SIMULATED E-COMMERCE PORTAL

Version 1.0

ARUN PRAKASH

ROSHNA RAMESH

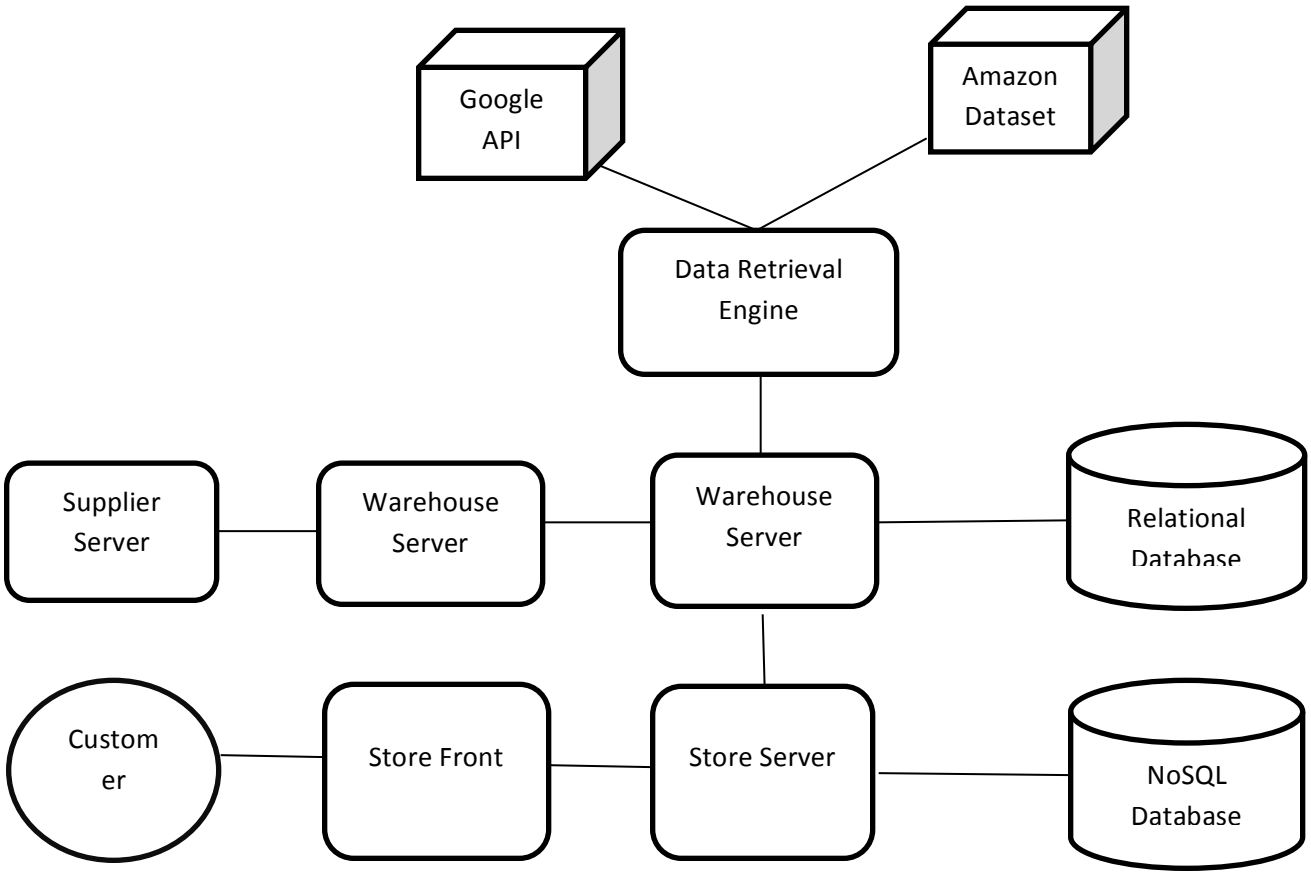
Contents:

1. PROJECT DESCRIPTION:

The Inventory Management System is developed to provide the customers their desired product by identifying the nearest retailer who has that product in stock. It also maintains that the warehouse supplies the products to the retail stores on demand and the supplier ships the requested products to the warehouse. The software can be used to maintain as large as a multinational retail corporation. It uses the Google Maps API to identify the locality of the retail stores and warehouses. The retail store managers, warehouse administrators and the suppliers can log into the system and keep track of the products and stock. The data is maintained in large databases.

The customer will not have any login id. They will use the console to specify the products details. The system does not incorporate the credit card processing services. It also does not allow the users of the system to view and edit their profiles. It does not provide any billing services.

2. SYSTEM ARCHITECTURE AND DESIGN:



2.1 Store Front:

The Store Front acts as the interface between the customer and the distributed system. The customer is not granted permission to access any other part of the system other than the Store Front. The main function of the store front is to obtain the user input and send the input to the store server. It also interfaces with the Store Server to access MongoDB database to retrieve the product information and display it to the customer. The Query Processing Server accepts the customer's queries, processes it by making remote method calls to the respective servers and communicates the message back to the customer.

2.2 Store Server:

The Store Server provides two important functions – processing requests from the Customer and contacting the nearest Warehouse for receiving stock. The Store Server is responsible for receiving the purchase request and displaying the nearest stores which have the product available. If the stock of a particular product is below a threshold level, the store server places a request to the nearest warehouse in its range to buy stock.

The store server processes the user's request for a product and returns the appropriate result. If the user searches for a product which is not present in the Store Catalogue, the Store server returns an error message reporting an invalid search. Based on the availability of stock in the each Retail Store, the store server returns the results. The store server also queries information from the external API to locate the nearest Retail Stores to the user.

2.3 Warehouse Server:

The Warehouse Server is responsible for handling requests from retail stores and purchasing stock from suppliers. There exists one warehouse for numerous retail stores. The Warehouse server receives request from a store server for purchase of stock and sends a message back to it based on the availability. If the warehouse runs out of stock, the warehouse server makes a remote call to the supplier server for purchase of stock; so as to prevent the organization from running into loses.

The Warehouse server receives requests from different Retail Stores for the purchase of stock and updates its current stock based on the stocks being sent to the Stores. Once the supplier server approves the request, it sends a response to the warehouse server updating the stock count in the corresponding servers.

2.4 Supplier Server:

The Supplier Server provides the functionality of interacting with various warehouses and performing transactions for the purchase of stocks. The supplier server interacts only with the Warehouse server and there exists no contact between a Retail Store and a supplier. The Supplier Server handles requests from warehouses for the purchase of stocks. The Warehouse server makes a remote call to the supplier server and places a request. Based on the supplier's availability of stock that is retrieved from the database, the supplier server sends a reply approving or rejecting the purchase request from the warehouse. The supplier server frequently renews the stock count on the database such that there is no situation where the supplier cannot provide a warehouse with the requested stocks. Every request is approved by a supplier order and an acknowledgement.

2.5 Query Processing Server:

The Query processing server provides 2 important functionalities: It accepts queries from all the components of the system through the remote method calls and process those queries to retrieve data from both the relational and NoSQL databases. If proper records are found for the query in the database, it passes them on to the corresponding components through remote method calls. If the query fails, (i.e) the data doesn't exist in the database, it should be retrieved from the external APIs by using the Data Extraction Engine. The QPS should be capable of handling multiple hits from the clients. All the records retrieved are passed back to the corresponding components through the remote method calls.

2.6 MySQL:

MySQL is used to maintain a relational database and it holds all the details that will have to strictly follow the ACID properties. The data from the Google maps API, the retail store, warehouse and customer data are placed in the MySQL Database. We also have products table stored which is populated with the data obtained from the Amazon datasets.

2.7 NoSQL:

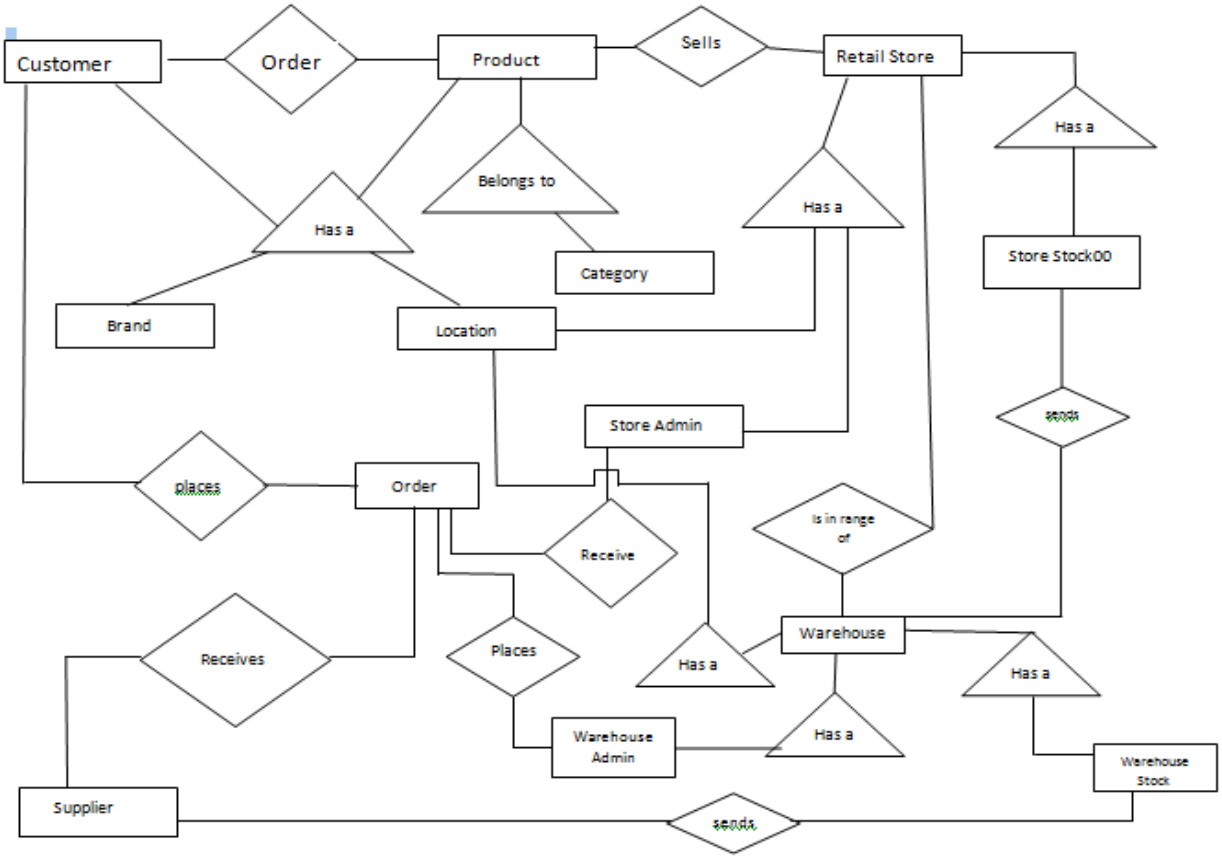
The main advantage of using NoSQL databases is the added functionalities they possess over Relational databases. They are built to allow insertion of data without a pre-defined schema.

Also, NoSQL databases allow auto sharing of data across multiple servers on the cloud. In the inventory management system, the NoSQL database in which the hit counter details are stored is used to process the queries sent by the store server. The NoSQL database used is MongoDB

3. DATABASE DESIGN

Table Name	Primary Key	Attr1	Attr2	Attr3	Attr4	Attr5	Attr6	Attr7
Customer	Customer_ID	Customer_Name	Location ID					
Product	Product_ID	Product_Name	Brand_ID	Quantity	Price	Category_ID		
Brand	Brand_ID	Brand_Name						
Retail Store	StoreID	Store	Location ID	Warehouse ID				
Retail StoreStock	Store_ID	Product_ID	Stock_Left					
POrder	OrderID	Customer Id	StoreID	ProductID	Qty	Price	Order Date	
Store Admin	Employee_ID	Employee_Name	Store_ID	Username	Pass-word			
Warehouse Admin	Employee_ID	Employee_Name	Warehouse_ID	Username	Pass-word			
Warehouse	Warehouse ID	Warehouse Name						
WareHouseStock	Warehouse_ID	Product_ID	Stock_Left					

OrderToWarehouse	Order_ID	Warehouse_ID	Store_ID	Product_ID	Qty	Price	SupplyDate	Status
OrderToSupplier	Order_ID	Warehouse_ID	Product_ID	Qty	Price	OrderDate	Shipment ID	Status
Supplier	SupplierID	Supplier Name	Username	Password				
Location	LocationID	Street_ Address	City	State	Zip_Code	Lat	Lng	
Category	CategoryID	Category Name						
SiteAdmin	SiteAdmin ID	Username	Password					
Shipment	Shipment ID	ShipmentType						
HitTable	HitDate	NumberOf Hits						



4. FUNCTIONAL REQUIREMENTS

Functional requirements define the fundamental actions that are performed by the system.

4.1 Place_Order

The user has to give details about the product to place his order.

Inputs

- Product Name
- Quantity

Processing

The function will match the customer's requirement with the database and retrieve the list of products available which is given as output to the user.

Output

A list of all products that match the customer specifications.

4.2 Find_Location

Input

The customer provides the zip code of his locality.

Processing

The system uses Google Maps API which searches its database to match the customer's zip code with that of all the retail stores present within a particular radius.

Output

A list of all retail stores nearest to the customer.

4.3 Check_availability

Input

- Retail stores
- Product ID
- Quantity

Processing

The system looks into the database of all retail stores,which was obtained as output in the previous function,to check the availability of the product using the product's ID and the quantity specified.If it is not present in one store,it will move on to the other store.

Output

The retail store's location where the product is available will be provided to the customer.

4.4 Update_stock

Input

- Product id
- Quantity
- Store Location

Processing:

The system updates the quantity of the product in the database after the purchase made by the customer.

Output:

Acknowledgement after the update in the database.

4.5 Check_Warehouse_Stock

Input:

Request from the retailer when a trigger from the database is received

Processing:

Checks with the warehouse database for the quantity of the stock available.

Output:

A boolean value depending upon the stock availability

4.6 Retailer_request_product

Input

- Product ID
- Quantity
- Warehouse ID

Processing:

Request is placed to the warehouse by the retail store manager with the above input details

Output:

An acknowledgement from the warehouse manager saying that the order has been placed

4.7 Warehouse_request_product

Input

- Product ID
- Quantity
- Brand
- Supplier ID

Processing:

Request is placed to the supplier by the warehouse manager with the above input details.

Output:

An acknowledgement from the supplier saying that the order has been placed.

5. NON-FUNCTIONAL REQUIREMENTS

5.1 Reliability

The system should ensure reliability. It will be tested to ensure performance and an absence of bugs in the system. Any failure in the system should be resolved promptly. Reliability ensures recoverability of data and networks in the system in case of system failure within 24 hours. Daily backups of the database are also ensured.

5.2 Extensibility

The system will be developed using open source software and web service based architecture which will enable future extensibility of the system.

5.3 Maintainability

The system ensures maintainability by accommodating changes in the system and analyzing the impact of changes on the system. It also verifies the need for change in a system. The system will be developed in Java which being an object oriented language is easy to maintain.

5.4 Security

The system shall be designed to ensure secure access of confidential data as well as data consistency and protection. All external communications from the system's server must be encrypted. Also, change in inventory will be performed only by the Retail Store manageror the Warehouse manager.

5.5 Portability

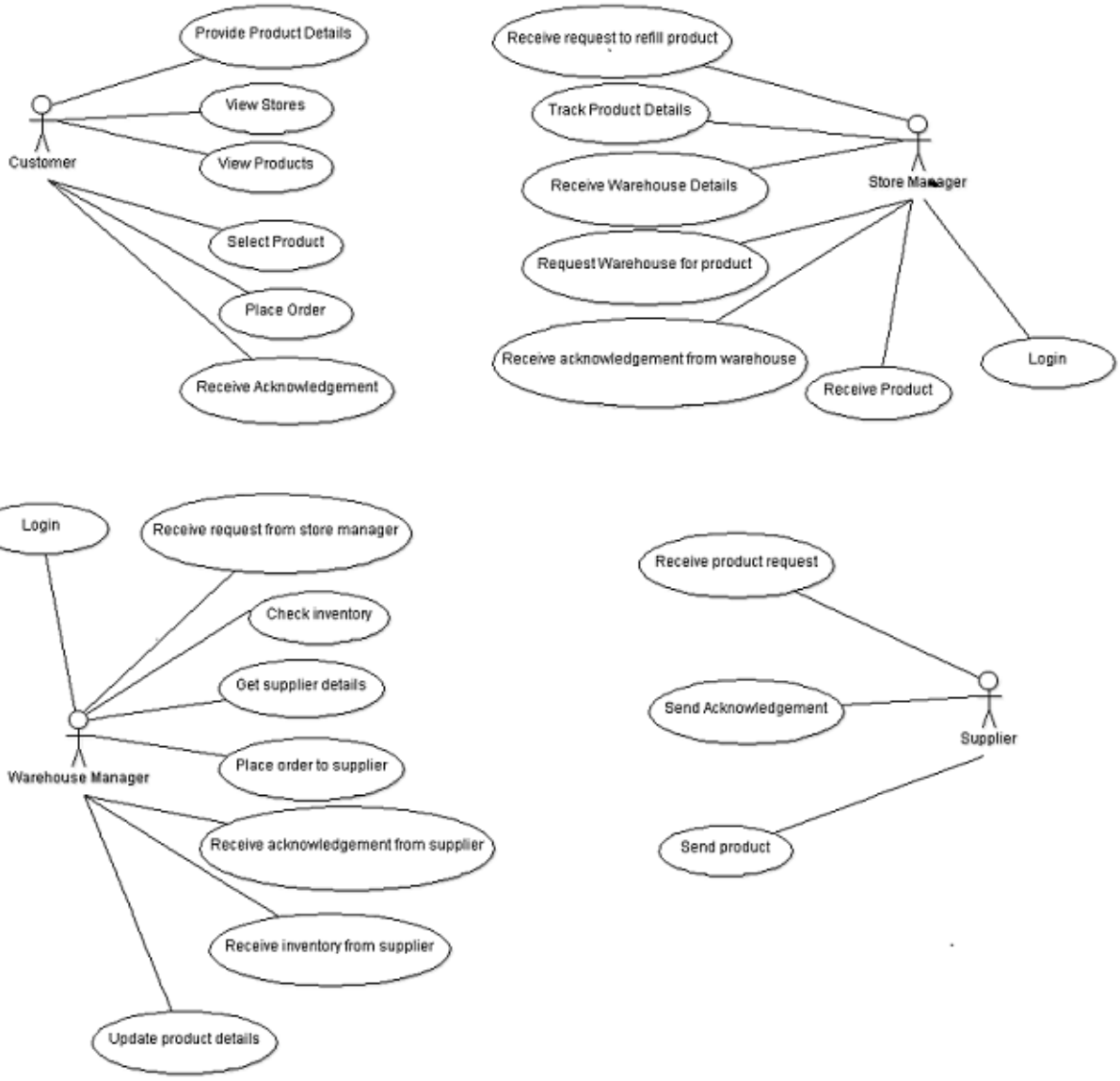
The system will be designed using open source software which will ensure that the system will deploy not only on Windows but also on Linux with minimal changes in code. Also, this will ensure code reuse. The system will also operate with any relational database.

5.6 Availability

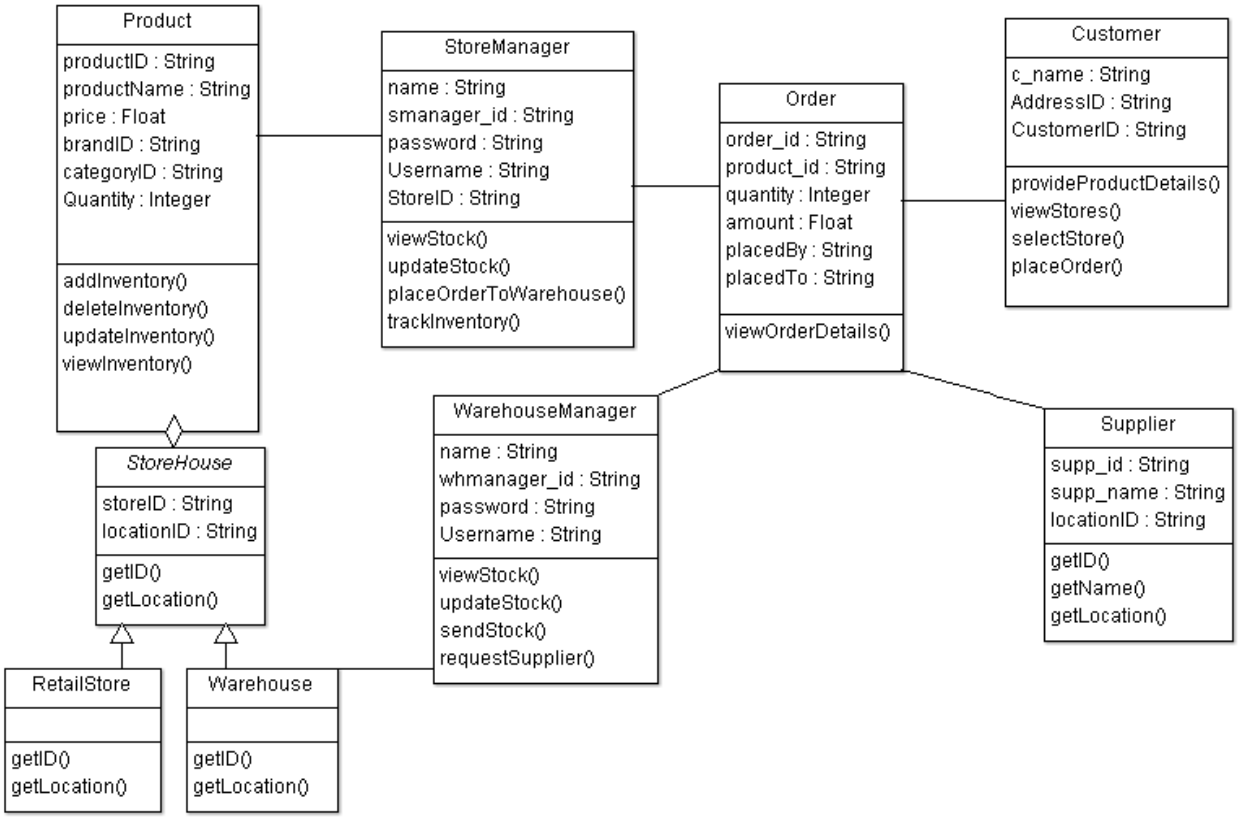
The system shall be designed with error recovery routines in order to handle failures ensuring 24x7 availability.

6. UML

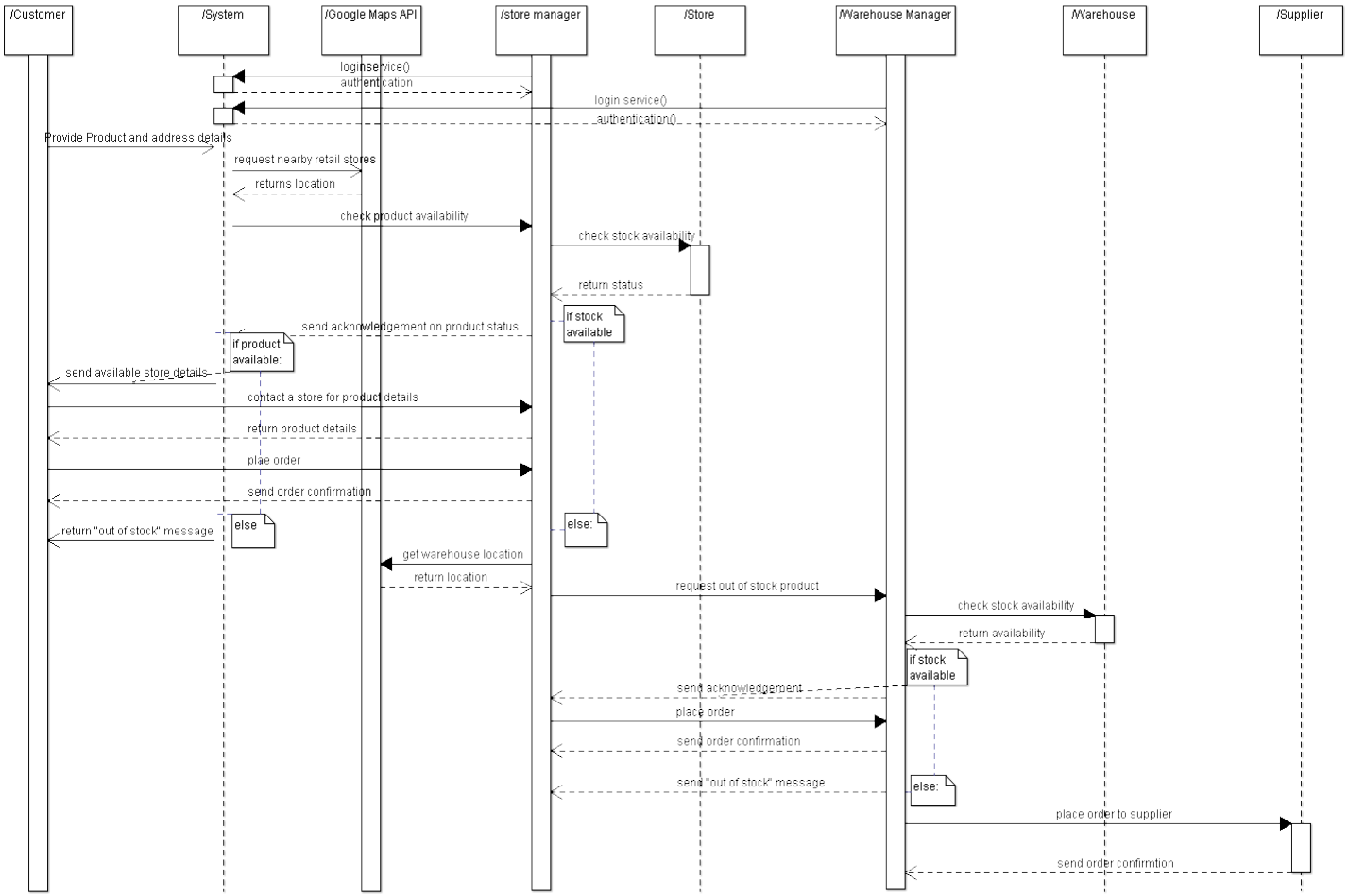
6.1 Use Case Diagram



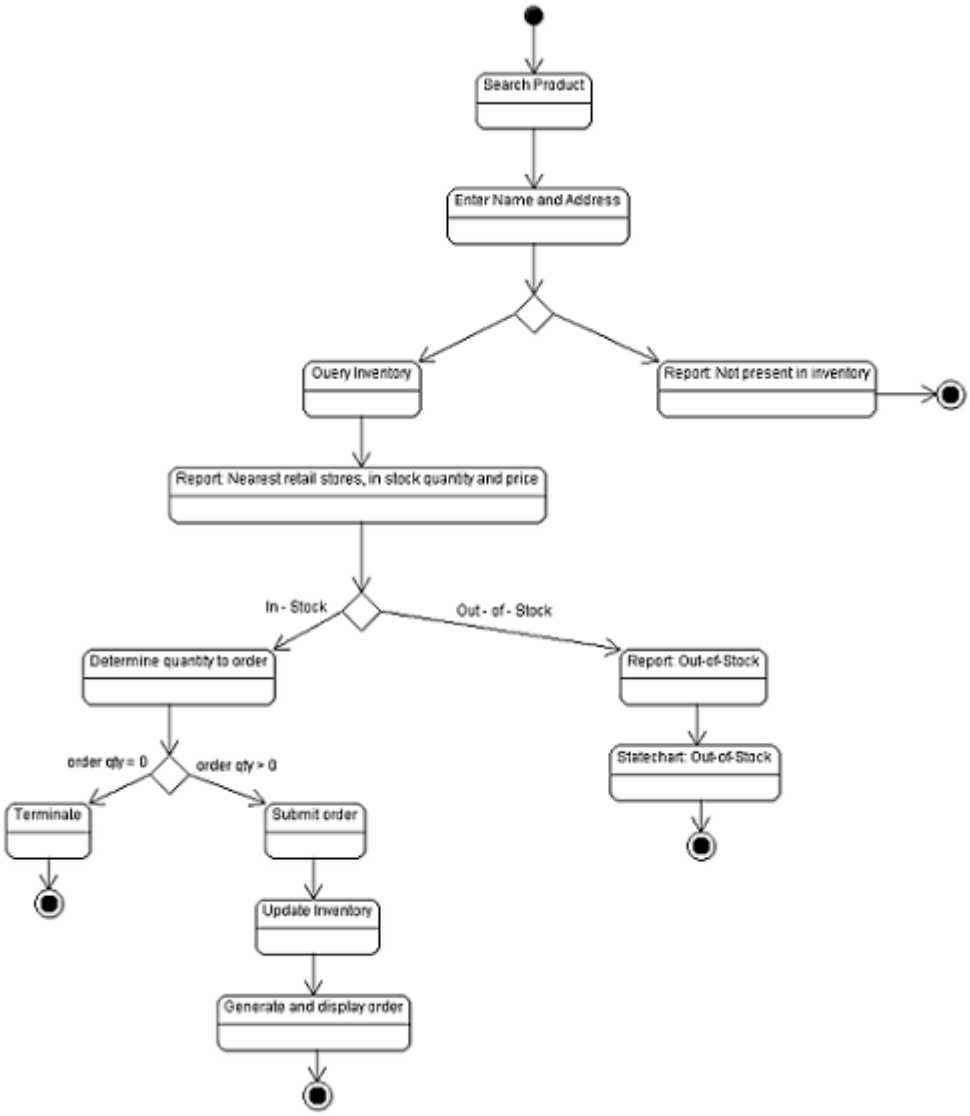
6.2 Class Diagram



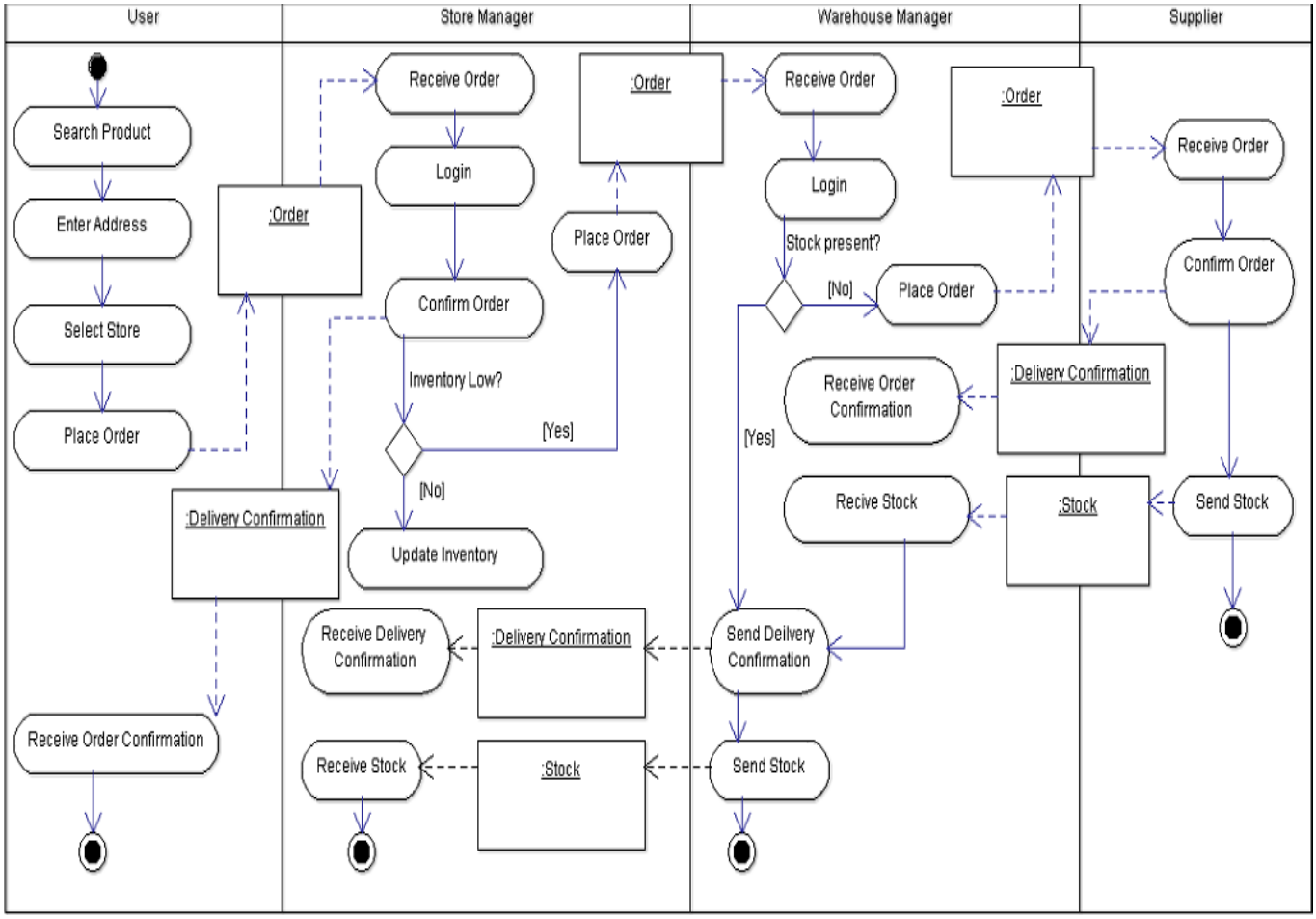
6.3 Sequence Diagram



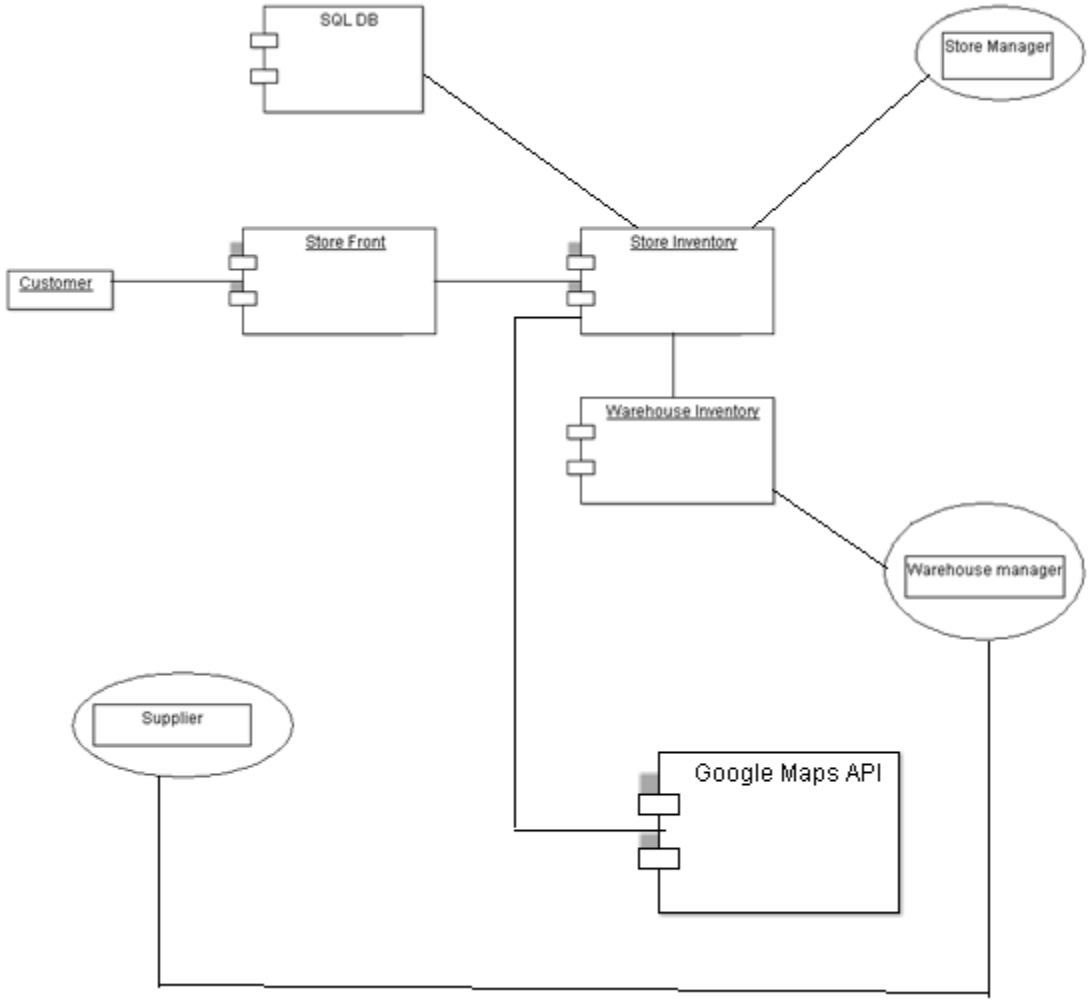
6.4 STATE DIAGRAM



6.5 ACTIVITY DIAGRAM



6.6 COMPONENT DIAGRAM



7 CLASS STRUCTURE AND INTERFACES

Project: CheckStoreOrdersInWarehouse
Class: CheckOrders.java

Project: Customer
Package: cust
Classes: AddressConverter Geometry getlatlang GoogleResponse Location Result
Package: monodb
Classes: Request

Project: PlaceOrderToStore
Class: PlaceOrderToStore

Project: ReplaceStockToStore
Class: SetStoreOrder

Project: StoreManagerClient
Package: org.apache.ws.axis2
Classes: LoginVerification

Project: SupplierClient
Package: pack
Classes: Product
Package: org.apache.ws.axis2
Classes: Product, WarehouseOrder, LoginVerification
Project: TrackProduct
Classes: Tracking

Project: WarehouseManagerClient
Package: org.apache.ws.axis2
Classes: LoginVerification

Project: WarehouseOrdersInSupplier
Classes: CheckOrders

8 RESTRICTIONS:

The project has to be developed as a course project for the Engineering Distributed Objects for Cloud Computing class at the University of Illinois at Chicago. The project has to be developed by 5th December 2014. The project must follow the project specification of the course project. The application must have 20 database tables and 4 columns in each table. Tables must have primary keys and foreign keys.

9 LIMITATIONS:

The Inventory Management System project fails to achieve the following and has a few of the following Limitations:

1. As the database size has to be kept below 2GB the application cannot store all the required data which might be present in the Amazon Data sets
2. The application fails to have a high fault tolerance, if the database is down then exceptions are thrown but not consistently handled.
3. The consistency of the application is compromised as the data is not consistent with the social network API.
4. Availability is compromised because the database is not replicated to continuously. Nor is there any backup database to provide service to any user.
5. Even though the QPS allows scalability there is no way to handle scalability.
6. The Application is run on java, so it can only run on machines where java is present.
7. Reliability can be an issue as the application is distributed and messages can be lost and there is no synchronization between the client and the QPS and the database which are all on different machines.

10 TEST PLANS

The test plan for the IMS has been specified as below. Each class and module must is tested by the author of the module, the author has written JUnit and JMeter test cases for their particular modules.

Test Suite:
Example - TestCase1.java and TestCase2.java: This class tests the functionality of supplier to ensure he can restore the stock to the warehouse.
TestCase3.java and TestCase4.java : Ensures Warehouse manager client is working
TestCase5.java and TestCase6.java : Ensures Store manager functionalities are working

Integration Testing: After all the modules, QPS, MYSQL and web services have been completed an integration test of the entire system should be performed. Once the each module has been integrated with another the all the unit tests have to be re performed.

System Testing: Once integration has been done and integration tests have been done, The IMS software should System testing where the integration and unit test have to be performed along with system tests.

11. TECHNOLOGIES USED:

The Programming language used is java.
The development IDE is Eclipse.
Databases used are MYSQL which is the relational DB
The non - relational database used is MONGODB
We also used software like notepad++ and beyond compare.