6th - In class Assignment :-

Method Name : Login to Application Class Name : UserLogin SO : 01

Client : Users

Associated use case : Login, Signup.

Description of Responsibilities : for this condition when User user tries to Login there should be username and password. It should retrieve the values from the data base. It should not support to be NULL.

Arguments Received : It will call the UserController method with the @RequestMapping from the controller and it will go and verify the DB. If will be correct the User data.

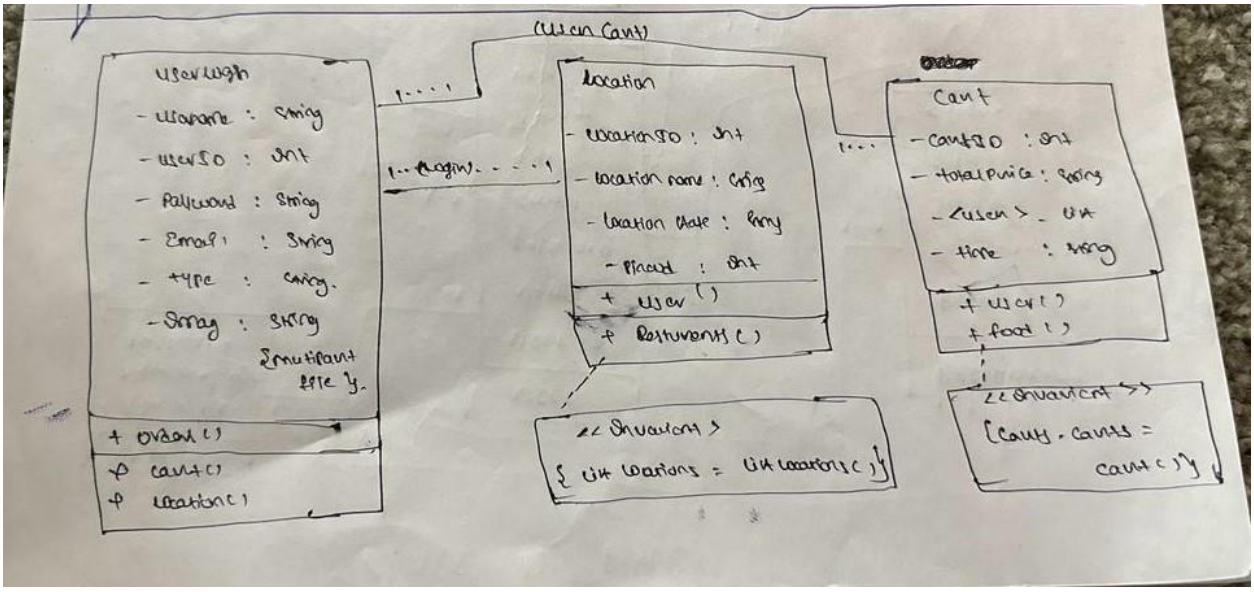
Types of value returned : It will return the data from the DB. Other wise it will throw HTTP 500 or HTTP 400 status NOT OK normally. or 200 OK.

Pre condition : for this case the precondition will be there need to be the valid username and password. and it should not support to be NULL.

Post condition : we are using the static variable UserID = 0 ; It will refer go and fetch the User details from the repository method.

@ Repository

Public interface User



Method Name : CartList Classname : CartItems ID: 08

Clients : Users

Associated usecase : User, food, Orders.

Description of responsibility : When User user added items into his cart. There will be a list of items will be displayed. userId should not appear to be Null.

Arguments Received : When User Controller tries to fetch the cart items list, it will fetch the all items of the cart with his userID. from the DB.

Type of value Returned : It will return the HTTP : status Response 500 ok or 200 Bad request.

Pre - condition : The pre condition would be the user should have items in the cart.

Post - condition : This will fetch the data from the controller and validate the userID. and shows the cart items with cartID.

```
//method for the login to application:
@Entity
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int userId;
    private String name;
    private String username;
    private String password;
    private String email;
    private String type;
    private String role;

    @Lob
    private String image;

    @OneToMany(mappedBy = "user", targetEntity = Orders.class, fetch = FetchType.EAGER)
    private List<Orders> orders;

    @OneToOne(mappedBy = "user", fetch = FetchType.EAGER, cascade = CascadeType.ALL, orphanRemoval = true)
    private Cart cart;

//Pre condition for this method:
@RequestMapping(value = "/login")
    public String userloginpage(Model model, @ModelAttribute("user") User user) {
        List<User> users = userService.getUsers();

        List<User> us = new ArrayList<User>();
        System.out.println(user.getName() + user.getPassword());
```

```
        List<User> us = new ArrayList<User>();
        System.out.println(user.getName() + user.getPassword());

        List<User> user1 = users.stream()
                .filter(e -> e.getUsername().equals(user.getUsername()) && e.getPassword().equals(user.getPassword()))
                .collect(Collectors.toList());

        for (User u : user1) {

//post condition for this method.
            if ((u.getUsername().equals(user.getUsername()) && u.getPassword().equals(user.getPassword()))) {

                userId = u.getUserId();

                if (u.getRole().equalsIgnoreCase("admin")) {
                    for (User us1 : users) {
                        if (us1.getUserId() == userId) {
                            us.add(us1);
                        }
                    }
                    model.addAttribute("us", us);
                    return "adminhomepage";
                } else {

                    for (User us1 : users) {
                        if (us1.getUserId() == userId) {
                            us.add(us1);
                        }
                    }
                    model.addAttribute("us", us);
```

```
                }
//invariant for this method to verify the data.
            } else {
                model.addAttribute("error", new ErrorDto("invalid user and password"));
                return "userlogin";
            }
        }
        return "userlogin";
    }
```

The above is for the Login method and the below is for the Cart list delete items.

```
//method for the login to application:
@Entity
public class CartItems {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int cartItemId;
    private int foodId;
    private String foodName;

    private String description;
    private double price;
    private String type;

    private LocalTime time;

    @ManyToOne(cascade = CascadeType.ALL)
    private Cart cart;

//Pre condition for this method:

    @RequestMapping("/cartlistt")
    public String getCart(Model model, @ModelAttribute(name = "cart") Cart cart) {
        List<Cart> cartsl = service.getAllCart();
        List<Food> carts = new ArrayList<Food>();
        User user = new User();
        //post condition for thsi method.
        userId = new Usercontroller().userId();
        for (Cart c : cartsl) {
            if (c.getUser().getUserId() == userId) {
                List<Food> foodl = c.getFood();
                for (Food f : foodl) {
                    carts.add(f);
                }

            }
        }

//Invariant for this method.
        model.addAttribute("carts", carts);

        return "cart";

    }
```