1. **INTRODUCTION**

Furniture is what turns your house into a home. Well-designed furniture is not just functional or utilitarian (for example, tables and desks for work or cabinets for storage) but also a reflection of your taste and style. We have hundreds of furniture designs in solid wood, engineered wood, metal, etc., as well as soft furnishing items.Urban Furnish offers a unique selection of stylish, contemporary, and chic [furniture online](https://www.urbanladder.com/?src=g_seo_hpcontent_hplink). Our online furniture range includes Dressing tables, S[ofa sets](https://www.urbanladder.com/sofas-by-material?src=g_seo_hpcontent_sofas), Cots and B[eds](https://www.urbanladder.com/double-beds?src=g_seo_hpcontent_beds), [TV units](https://www.urbanladder.com/tv-units?src=g_seo_hpcontent_tv_units), Swings, and lots more. All our wooden furniture designs are available online at urban furnish portal. On our portal, you can browse as many furniture designs across categories as you like, easily select and buy the ones you like, and stay updated about new and contemporary designs.They can also place the order of their desire design making their life more comfortable and luxurious. Also it will be convenient for the furniture shop owner, as they do not have to decorate inside of the store to display the furniture. This also saves lot of money and labor work.

This application showcases all the furniture products for shopping, making the customer a comfortable online shopping experience with a real look and feel of the furniture. To buy products, the customer has to have an account.Once the customer authenticates his credentials, not only he can view the products, he can place an order to buy those products.This application then generates email regarding the payment details for that particular customer.

The objectives of this proposed web application system are:

• To computerize customer and admin database

• To maintain data consistency and integrity

• Automate the registration process without any physical human interaction

**2. FUNCTIONAL REQUIREMENTS**

**2.1 MVC Architecture**

MVC stands for Model, View and Controller. Figure 2.1 separates application into three components - Model, View and Controller.

**Model**: Model represents shape of the data and business logic. It maintains the data of the application. Model objects retrieve and store model state in a database. Model is a data and business logic.

**View**: View is a user interface. View display data using model to the user and also enables them to modify the data. View is a User Interface.

**Controller**: Controller handles the user request. Typically, user interact with View, which in-turn raises appropriate URL request, this request will be handled by a controller. The controller renders the appropriate view with the model data as a response. Controller is a request handler.

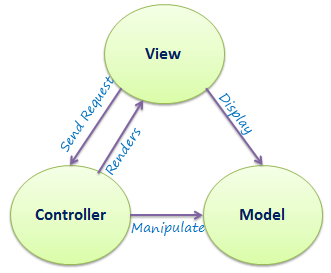
****

Figure 2.1 MVC Architecture

**2.2 Modules**

**Admin**

The administrative module is managed by the administrator. It is responsible for adding products

and order confirmation.

**User**

User should create an account to get access to the website and then can login with the credentials

given. User can view the products of different categories. After viewing, the user can order the

product the one which he like.

**Features of Proposed System:**

* Customers can shop from their shop
* Customers can view a large number of products available in store
* Customer reach for a shop is wider and customers around the globe can visit the store and purchase products

**2.3 System Requirements**

**Software Requirements**

Operating System : Windows 10

Programming Language : JAVA, JSP, HTML5

Tool : Eclipse Mars

OS Configuration : 64-bit

Database : MySql

**Hardware Requirements**

Processor : Intel i3

RAM : 4GB

**3. SYSTEM DESIGN**

**3.1 UML Diagrams**

**Unified Modeling Language:**

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

UML is specifically constructed through two different domains they are:

UML Analysis modeling, this focuses on the model and structural model views of the

system.

UML design modeling, which focuses on the behavioral modeling, implementation

modeling and environmental model views.

**Use Case Diagram:**

A UML use case diagram is the primary form of system/software requirements for a new

software program under developed. Figure 3.1 specifies the expected behaviour (what),

and not the exact method of making it happen (how). Use cases once specified can be

denoted both textual and visual representation (such as UML). A key concept of use case

modelling is that it helps us design a system from end user's perspective. It is an effective

technique for communicating system behaviour in the user's terms by specifying all

externally visible system behaviour.

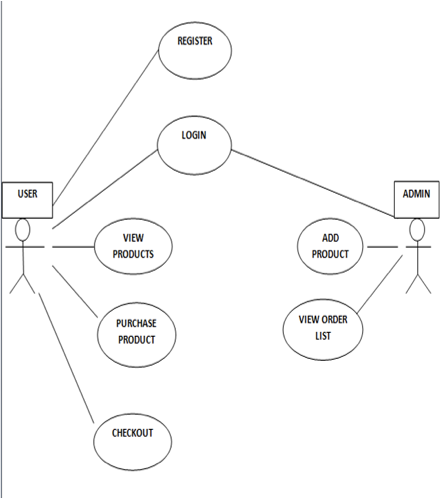


Figure 3.1: Use Case Diagram

**Activity Diagram:**

Figure 3.2 describe how activities are coordinated to provide a service which can be at different levels of abstraction. Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve a number of different things that require coordination, or how the events in a single use case relate to one another, in particular, use cases where activities may overlap and require coordination.

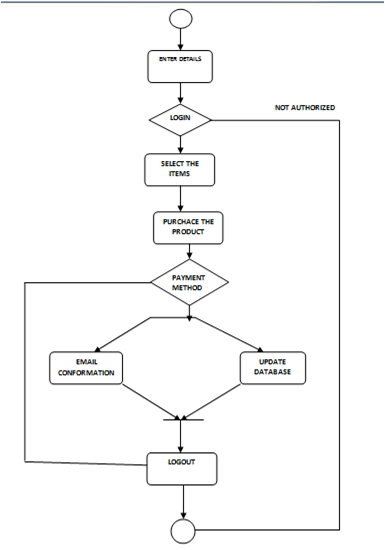
****

Figure 3.2: Activity diagram

**4.IMPLEMENTATION**

This application showcases all the furniture products for shopping, making the customer a comfortable online shopping experience with a real look and feel of the furniture. To buy products, the customer has to have an account.Once the customer authenticates his credentials, not only he can view the products, he can place an order to buy those products.This application then generates email regarding the payment details for that particular customer.

**4.1 Sample Code**

**loginDao.java**

package dao;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class logindao {

public boolean register(String firstname, String lastname,String youremail, String password ) throws SQLException {

boolean status = false;

try {

Class.forName("com.mysql.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/urban\_furnish", "root", "root");

PreparedStatement ps=con.prepareStatement( "insert into user\_details values(?,?,?,?)");

ps.setString(1,firstname);

ps.setString(2,lastname);

ps.setString(3,youremail);

ps.setString(4,password);

int i=ps.executeUpdate();

if(i>0) {

String sql = "select youremail from user\_details;";

Statement stat = con.createStatement();

stat.executeQuery(sql);

status = true;

}

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

return status;

}

public boolean validate(String youremail, String password) {

boolean status = false;

try {

Class.forName("com.mysql.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/urban\_furnish", "root", "root");

PreparedStatement ps = con.prepareStatement("Select youremail,password from user\_details where youremail=? and password=?");

ps.setString(1, youremail);

ps.setString(2, password);

ResultSet rs = ps.executeQuery();

if (rs.next()) {

status = true;

}

} catch (ClassNotFoundException | SQLException e) {

e.printStackTrace();

}

return status;

}

}

**Login.java**

package controller;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.SQLException;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import dao.logindao;

import dao.welcomedao;

@WebServlet("/login")

public class login extends HttpServlet {

private static final long serialVersionUID = 1L;

public login() {

super();

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.getWriter().append("Served at: ").append(request.getContextPath());

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

response.setContentType("text/html");

PrintWriter out = response.getWriter();

String youremail=request.getParameter("youremail");

String password=request.getParameter("password");

logindao ldao = new logindao();

boolean result = ldao.validate(youremail, password);

if (result) {

welcomedao weldao = new welcomedao();

try {

String name = weldao.getUserName(youremail);

HttpSession session = request.getSession();

session.setAttribute("name", name);

RequestDispatcher rd=request.getRequestDispatcher("Welcome.jsp");

rd.forward(request,response);

} catch (SQLException e) {

e.printStackTrace();

}

} else {

out.print("Sorry your email or password error!!");

RequestDispatcher rd=request.getRequestDispatcher("Login.jsp");

rd.include(request,response);

}

}

}

**Product.java**

package controller;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import dao.logindao;

import dao.productdao;

import dao.logindao;

@WebServlet("/product")

public class product extends HttpServlet {

private static final long serialVersionUID = 1L;

public product() {

super();

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.getWriter().append("Served at: ").append(request.getContextPath());

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

String product\_id= request.getParameter("product\_id");

String product\_name= request.getParameter("product\_name");

String category= request.getParameter("category");

String description = request.getParameter("description");

String price = request.getParameter("price");

String product\_image = request.getParameter("product\_image");

productdao pdao = new productdao();

try {

boolean result = pdao.product(product\_id, product\_name,category,description,price,product\_image);

if (result) {

out.println("sucessfully registered");

RequestDispatcher rdsp = request.getRequestDispatcher("adminwelcome.jsp");

rdsp.include(request, response);

} else {

RequestDispatcher rdsp = request.getRequestDispatcher("adminwelcome.jsp");

rdsp.include(request, response);

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

**SendEmail.jsp**

<%@ page import = "java.io.\*,java.util.\*,javax.mail.\*"%>

<%@ page import = "javax.mail.internet.\*,javax.activation.\*"%>

<%@ page import = "javax.servlet.http.\*,javax.servlet.\*" %>

String userEmailAddress = (String)request.getParameter("email");

String result;

final String from = "urbanfurnish123@gmail.com";

final String password = "urban\_furnish";

Properties properties = System.getProperties();

properties.setProperty("mail.smtp.host", "smtp.gmail.com");

properties.setProperty("mail.smtp.socketFactory.port", "465");

properties.setProperty("mail.smtp.socketFactory.class","javax.net.ssl.SSLSocketFactory");

properties.setProperty("mail.smtp.auth", "true");

properties.setProperty("mail.smtp.port", "465");

Session mailSession = Session.getInstance(properties,

new javax.mail.Authenticator() {

protected PasswordAuthentication getPasswordAuthentication() {

return new PasswordAuthentication(from, password);

}

});

try {

Message message = new MimeMessage(mailSession);

message.setFrom(new InternetAddress("urbanfurnish123@gmail.com"));

message.setRecipients(Message.RecipientType.TO,

InternetAddress.parse(userEmailAddress));

message.setSubject("Urban Furnish");

message.setText("Dear Customer," + "\n\n Your Order has been Confirmed, Thank You!"+"\n\n Your Order will be delivered to you within 7-15 days and do ensure you collect the bill and pay the amount by hand cash." );

Transport.send(message);

} catch (MessagingException e) {

throw new RuntimeException(e);

}

**4.2 Output Screenshots:**

Figure 4.1 is the first page that a user view as soon as the user visits the website. This includes the contact details of the owner and about the store.

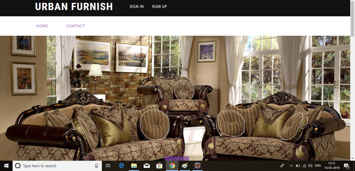
****

Figure 4.1: Home Page

Figure 4.2 says when the user clicks on signup button, the user views all the required fields and

then the user fills details and then automatically it redirects to login page.

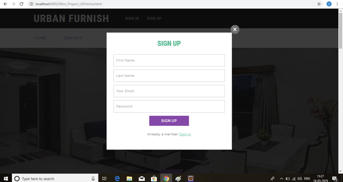
****

Figure 4.2: User Signup Page

Figure 4.3 shows how it validates with the registered details of the user and then redirects to the

profile page.

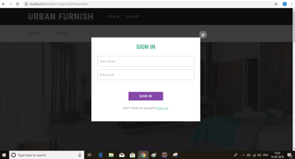
****

Figure 4.3: Login Page

Figure 4.4 describes that the user can view all the categories. And there are different types of

categories like dressing tables, sofa, wardrobes, and so on.

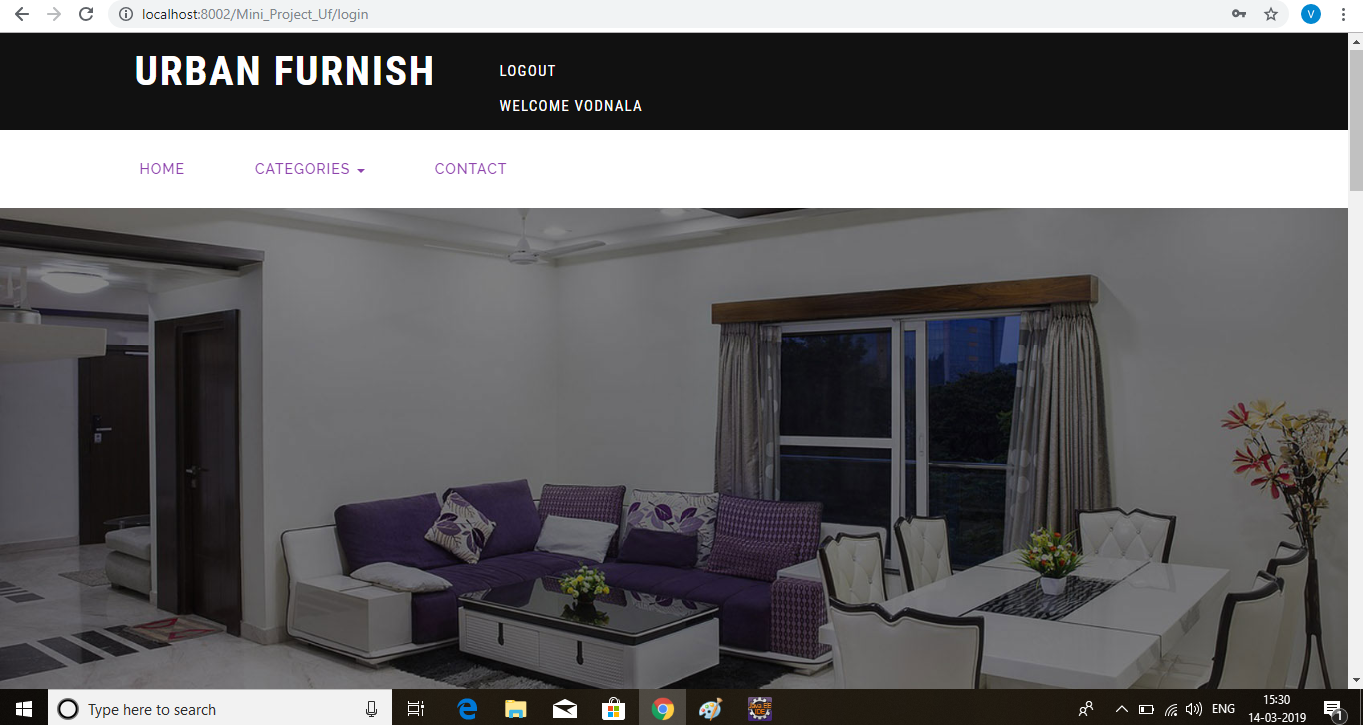
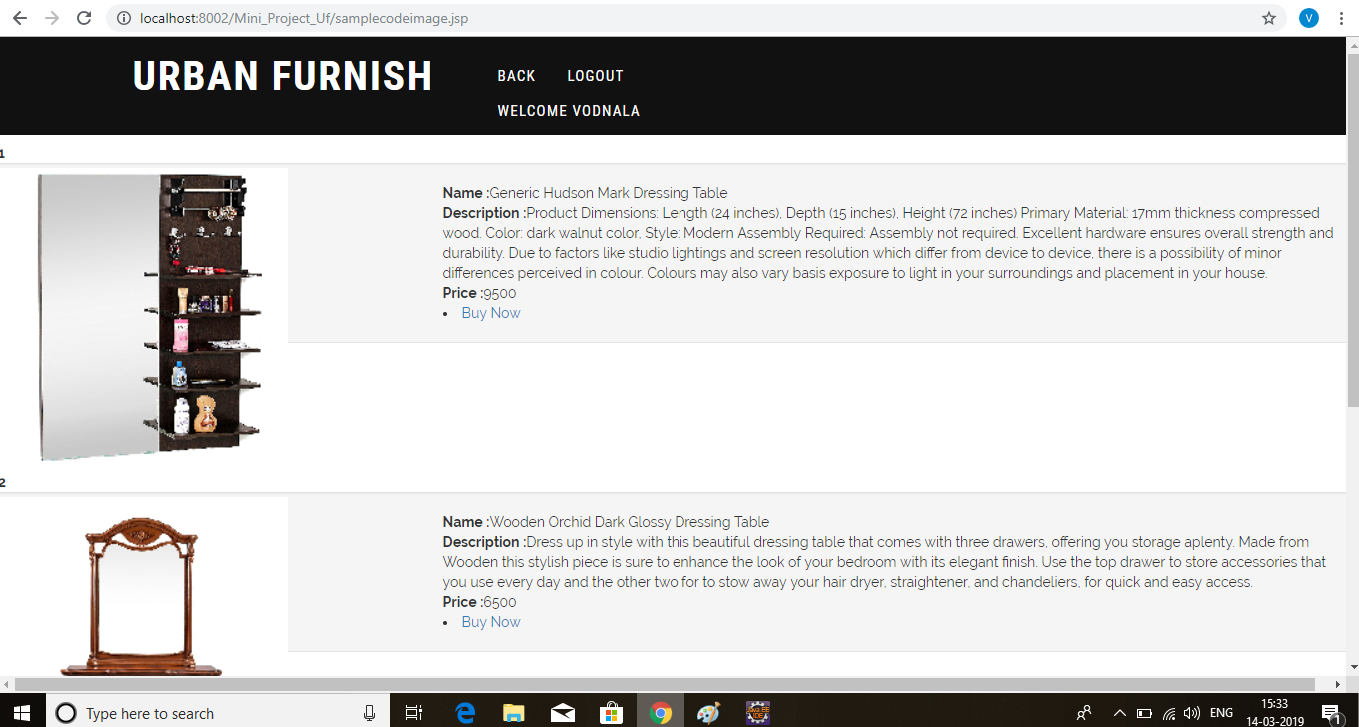
****

Figure 4.4: User Portal Page

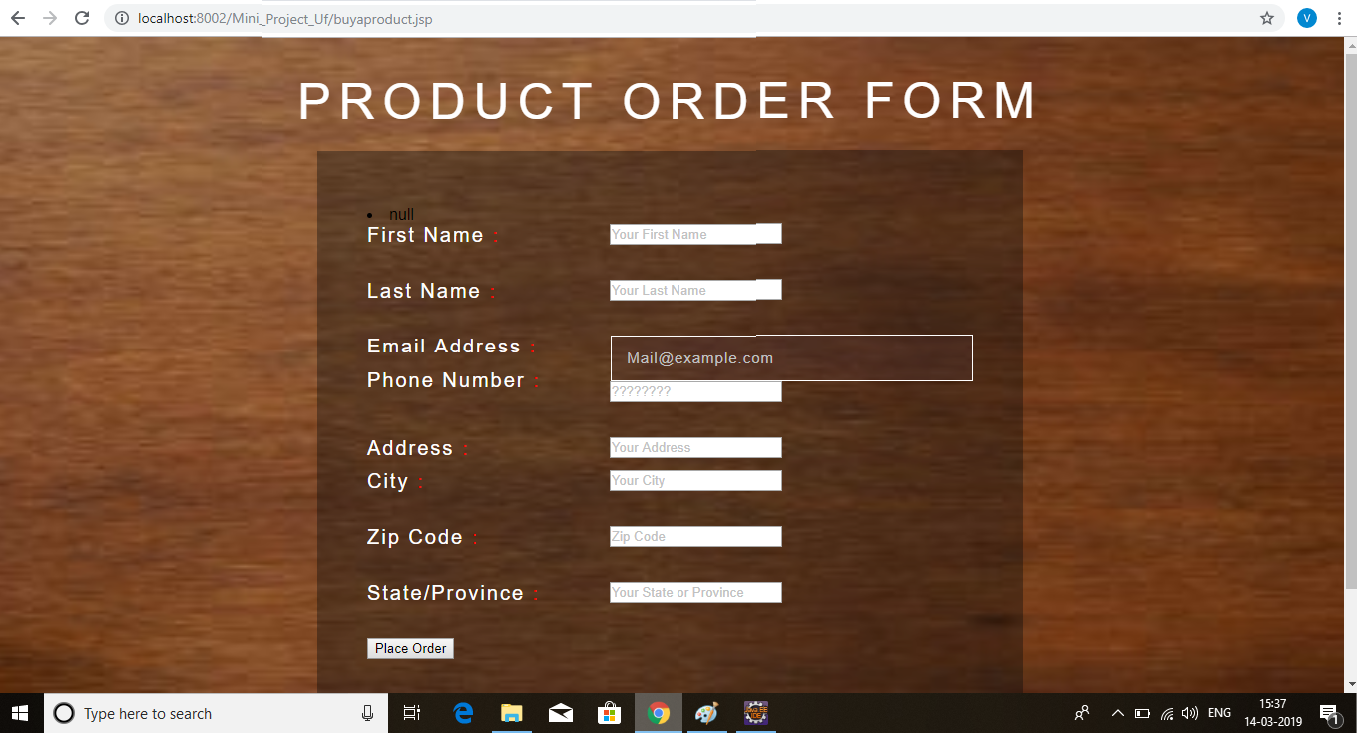
In figure 4.5, whenever the user selects the particular category then the user can view all the

products related to the selected category.

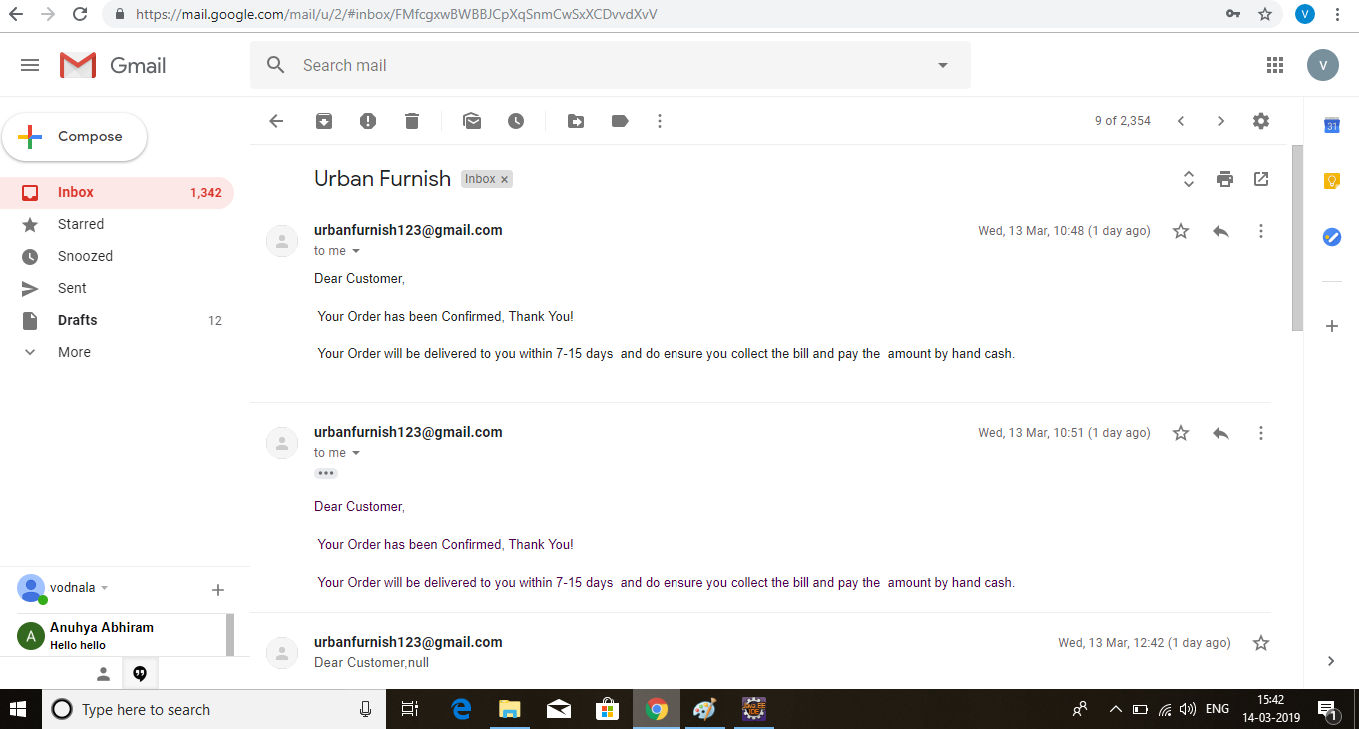
**** Figure 4.5: Product Categories Page

In figure 4.6, after selecting a particular product ,then the user have to fill the order form

(Shipping address).

**** Figure 4.6: Order Page

In Figure 4.7, Here the user receives the order confirmation details.

 Figure 4.7: Email Confirmation Page

In figure 4.8, Here the admin can view all the products of all the categories. Admin can also

update and delete the products from the portal.

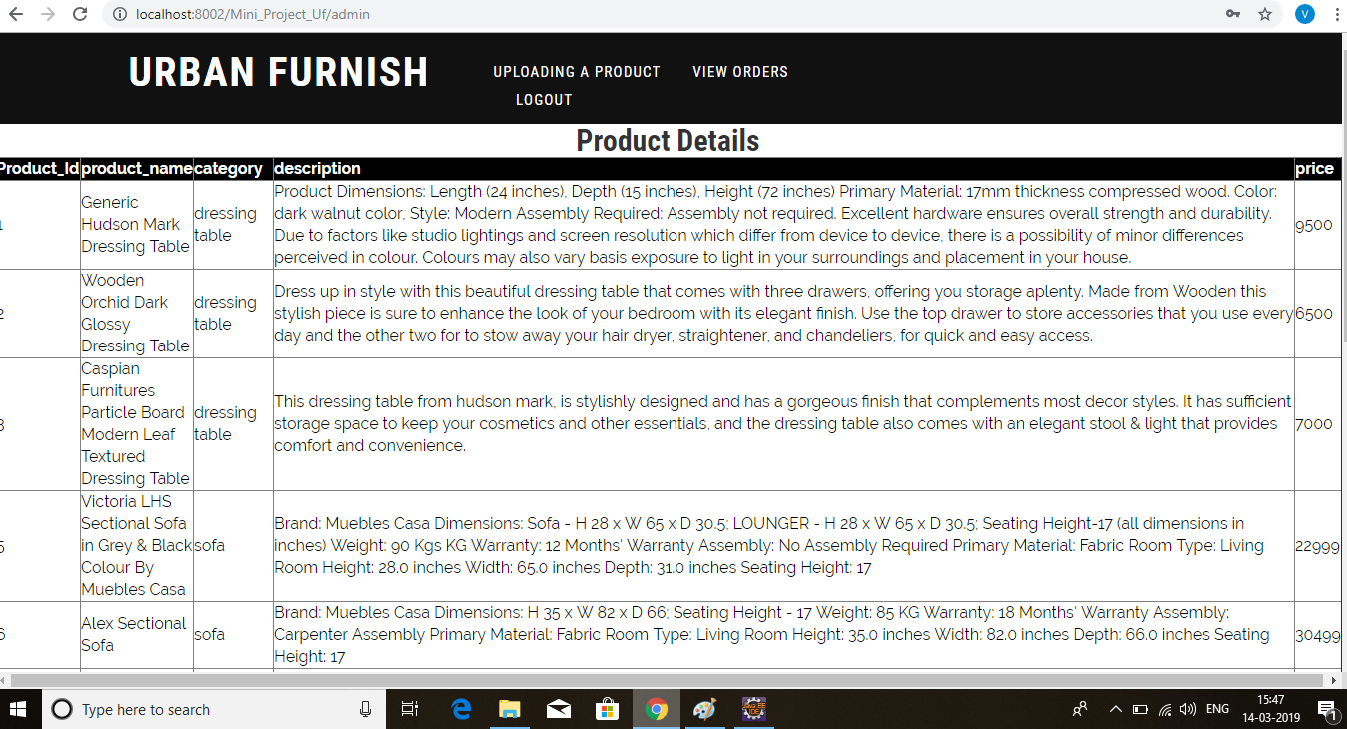


Figure 4.8: Admin Page

In figure 4.9, the admin can upload the products into the database of all the categories. And here he have to give all description of the product and all the details.

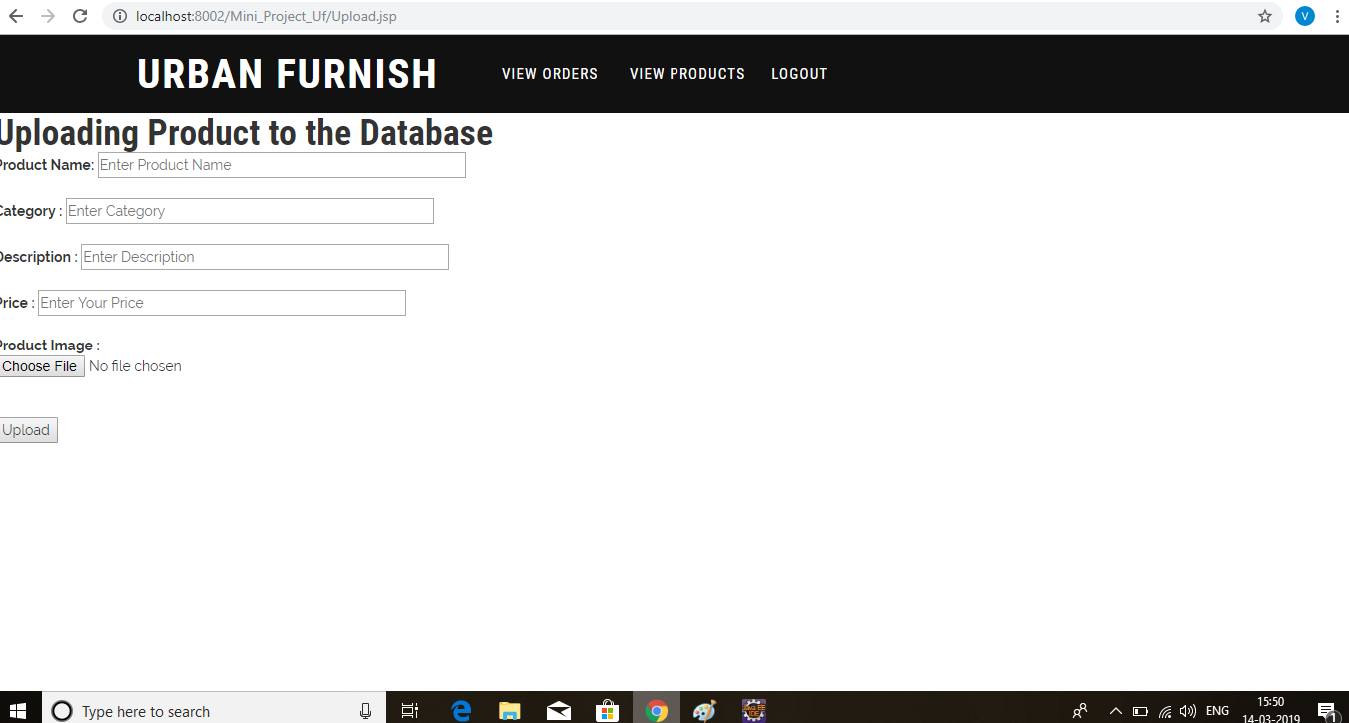


Figure 4.9: Admin adding Products

In figure 4.10, the admin can view the order list and can confirm the product.

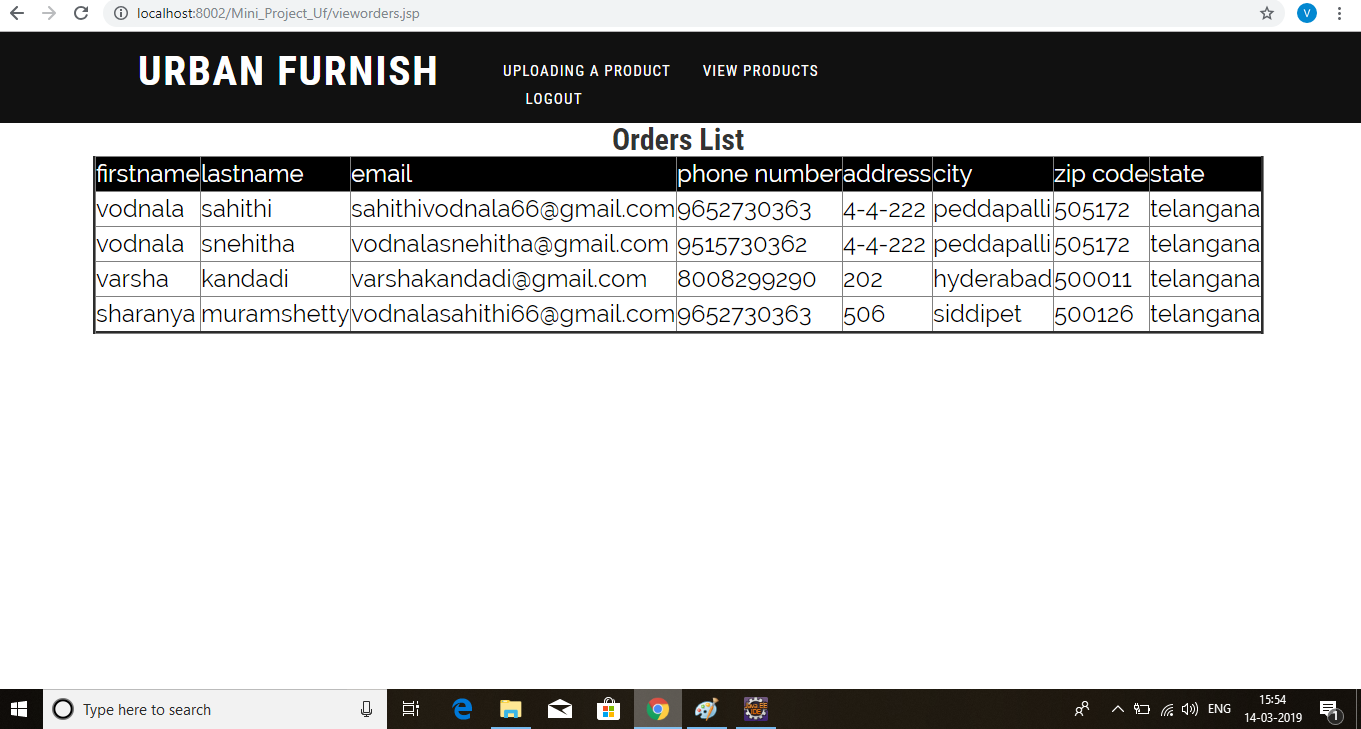


Figure 4.10: Orders List

**5. CONCLUSION**

In the current market for online shopping we do not have many applications to be precisely for furniture shopping. This application facilitates more user friendly interfaces, virtual shopping experience, on the go selections, dimensions of the furniture and easy payment options. In this technical world everything we want online but due to some online payment issues and customer specifications there might be wrong delivery of product sometimes by keeping these points in mind we have developed this furniture portal exclusively with cash and delivery. This project will be very much helpful for the present generation which runs shortage of time.

**Future enhancements**

Further, in future we would like to recommend certain valuable additions to the project.The most

important is to get verification link to the user mail.To make website more users friendly.

Configuration features can be changed according to the users requirements. Although the project

is complete and ready for implementation there is always room for improvement.To make online

payment method.

**REFERENCES**

1. <https://www.w3schools.com/bootstrap4/bootstrap_templates.asp>
2. <https://www.geeksforgeeks.org/send-email-using-java-program/>
3. <https://www.javacodegeeks.com/2012/05/load-or-save-image-using-hibernate.html>
4. <https://www.amazon.in/Bluewud-Dressing-Shelves-Mirror-Hanging/dp/B06WD67PF1?tag=googinhydr18418-21&tag=googinkenshoo-21&ascsubtag=_k_EAIaIQobChMIwp6-3sqB4QIVCB4rCh1DMQM2EAQYAiABEgI4AfD_BwE_k_&gclid=EAIaIQobChMIwp6-3sqB4QIVCB4rCh1DMQM2EAQYAiABEgI4AfD_BwE>
5. <http://www.java2s.com/Code/Jar/s/Downloadsmtp144jar.htm>
6. <https://dev.mysql.com/downloads/connector/j/>
7. <https://stackoverflow.com/questions/16627910/how-to-code-a-very-simple-login-system-with-java>
8. <https://www.w3schools.com/css/default.asp>
9. <https://www.javatpoint.com/retrieving-image-from-oracle-database>
10. <https://www.homecentre.in/HyderabadStores?gclid=Cj0KCQjwhuvlBRCeARIsAM720HoqOCXPK6_IT2o0qMY90q6nrz-Fw13EMbyIFo2k8EIs-bjQM9Q2wj8aAsXKEALw_wcB>