DE-Store Prototype - Software Architecture Coursework

# **Possible Software Architectures**

In this section, I am going to discuss the descriptions of the Three-Tier Architecture and the Two-Tier Client Architecture. I am also going to discuss the advantages and disadvantages of both architectures and how these both architectures can be used in the DE-Store prototype.

**Client-Server Architecture**

A client-server architecture would divide the application into two parts, ‘client and server’. The app implemented on a computer network that connects the client to the server. The server part of the application gives the central functionality such as any number of clients can connect to the server and request which it operates a task. The server accepts these requests, performs the needed task and return any results to the client, as required.

For example, if the DE-Store implemented by using the Two-Tier Client-Server Architecture, a store manager can search and look at the information of each product by typing a particular sale offer and then set the price of a specific product. The DE-Store application can give an interface and a methodology of selecting or locating a product’s information, as well-displaying products details and enabling user to set a product price.

The DE-Store application might be in a monolithic Desktop Application structure, but if anything modified or updated, the whole application redistributed again. Constant redistribution would not be suitable for the DE-Store, as each product price would keep changing, and each customer loyalty card might change regularly as well. An improvement would be splitting the DE-Store into two parts. One part is where the client can give an interface for the store managers and other employees and distribute to them. Another section can be handled and run on the DE-Store own server machine. The client application can display information and be used to pass information to the server for searching such as products information. The client application is known as the Presentation Tier, and the server application is known as the Data Tier. This client-server model is known as the Two-Tier Architecture.

In the Two-Tier Architecture, many clients can connect to the server application and request details about products or customers. The server has to process these requests and transmit the reply to the client, which initiated the request and not to any client. As long as the network is operating well and the server can catch up to replying to all requests it receives shows a Two-Tier Architecture can give the same amount of services as a One-Tier Architecture.

The collections of products and customers details can be stored centrally on the server and then be changed. This strategy enables other ‘centralised’ information to be maintained and transmitted to clients such as stock level of each product. The store managers and other employees of the client would find it much more comfortable and smaller to work with compared to the complete application. The DE-Store would have improved control and monitor the usage of the server application.

There is also variety of functionalities through a Two-Tier Architecture such as a client monitor the quantities of each product within a possible DE-Store prototype. If the client is a Desktop Application, it can request and display each product name, id and stock level from the webserver. The information you could obtain restricted to the pre-defined views of the information given by the server. So while the server’s information could include a product stock level, if you want to find how the stock level of that product changes every month, you need to calculate it yourself based on the values that the server provides. However, you might also access this server over the internet using a more intelligent client such as a mobile app. This client could extract the stock level of a product and calculate any values you desire. This type of client can also produce a report in a pie or bar chart form, which displays total profit, several customers and number of purchases within the DE-Store.

In summary, the desktop app client displays information which the server distributes is known as a thin client, and the more intelligent mobile app client that enables you to take the information the server provides and to manipulate it and display it in different ways according to personal user requirements is known as a thick client.

The Two-Tier approach is highly beneficial for applications which operate over a network. The client distributed to user could change, while the server part can be a centralised component which maintains dynamic, global data within a consistent and secure methodology for the DE-Store and the Store Managers and Normal Employees can access and use. If a third party component, such as a web browser, gives the functionality needed to support the application, then It can be adopted as part of the solution, with a massive saving of development effect. However, there is a disadvantage from splitting the application through a network where data sent over what could be a slow or unreliable connection.

**Three-Tier Architecture**

The Three-Tier Architecture is even split up into Data-Tier and Presentation-Tier similar to the Two-Tier Architecture. However, the client and server furthermore subdivided into different modules. The client can be responsible for received data processing and presenting the information. This strategy is useful for generating a report in a pie chart to show how the DE-Store is performing by calculating total profit, number of purchases and number of customers.

The server software could have one or multiple data storage, such as a database system. Three-Tier Architecture can make the DE-Store able to have two databases. One database can contain a table that contains id, name, sale offer, quantity and price of each product. Another database can include a table which stores purchase information that indicates buyer id, product id, product name, quantity and amount paid. Splitting data in this methodology enable occasional backups of more static data with more frequent backups of the dynamic data.

In a Three-Tier Architecture, more layers added on the server-side which is a data tier and another layer which deal with interactions with the data tier, such as retrieving requested information or validating data which inputted into the data stores. This layer between the server and the data tier is called middle tier, application tier or middleware. An application which uses middleware to handle data requests between a user and a database is known to use a Multi-Tier Architecture. A Three-Tier Architecture is a type of Multi-Tier Architecture.

Similar to the Two-Tier Architecture, there are benefits to breaking down the DE-Store into multiple tiers. Each Tier can be modified more naturally as it relies on the precise information of the other levels with that it interacts. This strategy relies on how careful the developers to adapt to a methodology which makes sure the layers or components entirely loosely coupled, which is not guaranteed by just separating layers alone. Developers are required to indicate precise and minimised interactions between the tiers. Reducing interactions between levels is more straightforward with a Three-Tier Architecture compared to the Two-Architecture as the data-tier and presentation tier does not interact with each other in a Three-Tier Architecture as there is a middleware between the two Tiers.

Another advantage of the Three-Tier architecture over Two-Tier is that it is more scalable, so it makes the DE-Store more expandable easier and have a network configuration of many employees and store managers as the presentation layer, and the data layer is separate from each other so multiple of employees logged into the DE-Store at one time would be more of a problem in a Two-Tier application compared with a Three-Tier Architecture. Another advantage of the Three-Tier Architecture over Two-Tier Architecture is that by adding Application Tier between the Data Tier and the Presentation Tier can make the DE-Store more secure as the passwords of each employee can be encrypted and stored in a database which would not be accessed by the client (Presentation Layer) as there is a middleware between the Presentation Layer and the data layer. This type of security is not present in a Two-Tier Architecture.

# **Chosen Architecture – Three Tier Architecture**

The Three-Tier Architecture makes it more complex to create Applications like the DE-Store than using the Two-Tier Client and Server Architecture as is more difficult to build and maintain an Application with more layers compared to developing an application with fewer levels. However, reducing interactions between levels is more straightforward with a Three-Tier Architecture compared to the Two-Architecture as the data-tier and presentation tier does not interact with each other in a Three-Tier Architecture as there is a middleware between the two Tiers so Three-Tier Architecture would be better at making the DE-Store to be loosely coupled compared to the Two-tier Architecture so Three-Tier Architecture would be easier for developers at modifying and maintaining the DE-Store compared to the Two-Tier Architecture.

There are more strict requirements to the speed of the network between the database server and the application servers which would slow down the application servers of the DE-Store when it becomes more expandable which is not a problem with Two-Tier Architecture. However, the Three-Tier Architecture is more Scalable than the Two-Tier Architecture which means it makes the DE-Store more expandable easier and has a network configuration of unlimited amount of employees, store managers, customers and products as the data tier can store multiple databases which can store unlimited amount of store managers, customers, employees and products.

The Three-Tier Architecture would make the cost of software, hardware and maintaining software more expensive. Therefore, this would be a problem when there is changing requirements of the DE-Store in the future. However, by adding Application Tier between the Data Tier and the Presentation Tier can make the DE-Store more secure compared to implementing the DE-Store in a Two-Tier Architecture as the passwords of each employees can be encrypted and stored in a database which would not be accessed by the client (Presentation Layer) as there is a middleware between the Presentation Layer and the data layer. This benefit means that Three Tier can protect the DE-Store from security breaches more efficiently than a Two-Tier Architecture would be able to.

Therefore, I have decided to use Three-Tier Architecture instead of Two-Tier Client-Server Architecture because it provides higher qualities than the Two-Tier Architecture would be able too. The Three-Tier Architecture would make the DE-Store more scaleable compared to if the DE-Store implemented in a Two-Tier Architecture. The Three-Tier Architecture would make the DE-Store modules more loosely coupled compared to if the DE-Store achieved in the Two-Tier Architecture. The Three-Tier Architecture implemented DE-Store would be more secure than a Two-Tier Architecture applied DE-Store.

# **Design**

## **Entity-Relationship Diagram**

A screenshot of a cell phone

Description automatically generated

One customer can order 0 to many products.

One Store Manager can give financial approval to one customer.

One Store Manager can give loyalty card to one customer.

One Employee can give financial approval to one customer.

One Employee can give financial approval to one customer.

One Store Manager can set price to one product.

One Purchase would collect one purchase information from one customer.

## **A close up of text on a white background Description automatically generated**

## **Three Tier Architecture Design**

A screenshot of a cell phone

Description automatically generated

# **Evaluation of Design and Implementation**

The DE-Store prototype has successfully implemented as a desktop Application structured in a Three-Tier Architecture. It has successfully implemented the specification, and the high quality attributes that a Three-Tier Architecture able to offer.

I have managed to achieve a Price control feature within the DE-Store where the store manager can set the price of the products and select products on different sale offers. The store manager can search for products details with a specific sale offer by inputting the sale offer within a text box and clicking on the search button. The sale offer string transmitted to the application tier and the application tier process string as a query and send the query to the data tier. The data tier would execute the query and would then retrieve each product information from the database and would send a list of products to the application tier. The application tier would process the list and would send it to the Presentation Tier. The presentation Tier would display each product information with that specific sale offer within a list box.

A Store Manager can also set a price of a particular product by typing in product id within a text box and the intended amount in another text box. The Store Manager can then press the ‘Set Price’ button. The Client would send the price and phone number to the Application Server. The Application Tier would process these strings to the data server. The data tier would send query to the database to set the price of the product with that particular product id, and then the data tier would send a true boolean value to the application tier, and the application value would send true Boolean value to the presentation Tier. A message box is shown on the screen stating that the product price is successfully set.

I have managed to successfully implement an Inventory control feature on the DE-Store where the DE-Store makes warning messages for items that are low in stock and sends email messages to each stock manager email box. This occurs when the Data Tier checks quantity of each product in the database, when the amount is 5 or less, the data server would retrieve email messages of each stock manager from the database and send it to the application server, the application server would then send email message to each Stock Manager and send message to a client that a user is currently accessing a warning message generated on the screen.

I have also managed to implement a Loyalty Card feature where a discount loyalty card is given to loyal customers. Each customer that has brought least 50 products provided a 50% discount loyalty card. The Data Server would transmit Query to the database asking for number of products delivered and phone number of each customer. For every phone number and number of products brought received from the database, the data tier would send the query to the database to set a loyalty card discount to the customer if the number of products brought is least 50. The data server would also send these details to the Application Tier, and the application tier would send it to both Store Manager and Employee clients so all employees can see what customers have a loyalty card and who does not by displaying customers details within a list box.

I have also managed to implement a Finance Approval feature where finance approval given to customers with good credit. The Data Server would transmit a query to the database to retrieve credit and phone number of each customer. For every phone number and credit received from the database, the data tier would send the query to the database to set finance approval of the customer to ‘yes’ value if the customer credit is good. Otherwise, query sent to the database to set finance approval of the customer to ‘no’ value. The data server would also send these details to the Application Tier, and the application tier would send it to both Store Manager and Employee client so all employees can see what customers have finance approval and what customers do not by displaying each customer details within a text box.

I have also managed to implement the high qualities that a Three-Tier Architecture can offer. I have managed to make the DE-Store secure in a way that only Store Managers and employees can log in into their accounts. When a Store Manager or Employee register account, their password is encrypted and then stored within the DE-Store database. When a user attempts to logs in by typing in username and password in the appropriate text boxes within the Login GUI that’s present in a client, and the client would then send the inputted password and username to the Application Server, and the Application Server would process these strings and send it to the data server. The data server would send a query to the database to retrieve usernames and passwords from Store Managers or Employees Table(depend on user). The Data server would compare inputted username to each username from the table, and if there is a match, the data Tier would compare the inputted password with the decrypted password of that specific username. If both passwords match, data server will send a valid Boolean value to the application server, and the application server would send the Boolean value to the client. The user would be able to go to the menu GUI. Otherwise, a message stating that the inputted strings are wrong would appear. In this way the Client would not acknowledge the password of each user as there is a middleware between client and the data server. This strategy would prevent any security breaches.

I have also managed to make the DE-Store loosely coupled by separating the Data Tier and the Client Tier by placing a middleware between these layers. The proof is that when I modify a function within the Data Server, none of the GUI’s within both Store Manager and Employee Clients is affected so, therefore, is easy to change and maintain the DE-Store application which is useful for any changing requirements in the future.

I have managed to make the DE-Store scaleable in a way where the DE-Store can add unlimited amount of products, customers, employees and store managers to the database within the Data Tier as the company can pay for more space within the data server. Therefore, the DE-Store can be expandable and manage to adapt to any changing requirements in the future.

A suggestion to enhance the DE-Store in the future is to horizontally scale the application servers so a server can deal with price control, another server can deal with customers, and another server can deal with store managers and ordinary employees. This strategy would improve the DE-Store as different responsibilities spread out over different systems so they can run on different operating systems, able to be separated individually and do not affect other services when compromised which would make the DE-Store more secure and maintainable.