Criterion A: Planning

The Client & Outline of Problem

My Client is the owner of a jewellery business whose role is primarily focused on taking orders from clients and creating new designs which her suppliers can manufacture. She faces the problem of managing her orders and inventory effectively as her jewellery suppliers and production all take place in India. The difficulties of effectively managing all current orders without having to start over with certain details arise. For example, if the price and design of a specific product are not stored properly, the data will have to be recalculated and designed from scratch each time. This leads her to feel disorganised and affects the efficiency of the entire process of making jewellery for her clients. My client feels this is a big problem as it may also cause her to make mistakes with her client's orders and details, and so she may have to re-do orders, causing her an extra cost and increased production time.

Product Justification

Firstly, building the system is important because it would benefit the client to a large extent. This is because she has a large number of clients and orders which are ongoing at a time. Because of this, it is difficult to manage all orders at once while also taking into account the variety of key information needed for an order without an organised system. This system would help her because it would reduce her workload as she can focus more on expanding her market and designing new pieces of jewellery **(see Appendix, Interview 1, See Q3)**. After trying multiple existing products my client felt none of them worked for what she needed, and felt it was more work to use those systems, therefore she reverted to using a very scattered method. **(see Appendix, Interview 1, See Q2**).

Software Justification

Tool	**Feature**	**Reason for use**
Java	Highly Portable Language	Makes it easy for my client to use as it is accessible on her laptop **(see Appendix, Interview 1, See Q3)**
	Small Memory Footprint	It takes up less space and storage on my clients CPU while running the system, and makes it more efficient for her to run.
	Large variety of accessible libraries	Writing code is more efficient because of reusability. This time can instead be spent on solving other problems which may require more focus.
Java Swing	Very lightweight	The features and syntax are minimal, such as the GUI building tools. Therefore it is more convenient to use when building the system to match the clients requirements.

	Uses platform independent resources	This helps to make sure that if my client runs the system on a different computer, the system will look and run in the same manner.
Netbeans IDE	Drag-and-drop GUI builder	Easy to design and change GUI and coding behind it, based on clients specific needs and feedback.
MySQL	Relational Database System	The clients data can be easily organised and catagorised. Furthemore, it is easy to change and access the items which are stored in the database.
	Uses SQL	Stores large amounts of data and retrieves it from a database extremely efficiently. Matches clients requirements **(see Appendix, Interview 1, See Q4).**

Word count: 515

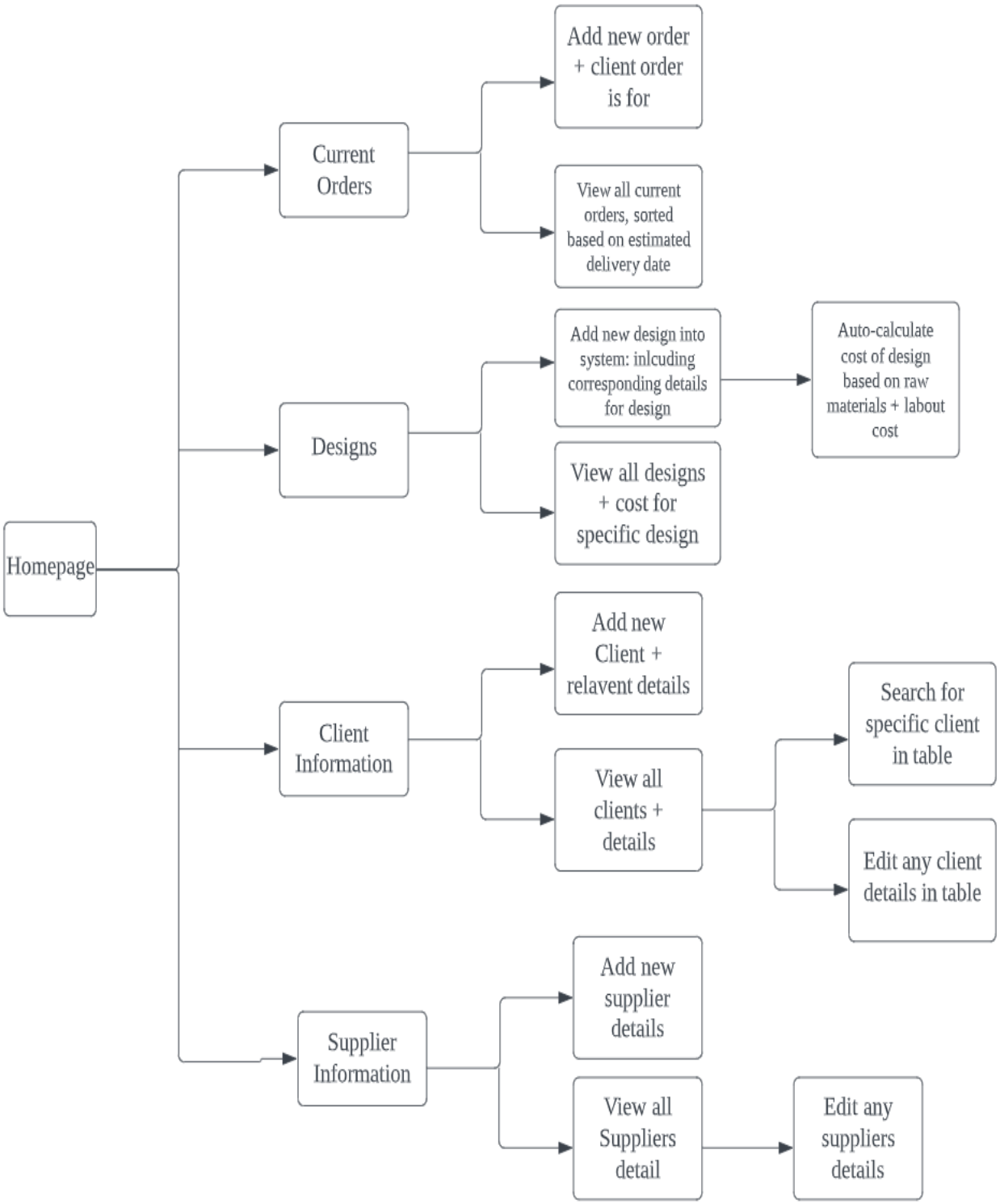
Criteria for Success

1. Allow user to login to system with username and password
2. Allow user to manage client information in the system
    a. User can add client details with corresponding sizes
    b. User can view and edit client details
3. Allow user to add new order with all relevant details needed for the order
    a. Auto-generated price linked to design of piece and quantity of pieces
    b. User can see expected delivery date based on design
4. Allow user to view all current orders
    a. User can view order details including client and piece in the order
5. Sort all current orders based on priority of estimated order delivery
6. Allow user to view whether payment is completed in order to begin order
7. Allow user to search for specific items in the system
    a. User can search for specific clients
    b. User can search for specific designs
    c. User can search for specific orders
8. Allow user to input designs and information regarding design (type, raw materials, labour cost)
    a. Auto-generate total cost for design based on raw materials and labour cost
    b. User can view all designs in system they have previously entered
    c. User can edit all designs in system they have previously entered
9. Allow users to view supplier information including contact detail and role
    a. Allow users to add new supplier information
10. Each order has an automated order number
    a. When a client is added into the system a serial number is automatically generated for that client
    b. When a design is added into the system a serial number is automatically generated for that design
11. Order will delete from table once client has completed the order
12. User can see a text file which has the names and order details for clients who have not paid
13. User can see their top clients, and the top clients most ordered 3 designs.

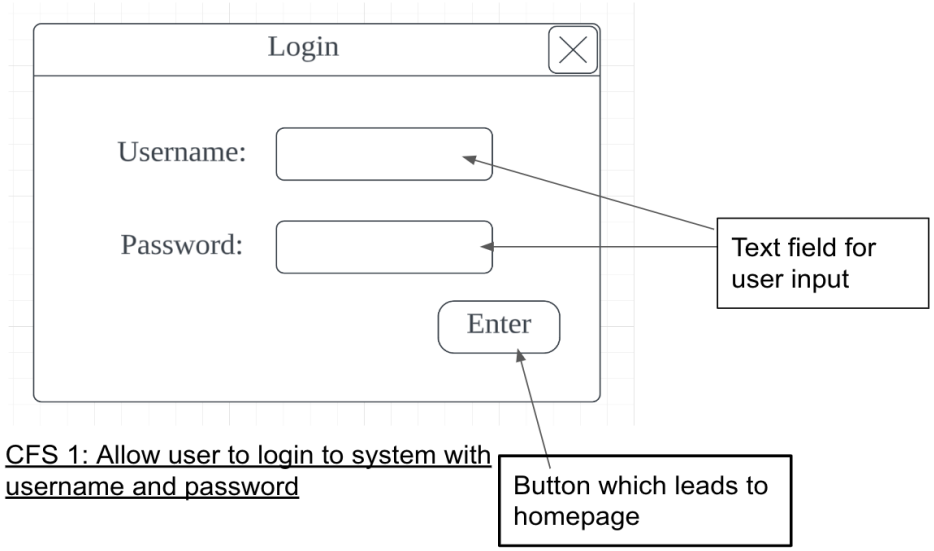
Criterion B: Design

Table of Contents

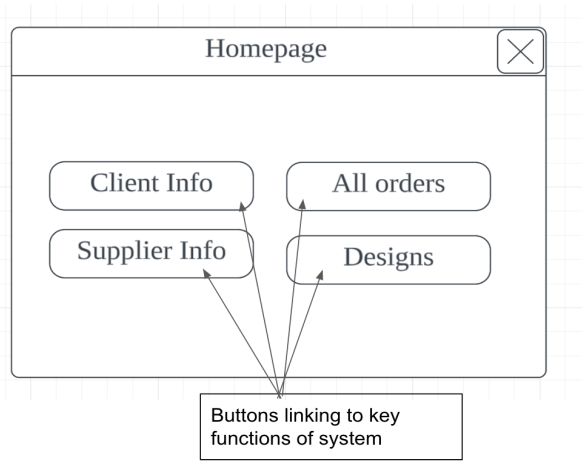
System: Overview, Structured Diagram



GUI: Screen Design

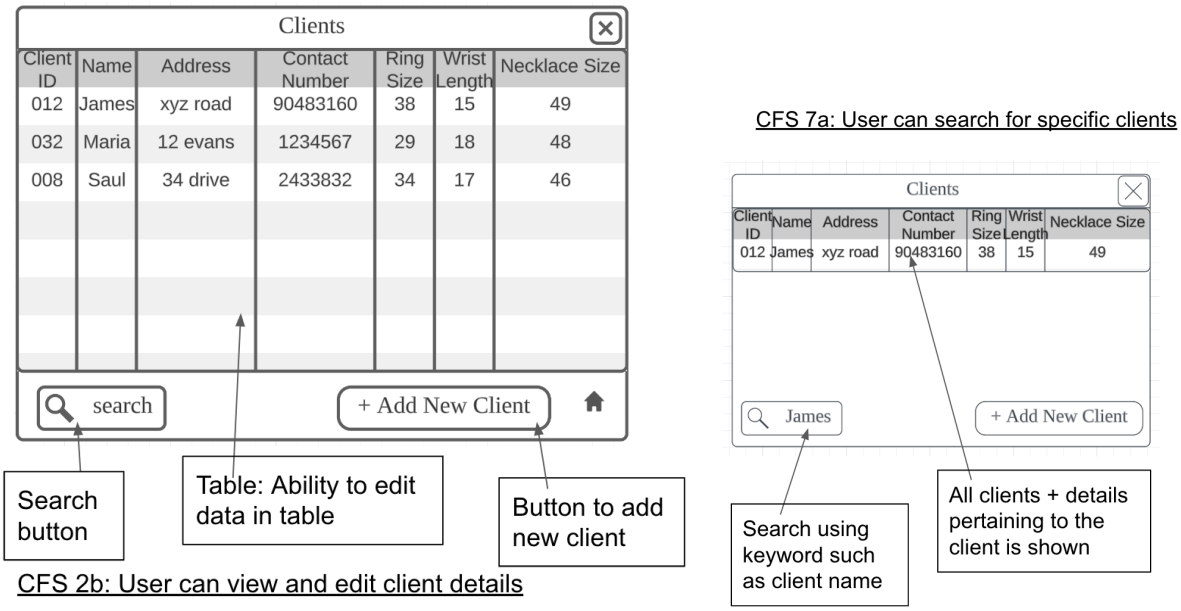


This page shows it allows the users information regarding her orders,suppliers, clients and designs to be confidential and only accessible by her.

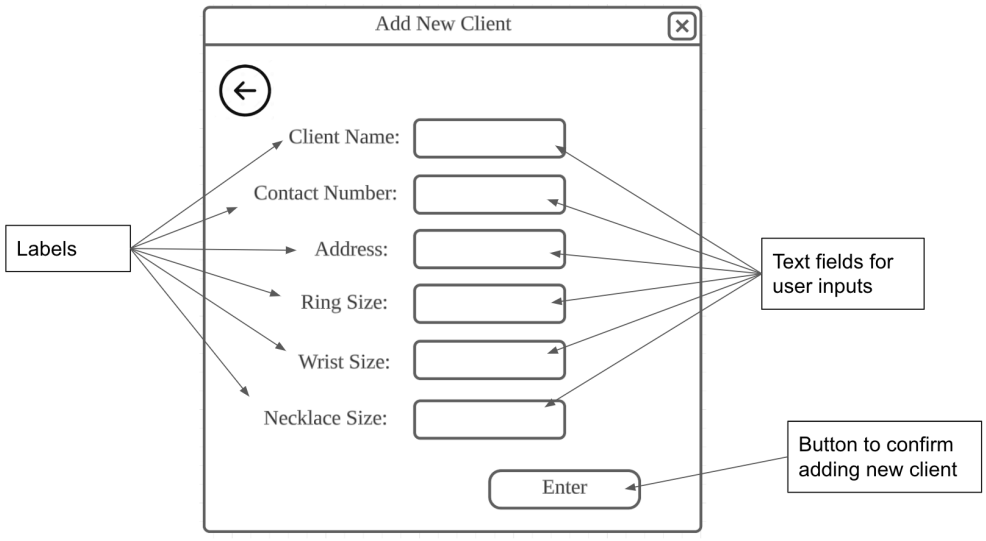


The minimalistic design of the homepage makes it easy for the user to identity what function they need to access in the system.

Client Information GUI Screens:

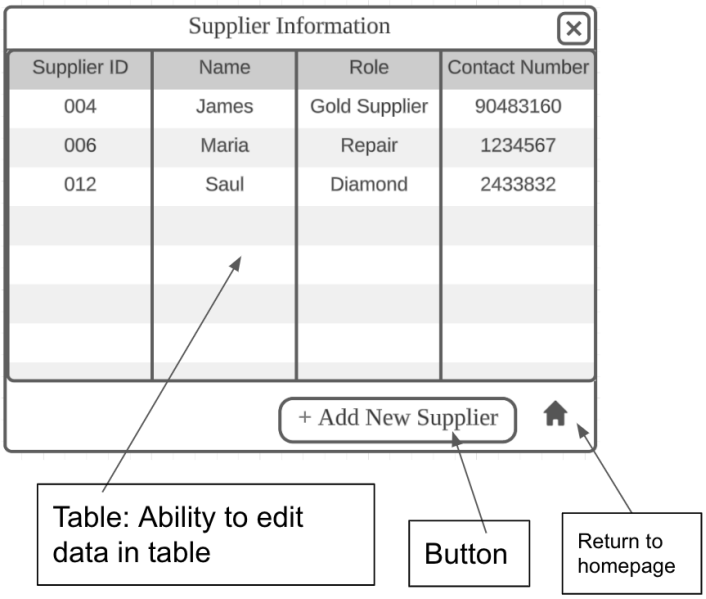


The screen also demonstrates criteria 10a, which presents all success criteria in an organised manner, where they are able to edit information through the table itself. Additionally, they can also add new client information, or search for specific information based on what they are looking for.



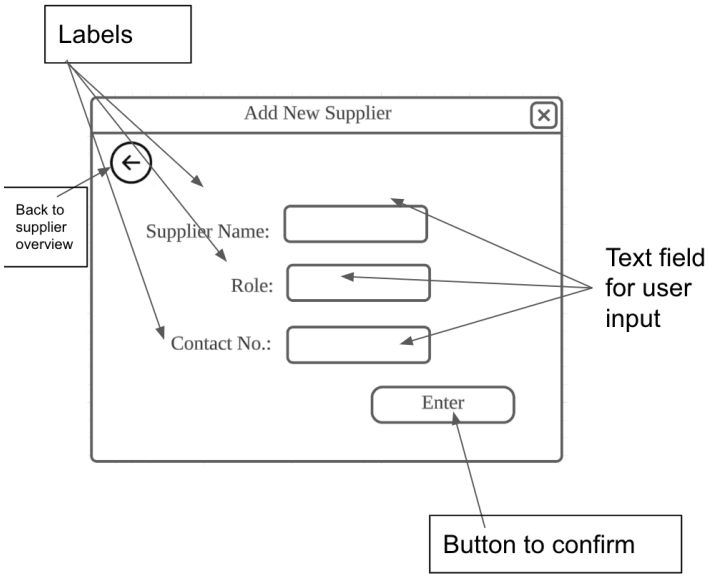
CFS 2a: User can add client details with corresponding sizes

Supplier Information GUI Screens:



CFS 9: Allow users to view and edit supplier details including contact detail and role

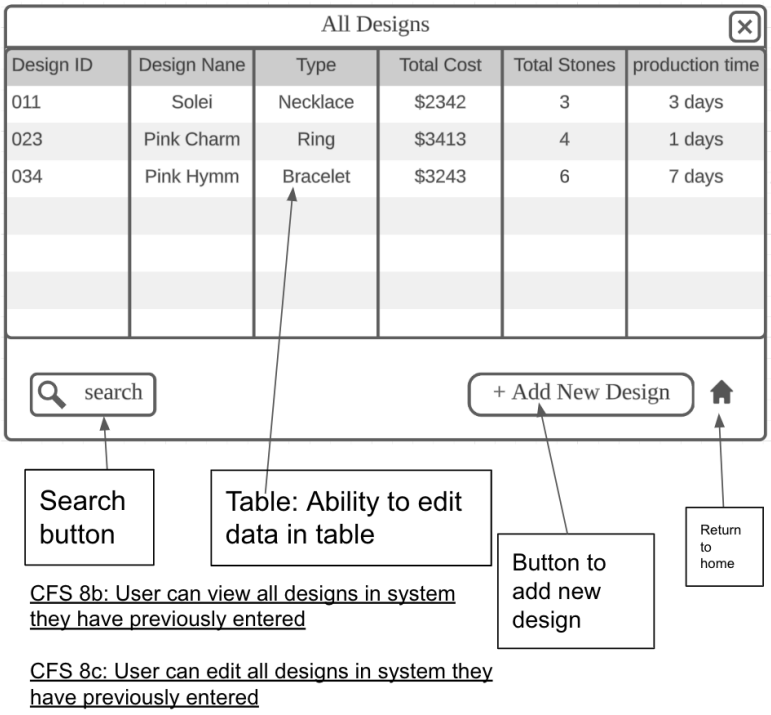
Simiilarly to the client section, the supplier information presents details of each supplier in an organised manner which is simplistic for the client. The homebutton has been added to help the user **(see Appendix, Interview 2)**



CFS 9a: Allow users to add new supplier information

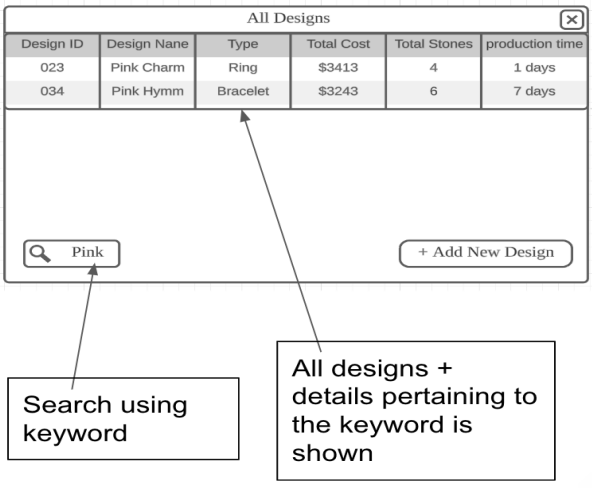
The user then has the ability to add new suppliers which will be presented in the table and add the relevant details which refer to the suppliers.

Designs GUI Screens:



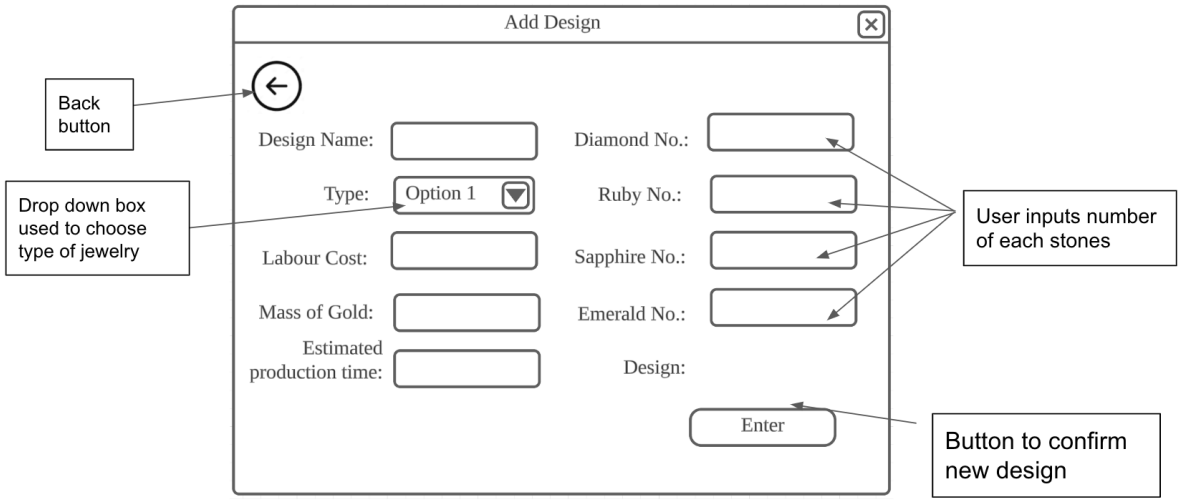
The screen provides an organised manner to view important details regarding each jewelry design.

CFS 7b: User can search for specific designs



A search function is also available to make it easy for the user to find specific designs they are looking for/need to edit.

CFS 8: Allow user to input designs and information regarding design (type, raw materials, labour cost)

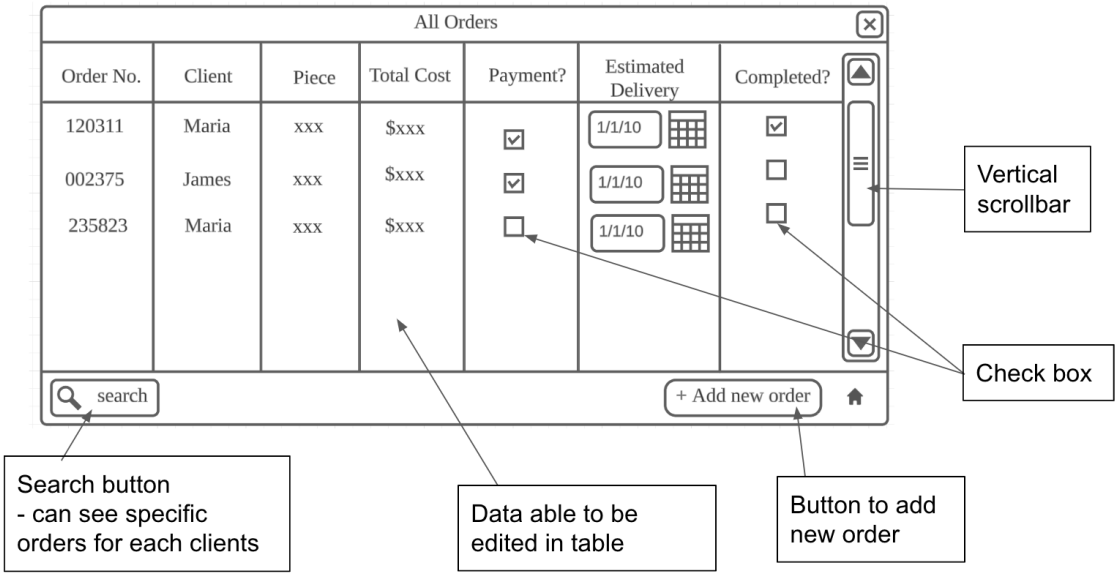


This provides a very easy way for the client to enter all raw materials and information regarding the design. From there the cost of the piece can be auto-calculated, achieving criteria 8a.

Orders GUI Screens:

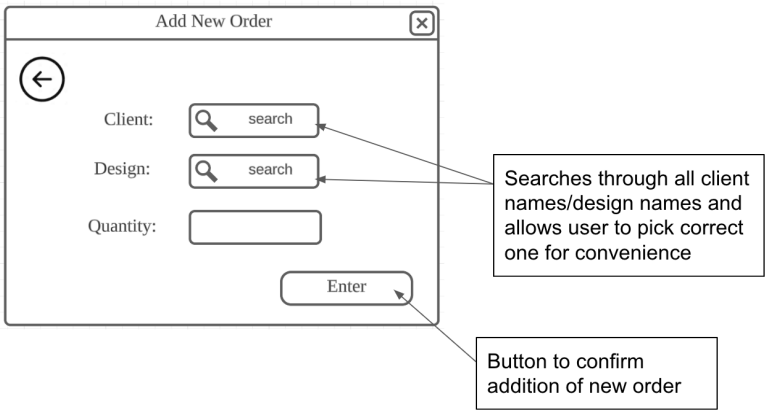
CFS 4: Allow user to view all current orders
CFS 4a: User can view order details including client and piece in the order
CFS 4b:User can edit any information regarding orders

CFS 3a: Auto-generated price linked to design of piece and quantity of pieces
CFS 3b: User can see expected delivery date based on design
CFS 5: Sort all current orders based on priority of estimated order delivery
CFS 6:Allow user to view whether payment is completed in order to begin order

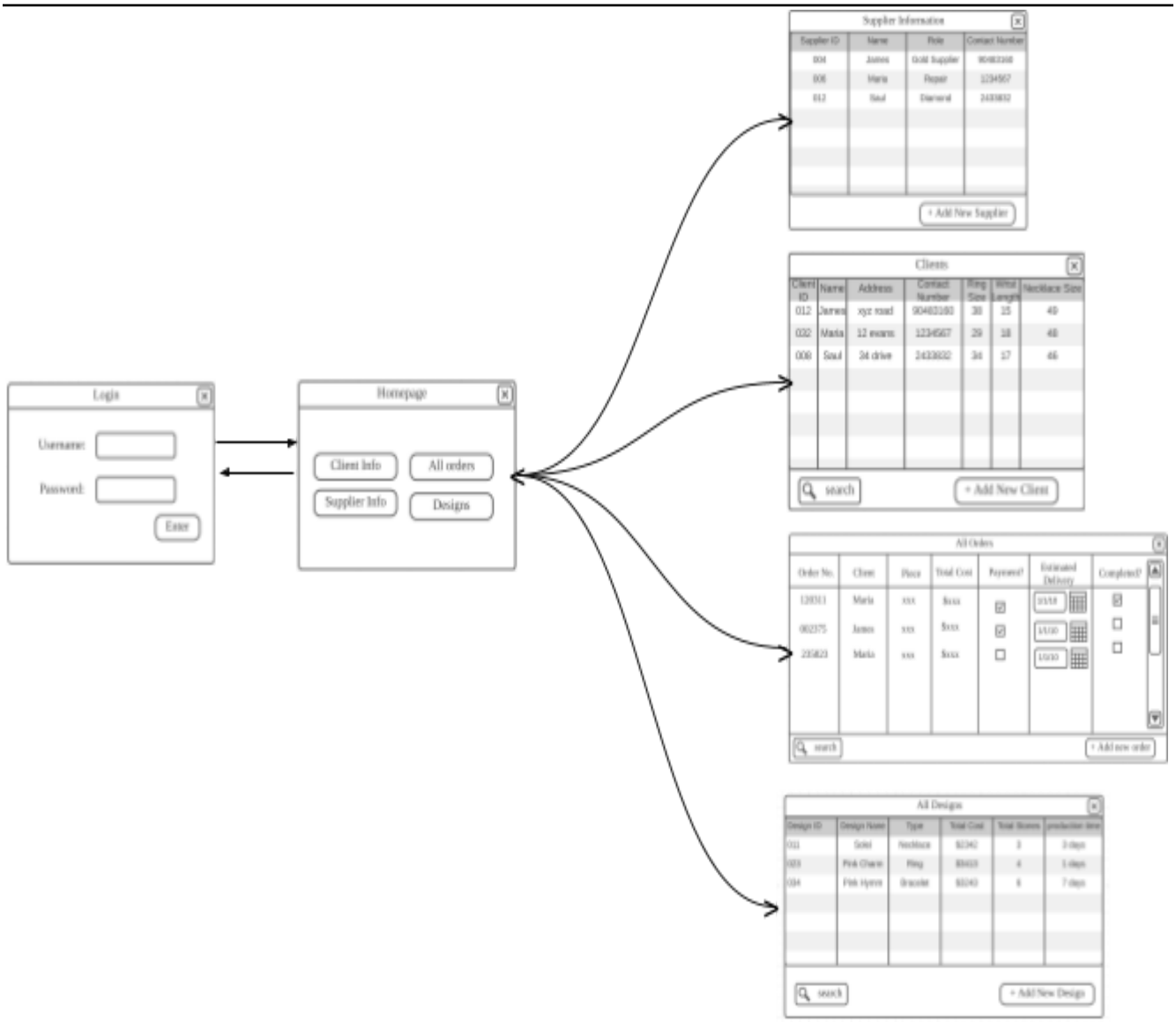


This screen is the main overview of all orders in the system in which the user currently needs to see. It sorts it based on the earliest estimated delivery and if payment is completed. Once the order is completed the user can easily tick the box to make it more organised. The user can then also add any new orders and the client/design/quantity the order contains.

CFS 3: Allow user to add new order with all relevant details needed for the order



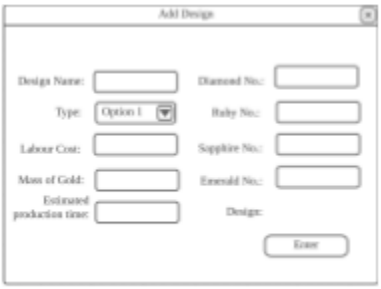
GUI: Navigation, Links & Connections

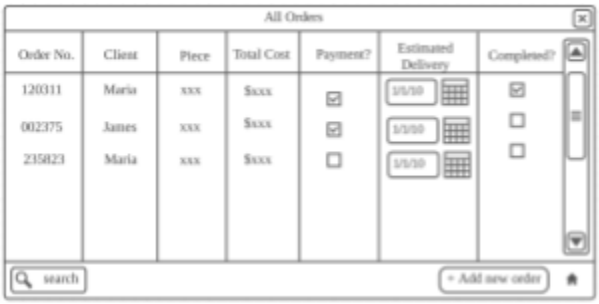


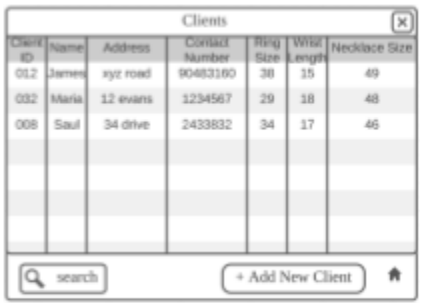

Supplier Information
Supplier ID
Name
Role
Contact Number
004
James
Gold Supplier
90483160
006
Maria
Repair
1234567
012
Saul
Diamond
2433832
+ Add New Supplier


All Designs
Design ID
Design Name
Type
Total Cost
Total Stones
production time
011
Solei
Necklace
$2342
3
3 days
023
Pink Charm
Ring
$3413
4
1 days
034
Pink Hymn
Bracelet
$3243
6
7 days
search
+ Add New Design


Add New Supplier
Supplier Name:
Role:
Contact No.:
Enter


Add Design
Design Name:
Type:
Option 1
Labour Cost:
Mass of Gold:
Estimated production time:
Diamond No.:
Ruby No.:
Sapphire No.:
Emerald No.:
Design:
Enter

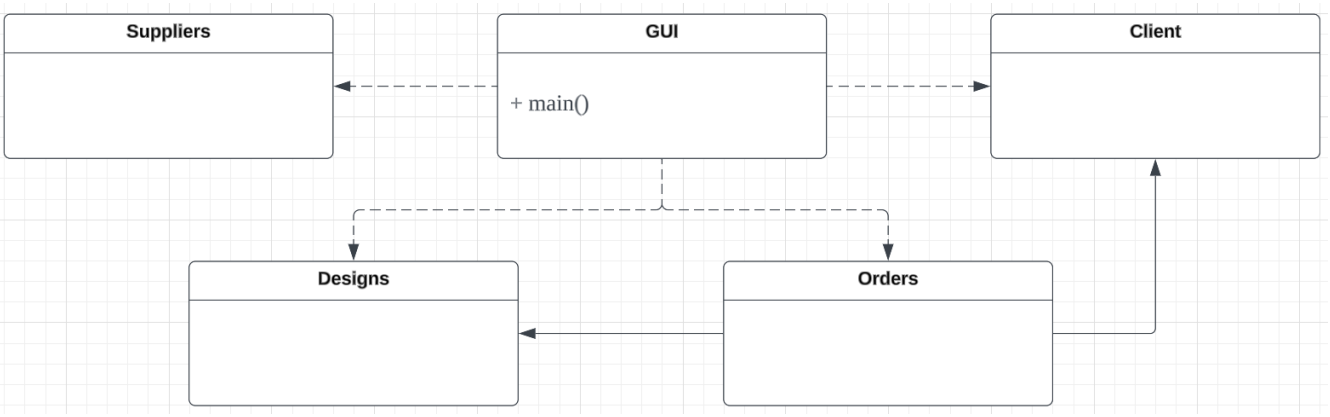

All Orders
Order No.
Client
Piece
Total Cost
Payment?
Estimated Delivery
Completed?
120311
Maria
xxx
$xxx
1/5/10
002375
James
xxx
$xxx
1/3/10
235823
Maria
xxx
$xxx
1/3/10
search
+ Add new order


Clients
Client ID
Name
Address
Contact Number
Ring Size
Wrist Length
Necklace Size
012
James
xyz road
90483160
38
15
49
032
Maria
12 evans
1234567
29
18
48
008
Saul
34 drive
2433832
34
17
46
search
+ Add New Client


Add New Order
Client:
search
Design:
search
Quantity:
Enter

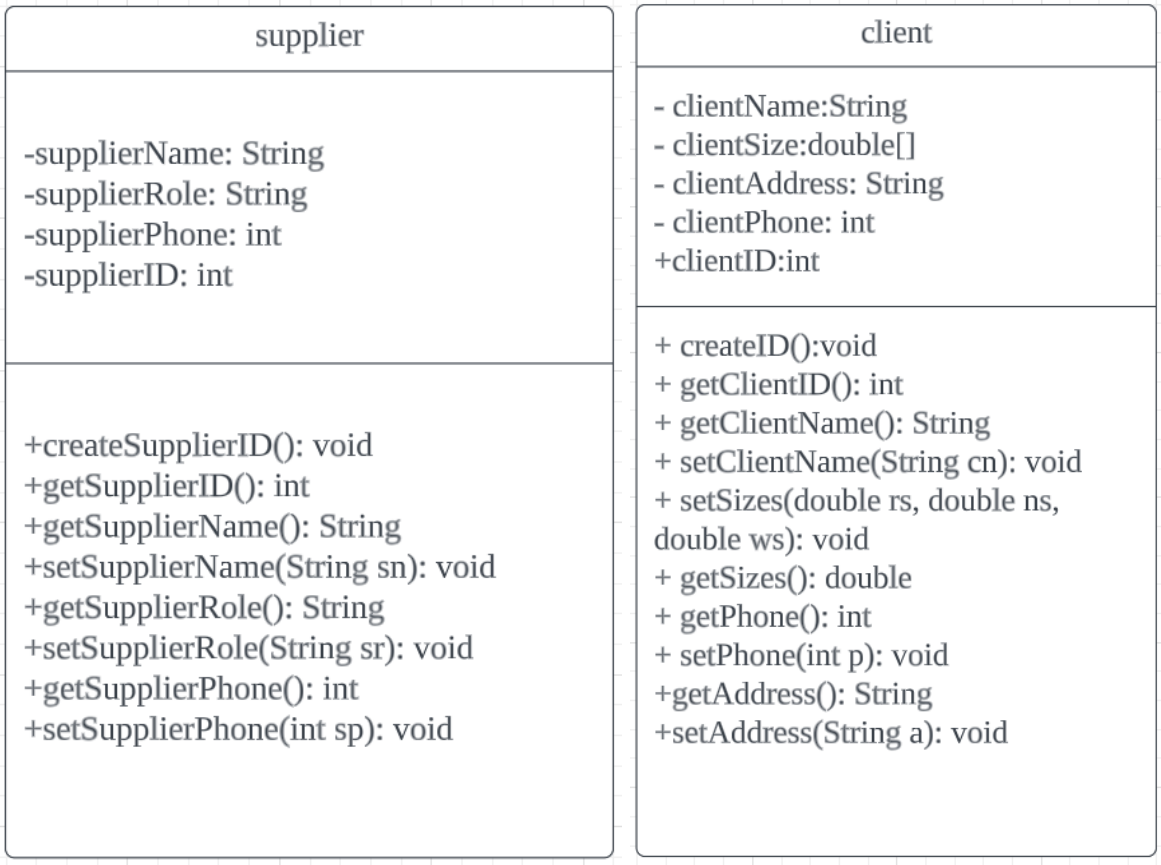

Add New Client
Client Name:
Contact Number:
Address:
Ring Size:
Wrist Size:
Necklace Size:
Enter

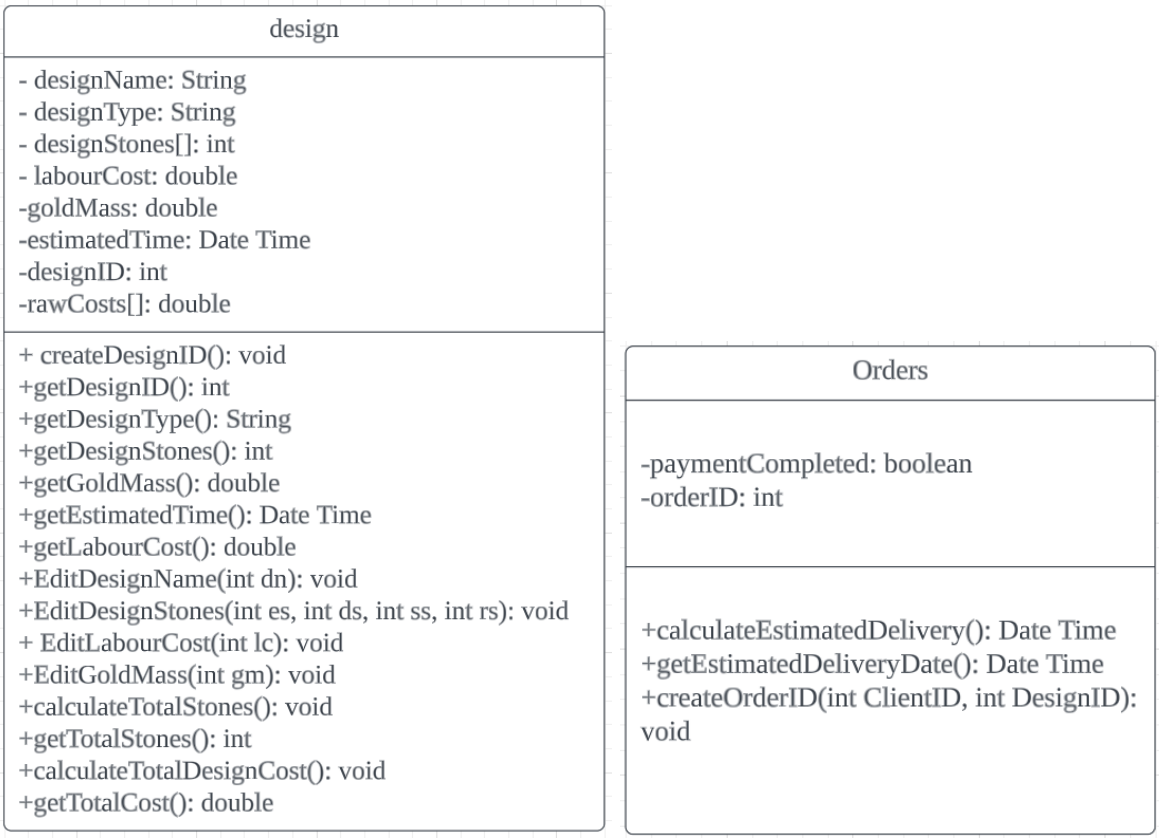
UML class relationships



The orders class is a child class to both the client class and designs class, and so inherits the attributes and methods of those classes.

UML Classes for supplier, client, object, and designs which can be instantiated in the main GUI class.





Internal Structures: Other Fields, Structures, Arrays

Structure Name	**Structure Function**
Raw Costs	- 1D Array - Contains set prices for different types of stones, and set price of stone - Last index contains set price for gold
Table Values	- 2D ArrayList - Holds all results for a database query that needs to be presented on a jTable for the user to see - Use of array list so the size does not need to be predetermined

File Structures: Database Design, E-R Diagrams

CLIENTS		
Field Name	**Data Type**	**Description**
ClientID	Integer	Primary Key, This will be the unique identifier for each item in the

		database
ClientName	Text	This field will hold the clients full name
ringSize	Double	This field will hold the client's ring size which can be accessed each time the client orders a ring
neckSize	Double	This field will hold the client's neck size which can be accessed each time the client orders a necklace
wristSize	Double	This field will hold the client's wrist size which can be accessed each time the client orders a bracelet
clientAddress	Text	This field will hold the client's address which can be used to deliver their order to the client
ContactNum	Integer	This field will hold the client's contact number

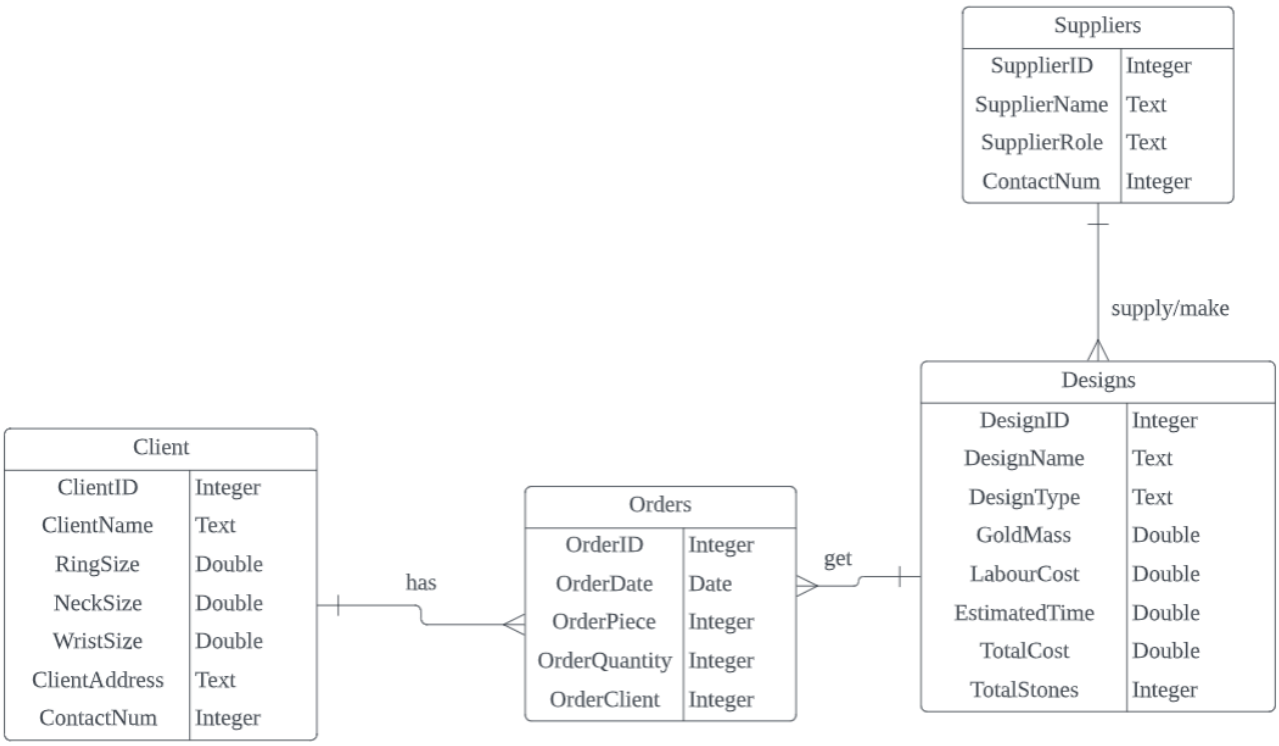
SUPPLIER		
Field Name	**Data Type**	**Description**
SupplierID	Integer	Primary key, which is the unique identifier for the supplier.
supplierName	Text	This field will hold the supplier's name
supplierRole	Text	This field will hold the role of the supplier, e.g “Diamond Supplier”
contactNumber	Integer	This field will hold the contact number of the supplier

DESIGNS		
Field Name	**Data Type**	**Description**
DesignID	Integer	Primary key, which is the unique identifier for the supplier.
DesignName	Text	This field will hold the given name for the jewelry design
DesignType	Text	This field will hold what type of jewellery the design is. E.g “Necklace”, “Earring”
goldMass	Double	This field will hold the grams of gold used to make the jewelry piece
labourCost	Double	This field will hold the labour cost for making the jewelry piece
estimatedTime	Integer	The field will hold the approximate number of days for the jewelry piece to be made

totalStones	Integer	This field will hold the total number of stones the design has.
totalCost	Double	This field will hold the total cost of the design.

ORDERS		
Field Name	**Data Type**	**Description**
OrderID	Integer	Primary key, which is the unique identifier for the order.
OrderDate	Date	This field will hold the date the order is estimated to be done
OrderPiece	Integer	This field will hold the serial number of the design of the piece ordered
OrderQuantity	Integer	This field will hold the number of the pieces ordered
OrderClient	Integer	This field will hold the serial number of the client the order is for

E-R Diagram



Data: Security & Validation

Data Reasoning	Validation Rule	Reasoning
ID	Not empty	Every client/supplier/design/order needs to have an idea so it can be found in the database
Name	Not empty	Every item should have a name for identification purposes in the database.
Type	Not empty	Every design should have a type specified in the database.
Contact Number	Not empty	Every client/supplier name should have a contact number associated with it.
Size	Not empty	For every client, they must have each size associated with it
Stones Number	At least one size not empty	For the three size data items, at least one of the three sizes must not be empty.

Algorithms: Pseudocode, Flowcharts & Logic

CFS 1: Allow user to login to system with username and password

```
boolean logged_in = FALSE
function login(String username, String password)
      while (username = !valid_username AND password =! valid_password) Do
            output "Invalid username or password."
       end while
      logged_in = TRUE
      output “Successfully logged in”
end function
```

CFS 10: Each order has an automated order number which includes serial number for client and design

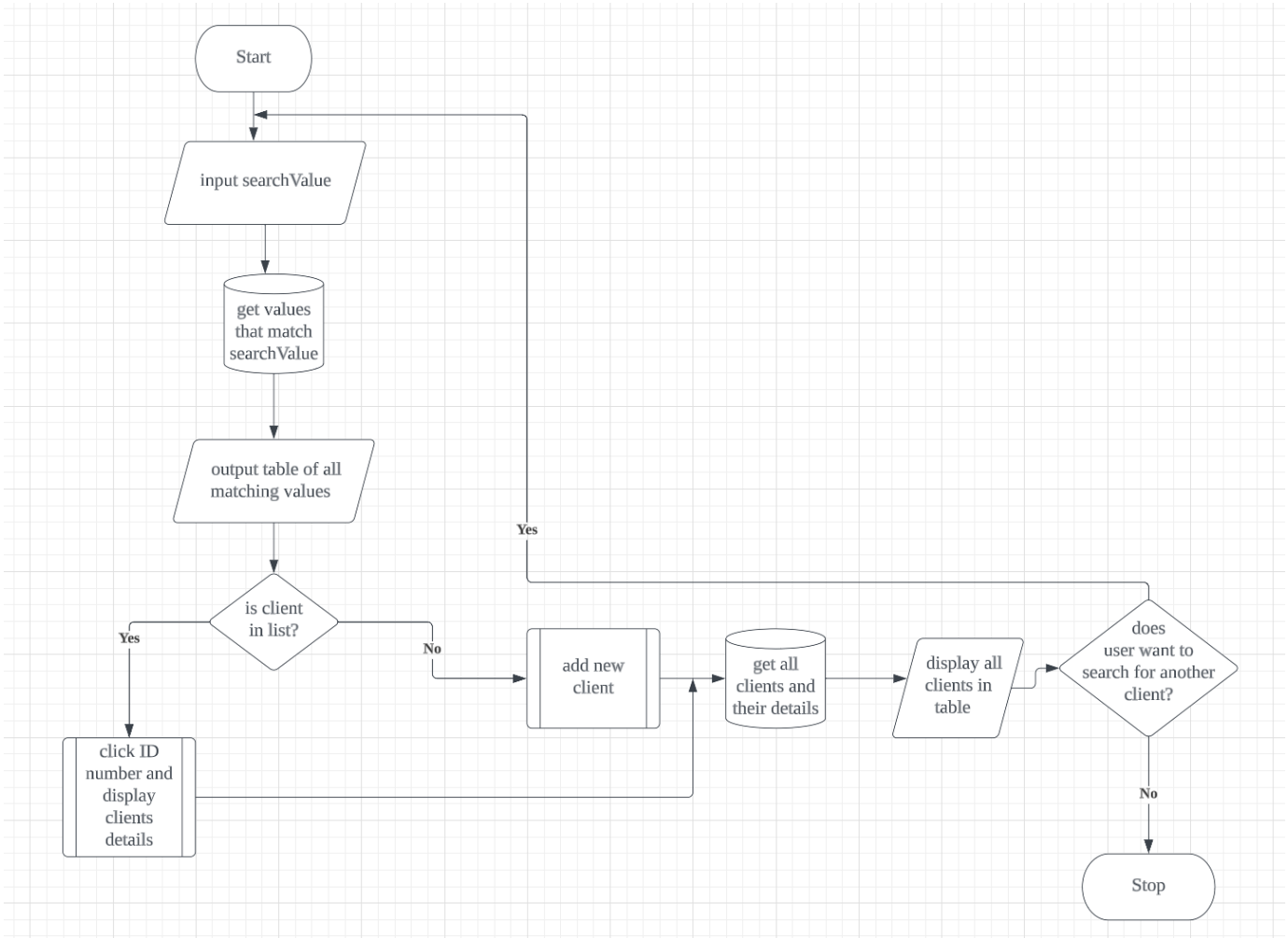
```
//this creates the clientID and stores it in clientDetails
function createClientID()
      clientID = clientID + 1
```

```
      loop while clientDetails[count] != null
            count = count + 1
      end loop
      clientDetails[count].setClientID(int clientID)
      addNewClient(clientObj)
      //the details for the new client are then added, and the clientID is
      used to access the position of the object in the arrayList of
      clientDetails to set the details into the object
end function

//this creates the designID and stores it in designDetails
function createDesignID()
      DesignID = DesignID + 1
      loop while DesignDetails[count] != null
            count = count + 1
      end loop
      DesignDetails[count].setDesignID(int DesignID)
      addNewDesign(DesignObj)
      //the details for the new designs are then added, and the designID
      is used to access the position of the object in the arrayList of
      designDetails to set the details into the object
end function

function createOrderID()
      addNewOrder(OrderObj)
      orderID = orderID + 1
      orderID = orderID + orderDetails[count].getOrderClient +
orderDetails[count].getOrderPiece
//position for count is already determined when new order object is
insantiated
      //accesses the linked field
      orderDetails[count].setOrderID(int orderID)
```

CFS 8a: Auto-generate total cost for design based on raw materials and labour cost

```
db = CONNECT(database)
function calculateTotalCost()
      totalCost = labourCost + (goldMass * rawCosts[-1])
      loop count from 0 to designStones.length
            stonesCost = (designStones[count] * rawCost[count]) + totalCost
      end loop
      totalCost = totalCost + stonesCost
end function
sql = "INSERT INTO DESIGNS (TotalCost) VALUES ("totalCost") WHERE
DesignID = dID;"
doSQL(sql,db)
```

CFS 7a: Allow user to search for specific clients



CFS 2a: User can add client details into system

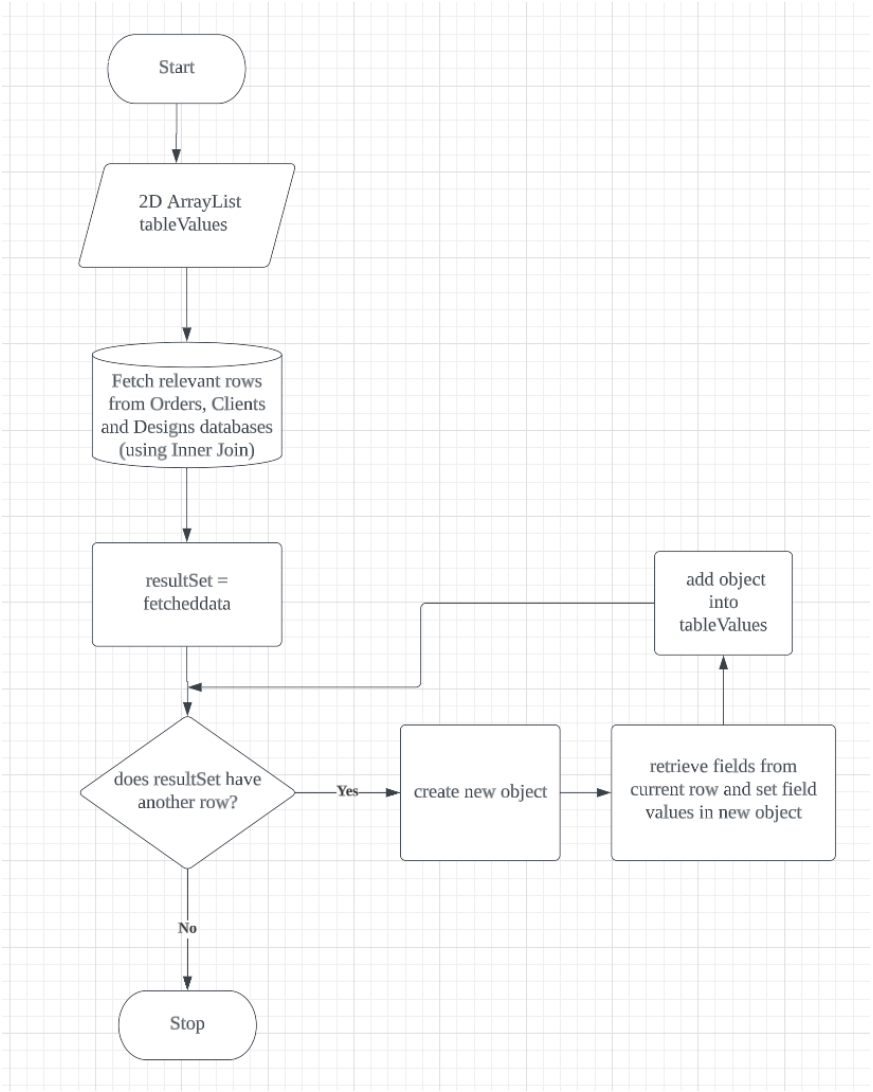
```
//checking if the new client is a duplicate
sameClient = false
loop count from 0 to clientDetails.length
     if clientName = clientDetails[count].getName() AND clientPhone =
clientDetails[count].getPhone() then
     sameClient = true
     output "Client has already been added to system"
     else
//check where the next object of client information can be added in list
          loop while clientDetails[count] != null
               count = count + 1
          end loop
          clientDetails[count].addNewClient(clientObj)
          sql = "INSERT INTO CLIENTS (ClientID, ClientName, ringSize,
          NeckSize, WristSize, clientAddress, ContactNum) VALUES (cID,
          cName, rSize, nSize, wSize, cAddress, cNum);"
          doSQL(sql,db)
     end if
end loop
```

CFS 3b: User can see expected delivery date based on design

```
currentDate = get currentDate(DD-MM-YYY)
function calculateEstimatedDelivery(currentDate)
     loop i from 0 TO designs.length
          if (designs[i].getID() == designID)then
               estimatedTime = designs[i].getEstimatedTime()
               estimatedDate = addDays(currentDate, estimatedTime)
          endif
     end loop
     return estimatedDate
end function

function addDays(date, days)
     dateObject = parseDate(date, "DD-MM-YYYY") // converts date into a
```

```
formate that can be manipulated
     estimatedDateObject = addDaystoDate(dateObject, days) //adds
estimated days to current date
     estimatedDate = formateDate(estimatedDateObject, "DD-MM-YYYY")
//changes date into correct format to display as DD-MM-YYYY
     return estimatedDate
end function

db = CONNECT(database)
sql = "INSERT INTO ORDERS (OrderDate) VALUES (estimatedDate) WHERE
OrderID = oID;"
doSQL(sql,db)
```

CFS 4a: Allow users to view all orders (queried from database row by row)



CFS 3b: Reminder message for an order that is not completed past estimated delivery date

```
db = CONNECT(database)
currentDate = get current Date
sql = "SELECT OrderDate FROM Orders WHERE OrderID = oID;"
estimatedDeliveryDate = doSQL(sql,db)
if currentDate > estimatedDeliveryDate then
      daysOverdue = calculateDaysLate(currentDate, estimatedDeliveryDate)
      output "This order is" + daysOverdue + "overdue"
end if
function calculateDaysLate(currentDate, estimatedDate)
      overdue = estimatedDate - currentDate
      covertToDays(overdue)
      return overdue
end function
```

SQL: Data Interrogation & Management

CFS	Function	SQL Query
2a. User can add client details with corresponding sizes	Add details for new user into clients database.	`INSERT INTO CLIENTS (ClientID, ClientName, ringSize,NeckSize,WristSize, clientAddress, ContactNum) VALUES (cID, cName, rSize, nSize, wSize, cAddress, cNum);`
2b. User can view and edit client details	Be able to view all client details stored in database, and select all details from the clients database.	`SELECT * FROM clients;`
	User can edit any specific field in database E.g editing client name	`UPDATE CLIENTS SET ClientName = data WHERE ClientID = cID;`

3. Allow user to add new order with all relevant details needed for the order	Insert order details in to orders table.	`INSERT INTO ORDERS (OrderID, OrderDate, OrderPiece, OrderQuantity) VALUES (oID, oDate, oPiece, oQuantity);`

4. Allow user to view all current orders including client who placed order, and total cost.
 	
5. Sort all current orders based on priority of estimated order delivery
	Be able to view all order details and order by date of delivery from nearest date to furthest date	`SELECT`
		`  Orders.OrderID, Orders.OrderDate, Orders.OrderPiece, Orders.OrderQuantity`
		`  Clients.ClientName, Designs.name`
		`FROM Orders`
		`JOIN Orders`
		`  ON Client.ClientID  = Orders.OrderClient`
		`JOIN Designs`
		`  ON Designs.DesignID = Orders.OrderPiece`
		`ORDER BY OrderDate ASC;`

7a. User can search for specific clients by name	Be able to find specific clients easily	`SELECT * FROM CLIENTS` `WHERE ClientName = data;`
7b. User can search for specific designs by name	Be able to find specific designs and their details easily	`SELECT * FROM DESIGNS` `WHERE DesignName = data;`
7c. User can search for specific orders by client	Be able to find what orders one client has made	`SELECT * FROM ORDERS WHERE ClientName = data;`
8. Allow user to input designs and information regarding design (type, raw materials, labour cost)	Be able to store all data corresponding to a new design	`INSERT INTO DESIGNS (DesignID, DesignName, DesignType, GoldMass, LabourCost, EstimatedTime, TotalCost, TotalStones)` `VALUES (dID, dName, dType, gMass, lCost, eTime, tCost, tStones)`
8b. User can view all designs in system they have previously entered	Be able to view all designs, and see all corresponding information	`SELECT * FROM DESIGNS;`

8c. User can edit all designs in system they have previously entered	Be able to edit different details in the design database E.g design name	`UPDATE DESIGNS`
		`SET DesignName = data`
		`WHERE designID = dID;`

9. Allow users to view	Be able to view all	`SELECT * FROM SUPPLIERS;`

supplier information including contact detail and role	supplier information	
9a. Allow users to add new supplier information	Be able to add new suppliers and their corresponding information into the databse	`INSERT INTO SUPPLIERS (SupplierID, SupplierName, SupplierRole, ContactNum)` `VALUES (sID, sName, sRole, sNum);`
11. Order will delete from table once client has completed the order	To make sure the orders stays updated with only current orders	`DELETE FROM ORDERS` `WHERE orderID = oID;`

Testing

Test Strategy

Test Strategy: **Functional and Validation testing**,

- Go through each of my success criteria and testing them
- Demonstrate the use of unit testing, which tests each small component of the system in order to see if it is working as intended. Furthermore, as the success criteria also links multiple functions of the system together, this shows how I will use integration testing to check the functions are working together as intended. Lastly, I will use validation testing which will evaluate if the system meets the clients requirements.

Test Plan

Criteria For Success	Action/Module	Test Data	Expected Result
1.Allow user to login to system with username and password	Enter authorized user,	Admin, 9@55w0rD	Allowed Access
	Enter unauthorized user	Me, 123456	Generate error message
2. Allow user to manage client information in the system a. User can add client details with corresponding sizes	Enter no name/address/contact number	(null)	Generate error message, “Name, Address, Contact number fields must be filled”

	Enter negative size	-34	Generate error message, “Size must be more than 0”
	Client details within acceptable parameters	John, 26 Beach Road, 9765732, 34	Should be accepted
2b. User can view and edit client details	View without editing	N/A	Should be accepted
	Edit details but cancel process	N/A	Verify data has not been changed,
	Edit details and submit editing	N/A	Verify details have been changed
3. Allow the user to add new order with all relevant details needed for the order	Enter an invalid client name which is not currently in the system	Sophie	Should not be accepted. Output which says “Client needs to be added into the system first through add clients page”
	Enter an invalid design name which is not currently in the system	Blue diamond square ring	Should not be accepted. Output which says “Design needs to be added into system first through add designs page”
	Enter negative order quantity	-2	Should not be accepted
	Enter valid order details	John, Solei, 2	Should be accepted
	Enter no order details	(null)	Should not be accepted/assume no new order
3a. Auto-generated price linked to design of piece and quantity of pieces	When clicking enter new order into the system.	N/A	Should show correct total cost based on quantity.
3b. User can see expected delivery date based on design	When clicking enter new order into the system	N/A	Should show correct estimated delivery date.
4. Allow user to view all current orders a. User can view order details	Click view orders	N/A	Will display all orders in the table.
	All order details are displayed	N/A	Should be accepted

including client and piece in the order			
5. Sort all current orders based on the priority of estimated order delivery	Click view orders	N/A	Order rows are ordered from nearest delivery date to furthest
	Add new order	Data which is needed to calculate order date which will appear between nearest and furthest date	New order will appear in table in correct spot based on the date
6. Allow user to view whether payment is completed in order to begin order	Check that payment has been made when user clicks completed box.	N/A	Generate error message: "Order should not be completed until payment has been made"
	Click payment checkbox	N/A	Able to click payment box to confirm payment which is saved
7a. User can search for specific clients	Entering invalid form of client detail to search	Pink necklace	Should not be accepted, display error message
	Entering valid form of client detail to search	Maria	Client is shown
7b. User can search for specific designs	Entering invalid form of design detail to search	12 Park Avenue	Should not be accepted, and display an error message
	Entering valid form of design detail to search	Bracelet	Design is shown
7c. User can search for specific orders	Entering invalid form of order detail to search	-2	Should not be accepted, display error message
	Entering valid form of order detail to search	Ava	Order is shown
8. Allow user to input designs and information regarding design (type,	Entering negative raw material	-5	Should not be accepted, generate error message "raw

raw materials, labour cost)			material must be 0 or more"
	Entering non-integer number of stones	1.5	Generate error message "Number of stones must be a whole number"
	Not entering type of jewellery	(null)	Should not be accepted, generate order message "must choose the type of jewellery"
	Not entering design name	(null)	Should not be accepted
	Entering a positive value for at least one stone	2,0,4,0	Design accepted and shown in the table of all designs
	Entering negative days	-8	Generate error message
8a. Auto-generate total cost for design based on raw materials and labour cost	Enter new design	N/A	Total cost is shown, calculated by raw materials
8b. User can view all designs in system they have previously entered	Click view all designs	N/A	All designs are shown
	All design details are displayed	N/A	Should be accepted
8c. User can edit all designs in system they have previously entered	Edit without confirming	N/A	Verify no changes have been made
	Edit with confirming changes	N/A	Verify changes have been made
9. Allow users to view supplier information including contact detail and role	View without editing	N/A	Should be accepted
	Edit details but cancel process	N/A	Verify data has not been changed
	Edit details and submit editing	N/A	Verify details have been changed
	All supplier details are displayed	N/A	Should be accepted
9a. Allow users to add	Not entering supplier name	(null)	Generate error

new supplier information			message
	Enter invalid supplier role	24	Generate error message
	Enter valid supplier details within paramters	Josh, Gold Supplier, 93412343	Should be accepted
10. Each order has an automated order number	Enter new valid order	N/A	Unique Order ID
10a. When a client is added to the system a serial number is automatically generated for that design	Enter new valid client	N/A	Unique Client ID appears in table
10b. When a design is added to the system a serial number is automatically generated for that design	Enter new valid design	N/A	Unique Design ID appears in Table
11. Order will delete from the table once a client has completed the order	Completed button is ticked	N/A	Order no longer appears in the table
12. User can see a text file which has the names and order details for clients who have not paid	Button is pressed to create text file	N/A	Text file appears in location on computer that user chooses
13. User can see their top clients, and the top clients most ordered 3 designs.	Button to order analysis page is pressed	N/A	Table with top clients and their 3 designs appears

Criterion C: Development

Table of Contents

Key Features Used

Feature	Purpose / Value
Use of SQL and Java	SQL stores all the users data; Java allows the manipulation with this data within the program.
Use of Properties file	Storing data that is not going to be changed frequently
Multiple Data Structures	Efficient method of handling restriction of jTable to accommodate clients' expansion of the database.
Anonymous Classes	Displaying checkboxes with the use of anonymous classes improves user experience.
Use of Text File	Creates a list of the clients who have not paid for their order and allows the user to save it to a location on their laptop.
Error Handling PopUps	Makes system more user-friendly and avoids causing errors in the program.
Use of Date Libraries	SQL and Java libraries used to present an estimated exact delivery date for orders.
2D ArrayList Manipulation and Sorting	Sort through all clients and orders showing clients top clients top designs.

Use of SQL and Java

Building a system using Java and SQL allows the client to access a database which saves the client's data, through a user-friendly front-end built using Java.

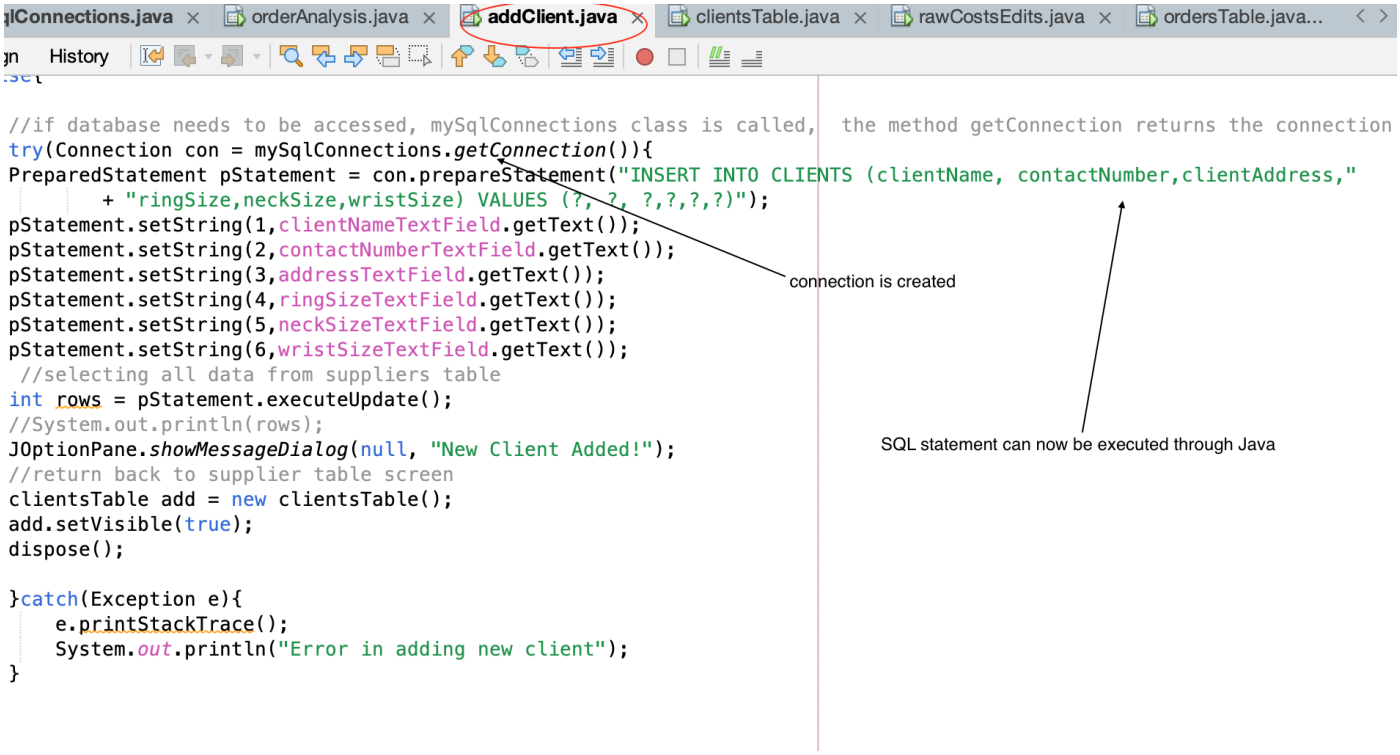
```java
public class mySqlConnections {

    public static void main(String[] args) {
        getConnection();
    }
    //this method creates the mysql connection
    public static Connection getConnection(){
        Connection con = null;
        //creates an instance of connection class in java sql libraries

        try(FileInputStream fs = new FileInputStream("src/main/java/com/mycompany/ia/loginDB.properties")){
            //properties url, files location
            Properties props = new Properties();
            props.load(fs);
            //reading the URL, user and password from properties file
            String url = props.getProperty("URL");
            String user = props.getProperty("user");
            String password = props.getProperty("password");

            con = DriverManager.getConnection(url, user, password);


        }
        catch(Exception e){
            e.printStackTrace();
            System.out.println("Error in connecting to database");

        }
        return con;
        //returns an open connection

    }
```

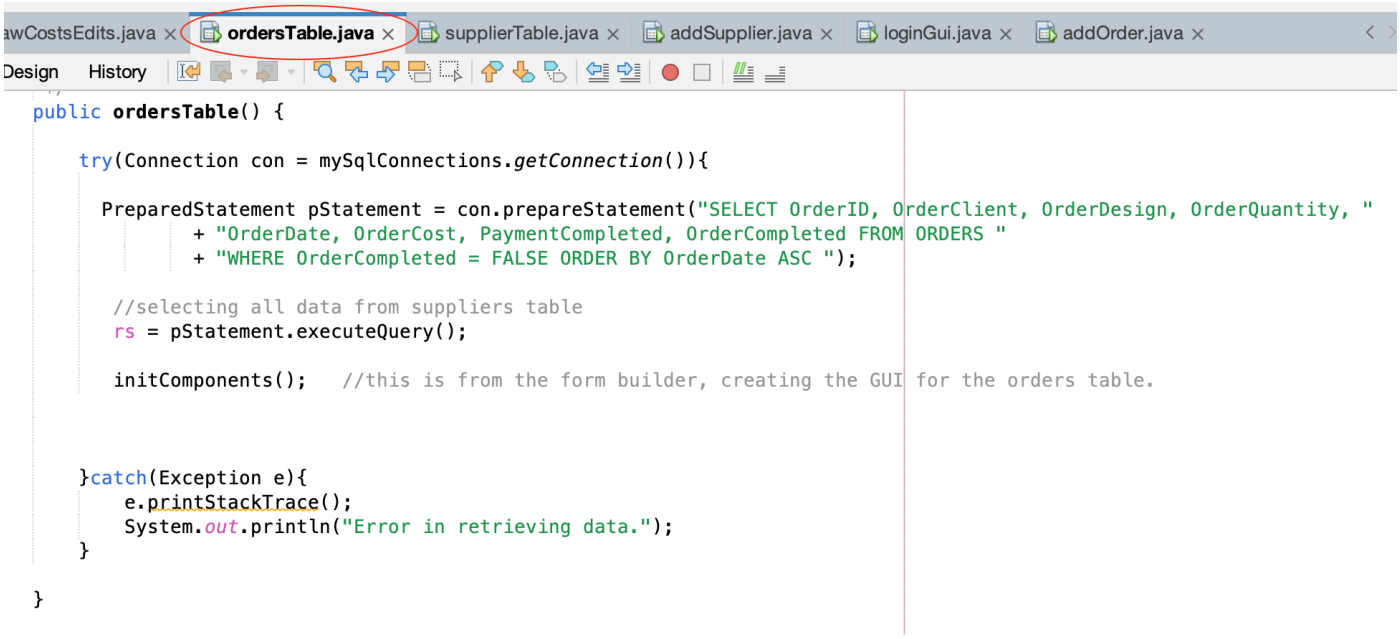
a separate class for connection, can be instantiated at anytime

loading properties file, separate file for additional security

```java
import java.sql.*;
```

API which allows the SQL statements to be executed





Complexity: MySQL Connector/J is a Java Database Connectivity API. Once installed and configured it defined the connection between SQL and Java. Additionally, importing the java.sql package then allowed the SQL queries to be executed through the java
system.

Ingenuity: Making this modular makes it easier if the program needs to be changed, it only needs to be changed in one place, making the code more flexible.

Use of Properties File to Store Data

Storing the username, password and JDBC driver location in a separate file makes the code more secure as it saves data separately so that it is not in the actual code itself.

```
#Properties file
#Login information for database

user = root
password = password

URL = jdbc:mysql://localhost:3306/clients
```

Additionally, the costs are constants which occasionally may need to be changed. Storing them in a properties file means they can be accessed throughout the system. It is also a more efficient way of storing in comparison to storing them in a database.

```
#raw costs properties file
diamondCost=10.4
emeraldCost=12.9
goldCost=5.6
rubyCost=11.2
sapphireCost=17.8
```

```
//getter which allows data to be retreieved from properties file
private void getProperties(){
    try(FileInputStream fs = new FileInputStream("src/main/java/com/mycompany/ia/rawCosts.properties")){
        //properties url, files location
        Properties props = new Properties();
        props.load(fs);
        //reading the stored cost from properties file

        rawCosts[0] = Double.parseDouble(props.getProperty("diamondCost"));
        //properties file takes value as string, need to convert to double
        rawCosts[1] = Double.parseDouble(props.getProperty("rubyCost"));
        rawCosts[2] = Double.parseDouble(props.getProperty("emeraldCost"));
        rawCosts[3] = Double.parseDouble(props.getProperty("sapphireCost"));
        rawCosts[4] = Double.parseDouble(props.getProperty("goldCost"));


    }                                                    reading based on a specific tag of data, e.g heading of gold cost
    catch(Exception e){
        e.printStackTrace();
        System.out.println("Error in connecting to properties file");

    }
}
```

```
//setter which stores/replaces costs in the properties file
private void setProperties(){
    try(FileInputStream fs = new FileInputStream("src/main/java/com/mycompany/ia/rawCosts.properties")){
        //properties url, files location
        Properties props = new Properties();
        props.load(fs);
        //setting the stored cost from text fields

        props.setProperty("diamondCost", diamondCost.getText()); //puts it into the memory
        props.setProperty("rubyCost", rubyCost.getText());
        props.setProperty("emeraldCost", emeraldCost.getText());
        props.setProperty("sapphireCost", sapphireCost.getText());
        props.setProperty("goldCost", goldCost.getText());

        FileOutputStream fos = new FileOutputStream("src/main/java/com/mycompany/ia/rawCosts.properties");

        props.store(fos, "raw costs"); //used to put into properties that is stored in java memory

    }
    catch(Exception e){
        e.printStackTrace();
        System.out.println("Error in connecting to properties file");

    }
}
```

Ingenuity: Out of three different methods: global variables, SQL database and a properties file. The properties file is the most efficient way of storing these costs because there is only one value at a time for each of these costs, and they only need to be changed occasionally by the client. The file also allows it to be accessed by different classes in the system.

Complexity: This requires dealing with another storage method alongside the SQL database, and Java system.

This helps to meet CFS 8a

Multiple Data Structures

To meet the client's criteria of displaying their various data in a table, a 2D static array was needed with the SQL data in order to show it in the table.

```
//gets meta data of results set:
    ResultSetMetaData metaData= rs.getMetaData();

    //gets number of columns in result set
    int columnCount = metaData.getColumnCount();

    //object 1D array for column names
    Object[] columnName = new Object[columnCount];

    //populating columnName array with the Column Names from database
    for (int i=0; i<columnCount; i++){
        columnName[i]=metaData.getColumnLabel(i+1);
    }
```

Complexity: The defaultTableModel constructor needs a static 2D array which needs the exact column and row size to be created. However, the ResultSet metadata did not provide the number of rows of data to display in the table. A dynamic data structure such as a 2D ArrayList first, allows me to find the row count. Which can then be used to create the 2D array with the specific size of columns for the jTable.

Ingenuity: There were limitations of traversal methods on ResultSet to find the row count, which led to using multiple data structures as a simpler method for this. A dynamic 2D ArrayList is not required but is the most efficient way to retrieve the SQL data and display it in a jTable.

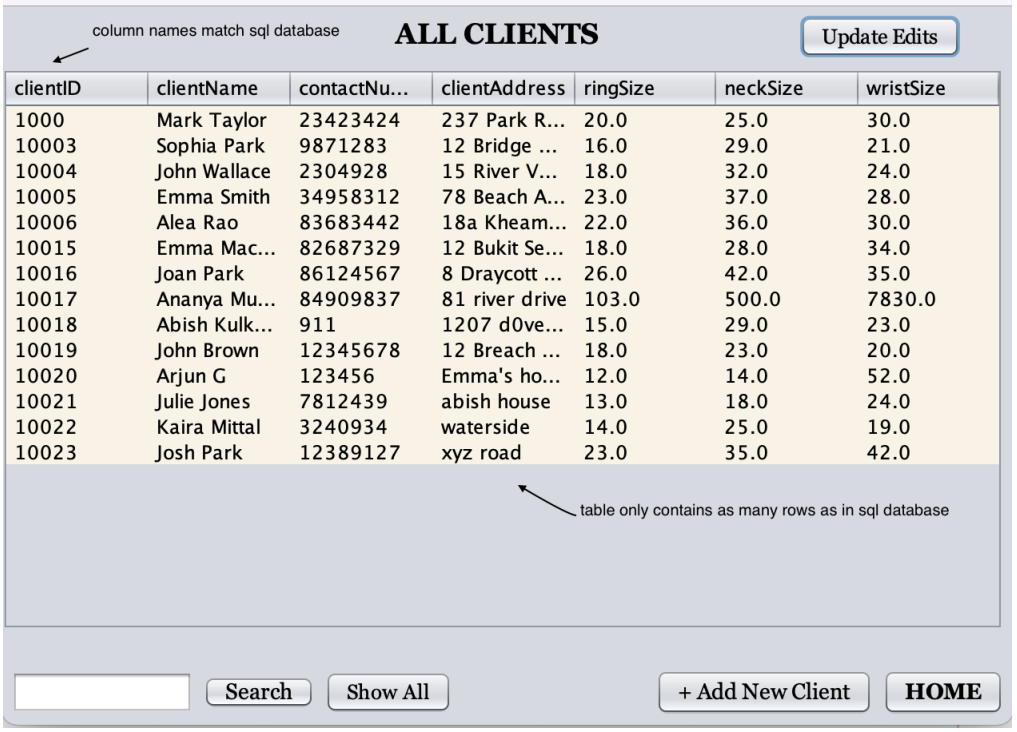
```
ArrayList<ArrayList<Object>> allData = new ArrayList<ArrayList<Object>>();
//creating 2d array list of objects and storing sql data from result set into it

//get the count, save and then use it
int r =0;
boolean tickbox = false;
while(rs.next()){
    ArrayList<Object> rowData = new ArrayList<>();
    for(int c=0; c<columnCount; c++){

        if(rs.getObject(c+1) instanceof Boolean){ //at this column, if the object is a boolean type
            //checking that if we have boolean values in the result set, then we want tickboxes
            tickbox = true;
        }

            rowData.add(rs.getObject(c+1)); //getting object + adding to arrayList as normal for its variable type
        }
    allData.add(rowData);
    }

int rowCount = allData.size();
```

```
            //2D object array which gets all the data from rs
            Object[][] data = new Object[rowCount][columnCount];

            //in order to populate table, cycling through data and putting into 2D array

            //nested loop: outside - while result set has another row, inside - looping through columns
            for (int row = 0; row < rowCount; row++) {
                ArrayList<Object> rowData = allData.get(row);
                for (int col = 0; col < columnCount; col++) {

                    data[row][col] = rowData.get(col);
                }
            }
```

The tableModel can be created using a constructor from the defaultTableModel package in the javax.swing package.

```
        DefaultTableModel tableModel = new DefaultTableModel(data,columnName);

       return tableModel;
```

```
public DefaultTableModel fillTable(){

    DefaultTableModel tableModel=new DefaultTableModel();
    try{

        tableModel = mySqlConnections.setDataToJTable(rs);

    }catch(SQLException e){

        System.out.println(e.getMessage());

    }
    return tableModel;


}
```



This method is suitable for any number of clients the user may have in the future.

Achieve CFS 2,4,8,9

Anonymous Classes

By default, boolean values were displayed as true or false. To make it user-friendly, checkboxes were needed. Research showed that using anonymous classes was the most appropriate technique for this feature.

```
//creating an if statement to see whether or not the table contains boolean values
if(tickbox){
    DefaultTableModel tickTableModel = new DefaultTableModel(data,columnName)

            //need to add types array in the constructer of creating the table model, and not outside the fillTable method
       {//changing to boolean class types, for the boolean columns
            Class[] types = new Class [] { //anonymous class - used generated code model and adapted it
   //looking at example form builder for jTable and recognising what it was doing,



    java.lang.Object.class, java.lang.Object.class, java.lang.Object.class, java.lang.Object.class,java.lang.Object.class,
                java.lang.Object.class, java.lang.Boolean.class, java.lang.Boolean.class
        //boolean class variables for checkboxes
        }; //maybe integers + floats instead of object class (future improvement)
        public Class getColumnClass(int columnIndex) {
            return types[columnIndex]; //this is  where the anonymous class is used for the table model
        } //no constructor for the class, just returning the types

       }

    ;
```

Complexity: Anonymous classes declare and instantiate a class without giving it a name, and without needing a constructor. While creating the table model, the getColumnClass Class returns each column type, ensuring that the jTable recognises the two boolean columns, and displays them as checkboxes.

Ingenuity: After seeing this technique in an example jTable with checkboxes, I used the Java documents to understand how anonymous classes work. By reading these and looking at multiple examples, I was able to apply this technique to my code.

This helps to achieve CFS 4

Use of Text File

The text file allows the client to transfer the list of unpaid orders easily. For instance, if the user needs to print the list, or send it to a supplier.

```
DefaultTableModel model = (DefaultTableModel) ordersTable.getModel();

JFileChooser fileChooser = new JFileChooser();
//creating a file chooser which takes the users files and displays them
int returnValue = fileChooser.showSaveDialog(null);

if (returnValue == JFileChooser.APPROVE_OPTION) { //for the location that is chosen
    File unpaidOrders = fileChooser.getSelectedFile(); //the chosen file is saved
    try {
        FileWriter writer = new FileWriter(unpaidOrders); //the following info is written on the file

        for (int i = 0; i < model.getRowCount(); i++) { //data from the table of clients where paid is false
            if (!Boolean.parseBoolean(model.getValueAt(i, 4).toString())) {
                writer.write(model.getValueAt(i, 1) + ": " + model.getValueAt(i, 2) + " x " + model.getValueAt(i, 3) + "\n");
            }
        }

        writer.close();
        System.out.println("Unpaid orders saved successfully!");
    } catch (IOException e) {
        e.printStackTrace();
        System.out.println("Error in creating list.");
    }
}
```

Complexity: Involves additional method of storing data, which can be stored anywhere on the user's location rather than just within the program files. Also involves accessing the user's file system, which will depend on the user's OS.

Achieves CFS 12

Error Handling

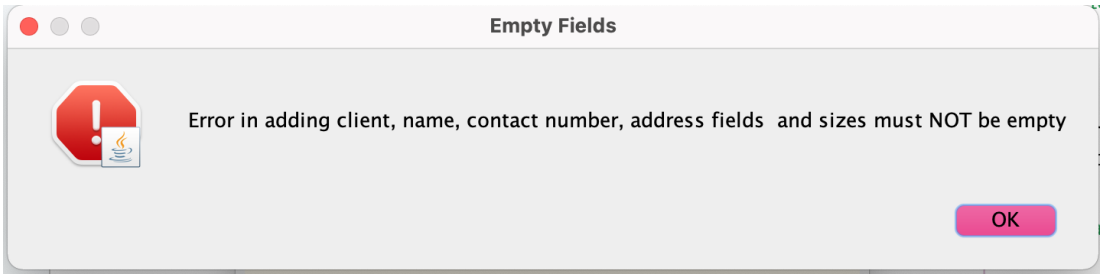
This checks for human error and minimises the risk of the system crashing due to an error.

1. PopUp with validation checks and to confirm changes using jOptionPane popUps

```
if(clientNameTextField.getText().equals("") || contactNumberTextField.getText().equals("")
        || addressTextField.getText().equals("") || (wristSizeTextField.getText().equals("") ||
        neckSizeTextField.getText().equals("") && ringSizeTextField.getText().equals(""))){

    JOptionPane.showMessageDialog(null, "Error in adding client, name, contact number, address fields  "
            + "and sizes must NOT be empty",  "Empty Fields", JOptionPane.ERROR_MESSAGE);
//if the button is clicked and one of the fields are empty, the error message will show
//else the data will be saved to the database, and execute the changes
```

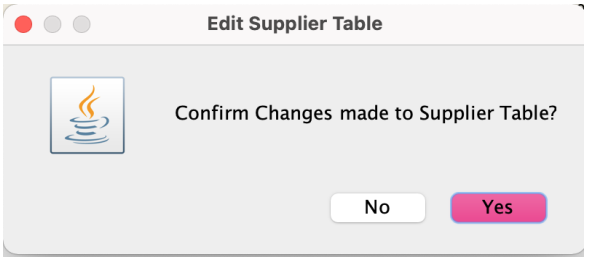
```
int min = 0;
int max = 10;
int step = 1;
int i = 1;
SpinnerModel value = new SpinnerNumberModel(i, min, max, step);
quantitySpinner = new javax.swing.JSpinner(value);
```

prevents client from inputting invalid number into spinner



2. Yes/No popup for users to click whether or not they want to confirm the option.

```
int ans = JOptionPane.showConfirmDialog(null, "Confirm Changes made to Supplier Table?",
        "Edit Supplier Table", JOptionPane.YES_NO_OPTION);

if (ans == JOptionPane.YES_OPTION){
    for (int i =0; i<supplierTable.getRowCount(); i++){
        String supplierID = supplierTable.getValueAt(i,0).toString();
        String supplierName = supplierTable.getValueAt(i,1).toString();
        String supplierRole = supplierTable.getValueAt(i,2).toString();
        String supplierContactNum = supplierTable.getValueAt(i,3).toString();
```



Ingenuity: Minimises the human error the user makes when entering/changing data in the system, and allows the user to check whether the data should be changed or not. Client needs to confirm before the program changes data in the database.

Use of Date Libraries

The client needs to be able to have estimated dates to know when orders should be delivered.

```
import java.time.LocalDate; //give local YYYY-MM-DD
import java.time.LocalTime; //give local time in HH-MM-SS
import java.sql.ResultSet;
import java.time.DayOfWeek;
import java.sql.Date;
```

```
public LocalDate findEstimatedDate(int days, int quantity){

    //first find number of dates, based on quantity, if time is after 12PM add one to total number of days
    int totalDays = days*quantity;
    if (LocalTime.now().isAfter(LocalTime.NOON)){
        totalDays++; //increment total days by 1 if it is afternoon
    }

    //set localDate (store data to return), created instantiation of LocalDate,
    //result is an object of the LocalDate class
    LocalDate result = LocalDate.now(); //will give the current days

    //increment the result every day out of totalDays
    int count = 0;
    while(count < totalDays){

        //increment the result by one day
        result = result.plusDays(1); //plus days is a method of the LocalDate class

        if(!(result.getDayOfWeek()==DayOfWeek.SATURDAY || result.getDayOfWeek()==DayOfWeek.SUNDAY)){
            //if the day of week is NOT saturday OR sunday, then increment the counter
            //future improvements: could look in to how to not include holidays also
            count++;
        }
    }

   return result;
}
```

Ingenuity: The algorithm takes into account aspects which assists the users business process by not including weekends, and taking the time in which the order was placed, providing a fair system for the user's order dates. Simple algorithm when not dealing with long time spans, which is appropriate for the user's requirements.

Complexity: Required researching lots of date libraries and functions and using the correct data type formatting and correct functions to achieve the correct estimated delivery date for the user. Additionally, it was required to use the SQL Date data type to insert the estimated delivery date into the orders database.

```
Date shippingDate = Date.valueOf(findEstimatedDate(estimatedTime, pieceQuantity));
//changes value of localDate into sql Date form


pStatement.setDate(4,shippingDate); //order date
```

Achieves CFS 3b

2D Array Manipulation and Sorting

2D ArrayList manipulation is carried out in order to meet the clients requirements of seeing their top clients and top designs.

```
while(clientsOrders.next()){ //going through the result set of order design + quantity for a specific client
    boolean designFound = false;
    for(int i =0; i<topDesigns.size(); i++){ //looping through array list to check if design is in it

        if(topDesigns.get(i).get(0).equals(clientsOrders.getObject(1))){
            //getting the element at row i, in the first columnn
            //seeing if result set value of order design is equal to the row in the topDesigns
            designFound = true;
            //increment quantity based on order quantity
            int currentQuantity = Integer.parseInt(topDesigns.get(i).get(1).toString());
            //getting the current quantity already in arraylist, turning into int
            currentQuantity = currentQuantity + Integer.parseInt(clientsOrders.getObject(2).toString());
            //updating the new quantity from the result set to value already in arrayList

            topDesigns.get(i).set(1, (Object) (Integer) currentQuantity);
            //boxing it into an integer object type
        }

    }
    if (!designFound){
        ArrayList<Object> newDesign = new ArrayList<>(); //creating new arrayList to add in new design
        newDesign.add(clientsOrders.getObject(1)); //object 1 is name of the design
        newDesign.add(clientsOrders.getObject(2)); //object 2 is quantity ordered


        topDesigns.add(newDesign); //adding newDesigns list to 2d topDesigns arrayList

    }

}
```

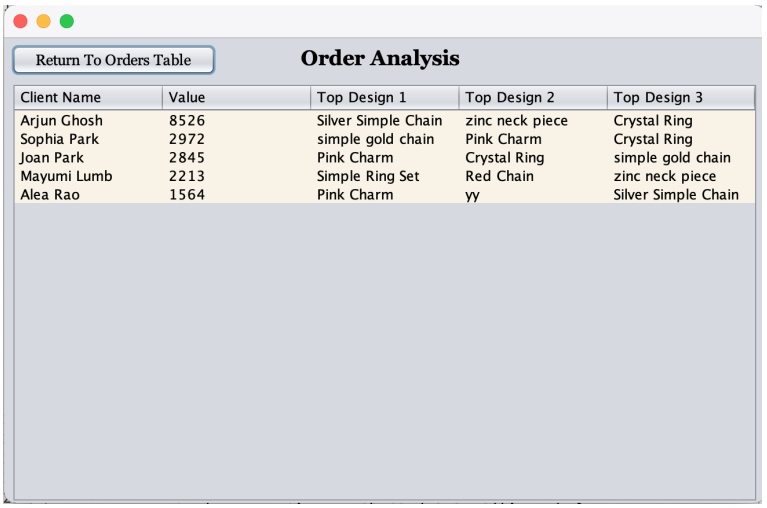
Complexity: To present this data, it requires the manipulation of multiple SQL tables by going through the result set and 2D ArrayList, and checking/adding values or incrementing values when necessary.

Each customer's top orders are searched using a collections sorting technique of a custom comparator:

```
//sort arrayList by largest quantity before adding into 2d array

// Sort the ArrayList using the custom Comparator
Collections.sort(topDesigns, customComparator);

// Print the sorted list
for (int i = 0; i < topDesigns.size(); i++) {
    ArrayList<Object> r = topDesigns.get(i);
    //System.out.println(r.get(0) + ": " + r.get(1));
}

orderedAnalysis[row][0] = topClients.getObject(1); //adding in client name to 2d array
orderedAnalysis[row][1] = topClients.getObject(2); //adding in value of client

for(int i=0; i<designsToShow; i++){
     //columnData is the number of columns to show
    orderedAnalysis[row][i+2]= topDesigns.get(i).get(0); //i+2 is getting the column of design name in table
}

row++; //incrementing row number
```

```
Comparator<ArrayList<Object>> customComparator = new Comparator<ArrayList<Object>>() {
    @Override
        public int compare(ArrayList<Object> list1, ArrayList<Object> list2) {
                  // Assuming the second column contains integers
                Integer value1 = (Integer) list1.get(1);
                Integer value2 = (Integer) list2.get(1);
                    // Sort in descending order
                return value2.compareTo(value1);
        }
};
```

Ingenuity: Efficient way to compare values within a 2D ArrayList and sort it by the top designs for each client, which can then easily be displayed in a jTable.



Achieves CFS 13

Sources

https://netbeans.apache.org/kb/docs/ide/mysql.html
https://stackoverflow.com/questions/10620448/how-to-populate-jtable-from-resultset
https://docs.oracle.com/javase/tutorial/uiswing/components/filechooser.html
https://netbeans.apache.org/kb/docs/java/gui-filechooser.html
https://docs.oracle.com/javase/tutorial/java/javaOO/anonymousclasses.html
https://www.tutorialspoint.com/how-can-we-add-insert-a-jcheckbox-inside-a-jtable-cell-in-java
https://www.codejava.net/java-se/swing/jcheckbox-basic-tutorial-and-examples
https://www.geeksforgeeks.org/how-to-sort-an-arraylist-in-ascending-order-in-java/#:~:text=Approach%3A%20An%20ArrayList%20can%20be,the%20Ascending%20Order%20by%20default.
https://www.javatpoint.com/java-jcheckbox
https://www.javatpoint.com/java-jtable

Extensibility

1. Descriptive Variable names

All swing elements, methods, variables and classes named to describe what they are.

```
private javax.swing.JButton addNewClientButton;
private javax.swing.JLabel allClientsTitle;
private javax.swing.JButton clientToHomeButton;
private javax.swing.JTable clientsTable;
private javax.swing.JCheckBoxMenuItem jCheckBoxMenuItem1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JButton searchButton;
private javax.swing.JTextField searchTextField;
private javax.swing.JButton showAllClientsButton;
private javax.swing.JButton updateClientsButton;
```

2. Use of Comments

Comments explain algorithms and methods, and provide notes for improvements.

```
//this is the search button to be able to search with specific client details within the table
try(Connection con = mySqlConnections.getConnection()){
   PreparedStatement pStatement = con.prepareStatement("SELECT DesignID, DesignName, "
           + "DesignType, GoldMass, labourCost, estimatedTime, totalStones, totalCost FROM "
           + "DESIGNS WHERE DesignName LIKE ? ORDER BY DesignName ASC");
   pStatement.setString(1,"%" + searchTextField.getText()+"%");
//concat with the wildcars for the setString
   //selecting search data from clients table
   rs = pStatement.executeQuery();
   designsTable.setModel(fillTable());
```

```
java.lang.Object.class, java.lang.Object.class, java.lang.Object.class, java.lang.Object.class,java.lang.Object.class,
            java.lang.Object.class, java.lang.Boolean.class, java.lang.Boolean.class
    //boolean class variables for checkboxes

        }; //maybe integers + floats instead of object class (future improvement)
```

3. Readability

Indentations and line control improves readability.

```
//method to calculate cost of new design, called when the add new design button is clicked
public double costOfPiece(){

    rawCostsEdits costs = new rawCostsEdits();

    //cost of labour + gold
    double totalCost = (Double.parseDouble(labourCostTextField.getText()) +
            (Double.parseDouble(goldMassTextField.getText()) * costs.rawCosts[4]));
    //need to get access to rawCosts array

    //cost of stones
    double stonesCost = ((Integer) diamondSpinner.getValue() * costs.rawCosts[0]) +
            ((Integer) rubySpinner.getValue() * costs.rawCosts[1]) +
            ((Integer) emeraldSpinner.getValue() * costs.rawCosts[2])+
            ((Integer) sapphireSpinner.getValue() * costs.rawCosts[3]);



    totalCost = stonesCost + totalCost;
    System.out.println(totalCost + " after adding stones");

    return totalCost;
}
```

4. Modularity

Extra classes and extra methods make it easier to read and edit the code.

```
//This class holds the general sql connection and other sql utility methods including:
//creating the jTable models, creating connection to the database

public class mySqlConnections {
```

```
public static String[] rsToComboBox(ResultSet rs) throws SQLException{
    //result set of one column, column of string variables only, store all the data from the result set,
    //into a 1d arrayList
    //get the number of rows in the arrayList, put the data from the arrayList into a 1d String array
    //return the string array

    ArrayList<Object> searchData = new ArrayList<Object>();

    //get the count, save and then use it
    int r =0;
    while(rs.next()){
        searchData.add(rs.getObject(r+1)); //adds the object from the result set into the arrayList
    }
    int rowCount = searchData.size(); //gets size of the arrayList

    //2D object array which gets all the data from rs
    String[] arrayData = new String[rowCount]; //column of string varai

    //nested loop: outside - while result set has another row, inside - looping through columns
    //int r = 0;
    for (int row = 0; row < rowCount; row++) {
        arrayData[row]=searchData.get(row).toString();
    }
    return arrayData;
}
```

Word Count: 998

Criterion E: Evaluation

Evaluation of Success Criteria

Criteria for Success	Met?	Evaluation
1. Allow user to login to system with username and password	Yes	The user appreciated the system's access restrictions for data confidentiality (See Appendix, Interview 4, Q1).
2a. User can add client details with corresponding sizes	Yes	The program's strict adherence to CFS ensures accurate data entry, enhancing user efficiency, achieving the CFS. (See Appendix, Interview 4, Q4).
2b. User can view and edit details	Yes	Organized tables allow easy viewing and understanding of client information (See Appendix, Interview 4, Q2).
3. Allow user to add new order with all relevant details needed for the order	Yes	The feature to select clients/designs prevents incorrect entries, through having a drop-down list with all existing clients/designs. (See Appendix, Interview 4, Q12).
3a. Auto-generated price linked to design of piece and quantity of pieces	Yes	Automatic cost calculation and display saved time and was valuable for user. (See Appendix, Interview 1, Q4)
3b. User can see expected delivery date based on design	Yes	The expected delivery date was shown for an order was valuable for the client as it can aid in planning ahead, avoiding non-working days. (See Appendix, Interview 4, Q10).
4a. User can view order details including client and piece in the order	Yes	Viewing current orders facilitates tracking, with prioritized order rows, meeting CFS for client & myself. (See Appendix, Interview 4, Q10)
5. Sort all current orders based on priority of estimated order delivery	Yes	
6. Allow user to view whether payment is completed in order to begin order	Yes	Provided an easy way for the user to keep track of clients payments with visual checkboxes.
7a. User can search for specific clients	Yes	Feature made it easy to see specific clients/designs based on parts of the name, & find specific details quicker. (See Appendix, Interview 4, Q3).
7b. User can search for specific designs	Yes	
7c. User can search for specific orders	Yes	On reflection, users could also be able to search by designs that have been ordered in orders table.
8. Allow user to input designs and information regarding design (type, raw materials, labour cost)	Yes	This met the CFS-8, also giving the user the flexibility of choosing number of stones with the use of JSpinners (See Appendix, Interview 4, Q8).

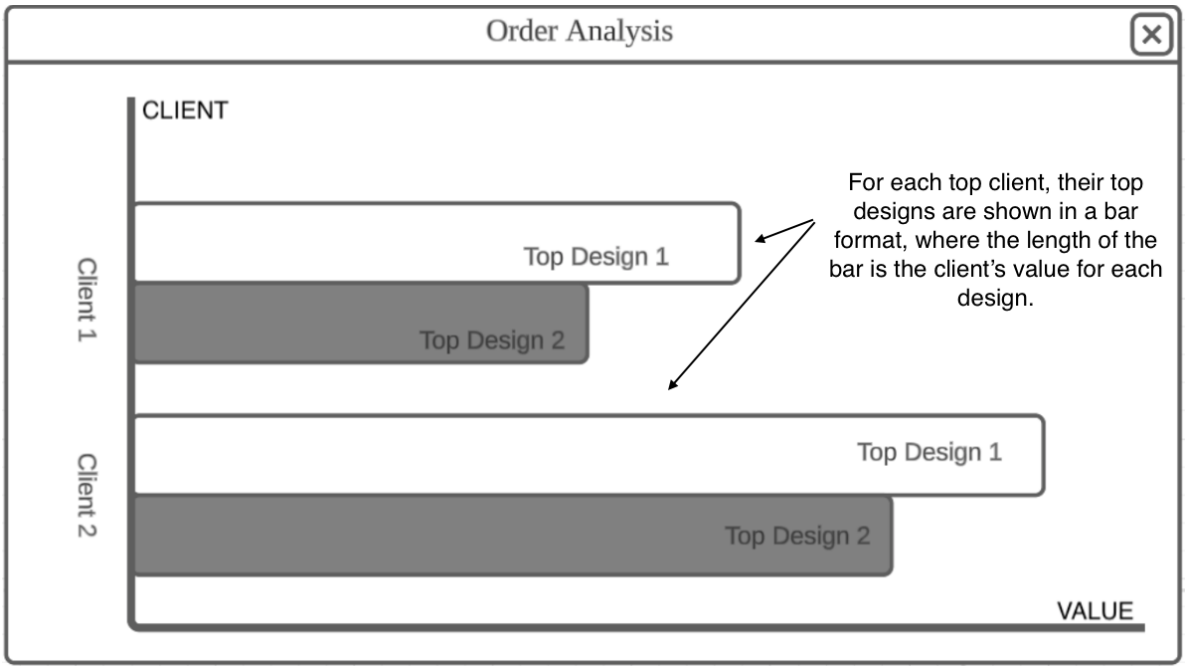
8a. Auto-generate total cost for design based on raw materials and labour cost	Yes	This gave accurate values for total cost upon testing, effective as it saved users time by not having to self-calculate for each design.
8b. User can view all designs in system they have previously entered	Yes	The user can see all designs, but on reflection user should be able to see the number of each stone rather than the total stone count in order to keep stock.
8c. User can edit all designs in system they have previously entered	Yes	This works similarly to CFS2b, making it easy for the user to adapt to the system, as the different functions work similarly.
9. Allow users to view supplier information including contact detail and role	Yes	The table shows all supplier details & functions the same as the client's table so it is easy to get used to using, making the user satisfied.
9a. Allow users to add new supplier information	Yes	This works same as CFS2a meeting users & my expectations.
10. Each order has an automated order number 10a. When a client is added into the system a serial number is automatically generated for that client 10b. When a design is added into the system a serial number is automatically generated for that design	Yes	Program intuitively provides a unique ID for each client/design/order.
11. Order will delete from table once client has completed the order	Yes	The client knew when orders were completed as they would now appear in the past orders table. (See Appendix, Interview 4, Q9).
12. User can see a text file which has the names and order details for clients who have not paid	Yes	Text files gave users a way to refer to clients who have not paid and users can transfer it easily to their phone so they do not have to check the system for this.
13. User can see their top clients, and the top clients most ordered 3 designs.	Yes	Table successfully shows analysis information displaying their top clients/designs, aiding sales strategies (See Appendix, Interview 4, Q14)

Recommendations for Further Development

Recommendation 1: Presenting top clients & top designs in a visual representation.

This is important as the client mentioned they would prefer this visual representation, as it is easier for them to understand the analysis. (See Appendix, Interview 4, Q14)

This can be implemented using the external library, JFreeCharts to create the visual display as follows:



Recommendation 2: Use of action listeners within the jTable to improve editing fields process.

The action listeners would know when a field has been edited by double-clicking the field, and it would ask the client using a confirmation screen to confirm the changes.

This ensures the user is not making accidental changes when editing details in the table, and it also saves time by not having to update each field in the table using the 'Update clients/orders etc.' button.

Possible Algorithm Change:

```
clientsTable.addMouseListener(new MouseAdapter() {
    public void mouseClicked(MouseEvent e) {
        if (e.getClickCount() == 2) { // Double-click event
            int selectedRow = clientsTable.getSelectedRow();
            int selectedColumn = clientsTable.getSelectedColumn();

            if (selectedRow >= 0 && selectedColumn >= 0) {
                // Get the current cell value
                String currentValue = clients
table.getValueAt(selectedRow, selectedColumn).toString();

                // Prompt the user for a new value
                String newValue =
JOptionPane.showInputDialog("Enter the new value:", currentValue);

                if (newValue != null) { // Check if the user
didn't cancel the edit
                    clientsTable.setValueAt(newValue, selectedRow,
selectedColumn);
                }
            }
```

Appendix

Table of Contents

Interview 1

Date: 15th March 2023

1. Could you please describe the current problem you are facing?
As a jewellery business owner, over time I have never really had a proper inventory system to manage my orders, information, and payments. This has begun to affect my efficiency of my orders and as I seem to forget client sizing for jewellry and therefore need to re measure each time the client places a new order. Along with this, my pricing is set for specific products but because I do not have a proper place to store it, I often forget the prices for products and have to recalculate it based of the design from scratch.

2. What is the system you are currently using and how does this cause you problems?
Currently, most of my information and data is on pieces of papers which is very scattered and not organised. I have a little bit of data in a spreadsheet but overall it is not the most effective method for these problems. This causes me problems because if the system is very disorganized then it makes my process more complicated and not efficient.

3. What would you expect from a successful solution to this problem?
The least amount of managing data the better. For example, I would like to only be able to input a piece of data once and then from there I am able to navigate through these different processes without having to enter anything again. In addition to this, I would expect a successful solution to be accessible to me. I would like it to be on my laptop as this is where it would be best to store all my data. This would help me alot in the future because it would mean I don't have to spend as much time on this part of my business.

4. What are some specific features that would be needed to be implemented in this system?
Some ideas of specific features that I would like to be implemented in this system would include:

- Keep clients details (email, phone number, name) and then also next to that be able to keep their measurement sizes for different types of jewelry (bracelet, necklace, ring etc)
- Be able to look at current orders that I have to work on and see when they are expected to be done, so I am able to notify the client. Along with this, for each order, I would like to be able to see if the payment has been made or not and for some clients give them the option to pay in installments. Therefore, I would also need to be able to calculate the dates when installment

payments need to be made.
- Maybe a supplier database with contact details for the different suppliers in India who work on the different parts of the jewelry creating process. This would make it easier for me to contact the right person if needed.
- One main section of the system would be to hold past designs and specifications about the design. Like: raw materials, labour cost, total cost, what type of jewelry it is, etc. I would also like to be able to input new pieces of jewelry and their raw materials. This would be helpful so I can begin to start clicking on pre existing designs instead of starting over with each order. For this, a way to calculate the price of a product based on the raw materials and labour cost would be extremely useful.

Interview 2

Date: 3rd May 2023

1. Based on the screens I just showed you, do you have any feedback on how you feel they could be better to suit your requirements?
I think that overall I am quite happy with the screens you have should me as they show the information I need in an organised way. However, there are a couple things which I think need to be adjusted within the screens. Firstly, I think that if there was a way to go back to the homepage from each of the different key areas then that would be extremely useful, as then I can do multiple things at once before closing the app. Also maybe adding in a confirmation button to make sure I don't add things in accidentally into the system. Other than that, since I will have a lot of orders, is there something you can add which makes it easier for me to scroll through the orders? Also, you may need to take note that each of the stones have a different base cost.

2. Can I please clarify how the design is created, and how the cost is generally calculated?
Sure. So firstly, I draw the design out and send it to my main supplier. They will then figure out how many stones I need and what type of stones, the mass of gold I will need, and how much it will cost to make it. After that I usually manually calculate the cost with the set prices for the stones and gold and then adding on the labour cost.

3. What are the different types of stones you include?
The four types of stones are: Ruby, Diamonds, Emeralds, and Sapphires

4. Is there anything else which you would like to add?
I think that having a serial number to be able to refer to orders would be extremely helpful as it makes it easier for me to send to my suppliers as a reference. Also, I already have been using client serial numbers so is there a way that this could be included in the order number?

Interview 3

Date: 13th August 2023

1. **Looking at the current version of the Login Page, are there any issues right now?**
   I like how it looks because it is quite simple and easy to use. But, would there maybe be a way to be able to see the password? So I can see it before I log in to try and check if it is right?

2. **Yes sure I can work on that! How about this homepage?**
   I like its simplicity, I can easily see the different things that I can do with the product!

3. **Good to hear. I wanted to also get your input on the layout of the client's page.**
   I like the table which shows all of my clients, could we maybe change the colour of it though? Maybe to a yellow colour? This could be the same for all the designs in the system.

Final Interview & Feedback

Date: 4th Nov 2023

1. **The product is finally finished! While you are testing it, I have a couple of questions to ask you to ensure the product meets the agreed-upon criteria! Firstly, were you able to login to the system?**
   Yes, I could easily log in with my username and password, and the system didn't let me enter if I entered the wrong username or password, which was good for keeping my data protected. I especially liked the show password button!

2. **Firstly, are you able to see and edit all the clients and the correct details?**
   Yes, I can! I like how it is ordered on the table, it makes it very easy for me. It is also very easy for me to edit the details and I like the confirmation screen in case I make a mistake. I also like how I can see a unique ID number for each client.

3. **Great! Are you able to search for specific clients?**
   Yes, I can easily search for the clients by their first or last name!

4. **Were you able to add a new client?**
   Yes, I can add a new client, and the error message was quite helpful if I forgot to add a detail.

5. **Moving on to the suppliers section, were you able to view, edit, and add suppliers like the client table?**
   Yes, I like how it works very similarly to the clients section so it is easy for me to use and get used to.

6. **Good to hear! Now how did you find the designs section**
   I liked how I could view all the designs in alphabetical order, this was a great addition. Also, the auto-calculated costs for the pieces save me a lot of time.

7. **Were you able to search for specific designs?**
   Yes, I could do this quite easily

8. **How did you find adding a new design?**
   I like the use of these buttons which only allow me to choose numbers for the costs, and I like how I can easily change the raw costs. Overall, it works super well in allowing me to add a new design

9. **Okay, now let's talk about the orders section. Do you like how you can see all the orders?**
   Yes, I like how you have put the name of the client and the design of the order. I also really like the checkboxes because they make it visually easier for me to see whether or not the payment is completed, and click when an order is completed. But I think if there was one thing that could be changed in the future, is there a way for the checkboxes to immediately update without me having to press the update orders button? Because I feel the current way is a bit complicated sometimes.

10. **Okay, I will keep that in mind. Do you like how the dates are formatted?**
    Yes, I think this is a great way for me to see the estimated dates easily, it saves me a lot of time from having to calculate it manually. I also noticed that you sorted it in the nearest to the furthest date which I like.

11. **That's good to hear! Were you able to search for specific client's orders easily?**
    Yes, but would there be a way for me to also search by specific designs that customers have ordered?

12. **I will look into adding that! How did you find adding a new order?**
    I thought it was super easy, as I could easily choose the client and design from the lists, so I did not risk writing the client's name or design name wrong.

13. **Great, now how about when you pressed the unpaid orders button?**
    I like how I could choose where to store the file! This made it super easy to go back to the file when I needed to.

14. **Wonderful! Lastly, how did you find the order analysis section?**
    I think this is one of my favourite features. It is super useful for me to be able to see my top clients and the top designs that have been ordered. But I was wondering if there was any way that this could be presented in some sort of chart style, like a graph? Just so it is visually easier for me to see.

15. **Okay sure, I will keep that in mind! Overall, do you have any other general comments?**
    I think overall I really like the system because it was really easy to pick up and understand! I also think that the confirmation screens are a great addition because they help me to make sure I'm not making mistakes on my part within my business.

16. **Alright, thank you for your feedback. I'm glad the system works to your liking.**
    Thank you!