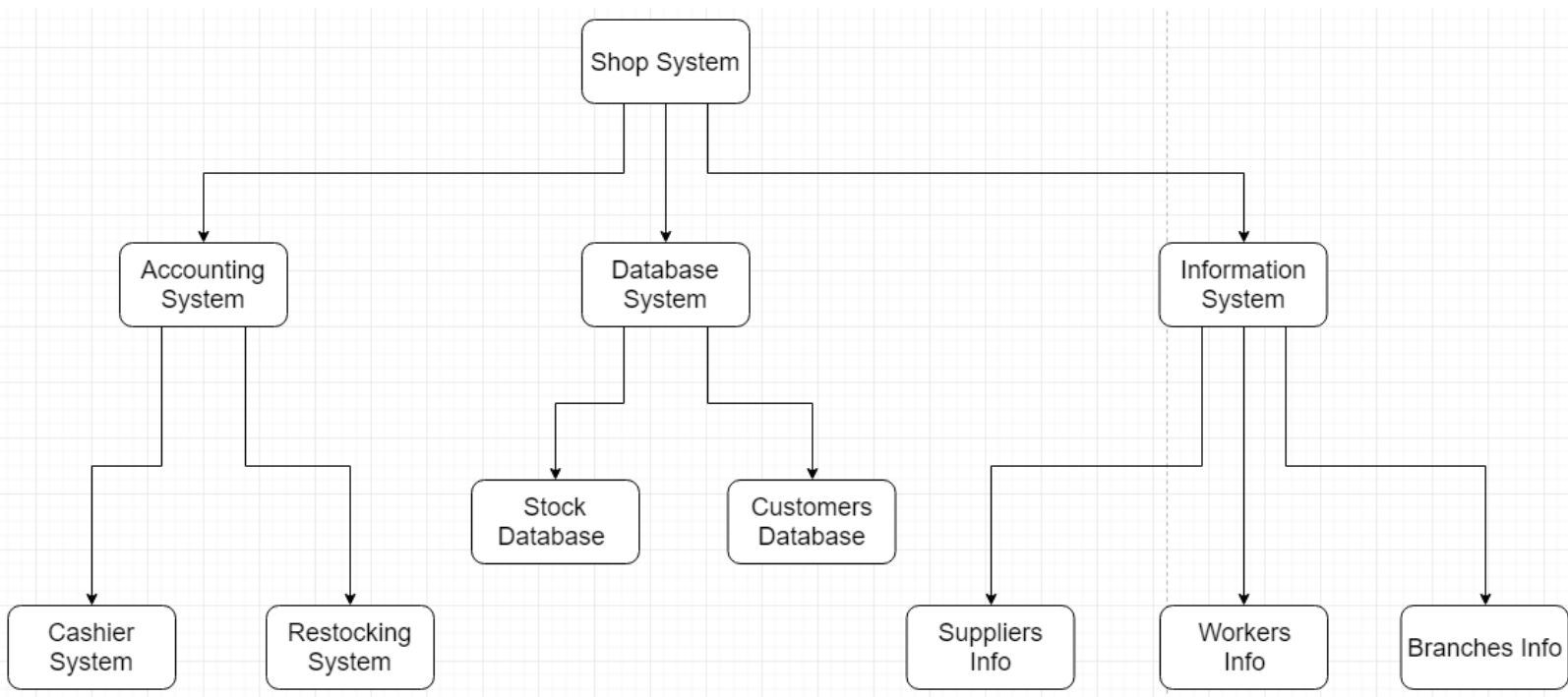
Deployment Viewpoint

- ❖ The Developers Team also were keen on improving the System's availability, reliability, performance, and Scalability.
- ❖ The Main Goal was to make Simple GUI forms so the workers can't find any difficulty in working with the System, so we used simple forms with simple buttons and combo boxes so the worker will not need to write anything , he will only choose and select the items from the list , and this made the System more reliable and simple for the workers.
- ❖ We put onto our consideration that the Manager of the shop would like to save money at buying the hardware so we made our system doesn't need high performance computers and takes small part of the RAM , so we saved a lot of money for the manager by this solution.
- ❖ We provide free maintenance for our system ,at any time any crash happened or a problem was detected in the System , there is a quick maintenance done to the system.
- ❖ We provide also training for the Workers to use our System , so they become more confident when working with.

Module View

Decomposition:

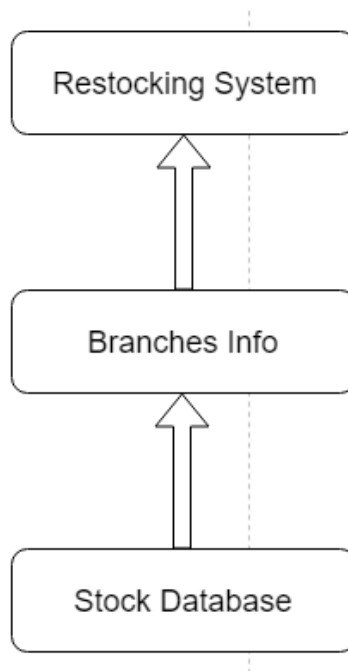
- Database System: is a submodule of the Shop System
- Accounting System: is a submodule of the Shop System
- Information System: is a submodule of the Shop System
- Stock Database: is a submodule of the Database System
- Customers Database: is a submodule of the Database System
- Cashier System: is a submodule of the Accounting System
- Restocking System: is a submodule of the Accounting System
- Suppliers Info: is a submodule of the Information System
- Workers Info: is a submodule of the Information System
- Branches Info: is a submodule of the Information System



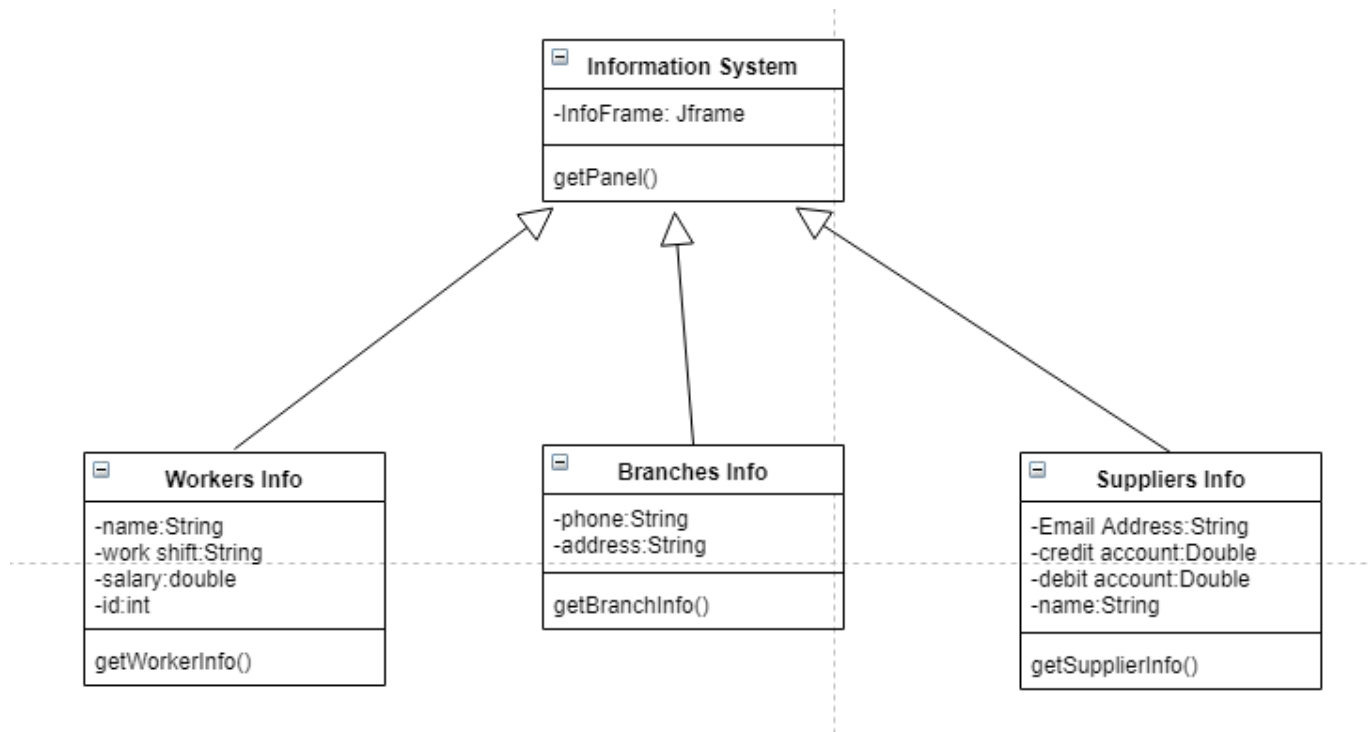
Uses:

- Cashier System use the Stock Database to check the price of each item.
- Restocking System use the Stock Database to check the remaining amount of each item.
- Cashier System use the Customer Database to print his info on the bill and to update his account.
- Branches Info use the Stock Database to know the detailed stock of each branch.

Layers:



Classes:



10.Design Decision

Issues: The main issue is finding a suitable supplier to deal with from extreme factory cloth line and B&B line.

Decision: The decision is to go with extreme factory

Status: The decision is approved from the CEO

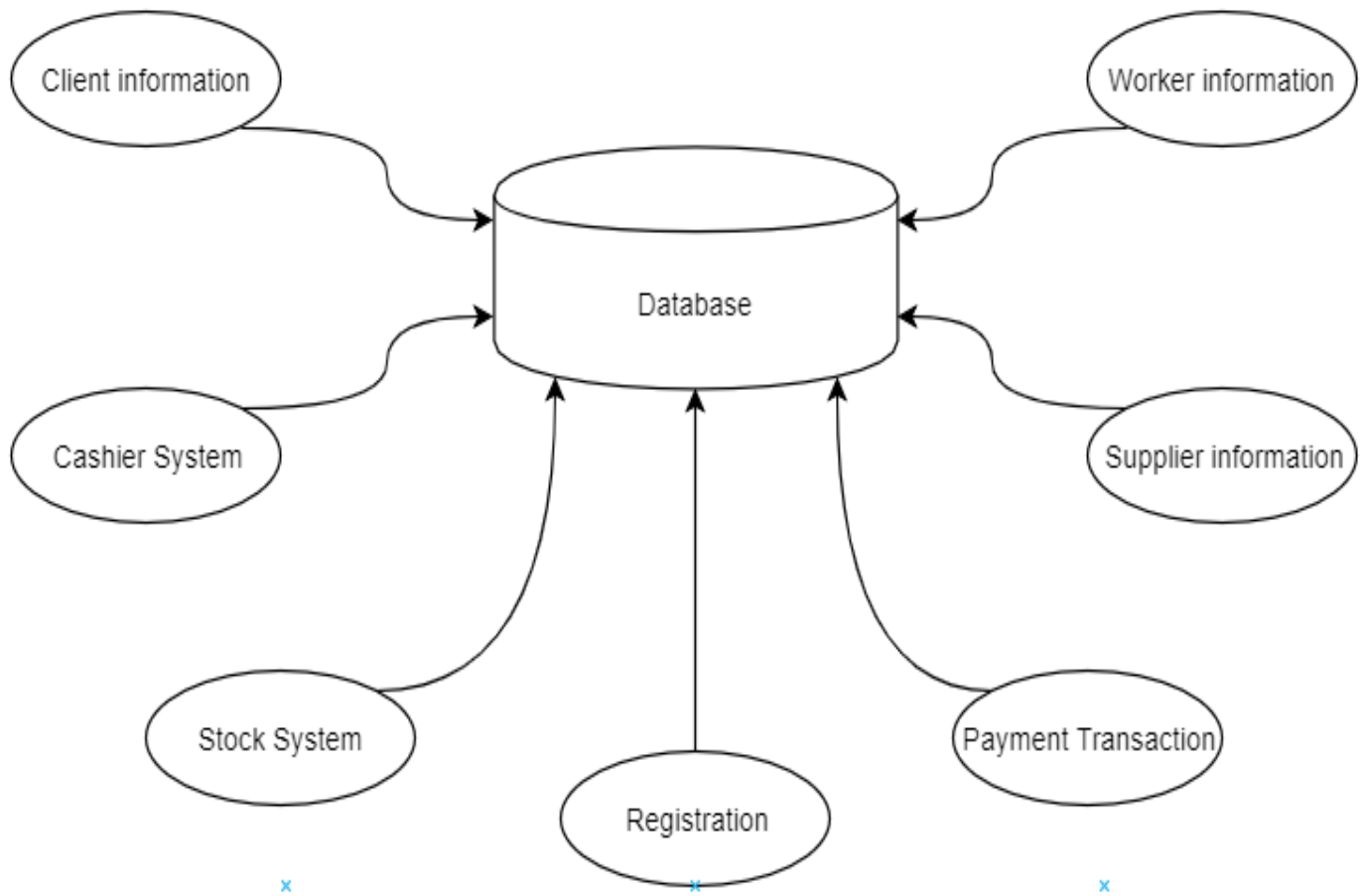
Assumptions: We assumed that that extreme factory would supply the order on an earlier date than B&B

Alternatives: B&B factory

Rationale: We chose extreme since their price is cheaper, their quality is higher and their they make discounts on bulk purchases

Implications: Our contract with Extreme factory will end after 3 years so we need to find another suitable supplier.

11. Architecture Style



Problem:

Program had to manage multiple information while sharing it to multiple clients, data should be lasting for a long time.

Context:

Data used in a branch had to be shared with multiple other clients.

Solution:

System Model: Information and data are centralized with each store branch using its own stock.

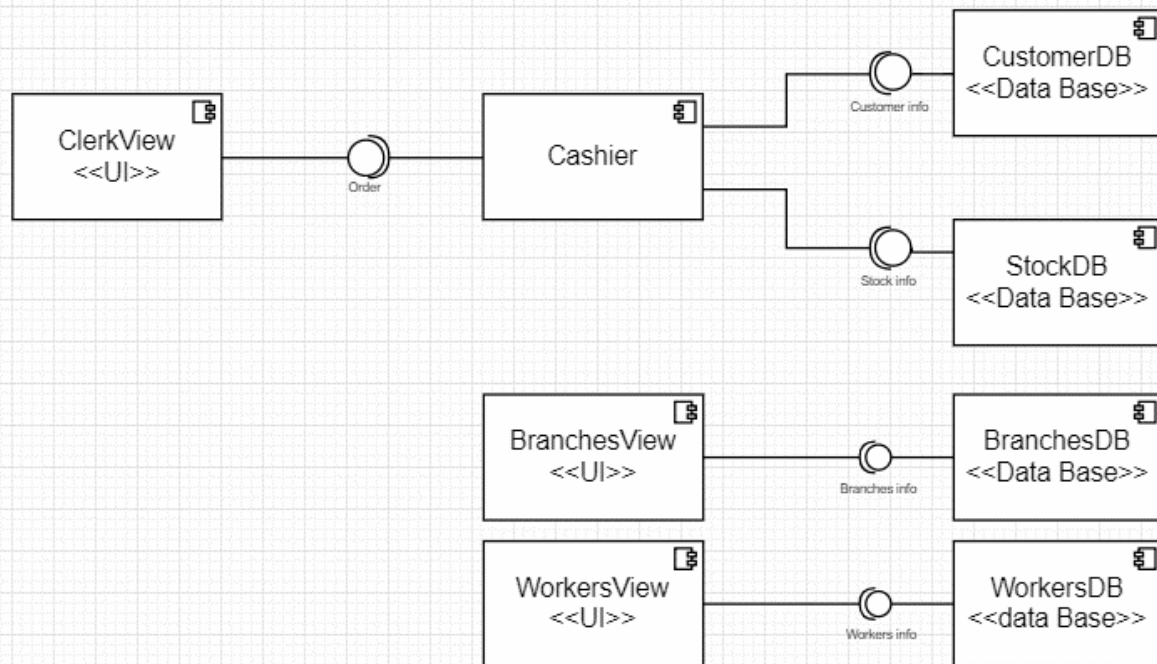
Component: Local processing and data sharing.

Connectors: Internet data streams.

Control Structure: Cashier's GUI controlled program.

It is a typical database system.

12.Component Model



Components

Definition

A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third-parties.

Components

1. ClerkView: a component that deals with User/clerk takes order info from him and passes it to Cashier component
2. Cashier: the component responsible for processing the order data and making sure it conforms with the stock (there is enough stock to cover order) and takes customer Information from the customer data base
3. CustomerDB: responsible for storing customer data and this info
4. StockDB: responsible for storing stock data and provides this data
5. BranchesView: interacts with the clerks shows branches info
6. BranchesDB: stores the branches info and provides them
7. WokersView: interacts with the clerks shows Workers info
8. WorkersDB: stores the Workers info and provides them

Component Process

The system was Developed by various independent ways and processes of development of components and composition each component is independent and reusable

Component Characteristics

1. ClerkView:

- Deployable: component can operate on its own
- Standardized: this component conforms to interface model with defined component interfaces and component meta-data.

2. Cashier:

- Independent: A component should be independent – it should be possible to compose and deploy it without having to use other specific components. In situations where the component needs externally provided services.
- Standardized: the component conforms to standardised component model. This model defines component interfaces required data is set in requires interface.
- Documented: component is well documented for future buyers and all semantics are specified.

3. CustomerDB

- Independent: can be deployed independently without having user specific components.
- Deployable: the component is self contained and is able to operate as a standalone entity.
- Documented: the component is fully documented and does not have to be compiled before deployment.

4. StockDB:

- Independent: can be deployed independently without having user specific components.
- Deployable: the component is self contained and is able to operate as a standalone entity.

5. BranchesDB

- Independent: can be deployed independently without having user specific components.
- Documented: the component is fully documented and does not have to be compiled before deployment.

Component Reusability

All components are reusable and are

1. No application specific methods
2. General names
3. Methods were added to broaden coverage
4. Consistent Exception handling added to all components

13. Cost Estimate

	A	B	C	D	E	F	G
1	Info Domain	Optimistic	Likely	Pessimistic	Est Count	Weight	FP Count
2	# of inputs	4	6	8	6	4	24
3	# of outputs	9	12	13	11	5	55
4	# of inquiries	14	16	18	16	4	64
5	# of files	1	2	3	2	10	20
6	# of external interfaces	1	2	3	3	7	21
7	UFC: Unadjusted Function Count:						184
8			Complexity Adjustment Factor:				1.07
9						FP:	197

	A	B	C	D	E	F	G	H
1	Complexity Factor: Fi	Value=0	Value=1	Value=2	Value=3	Value=4	Value=5	Fi
2	Backup and recovery	0	1	0	0	0	0	1
3	Data Communication	0	0	0	0	0	1	5
4	Distributed processing function	0	0	0	0	1	0	4
5	Is performance critical?	0	0	1	0	0	0	2
6	Existing operating environment	0	0	1	0	0	0	2
7	On-line data entry	0	0	0	0	0	1	5
8	Input transaction built over multiple screen	0	0	0	1	0	0	3
9	Master files updated on-line	0	1	0	0	0	0	1
10	Complexity of inputs, outputs, files, inquiries	1	0	0	0	0	0	0
11	Complexity of processing	0	0	0	0	0	1	5
12	Code design for re-use	0	0	1	0	0	0	2
13	Are conversion\installation included in design?	0	0	0	0	1	0	4
14	Multiple installation	0	0	0	1	0	0	3
15	Application designed to facilitate change by user	0	0	0	0	0	1	5
16							Sigma(F):	42
17		Complexity Adjustment Factor: $0.65 + 0.01 * 42 =$						1.07

Average LOC/FP in java =53

LOC =53 * 197 =10441

Assuming

Estimated project LOC = 10441

Organisational Productivity (similar project type) = 350 LOC/p-m

Burdened Labor Rate = 6000 \$/p-m

Then

Effort = $10441/350 = 30$ p-m

Cost per LOC = $6000/350 = 9$ \$

Project total cost = $6000 * 30 = 180000$ \$

Assuming

Estimated FP = 197

Organisational Productivity (similar project type) = 5.3 FP/p-m

Burdened Labor Rate = 6000 \$

Then

Estimated Effort = $197/5.3 = 37$ p-m

Cost per FP = $6000/5.3 = 1132$ \$/FP

Project Cost = $6000 \times 37 = 222000$ \$

Expert Judgement:

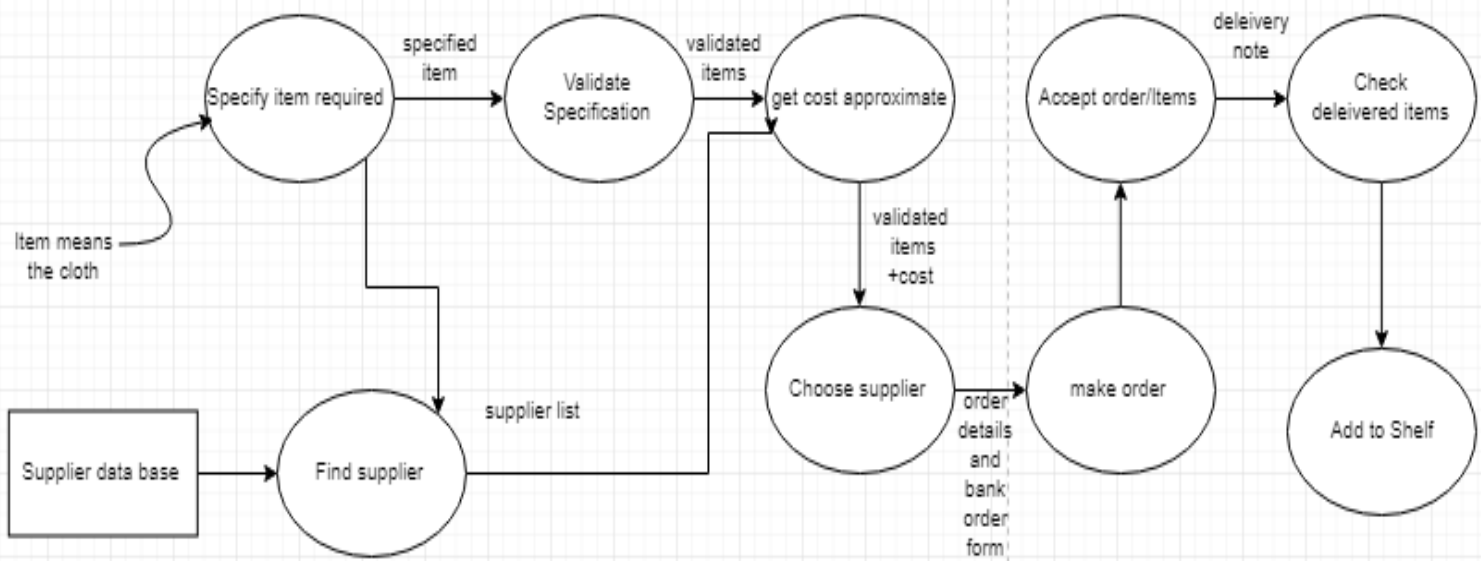
This cost estimation has been shown to a software expert , and the costs and numbers were approved by him.

Estimation by analogy:

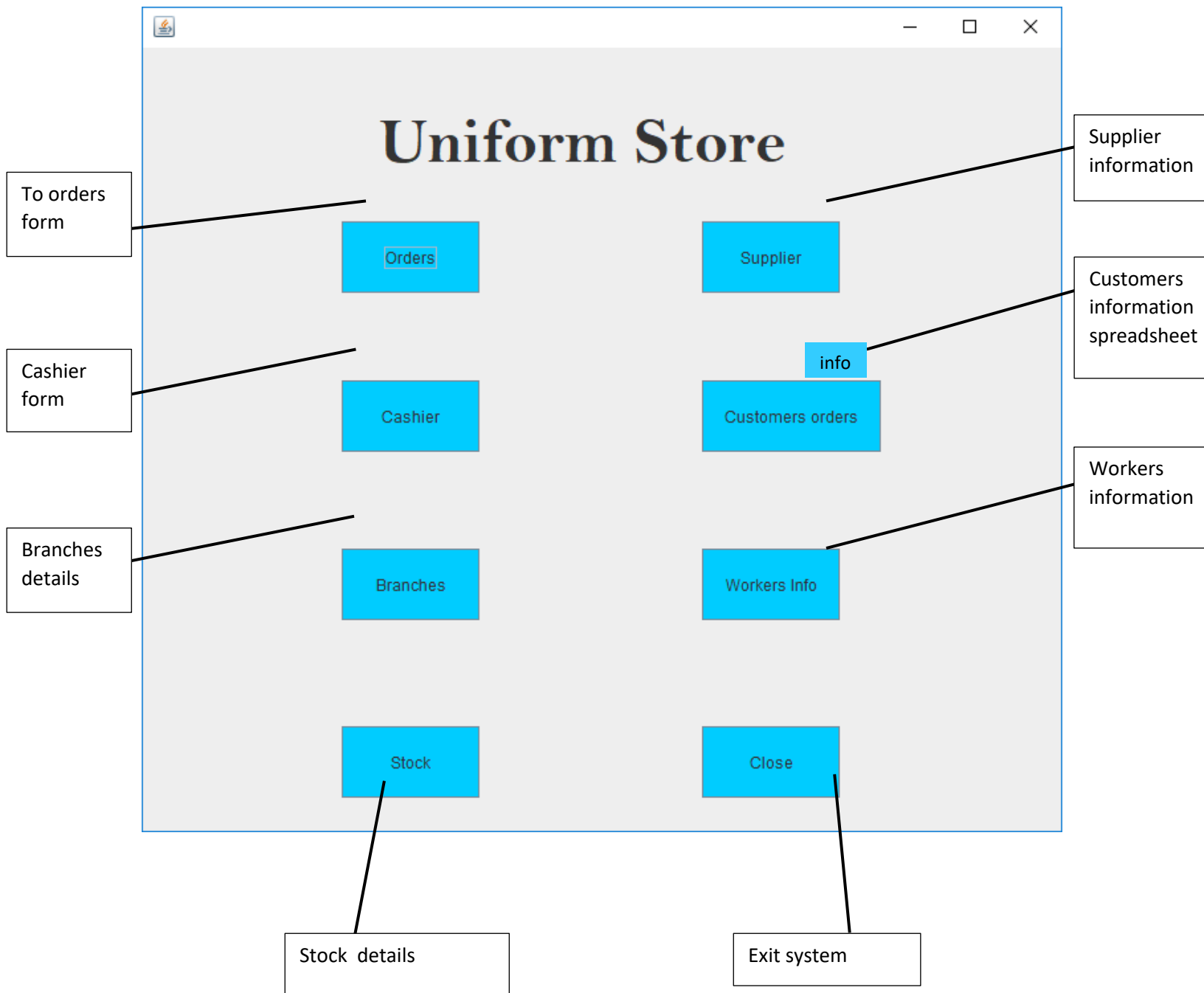
The cost estimation was compared with other cost estimations for projects of the same type , and all numbers were close.

14.Process Model

Item procurement process



15.User Manual



The image shows a software window titled "Workers Info". At the top left is a small icon, and at the top right are standard window controls (minimize, maximize, close). Below the title bar is a large header area with the text "Workers Info" in a bold, serif font. Underneath the header is a table with five columns: "Name", "Code", "Number", "Email", and "Shift". The table has four empty rows. Below the table is a large, empty rectangular area. In the bottom right corner of the window is a blue button with the text "Exit".

Name	Code	Number	Email	Shift

Exit

Workers' database

Exit form

The image shows a software window titled "Branches" with a standard Windows-style title bar (minimize, maximize, close buttons). The window contains a list of three branches, each with a corresponding number in a text box. An "Exit" button is located at the bottom right of the list. Three callout boxes with leader lines point to specific elements: "Branches' address" points to the third branch name, "Branches' number" points to the number of the second branch, and "Exit form" points to the "Exit" button.

	Number
1. NasrCity Branch	24442422
2. NewCairo Branch	24444522
3. Heliopolis Branch	24982922

Exit

Branches' address

Branches' number

Exit form

Choose item type

Choose size

Choose quantity

Total price

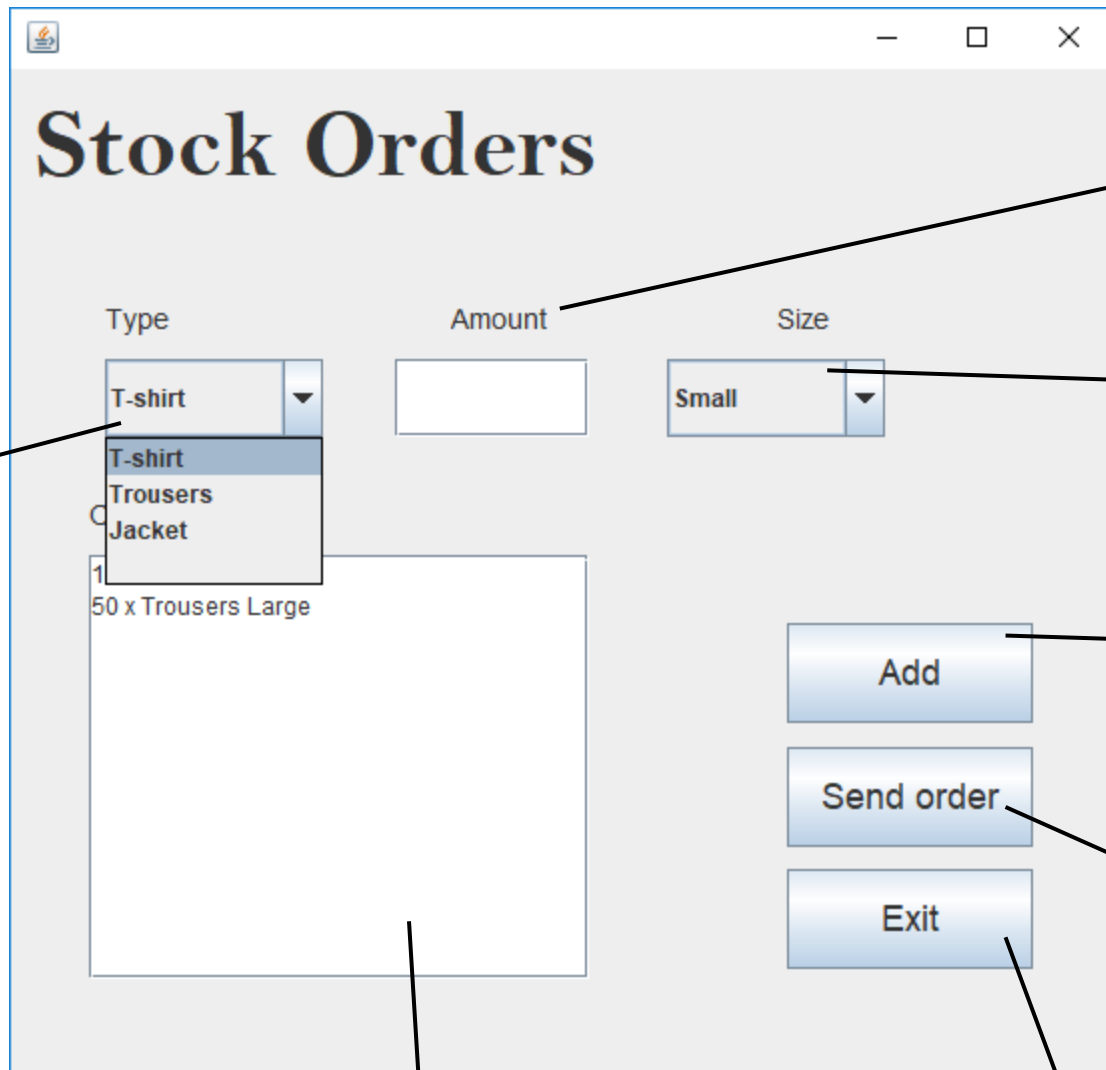
The screenshot shows a window titled "Cashier" with a standard Windows-style title bar (minimize, maximize, close buttons). Inside the window, there are four labels: "item:", "Size:", "Quantity:", and "Price:". Below "item:" is a dropdown menu showing "T-shirts". Below "Size:" is a dropdown menu showing "Small". Below "Quantity:" is a text input field containing the number "1". To the right of these fields, the text "Price: 100 LE" is displayed. Below the input fields is a rectangular box labeled "Invoice:" containing the text "1x Tshirt Medium". To the right of the invoice box are three stacked buttons: "Add", "Print", and "Exit".

Items chosen appears with their size and quantities

Add the chosen item in the invoice

Print invoice

Exit the form



The image shows a graphical user interface for a 'Stock Orders' application. The window has a title bar with standard minimize, maximize, and close buttons. The main area has a light gray background. At the top, the title 'Stock Orders' is displayed in a large, bold, black serif font. Below the title, there are three input fields: 'Type', 'Amount', and 'Size'. The 'Type' field is a dropdown menu with 'T-shirt' selected, and a list of options (T-shirt, Trousers, Jacket) is visible. The 'Amount' field is a text box. The 'Size' field is a dropdown menu with 'Small' selected. Below these fields is a large white rectangular area, likely for a list of items. To the right of this area are three blue buttons with white text: 'Add', 'Send order', and 'Exit'. Annotations with arrows point to various elements: 'Choose type of items for stock order' points to the 'Type' dropdown; 'Choose the amount of items for stock order' points to the 'Amount' text box; 'Choose size of items for stock orders' points to the 'Size' dropdown; 'Add the chosen item to stock order space' points to the 'Add' button; 'Press to send order' points to the 'Send order' button; 'Exit the form' points to the 'Exit' button; and 'Total stock order chosen' points to the large white area below the input fields.

Stock Orders

Type: T-shirt
Amount:
Size: Small

1
50 x Trousers Large

Add
Send order
Exit

Choose the amount of items for stock order

Choose size of items for stock orders

Choose type of items for stock order

Add the chosen item to stock order space

Press to send order

Exit the form

Total stock order chosen

Supplier Records

Name:

Abdel Rehim Works

Number:

01278528035

Address:

24 Madinet el nasr

Email:

abdelrehim99@gmail.com

Debit	Credit

Supplier's corporation name

Supplier's phone number

Supplier's corporation

Supplier's email address

Amounts unpaid to the supplier

Amounts outstanding on the supplier

	A	B	C	D
1	item \ branch	Madinet Nasr	New Cario	Heliopelis
2	T-shirts S	200	150	90
3	T-shirts M	134	161	155
4	T-shirts L	42	11	31
5	Trousers S	40	165	24
6	Trousers M	123	180	176
7	Trousers L	34	122	231
8	Jackets S	134	24	87
9	Jackets M	21	9	200
10	Jackets L	177	114	15

Items types
with their
sizes

Amount of each
item sizes that
exist In Madinet
Nasr Branch

Amount of each
item sizes that
exist In New Cairo
Branch

Amount of each
item sizes that
exist In Heliopolis
Branch

1	Name	Number	Address
2	Basma Magdy	128374655	Heliopolis
3	Youssef Ossama	122998788	Nasrcity
4	Amr Hossam	122998788	Shorouk
5	Ahmed Bahaa	122998788	New cairo
6	Mohamed Hisham	122998788	Maadi
7	Andrew Sameh	122998788	Nasrcity
8	Khalid Mohamed	122998788	6th October
9	Youssef Hammad	122998788	New cairo
10	Omar Makino	122998788	Rehab
11	Mohamed Abdel-Hamid	122998788	Heliopolis

Customers' full
names

Customers' phone
numbers

Customers' home
addresses