Distributed Systems

SE3020

Assignment 02

(2021S1_REG_WD_33)

Shop Online (Restful Web Services)

S.W.R.S.I.Rathnayake

A.R.W.M.B.W.D.B.Jayawardana

A.V.Joachim

K.A.Yathushan

Criteria	Good (10-8)	Average (4-7)	Poor (0-3)
Application of SOA principles in the architecture and the design			
Having clearly defined interfaces, that facilitate reusability			
Quality and the readability of the code, with meaningful and detailed comments.			
Integration of services using the Enterprise Service Bus (ESB)			
Comprehensiveness and the quality of the report			

Members & Contributions

IT Number	Name	Contribution
IT19062266	S.W.R.S.I.Rathnayake	• Delivery Service • Delivery Email Service
IT19127538	A.R.W.M.B.W.D.B.Jayawardana	• Mobile Payment Service • Mobile Payment Email Service
IT19037998	A.V.Joachim	• Item Service • Seller Service • Category Service
IT19513188	K.A.Yathushan	• Card Payment Service • Card Payment Email Service

Contents

Introduction

Shop Online is a collaborative online shopping platform. In this platform sellers can sell their items and buyers can buy goods easily. This platform mainly focuses on selling and purchasing on goods based on three main categories. Which includes,

- Furniture items.
- Musical Instruments
- Fitness Equipment.

Sellers need to register to the system if he wants to sell any item. But buyers can purchase any item without creating an account. Here we also provide a delivery service as well. There are two payment methods to pay online for the products.

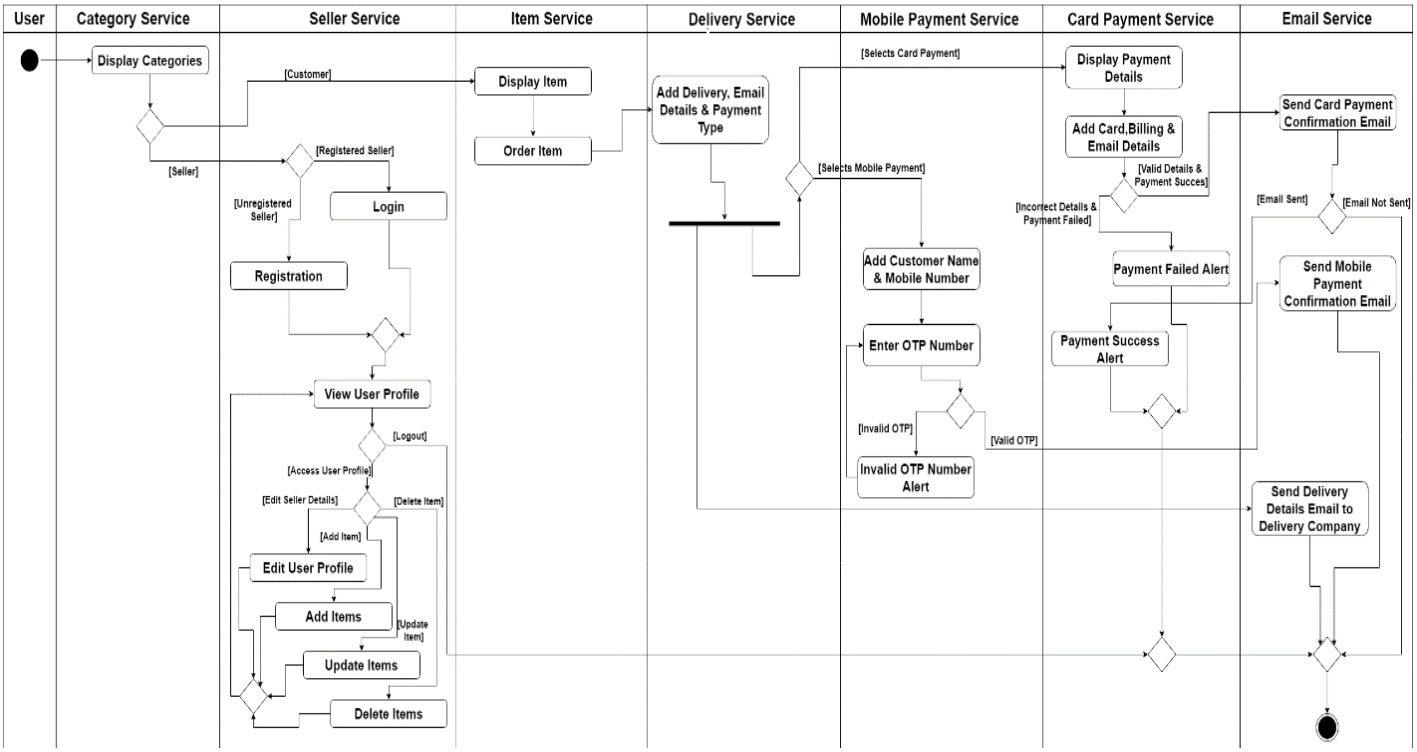
- Credit Card Payment.
- Mobile Payment.

This Restful web service application is developed using,

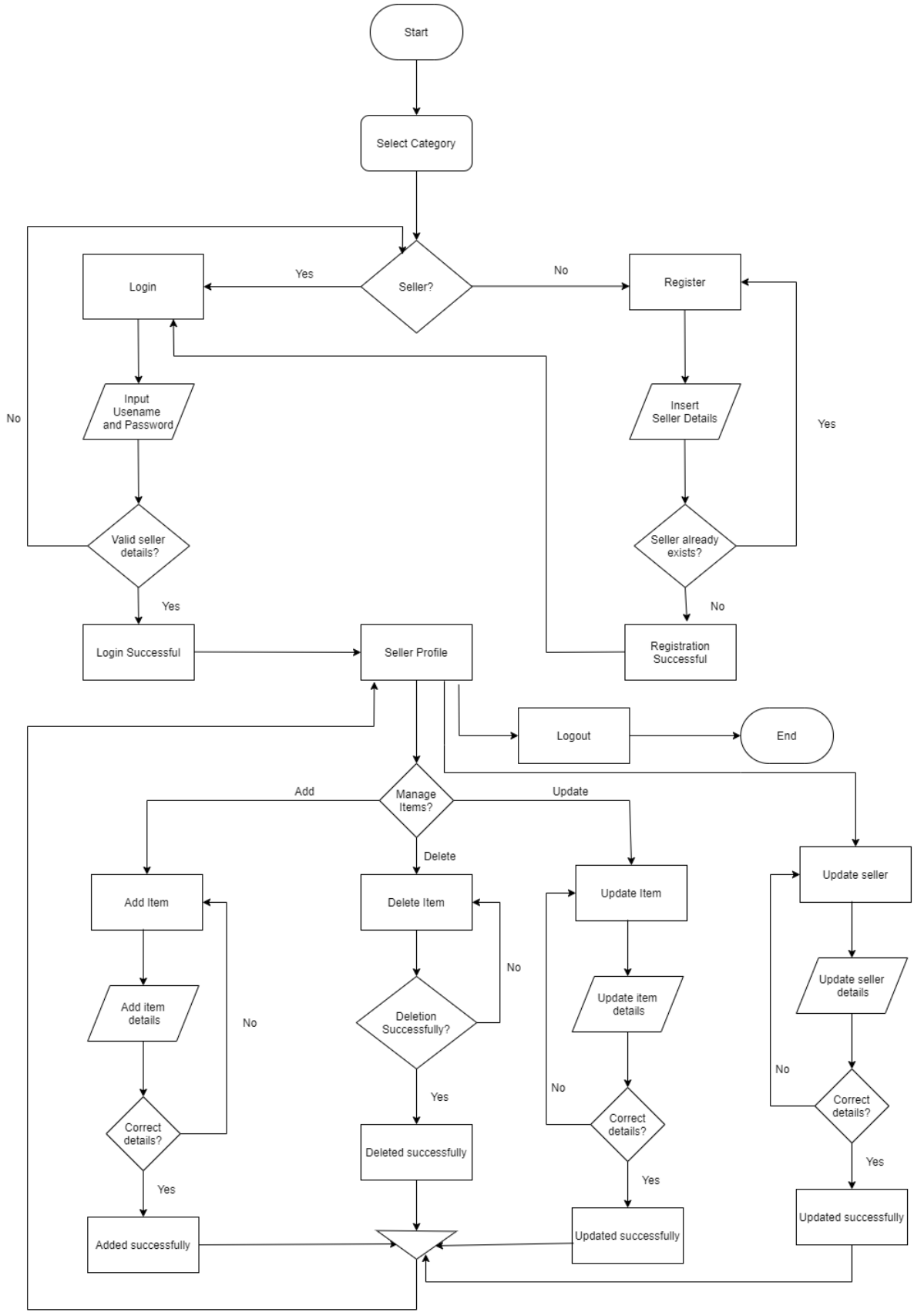
- React JS – Frontend
- Spring Boot – Backend
- MySQL - Database

Diagrams on Restful Web Services Interconnectivity.

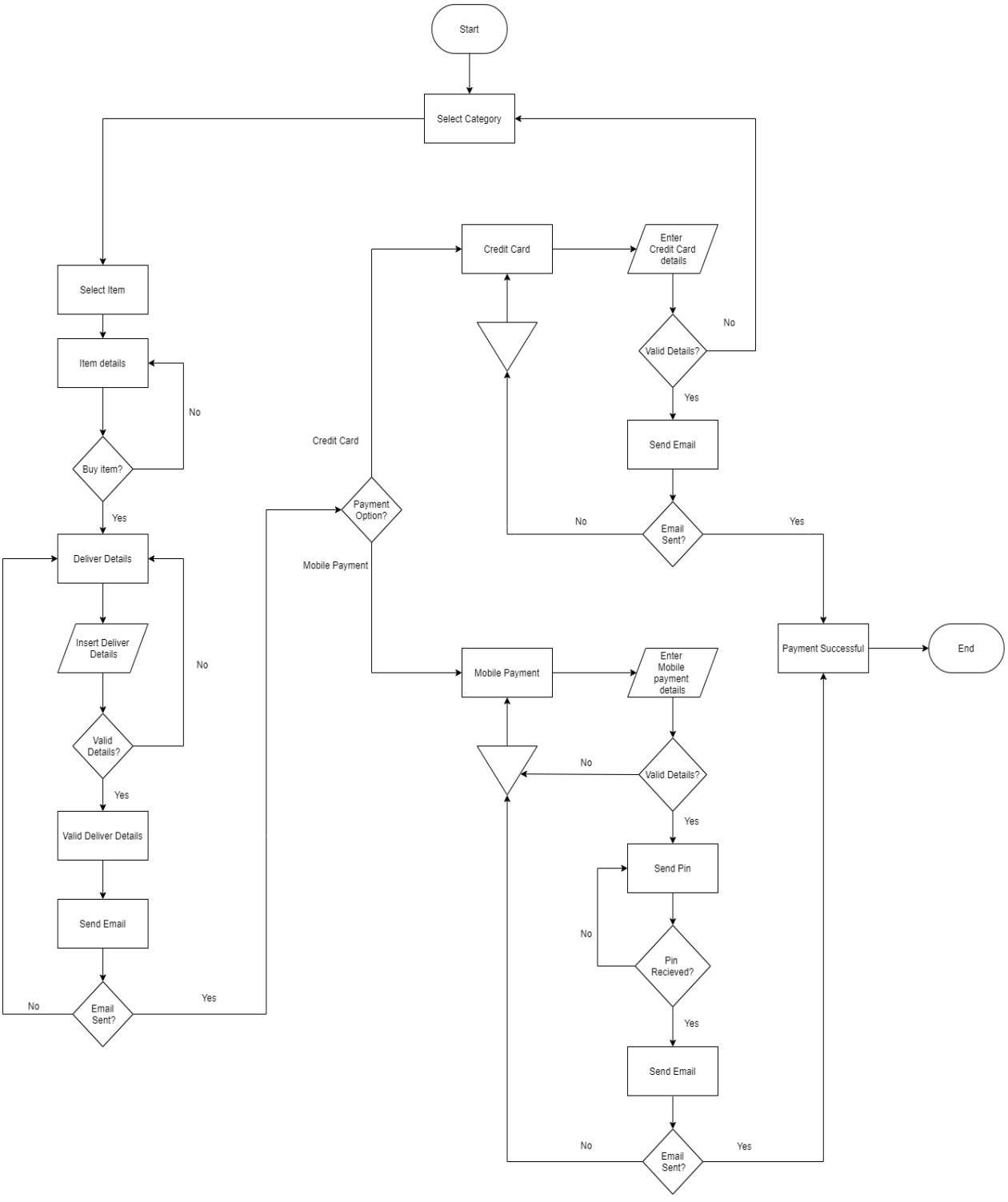
Activity Diagram



Flow Chart - Seller



Flow Chart - Buyer



List of Services & their Service Interfaces

	Category Service	Seller Service	Item Service	Delivery Service
Model	Category.java	Seller.java	Item.java	DeliverModel.java
Controller	CategoryController.java	SellerController.java	ItemController.java	DeliverController.java
Repository	CategoryRepository.java	SellerRepository.java	ItemRepository.java	DeliverRepository.java
Service	CategoryService.java	SellerService.java	ItemService.java	
Error Handler	MaxSizeErrorHandler.java	BadRequestException.java	MaxSizeErrorHandler.java	
Configurations				

	Mobile Payment Service	Card Payment Service	Email Service
Model	BillPayment.java, OTPSystem.java	PaymentClass.java	
Controller	BillPaymentController.java, OTPSystemRESTController.java	paypalControl.java	emailControl.java
Repository	BillPaymentRepo.java		
Service		paypalService.java	DeliverMail.java, emailService.java, NotificationService.java
Error Handler	ResourceNotFoundException.java		
Configurations		PaypalConfiguration.java	

Explanation on Services

Seller Service

In the Seller Service, a seller can login to their seller profile page to, view, update their seller profile details. If the new user wants to be a seller in our system, they can register their selves and login to their profile.

Item Service

Item service allows the sellers to add, view, update and delete their items in the seller profile page. And it also allows to display the available items to the user(buyer) in the display Items Page, and when the buyer selects a specific item from all the items, it will lead to another page where that page will display all the details of that selected item. These services are all provided through the Item Service.

```
spring.servlet.multipart.max-file-size=10MB
spring.servlet.multipart.max-request-size=10MB
```

The above code snippet restricts from uploading large image files of the item.

Category Service

Category service displays all the available categories to the user(buyer) in the category page. So this category page is the first page in the system which will help the buyer or seller to navigate to the Item or Seller services.

Delivery Service

When buyer click on the Buy now button, Browser navigate to the deliver page. There system asks for buyer name, buyer email address, buyer delivery address, buyer phone number, delivery option. Payment method. Total payment is calculate based on buyer's delivery option. If buyer select the deliver method as Fast delivery, buyer must pay additional 10 percent of item price. If all the details are validated two emails are send to Delivery Service and buyer. But if the buyer's selection is cash on delivery, he does not need to pay additional money and, he or she will not receive and email from the system. All the details will save inside the database. According to the buyer's selected payment option browser will navigate to either Card payment page or Mobile Payment page. In order to send an email to the buyer and the Delivery service I have use mail host as Gmail. And I have declared email username as my email, password as two step verification app password and also, enabled the authentication of smtp service. All of above things have done inside the application.properties file. Following code snippet is explains it.

```
spring.mail.host = smtp.gmail.com
spring.mail.username = [redacted]6@gmail.com
spring.mail.password = [redacted]

spring.mail.properties.mail.smtp.auth = true
spring.mail.port = 587
spring.mail.properties.mail.smtp.starttls.enable = true
```

Card Payment Service

In the Card Payment Service, system first displays a payment page with Payment type (as PayPal), Total amount, Description (Item code or Item name), Currency (as USD) and Intent (as Sale). After buyer selects Pay Now button the system redirects to PayPal Payment Gateway page. In order to redirect PayPal Payment gateway, the business should have PayPal Business account but in here we created and used a Sandbox PayPal account for the Payment Gateway. And in order to send the money to the sandbox account we added the Client id and Client secret code of PayPal account in application properties file as shown below.

```
paypal.mode=sandbox
paypal.client.id=ATMGsRBoVJ0GROBsIUUa574jwq_ZMjA9_RRN_75BoIux6JPhmnc3Y8pHxfOmvzlmuOD2Ym_nSoRXN32A
paypal.client.secret=EC_Ls_MxSPnx1p_FTUbu7YhjBoK4nZlVUs2RTPnbjf-Z9AoHYl9uftS1RwZWi3uOBG43o-OBLf0TZhaO
```

And in Payment Gateway page we choose Pay with Credit or Debit card button and it redirects to Checkout page. In PayPal Payment Gateway checkout page buyer should provide the Card Number, CVC, Card expiry date, Buyer's First name and Last name, Billing address, City, Phone Number and Email address (For test purpose we used test Credit Card Number, CVC and Expiry date provided by PayPal Developer account). After adding the details, the buyer clicks Continue button and if the provided card details are correct and the payment is success and card payment confirmation email is sent the system shows an alert box as Your Payment Transaction is Successful. If the provided card details are incorrect system shows an alert as Your Payment Transaction is Failed.

Card Payment Email Service

In card payment email service in order send an email we added the send email's Username and Password in application properties file.

```
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=yadev      @gmail.com
spring.mail.password=r          4
```

The email content to be send is added in emailService.java file. After buyer clicks the Continue button in PayPal Checkout page PayPal will send a json response to the system and we will get relevant details and the buyers email address from the json response and the relevant details will be added to the email content and the Card Payment Confirmation email will be sent to the buyer's email address which we got it from the PayPal json response. Only after that payment success alert will be shown.

Payment using mobile service.

When the user gives details to Deliver Page and confirm details buy pressing "Pay now" button it will send a 4-digit OTP number to customer's phone. The message service handle by "Twilio" APIs. After registering with Twilio, it provides Account Sid, Authentication token and a phone number using the we can use "Twilio" APIs. After pressing "Pay now" button, system will be redirected to "Confirm pin" page. Now user should provide the 4-digit pin which received through a SMS to the user's phone. If the pin is incorrect user can get another pin. If the pin is correct payment will be successful and system navigates the user to category list page.

```
ACCOUNT_SID = "AC0966b9e047313bee0c192d481601d1bd"
AUTH_TOKEN = "3d6e26ec859e005ee6260edb36c7826c"
PHONE_NUMBER = "+19705003288"
```

Mobile Payment Email service.

After user confirm the payment by pressing "Confirm" button, the system will send an email to user and the seller of that particular item.

```
spring.mail.username = dhananjayaarw@gmail.com
spring.mail.password = ibynqjalrwrzwuzb
```

Appendix with all the Codes.

Backend

Seller Service

1. Seller.java

```
package shop.online.model;
import javax.persistence.*;

@Entity
@Table(name = "seller")
public class Seller {
    private int sellerId;
    private String sellerName;
    private String sellerAddress;
    private String sellerPhone;
    private String sellerEmail;
    private String sellerPassword;
    private String sellerZip;
    private String sellerAccNo;

    public Seller() {

    }

    public Seller(int sellerId, String sellerName, String sellerAddress, String
sellerPhone, String sellerEmail,String sellerPassword, String sellerZip, String
sellerAccNo) {
        this.sellerId = sellerId;
        this.sellerName = sellerName;
        this.sellerAddress = sellerAddress;
        this.sellerPhone = sellerPhone;
        this.sellerEmail = sellerEmail;
        this.sellerPassword = sellerPassword;
        this.sellerZip = sellerZip;
        this.sellerAccNo = sellerAccNo;
    }

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    public int getSellerId() {
        return sellerId;
    }

    public void setSellerId(int sellerId) {
        this.sellerId = sellerId;
    }

    public String getSellerName() {
        return sellerName;
    }
```

```
    public void setSellerName(String sellerName) {
        this.sellerName = sellerName;
    }

    public String getSellerAddress() {
        return sellerAddress;
    }

    public void setSellerAddress(String sellerAddress) {
        this.sellerAddress = sellerAddress;
    }

    public String getSellerPhone() {
        return sellerPhone;
    }

    public void setSellerPhone(String sellerPhone) {
        this.sellerPhone = sellerPhone;
    }

    public String getSellerEmail() {
        return sellerEmail;
    }

    public void setSellerEmail(String sellerEmail) {
        this.sellerEmail = sellerEmail;
    }

    public String getSellerPassword() {
        return sellerPassword;
    }

    public void setSellerPassword(String sellerPassword) {
        this.sellerPassword = sellerPassword;
    }

    public String getSellerZip() {
        return sellerZip;
    }

    public void setSellerZip(String sellerZip) {
        this.sellerZip = sellerZip;
    }

    public String getSellerAccNo() {
        return sellerAccNo;
    }

    public void setSellerAccNo(String sellerAccNo) {
        this.sellerAccNo = sellerAccNo;
    }
}
```

2. SellerRepository.java

```
package shop.online.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import shop.online.model.Seller;

public interface SellerRepository extends JpaRepository<Seller, Integer> {
    Seller findBySellerEmail(String sellerEmail);
    Seller findBySellerPassword(String sellerPassword);
}
```

3. SellerService.java

```
package shop.online.service;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.jpa.repository.config.EnableJpaAuditing;
import org.springframework.stereotype.Service;
import shop.online.errorHandler.BadRequestException;
import shop.online.model.Seller;
import shop.online.repository.SellerRepository;
@Service
@EnableJpaAuditing
public class SellerService {

    @Autowired
    private SellerRepository sellerRepository;

    public Seller loginSeller(Seller seller) {
        Seller sellerExists =
sellerRepository.findBySellerEmail(seller.getSellerEmail());
        Seller sellerExists1 =
sellerRepository.findBySellerPassword(seller.getSellerPassword());
        if (sellerExists == null) {
            throw new BadRequestException("Invalid user name.");
        }
        if (sellerExists1 == null) {
            throw new BadRequestException("Invalid password.");
        }
        String sellerPassword = seller.getSellerPassword();
        if (sellerPassword.isEmpty()) {
            throw new BadRequestException("Invalid user name and password
combination.");
        }
        sellerExists.setSellerPassword("");
        sellerExists.setSellerId(0);
        return sellerExists;
    }
}
```

4. SellerController.java

```
package shop.online.controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import shop.online.model.Seller;
import shop.online.repository.SellerRepository;
import shop.online.service.SellerService;
import java.util.Optional;
import java.util.NoSuchElementException;
```

```
@CrossOrigin(origins = "http://localhost:3000")
@RestController
@RequestMapping("/api")
public class SellerController {

    @Autowired
    SellerRepository sellerRepository;
    @Autowired
    SellerService sellerService;

    /*For displaying seller information(seller-profile)*/
    @GetMapping("/seller/{sellerId}")
    public Optional<Seller> findSeller(@PathVariable Integer sellerId) {
        return sellerRepository.findById(sellerId);
    }
    /*For adding sellers to the system(seller-registration)*/
    @PostMapping("/seller")
    public void addSeller(@RequestBody Seller seller) {
        sellerRepository.save(seller);
    }
    /*For updating sellers information in the system(seller-update)*/
    @PutMapping("/seller/{sellerId}")
    public ResponseEntity<?> updateSeller(@RequestBody Seller seller,
@PathVariable Integer sellerId){
        try {
            Seller seller1 = sellerRepository.getOne(sellerId);
            seller1.setSellerId(sellerId);
            sellerRepository.save(seller);
            return new ResponseEntity<>(HttpStatus.OK);
        } catch (NoSuchElementException e){
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
    }
    @PostMapping("/seller/login")
    public Seller login(@RequestBody Seller seller) {
        return sellerService.loginSeller(seller);
    }
}
```

Item Service

1. Item.java

```
package shop.online.model;
import org.springframework.web.multipart.MultipartFile;
import javax.persistence.*;
@Entity
@Table(name = "item")
public class Item {

    private int itemId;
    private String itemName;
    private String itemPrice;
    private String itemQuantity;
    private String itemDescription;
    private String sellerNum;
    private String catName;
```

```
    @Lob
    private byte[] itemImage;
    private String sellerEmail;

public Item() {

    }

    public Item(String itemName, String itemPrice, String itemQuantity, String
itemDescription, String sellerNum, String catName, byte[] itemImage,String
sellerEmail) {
        this.itemName = itemName;
        this.itemPrice = itemPrice;
        this.itemQuantity = itemQuantity;
        this.itemDescription = itemDescription;
        this.sellerNum = sellerNum;
        this.catName = catName;
        this.itemImage = itemImage;
        this.sellerEmail = sellerEmail;
    }
      @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    public int getItemId() {
        return itemId;
    }
    public void setItemId(int itemId) {
        this.itemId = itemId;
    }
    public String getItemName() {
        return itemName;
    }
    public void setItemName(String itemName) {
        this.itemName = itemName;
    }
    public String getItemPrice() {
        return itemPrice;
    }
    public void setItemPrice(String itemPrice) {
        this.itemPrice = itemPrice;
    }
    public String getItemQuantity() {
        return itemQuantity;
    }
    public void setItemQuantity(String itemQuantity) {
        this.itemQuantity = itemQuantity;
    }
    public String getItemDescription() {
        return itemDescription;
    }
    public void setItemDescription(String itemDescription) {
        this.itemDescription = itemDescription;
    }
    public String getSellerNum() {
        return sellerNum;
    }
    public void setSellerNum(String sellerNum) {
        this.sellerNum = sellerNum;
    }
    public String getCatName() {
```

```
        return catName;
    }
    public void setCatName(String catName) {
        this.catName = catName;
    }
    public byte[] getItemImage() {
        return itemImage;
    }
    public void setItemImage(byte[] itemImage) {
        this.itemImage = itemImage;
    }
    public String getSellerEmail() {
            return sellerEmail;
      }
      public void setSellerEmail(String sellerEmail) {
            this.sellerEmail = sellerEmail;
      }
}
```

2. ItemRepository.java

```
package shop.online.repository;

import java.util.List;
import java.util.Optional;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import shop.online.model.Item;

public interface ItemRepository extends JpaRepository<Item, Integer> {
    /*For viewing item(viewItem-of a seller)-without blob*/
    @Query("SELECT i FROM Item i WHERE i.sellerNum = ?1")
    List<Item> findByPhone(String sellerNum);
    /*For displaying items for buyers-without blob*/
    @Query("SELECT i FROM Item i WHERE i.catName = ?1")
    List<Item> findByCatName(String catName);
}
```

3. ItemService.java

```
package shop.online.service;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import shop.online.model.Item;
import shop.online.repository.ItemRepository;
import java.util.Collections;
import java.util.List;
import java.util.Optional;
import java.util.stream.Stream;
@Service
public class ItemService {

    private static final Logger logger =
LoggerFactory.getLogger("ItemService.class");
```

```
    @Autowired
    private ItemRepository itemRepository;
    /*For adding item(addItem)-WITH BLOB*/
    public int saveImage(Item item) {
        try{
             itemRepository.save(item);
            return 1;
        } catch (Exception e) {
            logger.error("Error", e);
            return 0;
        }
    }
    /*For displaying single page item(viewSinglePageItem)-WITH BLOB*/
    public Item getSinglePage(int id) {
        Optional findById = itemRepository.findById(id);
        if (findById.isPresent()) {
            Item getImageDetails = (Item) findById.get();
            logger.info("id= " + getImageDetails.getItemId()
                    + " name= " + getImageDetails.getItemName()
                    + "price= " + getImageDetails.getItemPrice()
                    + "quantity= "+ getImageDetails.getItemQuantity()
                    + "des= "+ getImageDetails.getItemDescription()
                    + "sellerNum= "+ getImageDetails.getSellerNum()
                    + "category= "+ getImageDetails.getCatName()
                    + "image= "+ getImageDetails.getItemImage()
                    + "sellerEmail= "+ getImageDetails.getSellerEmail()
            );
            return getImageDetails;
        } else {
            return null;
        }
    }
    /*For displaying a sellers items (viewSellerPageItem)-WITH BLOB*/
    public List<Item> getAllSellerItems(String sellerNum) {
        return itemRepository.findByPhone(sellerNum);
    }
    /*For displaying all items for buyers(viewItemsPage)-WITH BLOB*/
    public List<Item> getBuyerItems(String catName) {
        return  itemRepository.findByCatName(catName);
    }
}
```

4. ItemController.java

```
package shop.online.controller;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;
import shop.online.model.Item;
import shop.online.repository.ItemRepository;
import shop.online.service.ItemService;
import java.util.List;
import java.util.NoSuchElementException;
```

```java
import java.util.Optional;
@CrossOrigin(origins = "http://localhost:3000")
@Controller
@RestController
@RequestMapping("/api")
public class ItemController {
    private static final Logger logger =
LoggerFactory.getLogger("ItemController.class");
    @Autowired
    private ItemService itemService;
    @Autowired
    ItemRepository itemRepository;
    /*For adding item(addItem)-without blob*/
    @PostMapping("/")
    public void addItem(@RequestBody Item item) {
        itemRepository.save(item);
    }
    /*For viewing item(viewItem-of a seller)-without blob*/
    @GetMapping("/items/{sellerNum}")
    public List<Item> viewSellerItems(@PathVariable String sellerNum) {
        return itemRepository.findByPhone(sellerNum);
    }
    /*For updating item(updateItem-of a seller)*/
    @PutMapping("/items/{itemId}")
    public ResponseEntity<?> updateItem(@RequestBody Item item, @PathVariable
Integer itemId) {
        try {
            Item item1 = itemRepository.getOne(itemId);
            item1.setItemId(itemId);
            itemRepository.save(item);
            return new ResponseEntity<>(HttpStatus.OK);
        } catch (NoSuchElementException e) {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
    }
    /*For deleting an item(deleteItem-of a seller)*/
    @DeleteMapping("/items/{itemId}")
    public void deleteItem(@PathVariable Integer itemId) {
        itemRepository.deleteById(itemId);
    }
    /*For displaying items for buyers-without blob*/
    @GetMapping("/buyer/{catName}")
    public List<Item> displayItemsBuyer(@PathVariable String catName) {
        return itemRepository.findByCatName(catName);
    }
    /*For displaying single page item for buyers-without blob*/
    @GetMapping("/buyerView/{itemId}")
    public Optional<Item> displaySinglePage(@PathVariable int itemId) {
        return itemRepository.findById(itemId);
    }
    /*For adding item(addItem)-WITH BLOB*/
    @PostMapping("/fileupload")
    public String fileUpload(@RequestParam(required = false, name = "itemName")
String itemName,
                             @RequestParam(required = false, name = "itemPrice")
String itemPrice,
                             @RequestParam(required = false, name =
"itemQuantity") String itemQuantity,
```

```
                                @RequestParam(required = false, name =
"itemDescription") String itemDescription,
                                @RequestParam(required = false, name = "sellerNum")
String sellerNum,
                                @RequestParam(required = false, name = "catName")
String catName,
                                @RequestParam(required = false, name = "file")
MultipartFile file,
                                @RequestParam(required = false, name = "sellerEmail")
String sellerEmail) {
        try {
            byte[] itemImage = file.getBytes();
            Item item = new Item(itemName, itemPrice, itemQuantity,
itemDescription, sellerNum, catName, itemImage,sellerEmail);
            int saveImage = itemService.saveImage(item);
            if (saveImage == 1) {
                return "success";
            } else {
                return "error";
            }
        } catch (Exception e) {
            logger.error("ERROR", e);
            return "error";
        }
    }
    /*For displaying single page item(viewSinglePageItem)-WITH BLOB*/
    @GetMapping("/getSingle/{itemId}")
    public String getDBSingle(@PathVariable String itemId, Model model){
        try{
            logger.info("Id= " + itemId);
            Item imagesObj = itemService.getSinglePage(Integer.parseInt(itemId));
            model.addAttribute("id",imagesObj.getItemId());
            model.addAttribute("name",imagesObj.getItemName());
            model.addAttribute("price",imagesObj.getItemPrice());
            model.addAttribute("quantity",imagesObj.getItemQuantity());
            model.addAttribute("description",imagesObj.getItemDescription());
            byte[] encode =
java.util.Base64.getEncoder().encode(imagesObj.getItemImage());
            model.addAttribute("image", new String (encode,"UTF-8"));
            model.addAttribute("email",imagesObj.getSellerEmail());
            return "Single Image";
        }catch (Exception e) {
            logger.error("Error", e);
            model.addAttribute("message", "Error in getting image");
            return "redirect:/";
        }
    }
    /*For displaying a sellers items (viewSellerPageItem)-WITH BLOB*/
    @GetMapping("/viewSeller/{sellerNum}")
    String showSellerItems (@PathVariable String sellerNum, Model map) {
        List<Item> imagesObj1 = itemService.getAllSellerItems(sellerNum);
        map.addAttribute("products", imagesObj1);
        return "View Seller Items";
    }
    /*For displaying all items for buyers(viewItemsPage)-WITH BLOB*/
    @GetMapping("/viewItems/{catName}")
    public String showBuyerItems (@PathVariable String catName, Model model1) {
        try {
            logger.info("catName= " + catName);
```

```
            List<Item> imagesObj2 = itemService.getBuyerItems(catName);
            model1.addAttribute("All Items", imagesObj2);
//            byte[] encode =
java.util.Base64.getEncoder().encode(imagesObj2.get(8));
//            model1.addAttribute("image", new String(encode,"UTF-8"));
            return "View All Items(Buyer)";
        } catch (Exception e) {
            logger.error("Error", e);
            model1.addAttribute("message", "Error in getting image");
            return "redirect:/";

        }
    }
}
```

Category Service

1. **Category.java**

```
package shop.online.model;
import javax.persistence.*;
@Entity
@Table(name = "category")
public class Category {
    private int categoryId;
    private String categoryName;
    @Lob
    private byte[] catImage;

    public Category() {
    }

    public Category(int categoryId, String categoryName) {
        this.categoryId = categoryId;
        this.categoryName = categoryName;
    }

    public Category(int categoryId, String categoryName, byte[] catImage) {
        this.categoryId = categoryId;
        this.categoryName = categoryName;
        this.catImage = catImage;
    }
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)

    public int getCategoryId() {
        return categoryId;
    }
    public void setCategoryId(int categoryId) {
        this.categoryId = categoryId;
    }
    public String getCategoryName() {
        return categoryName;
    }
    public void setCategoryName(String categoryName) {
        this.categoryName = categoryName;
    }
```

```
    public byte[] getCatImage() {
        return catImage;
    }
    public void setCatImage(byte[] catImage) {
        this.catImage = catImage;
    }
}
```

2. **CategoryService.java**

```
package shop.online.service;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import shop.online.model.Category;
import shop.online.repository.CategoryRepository;
import java.util.List;
@Service
public class CategoryService {
    private static final Logger Logger =
LoggerFactory.getLogger("ItemService.class");
    @Autowired
    private CategoryRepository categoryRepository;
    /*For displaying all categories for buyers(viewItemsPage)-WITH BLOB*/
    public List<Category> getAllCategories() {
        return categoryRepository.findAll();
    }
}
```

3. **CategoryRepository.java**

```
package shop.online.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import shop.online.model.Category;

public interface CategoryRepository extends JpaRepository<Category, Integer> {
}
```

4. **CategoryController.java**

```
package shop.online.controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import shop.online.model.Category;
import shop.online.repository.CategoryRepository;
import shop.online.service.CategoryService;
import java.util.List;
import java.util.Optional;
@CrossOrigin(origins = "http://localhost:3000")
@RestController
@RequestMapping("/api")
public class CategoryController {

    @Autowired
    CategoryRepository categoryRepository;
    @Autowired
    private CategoryService categoryService;
```

```
    /*For displaying all categories for buyers(viewItemsPage)-WITH BLOB*/
    @GetMapping("/viewCategories")
    public String showCategories(Model model) {
        List<Category> imageObj3 = categoryService.getAllCategories();
        model.addAttribute("categories",imageObj3);
        return "View All Categories";
    }
    @GetMapping("/categories")
    public List<Category> displayCategories() {
        return categoryRepository.findAll();
    }
    @GetMapping("/categories/{categoryId}")
    public Optional<Category> displayCategories1(@PathVariable int categoryId) {
        return categoryRepository.findById(categoryId);
    }
}
```

5. **MaxSizeErrorHandler.java**

```
package shop.online.errorHandler;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.multipart.MultipartException;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;
@ControllerAdvice
public class MaxSizeErrorHandler {
    private static final Logger logger =
LoggerFactory.getLogger("MaxSizeErrorHandler.class");
    @ExceptionHandler
    public String maxSizeError(MultipartException e, RedirectAttributes
redirectAttributes) {
        logger.info("Max File Size Exception Occurs");
        redirectAttributes.addFlashAttribute("message", "Exceed Max File Size
Error");
        return "redirect:/";
    }
}
```

6. **BadRequestException.java**

```
package shop.online.errorHandler;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;
@ResponseStatus(HttpStatus.BAD_REQUEST)
public class BadRequestException extends RuntimeException {
    private static final long serialVersionUID = 1L;
    public BadRequestException (String message) {
        super(message);
    }
    public BadRequestException(String message, Throwable cause) {
        super (message, cause);
    }
}
```

Delivery Service

1. **DeliverModel.java**

```
package shop.online.model;
import javax.persistence.*;
@Entity
@Table(name="deliveries")
public class DeliverModel {
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private long deliverid;
	@Column(name="buyerName")
	private String buyerName;
	@Column(name="buyerEmail")
	private String buyerEmail;
	@Column(name="buyerAddress")
	private String buyerAddress;
	@Column(name="buyerPhone")
	private String buyerPhone;
	@Column(name="deliverItemName")
	private String deliverItemName;
	@Column(name="deliveryCash")
	private double deliveryCash;
	@Column(name="totalPayment")
	private double totalPayment;
	@Column(name="deliveryOption")
	private String deliveryOption;
	@Column(name="payOption")
	private String payOption;

	public DeliverModel() {
	}


	public DeliverModel( String buyerName,String buyerEmail, String
buyerAddress, String buyerPhone, String deliverItemName,
			String deliveryOption,double totalPayment,double
deliveryCash,String payOption) {
		super();

		this.buyerName = buyerName;
		this.buyerEmail = buyerEmail;
		this.buyerAddress = buyerAddress;
		this.buyerPhone = buyerPhone;
		this.deliverItemName = deliverItemName;
		this.totalPayment = totalPayment;
		this.deliveryCash = deliveryCash;
		this.deliveryOption = deliveryOption;

		this.payOption = payOption;
	}
	public long getDeliverid() {
		return deliverid;
	}
	public void setDeliverid(int deliverid) {
		this.deliverid = deliverid;
	}
```

```
	public String getBuyerName() {
		return buyerName;
	}
	public void setBuyerName(String buyerName) {
		this.buyerName = buyerName;
	}
	public String getBuyerEmail() {
		return buyerEmail;
	}
	public void setBuyerEmail(String buyerEmail) {
		this.buyerEmail = buyerEmail;
	}
	public String getBuyerAddress() {
		return buyerAddress;
	}
	public void setBuyerAddress(String buyerAddress) {
		this.buyerAddress = buyerAddress;
	}
	public String getBuyerPhone() {
		return buyerPhone;
	}
	public void setBuyerPhone(String buyerPhone) {
		this.buyerPhone = buyerPhone;
	}
	public String getDeliverItemName() {
		return deliverItemName;
	}
	public void setDeliverItemName(String deliverItemName) {
		this.deliverItemName = deliverItemName;
	}
	public double getTotalPayment() {
		return totalPayment;
	}
	public void setTotalPayment(double totalPayment) {
		this.totalPayment = totalPayment;
	}
	public String getDeliveryOption() {
		return deliveryOption;
	}
	public void setDeliveryOption(String deliveryOption) {
		this.deliveryOption = deliveryOption;
	}
	public double getDeliveryCash() {
		return deliveryCash;
	}
	public void setDeliveryCash(double deliveryCash) {
		this.deliveryCash = deliveryCash;
	}
	public String getPayOption() {
		return payOption;
	}
	public void setPayOption(String payOption) {
		this.payOption = payOption;
	}
}
```

2. **DeliverMail.java**

```
package shop.online.service;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mail.SimpleMailMessage;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.stereotype.Service;
@Service
public class DeliverMail {
@Autowired
private JavaMailSender javaMailSender;
public void sendEmail(String toEmail, String body, String subject) {
		SimpleMailMessage message = new SimpleMailMessage();
		message.setFrom("sidathrathnayake96@gmail.com");
		message.setTo(toEmail);
		message.setText(body);
		message.setSubject(subject);
		javaMailSender.send(message);
		System.out.println("Mail Send");
	}
}
```

3. **DeliverRepository.java**

```
package shop.online.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import shop.online.model.DeliverModel;
public interface DeliverRepository extends JpaRepository<DeliverModel,Long> {
}
```

4. **DeliverController.java**

```
package shop.online.controller;
import java.io.IOException;
import java.net.URISyntaxException;
import java.util.*;
import javax.mail.MessagingException;
import javax.mail.internet.AddressException;
import javax.mail.internet.MimeMessage;
import javax.servlet.http.HttpServletResponse;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.context.event.ApplicationReadyEvent;
import org.springframework.boot.web.servlet.error.ErrorController;
import org.springframework.context.event.EventListener;
import org.springframework.core.io.ClassPathResource;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.mail.MailException;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.MimeMessageHelper;
import org.springframework.stereotype.Service;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.RestController;
import shop.online.model.DeliverModel;
import shop.online.repository.DeliverRepository;
import shop.online.service.DeliverMail;
@CrossOrigin(origins="http://localhost:3000")
```

```
@RestController
@RequestMapping("/api")
@Service
public class DeliverController implements ErrorController {

	@Autowired
	DeliverRepository deliverRepository;
	@Autowired
	private DeliverMail mail;

	@PostMapping("/deliveries")
	public DeliverModel createDelivery(@RequestBody DeliverModel deliverModel ,
HttpServletResponse response)throws URISyntaxException, IOException{

	if(deliverModel.getDeliveryOption().toString().equals("FastDelivery")) {
			mail.sendEmail("sidathrathnayake96@gmail.com", "You have to
send a package to, \n Customer Name : "+ deliverModel.getBuyerName() + ". \n
Customer Address : " + deliverModel.getBuyerAddress() + ".\n Customer Contact
number : " + deliverModel.getBuyerPhone() , "Package to Deliver");
			mail.sendEmail(deliverModel.getBuyerEmail(), "We have
informed our deliver Service . They will deliver your package very soon. Please
contact them if you want with following email address. \n Thank you for choosing
our service.\n Deliver Service Email : sidathrathnayake96@gmail.com." , "Informed
the Delivery Service");
			return deliverRepository.save(deliverModel);
		}
		else {
			return deliverRepository.save(deliverModel);
		}
	}
	@Override
	public String getErrorPath() {
		return null;
	}
}
```

Card Payment Service

1. **PaymentClass.java**

```
package shop.online.model;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.ToString;
@Data
@AllArgsConstructor
@NoArgsConstructor
@ToString
public class PaymentClass {
	private double total;
    private String currency;
    private String paytype;
    private String intent;
    private String description;
	public double getTotal() {
		return total;
```

```
	}
	public void setTotal(double total) {
		this.total = total;
	}
	public String getCurrency() {
		return currency;
	}
	public void setCurrency(String currency) {
		this.currency = currency;
	}
	public String getPaytype() {
		return paytype;
	}
	public void setPaytype(String paytype) {
		this.paytype = paytype;
	}
	public String getIntent() {
		return intent;
	}
	public void setIntent(String intent) {
		this.intent = intent;
	}
	public String getDescription() {
		return description;
	}
	public void setDescription(String description) {
		this.description = description;
	}
}
```

2. **paypalService.java**

```
package shop.online.service;
import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.ArrayList;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.paypal.api.payments.Amount;
import com.paypal.api.payments.Payer;
import com.paypal.api.payments.Payment;
import com.paypal.api.payments.PaymentExecution;
import com.paypal.api.payments.RedirectUrls;
import com.paypal.api.payments.Transaction;
import com.paypal.base.rest.APIContext;
import com.paypal.base.rest.PayPalRESTException;
@Service
public class paypalService {

	@Autowired
    private APIContext apiContext;
    public Payment createPayment(
             String payType,
            String description,
            Double totalCharge,
            String currency,
            String intent,
            String failUrl,
```

```
            String successUrl) throws PayPalRESTException {
        Amount amt = new Amount();
        amt.setCurrency(currency);
        totalCharge = new BigDecimal(totalCharge).setScale(2,
RoundingMode.HALF_UP).doubleValue();
        amt.setTotal(String.format("%.2f", totalCharge));

        Transaction transaction = new Transaction();
        transaction.setDescription(description);
        transaction.setAmount(amt);

        List<Transaction> transactions = new ArrayList<>();
        transactions.add(transaction);

        Payer payer = new Payer();
        payer.setPaymentMethod(payType.toString());

        Payment payment = new Payment();
        payment.setIntent(intent.toString());
        payment.setPayer(payer);
        payment.setTransactions(transactions);
        RedirectUrls redirectUrls = new RedirectUrls();
        redirectUrls.setCancelUrl(failUrl);
        redirectUrls.setReturnUrl(successUrl);
        payment.setRedirectUrls(redirectUrls);

        return payment.create(apiContext);
    }
    public Payment executePayment(String paymentId, String payerId) throws
PayPalRESTException{
        Payment payment = new Payment();
        payment.setId(paymentId);
        PaymentExecution paymentExecute = new PaymentExecution();
        paymentExecute.setPayerId(payerId);
        return payment.execute(apiContext, paymentExecute);
    }
}
```

3. emailService.java

```
package shop.online.service;
import java.io.File;
import java.io.IOException;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.List;
import javax.mail.MessagingException;
import javax.mail.internet.MimeMessage;
import com.paypal.api.payments.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.FileSystemResource;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.MimeMessageHelper;
import org.springframework.stereotype.Service;
import org.springframework.web.bind.annotation.ModelAttribute;
import com.google.gson.JsonObject;
@Service
public class emailService {
```

```
    @Autowired
    private JavaMailSender javaMailSender;
    public String sendEmail(Payment pay) throws MessagingException {
          MimeMessage msg = javaMailSender.createMimeMessage();
      // true = multipart message
      MimeMessageHelper helper = new MimeMessageHelper(msg, true);
      Payer payer = pay.getPayer();
      PayerInfo payerInfo = pay.getPayer().getPayerInfo();
      Transaction transaction = pay.getTransactions().get(0);
      String email =payerInfo.getEmail();
      helper.setTo(email);

      helper.setSubject("Payment Receipt of Shop Online");

      String name = payerInfo.getShippingAddress().getRecipientName();
      String currency=transaction.getAmount().getCurrency();
      String amount=transaction.getAmount().getTotal();
      String item=transaction.getDescription();
      String billAddress=payerInfo.getShippingAddress().getLine1();
      String ccode=payerInfo.getShippingAddress().getCountryCode();
      String pMethod=payer.getPaymentMethod();
      String city=payerInfo.getShippingAddress().getCity();
      RelatedResources rr = transaction.getRelatedResources().get(0);
      String receiptNo = rr.getSale().getReceiptId();
      String tranFee = rr.getSale().getTransactionFee().getValue();
           DateTimeFormatter dtf = DateTimeFormatter.ofPattern("dd-MMM-yyyy");
           LocalDate date = LocalDate.now();
           String curDate = dtf.format(date);
           String status = rr.getSale().getState().toUpperCase();

      helper.setText("<html>\r\n"
                  + "    <head>\r\n"
                  + "        <link
href=\"https://netdna.bootstrapcdn.com/bootstrap/3.0.0/css/bootstrap.min.css\"
rel=\"stylesheet\"/>\r\n"
                  + "        <script
src=\"https://netdna.bootstrapcdn.com/bootstrap/3.0.0/js/bootstrap.min.js\"></scri
pt>\r\n"
                  + "        <script src=\"https://code.jquery.com/jquery-
1.11.1.min.js\"></script>\r\n"
                  + "    </head>\r\n"
                  + "    <body align=\"center\">\r\n"
                  + "        <div class=\"container\">\r\n"
                  + "            <div class=\"row\">\r\n"
                  + "                <div class=\"well col-xs-10 col-sm-10 col-
md-6 col-xs-offset-1 col-sm-offset-1 col-md-offset-3\">\r\n"
                  + "                    <div class=\"row\">\r\n"
                  + "                        <div class=\"col-xs-6 col-sm-6 col-
md-6\">\r\n"
                  + "                            <address>\r\n"
                  + "                                <strong>Shop
Online</strong><br>\r\n"
                  + "                                1111 Peradeniya
Road,<br>\r\n"
                  + "                                Kandy, Sri Lanka<br>\r\n"
                  + "                                Phone No:(+94) 081-
2223335\r\n"
                  + "                            </address>\r\n"
                  + "                            <br>\r\n"
```

```
                    + "                             <p><em>Date : "+curDate+"
</em></p>\r\n"
                    + "                             <p><em>Receipt ID :
"+receiptNo+"</em></p>\r\n"
                    + "                         </div>\r\n"
                    + "                     </div>\r\n"
                    + "                     <div class=\"row\">\r\n"
                    + "                         <div class=\"text-center\">\r\n"
                    + "                             <h1>Payment Receipt</h1>\r\n"
                    + "                         </div>\r\n"
                    + "                         </span>\r\n"
                    + "                         <table class=\"table\">\r\n"
                    + "                             <tbody>\r\n"
                    + "                                 <tr>\r\n"
                    + "                                     <td class=\"col-md-
4\"><b><em>Full Name</em></b></h4></td>\r\n"
                    + "                                     <td class=\"col-md-6\">
"+name+" </td>\r\n"
                    + "                                 </tr>\r\n"
                    + "                                 <tr>\r\n"
                    + "                                     <td class=\"col-md-
4\"><b><em>Item Description</em></b></h4></td>\r\n"
                    + "                                     <td class=\"col-md-6\">
"+item+" </td>\r\n"
                    + "                                 </tr>\r\n"
                    + "                                 <tr>\r\n"
                    + "                                     <td class=\"col-md-
4\"><b><em>Payment Method</em></b></h4></td>\r\n"
                    + "                                     <td class=\"col-md-6\">
"+pMethod+" </td>\r\n"
                    + "                                 </tr>\r\n"
                    + "                                 <tr>\r\n"
                    + "                                     <td class=\"col-md-
4\"><b><em>Total Amount</em></b></h4></td>\r\n"
                    + "                                     <td class=\"col-md-6\">
"+amount+" </td>\r\n"
                    + "                                 </tr>\r\n"
                          + "
<tr>\r\n"
                          + "                                     <td class=\"col-
md-4\"><b><em>Transaction Fee</em></b></h4></td>\r\n"
                          + "                                     <td class=\"col-
md-6\"> "+tranFee+" </td>\r\n"
                          + "                                   </tr>\r\n"
                    + "                                 <tr>\r\n"
                    + "                                     <td class=\"col-md-
4\"><b><em>Currency Type</em></b></h4></td>\r\n"
                    + "                                     <td class=\"col-md-6\">
"+currency+" </td>\r\n"
                    + "                                 </tr>\r\n"
                    + "                                 <tr>\r\n"
                    + "                                     <td class=\"col-md-
4\"><b><em>Billing Address</em></b></h4></td>\r\n"
                    + "                                     <td class=\"col-md-6\">
"+billAddress+" </td>\r\n"
                    + "                                 </tr>\r\n"
                    + "                                 <tr>\r\n"
                    + "                                     <td class=\"col-md-
4\"><b><em>City</em></b></h4></td>\r\n"
```

```
                    + "                                    <td class=\"col-md-6\">
"+city+" </td>\r\n"
                    + "                                </tr>\r\n"
                    + "                                <tr>\r\n"
                    + "                                    <td class=\"col-md-
4\"><b><em>Country Code</em></b></h4></td>\r\n"
                    + "                                    <td class=\"col-md-6\">
"+ccode+" </td>\r\n"
                    + "                                </tr>\r\n"
                          + "                                    <tr>\r\n"
                          + "                                        <td class=\"col-
md-4\"><b><em>Payment Status</em></b></h4></td>\r\n"
                          + "                                        <td class=\"col-
md-6\"><strong><b>"+status+"</b></strong></td>\r\n"
                          + "                                    </tr>\r\n"
                    + "                            </tbody>\r\n"
                    + "                        </table>\r\n"
                    + "                    </div>\r\n"
                    + "                </div>\r\n"
                    + "            </div>\r\n"
                    + "        </div>    \r\n"
                    + "    </body>\r\n"
                    + "</html>", true);

        javaMailSender.send(msg);
        return "paySuccess";
      }
}
```

4. **paypalControl.java**

```
package shop.online.controller;
import javax.mail.MessagingException;
import com.paypal.api.payments.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestAttribute;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.view.RedirectView;
import com.paypal.base.rest.PayPalRESTException;
import shop.online.model.PaymentClass;
import shop.online.service.paypalService;
@CrossOrigin(origins = "http://localhost:3000")
@RestController
public class paypalControl {

      @Autowired
    private paypalService service;
    public static final String SUCCESS_URL = "pay/paySuccess";
    public static final String FAIL_URL = "pay/payFailed";
    //@GetMapping("/")
    //public String PayInput() {
```

```
      //  return "PayInput";
    //}
    @PostMapping("/pay")
    public String payment(@RequestBody PaymentClass pay) {
      try {
                    Payment payment =
service.createPayment(pay.getPaytype(),pay.getDescription(),pay.getTotal(),
pay.getCurrency(),
                                  pay.getIntent(), "http://localhost:8080/" +
FAIL_URL,
                                  "http://localhost:8080/" + SUCCESS_URL);
                    for(Links link:payment.getLinks()) {
                          if(link.getRel().equals("approval_url")) {
                                return link.getHref();
                          }
                    }
              } catch (PayPalRESTException ex) {

                    ex.printStackTrace();
              }
        return "redirect:/";
    }
    /*@GetMapping(value = FAIL_URL)
    public String cancelPay() {
        return "payFailed";
    }*/

    /*@GetMapping(value = SUCCESS_URL)
    public String successPay(@RequestParam("paymentId") String paymentId,
@RequestParam("PayerID") String payerId) throws MessagingException {
        try {
            Payment payment = service.executePayment(paymentId, payerId);
            String result = emailServ.sendEmail(payment);
            if (payment.getState().equals("approved")) {
             //return new ModelAndView(new RedirectView(result));
             return result;
            }
        } catch (PayPalRESTException ex) {
            System.out.println(ex.getMessage());
        }
        return "redirect:/";
    }*/

}
```

5. **emailControl.java**

```
package shop.online.controller;
import javax.mail.MessagingException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import com.paypal.api.payments.Payment;
import com.paypal.base.rest.PayPalRESTException;
import shop.online.service.emailService;
import shop.online.service.paypalService;
@Controller
public class emailControl {
```

```
    @Autowired
    private paypalService service;
    @Autowired
    private emailService emailServ;
    public static final String SUCCESS_URL = "pay/paySuccess";
    public static final String FAIL_URL = "pay/payFailed";
    @GetMapping(value = FAIL_URL)
    public String cancelPay() {
        return "payFailed";
    }
    @GetMapping(value = SUCCESS_URL)
    public String successPay(@RequestParam("paymentId") String paymentId,
@RequestParam("PayerID") String payerId) throws MessagingException {
        try {
            Payment payment = service.executePayment(paymentId, payerId);
            String result = emailServ.sendEmail(payment);
            if (payment.getState().equals("approved")) {
             //return new ModelAndView(new RedirectView(result));
             return result;
            }
        } catch (PayPalRESTException ex) {
            System.out.println(ex.getMessage());
        }
        return "redirect:/";
    }
}
```

Mobile Payment Service

1. **BillPayment.java**

```
package shop.online.model;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
@Entity
@Table(name ="via_mobile_bill")
public class BillPayment {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    @Column(name = "custName")
    private String custName;
    @Column(name = "custEmail")
    private String custEmail;
    @Column(name = "custPhone")
    private String custPhone;
    @Column(name = "sellerEmail")
    private String sellerEmail;
    @Column(name = "itemName")
    private String itemName;
    @Column(name = "itemPrice")
    private double itemPrice;
```

```
	public BillPayment() {
	}

	public BillPayment(String custName, String custEmail, String custPhone,
String sellerEmail, String itemName, double itemPrice) {
		this.custName = custName;
		this.custEmail = custEmail;
		this.custPhone = custPhone;
		this.sellerEmail = sellerEmail;
		this.itemName = itemName;
		this.itemPrice = itemPrice;
	}
	public long getId() {
		return id;
	}
	public void setId(long id) {
		this.id = id;
	}
	public String getCustName() {
		return custName;
	}
	public void setCustName(String custName) {
		this.custName = custName;
	}
	public String getCustEmail() {
		return custEmail;
	}
	public void setCustEmail(String custEmail) {
		this.custEmail = custEmail;
	}
	public String getCustPhone() {
		return custPhone;
	}
	public void setCustPhone(String custPhone) {
		this.custPhone = custPhone;
	}
	public String getSellerEmail() {
		return sellerEmail;
	}
	public void setSellerEmail(String sellerEmail) {
		this.sellerEmail = sellerEmail;
	}
	public String getItemName() {
		return itemName;
	}
	public void setItemName(String itemName) {
		this.itemName = itemName;
	}
	public double getItemPrice() {
		return itemPrice;
	}
	public void setItemPrice(double itemPrice) {
		this.itemPrice = itemPrice;
	}
}
```

2. **OTPSystem.java**

```
package shop.online.model;
public class OTPSystem {
	private String mobileNumber;
	private String otp;
	private long expiryTime;
	public OTPSystem() {

	}
	public String getMobileNumber() {
		return mobileNumber;
	}
	public void setMobileNumber(String mobileNumber) {
		this.mobileNumber = mobileNumber;
	}
	public String getOtp() {
		return otp;
	}
	public void setOtp(String otp) {
		this.otp = otp;
	}
	public long getExpiryTime() {
		return expiryTime;
	}
	public void setExpiryTime(long expiryTime) {
		this.expiryTime = expiryTime;
	}
}
```

3. BillPaymentRepo.java

```
package shop.online.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import shop.online.model.BillPayment;
@Repository
public interface BillPaymentRepo extends JpaRepository<BillPayment, Long>{
}
```

4. NotificationService.java

```
package shop.online.service;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mail.MailException;
import org.springframework.mail.SimpleMailMessage;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.stereotype.Service;
import shop.online.model.BillPayment;
@Service
public class NotificationService {
	private JavaMailSender javaMailSender;
	private static final String ORGANICATOIN_EMAIL = "dhananjayaarw@gmail.com";
	private static final String BUYER_SUBJECT = "Online Shopping buy items";
	private static final String SELLER_SUBJECT = "Online Shopping sell items";

	@Autowired
	public NotificationService(JavaMailSender javaMailSender) throws
MailException{
		this.javaMailSender = javaMailSender;
	}
	public void sendNotification(BillPayment billPayment) {
```

```
            final String  BUYER_CONTENT = "Item name :-
"+billPayment.getItemName()+"\n"
                        + "Price :- "+billPayment.getItemPrice()+"\n\n"
                                    + "Mobile number :-
"+billPayment.getCustPhone()+"\n"
                                                + "Thank you for using the
service";
            final String SELLER_CONTENT = "Item name :-
"+billPayment.getItemName()+"\n"
                        + "Thank you for using the service";
            //send email
            SimpleMailMessage customerMailMessage = new SimpleMailMessage();
            SimpleMailMessage sellerMailMessage = new SimpleMailMessage();

            customerMailMessage.setTo(billPayment.getCustEmail());
            customerMailMessage.setFrom(ORGANICATOIN_EMAIL);
            customerMailMessage.setSubject(BUYER_SUBJECT);
            customerMailMessage.setText(BUYER_CONTENT);

            sellerMailMessage.setTo(billPayment.getSellerEmail());
            sellerMailMessage.setFrom(ORGANICATOIN_EMAIL);
            sellerMailMessage.setSubject(SELLER_SUBJECT);
            sellerMailMessage.setText(SELLER_CONTENT);

            javaMailSender.send(customerMailMessage);
            javaMailSender.send(sellerMailMessage);
      }
}
```

5. **BillPaymentController.java**

```
package shop.online.controller;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mail.MailException;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import shop.online.model.BillPayment;
import shop.online.repository.BillPaymentRepo;
import shop.online.service.NotificationService;
@CrossOrigin(origins = "http://localhost:3000")
@RestController
@RequestMapping("/api/v1/")
public class BillPaymentController {
      @Autowired
      private BillPaymentRepo billPaymentRepo;
      @Autowired
      private NotificationService notificationService;
      //add payment
      @RequestMapping("/appPayment")
      @PostMapping("/appPayment")
      public void addPayment(@RequestBody BillPayment billPayment) {
            //send notificationService
            try {
                  notificationService.sendNotification(billPayment);
```

```
			} catch (MailException e) {
				e.printStackTrace();
			}
		}
}
```

6. OTPSytemRESTController.java

```
package shop.online.controller;
import java.util.HashMap;
import java.util.Map;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
import com.twilio.Twilio;
import com.twilio.rest.api.v2010.account.Message;
import com.twilio.type.PhoneNumber;
import shop.online.model.OTPSystem;
@CrossOrigin(originPatterns = "http://localhost:3000")
@RestController
public class OTPSystemRESTController {
	private final static String ACCOUNT_SID =
"AC0966b9e047313bee0c192d481601d1bd";
	private final static String AUTH_TOKEN =
"3d6e26ec859e005ee6260edb36c7826c";
	private Map<String, OTPSystem> otp_data = new HashMap<>();
	private final static String USER_PHONE_NUMBER = "+94767185502";
	private final static String PHONE_NUMBER = "+19705003288";
	static {
		Twilio.init(ACCOUNT_SID, AUTH_TOKEN);
	}
	@RequestMapping(value = "/mobilenumbers/{mobilenumber}/otp", method =
RequestMethod.POST)
	public ResponseEntity<Object> sentOTP(@PathVariable("mobilenumber") String
mobilenumber){
		OTPSystem otpSystem = new OTPSystem();
		otpSystem.setMobileNumber(mobilenumber);
		otpSystem.setOtp(String.valueOf(((int)(Math.random()*(10000-1000)))+
1000));
		otpSystem.setExpiryTime(System.currentTimeMillis()+20000);
		otp_data.put(mobilenumber, otpSystem);
		Message.creator(new PhoneNumber(USER_PHONE_NUMBER), new
PhoneNumber(PHONE_NUMBER), "Your OTP is :" + otpSystem.getOtp()).create();
		return new ResponseEntity<>("OTP is send
successfully",HttpStatus.OK);
	}
	@RequestMapping(value = "/mobilenumbers/{mobilenumber}/otps",method =
RequestMethod.PUT)
	public ResponseEntity<Object> verifyOTP(@PathVariable("mobilenumber")
String mobilenumber,@RequestBody OTPSystem requestBodyOTPSystem){
		if(requestBodyOTPSystem.getOtp() == null ||
requestBodyOTPSystem.getOtp().trim().length() <= 0) {
```

```
                    return new ResponseEntity<>("Please provide a
OTP",HttpStatus.OK);
              }
              if(otp_data.containsKey(mobilenumber)) {
                    OTPSystem otpSystem = otp_data.get(mobilenumber);
                    if(otpSystem != null) {
                          if(otpSystem.getExpiryTime() >=
System.currentTimeMillis()) {

       if(requestBodyOTPSystem.getOtp().equals(otpSystem.getOtp())) {
                                          otp_data.remove(mobilenumber);
                                          return new ResponseEntity<>("OTP
verification success !",HttpStatus.OK);
                                    }
                                    return new ResponseEntity<>("Invalid OTP
!",HttpStatus.OK);
                              }
                              return new ResponseEntity<>("OTP is
expired...!",HttpStatus.OK);
                    }
                    return new ResponseEntity<>("Something went wrond ...!!",
HttpStatus.OK);
              }
              return new ResponseEntity<>("Mobile number not found",
HttpStatus.OK);
       }
}
```

7. **ResourceNotFoundException.java**

```
package shop.online.errorHandler;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;
@ResponseStatus(value = HttpStatus.NOT_FOUND)
public class ResourceNotFoundException extends RuntimeException{

       private static final long serialVersionUID = 1L;
       public ResourceNotFoundException(String message) {
              super(message);
       }
}
```

Frontends

(All below codes are copied no screen shots given)

1. **App.js**

```
import React from 'react';
import '../src/css/styles.css';
import {BrowserRouter as Router,Switch,Route} from 'react-router-dom';
import SellerProfile from './components/SellerProfile';
import Registration from './components/Registration';
import AddItem from './components/AddItem';
import EditRegistration from './components/EditRegistration';
import LoginNew from './components/LoginNew';
import DeleteItems from './components/DeleteItems';
```

```
import UpdateItems from './components/UpdateItems';
import DisplayCategories from './components/DisplayCategories';
import DisplayItems from './components/DisplayItems';
import SingleItem from './components/SingleItem';
import DeliverPage from "./components/deliver/DeliverPage"
import Pay from './components/Pay';
import Model from "./components/Model";
class App extends React.Component {
  render(){
    return (
      <Router>
          <Switch>
          <Route path="/" exact component={DisplayCategories}/>
          <Route path="/towardsRegisterPage" exact component={Registration} />
          <Route path="/edit" exact component={UpdateItems} />
          <Route path="/afterEditItem" exact component={SellerProfile} />
          <Route path="/editSeller" exact component={EditRegistration} />
          <Route path="/editItem" exact component={UpdateItems} />
          <Route path="/deleteItem" exact component={DeleteItems} />
          <Route path="/addItem" exact component={SellerProfile} />
          <Route path="/afterdeleteItem" exact component={SellerProfile} />
          <Route path="/displayCategoriesItems" exact component={DisplayItems}
 />
          <Route path="/displaySinglePage" exact component={SingleItem} />
          <Route path="/addi" exact component={AddItem} />
          <Route path="/afterEditSeller" exact component={SellerProfile} />
          <Route path="/login" exact component={SellerProfile} />
          <Route path="/afterRegisterPage" exact component={LoginNew} />
          <Route path="/afterRegisterP" exact component={LoginNew} />
          <Route path="/deliveries" exact component={DeliverPage}/>
          <Route path="/creditcard" exact component={Pay} />
          <Route path="/pinConfirm" component = {Model} />
          </Switch>
      </Router>
    );
  }
}
export default App;
```

Components

1. **DisplayCategories.js**

```
import React, { Component } from 'react'
import axios from 'axios';
import HeaderNormal from './HeaderNormal'
import { Redirect } from "react-router";
export default class DisplayCategories extends Component{
```

```
    constructor(props){
        super(props)
        this.state = {
            category: []
        };
    }
    componentDidMount(){
        this.showCatgeories();
    }
    showCatgeories(){
        axios.get("http://localhost:8080/api/categories")
            .then(response => response.data)
            .then((data) => {
                this.setState({category: data});

            });
    }
    // this.state.category.map((category) =>
    state = {
    redirect: false
    }
    redirectHandler6 = () => {
        this.setState({ redirect: true })
        this.renderRedirect6();
    }
    renderRedirect6 = () => {
        if (this.state.redirect) {
            return <Redirect to='/displayCategoriesItems' />
        }
    }
    render() {
        return(
            <div>
                <HeaderNormal/>
                <div className="App">
                    <h1>Categories</h1>
                    {
                        this.state.category.map((category) =>(
                            <div className="containerMain" key={this.categoryI
d}>
                                <div className="cat">
                                        <a onClick={this.redirectHandler6}>
                                            <div className="card cardcat">
                                                <img src={`data:itemImage/jpg;
base64,${category.catImage}`}/>
                                                    <label id="catName"><cente
r><b>{category.categoryName}</b></center></label>
                                            </div>
```

```
                                        </a>{this.renderRedirect6()}
                                    </div><br/>
                                </div>
                            ))
                        }
                    </div>
                </div>
            )
        }
}
```

2. **DisplayItems.js**

```
import React, { Component } from 'react'
import axios from 'axios';
import HeaderNormal from './HeaderNormal'
import { Redirect } from "react-router";
export default class DisplayItems extends Component {
    constructor(props){
        super(props)
        this.state = {
            items: []
        };
    }
    componentDidMount(){
        this.showItemCategories();
    }
    showItemCategories = (catName) =>{
        axios.get("http://localhost:8080/api/buyer/Furniture",catName)
            .then(response => response.data)
            .then((data) => {
                this.setState({items: data});

            });
    }
    state = {
    redirect: false
    }
    redirectHandler7 = () => {
        this.setState({ redirect: true })
        this.renderRedirect7();
    }
    renderRedirect7 = () => {
        if (this.state.redirect) {
            return <Redirect to='/displaySinglePage' />
```

```
        }
    }
    render() {
        return(
            <div>
                <HeaderNormal/>
                <div className="contApp1">
                    <h1>Items</h1>
                        {
                            this.state.items.map((items) => (
                                    <div class="card cardcat1">
                                        <a onClick={this.redirectHandler7}>
                                            <img src={`data:itemImage/jpg;base
64,${items.itemImage}`} />

                                            <h4>{items.itemName}</h4>
                                            <p class="price">${items.itemPrice
}</p>

                                        </a>{this.renderRedirect7()}
                                    </div>
                                ))
                        }
                </div>
            </div>
        )
    }
}
```

3. **Registration.js**

```
import React, { Component } from 'react'
import axios from 'axios';
import HeaderNormal from './HeaderNormal'
import { Redirect } from "react-router";

export default class Registration extends Component{
    constructor(props) {
        super(props);
        this.state = this.initialState;
        this.sellerChange = this.sellerChange.bind(this);
        this.submitSeller = this.submitSeller.bind(this);
    }

    initialState = {
        sellerName:'', sellerAddress:'', sellerPhone:'',sellerEmail:'',sellerZ
ip:'',sellerAccNo:'',sellerPassword:''
    }

    submitSeller = event => {
```

```
        event.preventDefault();

        const seller = {
            sellerName: this.state.sellerName,
            sellerAddress: this.state.sellerAddress,
            sellerPhone: this.state.sellerPhone,
            sellerEmail: this.state.sellerEmail,
            sellerZip: this.state.sellerZip,
            sellerAccNo: this.state.sellerAccNo,
            sellerPassword: this.state.sellerPassword
        };

        axios.post("http://localhost:8080/api/seller", seller)
            .then(response => {
                if(response.data != null) {
                    this.setState(this.initialState);
                    alert("Seller Saved Successfully");
                    window.location.href = "/afterRegisterPage";
                }
            })
    }

    sellerChange = event => {
        this.setState({
            [event.target.name]:event.target.value
        });
    }

    state = {
        redirect: false
    }
    redirectHandler10 = () => {
        this.setState({ redirect: true })
        this.renderRedirect10();
    }
    renderRedirect10 = () => {
        if (this.state.redirect) {
            return <Redirect to='/afterRegisterPage' />
        }
    }

    render() {

        const{sellerName,sellerAddress,sellerPhone,sellerEmail,sellerZip,selle
rAccNo,sellerPassword} = this.state;

        return (
```

```
            <div>
                <HeaderNormal/>
                <div className="signup-form">
                    <form className="form-
horizontal" onSubmit={this.submitSeller}>
                        <div className="row">
                          <div className="col-8 offset-4">
                              <h2>Sign Up</h2>
                          </div>
                        </div>
                       <div className="form-group row">
                          <label className="col-form-label col-
4" htmlFor="name">Name</label>
                                <div className="col-8">
                                    <input type="text" className="form-
control" name="sellerName" required="required" value={sellerName} onChange={th
is.sellerChange} />
                                </div>
                       </div>
                       <div className="form-group row">
                          <label className="col-form-label col-
4" htmlFor="address">Address</label>
                                <div className="col-8">
                                    <input type="text" className="form-
control" name="sellerAddress" required="required" value={sellerAddress} onChan
ge={this.sellerChange}/>
                                </div>
                       </div>
                       <div className="form-group row">
                          <label className="col-form-label col-
4" htmlFor="number">Tel-Number</label>
                                <div className="col-8">
                                    <input type="number" className="form-
control" name="sellerPhone" required="required" value={sellerPhone} onChange={
this.sellerChange}/>
                                </div>
                       </div>
                       <div className="form-group row">
                          <label className="col-form-label col-
4" htmlFor="email">Email</label>
                                <div className="col-8">
                                    <input type="email" className="form-
control" name="sellerEmail" required="required" value={sellerEmail} onChange={
this.sellerChange}/>
                                </div>
                       </div>
                       <div className="form-group row">
```

```
                            <label className="col-form-label col-
4" htmlFor="zip">Zip-Number</label>
                                <div className="col-8">
                                    <input type="number" className="form-
control" name="sellerZip" required="required" value={sellerZip} onChange={this
.sellerChange}/>
                                </div>
                            </div>
                            <div className="form-group row">
                                <label className="col-form-label col-
4" htmlFor="accNo">Account-No</label>
                                <div className="col-8">
                                    <input type="number" className="form-
control" name="sellerAccNo" required="required" value={sellerAccNo} onChange={
this.sellerChange}/>
                                </div>
                            </div>
                            <div className="form-group row">
                                <label className="col-form-label col-
4" htmlFor="password">Password</label>
                                <div className="col-8">
                                    <input type="password" className="form-
control" name="sellerPassword" required="required" value={sellerPassword} onCh
ange={this.sellerChange}/>
                                </div>
                            </div>

                            <div className="form-group row">
                                <div className="col-8 offset-5">
                                    <p><label className="form-check-label">
                                            <input type="checkbox" required="requi
red"/> I accept the Terms and policies.
                                        </label>
                                    </p>
                                </div>
                                    <button type="submit"  className="btn btn-
primary btn-lg">Sign Up</button>
                                    {/* onClick={this.redirectHandler10}
                                    {this.renderRedirect10()} */}
                            </div>
                        </form>

                    </div>
                    <div className="text-
center">Already have an account? <a href="#">Login here</a></div>
                </div>
            )
        }
```

```
}
```

4. **EditRegistration.js**

```
import React, { Component } from 'react'
import axios from 'axios';
import HeaderLogged from './HeaderLogged'
import { Redirect } from "react-router";

export default class EditRegistration extends Component{
    constructor(props) {
        super(props);
        this.state = this.initialState;
        this.sellerUpdateChange = this.sellerUpdateChange.bind(this);
        this.updateSeller = this.updateSeller.bind(this);
        this.state = {
            profile: []
        };
    }

    initialState = {
        sellerName:'', sellerAddress:'', sellerPhone:'',sellerEmail:'',sellerZ
ip:'',sellerAccNo:'',sellerPassword:''
    }

    componentDidMount(){
        this.sellerDisplay();

    }

    sellerDisplay = (sellerId) => {
        axios.get("http://localhost:8080/api/seller/1", sellerId)
            .then(response => response.data)
            .then((data) => {
                this.setState({profile: data});

            });
    }


    updateSeller = event => {
       event.preventDefault();

        const seller = {
            sellerId: this.state.sellerId,
            sellerName: this.state.sellerName,
            sellerAddress: this.state.sellerAddress,
            sellerPhone: this.state.sellerPhone,
```

```
            sellerEmail: this.state.sellerEmail,
            sellerZip: this.state.sellerZip,
            sellerAccNo: this.state.sellerAccNo,
            sellerPassword: this.state.sellerPassword
        };

        axios.put("http://localhost:8080/api/seller/1", seller)
            .then(response => {
                if(response.data != null) {
                    this.setState(this.initialState);
                    alert("Seller Updated Successfully");
                    window.location.href = "/afterEditSeller";
                }
            })
    }

    sellerUpdateChange = event => {
        this.setState({
            [event.target.name]:event.target.value1
        });
    }

    state = {
        redirect: false
    }
    redirectHandler8 = () => {
        this.setState({ redirect: true })
        this.renderRedirect8();
    }
    renderRedirect8 = () => {
        if (this.state.redirect) {
            return <Redirect to='/afterEditSeller' />
        }
    }


    render() {

        const{sellerId,sellerName,sellerAddress,sellerPhone,sellerEmail,seller
Zip,sellerAccNo,sellerPassword} = this.state;

        return (
            <div>
                <HeaderLogged/>
                <div className="signup-form">
                    <form className="form-
horizontal" onSubmit={this.updateSeller}>
                            <div className="row">
                              <div className="col-8 offset-4">
```

```
                                <h2>Edit Details</h2>
                            </div>
                        </div>
                        {/* value={this.state.profile.sellerName} */}
                    {/* <div className="form-group row">
                        <label className="col-form-label col-
4" htmlFor="name">ID</label>
                                <div className="col-8">
                                    <input type="text" className="form-
control" name="sellerId" value={this.state.profile.sellerId} value1={sellerId}
 onChange={this.sellerUpdateChange}/>
                                </div>
                    </div> */}
                    <div className="form-group row">
                        <label className="col-form-label col-
4" htmlFor="name">Name</label>
                                <div className="col-8">
                                    <input type="text" className="form-
control" name="sellerName" value={this.state.profile.sellerName} value1={selle
rName}  onChange={this.sellerUpdateChange}/>
                                </div>
                    </div>
                    <div className="form-group row">
                        <label className="col-form-label col-
4" htmlFor="address">Address</label>
                                <div className="col-8">
                                    <input type="text" className="form-
control" name="sellerAddress" value={this.state.profile.sellerAddress} value1=
{sellerAddress} onChange={this.sellerUpdateChange} required="required" />
                                </div>
                    </div>
                    <div className="form-group row">
                        <label className="col-form-label col-
4" htmlFor="number">Tel-Number</label>
                                <div className="col-8">
                                    <input type="number" className="form-
control" name="sellerPhone" value={this.state.profile.sellerPhone} value1={sel
lerPhone} onChange={this.sellerUpdateChange} required="required" />
                                </div>
                    </div>
                    <div className="form-group row">
                        <label className="col-form-label col-
4" htmlFor="email">Email</label>
                                <div className="col-8">
                                    <input type="email" className="form-
control" name="sellerEmail" value={this.state.profile.sellerEmail} value1={sel
lerEmail} onChange={this.sellerUpdateChange} required="required" />
                                </div>
```

```
                        </div>
                        <div className="form-group row">
                            <label className="col-form-label col-
4" htmlFor="zip">Zip-Number</label>
                                <div className="col-8">
                                    <input type="number" className="form-
control" name="sellerZip" value={this.state.profile.sellerZip} value1={sellerZ
ip}  onChange={this.sellerUpdateChange} required="required" />
                                </div>
                        </div>
                        <div className="form-group row">
                            <label className="col-form-label col-
4" htmlFor="accNo">Account-No</label>
                                <div className="col-8">
                                    <input type="number" className="form-
control" name="sellerAccNo" value={this.state.profile.sellerAccNo} value1={sel
lerAccNo}  onChange={this.sellerUpdateChange} required="required" />
                                </div>
                        </div>
                        <div className="form-group row">
                            <label className="col-form-label col-
4" htmlFor="password">Password</label>
                                <div className="col-8">
                                    <input type="password" className="form-
control" name="sellerPassword" value={this.state.profile.sellerPassword} value
1={sellerPassword}  onChange={this.sellerUpdateChange} required="required" />
                                </div>
                        </div>

                        <div className="form-group row">
                            <div className="col-8 offset-5">

                            </div>
                                    <button type="submit" onClick={this.redire
ctHandler8}  className="btn btn-primary btn-
lg">Update</button>{this.renderRedirect8()}
                        </div>
                    </form>

                </div>

            </div>
        )
    }
}
```

5. **LoginNew.js**

```
import React, { Component } from 'react'
import abcshooping from '../abcshooping.jpg';
import axios from 'axios';
import HeaderNormal from './HeaderNormal'
import { Redirect } from "react-router";

export default class LoginNew extends Component {

    constructor(props) {
        super(props);
        this.state = this.initialState;
        this.login = this.login.bind(this);
        this.credentialChange = this.credentialChange.bind(this);
    }

    initialState = {
        sellerEmail:'',sellerPassword:''
    }

    login = event => {

        event.preventDefault();

        const seller = {
            sellerEmail: this.state.sellerEmail,
            sellerPassword: this.state.sellerPassword
        };

        axios.post("http://localhost:8080/api/seller/login", seller)
            .then(response => {
                if(response.data != null) {
                    this.setState(this.initialState);
                    alert("Login Successfull");
                    window.location.href = "/login";
                }
                else {
                    alert("Login Faliure")
                }

            })
    }

    credentialChange = event => {
        this.setState({
            [event.target.name] : event.target.value
        });
    };
```

```
    state = {
        redirect: false
    }
    redirectHandler = () => {
        this.setState({ redirect: true })
        this.renderRedirect();
    }
    renderRedirect = () => {
        if (this.state.redirect) {
            return <Redirect to='/towardsRegisterPage' />
        }
    }

    redirectHandler9 = () => {
        this.setState({ redirect: true })
        this.renderRedirect9();
    }
    renderRedirect9 = () => {
        if (this.state.redirect) {
            return <Redirect to='/login' />
        }
    }

    render() {
        const{sellerEmail,sellerPassword} = this.state;
        return (
            <div>
               <HeaderNormal/>
                <div className="container1">
                    <div className="row justify-content-center">
                        <div className="col-md-8">
                            <div className="card-group mb-0">
                                <div className="card p-4">
                                    <div className="card-body">
                                    <form onSubmit={this.login}>
                                        <h1>Login</h1>
                                        <p className="text-
muted">Sign In to your account</p>
                                            <div className="input-group mb-3">
                                                <span className="input-group-
addon"><i className="fa fa-user"></i></span>
                                                    <input type="email" classN
ame="form-
control" placeholder="sellerEmail" name="sellerEmail" value={sellerEmail} onCh
ange={this.credentialChange} required/>
                                            </div>
                                            <div className="input-group mb-4">
```

```
                                                <span className="input-group-
addon"><i className="fa fa-lock"></i></span>

                                                    <input type="password" cla
ssName="form-
control" placeholder="sellerPassword" name="sellerPassword" onChange={this.cre
dentialChange} required/>
                                                </div>
                                                <div className="row">
                                                    <div className="col-6">
                                                        <button type="submit"  cla
ssName="btn btn-success px-4" id="btnLogin">Login</button>
                                                        {/* onClick={this.redirect
Handler9}{this.renderRedirect9()} */}
                                                    </div>
                                                </div>
                                            </form>
                                        </div>
                                    </div>
                                    <div className="card text-white shopping py-
5" style={{backgroundImage:'url('+abcshooping+')'}}>
                                        <div className="card-body text-center">
                                            <div className="shoppingpic">
                                                <h2>Sign up</h2>
                                                <p>Lorem ipsum dolor sit amet, con
sectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolor
e magna aliqua.</p>
                                                <button type="submit" className="b
tn-
success" id="btnlRegister" onClick={this.redirectHandler}>Register Now!</butto
n>{this.renderRedirect()}
                                            </div>
                                        </div>
                                    </div>
                                </div>
                            </div>
                        </div>

                    </div>

                </div>
        )
    }
}
```

6. **SellerProfile.js**

```
import React, { Component } from 'react'
```

```
import axios from 'axios';
import HeaderLogged from './HeaderLogged'
import { Redirect } from "react-router";
export default class SellerProfile extends Component{
    constructor(props){
        super(props)
        this.state = {
            items: [],profile:[]
        };
        this.myFunction=this.myFunction.bind(this);

    }

    componentDidMount(){
        this.showSellerItems();
        this.sellerprofile();
    }

    showSellerItems = (sellerNum) => {
        axios.get("http://localhost:8080/api/items/0781231231", sellerNum)
            .then(response => response.data)
            .then((data) => {
                this.setState({items: data});
            });
    }

    sellerprofile = (sellerId) => {
        axios.get("http://localhost:8080/api/seller/1", sellerId)
            .then(response => response.data)
            .then((data) => {
                this.setState({profile: data});

            });
    }

    myFunction() {
        var input, filter, table, tr, td, i, txtValue;
      input = document.getElementById("searchBar");
      filter = input.value.toUpperCase();
      table = document.getElementById("myTable");
      tr = table.getElementsByTagName("tr");
      for (i = 0; i < tr.length; i++) {
        td = tr[i].getElementsByTagName("td")[0];
        if (td) {
          txtValue = td.textContent || td.innerText;
          if (txtValue.toUpperCase().indexOf(filter) > -1) {
            tr[i].style.display = "";
          } else {
```

```
            tr[i].style.display = "none";
          }
        }
      }
    }

    state = {
        redirect: false,
        redirect1: false,
        redirect2: false
    }
    redirectHandler1 = () => {
        this.setState({ redirect: true })
        this.renderRedirect1();
    }
    renderRedirect1 = () => {
        if (this.state.redirect) {
            return <Redirect to='/editSeller' />
        }
    }

    redirectHandler2 = () => {
        this.setState({ redirect1: true })
        this.renderRedirect2();
    }
    renderRedirect2 = () => {
        if (this.state.redirect1) {
            return <Redirect to='/editItem' />
        }
    }

    redirectHandler3 = () => {
        this.setState({ redirect2: true })
        this.renderRedirect3();
    }
    renderRedirect3 = () => {
        if (this.state.redirect2) {
            return <Redirect to='/deleteItem' />
        }
    }
    render() {
        return (

            <div>
                <HeaderLogged/>
                 <div className="container4">
                    <div className="row ">
                        <div className="col-sm-4">
```

```
                            <div className="card1">
                                <div className="card-body">
                                    <div className="d-flex flex-column align-
items-center text-center">
                                        <img src="https://bootdey.com/img/Cont
ent/avatar/avatar7.png" alt="Admin" className="rounded-circle" width="100"/>
                                            <div className="mt-3">
                                                <h4>{this.state.profile.seller
Name}</h4>
                                                    <button className="btn btn
-
primary" name="submit" id="btnEditS" type="submit" onClick={this.redirectHandl
er1}>Edit</button>{this.renderRedirect1()}
                                            </div>
                                        </div>
                                        <ul className="list-group list-group-
flush">
                                            <li className="list-group-item d-
flex justify-content-between align-items-center flex-wrap">
                                                <label>Phone-
Number</label><label className="text-
secondary">{this.state.profile.sellerPhone}</label>
                                            </li>
                                            <li className="list-group-item d-
flex justify-content-between align-items-center flex-wrap">
                                                <label>Email</label><label cla
ssName="text-secondary">{this.state.profile.sellerEmail}</label>
                                            </li>
                                            <li className="list-group-item d-
flex justify-content-between align-items-center flex-wrap">
                                                <label>Account-
No</label><label className="text-
secondary">{this.state.profile.sellerAccNo}</label>
                                            </li>
                                            <li className="list-group-item d-
flex justify-content-between align-items-center flex-wrap">
                                                <label>Zip-
Code</label><label className="text-
secondary">{this.state.profile.sellerZip}</label>
                                            </li>
                                            <li className="list-group-item d-
flex justify-content-between align-items-center flex-wrap">
                                                <label>Address</label><label c
lassName="text-secondary">{this.state.profile.sellerAddress}</label>
                                            </li>
                                        </ul>
                                </div>
                            </div>
```

```
                    </div>

                        <div className="col-sm-8">
                            <form className="form-inline row">
                                <input className="form-
control col" id="searchBar" onKeyUp={this.myFunction} type="search" placeholde
r="Search" aria-label="Search"/>
                                <button className="btn btn-outline-
success col-sm-2" type="submit" id="seacrchbtn">Search</button>
                            </form>
                                <table id="myTable">
                                    <thead>
                                        <tr>
                                            <th>Item Name</th>
                                            <th>Item Price</th>
                                            <th>Item Quantity</th>
                                            <th>Item Description</th>
                                            <th>Category Name</th>
                                            <th>Item Image</th>
                                            <th>Edit</th>
                                            <th>Delete</th>
                                        </tr>
                                    </thead>

                                    <tbody>

                                        {
                                            this.state.items.map((items) =
> (

                                                <tr>
                                                    <td>{items.itemName}</
td>

                                                    <td>{items.itemPrice}<
/td>

                                                    <td>{items.itemQuantit
y}</td>

                                                    <td>{items.itemDescrip
tion}</td>

                                                    <td>{items.catName}</t
d>

                                                    <td><img src={`data:it
emImage/jpg;base64,${items.itemImage}`} id="sImage" /></td>

                                                    <td>
                                                        <a onClick={this.r
edirectHandler2}  className="btn btn-
dark" id="btnEditS1">Edit</a> {this.renderRedirect2()}
                                                    </td>
```

```
                                                    <td>
                                                        <a  onClick={this.
redirectHandler3} className="btn btn-
dark" id="btnDelS1">Delete</a>  {this.renderRedirect3()}
                                                    </td>

                                                </tr>
                                            ))
                                        }

                                    </tbody>

                                </table>
                            </div>
                        </div>
                    </div>
                )
                </div>
            )
    }
}
```

7. **AddItem.js**

```
import React, { Component } from 'react';
import axios from 'axios';
import HeaderLogged from './HeaderLogged'
import { Redirect } from "react-router";

export default class AddItem extends Component{
    constructor(props) {
        super(props);
        this.state = this.initialState;
        this.itemChange = this.itemChange.bind(this);
        this.submitItem = this.submitItem.bind(this);
    }

    initialState = {
        itemName:'', catName:'', itemQuantity:'',itemPrice:'',itemDescription:
'',sellerNum:'',file:'',sellerEmail:''
    }

    submitItem = event => {

        event.preventDefault();

        const item = {
```

```
            itemName: this.state.itemName,
            catName: this.state.catName,
            itemQuantity: this.state.itemQuantity,
            itemPrice: this.state.itemPrice,
            itemDescription: this.state.itemDescription,
            sellerNum: this.state.sellerNum,
            itemImage: this.state.file,
            sellerEmail: this.state.sellerEmail
        };

        axios.post("http://localhost:8080/api/", item)
            .then(response => {
                if(response.data != null) {
                    this.setState(this.initialState);
                    alert("Item Saved Successfully");
                    window.location.href = "/addItem";

                }
            })
    }

    itemChange = event => {
        this.setState({
            [event.target.name]:event.target.value
        });
    }

    state = {
        redirect: false
    }
    redirectHandler4 = () => {
        this.setState({ redirect: true })
        this.renderRedirect4();
    }
    renderRedirect4 = () => {
        if (this.state.redirect) {
            return <Redirect to='/addItem' />
        }
    }

    render() {

        const{itemName,catName,itemQuantity,itemPrice,itemDescription,sellerNu
m,itemImage,sellerEmail} = this.state;

        return(
            <div>
                <HeaderLogged/>
```

```
            <div className="container2">
                <form id="contact" action="/sellerprofile" method="POST" onSub
mit={this.submitItem} encType="multipart/form-data">
                    <h3>Add Your Items Here</h3>
                    <h4>Contact us for custom quote</h4>
                        <fieldset>
                            <input placeholder="Item Name" type="text" tabinde
x="1" name="itemName" value={itemName} onChange={this.itemChange} required />
                        </fieldset>
                        <fieldset>
                            <select placeholder="Select Category" type="text"
name="catName" value={catName} onChange={this.itemChange} tabindex="2" require
d >
                                <option>Select Category</option>
                                <option>Furniture</option>
                                <option>Music Instruments</option>
                                <option>Fitness Equipments</option>
                            </select>
                        </fieldset>
                        <fieldset>
                            <input placeholder="Item Price" type="text" tabind
ex="4" name="itemPrice" value={itemPrice} onChange={this.itemChange} required/
>
                        </fieldset>
                        <fieldset>
                            <textarea placeholder="Item Description" tabindex=
"5" name="itemDescription" value={itemDescription} onChange={this.itemChange}
required></textarea>
                        </fieldset>
                        <fieldset>
                            <input placeholder="Seller Number" type="number" t
abindex="6" name="sellerNum" value={sellerNum} onChange={this.itemChange} requ
ired/>
                        </fieldset>

                        <fieldset>
                            <input placeholder="Seller Email" type="email" tab
index="7" name="sellerEmail" value={sellerEmail} onChange={this.itemChange} re
quired/>
                        </fieldset>

                        <fieldset>
                            <input placeholder="Select The Picture" type="file
" tabindex="8" name="itemImage" value={itemImage} onChange={this.itemChange} r
equired />
                        </fieldset>
                        <fieldset>
```

```
                        <button type="submit"  id="contact-
submit">Submit</button>
                    </fieldset>
                </form>
            </div>
        </div>
        )
    }
}
```

8. **UpdateItems.js**

```
import React, { Component } from 'react'
import axios from 'axios';
import { Redirect } from "react-router";
import HeaderLogged from './HeaderLogged'
export default class  UpdateItems extends Component {
    constructor(props) {
        super(props);
        this.state = this.initialState;
        this.updateItemChange = this.updateItemChange.bind(this);
        this.updateItem = this.updateItem.bind(this);
        this.state = {
            item: []
        };
    }

    initialState = {
        itemName:'', itemQuantity:'',itemPrice:'',itemDescription:''
    }

    componentDidMount(){
        this.itemDisplay();

    }

    itemDisplay = (itemId) => {
        axios.get("http://localhost:8080/api/buyerView/8", itemId)
            .then(response => response.data)
            .then((data) => {
                this.setState({item: data});

            });
    }

    updateItem = event => {
        event.preventDefault();
```

```
        const item = {
            itemId: this.state.itemId,
            itemName: this.state.itemName,
            itemQuantity: this.state.itemQuantity,
            itemPrice: this.state.itemPrice,
            itemDescription: this.state.itemDescription

        };

        axios.put("http://localhost:8080/api/items/8", item)
            .then(response => {
                if(response.data != null) {
                    this.setState(this.initialState);
                    alert("Item Updated Successfully");
                    window.location.href = "/afterEditItem";
                }
            })
     }

    updateItemChange = event => {
        this.setState({
            [event.target.name]:event.target.value
        });
    }

    state = {
        redirect: false
    }
    redirectHandler12 = () => {
        this.setState({ redirect: true })
        this.renderRedirect12();
    }
    renderRedirect12 = () => {
        if (this.state.redirect) {
            return <Redirect to='/afterEditItem' />
        }
    }

    render() {

        const{itemId,itemName,itemQuantity,itemPrice,itemDescription} = this.s
tate;
       // const history = useHistory();
       // const navigateTo = () => history.push('/editSubmit');

        return (
            <div>
                <HeaderLogged/>
```

```
            <div class="container2">
                <form id="contact">
                    <h3>Update Your Items Here</h3>
                        <fieldset>
                            <input placeholder="Item ID" name="itemId" value={
this.state.item.itemId} type="hidden" tabindex="1" required autofocus/>
                        </fieldset>
                        <fieldset>
                            <input placeholder="Item Name" name="itemName" val
ue={this.state.item.itemName} type="text" tabindex="1" required autofocus/>
                        </fieldset>
                        <fieldset>
                            <input placeholder="Item Price" name="itemPrice" v
alue={this.state.item.itemPrice} type="text" tabindex="4" required/>
                        </fieldset>
                        <fieldset>
                            <textarea placeholder="Item Description" name="ite
mDescription" value={this.state.item.itemDescription} tabindex="5" required></
textarea>
                        </fieldset>
                        <fieldset>
                            <button name="submit" onClick={this.redirectHandle
r12} type="submit" id="contact-
submit">Submit</button>{this.renderRedirect12()}
                        </fieldset>
                </form>
            </div>
        </div>
        )
    }
}
```

9. **DeleteItems.js**

```
import React, { Component } from 'react'
import axios from 'axios';
import { Redirect } from "react-router";
import HeaderLogged from './HeaderLogged'
export default class DeleteItems extends Component {
    constructor(props) {
        super(props);
        this.state = {
            profile: []
        };
        this.deleteItems = this.deleteItems.bind(this);
        this.displayDItems = this.displayDItems.bind(this);
    }
```

```
    displayDItems = (itemId) => {
        axios.get("http://localhost:8080/api/buyerView/9", itemId)
            .then(response => response.data)
            .then((data) => {
                this.setState({profile: data});

            });

    }

    componentDidMount(){
        this.displayDItems();
    }

    deleteItems = (itemId) => {
        axios.delete("http://localhost:8080/api/items/9", itemId)

        alert("Item Deleted Successfully");
        window.location.href = "/afterdeleteItem";
    }

    state = {
        redirect: false
    }
    redirectHandler5 = () => {
        this.setState({ redirect: true })
        this.renderRedirect5();
    }
    renderRedirect5 = () => {
        if (this.state.redirect) {
            return <Redirect to='/afterdeleteItem' />
        }
    }

    render () {
        return (
            <div>
                <HeaderLogged/>
           <div class="container2k">
                <form id="contact" onSubmit={this.deleteItems}>
                    <h3>Delete Your Items Here</h3>
                        <fieldset>
                            <input placeholder="Item ID" type="number" name="i
temId" value={this.state.profile.itemId}  tabindex="1" required autofocus/>
                        </fieldset>
                        <fieldset>
                            <input placeholder="Item Name" type="text" name="i
temName" value={this.state.profile.itemName} tabindex="2" required autofocus/>
```

```
                        </fieldset>
                        <fieldset>
                            <button name="submit" type="submit" id="contact-
submit" onClick={this.redirectHandler5}>Submit</button>{this.renderRedirect5()
}
                        </fieldset>
                </form>
            </div>
        </div>
        )
    }
}
```

10. HeaderNormal.jsx

```
import React from 'react'
export default class HeaderNormal extends React.Component{
    render(){
        return (
            <div>
                <header id="header">
                    <nav>
                        <div className="container">
                            <div className="text-center">
                            <h1>Shop Online</h1>
                                <div>
                                    <button className="btn" ><a href="/towards
RegisterPage">Register</a></button>
                                    <button className="btn" ><a href="/login">
Login</a></button>
                                </div>
                            </div>
                        </div>

                    </nav>
                </header>
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.min.js
" integrity="sha512-
894YE6QWD5I59HgZOGReFYm4dnWc1Qt5NtvYSaNcOP+u1T9qYdvdihz0PPSiiqn/+/3e7Jo4EaG7Tu
bfWGUrMQ==" crossOrigin="anonymous"></script>
<script src="/js/index.js"></script>
</div>
        );
    }
}
```

11. **HeaderLogged.jsx**

```
import React from 'react'
export default class headerLogged extends React.Component{
    render(){
        return (
            <div>
                <header id="header">
                    <nav>
                        <div className="container" >
                            <div className="text-center" >
                                <h1>Shop Online</h1>
                                <div>
                                <button className="btn" ><a href="/">Logout</a
></button>
                                <button className="btn" ><a href="/addi">Add I
tem</a></button>
                                </div>
                            </div>
                        </div>
                    </nav>
                </header>
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.min.js
" integrity="sha512-
894YE6QWD5I59HgZOGReFYm4dnWc1Qt5NtvYSaNcOP+u1T9qYdvdihz0PPSiiqn/+/3e7Jo4EaG7Tu
bfWGUrMQ==" crossOrigin="anonymous"></script>
<script src="/js/index.js"></script>
</div>
        );
    }
}
```

12. **DeliverPage.jsx**

```
import React from 'react';
import 'react-router-dom';
import services from '../../services/service'
import deliverData from '../../services/deliverService'
import HeaderNormal from '../HeaderNormal';

export default class DeliverPage extends React.Component{
    constructor(props){
        super(props);
        this.onChangeBuyerName = this.onChangeBuyerName.bind(this);
        this.onChangeBuyerEmail = this.onChangeBuyerEmail.bind(this);
        this.onChangeBuyerAddress = this.onChangeBuyerAddress.bind(this);
        this.onChangeBuyerPhone = this.onChangeBuyerPhone.bind(this);
        this.onChangeDeliveryOption = this.onChangeDeliveryOption.bind(this);
        this.onChangePayOption = this.onChangePayOption.bind(this);
```

```
        this.saveDelivery = this.saveDelivery.bind(this);
        this.newDelivery = this.newDelivery.bind(this);

        this.state = {
            deliverid:null,
            buyerName:"",
            buyerEmail:"",
            buyerAddress:"",
            buyerPhone:"",
            deliverItemName:this.deliverItemName,
            deliveryCash:this.deliveryCash,
            totalPayment:this.totalPayment,
            deliveryOption:"",
            payOption:"",

            submitted: false
        };
    }
onChangeBuyerName(e){
    this.setState({
        buyerName: e.target.value
    });
}
onChangeBuyerEmail(e){
    this.setState({
        buyerEmail: e.target.value
    });
}
onChangeBuyerAddress(e){
    this.setState({
        buyerAddress: e.target.value
    });
}
onChangeBuyerPhone(e){
    this.setState({
        buyerPhone: e.target.value
    });
}
onChangeDeliveryOption(e){
    this.setState({
        deliveryOption: e.target.value
    });
}
onChangePayOption(e){
    this.setState({
        payOption: e.target.value
    });
}
```

```
deliveryCash = (localStorage.getItem('itemPrice') * (10/100));
totalPayment = (localStorage.getItem('itemPrice') * (110/100)).toFixed(2);
deliverItemName = localStorage.getItem('itemName');
    saveDelivery(){
        var data = {
            buyerName:this.state.buyerName,
            buyerEmail:this.state.buyerEmail,
            buyerAddress:this.state.buyerAddress,
            buyerPhone:this.state.buyerPhone,
            deliverItemName:this.state.deliverItemName,
            deliveryCash:this.state.deliveryCash,
            totalPayment:this.state.totalPayment,
            deliveryOption:this.state.deliveryOption,
            payOption:this.state.payOption
    };
    deliverData.create(data)
     .then(response =>{
         this.setState({
             deliverid:response.data.deliverid,
             buyerName:response.data.buyerName,
             buyerEmail:response.data.buyerEmail,
             buyerAddress:response.data.buyerAddress,
             buyerPhone:response.data.buyerPhone,
             deliverItemName:response.data.deliverItemName,
             deliveryCash:response.data.deliveryCash,
             totalPayment:response.data.totalPayment,
             deliveryOption:response.data.deliveryOption,
             payOption:response.data.payOption,

             submitted:true
         });
     })
     .catch(e => {
         console.log(e);
     })
     if(this.state.payOption === "CreditCard"){
        this.props.history.push("/creditcard");
     }
     else if(this.state.payOption === "MobilePayment"){
        this.props.history.push("/pinConfirm");
        services.sendPin().then(res =>{
        });
     }
    }
    newDelivery(){
        this.setState({
            deliverid:null,
            buyerName:"",
```

```
            buyerEmail:"",
            buyerAddress:"",
            buyerPhone:"",
            deliverItemName:this.deliverItemName,
            deliveryCash:this.deliveryCash,
            totalPayment:this.totalPayment,
            deliveryOption:"",
            payOption:"",

            submitted:false
        });
    }
    render(){
        localStorage.setItem('buyerName',this.state.buyerName);
        localStorage.setItem('buyerEmail',this.state.buyerEmail);
        localStorage.setItem('buyerPhone',this.state.buyerPhone);
        localStorage.setItem('totalPayment',this.state.totalPayment);
        return(
            <div>
                <HeaderNormal/>
                <main id="site-main">
                    <div className="container">
                        <div className="form-title text-center">
                        <h2 className="text-
dark">Select a delivery option</h2>
                        <span className="text-
dark">Use below form to choose a delivery option</span>
                        </div>
                                <form id="add_user" method="POST">

                                        <div>
                                        <div className="form-group">
                                        <label htmlFor="name" className="text-
dark">Name :</label>
                                        <input type="hidden" name="deliverid"
></input>
                                        <input type="text" name="buyerName" cl
assName="form-
control" onChange={this.onChangeBuyerName} placeholder="Donnald Johonson" requ
ired/>
                                        </div>

                                        <div className="form-group">
                                        <label htmlFor="itemName" className="t
ext-dark">Item Name :</label>
                                        <input type="text" name="itemName" cla
ssName="form-control" value={this.deliverItemName}/>
                                        </div>
```

```
                                        <div className="form-group">
                                        <label htmlFor="itemPrice" className="
text-dark">Item Price :</label>
                                        <input type="text" name="itemPrice" cl
assName="form-control" value={localStorage.getItem('itemPrice')}/>
                                        </div>

                                        <div className="form-group">
                                        <label htmlFor="deliveryCash" classNam
e="text-dark">Delivery Cash :</label>
                                        <input type="text" name="deliveryCash"
 className="form-control" value={this.deliveryCash}/>
                                        </div>

                                        <div className="form-group">
                                        <label htmlFor="totalPayment" classNam
e="text-dark">Total Payment :</label>
                                        <input type="text" name="totalPayment"
 className="form-control" value={this.totalPayment}/>
                                        </div>

                                        <div className="form-group">
                                        <label htmlFor="Email" className="text
-dark">Email :</label>
                                        <input type="text" id="buyerEmail" nam
e="buyerEmail" className="form-
control" onChange={this.onChangeBuyerEmail} placeholder="example@gmail.com" pa
ttern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,3}$" required/>
                                        </div>

                                        <div className="form-group">
                                        <label htmlFor="Address" className="te
xt-dark">Address :</label>
                                        <input type="text" name="buyerAddress"
 className="form-
control" onChange={this.onChangeBuyerAddress} placeholder="Your Delivery Addre
ss" required/>
                                        </div>

                                        <div className="form-group">
                                        <label htmlFor="Phone" className="text
-dark">Contact Number :</label>
                                        <input type="numbers" name="buyerPhone
" className="form-
control"  onChange={this.onChangeBuyerPhone} placeholder="Enter 10 digit valid
 contact number" pattern="[0-9]{10}" required/>
                                        </div>
```

```
                                        <span className="text-
dark">Note that we have given you two type of delvery options.</span>
                                        <span className="text-
dark">Deliver using a fast delivery hires the delivery company to deliver your
 product. it is fast method but it  will cost you 10% of your item payment.</s
pan>
                                        <span className="text-
dark">Cash on delivery is free of charge but it is slow compared to the other.
</span>

                                        <div className="form-group">
                                        <label htmlFor="deliveryoption" classN
ame="text-dark">Delivery option :</label>
                                            <div className="radio inline">
                                            <input type="radio" id="radio-
2" onChange={this.onChangeDeliveryOption} name="deliveryOption" value="FastDel
ivery" />
                                            <label htmlFor="radio-
2" className="radio-label">Fast Delivery</label>
                                            </div>

                                            <div className="radio inline">
                                            <input type="radio" id="radio-
3" onChange={this.onChangeDeliveryOption}  name="deliveryOption" value="CashOn
Delivery" />
                                            <label htmlFor="radio-
3" className="radio-label">Cash On Delivery</label>
                                            </div>
                                        </div>

                                        <div className="form-group">
                                        <label htmlFor="payoption" className="
text-dark">Payment option :</label>
                                            <div className="radio inline">
                                            <input type="radio" id="radio-
4" onChange={this.onChangePayOption} name="payOption" value="CreditCard" />
                                            <label htmlFor="radio-
4" className="radio-label">Credit Card</label>
                                            </div>

                                            <div className="radio inline">
                                            <input type="radio" id="radio-
5" onChange={this.onChangePayOption}  name="payOption" value="MobilePayment" /
>
                                            <label htmlFor="radio-
5" className="radio-label">Mobile Payment</label>
                                            </div>
```

```
                                        </div>

                                        <div className="form-group">
                                            <button className="btn text-
dark update" onClick={this.saveDelivery} >Go to Payments</button>
                                        </div>
                                        </div>
                                    </form>
                                </div>
                            </main>
                        <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/j
query.min.js" integrity="sha512-
894YE6QWD5I59HgZOGReFYm4dnWc1Qt5NtvYSaNcOP+u1T9qYdvdihz0PPSiiqn/+/3e7Jo4EaG7Tu
bfWGUrMQ==" crossOrigin="anonymous"></script>
                        <script src="/js/index.js"></script>
                        </div>
                );
            }
}
```

13. httprequest.js

```
import axios from 'axios';
export default axios.create({
    baseURL:"http://localhost:8080/api",
    headers:{
        "Content-type":"application/json"
    }
});
```

14. Pay.js

```
import React, { Component } from 'react'
import axios from 'axios';
import HeaderNormal from './HeaderNormal'
export default class Pay extends Component {
    constructor(props){
        super(props)
        this.state = this.initialState;
        this.paymentChange = this.paymentChange.bind(this);
        this.submitPay = this.submitPay.bind(this);
  }
  submitPay = event =>{
      event.preventDefault();
      const payment = {
        paytype:'paypal',
        description:localStorage.getItem('itemName'),
        total:localStorage.getItem('totalPayment'),
```

```
        currency:'USD',
        intent:'Sale'
      };

      axios.post("http://localhost:8080/pay", payment)
      .then(response => {
        if(response.data != null) {
            this.setState(this.initialState);
            window.open(response.data);
        }
        else{
            alert('Payment Failed');
        }
    })

  }

  paymentChange = event =>{
      this.setState({
        [event.target.name]:event.target.value
      });
  }


    render() {

        return (
            <div>
                <HeaderNormal/>

                <div className="container">
            <div className="view-account">
                <div className="content-panel">
                    <h2 className="title">Payment Details</h2><br/>
                    <div className="billing">
                        <div className="secure text-center margin-bottom-md">
                        <div className="accepted-cards">
                            <ul className="list-inline  row justify-content-
center">
                            <li className="col"><img src="https://www.uxfordev
.com/demo/1.0.6/assets/images/payment-icon-set/icons/visa-curved-
32px.png" alt="Visa"/></li>
                            <li className="col"><img src="https://www.uxfordev
.com/demo/1.0.6/assets/images/payment-icon-set/icons/mastercard-curved-
32px.png" alt="MasterCard"/></li>
                            <li className="col"><img src="https://www.uxfordev
.com/demo/1.0.6/assets/images/payment-icon-set/icons/maestro-curved-
32px.png" alt="Maestro"/></li>
```

```
                            <li className="col"><img src="https://www.uxfordev
.com/demo/1.0.6/assets/images/payment-icon-set/icons/american-express-curved-
32px.png" alt="American Express"/></li>
                            </ul>
                        </div>
                        </div>
                        <br/>
                        <form id="billing" className="form-
horizontal" onSubmit={this.submitPay}>
                            <div className="form-group row">
                                <label className="col-sm-5 control-
label">Payment Type</label>
                                <div className="col-sm-5">
                                <select className="form-
control" name="paytype" type="text" required>
                                    <option value="paypal">Paypal</option>
                                </select>
                            </div>
                            </div><br/>
                            <div className="form-group row">
                                <label className="col-sm-5 control-
label">Payment Description</label>
                                <div className="col-sm-7">
                                <input type="text" className="form-
control" name="description" placeholder="Item Code/Name" value={localStorage.g
etItem('itemName')} onChange={this.paymentChange} autoComplete="off" required/
>
                                </div>
                            </div><br/>
                            <div className="row form-group">
                                <label className="col-sm-5 control-
label">Total Charge</label>
                                <div className="form col-sm-4">
                                <input type="text" className="form-
control"  name="total" placeholder="Total Charging Amount" value={localStorage
.getItem('totalPayment')} onChange={this.paymentChange} required/>
                                </div>
                            </div><br/>
                            <div className="form-group row">
                                <label className="col-sm-5 control-
label">Currency Type</label>
                                <div className="col-sm-5">
                                <select className="form-
control" name="currency" type="text" required>
                                    <option value="USD">USD</option>
                                </select>
                                </div>
                            </div><br/>
```

```
                            <div className="form-group row">
                                <label className="col-sm-5 control-
label">Intent</label>
                                <div className="col-sm-7">
                                <input type="text" className="form-
control" name="intent" placeholder="Intent of Payment" value='Sale' readonly r
equired/>
                                </div>
                            </div>
                        <hr/>
                        <div className="action-wrapper text-center">
                            <div className="action-btn">
                            <button className="btn btn-success btn-
lg" type="submit" >
                                Pay Now
                                <i className="fa fa-chevron-right"></i>
                            </button>
                            </div>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>
        )
    }
}
```

15. **Model.jsx**

```
import React, {Component} from 'react';
import {Button, Form} from "react-bootstrap";
import service from "../services/service";
import '../../src'
import HeaderNormal from '../components/HeaderNormal'

export default class Model extends Component {
    constructor(props) {
        super(props);
        this.state = {
            custName:localStorage.getItem('buyerName'),
            custEmail:localStorage.getItem('buyerEmail'),
            custPhone:localStorage.getItem('buyerPhone'),
            sellerEmail:localStorage.getItem('sellerEmail'),
            itemName:localStorage.getItem('itemName'),
            itemPrice:localStorage.getItem('itemPrice'),
            otp:"",
```

```
            pinError:""
        }
        this.changeOtpHandler=this.changeOtpHandler.bind(this);
    }

    initialState = {
        pinError:""
    }

    changeOtpHandler = (event) =>{
        this.setState({otp: event.target.value});
    }

    cancelHandler = (e) =>{
        e.preventDefault();
        console.log("cancel button hit");
    }

    resendHandler = (e) =>{
        e.preventDefault();
        console.log('reSend button hit');
        service.sendPin().then((res) =>{
            console.log(res);
        });
    }

    confirmHandler = (e) =>{
        e.preventDefault();

        let pinNumber = {
            otp:this.state.otp
        };

        let paymeny = {
            custName: this.state.custName,
            custEmail: this.state.custEmail,
            custPhone: this.state.custPhone,
            sellerEmail: this.state.sellerEmail,
            itemName: this.state.itemName,
            itemPrice:this.state.itemPrice,
        }
        console.log("Confirm button");
        console.log(">>>>> " + JSON.stringify(pinNumber));
        service.getPinVerify(pinNumber).then((res) => {;
            this.initialState.pinError=res.data;
            if(this.initialState.pinError === "OTP verification success !"){
                service.addPayment(paymeny).then(()=>{
                    this.props.history.push('/');
```

```
                }).catch(newError =>{
                    console.error(newError);
                })
            }else{
                window.alert(res.data);
                return;
            }
        }).catch(error => {
            window.prompt(error);
        });
    }

    render() {
        return (
            <div>
                <HeaderNormal/>
                <Form className="form_2">
                    <h3>Verify your phone number</h3>
                    <br/>
                    <Form.Group controlId="formBasicEmail">
                        <Form.Label style={{fontSize:'medium'}}>Enter PIN</For
m.Label>
                        <Form.Control type="text" placeholder="PIN" name="otp"
                        value={this.state.otp} onChange={this.changeOtpHandler
}/>
                        <Button variant="success" style={{float:'right',margin
Left:'10px'}} onClick={this.confirmHandler}>
                        Confirm
                    </Button>
                        <Button onClick={this.resendHandler} style={{float:'ri
ght',marginLeft:'10px'}}>Resend Pin</Button>
                    </Form.Group>

                    {/* <Button variant="danger" style={{float:'right'}} onCli
ck={this.cancelHandler}>
                        Cancel
                    </Button> */}
                </Form>
            </div>
        );
    }
}
```

16. **SingleItem.js**

```
import React, { Component } from 'react'
import axios from 'axios';
import HeaderNormal from './HeaderNormal'
```

```
import { Redirect } from "react-router";
export default class SingleItem extends Component {
    constructor(props){
        super(props)
        this.state = {
            items: []
        };
    }
    componentDidMount(){
        this.showSingleItems();
    }
    showSingleItems = (itemId) => {
        axios.get("http://localhost:8080/api/buyerView/4", itemId)
            .then(response => response.data)
            .then((data) => {
                this.setState({items: data});

            });
    }
    state = {
        redirect: false
    }
    redirectHandler11 = () => {
        this.setState({ redirect: true })
        this.renderRedirect11();
    }
    renderRedirect11 = () => {
        if (this.state.redirect) {
            return <Redirect to='/deliveries' />
        }
    }
    render () {
        localStorage.setItem('sellerEmail', this.state.items.sellerEmail);
        localStorage.setItem('itemName', this.state.items.itemName);
        localStorage.setItem('itemPrice', this.state.items.itemPrice);
        return(
            <div>
                <HeaderNormal/>
            <div className="contApp2">
                <div class="row border border-5">
                    <div class="col-5">
                        <img src={`data:itemImage/jpg;base64,${this.state.item
s.itemImage}`}/>
                    </div>
                    <div class="col-5">
                        <h5>{this.state.items.itemName}</h5>
                        <p class="mb-2 text-
muted  ">Category Name :  {this.state.items.catName}</p>
```

```
                        <p><span class="mr-
1"><strong>Price :  $ . {this.state.items.itemPrice}</strong></span></p>
                        <p class="pt-1">{this.state.items.itemDescription}</p>
                            <div class="table-responsive">

                            </div>
                            <hr/>

                        <button type="button" onClick={this.redirectHandler11}
 class="btn btn-primary btn-md mr-1 mb-
2" id="buyNow">Buy now</button>{this.renderRedirect11()}

                    </div>
                </div>
            </div>
        </div>
        )
    }
}
```

Services

1. **deliverService.js**

```
import http from "../httprequests";
class deliverData{
    create(data){
        return http.post("/deliveries", data);
    }
}
export default new deliverData();
```

2. **service.js**

```
import axios from "axios";
const  ADD_PAYMENT_URL = "http://localhost:8080/api/v1/appPayment";
const  GET_PIN = "http://localhost:8080//mobilenumbers/+94767185502/otp";
const  VERIFY_PIN = "http://localhost:8080//mobilenumbers/+94767185502/otps"
class Service{
    addPayment(payment){
        return axios.post(ADD_PAYMENT_URL,payment);
    }
    sendPin(){
        return axios.post(GET_PIN);
    }
    getPinVerify(pinNumber){
        return axios.put(VERIFY_PIN,pinNumber);
    }
```

```
}
export default new Service()
```