**Class: CustomerController.java**

Function: addingProPic

Parameters: ActionEvent

Returns: void

Purpose: Adding a profile picture of various formats for users.

1 void addingProPic**(**ActionEvent event**)** **{**

2

3

4 FileChooser fileChooser **=** **new** FileChooser**();**

5

6 //Setting extension filter

7 FileChooser**.**ExtensionFilter extFilterJPG **=**

8 **new** FileChooser**.**ExtensionFilter**(**"JPG files (\*.JPG)"**,** "\*.JPG"**);**

9 FileChooser**.**ExtensionFilter extFilterjpg **=**

10 **new** FileChooser**.**ExtensionFilter**(**"jpg files (\*.jpg)"**,** "\*.jpg"**);**

11 FileChooser**.**ExtensionFilter extFilterPNG **=**

12 **new** FileChooser**.**ExtensionFilter**(**"PNG files (\*.PNG)"**,** "\*.PNG"**);**

13 FileChooser**.**ExtensionFilter extFilterpng **=**

14 **new** FileChooser**.**ExtensionFilter**(**"png files (\*.png)"**,** "\*.png"**);**

15 fileChooser**.**getExtensionFilters**()**

16 **.**addAll**(**extFilterJPG**,** extFilterjpg**,** extFilterPNG**,** extFilterpng**);**

17

18 //Showing open file dialog

19 File file **=** fileChooser**.**showOpenDialog**(null);**

20

21 **try** **{**

22 BufferedImage bufferedImage **=** ImageIO**.**read**(**file**);**

23 Image image **=** SwingFXUtils**.**toFXImage**(**bufferedImage**,** **null);**

24 customerImg**.**setImage**(**image**);**

25 proPicLbl**.**setVisible**(false);**

26 **}** **catch** **(**IOException ex**)** **{**

27 Logger**.**getLogger**(**CustomerController**.**class**.**getName**()).**log**(**Level**.**SEVERE**,** **null,** ex**);**

28 **}**

29 **}**

Figure: Code Snippet (adding profile picture)

Description:

Here, in this method, we have shown a way to add a profile picture for the user’s account, and this act of adding a profile picture is going to be triggered as soon as the user clicks on the button called ‘Add Image’. To set an extension filter we used the FileChooser class, which will find any image of the mentioned types, (i.e. .jpg, .png, etc regardless of the sensitivity of characters (lines 7-16) from the local drive. To open the file dialog, the file chooser instance(19th line) has been used, and then in the try block, we set the image (24th line) as well as simultaneously we made the label invisible (25th line) too. After that, we used a catch block for handling errors during this procedure.

When all these will be done, the chosen photo will be added to the specific place , selected earlier during building the UI. Then the picture will be visible to the user and the outlook of the photo is given here.

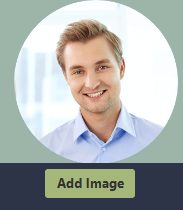


Figure: Result of adding profile picture

**Class: Ecommerce.java**

Function: Pagination call

Parameters: none

Purpose: Keep sliding all the selected products in the front page.

1 Pagination call**(){**

2

3 String**[]** photos **=** **{** "img/4.png"**,**"img/5.png"**,**"img/1.png"**,**"img/2.png"**,**"img/3.png"**,**"img/7.png"**};**

4 Pagination p **=** **new** Pagination**(**photos**.**length**);**

5

6 p**.**setPageFactory**((**Integer pageIndex**)** **->** **{**

7 **return** **new** ImageView**(**getClass**().**getResource**(**photos**[**pageIndex**]).**toExternalForm**());**

8 **});**

9

10 p**.**getStyleClass**().**add**(**Pagination**.**STYLE\_CLASS\_BULLET**);**

11

12 Timeline fiveSecondsWonder **=** **new** Timeline**(new** KeyFrame**(**Duration**.**seconds**(**3**),** event **->** **{**

13 int pos **=** **(**p**.**getCurrentPageIndex**()+**1**)** **%** p**.**getPageCount**();**

14 p**.**setCurrentPageIndex**(**pos**);**

15

16 **}));**

17 fiveSecondsWonder**.**setCycleCount**(**Timeline**.**INDEFINITE**);**

18 fiveSecondsWonder**.**play**();**

19 **return** p**;**

20 **}**

Figure: Pagination snippet

Description:

The following code has been used to continuously slide images of relevant products for every 3 seconds. At first, on line 3 we made a string of photos where we inserted all the names of those photos we want to be sliding. Then on line 4, we created an object of pagination type where we also loaded the length of the string of photos. From line 6 to 10, we set our index to the very first image of the string and made it viewable. On line 10, we chose a better design for the bullets that will be used to change the sliding photos manually. On line 12, we set the duration of sliding images for 3 seconds, that means the photos we added in out string at line 3 will be kept sliding automatically after every 3 seconds. On line 17 we made it constant for an infinite time. Finally, we displayed it in our default home page.



After 3 seconds…

This is the result of the following code snippet. The photos will be changed automatically after 3 seconds and they will go round the 6 photos. But what if we want to change them manually? Yes, to do so, we have to click on the bullets locating at the bottom of those photos.

The photo is showing 2/6, that means, the 2nd photo among the 6 is now on the view and it will stay for 3 seconds. If we click on the right arrow bullet it will be changed to 3/6 and if we click on the left arrow bullet then the 1/6 will be displayed.

**Class: HomeController.java**

Function: logging

Parameters: ActionEvent

Purpose: Verifying the users credentials and giving access to the authentic users.

1 @FXML

2 public void logging**(**ActionEvent event**)** **throws** SQLException **{**

3 //Processing Icon Play()

4 progress**.**setVisible**(true);**

5 PauseTransition pt**=** **new** PauseTransition**();**

6 pt**.**setDuration**(**Duration**.**seconds**(**3**));**

7 pt**.**play**();**

8 pt**.**setOnFinished**(new** EventHandler**<**ActionEvent**>()** **{**

9

10 public void handle**(**ActionEvent ev**)** **{**

11 progress**.**setVisible**(false);**

12 Connection con **=** **null;**

13 PreparedStatement pst **=** **null;**

14 Statement \_deleteTableDtataStmt **=** **null;**

15 ResultSet rs **=** **null;**

16

17 **if(**mailh**.**getText**().**equals**(**"admin"**)&&** passh**.**getText**().**equals**(**"111222"**)){**

18

19 ///loading admin

20 FXMLLoader fxmlLoader **=** **new** FXMLLoader**(**getClass**().**getResource**(**"admin.fxml"**));**

21 Parent roo **=** **null;**

22 **try** **{**

23 roo **=** **(**Parent**)** fxmlLoader**.**load**();**

24 **}** **catch** **(**IOException ex**)** **{**

25 Logger**.**getLogger**(**HomeController**.**class**.**getName**()).**log**(**Level**.**SEVERE**,** **null,** ex**);**

26 **}**

27 Stage stage **=** **new** Stage**();**

28 stage**.**setMaximized**(true);**

29 stage**.**setScene**(new** Scene**(**roo**));**

30 stage**.**show**();**

31 **}**

32 //...snip...

Figure: admin log in

Description:

The code on line 3 to 7 is responsible for displaying the progress bar for 3 seconds. When 3 seconds will be over then a new event will be started. From line 11 to 15, we declared some variables which will be used later on for connecting to the database. On line 17, this is a conditional statement for checking the authenticity of the user. If the mail address entered by the user is ‘admin’ and the password is ‘111222’ then they user will gain the access as the admin. As a result, the admin panel will be visible on the screen.

34 **else{**

35 String sql**=**"select \* from customer where email=? and password=?" **;**

36 **try{**

37 con **=** DriverManager**.**getConnection**(**"jdbc:mysql://localhost/ecommerce?autoReconnect=true&useSSL=false"**,**"root"**,**"12345"**);**

38 pst**=**con**.**prepareStatement**(**sql**);**

39 pst**.**setString**(**1**,** mailh**.**getText**());**

40 pst**.**setString**(**2**,**passh**.**getText**());**

41 rs**=**pst**.**executeQuery**();**

42 Statement stmt **=** con**.**createStatement**();**

43

44 //...snip...

On the other hand, if the entered email and password do not get matched with the admin panel then we need to check it from the database because the user might have entered his/her credentials as a registered user. On line 35, we have created SQL statement to fetch the data of ‘customer’ table where we will get their emails and passwords. But initially we need to connect to the MySQL server. Connection has been stablished using the code on line 37 where the user name is ‘root’ and password is ‘12345’. We used these credentials during the installation of MySQL server in our computer. We used MySQL database because MySQL is a relational database management system based on SQL – Structured Query Language. The application is used for a wide range of purposes, including data warehousing, e-commerce, and logging applications, for more details, please visit: <https://www.123-reg.co.uk/support/servers/what-is-mysql-and-why-do-i-need-it/>.

This is how we will give access to the authentic user (line 38 to 42) so that, they can choose and buy their desired products. If the entered email and password have no match with the database then a notification will appear on the screen.

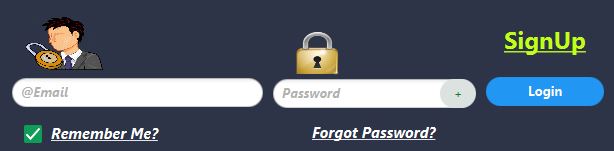


Figure: Authentication

This is the figure where the user may enter their respective credentials. There is an email field and a password field. As soon as the user will enter these fields, they should hit enter button from keyboard or click on the ‘Login’ button to get pass. If they make any mistake during this progress, a notification will be visible there and a progress bar will be visible for 3 seconds as mentioned earlier.

**Class: SignUpController.java**

Function: RunVerEmailbtn

Parameters: ActionEvent

Purpose: Sending random code for email verification.

1 Random rnd **=new** Random**();**

2 int code**=**rnd**.**nextInt**(**999999**-**100000**)+**100000**;**

3 codd**=**code**;** //static int codd;

4

5 final String username **=** "your email address"**;** // enter the senders email

6 final String password **=** "enter your password"**;**// enter the corresponding password

7

8 Properties props **=** **new** Properties**();**

9 props**.**put**(**"mail.smtp.auth"**,** "true"**);**

10 props**.**put**(**"mail.smtp.starttls.enable"**,** "true"**);**

11 props**.**put**(**"mail.smtp.host"**,** "smtp.gmail.com"**);**

12 props**.**put**(**"mail.smtp.port"**,** "587"**);**

13

14 Session session **=** Session**.**getInstance**(**props**,**

15 **new** javax**.**mail**.**Authenticator**()** **{**

16 protected PasswordAuthentication getPasswordAuthentication**()** **{**

17 **return** **new** PasswordAuthentication**(**username**,** password**);**

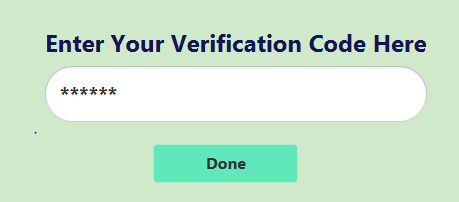
18 **}**

19 **});**

Description:

On line 1, we created a random type variable (‘rnd’) and on the next line we generated a random number. This number will be sent to the user to verify his/her authenticity. The variable ‘codd’ was initially declared as static integer. On line 5, we assigned an email address which will act as the sender during verification and the password variable is the corresponding password to the sender’s email. From line 8 to 12, we initiated all the needful to activate SMTP (Simple Mail Transfer Protocol) where we used the port number 587. Generally, port 587 is used as the default port number for SMTP during transferring emails. At the end segment we created a session to use for that time being.





As soon as the user will fill-up the signup form successfully and clicks on the ‘Very Email’ button, a new section will appear on the right side for further processing. It will be asked to enter the verification code that has been sent to the respective email given by user. The user needs to login his/her Gmail account, collect or copy the code and paste it there. Then he/she should click on the ‘Done’ button.

**Class: SignUpController.java**

Function: verDoning

Parameters: ActionEvent

Purpose: This method will not only verify the entered code number but also save the necessary documents in the database that was put down by the user during signup.

1 @FXML

2 void verDoning**(**ActionEvent event**)** **{**

3

4 String getstr**=**verCode**.**getText**();**

5

6 int makeint**=**Integer**.**parseInt**(**getstr**);**

7

8 **if(**makeint **==** codd**){**

9 sorryLbl**.**setVisible**(false);**

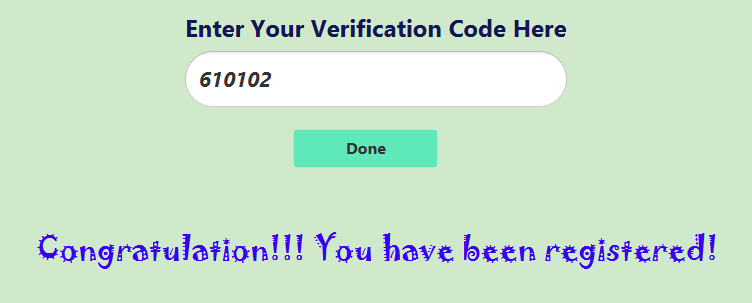
10 System**.**out**.**println**(**codd**);**

11 congrast**.**setVisible**(true);**

12 // snip....

Description:

If the user enters the exact code that was sent to his/her email account then a label saying congratulation will appear. On the other hand, if it does not match then another relevant label will appear.



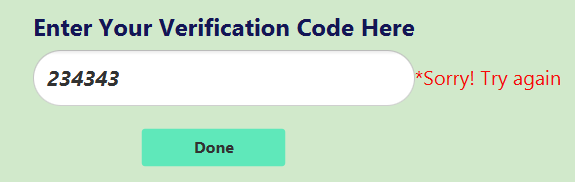
 Figure: Trying with wrong password

Figure: Trying with correct password

So, from the above figures it is evident that, if the user tries using wrong password then they will see a label ‘Sorry Try again’ written on it. But if he/she gives the correct code collecting from their email then they will be congratulated since he/she has been registered successfully.

Database connection:

1 conn **=** DriverManager**.**getConnection**(**"jdbc:mysql://localhost:3306/ecommerce?autoReconnect=true&useSSL=false"**,**user**,** pass**);**

2

3 //insert statement

4 String query **=** " insert into customer(firstName,lastName,email,password,city,country)"

5 **+** " values (?, ?,?,?,?,?)"**;**

6

7 // create the mysql insert preparedstatement

8 PreparedStatement preparedStmt **=** conn**.**prepareStatement**(**query**);**

9 preparedStmt**.**setString **(**1**,** Fname**.**getText**());**

10 preparedStmt**.**setString **(**2**,** Lname**.**getText**());**

11 preparedStmt**.**setString **(**3**,** Email**.**getText**());**

12 preparedStmt**.**setString **(**4**,** Password**.**getText**());**

13 preparedStmt**.**setString **(**5**,** city**.**getText**());**

14 preparedStmt**.**setString **(**6**,** country**.**getText**());**

15

16 // execute the preparedstatement

17 preparedStmt**.**execute**();**

18

19 conn**.**close**();**

20 //snip...

The following code is responsible for connecting to the database and inserting data in it. Initially we stablished connection with the ecommerce database using port 3306. We shall save the user’s first name, last name, email, password, city and country in the database. Then we executed the following command and when everything was done then we closed the connection.

**Class: CustomerController.java**

Function: Searching

Parameters: ActionEvent

Purpose: This method will be activated when the user will click on the ‘Go’ button for searching purpose and it will also show the relevant search result on the screen.

1 @FXML

2 void searching**(**ActionEvent event**)** **{**

3 sch **=** searchBar**.**getText**();** // public String sch;

4

5 **if(**sch**.**equals**(**""**)){**

6 System**.**out**.**println**(**"null"**);**

7 **}**

8 **else{**

9 **try** **{**

10 img2Name**.**setVisible**(false);**

11 img2Model**.**setVisible**(false);**

12 img2Price**.**setVisible**(false);**

13 img2**.**setVisible**(false);**

//snip ...

Description:

Initially, we created a variable named ‘sch’ as public String. Then we assigned there the text received from the search bar entered by the user. So, at first, we checked whether the user is trying with ‘null’ value or not (visible on line 5). After that we accomplished the trickiest part. We have 4 products on the screen. But when the user will search for product then only one relevant product will be visible on screen. Rest of the products (3 out of 4) will become invisible. That’s why we had to make visibility false for all other products. We did this for all the corresponding elements to that particular set i.e. name, model, price as well as the image. A demo image has been given below for better clarity,

**Visibility: False**

**Visibility: False**

**Visibility: True**

**Visibility: False**

That is how the most relevant product will be displayed on the screen. The searched product will be displayed by replacing the position of product 1 (green colored box in the image).

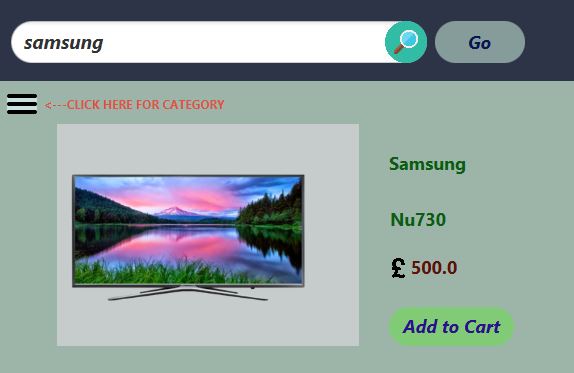


Figure: Search bar

That is the user interface of search bar where the user will type the product name they want to search for and then the user should click on the ‘Go’ button.

All the products and relevant information are saved in the database. So, we need to connect to the database. We have various products available category wise. As a result, when a specific product will be searched, it will go through all the categories and find out the most pertinent one on the screen.

For example:



Here on the figure the user has typed for ‘samsung’ and then when he will click on the ‘Go’ button, all the other products will be invisible and this will also trigger to connect to the database. So the following code will be run,

1 // 1. Get a connection to database

2

3 conn **=** DriverManager**.**getConnection**(**"jdbc:mysql://localhost:3306/ecommerce?autoReconnect=true&useSSL=false"**,**user**,** pass**);**

4

5 // 2. Create a statement

6 stmt **=** conn**.**createStatement**();**

7

8 // 3. Execute SQL query

9 rs **=** stmt**.**executeQuery**(**"select \* from product where prodName REGEXP '"**+**sch**+**"' "**);**

10 **while** **(**rs**.**next**())** **{**

11

12 img1Name**.**setText**(**rs**.**getString**(**"prodName"**));**

13 img1Model**.**setText**(**rs**.**getString**(**"prodModel"**));**

14 img1Price**.**setText**(**rs**.**getString**(**"prodPrice"**));**

15 InputStream is **=** rs**.**getBinaryStream**(**"prodPhoto"**);**

16 InputStream input **=** **new** ByteArrayInputStream**(**rs**.**getBytes**(**"prodPhoto"**));**

17 Image imge **=** **new** Image**(**input**);**

18 img1**.**setImage**(**imge**);**

19 //snip....

On line 3, we made a connection with the ecommerce database using the username and password. Then from the product table of the database we searched for ‘prodName’. Here the ‘prodName’ is the name of the product entered by the user for searching purpose. This loop will continue running until it finds the most relevant one.

When there will be a match, it will fetch ‘prodName’, ‘prodModel’, ‘prodPrice’, and ‘prodPhoto’ (from line 12 to 15) and these will be set on the screen. For the following search, we used ‘REGEXP’ keyword (on line 9). This keyword provides a powerful and flexible pattern match that can help us implement power search utilities for our database system more over it is not case sensitive. To know more about the ‘REGEXP’ keyword please visit: <https://www.geeksforgeeks.org/mysql-regular-expressions-regexp/>

Example:

Here on the photo, the user has typed only ‘t-‘. Now we need to notice that ‘t’ is a small letter. The user has not typed the full name of the product! But when he/she clicks on the ‘Go’ button, it brings out the most appropriate product on the screen. In that case that is ‘T-shirt’.

