RAMAIAH UNIVERSITY OF APPLIED SCIENCES

ASSIGNMENT - 2

Course Code	19CSC302A
Course Name	Database Systems
Programme	B. Tech
Department	CSE
Faculty	FET

Name of the Student	K Srikanth
Reg. No	17ETCS002124
Semester/Year	5th Semester/3rd Year
Course Leader/s	Ami Rai E.

Declaration Sheet				
Student Name	K Srikanth			
Reg. No	17ETCS002124			
Programme	B. Tech		Semester/Year	5th Semester/ 3rd Year
Course Code	19CSC302A			
Course Title	Database Systems			
Course Date	14/09/2020	to	16/02/2021	
Course Leader	Ami Rai E.			

Declaration

The assignment submitted herewith is a result of my own investigations and that I have conformed to the guidelines against plagiarism as laid out in the Student Handbook. All sections of the text and results, which have been obtained from other sources, are fully referenced. I understand that cheating and plagiarism constitute a breach of University regulations and will be dealt with accordingly.

Signature of the Student		Date	
Submission date stamp (by Examination & Assessment Section)			

Signature of the Course Leader and date	Signature of the Reviewer and date

Faculty of Engineering & Technology			
Ramaiah University of Applied Sciences			
Department	Computer Science and Engineering	**Programme**	B. Tech. Computer Science and Engineering
Semester/Batch	5th/2018		
Course Code	19CSC302A	**Course Title**	Database Systems
Course Leader(s)	A. Prabhakar, Gp Capt. N Rath VSM, Ami Rai E.		

Assignment - 02					
Register No.		17ETCS002124	Name of Student	K Srikanth	
Sections		**Marking Scheme**	**Max Marks**	**First Examiner Marks**	**Second Examiner Marks**
Part A	A1.1	Introduction to parallel database system	01		
	A1.2	Discussion on query processing in parallel database system	02		
	A1.3	Discussion on challenges	02		
		Part-A Max Marks	**05**		
Part B1	B1.1	Identification of candidate key(s)	02		
	B1.2	Highest normal form	02		
	B1.3	Decomposition of the tables	04		
		B1 Max Marks	**08**		
Part B2	B2.1	Design and implementation of GUI	03		
	B2.2	Connection of front end with the database	03		
	B2.3	Discussion on the results	03		
	B2.4	Concluding remarks (Summary, limitations, improvements	03		
		B2 Max Marks	**12**		
		Total Assignment Marks	**25**		

RAMAIAH UNIVERSITY OF APPLIED SCIENCES

Course Marks Tabulation				
Component-1(B)Assignment	**First Examiner**	**Remarks**	**Second Examiner**	**Remarks**
A				
Marks (out of 10)				
Signature of First Examiner				Signature of Second Examiner

Please note:

1. Documental evidence for all the components/parts of the assessment such as the reports, photographs, laboratory exam / tool tests are required to be attached to the assignment report in a proper order.
2. The First Examiner is required to mark the comments in RED ink and the Second Examiner's comments should be in GREEN ink.
3. The marks for all the questions of the assignment have to be written only in the **Component – CET B: Assignment** table.
4. If the variation between the marks awarded by the first examiner and the second examiner lies within +/- 3 marks, then the marks allotted by the first examiner is considered to be final. If the variation is more than +/- 3 marks then both the examiners should resolve the issue in consultation with the Chairman BoE.

Assignment - 02

Instructions to students:

1. The assignment consists of **3**questions: Part A –**1** Question, Part B- **2**Questions.
2. Maximum marks is**25**.
3. The assignment has to be neatly word processed as per the prescribed format.
4. **Submission Date:** 16/01/2020
5. **Submission after the due date is not permitted.**
6. **IMPORTANT**: It is essential that all the sources used in preparation of the assignment must be suitably referenced in the text.
7. Marks will be awarded only to the sections and subsections clearly indicated as per the problem statement/exercise/question

Part A
Question A1

A1.1)

Introduction

Now a days the Companies are needed to handle huge amount of data with high data transfer rate. The client server is not much efficient to improve the efficiency this a gave birth to the concept of Parallel Databases. Parallel database system improves performance of data processing using multiple Hardware's like multiple CPU and disks are used parallelly. This also performs very good operations like query processing and data loading.

The Basic Idea to Build a Parallel Database was done keeping 4 Points in mind,

- **Improvement in Performance**
  The performance of the system can be improved by connecting multiple CPU and disks in parallel.
- **Availability of data**
  Data can be copied to multiple locations to improve the availability of data which would help if some other process is using the data.
- **Reliability**
  Reliability of system can be improved with accuracy in the system and availability of data as mentioned in point 2.
- **Distributed access of data**
  Now a days Companies are having many branches in multiple cities so to Access the data the parallel database system would be the solution.

A1.2)

Parallel query processing designates the transformation of high-level queries into execution plans that can be efficiently executed in parallel, on a multiprocessor computer. This is achieved by exploiting the way data is placed in parallel and the various execution techniques offered by the parallel database system. As in query processing, the transformation from the query into the execution plan to be executed must be both correct and yield efficient execution. A relational query execution plan is graph/tree of relational algebra operators.

Parallel processing divides a large task into many smaller tasks, and executes the smaller tasks concurrently on several nodes/processor. As a result, the larger task completes more quickly. In sequential processing, independent tasks compete for a single resource. Only task 1 runs without having to wait. Task 2 must wait until task 1 has completed; task 3 must wait until tasks 1 and 2 have completed, and so on. In parallel processing, more CPU power is assigned to the tasks. Each independent task executes immediately on its own processor: no wait time is involved.

A parallel database system should maintain the following characteristics:

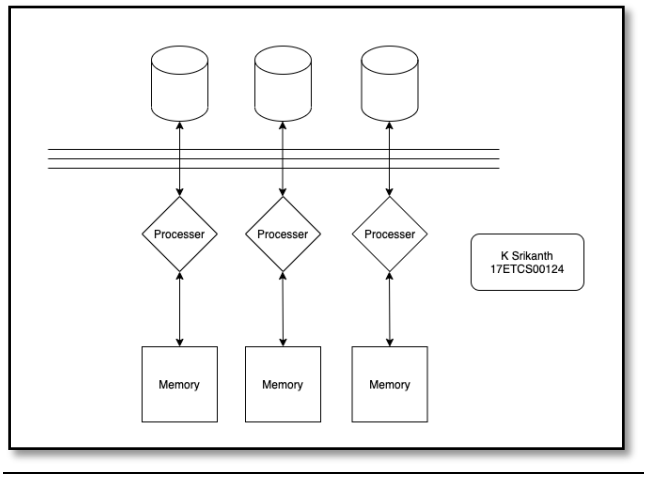
- **Concurrency Management**
- **Task Synchronization**
- **Resources sharing**
- **Data Placement**
- **Network Scaling**

Among all the above characteristics, synchronization is a critical success factor. To synchronize the database in loosely coupled architecture / shared nothing architecture mostly use locking mechanism. Same mechanism also implements with tightly coupled architecture for data placement.

In a parallel database system, proper data placement is essential for load balancing. Ideally, interference between concurrent parallel operations can be avoided by having each operation work on an independent dataset. These independent datasets can be obtained by the delustering (horizontal partitioning) of the relations based on hash function or range index and allocating each partition to a different memory module. The execution strategy that is proposed here as follows:

- First, the total amount of work that has to be done to evaluate the query is minimized.
- Then, try to distribute that minimal amount of work equally over the available processors.

Architecture when Memory Module Not Shared



1. Separate the all relations along with it predicates those are use in user's query.
2. Assign the separate relation to separate processor / CPU along with its memory module.
3. Transfer the intermediates result set with predicates to other CPU to evaluate result.
4. Repeat step 3 till all predicates not evaluated.

The above four steps are only for retrieve / select statements. If the SQL statements update or insert statements then

1. Partitioning: distributing the tuples of a relation over several disks / memory modules which will help to allow parallel databases to exploit the I/O bandwidth of multiple disks by reading and writing them in parallel or Relations are decluttered (partitioned horizontally) based on range index, hash function methods.
2. Each CPU accesses the separate memory module and insert or update the data in the database in the accessible memory module. Note when the CPU will access the cluster of databases, concurrency control protocols (lock mechanism) must be enabled.
3. Combine the all-memory module and transfer the data into the datafile.

Example

Consider the following query:

MySQL Query

```
>> Update table into C values (select table from A, B where A.x = B.y);
```

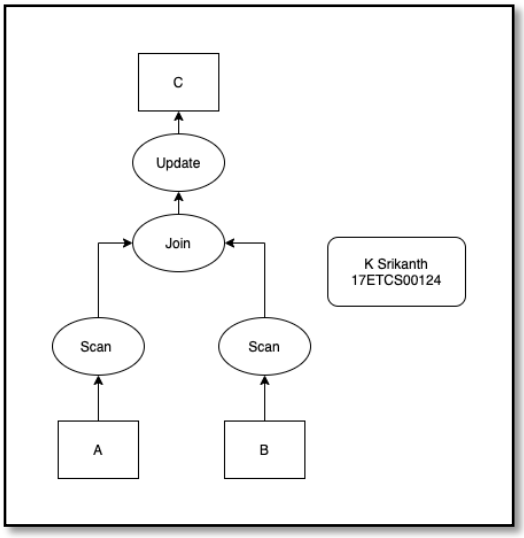


Figure 1 Normal Execution

Simple conventional plan is shown above Figure 1.

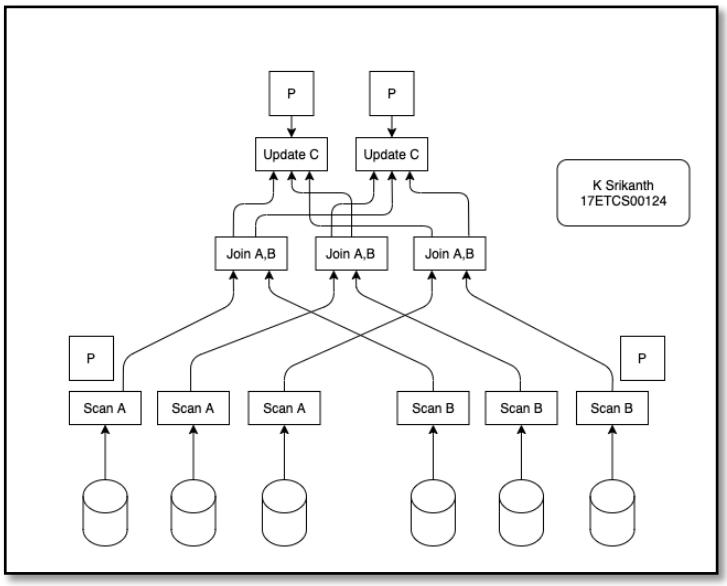


Figure 2 parallel query execution

But when the same query executes in multiprocessor environment then the query executes in parallel manner shown above and response time will be minimized and system throughput increase dramatically.

A1.3)

Sometimes the overhead involved in parallelism outweigh the benefits of it.

- Startup: The time needed to start a parallel operation may dominate the actual computation time.
- Interference: When accessing shared resources, each new process slows down the others
- Skew: The response time of a set of parallel processes is the time of the slowest one

Parallel data management techniques should intend to overcome these barriers. Apart from these, there are other challenges that needs to be taken into account.

Load Balancing:

The execution time of a parallel algorithm on a given processor is determined by the time required to perform its portion of the computation plus the overhead of any time spent performing communication or waiting for remote data values to arrive. The execution time of the algorithm as a whole is determined by the longest execution time of any of the processors. For this reason, it is desirable to balance the total computation and communication between processors in such a way that the maximum per-processor execution time is minimized. This is referred to as load balancing, since the conventional wisdom is that dividing work between the processors as evenly as possible will minimize idle time on each processor, thereby reducing the total execution time.

Data distribution:

Another challenge in parallel programming is the distribution of a problem's data. Most conventional parallel computers have a notion of data locality. This implies that some data will be stored in memory that is "closer" to a particular processor and can therefore be accessed much more quickly. Data locality may occur due to each processor having its own distinct local memory—as in a distributed memory machine—or due to processor-specific caches as in a shared memory system. Due to the impact of data locality, a parallel programmer must pay attention to where data is stored in relation to the processors that will be accessing

Conclusion:

Parallel DBMSs have become a reality in the last few years. They provide the functionality of centralized DBMSs, but in a multiprocessor system. Parallel DBMSs are perhaps the only realistic approach to meet the performance requirements of a variety of important applications which place significant throughput demands on the DBMS. In order to meet these requirements parallel DBMSs, need to be designed with special consideration for the protocols and strategies.

Part B

Question B1

B1.1)

Given,

EMPLOYEE (ID, Name, Type, Salary, Designation, Address)

Functional Dependencies,

- **ID -> Type, Address**
- **Type -> Salary**
- **Name ->Designation**

To Find the Candidate Keys and the Normalization I will simplify the given Functional Dependencies and the schema into,

Schema (A, B, C, D, E, F) ← Schema

Functional Dependencies

- **A → C, F**
- **C → D**
- **B → E**

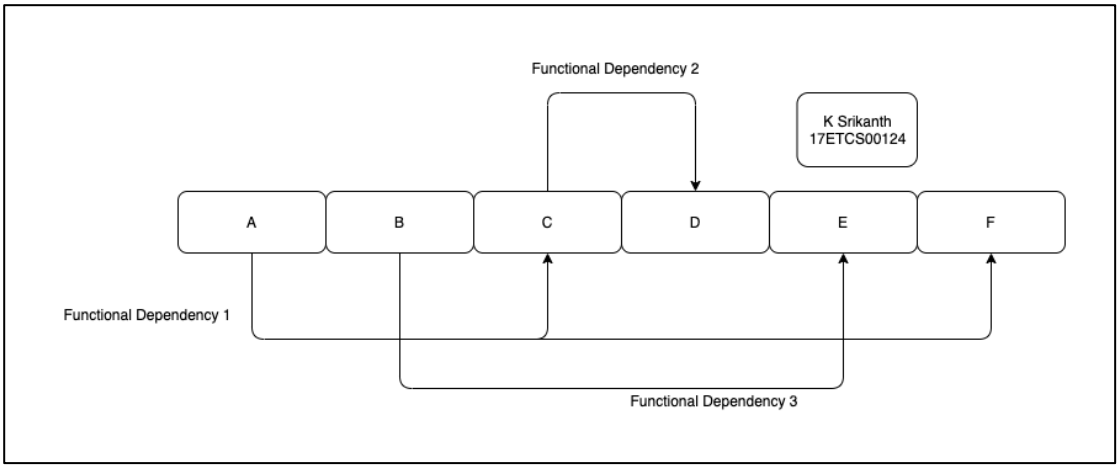


Figure 3 Simplified Functional Dependencies

Now that I've simplified the Schema and Functional Dependencies, now we find the candidate keys
In order to find the candidate keys, we can find the closure of the attribute or set of attributes. If the closure of among them can find all the attributes in the relation, we can say that attributes is are candidate keys.
So, Let's now take A,B Closure

```
{AB}* = {A, B, C, D, E, F}
```

Now, as we can **{AB}*** can all the attributes in the schema so **AB is Candidate Key** and **{A, B}** are the **prime attributes** and **{C, D, E, F} are the non-prime attributes.**

B1.2)

In Order to find the Highest Normal Form that our Schema is in we write down all the Functional Dependencies
Given are 3 Functional Dependencies,

Functional Dependencies

- **A → C, F**
- **C → D**
- **B → E**

Let's say that the attributes are not multivalued and composite.

So, the Condition of the **1st Normal Form(1NF**) is that it does not allow any multivalued or composite attributes or any attribute which is both multivalued and composite. As we assumed that all the attributes are not multivalued or composite so this concludes that our Functional Dependencies are in **1NF Normal Form**. Now let's look at the Condition of 2nd Normal Form(2NF) is that a relation schema must be in 1NF and every nonprime attribute in the relation must be fully functionally dependent on the prime attributes of the relation. As stated in B1.1 the Candidate key is {AB}

So, The Prime attributes are {A, B} and {C, D, E, F} are the non-prime attributes.

Let's take **"C",** as we can see that partially dependent on **"A"** but we can determine **"C" with just "A"** Attribute, so we can say that the **"C" is not fully dependent** on **prime attributes so it's not in 2nd Normal Form** and the Given relation with Functional Dependencies is in **1NF Normal Form.**

B1.3)

Now, we know that our relation is in 1NF Normal Form.

- **The Prime attributes: {A, B}**
- **The non-prime attributes {C, D, E, F}**

- ***Attribute C*:** We can see that " **C** " can be determined with the " ***A*** " attribute and we don't need "**B**" attribute. **So, "C" is partially dependent on "A ".**
- ***Attribute D*:** We can see that **"D"** can be determined with the **"C" attribute** is dependent on the **"A" attribute** and thus we don't require **"B"**. So, "**D" is partially dependent on "A".**
- ***Attribute E*:** We can see that **"E"** can be determined with the **"B"** and we don't need the **"A".** So, **"E"** is partially dependent on **"A".**
- ***Atrribute F*:** We can see that **"F"** can be determined with the **"A"** and we don't require the **"B".** So, "**F**" is partially dependent on "***A***".

As we already know that the given Relation and Functional Dependencies in 1NF so we have to normalize it to 2NF 2nd Normal Form

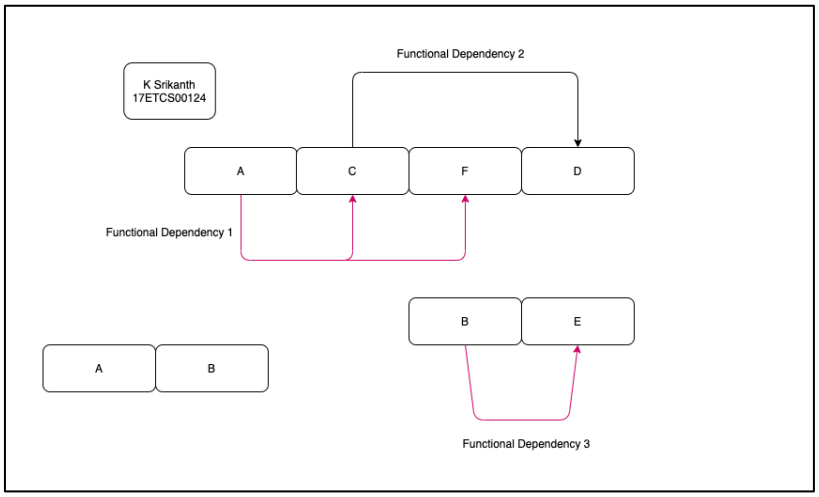


Figure 4 2'nd Normal Form for the Image 3

As we can see that Image 3 has all non-prime attribute fully functionally dependent on the prime attributes it's also in **1st Normal Form.** Now we can say that it's in **2nd Normal Form.**

Now that our relation is in 2nd Normal Form. Now, we convert it in 3rd Normal Form(3NF) where the condition is that a relation must be in 2nd Normal Form and no nonprime attribute of the relation is transitively dependent on the primary key.

Now the **Attribute "D"** is transitively dependent on the **"A" attribute as you can see**

when $\mathbf{A \rightarrow C}$ **and** $\mathbf{C \rightarrow D}$**.** So, we can say that its transitive. So, the relation is not in 3rd Normal Form.

Now to normalize this relation into **3rd Normal form** we need to decompose the relations to new relations such that it contains the non-key attribute which functionally determines the other non-key attributes.

After Normalizing it into **3rd Normal form.**

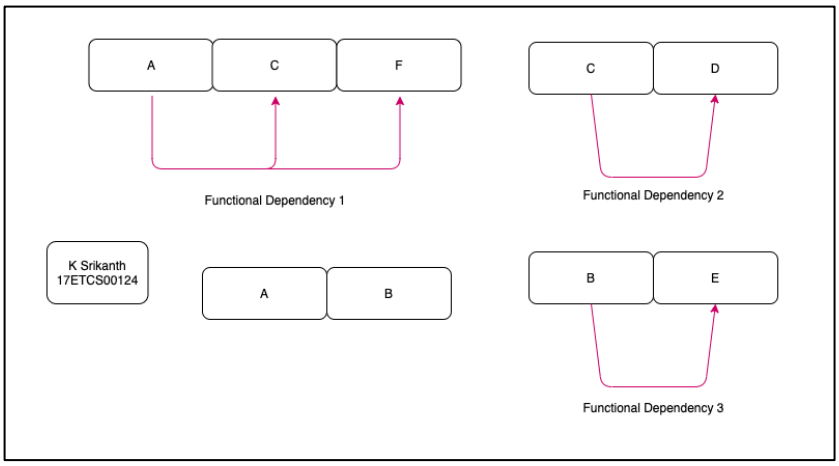


Figure 5 3'nd Normal Form for the Image 3

Now looking at **Image 5** we see that it's in 3rd Normal Form where no-prime attribute is transitively dependent on the prime attribute. Now we can say that it's in **3nd Normal Form.**

Now that our relation is in **2nd Normal Form.** Now, we convert it **Boyce-Codd normal form (BCNF)** where the condition is that if there exists a n**ontrivial functional dependency** in the relation say $\mathbf{X \rightarrow A}$**, then X** must be a super key of the relation.

So, let's roll back our Functional Dependencies that was given,

- **A → C, F**
- **C → D**
- **B → E**

As we see that all the 3 Functional Dependencies are Non-trivial. We already know that **{AB}** forms the candidate key. So, we can consider **{ABC} to be the super key. When you look at the 1st Dependency "A" is a part of Super key, and** when you consider next two dependencies on the left side all attributes are **part of Super key. So, when all the Dependencies care** dependencies are non-trivial and on the Left-hand side of the functional dependency is part of super key. Then we can say that it's in **Boyce-Codd normal form (BCNF).**

Question B2

B2.1)

Implementation of Graphic User Interface,

Login

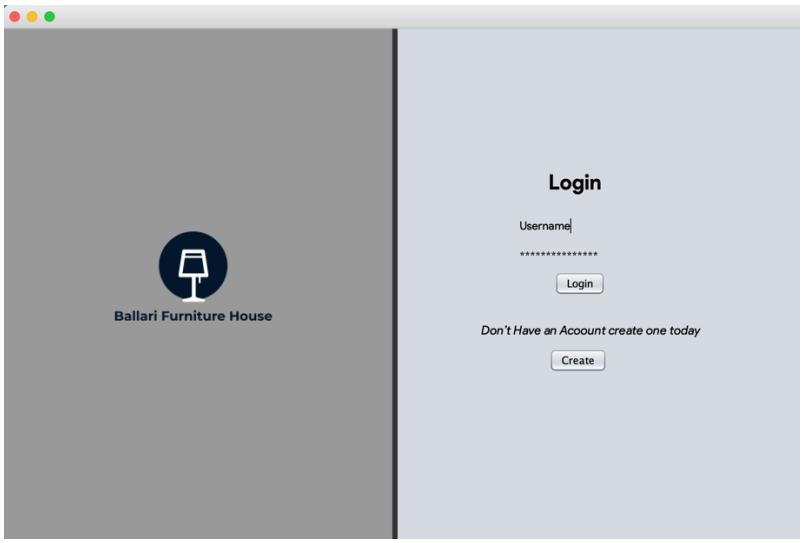


Figure 6 Swing Interface for Login Page

This the basic UI layout (Login Page) where we can see that it have **two text fields** where the user can enter his/her **User Name and Password** and has two buttons which are **Login** which verifies your credentials and proceeds to next page and when you click on **Create** it lets the **User** take to Signup page which will let you create an account to login. **UI Layout (Image 2)**

Signup

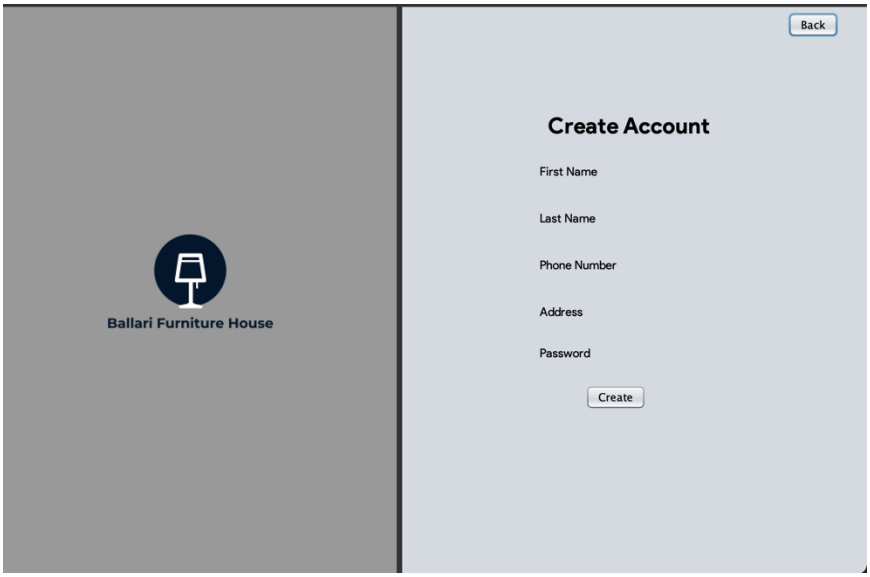


Figure 7 Swing Interface for Signup Page

This the basic UI layout (Signup Page) where we can see that it have **Five text fields** where the user can enter his/her **First Name Followed by Last Name, Phone Number, Address and Password** and has two buttons which are **Create** which will let you create an account and using the credentials **User** can **login and will be Redirected** and when you click on **Back** it lets the **User** take to **Login** if user wishes not to create an account today

Home Page

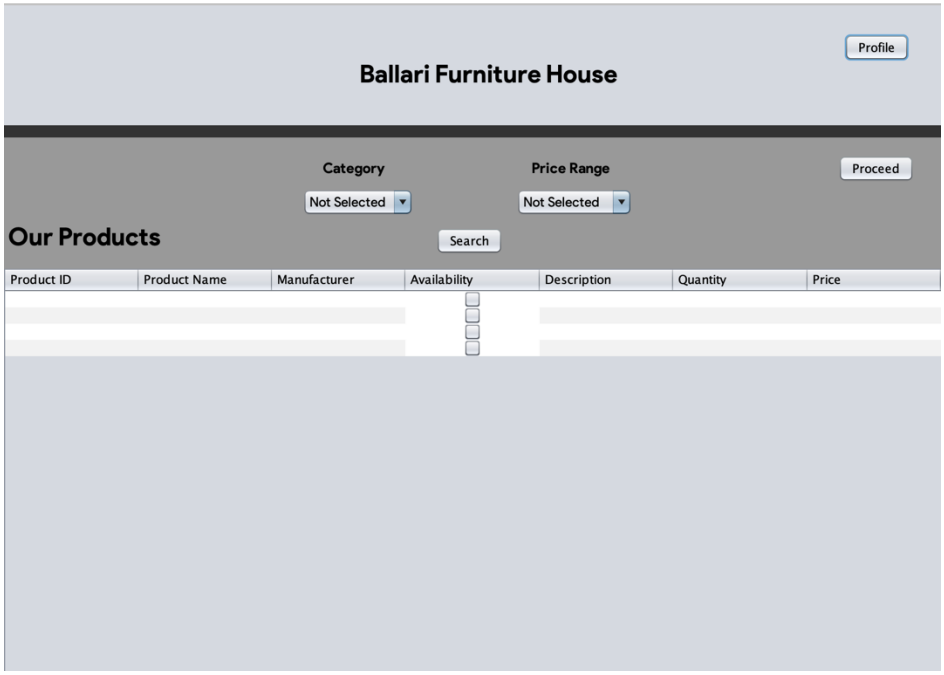


Figure 8 Swing Interface for Home Page

This the basic UI layout (Home Page) where we can see that it has a table with **Seven Columns** where the user can see all the products and **their availability** and if he/she wishes to **search with a category or price range** keeping in check then the interface provides with that option . This Page has two buttons which are **Profile** which will let you **Edit** an account or Look at the products that **User** has **ordered recently** and when you click on an **Product** in the table and click on **Proceed** it lets the **User take to Payment Page** where you can further proceed.

Payment Page

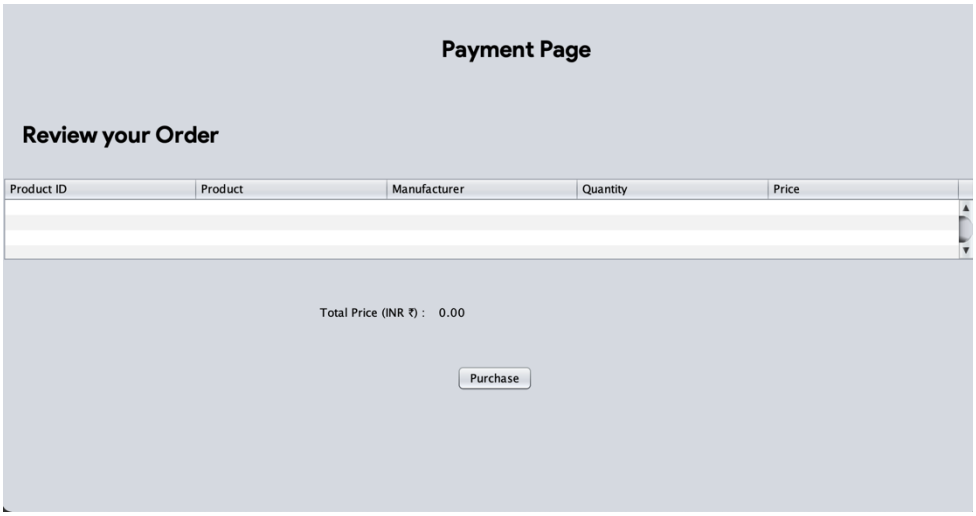


Figure 9 Swing Interface for Payment Page

This the basic UI layout (Payment Page) where we can see that it has a table with **Five Columns** where the user can Review **his/her products** and at the end it **displays the total price that User has in their cart.** This Page a button with **makes Successful order after the payment** and **redirects the User to Home Page.**

Profile Page

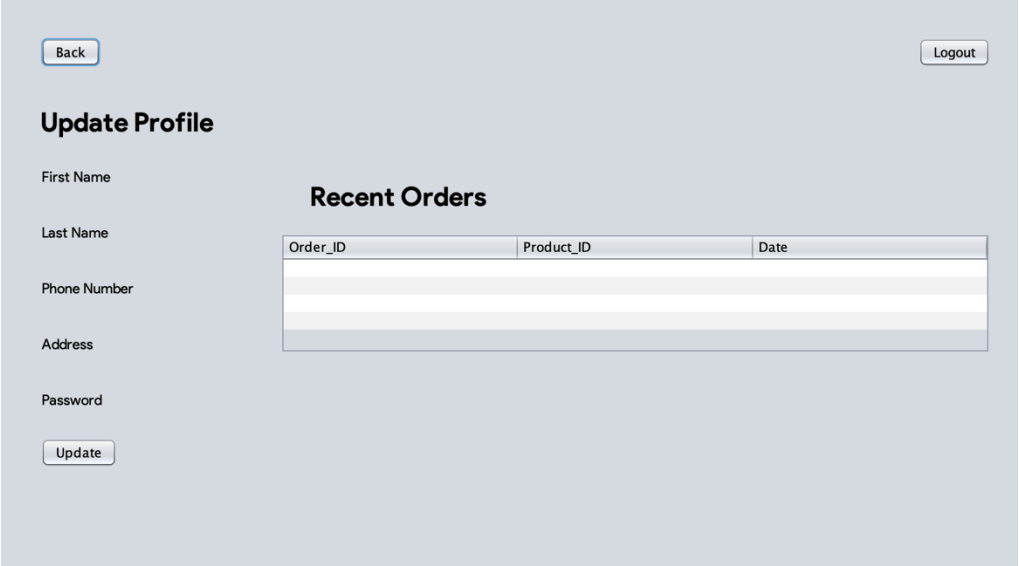


Figure 10 Swing Interface for Profile Page

This the basic UI layout (Profile Page) where we can see that it have **Five text fields** where the user can edit their info by his/her **First Name Followed by Last Name, Phone Number, Address and Password.** and has a table where the **User can see his/her Recent order** if they exist. Also this page has two buttons which are **Back** which will **Redirect the User to Home Page** and when you click on **Logout** it lets the **User** take to **Logout** and the user will be **Redirected to Login Page.**

B2.2)

Implementation of JDBC (Java Database Connectivity) Driver

Login

Login Button (login.java)

```
private void Login_ClickActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_Login_ClickActionPerformed
    // K Srikanth 17ETCS002124
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/assignment", "root", "Sri123");

        Statement st = con.createStatement();
        String user = Username_Text.getText();
        String pass = Password_Text.getText();    The method getText() from the type JPasswordField is deprecated
        String sel = "select * from user where User_ID = '" + user + "' and Password = '" + pass + "';";
        ResultSet rs = st.executeQuery(sel);
        if (rs.next()) {
            JOptionPane.showMessageDialog(null, "Welcome Back, " + user + ":)");
            HomePage mp = new HomePage(user);
            mp.setVisible(true);
            dispose();
        } else {
            JOptionPane.showMessageDialog(null, "Seems Like You Entered Wrorg Username or Password. \nOpps ! Try again. ");
        }

        con.close();

    } catch (ClassNotFoundException ex) {
        Logger.getLogger(Login.class.getName()).log(Level.SEVERE, null, ex);
    } catch (SQLException ex) {
        Logger.getLogger(Login.class.getName()).log(Level.SEVERE, null, ex);
    }
}//GEN-LAST:event_Login_ClickActionPerformed
```

Figure 11 Java Program for Login Button (Action)

This function called up when the **login button is clicked** and it basically **checks if the Username and the Password exist in the Database** and if they are then it displays and dialog message saying **“Welcome Back, User” if else the Username and the Password don’t exist in the Database** then it displays a dialog message saying **“Seems Like You Entered Wrong Username or Password. Opps! Try again.”**

MySQL Query

```
>> select * from user where User_ID = '" + user + "' and Password = '" + pass + "';
```

Create Button (login.java)

```
private void Create_ClickActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_Create_ClickActionPerformed
    // K Srikanth 17ETCS002124
    Signup someobj = new Signup();
    someobj.setVisible(true);
}//GEN-LAST:event_Create_ClickActionPerformed
```

Figure 12 Java Program for Create Button (Action)

This function called up when the **Create button is clicked** and if the **User doesn’t have an account to login** then this action will take the **User to Signup Page**

Signup

Create Account Button (signup.java)

```
private void Create_ButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_Create_ButtonActionPerformed
    // K Srikanth 17ETCS002124
    String First_Name;
    String Last_Name;
    String Phone_Number;
    String Address;
    String Password;
    First_Name = F_Name_var.getText();
    Last_Name = L_Name_var.getText();
    Address = Add_var.getText();
    Phone_Number = Ph_var.getText();
    Password = Pass_var.getText();
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/assignment", "root", "Sri123");
        pst = con.prepareStatement("insert into user(First_Name,Last_Name,Phone_Number,Address,Password) values (?,?,?,?,?)");
        pst.setString(1, First_Name);
        pst.setString(2, Last_Name);
        pst.setString(3, Phone_Number);
        pst.setString(4, Address);
        pst.setString(5, Password);
        pst.executeUpdate();
        JOptionPane.showMessageDialog(this, "Account Has Been Created Please Login via Username from your phone");
        F_Name_var.setText("First Name");
        L_Name_var.setText("Last Name");
        Add_var.setText("Address");
        Ph_var.setText("Phone Number");
        Pass_var.setText("Password");
        F_Name_var.requestFocus();

        Login someobj = new Login();
        someobj.setVisible(true);

    } catch (ClassNotFoundException ex) {
        Logger.getLogger(Signup.class.getName()).log(Level.SEVERE, null, ex);

    } catch (SQLException ex) {
        Logger.getLogger(Signup.class.getName()).log(Level.SEVERE, null, ex);
    }

}//GEN-LAST:event_Create_ButtonActionPerformed
```

Figure 13 Java Program for Create Button (Action)

This function called up when the **Create button is clicked** and it basically takes all the inputs from **five text fields and inserts it into the Database** after a **successful insertion** it displays a dialog message saying **"Account Has Been Created Please Login via Username from your phone"** and will be redirected to login page.

MySQL Query

```
>> insert into user(First_Name,Last_Name,Phone_Number,Address,Password) values (?,?,?,?,?)
```

Back Button (signup.java)

```
private void Back_ButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_Back_ButtonActionPerformed
    // K Srikanth 17ETCS002124
    Login someobj = new Login();
    someobj.setVisible(true);
}//GEN-LAST:event_Back_ButtonActionPerformed
```

Figure 14 Java Program for Back Button (Action)

This function called up when the **Back button is clicked** and it basically **redirects the User** to **Login Page if the User wishes to go back.**

Home Page

Viewtable Custom Function (HomePage.java)

```
public void viewtable() {
    // K Srikanth 17ETCS002124
    int c;
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/assignment", "root", "Sri123");
        pst = con.prepareStatement("select Product_ID,Product_Name,Manufacturer,Availability_Status,Description,Quantity,Product_price from product");
        rs = pst.executeQuery();
        ResultSetMetaData rsd = rs.getMetaData();
        c = rsd.getColumnCount();
        DefaultTableModel dft = (DefaultTableModel) Display_table.getModel();
        dft.setRowCount(0);

        while (rs.next()) {
            Vector v2 = new Vector();

            v2.add(rs.getString("Product_ID"));
            v2.add(rs.getString("Product_Name"));
            v2.add(rs.getString("Manufacturer"));
            v2.add(rs.getBoolean("Availability_Status"));
            v2.add(rs.getString("Description"));
            v2.add(rs.getString("Quantity"));
            v2.add(rs.getDouble("Product_price"));

            dft.addRow(v2);
        }
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(HomePage.class.getName()).log(Level.SEVERE, null, ex);
    } catch (SQLException ex) {
        Logger.getLogger(HomePage.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Figure 15 Java Program for viewtable function (custom)

This function called up when **the User is redirected to Homepage and in the table, it displays all the products that are available with Seven Columns** where the user can see the details of each product such as its **Product ID and Product Name followed by the Maker of the that product with Description of the product and is it Available of not in the store if its Available then what is its Quantity and Price.** This data is retrieved from the MySQL database.

MySQL Query

```
>> select
Product_ID,Product_Name,Manufacturer,Availability_Status,Description,Quantity,Product_price from
product
```

Viewsearchtable Custom Function (HomePage.java)

```
public void viewsearchtable(int index, int Price) {
    // K Srikanth 17ETCS002124
    String[] cat = {"Desk", "Bed", "Sofa", "Dinning_Table", "Outdoor"};
    int[] pricemin = {0, 1000, 10000, 25000};
    int[] pricemax = {1000, 10000, 25000, 100000000};

    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/assignment", "root", "Sri123");
        pst = con.prepareStatement(" select * from product where Category = ? and Product_price >= ? and Product_price <= ?");
        System.out.println(cat[index]);
        System.out.println(pricemin[Price]);
        System.out.println(pricemax[Price]);
        pst.setString(1, cat[index]);
        pst.setInt(2, pricemin[Price]);
        pst.setInt(3, pricemax[Price]);
        rs = pst.executeQuery();
        ResultSetMetaData rsd = rs.getMetaData();
        c = rsd.getColumnCount();
        DefaultTableModel dft = (DefaultTableModel) Display_table.getModel();
        dft.setRowCount(0);
        while (rs.next()) {
            Vector v2 = new Vector();
            v2.add(rs.getString("Product_ID"));
            v2.add(rs.getString("Product_Name"));
            v2.add(rs.getString("Manufacturer"));
            v2.add(rs.getBoolean("Availability_Status"));
            v2.add(rs.getString("Description"));
            v2.add(rs.getString("Quantity"));
            v2.add(rs.getDouble("Product_price"));
            dft.addRow(v2);
        }
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(HomePage.class.getName()).log(Level.SEVERE, null, ex);
    } catch (SQLException ex) {
        Logger.getLogger(HomePage.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Figure 16 Java Program for viewsearchtable function (custom)

This function called up when **the User is wants to search for a product via a category or via price range in the Homepage table, it displays all the products that are available with Seven Columns** where the user can see the details of each product such as its **Product ID and Product Name followed by the Maker of the that product with Description of the product and is it Available of not in the store if its Available then what is its Quantity and Price in that selected category and price range.**

MySQL Query

```
>> select * from product where Category =? and Product_price >= ? and Product_price <= ?
```

Search Button (HomePage.java)

```
private void Search_ButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_Search_ButtonActionPerformed
    // K Srikanth 17ETCS002124
    int price_index = Price_Box.getSelectedIndex();
    int catagory_index = Catagory_Box.getSelectedIndex();
    catagory_index = catagory_index - 1;
    price_index = price_index - 1;
    System.out.println(catagory_index);
    System.out.println(price_index);
    viewsearchtable(catagory_index, price_index);
}//GEN-LAST:event_Search_ButtonActionPerformed
```

Figure 17 Java Program for Search Button (Action)

This function called up when the **Back button is clicked** and it basically take the indexes from the **drop box and according to that indexes the viewsearchtable function is called.**

Profile Button (HomePage.java)

```
private void Profile_buttonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_Profile_buttonActionPerformed
    // K Srikanth 17ETCS002124
    Profile_Page someobj = new Profile_Page(user, this);
    someobj.setVisible(true);
}//GEN-LAST:event_Profile_buttonActionPerformed
```

Figure 18 Java Program for Profile Button (Action)

This function called up when the **Profile button is clicked** and it basically **redirects the User** to **Profile Page if User wants to perform some actions.**

Proceed Button (HomePage.java)

```
private void Proceed_ButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_Proceed_ButtonActionPerformed
    // K Srikanth 17ETCS002124
    try {
        DefaultTableModel model = (DefaultTableModel) Display_table.getModel();
        int index = Display_table.getSelectedRow();
        int Product_ID = Integer.parseInt(model.getValueAt(index, 0).toString());
        String Product_Name = model.getValueAt(index, 1).toString();
        String Manufacturer = model.getValueAt(index, 2).toString();
        int Quantity = 1;
        double Product_price = Double.parseDouble(model.getValueAt(index, 6).toString());
        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/assignment", "root", "Sri123");
        pst1 = con.createStatement();
        String Cartquery = "Create TEMPORARY TABLE cart(Product_ID int NOT NULL,Product_Name VARCHAR
        (50) NOT NULL,Manufacturer VARCHAR (50) NOT NULL,Quantity int NOT NULL,Product_Price double(10,2) NOT NULL)";
        pst = con.prepareStatement("insert into #cart values(?,?,?,?,?)");
        System.out.println(Product_ID);
        pst.setInt(1, Product_ID);
        pst.setString(2, Product_Name);
        pst.setString(3, Manufacturer);
        pst.setInt(4, Quantity);
        pst.setDouble(5, Product_price);
        pst1.execute(Cartquery);
        pst.executeUpdate();
        CartPage someobj = new CartPage(user, this);
        someobj.setVisible(true);
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(HomePage.class.getName()).log(Level.SEVERE, null, ex);
    } catch (SQLException ex) {
        Logger.getLogger(HomePage.class.getName()).log(Level.SEVERE, null, ex);
    }
}//GEN-LAST:event_Proceed_ButtonActionPerformed
```

Figure 19 Java Program for Proceed Button (Action)

This function called up when **the User is wants to add a product to cart and can review it in payment page so the MySQL query that you see here is it deletes all the content in table first and then it inserts it into cart table which acts as a temporary table for transaction to happen.**

MySQL Query

```
>> Create TEMPORARY TABLE cart(Product_ID int NOT NULL,Product_Name VARCHAR (50) NOT
NULL,Manufacturer VARCHAR (50) NOT NULL,Quantity int NOT NULL,Product_Price double(10,2) NOT
NULL)
```

```
>> insert into #cart values(?,?,?,?,?)
```

Payment Page

Viewtable Custom Function (CartPage.java)

```
public void viewtable() {
    // K Srikanth 17ETCS002124
    double sum = 0;
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/assignment", "root", "Sri123");
        pst = con.prepareStatement("select * from #cart");
        rs = pst.executeQuery();
        ResultSetMetaData rsd = rs.getMetaData();
        DefaultTableModel dft = (DefaultTableModel) Review_Table.getModel();
        dft.setRowCount(0);
        while (rs.next()) {
            Vector v2 = new Vector();
            v2.add(rs.getString("Product_ID"));
            v2.add(rs.getString("Product_Name"));
            v2.add(rs.getString("Manufacturer"));
            v2.add(rs.getString("Quantity"));
            v2.add(rs.getDouble("Product_price"));
            sum = sum + rs.getDouble("Product_price");
            dft.addRow(v2);
        }
        Money_Box.setText(String.valueOf(sum));
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(HomePage.class.getName()).log(Level.SEVERE, null, ex);
    } catch (SQLException ex) {
        Logger.getLogger(HomePage.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Figure 20 Java Program for View Function (Custom)

This function called up when **the User is redirected to Payment Page and in the table,** it **displays all the products that are selected by the User** and the User can review his/her order and **make a payment** through a Payment API (not implemented).

MySQL Query

```
>> select * from #cart
```

Purchase Button (CartPage.java)

```
private void Purchase_ButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_Purchase_ButtonActionPerformed
    // K Srikanth 17ETCS002124
    JOptionPane.showMessageDialog(this, "You Order was Placed Sucessfully \nThank You for Shopping with Us.");
    try {
        DefaultTableModel model = (DefaultTableModel) Review_Table.getModel();
        int rowscount = model.getRowCount();
        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/assignment", "root", "Sri123");
        pst = con.prepareStatement("insert into orders(Product_ID,User_ID) values(?,?)");
        pst1 = con.prepareStatement(" drop table cart");
        for (int i = 0; i < rowscount; i++) {
            int Product_ID = Integer.parseInt(model.getValueAt(i, 0).toString());
            pst.setInt(1, Product_ID);
            System.out.print(user);
            pst.setString(2, user);
            pst.executeUpdate();
            pst1.executeUpdate();
        }
        pst1.executeUpdate();
        home.setVisible(true);
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(HomePage.class.getName()).log(Level.SEVERE, null, ex);
    } catch (SQLException ex) {
        Logger.getLogger(HomePage.class.getName()).log(Level.SEVERE, null, ex);
    }
```

Figure 21 Java Program for Purchase Button (Action)

This function called up when the **User clicks on the purchase button** and it displays a **dialog message "You Order was Placed Successfully. Thank You for Shopping with Us." And the data is inserted into Orders table if the order is successful.**

MySQL Query

```
>> insert into orders(Product_ID,User_ID) values(?,?)
```

```
>> drop table #cart
```

Profile Page

Update Button (Profile.java)

```
private void Update_ButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_Update_ButtonActionPerformed
    // K Srikanth 17ETCS002124
    String First_Name;
    String Last_Name;
    String Phone_Number;
    String Address;
    String Password;

    First_Name = F_Name_var.getText();
    Last_Name = L_Name_var.getText();
    Address = Add_var.getText();
    Phone_Number = Ph_var.getText();
    Password = Pass_var.getText();
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/assignment", "root", "Sri123");
        pst = con.prepareStatement("update user set First_Name = ?,Last_Name = ?,Phone_Number = ? ,Address = ? , Password= ? where Phone_Number = ?");
        pst.setString(1, First_Name);
        pst.setString(2, Last_Name);
        pst.setString(3, Phone_Number);
        pst.setString(4, Address);
        pst.setString(5, Password);
        pst.setString(6, Phone_Number);
        pst.executeUpdate();
        JOptionPane.showMessageDialog(this, "Hey, " + First_Name + " Your Account has been Updated.");
        F_Name_var.setText("First Name");
        L_Name_var.setText("Last Name");
        Add_var.setText("Address");
        Ph_var.setText("Phone Number");
        Pass_var.setText("Password");
        F_Name_var.requestFocus();
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(Profile_Page.class.getName()).log(Level.SEVERE, null, ex);

    } catch (SQLException ex) {
        Logger.getLogger(Profile_Page.class.getName()).log(Level.SEVERE, null, ex);
    }
}//GEN-LAST:event_Update_ButtonActionPerformed
```

This function called up when the **Update button is clicked** when the U**ser wants to Update his/her data** and the MySQL Query **Updates it into the Database** after a **successful Updating** it displays a dialog message saying **“Hey, User" Your Account has been Updated.”**

MySQL Query

```
>> insert into user(First_Name,Last_Name,Phone_Number,Address,Password) values (?,?,?,?,?)
```

Viewtable Custom Function (Profile.java)

```
public void viewtable() {
    // K Srikanth 17ETCS002124
    int c;
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/assignment", "root", "Sri123");
        pst = con.prepareStatement("select * from orders join product where product.product_id=orders.product_id and orders.User_ID = ?");
        pst.setString(1, user);
        rs = pst.executeQuery();
        ResultSetMetaData rsd = rs.getMetaData();
        c = rsd.getColumnCount();
        DefaultTableModel dft = (DefaultTableModel) Display_Table.getModel();
        dft.setRowCount(0);

        while (rs.next()) {
            Vector v2 = new Vector();
            for (int i = 1; i < c; i++) {
                v2.add(rs.getString("Order_ID"));
                v2.add(rs.getString("Product_Name"));
                v2.add(rs.getObject("Date_Time"));
            }
            dft.addRow(v2);
        }
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(HomePage.class.getName()).log(Level.SEVERE, null, ex);
    } catch (SQLException ex) {
        Logger.getLogger(HomePage.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

This function called up when **the User is goes to Profile Page and in the table, it displays all the products that the user has brought over the period of time** as we can see in **Image** 5 there are **Three Columns** where the user can see the details of his /her product such as its **Order ID and Product Name followed by the Date that the order was made.**

MySQL Query

```
>> select * from orders join product where product.product_id=orders.product_id and
orders.User_ID = ?
```

Purchase Button (Profile.java)

```
private void Back_ButtonActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_Back_ButtonActionPerformed
    // K Srikanth 17ETCS002124
    HomePage someobj = new HomePage();
    someobj.setVisible(true);
}//GEN-LAST:event_Back_ButtonActionPerformed
```

This function called up when the **Back button is clicked** and it basically **redirects the User** to **Home Page if the User wishes to go back.**

B2.3)

Results

Scenario 1: (User Buying a Product)

In this scenario the user buys the product so to do that the user has to create an account let's click on create button to create a new account

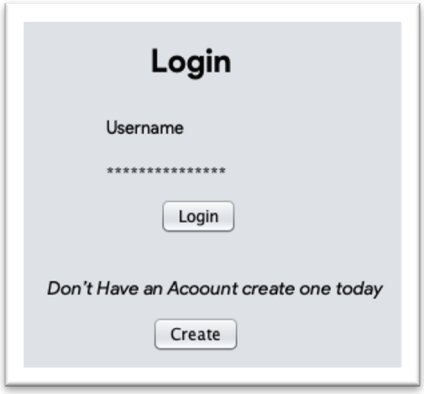


Figure 22 Swing UI when the Application launches

Now that we clicked **on "create" button** it will take the user to **signup page** where user can provide his/her details and then **click "create".**

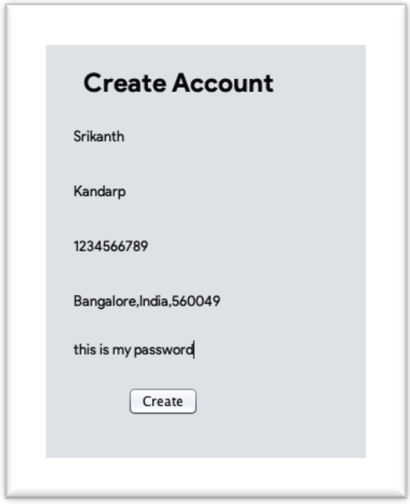


Figure 23 Swing UI when user is creating an account

After creating an account, the user will recive a message **"Your Account is created"** after a successful creation and user can now login with his **username sent to his phone.**

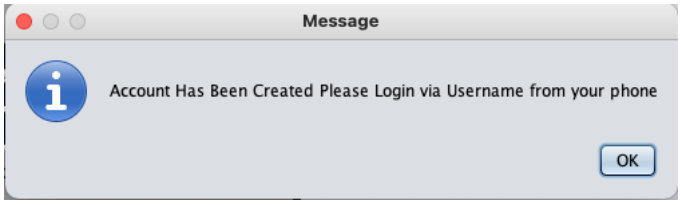


Figure 24 Swing Dialog Box after account successfully created

As we can see in the database the User is added to the database with **User Name "18"**

```
+---------+--------------------+------------+-----------+--------------+------------------------+
| User_ID | Password           | First_Name | Last_Name | Phone_Number | Address                |
+---------+--------------------+------------+-----------+--------------+------------------------+
|      18 | this is my password | Srikanth   | Kandarp   |   1234566789 | Bangalore,India,560049 |
+---------+--------------------+------------+-----------+--------------+------------------------+
1 row in set (0.00 sec)
```

Figure 25 MySQL Result that Data has been entered successfully.

So, now that our Account it created lets **login via the credentials.**

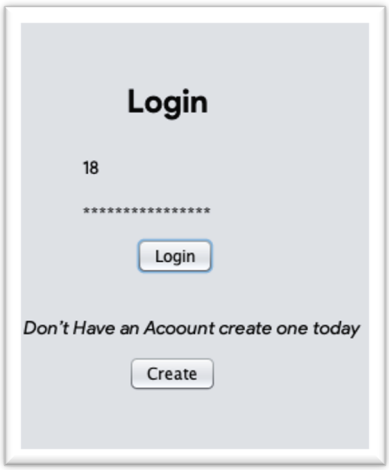


Figure 26 User Logging in using his credentials

Now after entering the credentials the system will **verify the credentials**, they match with the one in database then it will further proceed to **Home Page.**

Now that our credentials are verified, we are **now in-home page of the application.**

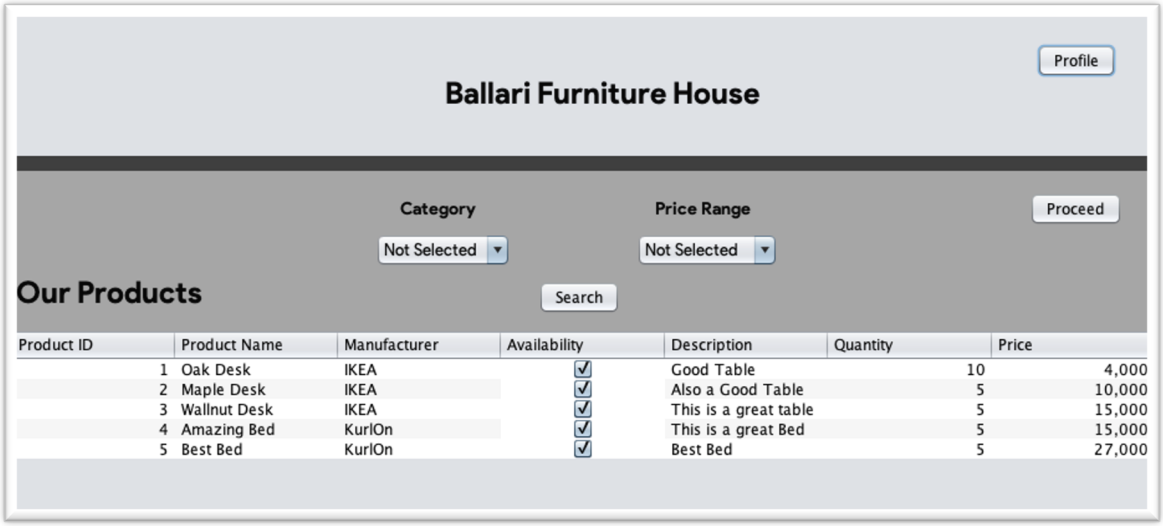


Figure 27 Home Page of the Application

As you can see in **Image 22** the user can see **all the products that are available** in store and can decide accordingly.

Okay, now let's **select a product and purchase it,** I think I am going to go with **Oak Desk** as I am in desperate need of a new desk because it can help me submit my assignment way faster. Lol and click on Proceed Button.

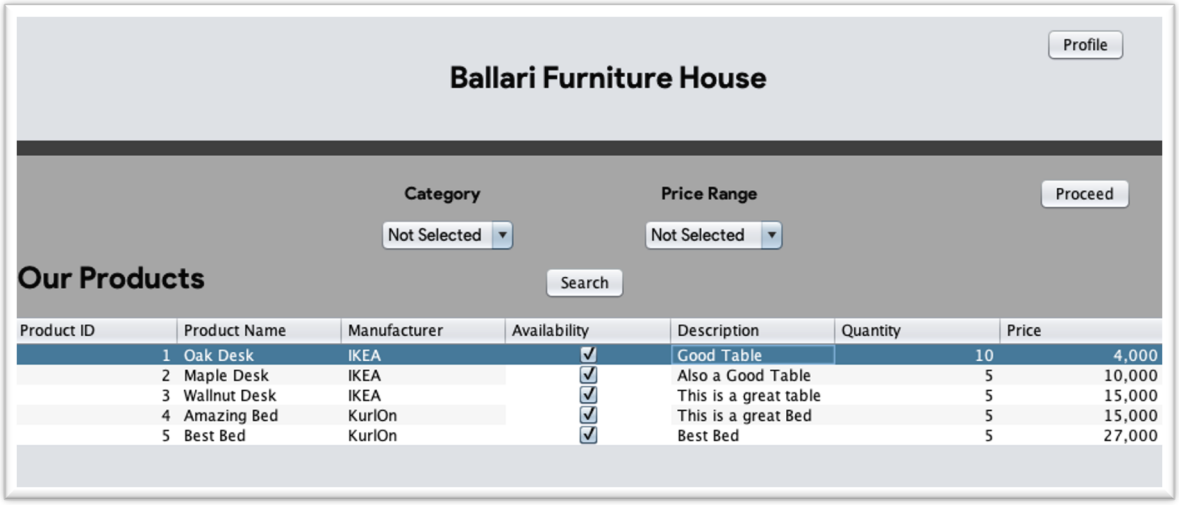


Figure 28 User Selecting "Oak Desk" and Clicking Proceed

Now that we clicked on Proceed Button the User is Redirected to Cart Page where he can see his **"Good Table"**

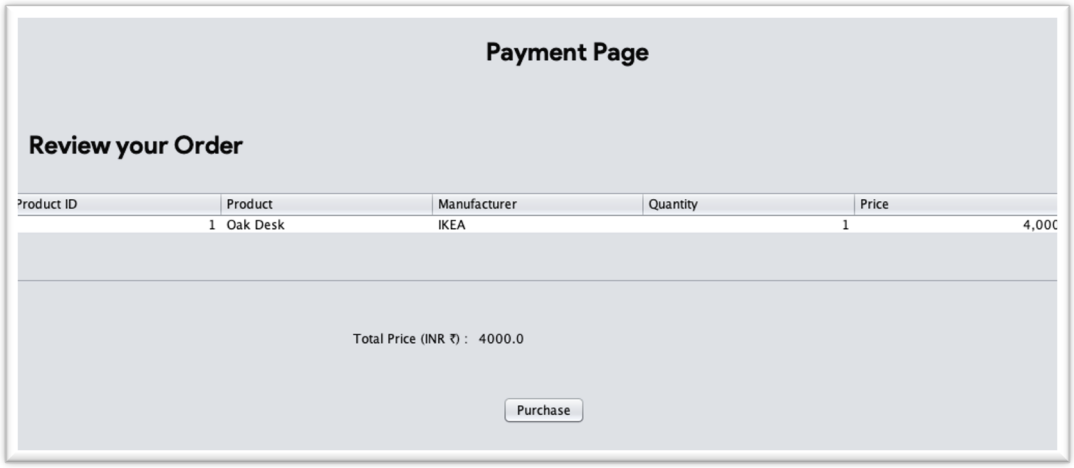


Figure 29 User Reviewing his order

As you can see **in Image 24** the user can see all the products that he selected to buy. (Because I am low on budget, I think I am going to go with only one desk). So, the total **amount is 4000.00** and then the user clicks on purchase and when he clicks on that he will be prompting with this message **"You Order was Placed Successfully. Thank You for Shopping with Us."**

Figure 30 Dialog when Payment is Successful

And when you click on "ok" the user will be redirected to Home Page.

Scenario 2: (User Entering Wrong Username or Password)

In this scenario let's not type in the right username or password and see what happens

Figure 31 User entering wrong credentials

As we can see the system lets you know that **the credentials that you entered were wrong** and when you click on **"okay"** it will let you try again.

Figure 32 Dialog when credentials are incorrect.

Scenario 3: (User Search's for a Product)

In this scenario after logging in the user wishes to search for the product in the Home Page

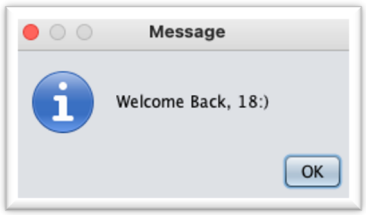


Figure 33 Dialog when credentials are valid as User Logs in.

Now, let's assume that the user wants to buy a bed with his budget being more than 25000 INR, and the product can be selected and proceeded further to purchase.

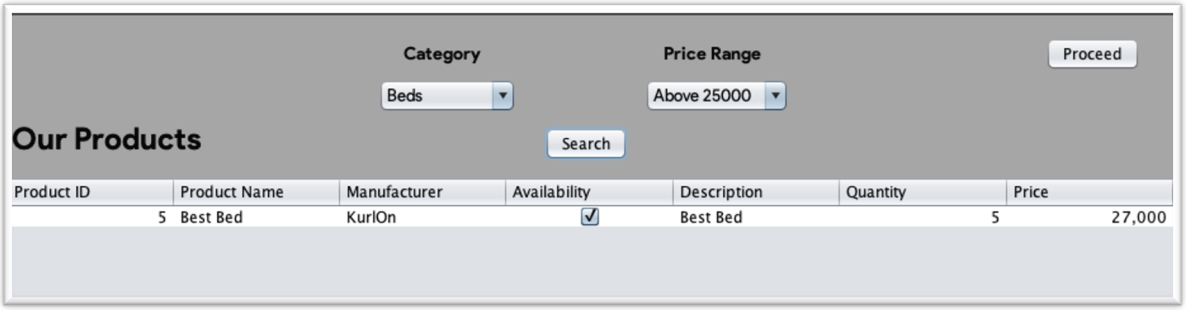


Figure 34 Application displaying the search result

Now that we clicked on Proceed Button the User is Redirected to Cart Page where he can see his **"Best Bed"** and purchase it via **Purchase Button.**

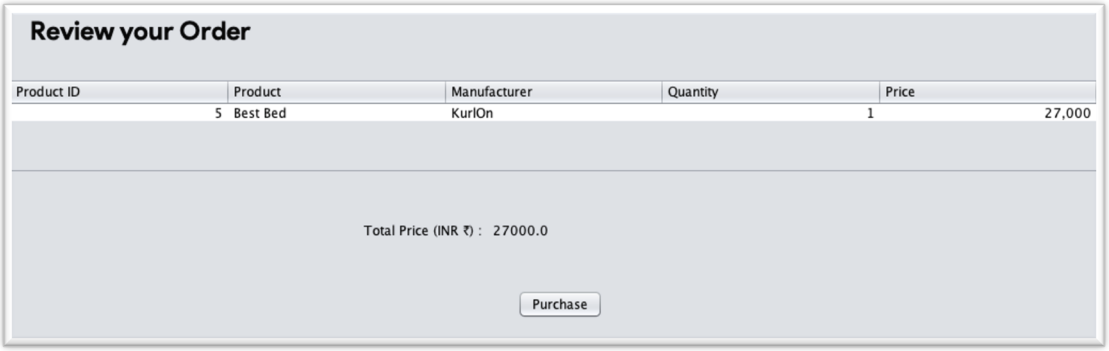


Figure 35 User Reviewing his order

As we can see in the database User's Order is added to the database with **Product Name "KurlOn ".**

```
mysql> select * from cart;
+------------+--------------+--------------+----------+---------------+
| Product_ID | Product_Name | Manufacturer | Quantity | Product_Price |
+------------+--------------+--------------+----------+---------------+
|          5 | Best Bed     | KurlOn       |        1 |      27000.00 |
+------------+--------------+--------------+----------+---------------+
1 row in set (0.00 sec)
```

Figure 36 MySQL Result that Data has been entered successfully.

Scenario 4: (User Updates his/her Profile)

From the Home Page when the User clicks on the Profile Button it Redirects the User to Profile Page where he/she can update their profile.

Figure 37 User Updating Profile

After the User Clicks on the **Update Button** a Dialog Message pop up saying "**Hey, User Your Account has been Updated.**"

Figure 38 Dialog when User Updates his/her Profile.

As we can see in the database the User is updated on the database with **User Name "18".**

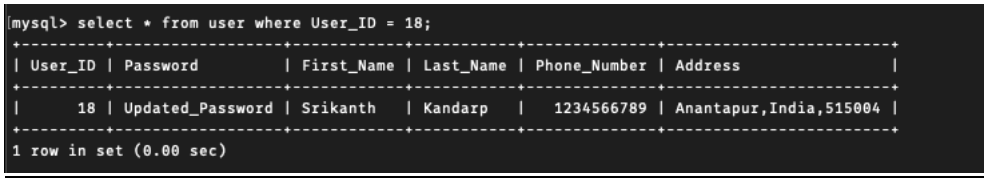


Figure 39 MySQL Result that Data has been Updated successfully.

Scenario 5: (User checks his/her Recent Orders)

From the Home Page when the **User clicks on the Profile Button** it Redirects the User to **Profile Page** where he/she can see their **Recent Orders**.
As we can see **the User "18" has ordered a "Oak Desk and Best Bed" on 20th January 2021 with Order ID being 17 and 18.**

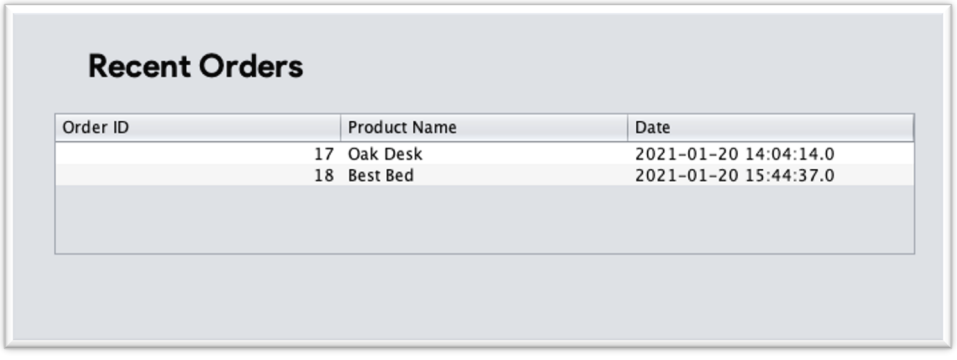


Figure 40 Recent Orders of User "18"

As we can see in the database the User is updated with his Orders.

Figure 41 MySQL Result that Order was placed successfully.

B2.4)

Summary

- **Overview of the Product**

Ballari Furniture House is an Online Furniture Shop built using **Java Swing UI and MySQL.** Which allows the Users to create an account **(Image 7)** to purchase a **Furniture Product and** after creating an account the **user can login (Image 6)** to proceed to the **Home Page (Image 8)** and can view all the products that are available in the Shop. If the **user** wishes to search the product by its category and price range then the **Application allows the user to do that.** After selecting the product **user can checkout which will take the user** to Payment Page **(Image 9)** where **the user can review his/her order and make the payment** later he/she can see it in his **Profile recent orders.**

- **Connection**

The Connection was done using JDBC (Java Database Connectivity) to connect the GUI and Database JDBC offers a programming-level interface that handles the mechanics of Java applications communicating with a database or RDBMS. The JDBC API supports communication between the Java application and the JDBC manager.

- **Using Temporary Tables**

Temporary tables are used to exchange or store the intermediate results of your data the best example would be cart in an online application where the system has to store the user cart data and it's not a constant so using a temporary table can help us perform this operation.

Creating a Temporary Table,

MySQL Query

```
>> Create TEMPORARY TABLE cart (
Product_ID int NOT NULL,
Product_Name VARCHAR (50) NOT NULL,
Manufacturer VARCHAR (50) NOT NULL,
Quantity int NOT NULL,
Product_Price double(10,2) NOT NULL);
```

Metadata of the Temporary Table

```
mysql> desc cart;
+---------------+--------------+------+-----+---------+-------+
| Field         | Type         | Null | Key | Default | Extra |
+---------------+--------------+------+-----+---------+-------+
| Product_ID    | int          | NO   |     | NULL    | NULL  |
| Product_Name  | varchar(50)  | NO   |     | NULL    | NULL  |
| Manufacturer  | varchar(50)  | NO   |     | NULL    | NULL  |
| Quantity      | int          | NO   |     | NULL    | NULL  |
| Product_Price | double(10,2) | NO   |     | NULL    | NULL  |
+---------------+--------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

Figure 42 Metadata of Cart (Temporary Table)

As you can see when you create the Table it is not stored in the database,

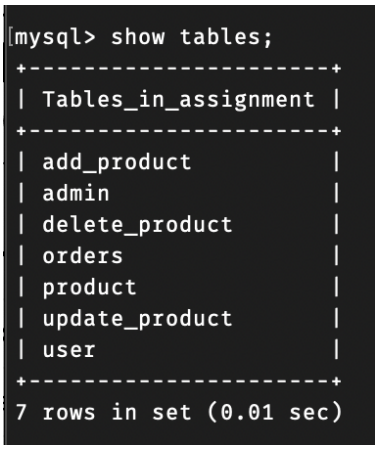


Figure 43 MySQL tables after creating "cart" temporary table

The Reason that the table is not in the database, is because it creates the table in the memory pool where when your memory is shutdown the data is lost and when the user logs in back again the system has to create the table and drop after the user buys the product. This is how Temporary Table works in a nutshell.

Limitations

Firstly, Users can add only one quantity of the product in the cart but its restricted one product per user yet user can still add multiple different products but not same product. Secondly, Multiple users cannot login at the same time if a user is logged in already. The User Name is not sent to Customer's Phone as there is no API call that handles that request.

Improvements

After Completing this assignment I've felt like my system design wasn't the best. There were few things that were lacking if another user comes online the system can't process his/her cart because of there is only one cart but this could be overcome by storing the cart in the user local memory as an object or as an array but using temporary table did overcome this issue but if multiple users cannot access the table it if some user is accessing it. As mentioned in the limitation's user can't customize their quantity. I felt like that could have been added. Handling Multiple User can be an improvement in order to do that we need a server when can handle all the requests made by the users to process the queries and API calls.