Project work
Java servlet
Nam Lê | Java Palvelinohjelmointi | 2018

Overview

This project is a small website, built with Java, included JSP, JSTL, Servlet and use MySQL as database technology. It can be applied on regular bakery store, with some minor configurations.

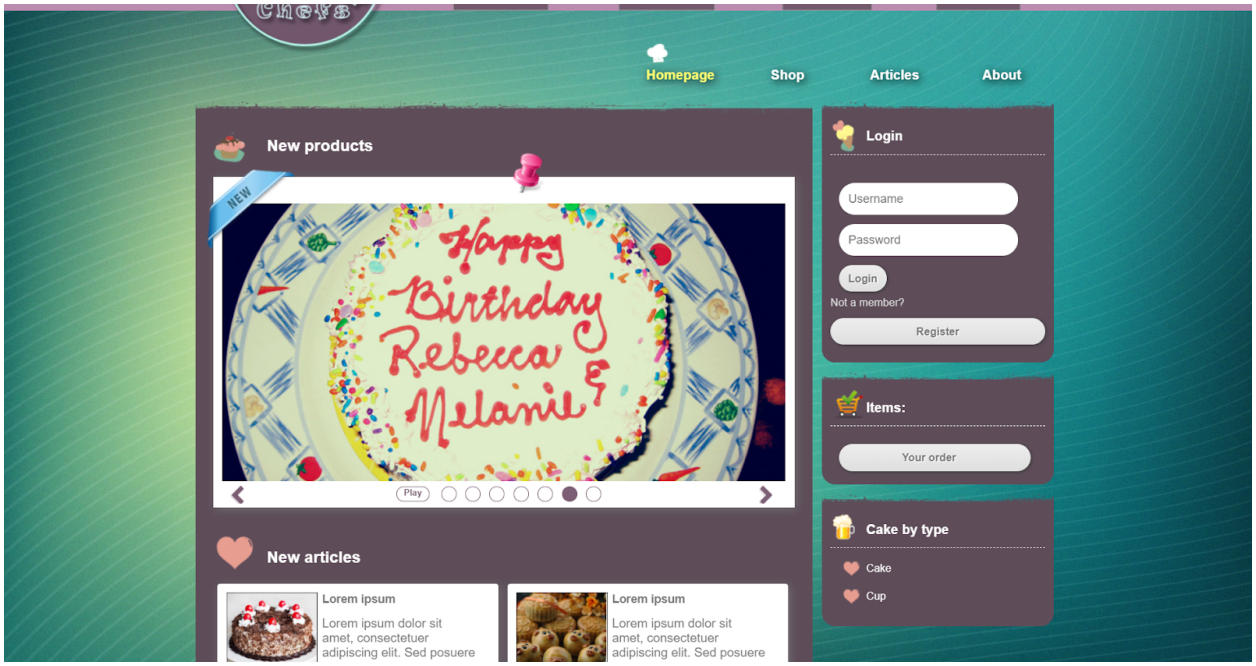
Detail

Website usage

All the user information, order and products detail is saved in database.

This website has authentication function, which let users register for a new username/password and authenticated users can have some benefits, which is saving information for future use. However, even if user doesn't want to login, the website still function normally, but the unlogged in users would have to enter their information every time they place an order.

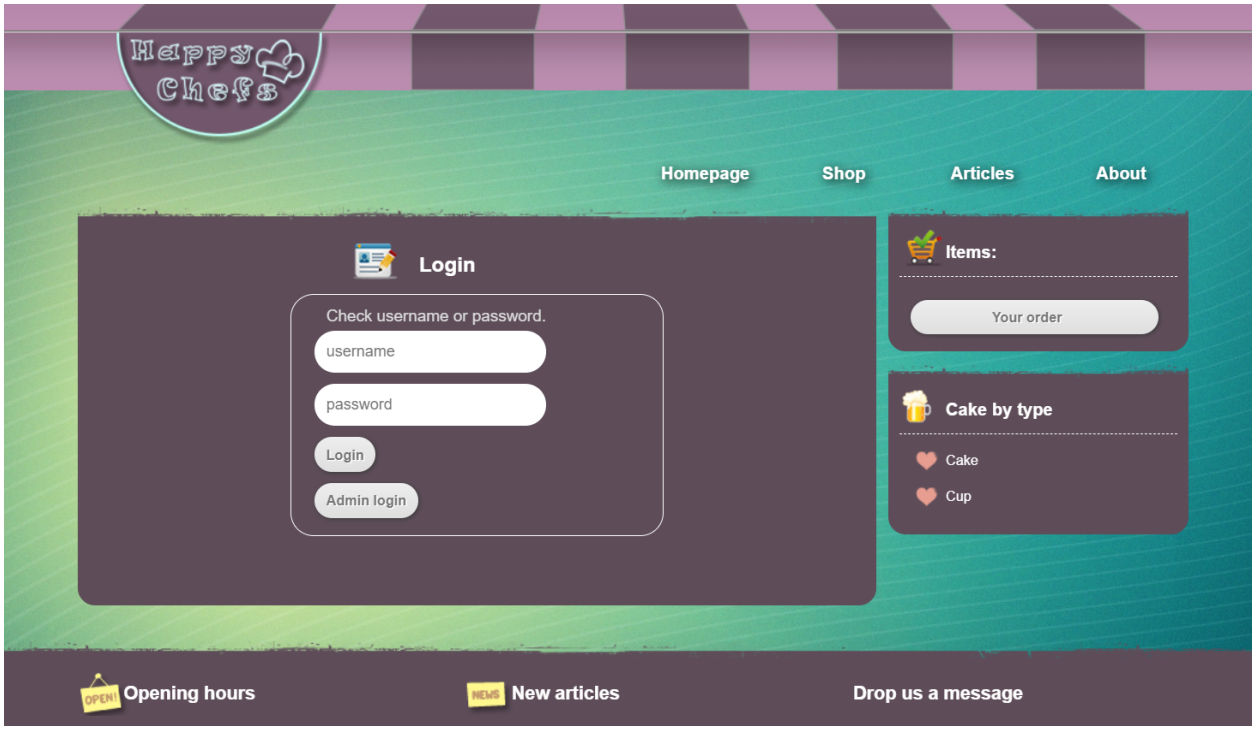
There are two kind of user, administrator and normal user. Administrator login with their username/password will have admin page. Comparing to normal user, administrator has more power, with the ability to add/edit/delete products. Normal user can only choose the website to display all products, place order, as well as edit their information. (The editing function is not working yet).



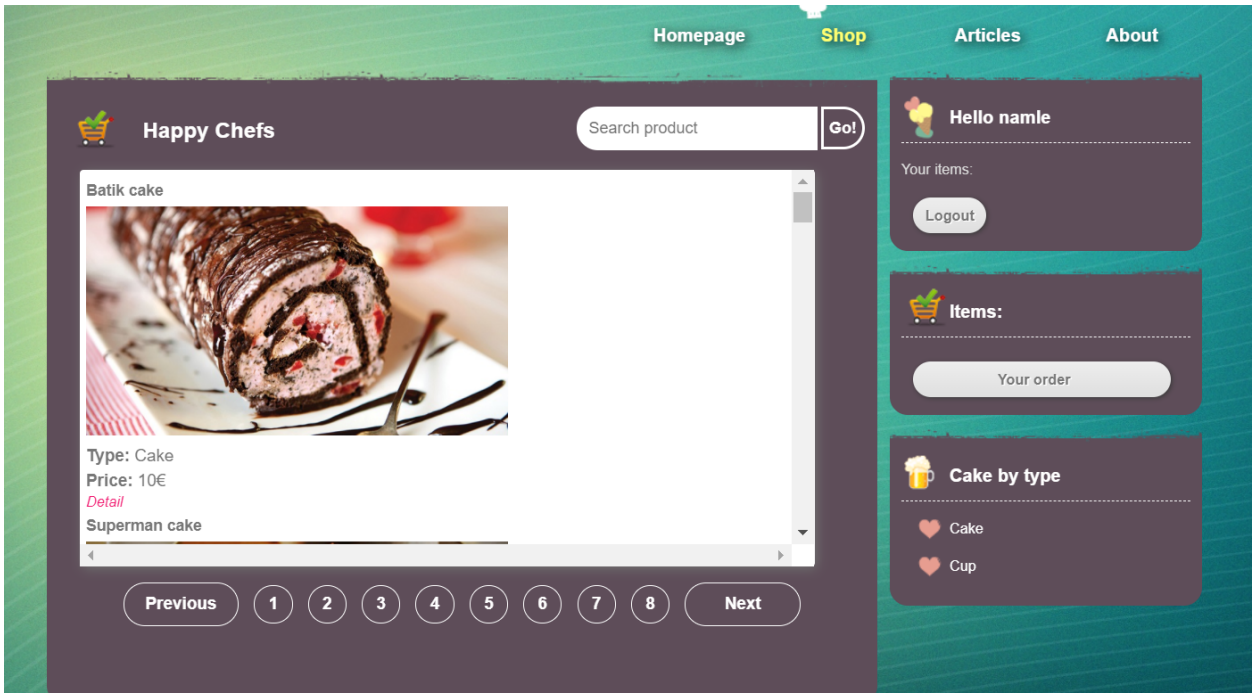
The picture above showing the homepage of the website. In the middle, there are a slideshow, which shows all the newest products, this function is provided by a Javascript plugin called Slide-show. On the left sidebar, there are authentication area, added items bucket (all products that user want to buy shows here, include the amount of each product and the price).

The login function let users enter their login information. After enter the required information, user will be redirected to homepage, from there they can start to choose product and place order. If user enter the wrong credential information, they will be redirected to login again page. There are also new articles, where user can read some articles about the products.

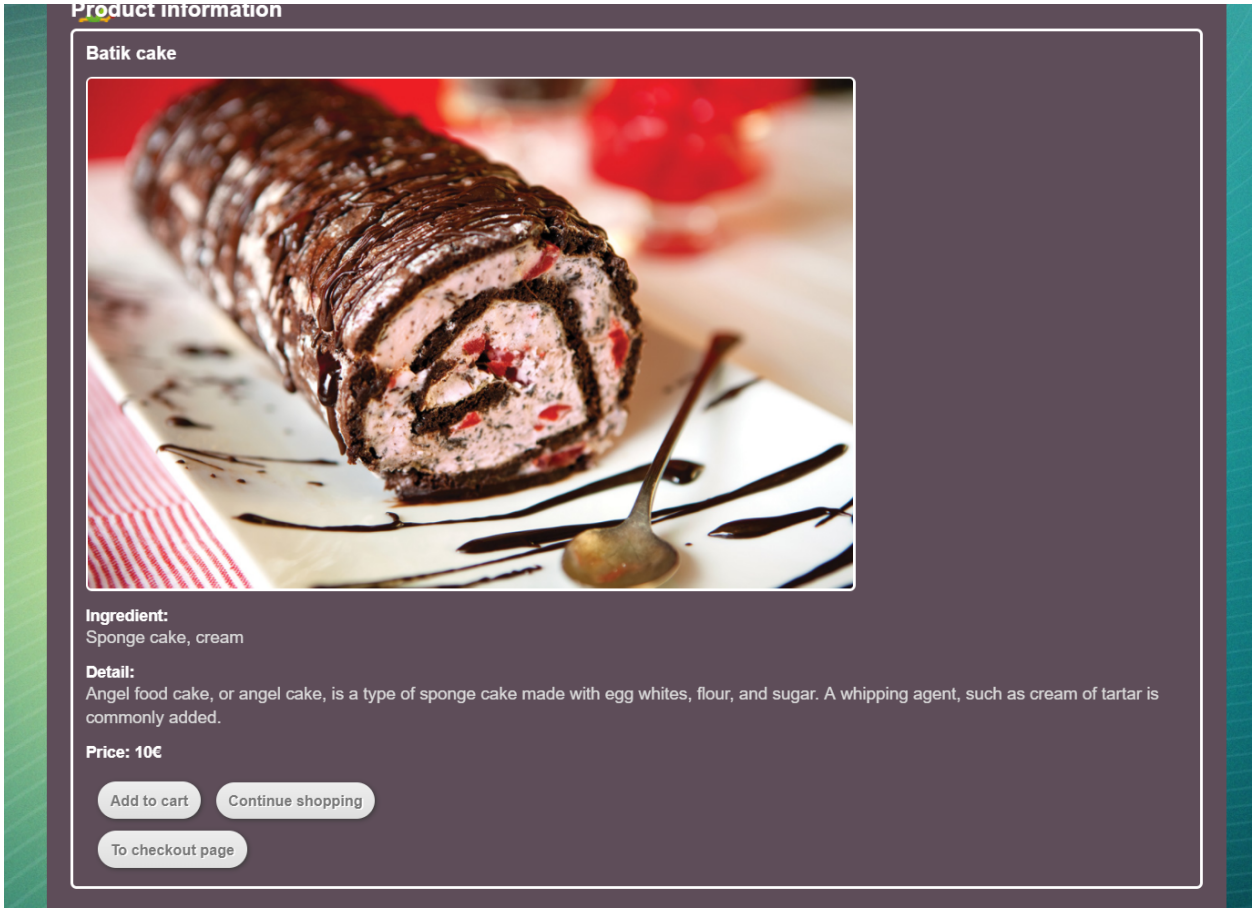
There are a filter for user as well, which is Cake by type. If user wishes to choose only product listed in Cake or only Cup category, then they can click on each category listed, product will be displayed by wished type.



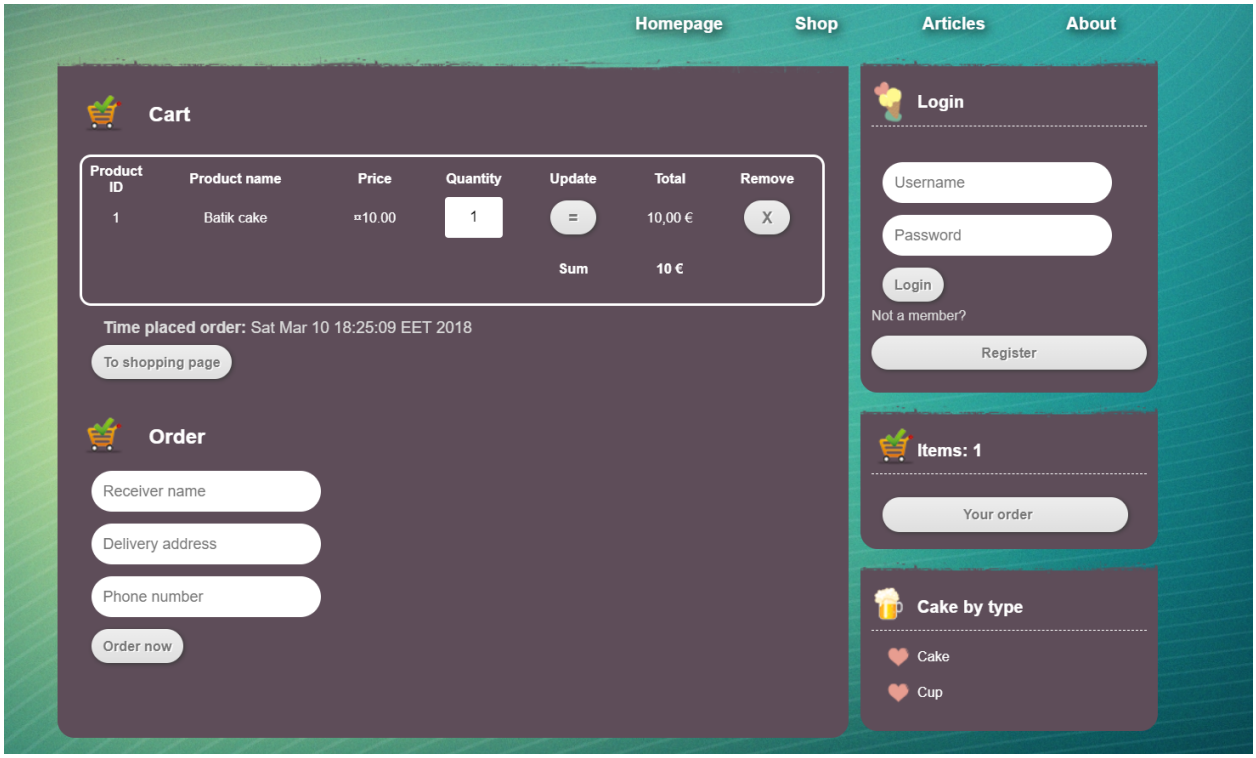
There are a navigation bar, where user can navigate around the website.



The picture above is showing Shop page, which lists all the products that the store is currently selling. User can click on Detail, to enter product page, where every detail of the selected product will be displayed, include the ingredient, brief, price…

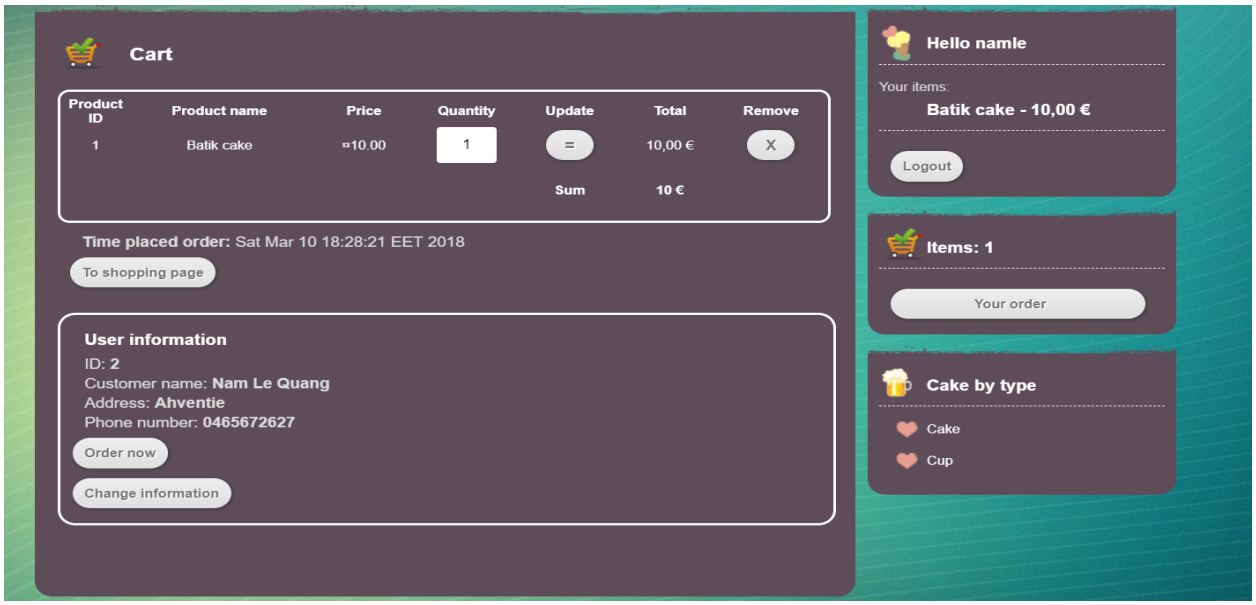


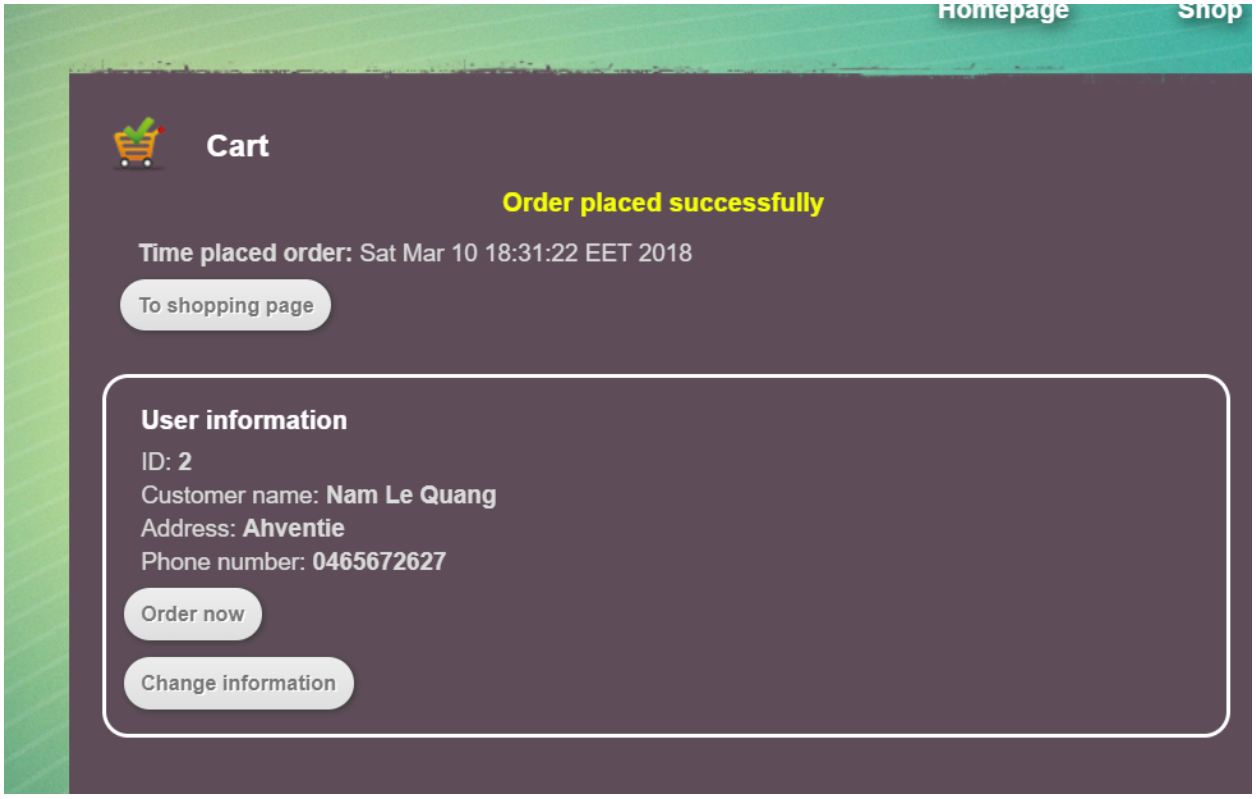
The above picture shows an example of detail product information page. User can select add to cart, or continue shopping or go to checkout page. A prompt will be displayed to show that the selected product has added successfully to user's cart.



In this page, user can change the quantity by entering the amount of product that they want, after that they can update by pressing update button. Price will change accordingly also.

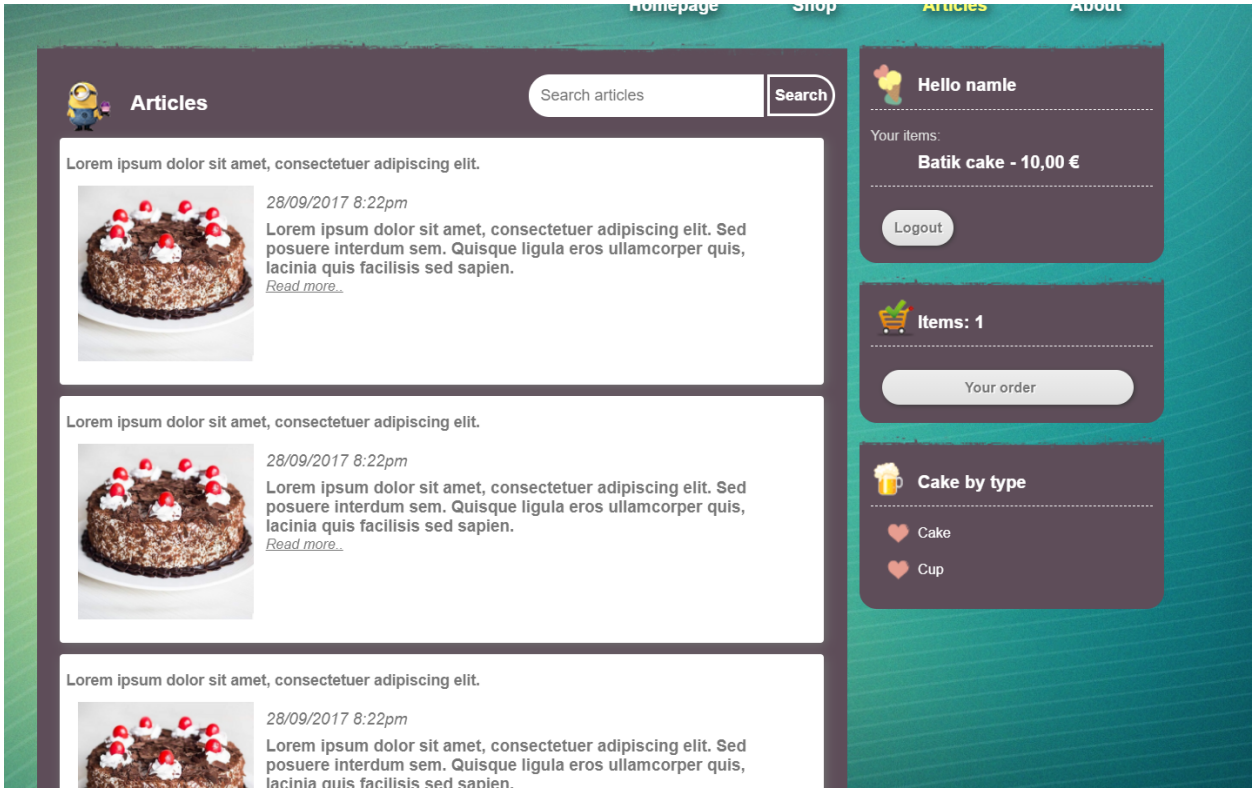
The unlogged in user will have to enter the required information to place the order. If they don't enter all the information, the Order now button will not work and a prompt will be display to tell the user also.





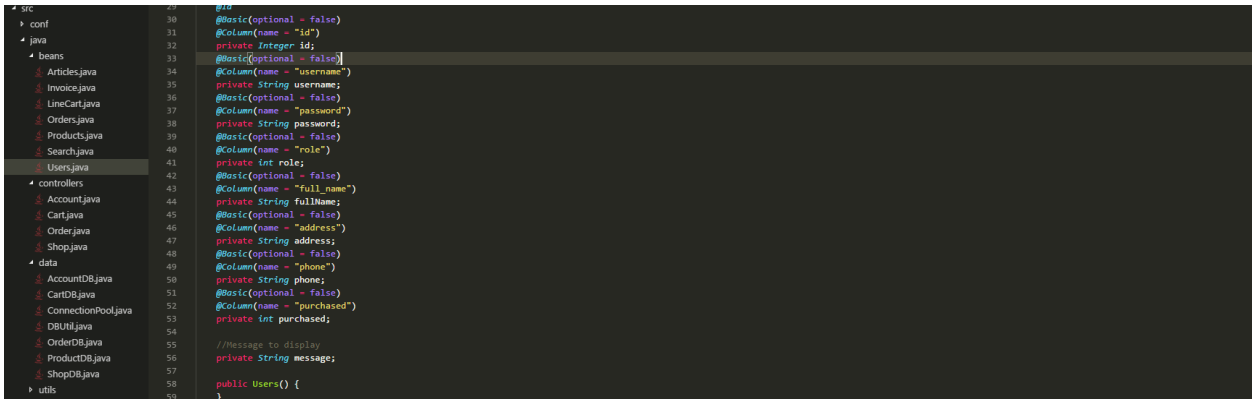
After confirm the information, user press Order now button, an order will be placed and recorded to database for delivering.

Above is an example of logged in user, all information is fetch automatically so user can save time of enter delivery information.



The articles page shows all the articles which has written.

Code



Picture above shows code structure of this project. It is divided into three packages, which is beans(generated from database, and all code that process data from database).

Package beans

```java
}

public Users(Integer id) {
    this.id = id;
}

public Users(String username, String password) {
    this.username = username;
    this.password = password;
}

public Users(Integer id, String username, String password, int role, String fullName, String address, String phone, int purchased) {
    this.id = id;
    this.username = username;
    this.password = password;
    this.role = role;
    this.fullName = fullName;
    this.address = address;
    this.phone = phone;
    this.purchased = purchased;
}

public Users(String username, String password, String fullName, String address, String phone) {
    this.username = username;
    this.password = password;
    this.fullName = fullName;
    this.address = address;
    this.phone = phone;
}

//contructor for cart user display
public Users(int id, String fullName, String address, String phone) {
    this.id = id;
    this.fullName = fullName;
    this.address = address;
    this.phone = phone;
}

public String getMessage() {
    return message;
}

public Integer getId() {
    return id;
}

public void setId(Integer id) {
    this.id = id;
}
```

Example code of Users class, which is generated with annotation from database. All the other class in beans packages are similar structure.

Package controllers

The second package is the controllers, which hold all the controller of the website, included Account, Cart, Order and Shop.

Account controller

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

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

    if (action == null) {
        return;
    }

    ConnectionPool pool = ConnectionPool.getInstance();
    Connection conn = pool.getConnection();
    AccountDB accountDB = new AccountDB(conn);

    if (action.equals("dologin")) {
        String username = request.getParameter("username");
        String password = request.getParameter("password");

        HttpSession session = request.getSession(); //Set session for user

        session.setAttribute("username", username);
        session.setAttribute("password", "");

        try {
            if (accountDB.login(username, password)) {

                Users user = accountDB.show(username);

                int id = user.getId();
                String fullname = user.getFullName();
                String address = user.getAddress();
                String phone = user.getPhone();

                session.setAttribute("id", id);
                session.setAttribute("fullname", fullname);
                session.setAttribute("address", address);
                session.setAttribute("phone", phone);
                session.setAttribute("user", user);

                request.getRequestDispatcher("/index.jsp").forward(request, response);
            } else {
                request.setAttribute("message", "Check username or password.");
                request.getRequestDispatcher("/login.jsp").forward(request, response);
            }
        } catch (SQLException ex) {
            request.getRequestDispatcher("/register.jsp").forward(request, response);
        }
    } else if (action.equals("register")) {
        String newusername = request.getParameter("newusername");
        String newpassword1 = request.getParameter("newpassword1");
        String newpassword2 = request.getParameter("newpassword2");
        String newfullname = request.getParameter("newfullname");
        String newaddress = request.getParameter("newaddress");
        String newphone = request.getParameter("newphone");

        request.setAttribute("newusername", newusername);
        request.setAttribute("newpassword1", "");
        request.setAttribute("newpassword2", "");
        request.setAttribute("newfullname", newfullname);
        request.setAttribute("newaddress", newaddress);
        request.setAttribute("newphone", newphone);

        request.setAttribute("message", "");

        if (!newpassword1.equals(newpassword2)) {
            //Password not match
            request.setAttribute("message", "Password not match.");
            request.getRequestDispatcher("/register.jsp").forward(request, response);
        } else {
            Users user = new Users(newusername, newpassword1, newfullname, newaddress, newphone);

            if (!user.validate()) {
                //Password wrong format
                request.setAttribute("message", user.getMessage());
                request.getRequestDispatcher("/register.jsp").forward(request, response);
            } else {
                try {
                    if (accountDB.exists(newusername)) {
                        //Username exists
                        request.setAttribute("message", "Username is taken.");
                        request.getRequestDispatcher("/register.jsp").forward(request, response);
                    } else {
                        accountDB.create(newusername, newpassword1, newfullname, newaddress, newphone);
                        request.getRequestDispatcher("/success.jsp").forward(request, response);
                    }
                } catch (SQLException e) {
                    request.getRequestDispatcher("/error.jsp").forward(request, response);
                }
            }
        }

    } else if (action.equals("dologout")) {
        request.getSession().invalidate();
        response.sendRedirect(request.getContextPath() + "/index.jsp");
    }

    try {
        conn.close();
    } catch (SQLException ex) {
        Logger.getLogger(Account.class.getName()).log(Level.SEVERE, null, ex);
```

Account controller uses only POST method, which only get username/password from users, without displaying it on URL. User enter their credential information, then these input will be compared to the information fetched from database, which will let user login or not. If user clicked on Register button and after enter all the required information, all those information will be written to database. If use wishes to have a username that already taken, they will be prompted to choose another username. This controller also handle login/logout function of the website.

Cart controller

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    String action = request.getParameter("action");

    if (action == null) {
        return;
    }

    switch (action) {
        case "add":
            addProductToCart(request, response);
            request.setAttribute("addsuccess", "Add to cart successfully");
            getServletContext().getRequestDispatcher("/order.jsp").forward(request, response);
            break;
        case "remove":
            removeProductFromCart(request, response);
            getServletContext().getRequestDispatcher("/cart.jsp").forward(request, response);
            break;
        case "update":
            updateProductFromCart(request, response);
            getServletContext().getRequestDispatcher("/cart.jsp").forward(request, response);
            break;
        default:
            getServletContext().getRequestDispatcher("/cart.jsp").forward(request, response);
            break;
    }
}

private void showCart(HttpServletRequest request, HttpServletResponse response) {
    HttpSession session = request.getSession();
    CartDB cart = (CartDB) session.getAttribute("cart");

    if (cart == null || cart.getCount() == 0) {
        request.setAttribute("emptyCart", "Your cart is empty!");
    } else {
        request.getSession().setAttribute("cart", cart);
    }

    java.util.Date today = new java.util.Date();

    Invoice invoice = new Invoice();
    invoice.setInvoiceDate(today);
    invoice.setLineCarts(cart.getItems());

    session.setAttribute("invoice", invoice);
}

private void addProductToCart(HttpServletRequest request, HttpServletResponse response) {
    String cartid = request.getParameter("cartid");

    HttpSession session = request.getSession();
```

This controller provide show/add/remove/update product function. GET method is in use, for only show function, add/remove/update is handle by it own method, and all these methods is called inside POST method, by this way it is easier to modify code and handle function.

Method showCart fetch all selected products for user. Which addProductToCart, removeProductFromCart, updateProductFromCart will add, remove, update selected product to user's cart.

Order controller

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

    HttpSession session = request.getSession();

    if (session.getAttribute("username") != null) {
        addOrderLogged(request, response);
        session.setAttribute("success", "Order placed successfully");
        getServletContext().getRequestDispatcher("/cart.jsp").forward(request, response);
    } else if (session.getAttribute("username") == null) {
        addOrderNoLogged(request, response);
        session.setAttribute("success", "Order placed successfully");
        getServletContext().getRequestDispatcher("/cart.jsp").forward(request, response);
    } else {
        getServletContext().getRequestDispatcher("/thanks.jsp").forward(request, response);
    }
}

private void addOrderLogged(HttpServletRequest request, HttpServletResponse response) {
    HttpSession session = request.getSession();
    CartDB cart = (CartDB) session.getAttribute("cart");

    Users user = (Users) session.getAttribute("user");

    try {
        int id = user.getId();
        String fullname = user.getFullName();
        String address = user.getAddress();
        String phone = user.getPhone();

        Invoice invoice = new Invoice();

        invoice.setLineCarts(cart.getItems());
        List<LineCart> line = invoice.getLineCarts();

        int prdId;
        String prdName;
        String prdQuantity;
        int prdCost;

        for (LineCart lc : line) {
```

Order controller use POST method only, to let user place the order. There are two kind of user order also, which is handled by separate method. Method addOrderLogged will process order from logged in user. This method fetch user information saved in database, so logged in user doesn't need to add information on placing order. Method addOrderNoLogged process other order, which will ask user to provide required information to place the order, such as address, phone number…

Shop controller

```
public class Shop extends HttpServlet {

    private DataSource ds;

    @Override
    public void init(ServletConfig config) throws ServletException {
        try {
            InitialContext initContext = new InitialContext();

            Context env = (Context) initContext.lookup("java:comp/env");

            ds = (DataSource) env.lookup("jdbc/dbs");

        } catch (NamingException e) {
            throw new ServletException();
        }
    }

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

        Connection conn = null;

        try {
            conn = ds.getConnection();
        } catch (SQLException ex) {
            throw new ServletException();
        }

        ShopDB shopAction = new ShopDB(conn);

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

        //set display with type
        if (type != null) {
            shopAction.displayWithType(1, type);
            request.getRequestDispatcher("/shop.jsp").forward(request, response);
        } else {
            request.getRequestDispatcher("/index.jsp").forward(request, response);
        }
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String action = request.getParameter("action");

        if (action == null) {
            return;
        }

        Connection conn = null;

        try {
            conn = ds.getConnection();
        } catch (SQLException ex) {
            throw new ServletException();
        }
```

This controller suppose to provide functions that work within Shop page, however these functions, such as search for specific product is not yet implemented.

Package data

This package contains classes that handles everything related to database.

```
public class ConnectionPool {

    private static ConnectionPool pool = null;
    private static DataSource dataSource = null;

    public synchronized static ConnectionPool getInstance() {
        if (pool == null) {
            pool = new ConnectionPool();
        }
        return pool;
    }

    private ConnectionPool() {
        try {
            InitialContext ic = new InitialContext();
            dataSource = (DataSource) ic.lookup("java:/comp/env/jdbc/dbs");
        } catch (NamingException e) {
            System.err.println(e);
        }
    }

    public Connection getConnection() {
        try {
            return dataSource.getConnection();
        } catch (SQLException sqle) {
            System.err.println(sqle);
            return null;
        }
    }

    public void freeConnection(Connection c) {
        try {
            c.close();
        } catch (SQLException ex) {
            System.err.println(ex);
        }
    }

}
```

The ConnectionPool class, provide connection to database, getConnection and freeConnection. All classes use this connection pool to fetch data from database.

Class AccountDB

```
public boolean login(String username, String password) throws SQLException {

    boolean st = false;

    String sql = "SELECT * FROM USERS WHERE username=? AND password=?";

    PreparedStatement ps = conn.prepareStatement(sql);

    ps.setString(1, username);
    ps.setString(2, password);

    ResultSet rs = ps.executeQuery();

    st = rs.next();

    rs.close();

    return st;
}

public void create(String newusername, String newpassword,
        String newfullname, String newaddress, String newphone) throws SQLException {
    String sql = "INSERT INTO dbs.users (username, password, full_name, address, phone) VALUES (?,?,

    PreparedStatement ps = conn.prepareStatement(sql);

    ps.setString(1, newusername);
    ps.setString(2, newpassword);
    ps.setString(3, newfullname);
    ps.setString(4, newaddress);
    ps.setString(5, newphone);

    ps.executeUpdate();

    ps.close();
}

public boolean exists(String username) throws SQLException {
    boolean st = false;

    String sql = "SELECT * FROM USERS WHERE username=?";
```

This class handle login/create/show for Account controller, it also check if a username is already exists (in register action) to prompt to user.

CartDB

```
public class CartDB implements Serializable {

    private List<LineCart> items;

    public CartDB() {
        items = new ArrayList<>();
    }

    public void setItems(List<LineCart> lineCarts) {
        items = lineCarts;
    }

    public List<LineCart> getItems() {
        return items;
    }

    public int getCount() {
        return items.size();
    }

    public void addItem(LineCart cart) {
        String cartid = cart.getProduct().getId().toString();
        int quantity = cart.getCartQuantity();

        for (LineCart lineCart : items) {
            if (lineCart.getProduct().getId().equals(cartid)) {
                lineCart.setCartQuantity(quantity);
                return;
            }
        }
        items.add(cart);
    }

    public void removeItem(LineCart cart) {
        String cartid = cart.getProduct().getId().toString();

        for (int i = 0; i < items.size(); i++) {
```

CardDB class provide addItem and removeItem, let user add/remove item from their cart.

OrderDB

```
public void insertOrderLogged(int usr_id, String usr_fullname, String usr_address, String usr_phone,
        int prdId, String prdName, String prdQuantity, int prdCost) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection conn = pool.getConnection();
    PreparedStatement ps = null;

    String sql = "INSERT INTO dbs.order(user_id,full_name,address,phone,prd_id,prd_name,prd_quan,prd_cos
            + "VALUES(?,?,?,?,?,?,?,?)";

    try {
        ps = conn.prepareStatement(sql);

        ps.setInt(1, usr_id);
        ps.setString(2, usr_fullname);
        ps.setString(3, usr_address);
        ps.setString(4, usr_phone);

        ps.setInt(5, prdId);
        ps.setString(6, prdName);
        ps.setString(7, prdQuantity);
        ps.setInt(8, prdCost);

        ps.executeUpdate();
    } catch (SQLException e) {
        System.err.println(e);
    } finally {
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(conn);
    }
}

public void insertOrderNoLogged(String customername, String address, String phone,
        int prdId, String prdName, String prdQuantity, int prdCost) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection conn = pool.getConnection();
    PreparedStatement ps = null;
```

OrderDB class insert records to database, method insertOrderLogged handles order from logged in users, insertOrderNoLogged handles order from unlogged in users.

ProductDB

```java
public static Products selectProduct(String prdId) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;
    ResultSet rs = null;

    String sql = "SELECT * FROM products WHERE id =?";

    try {
        ps = connection.prepareStatement(sql);
        ps.setString(1, prdId);

        rs = ps.executeQuery();

        if (rs.next()) {
            Products p = new Products();
            p.setId(rs.getInt("id"));
            p.setName(rs.getString("name"));
            p.setType(rs.getString("type"));
            p.setBrief(rs.getString("brief"));
            p.setDetail(rs.getString("detail"));
            p.setIngredient(rs.getString("ingredient"));
            p.setPrice(rs.getInt("price"));
            return p;
        } else {
            return null;
        }
    } catch (SQLException e) {
        System.err.println(e);
        return null;
    } finally {
        DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
    }
}
```

ProductDB class provide methods for display product, fetch product information from the database.