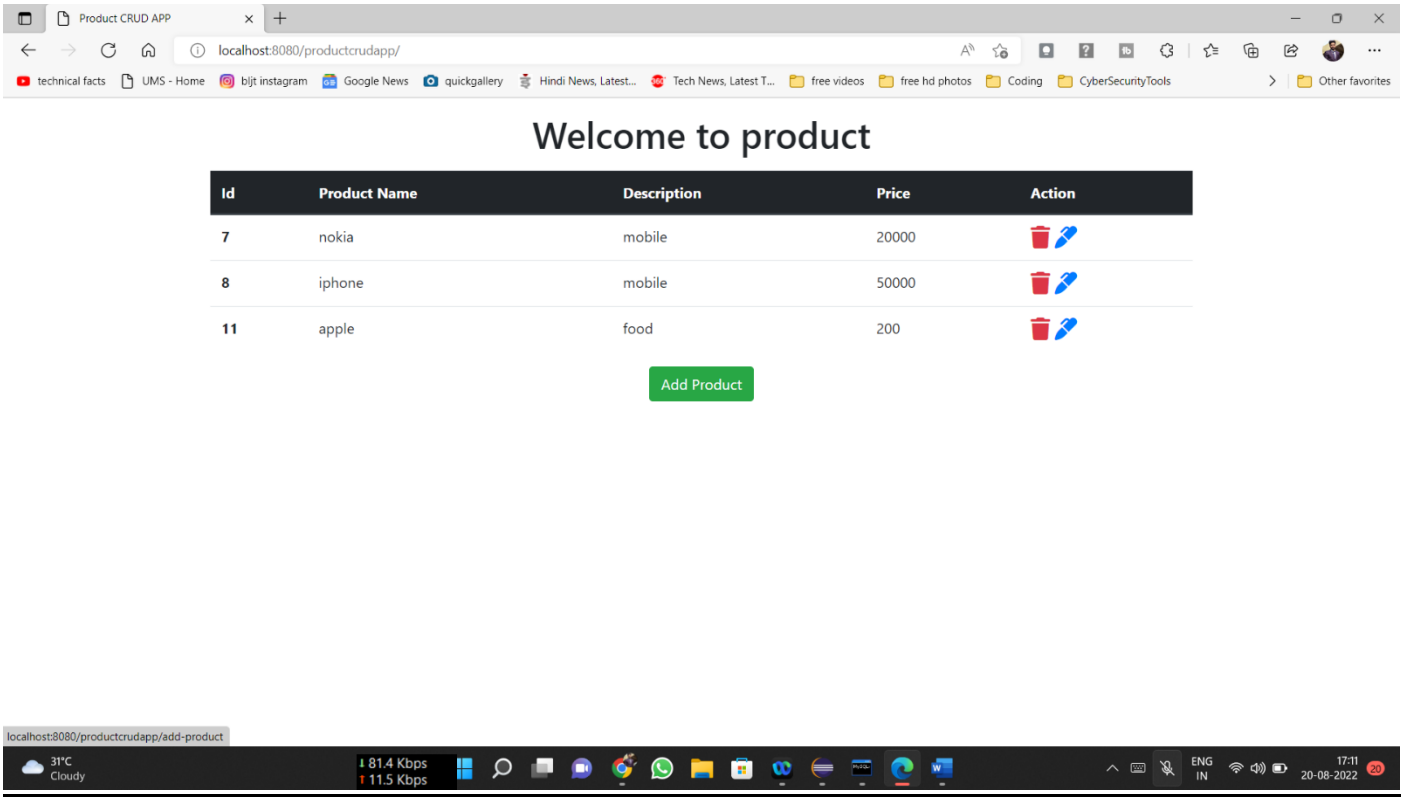
# Documentation

## Installation Guide:

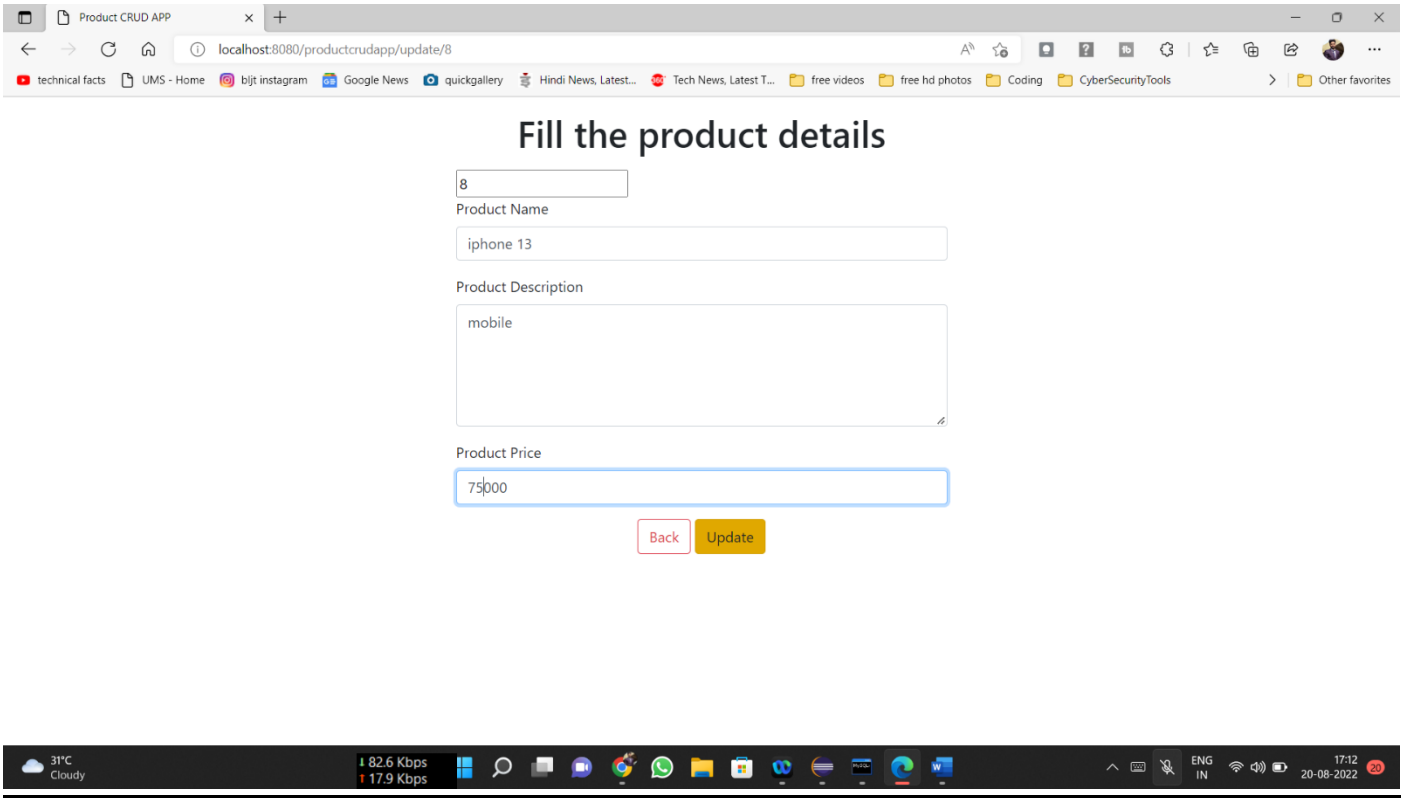
1. GitHub link: https://github.com/baljeet-singh97/JAVA-Projects/tree/main/Phase%203/productcrudapp
2. Download the entire project as Zip in local system.
3. import the project in Eclipse IDE

## How to use:

## Here you can add product delete existing product and also can modify the product



## In modify product user can modify the product from product id also write the product id and details that product will be modified




## Code Description

## Product.java

Defining the id name description and price of the product using hibernate creating the table.

```
private int id;
        private String name;
        private String description;
        private long price;
```

Defining getter setter of every variables.

```
private int id;
        private String name;
        private String description;
        private long price;
```

```
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
```

Defining to_string method to get the original data printed.

```
public String toString() {
		return "Product [id=" + id + ", name=" + name + ", description=" + description + ", price=" +
price + "]";
	}
```

## ProductDao.java

Autowired the hibernateTemplate so that we can use it anywhere in the spring

```
@Autowired
	private HibernateTemplate hibernateTemplate;
```

To save the products in the database created a method createProduct and passing the product in it to save or update the product if product alreade exist. product using transactional to do any kind of activity like save update or delete.

```
@Transactional
	public void createProduct(Product product)
	{
		this.hibernateTemplate.saveOrUpdate(product);
```

```
}
```

gettin all products in a list and returning to the user

```
public List<Product> getProducts()
{
	List<Product> products = this.hibernateTemplate.loadAll(Product.class);
	return products;
}
```

Delete the single product taking id from the user and deleting that product using transactional to do any kind of activity like save update or delete.

```
@Transactional
public void deleteProduct(int pid)
{
	Product p = this.hibernateTemplate.load(Product.class, pid);
	this.hibernateTemplate.delete(p);
}
```

To get the single product using product id

```
@Transactional
public Product getProduct(int pid)
{
	return this.hibernateTemplate.get(Product.class, pid);


}
```

## MainController.java

Any request coming from add-product will be handled by this method , it will redirect the usr to add_product_form where user will enter all the details of new product and the data will be redirected to handleproduct method.

```
@RequestMapping("/add-product")
	public String addProduct(Model m)
```

```
	{
		m.addAttribute("title", "Add Product");


		return "add_product_form";
	}
```

This method will handle all the request coming to add the data it will add the data to the database and redirect to the page with all the updated data

```
@RequestMapping(value= "/handle-product", method=RequestMethod.POST)
	public RedirectView handleProuduct(@ModelAttribute Product product, HttpServletRequest request)
	{
		System.out.println(product);
		productDao.createProduct(product);
		RedirectView redirectView = new RedirectView();
		redirectView.setUrl(request.getContextPath()+"/");


		return redirectView;
	}
```

This method is handling all the delete request delete request come with the product id and it will delete the product from the database using that id as shown below.

```
@RequestMapping("/delete/{productId}")
	public RedirectView deleteProduct(@PathVariable("productId") int productId  ,HttpServletRequest request)
	{
		this.productDao.deleteProduct(productId);
		RedirectView redirectView = new RedirectView();
		redirectView.setUrl(request.getContextPath()+"/");


		return redirectView;
	}
```

This method is handling all the update request coming form update href, with product id.

```
@RequestMapping("/update/{productId}")
        public String updateFormt(@PathVariable("productId") int pid, Model model)
        {
                Product product = this.productDao.getProduct(pid);
                model.addAttribute("product", product);
                return "update_form";
        }
```