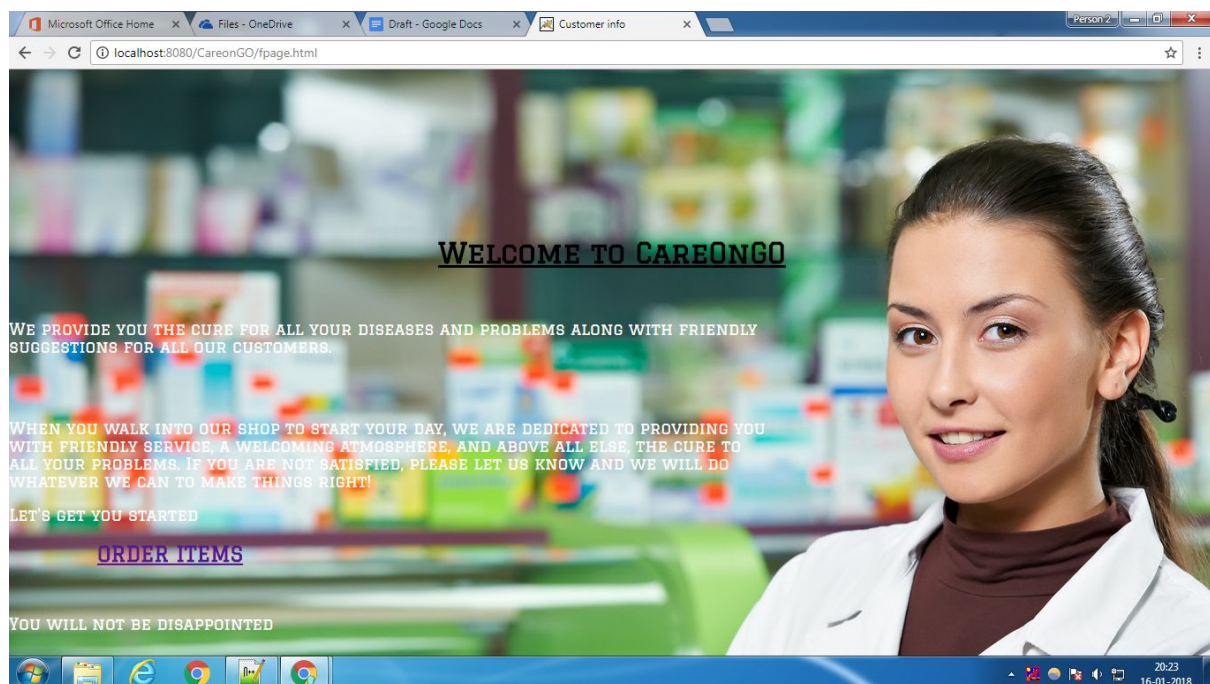


# CareonGo Pharmacy OMS

**Objective** - The main aim of this project is to create an order management system for careongo pharmacy in order to take their business online. The project consists of two ends - customer end and retailer end.

First let us look at the Customer end in order to know what the users will be looking at.

## Customer end -



The first page consists purely of HTML and CSS.

In HTML, I have used heading tags in order to give the major headings and used <center> tag to align it to the center. Other than that I have used <p> for paragraphs that consist of the text. I have used style tag to use the background image and set

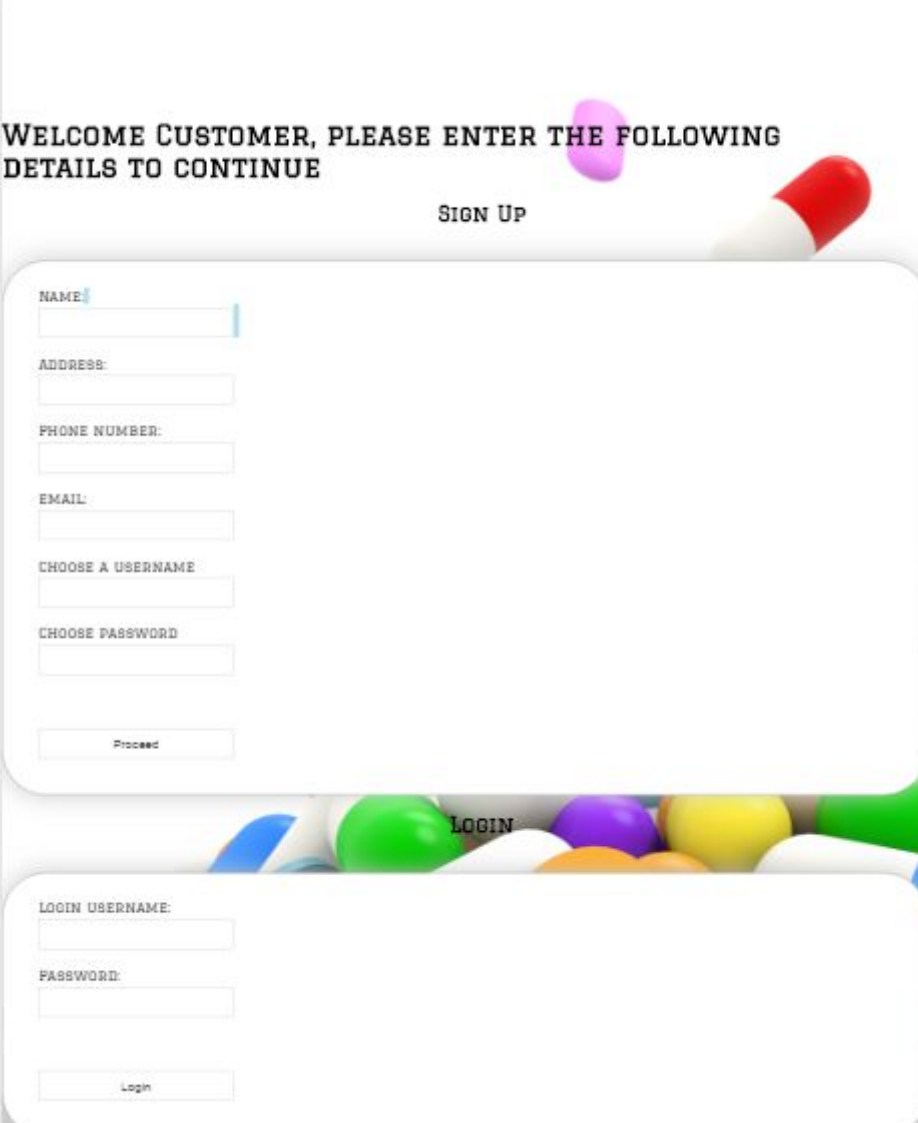
it to no-repeat so that the images are not repeated. The only major “Order Items” link you see is a hyperlink set by href that points to my next page .

Let us skip to the next page that is called when I click order items. It displays a page with two basic options - Sign up and login. Sign up is for people who are using the page for the first time. Login is for pre-existing users. Right now, there is no error msg for wrong credentials for login yet. So we will discuss what will happen for both the options.

## **1. Sign Up -**

Entering the credentials required for sign up and proceeding does 2 things. It updates my database and also moves on to my backend java code. In my backend code, the values that i

have input are added to my customers table and a customer id



WELCOME CUSTOMER, PLEASE ENTER THE FOLLOWING DETAILS TO CONTINUE

SIGN UP

NAME:

ADDRESS:

PHONE NUMBER:

EMAIL:

CHOOSE A USERNAME

CHOOSE PASSWORD

LOGIN

LOGIN USERNAME:

PASSWORD:

is generated. It then redirects to other java page which has my order html page.

## 2. Login -

Entering the credentials here is going to take the control over to my backend code that just verifies the username and password in order to work with the customer id already present for the user. It then redirects to the order page which is what my sign-up page points to as well.

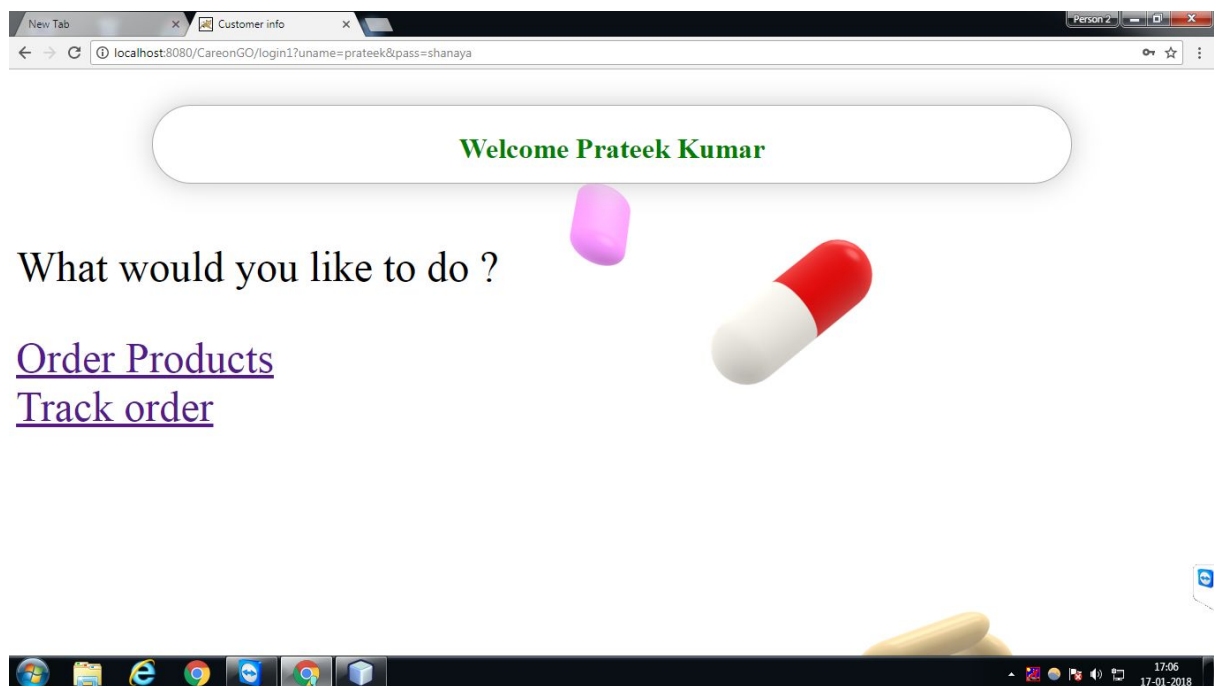
Now the main question arises that how does the database and java connection works as well as how is the insertion taking place. Well in order for any of the other pages to make sense, we need to be know what is going on in any of the java pages.

You see I keep calling them my java pages, they are not purely java programs but servlets. **Servlet** technology is used to create web application which is an application that can be accessed on the web. Servlets gives access to all the Java APIs. Inside the java servlet program, I use various imports. `HTTPServlet`, `HttpServletRequest`, `HTTPServletResponse` are the major ones as they allow us to request service from other pages as well as respond to the requests. We import `sql` in order to use all the sql java functions. While coding we may face various exceptions, so to keep that in check, we use `try` and `catch` inside the code as well import those exceptions in order to know what exception occurs. We use `WebServlet` annotation. It is used to declare a servlet. So we create a java class that extends the `HTTPServlet` class that we will see why ? Inside this we use `doGet`, `doPost` functions. Now `doGet` can send limited data to get response context. `doPost` sends unlimited data along with the request to web resource. If a form is submitted with GET post, one should use `doGet` and vice versa else an exception is thrown. Now these methods have two parameters namely the request and response. Their work is exactly what it sounds. These methods also throws exceptions. If no matching catch is found then the default exception handler will halt the program. `throws` is a keyword in Java which is used in the signature of method to indicate that this method might throw one of the listed type exceptions. Next we use `PrintWriter` out and initialise the response `getWriter`

which basically. `PrintWriter` prints text data to a character stream. `getWriter` returns a `PrintWriter` object in order to send character text to client. We declare some strings and initialise all the values that we got from the HTML page. Now basically, we need a connector that would connect my database with the code. Since i use MySQL , therefore I use the JDBC driver and use the jar file in the project for the connection. We use `getSession()` method in the `HTTPSession` class to return the current session. If it doesn't exist, then the method creates one. We get the values from the page by using `request.getParameter` that basically requests that parameter from the page. The method takes a string as a parameter which is usually the name in the tags of the html input tags. Next we use `out.println` to print the html code. Anything we write here is taken as a part of the HTML page and all tags do work. Next, for all the code that might throw an exception, we enclose it in try block followed by catch so as to initialise what to do with the error. In the try block, we first use `Class.forName` is used to return the objects associated with the class. Here the class we call is `com.mysql.jdbc.Driver`. It dynamically loads the driver's class file in the memory. Now we establish connection using `getConnection(url)`. Here the url basically consists of **`jdbc:mysql://hostname/`** `databaseName` followed by username and password for the database. Once a connection is obtained we can interact with the database. The *JDBC Statement* and *PreparedStatement* interfaces define the methods and properties that enable you to send SQL commands and receive data from your database. *Statement* is used basically when we use a static sql query and is used to access the database. *PreparedStatement* is used to accept parameters at runtime. In case of select statements we use `executeQuery()` that returns a `ResultSet` whereas `executeUpdate` is used to execute any

insertion, deletion or update queries. ResultSets are nothing but the data that is received from database. That's why in my program, the ResultSets are used only for select queries. This is how I have established connection in all my servlets.

Now here I give the user 2 choices, to order new products or to track his early order.



### ***Ordering new products -***

While putting the html code inside the out.println, i used the form tag and defined the action as to where the control should go once the Submit button is clicked. Now in order to pass a value that is not a name in the input tag, we use the session. We can set a value to be used in between servlets by using session.setAttribute() with the parameters as to what the value should be passed as and what value should be passed. Now in the receiving end, we use getAttribute() instead of getParameter. Remember the return type of getAttribute is an

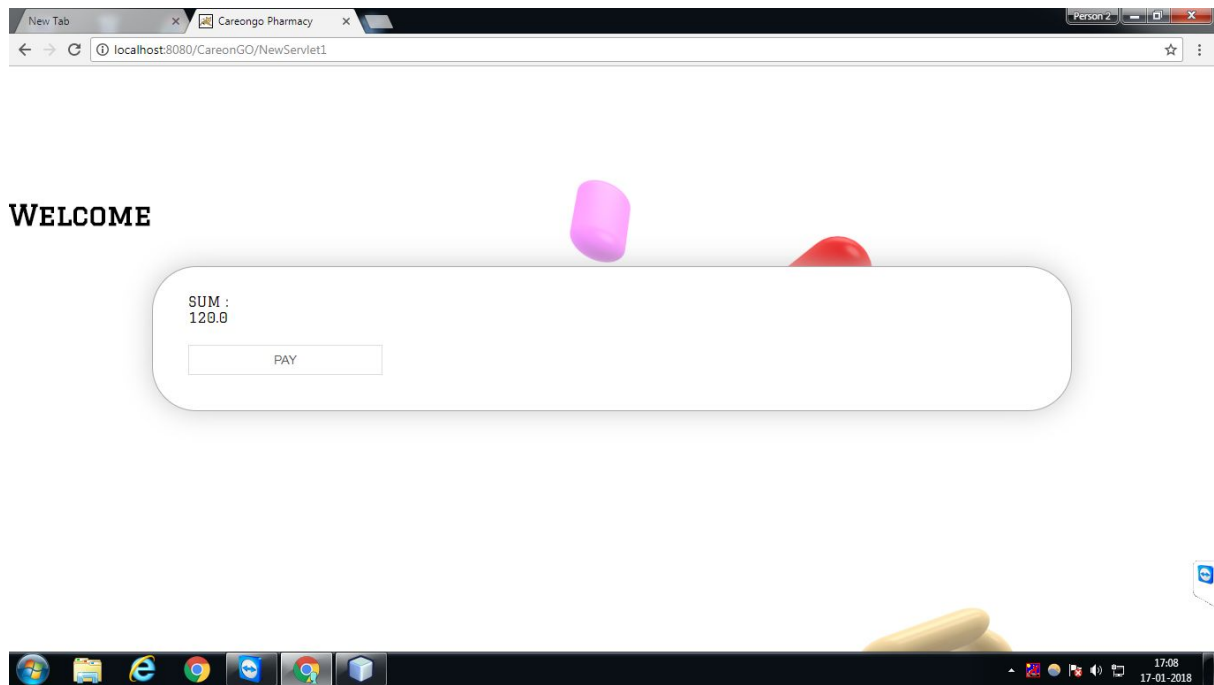
object and thus should be converted to the desired data type. In the order page I display a table with the list of products available for sale along with the quantity available, the price per unit and an input box for desired quantity.

WELCOME 2

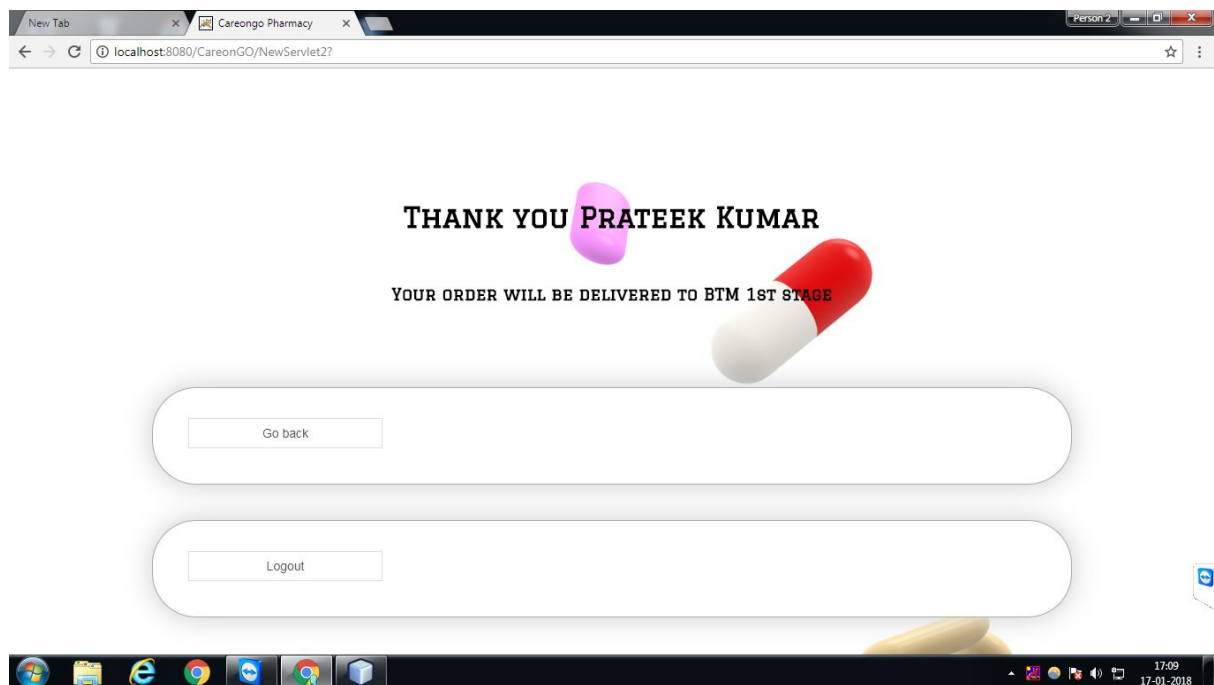
MEDICINES					
PRODUCT	PRICE/QUANTITY IN Rs	AVAILABLE QUANTITY	TYPE	DESCRIPTION	NEEDED QUANTITY
CROCIN	2	5	TABLET	HEADACHE	
DCOLD	2	10	TABLET	COLD	
IODEX	30	14	BALM	HEADACHE, SORE-THROAT	
Moov	50	7	BALM	MUSCLE RELAXANT	
NIMUSLIDE	3	23	TABLET	HEADACHE, BODYACHE	

Submit

Now i never had to call this input box again and again. The servlet programming allows me to print html code in my java program so i just put it in a while loop. I used the select query for retrieving the data. While giving a name for my input tag, I created an array that stored all the values that are entered in the quantities. After entering the quantities ,the user shifts to a page where he is shown the total amount and a pay button.



Here i ordered 4 quantities of IODEX.  
Now the button is just for the show which redirects to a page that shows a Thank You message.



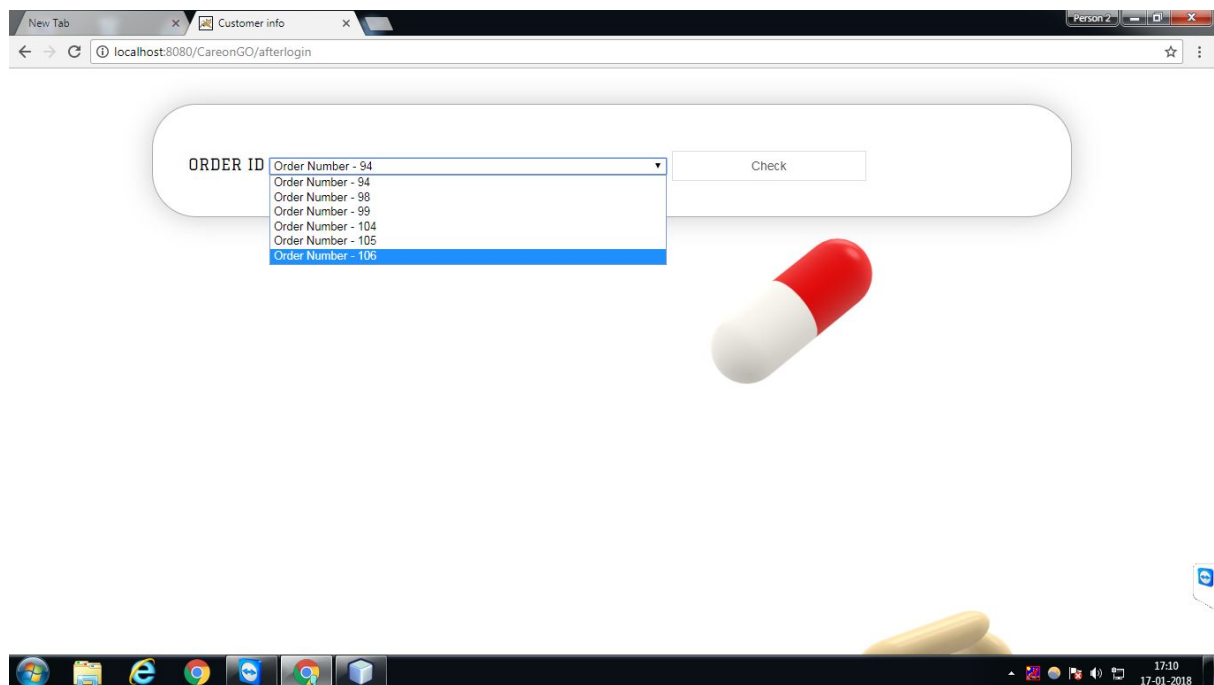
How did I calculate the amount to be paid ? This was a challenging task. So I used `getParameterValues()` in order to store the values of the array in an array when the page redirects. Now I have 5 products, so if I gave quantities for the



first two, my array looked like [2,2,,,]. So, i use an if condition that checks if the value is not null. Now my array indexes are in sync with the product ids with a difference of 1. So I call in the unit price and convert the string value into float and do the same with the array value at that index. Then, I multiply the values with each other and keep storing a variable called sum so that I know the total order. I enter the details into the database. I will talk about the database later for now.

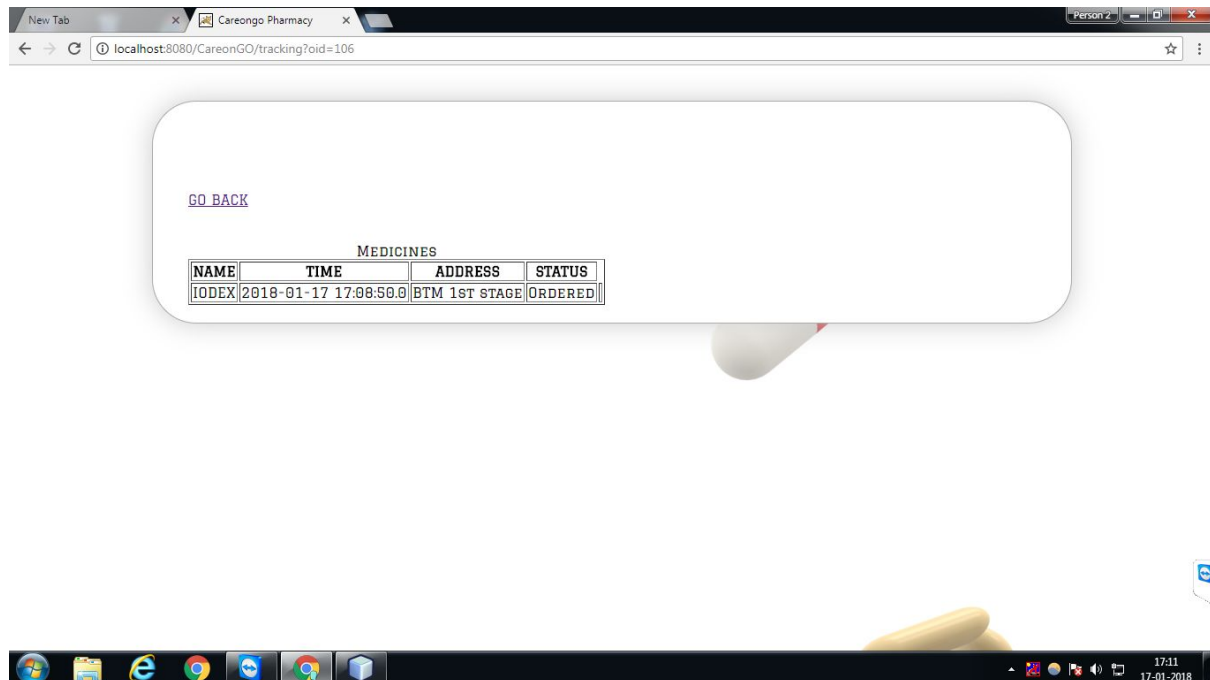
## ***Tracking order -***

For this I take the customer id that check with the username and password, and then display all orders with that customer id stored in my order table.



So with the `setAttribute()` and `getAttribute()` . I use a dropdown option to display all the orders listed with the customer id. Now, whatever order id is selected from the dropdown menu, it's value is stored with the name in the input tag, so when i request the `getParameter ()` with the name , i can get the selected order

id. With this, the order status , along with the amount and timestamp is displayed.



How does the timestamp thing in my project work ?

In NewServlet1.java, I use additional imports like `java.util.calender`, `date`, `SimpleDateFormat`. These are imported in order to use the methods that these packages provide. First we create an object for Date and Calender. We use `calender.setTime(date)` so as to set the current date and time. I create another variable `strDate` and store the time in it. I call the `SimpleDateFormat` whose sole purpose is to decide the format of the date. Now any timestamp in database is stored in the format `yyyy-mm-dd HH:mm:ss` . Mind that it's H and not h. This is the 24 hour and 12 hour format. I convert my `strDate` to this format. Now after calculating the total bill, i use the insert command. I mark the values as `(?,?)`. As we discussed, we use `preparedStatement` for such cases where we provide values later. We use `setString` command that sets the ? value as

whatever the variable is passed. I put all of it in my order\_det table, but for that I need the order id for the current order, but it is an auto increment, so how do i get it ? The answer is timestamp. strDate allows me to retrieve the order id that uses the generated timestamp. Once this is done, it's time update the stock and delete the amount that has been ordered.

Every product of mine has a different batch number and thus the challenge was to make sure that the stocks get deleted irrespective of the Batch numbers and also, storing the batch numbers when required. So to do this, I took the Batch number and quantity with respect to the product ids under the order id. Then while the Batch number query ran, i checked for supply and demand. If supply was less, it would then update the quantity of the batch number as 0 and move on to the next, else, it would subtract the supply and demand and print the result to the quantity in my database.

There ends the customer end.

Now we move on to the retailer side and see what goes on.

### **Retailer's end -**

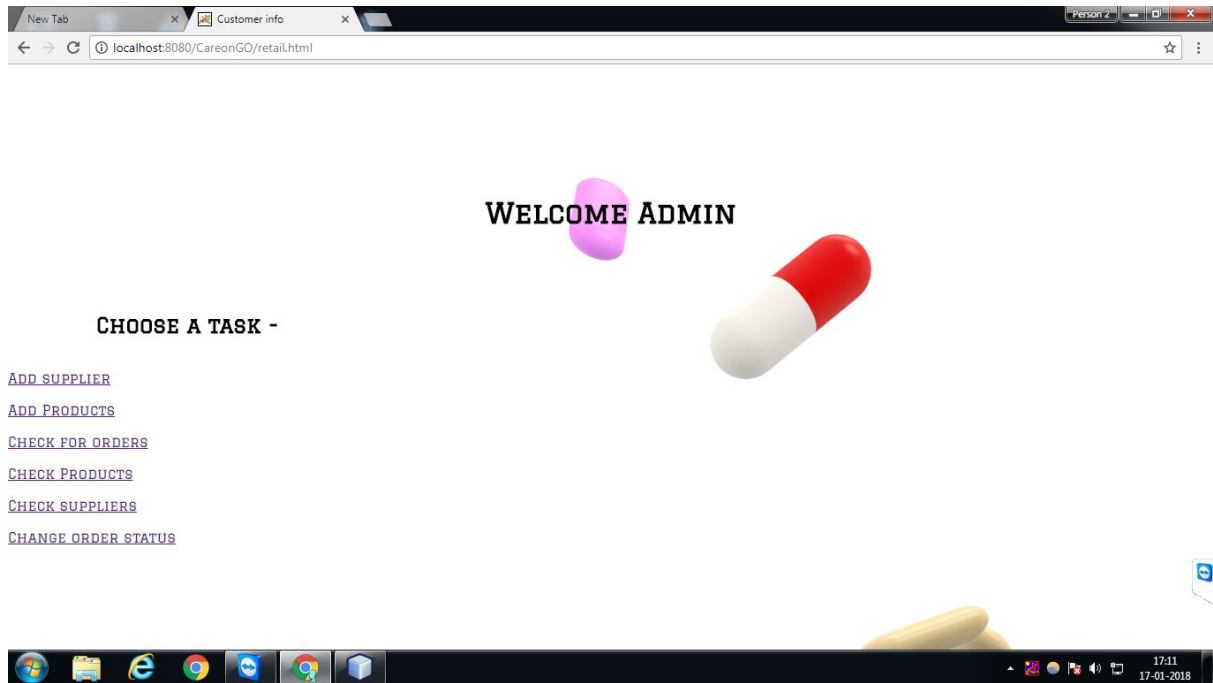
So there had to be a login page for the retailer but since it's not part of the main page, I haven't given it any login.

The first page of the retailer provides him several options.

He can

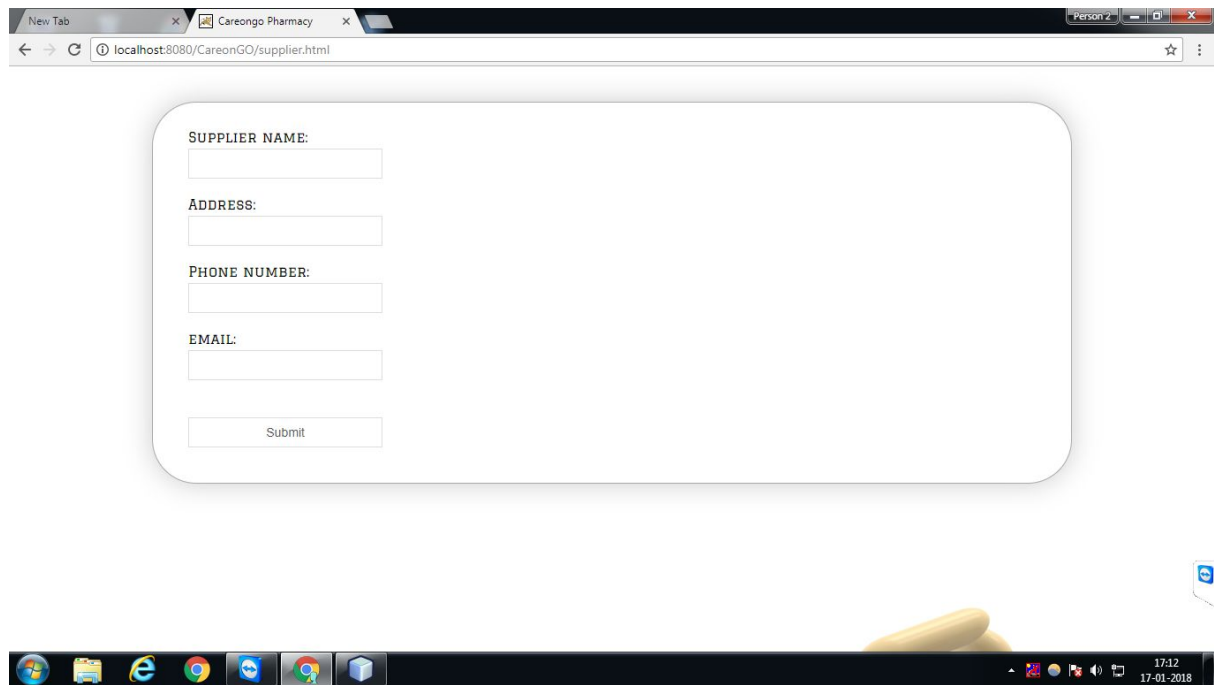
- Add supplier
- Add products

- Check for orders
- Check products
- Check suppliers
- Change order status (under construction)



All these options have a link provided by href. So any of these options takes the control to a different page.

**Add Supplier** - The add supplier option takes you to an html page where we provide the basic supplier details like the name address phone number. The Suppliers are then added to the database along with an auto incremented Supplier ID. After the supplier is added, it comes back to the main page.



***Check suppliers*** - This page displays nothing but the suppliers and their details. It is been made using a servlet program and thus uses a select query to display everything in a tabular form

To print everything in a table form, we use `<table>` tag . It is followed by a `<tr>` tag that is used for the rows along with `<th>` that gives the table headers. Now, in my select statement , i give a query that returns multiple values, so i use `while` `resultset.next()`. I use a while loop so i don't have to display the table data again and again. Inside the loop, i use the `<tr>` tag followed by the `<td>` tag that displays the table data. After every row I close the `<tr>` tag. The submit button that has a separate value takes me back to the main page.



**Add products** - Probably one of the most stressful work was to design this page. It is a java servlet code that has the html code running in `out.println`.

First I have used the radio buttons so as to select if the product name already exists and is an addition or it is a new product all together. The radio buttons are given the same name so that if one is selected, the other one gets deselected. However, the values of both the radio buttons are different and in my case, really useful.

A product is being added that means the dealer must have already been added. So, I create a dropdown to select the dealer and the names are retrieved from the database. I also have a static dropdown for the type. Other details include batch numbers, expiry date and price. For the existing products also I have a dropdown to select the product.

All these values will be stored under the unique names of the input tags that i can then retrieve using `getParameter` when the

form action sends the code to the next page. I also call in the radio button with the name. The selected radio button value will be called.

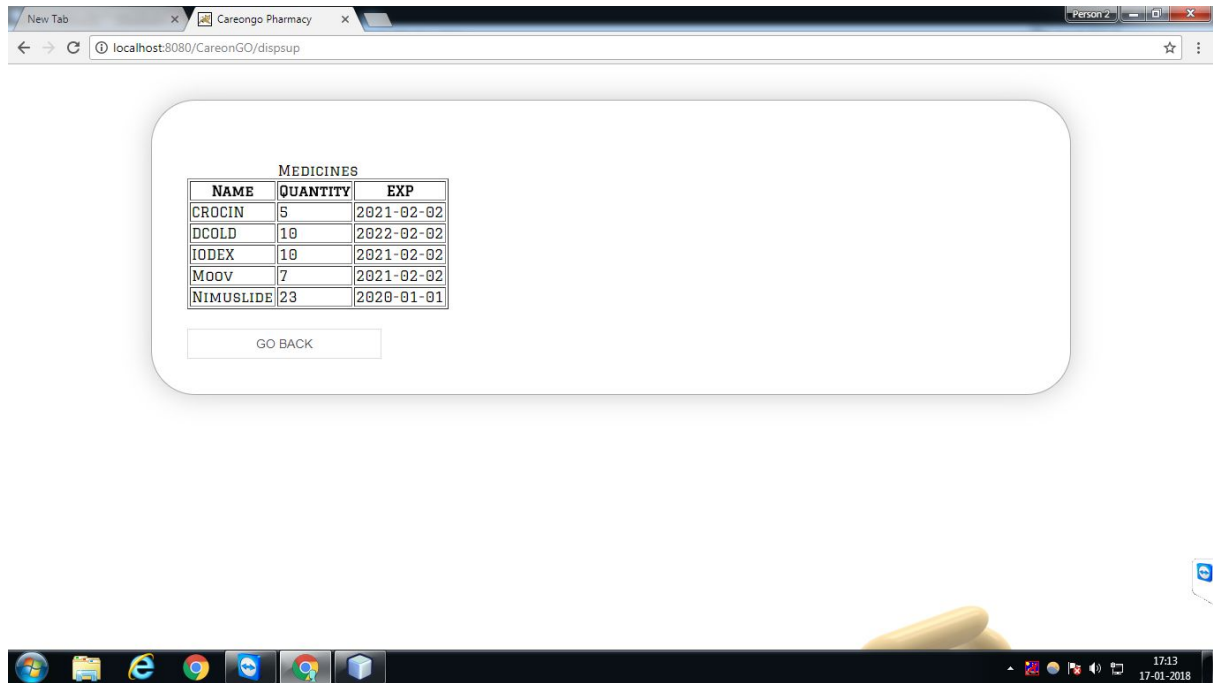
The screenshot shows a web browser window with the address bar displaying 'localhost:8080/CareonGO/inventinsert'. The browser has two tabs: 'New Tab' and 'Careongo Pharmacy'. The page content is a form titled 'PRODUCT ALREADY EXISTS? CROICIN' and 'YOU DON'T? ENTER THE'. The form fields are as follows:

- PRODUCT NAME:
- DEALER:
- BATCH NUMBER:
- MFG :
- EXP :
- QUANTITY :
- SALES PRICE:
- PURCHASE PRIE:
- DESCRIPTION:

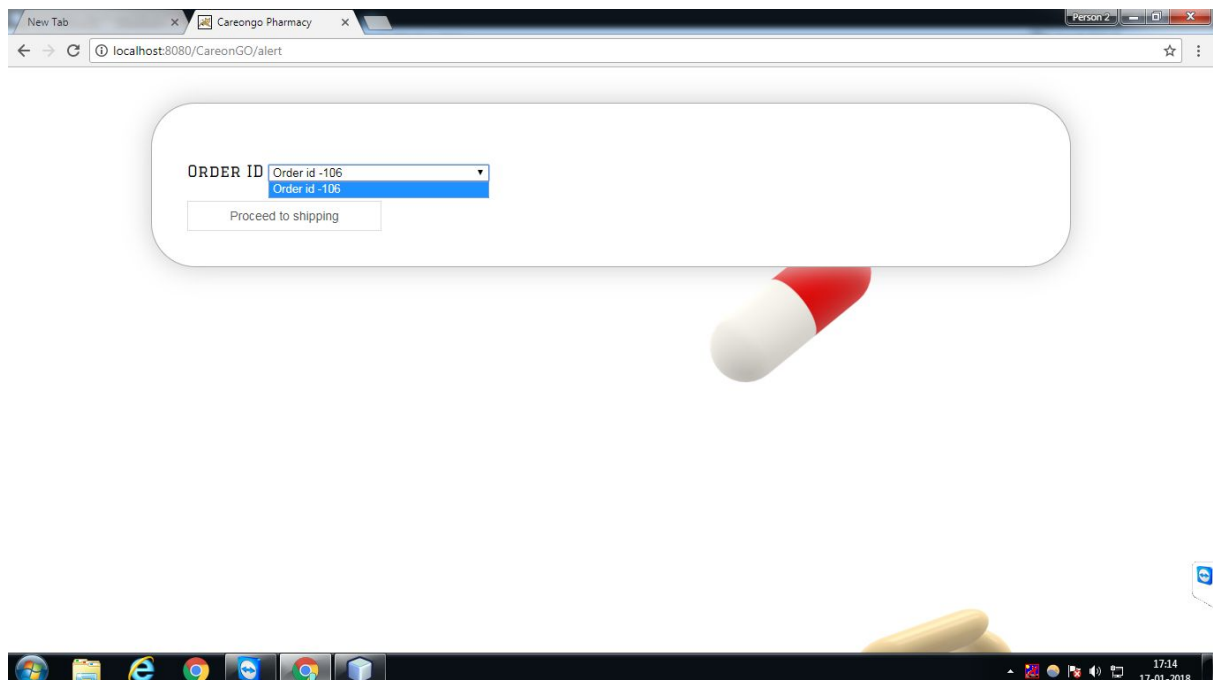
The browser's taskbar at the bottom shows various application icons and the system clock indicating 17:33 on 17-01-2018.

So in the next java servlet, i use an if condition to check which radio button was selected. If the already present product button was selected, so i select the product id of the product. If a new product button was selected, the product name is inserted into the products table along with the description and it's product id is called. Now when we have the product id from products table, we store the data in the inventory table and use `sendRedirect`. This basically redirects the page to the main page.

**Check products** - This page basically allows the retailer to check for the products that he has with him. It displays the quantity irrespective of the batch numbers as well as the expiry date so that he can be prepared for his next action.



**Check orders** - So everytime an order is placed from the user, the order status changes to ordered. So for this, the page redirects to a servlet that has a dropdown menu that shows the orders that are in Ordered status. The retailer chooses ship and update option so as to change the status to shipped.





You just saw above the Prateek had an order namely 106. So it's still under the ordered status. But when the proceed to shipping is pressed,



It shows the order and then asks to ship.

This is the entire transaction that happens in the project.

All my java servlets use the same template i.e the same import statements, the same jdbc connection code in the try block along with a catch.

So once explained above, it gets applied for all the servlet codes. All my html page has the same css styling.







# **DATABASE -**








Let me give a brief idea about the database -









My database consists of 6 tables -





- Products
- Suppliers
- Orders
- Order\_det
- Inventory
- customers













All these have various columns that can be well explained by the diagram -






SUPPLIERS		
Table stores the supplier's information		
 SUPPLIER_ID	numeric(10)	
SUPPLIER_NAME	varchar(10)	
EMAIL_ADDRESS	text(10)	
PHONE_NUMBER	integer(10)	
 <a href="#">Add field</a>		

ORDERS		
Table stores information about ORDERS		
 ORDER_ID	numeric(10)	
ORDER_DATE	datetime(10)	
ORDER_STATUS	varchar(10)	
CUSTOMER_ID	numeric(10)	
ORDER_AMOUNT	double(10)	
 <a href="#">Add field</a>		

ORDER_ITEMS		
Table stores information about ORDER details		
 ORDER_ITEM_ID	numeric(10)	
ORDER_ID	numeric(10)	
BATCH_NUMBER	double(5)	
PRODUCT_ID	numeric(10)	
ITEM_QUANTITY	integer(5)	
ITEM_PRICE	double(5)	
 <a href="#">Add field</a>		

PRODUCTS		
Table stores information about Products		
 PRODUCT_ID	numeric(10)	
PRODUCT_NAME	varchar(100)	
 <a href="#">Add field</a>		

INVENTORY		
Table stores information about Inventory .		
 INVENTORY_ID	numeric(10)	
BATCH_NUMBER	numeric(10)	
BATCH_DATE	date(10)	
SUPPLIER_ID	numeric(10)	
PRODUCT_ID	numeric(10)	
MFG_DATE	date(10)	
EXP_DATE	date(10)	
QUANTITY	integer(10)	
UNIT_PURCHASE_PRICE	double(10)	
UNIT_SELL_PRICE	double(10)	
 <a href="#">Add field</a>		

CUSTOMERS		
Table stores information about CUSTOMERS		
 CUSTOMER_ID	numeric(10)	
NAME	varchar(100)	
EMAIL_ADDRESS	mediumtext(20)	
PHONE_NUMBER	integer(10)	

Suppliers table consists of an auto incremented supplier\_id and other columns are inserted through the retailer.

Products table consists of an auto incremented product\_id and the product name is inserted by the retailer. It also consists of a column called description that is not visible in the diagram.

Inventory table is the stock table. It consists of an inventory\_id which is auto incremented along with batch numbers, mfg date, exp date, quantity, purchase and selling price. All these are provided by the retailer. The product\_id and supplier\_id are foreign keys that are inserted as per the supplier and product selected by the user in the dropdown menu. In case of the new product, the insertion first happens in the products table and then the product\_id is used.

Customer - This table just contains the basic customer details and an auto incremented customer\_id is generated everytime a new customer is added.

Orders - The order table consists of the order id which is auto incremented along with date, which i mentioned as the timestamp, the status of the order, the price of the order and the customer number which is a foreign key referenced from the customers table.

Order\_det - This table was solely created to make the order table look less fussy. This consists of an auto incremented order\_det\_id that is of no use the order id is referenced from the order table similar to the product\_id. It displays the quantity and the batch number of the products that are ordered.

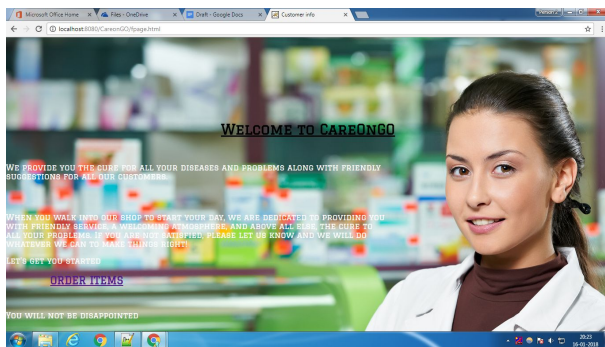
*This database acts as the heart of the project.*

What are the files and what do they do ?

Now that i have already described how the project works, let me just tell you about what each file's purpose is in brief.

## **HTML files -**

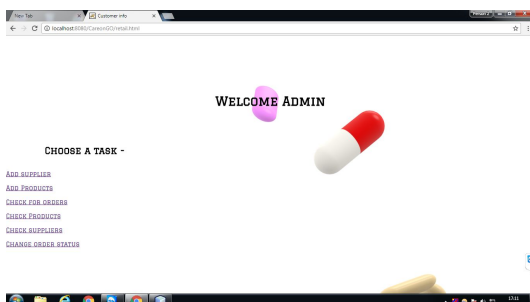
**fpage.html** - This is the first page that the customer looks at.



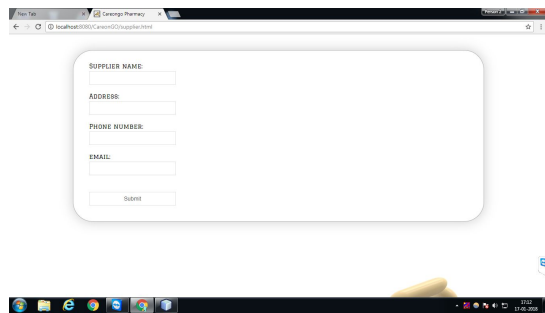
**index.html** - This is the signup or the login page.

A screenshot of a web browser displaying the 'index.html' file. The page has a white background with a red and white pill graphic. At the top, it says 'WELCOME CUSTOMER, PLEASE ENTER THE FOLLOWING DETAILS TO CONTINUE' and 'Sign Up'. Below this is a registration form with fields for 'NAME', 'ADDRESS', 'PHONE NUMBER', 'EMAIL', 'USERNAME & PASSWORD', and 'CONFIRM PASSWORD'. A 'Register' button is at the bottom. Below the registration form is a login section with fields for 'USER USERNAME' and 'PASSWORD', and a 'Login' button.

**retail.html** - This is the first page that the retailer looks at that displays him the menu.



**supplier.html** - This page is for the adding the supplier details.

A screenshot of a web browser window displaying a form titled 'supplier.html'. The form is enclosed in a rounded rectangle with a thin border. It contains five input fields: 'SUPPLIER NAME', 'ADDRESS', 'PHONE NUMBER', 'EMAIL', and a 'Submit' button at the bottom. The browser's address bar shows the URL 'localhost:5500/Supplier.html'. The Windows taskbar is visible at the bottom of the screen.

## **CSS pages -**

**Style.css** - This css style page is the only css page and is used has all the properties defined. For the body of all my html pages, I have used a font called Graduate. It really brings out the texts. For the form in my html pages, the CSS page has defined a small solid border along with auto margins on both left and right direction. The radius is 50px for the border. For all the inputs that I am taking, the CSS file adds some spacing , and puts it all in a border with a constant font size.

The input button changes color on hovering so for that I have set the background as #eee. I don't really know the colour codes. For the submit option that you will be seeing in the later pages, my text is always uppercase with a different background colour. The width is the same for all submit boxes. It also has the hover option and the colour lightens a bit. I have made some minor changes for paragraphs, captions and headings by adding some padding and font size for paragraph.

## **Java servlets -**

**Login1** - This servlet does the processing for the login details that are received during the login time and displays the order and tracking option.

**Afterlogin** - This servlet basically captures the order id if the tracking is selected .

**Custrack** - This servlet basically displays the order ids linked with the customer id

**Tracking** - This servlet is to provide the customer the order status.

**Custcat** - This is where the customer can see all the products after clicking order. It is also pointed to after the sign up

**NewServlet1** - This is the servlet where all the needed quantities and product ids are inserted into the order table.

**NewServlet2** - This servlet basically updates the quantity in the inventory table and assigns batch numbers to the order\_det table.

**Custinsert** - This servlet runs right after sign up page and inserts the customer details to customers table.

**supplierdisplay** - This servlet is used to display the suppliers in the suppliers table

**Supinsert** - This servlet is used to insert the supplier details in suppliers after supplier.html takes the values.

**Dispusp** - This servlet displays the products when the retailer clicks check products.

**Inventinsert** - This is used to add the products to the inventory. This has the html page running to display the radio button.

**Inventins** - This is the postprocessing page of inventinsert where the actions as per the radio buttons takes place

**Alert** - This servlet is used to display the orders that have the status as Ordered

**Processal** - This is used to process the orders and make them as shipped.

These are all the servlets that were used in the coding.

One of the major issues that arises while trying to host these pages is the lack of an environment for these pages i.e a server. So we need a server and that is where Apache Tomcat comes to play. I have used Tomcat 9 for this.

What is apache tomcat ? It is an open source web server tool that basically allows the implementation of Java Servlets and Java Server Pages to promote a Java friendly environment.

I have used Netbeans to create my project which is nothing but an IDE so as to keep my pages and programs all under one project.

So whenever I run the project, the browser opens  
localhost:8080/Careongo/filename

Here localhost can be replaced by my IP address and 8080 is  
the port. Careongo is the name of the project.