BSc (Hons) in Computing
Level C/I/H

INDIVIDUAL ASSIGNMENT

Module Code & Title: COSE60590 Engineering Enterprise Application

Prepared by: [Ashif Shakib] [(CB007534)] [HF20A1SEENG]

Date of Submission: 31st May 2021

Instructor: Mr. Nipunu Wijesinghe

Word Count: 1125

Acknowledgement

I would like to thank Mr. Nipunu for helping and guiding me to finish this Project. And I would like to thank my batchmates and parents who motivated me to finish this project.

Contents

Chapter 01: Web Application

Introduction

Foody is a Food Restaurant located in Colombo. Due to the Current Situation in the Country, it is not safe to have food in the Restaurant. As a solution for this, Foody Restaurant Owner came up with an online Solution where people can make foods order and make the Foods Home Delivery. So, people do not need to gather in the restaurant. Instead of Gathering, they can Order the food from home.

Requirement Analysis

Functional Requirements

Register

Admin Credentials will be added using Database Backend. Customer Have to Register with the System to Make Orders.

Login

To Login to the System. Users must provide Email and password.

View Menu - Customer

Customer Can view the Available Food Menu.

Order Food - Customer

Customer can Order foods form the menu.

Contact Admin

Customer can send messages to the Admin.

Add Reviews

Customer can add reviews about the service, food.

Read Reviews

Customers can read the reviews which are added by other customers.

Update Delivery Status – Customer

After Receiving the order, Customer make the order complete by making the Order Status “Delivered”.

View Profile - Customer

Customer can view the Profile details.

View New Orders – Admin

Admin can view all the New Order.

Update Delivery Status - Admin

Admin will change the status to “On the way” when the Order is on the way to the Customer.

View On the way Orders

Admin can view all the on the way orders.

View Completed Orders

Admin can view all the orders which are delivered to the Customers.

View Contact Messages

Admin can view all the Messages which are sent by Customers.

Add New Food

Admin can Add new Foods to the Menu.

Delete Food

Admin can Deleted the Foods from the Menu

View Food List

Admin can View all the Available foods.

Non-Functional Requirements

- User Friendly GUI – System will have a User friendly and eye-catching Web Application.
- Reliability - Reliability is the degree to which the specified functions are continuously executed by the software system without loss.
- Maintainability - Maintainability is the simplicity with which bugs can be detected and corrected in the software code.
- Usability – user will be a very easy one for use. System will have an instruction page where every step will be displayed.
- Modifiability - Modifiability is the degree to which improvements to a software framework can be developed and executed reliably and cost-effectively.

Hardware Requirements

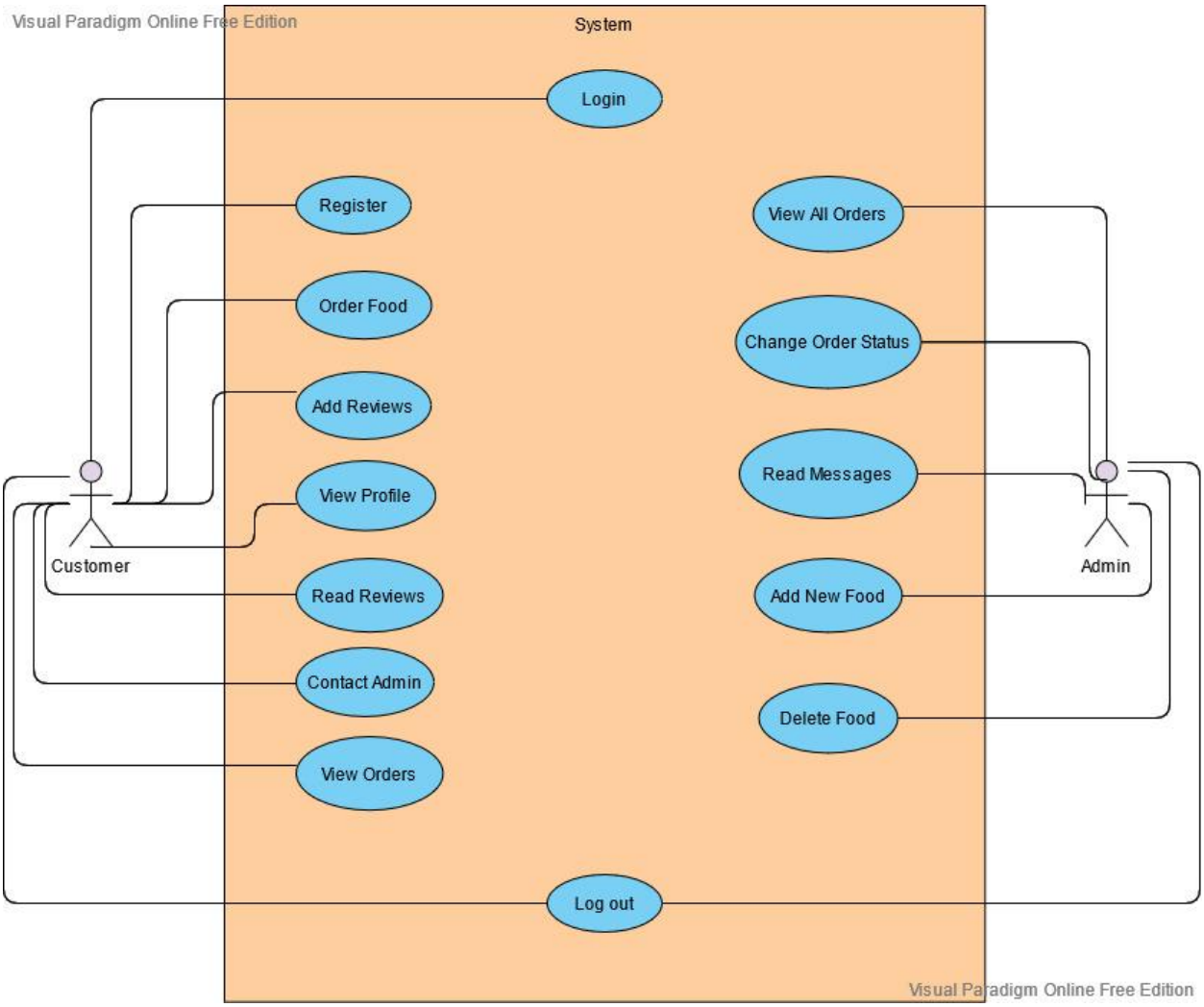
- Device Mechanism - MSI GF63 Thin 95c
- RAM – 8GB
- VGA- 4GB NVIDIA GEFORCE GTX 1650
- Core i5 9th Gen

Software Requirements

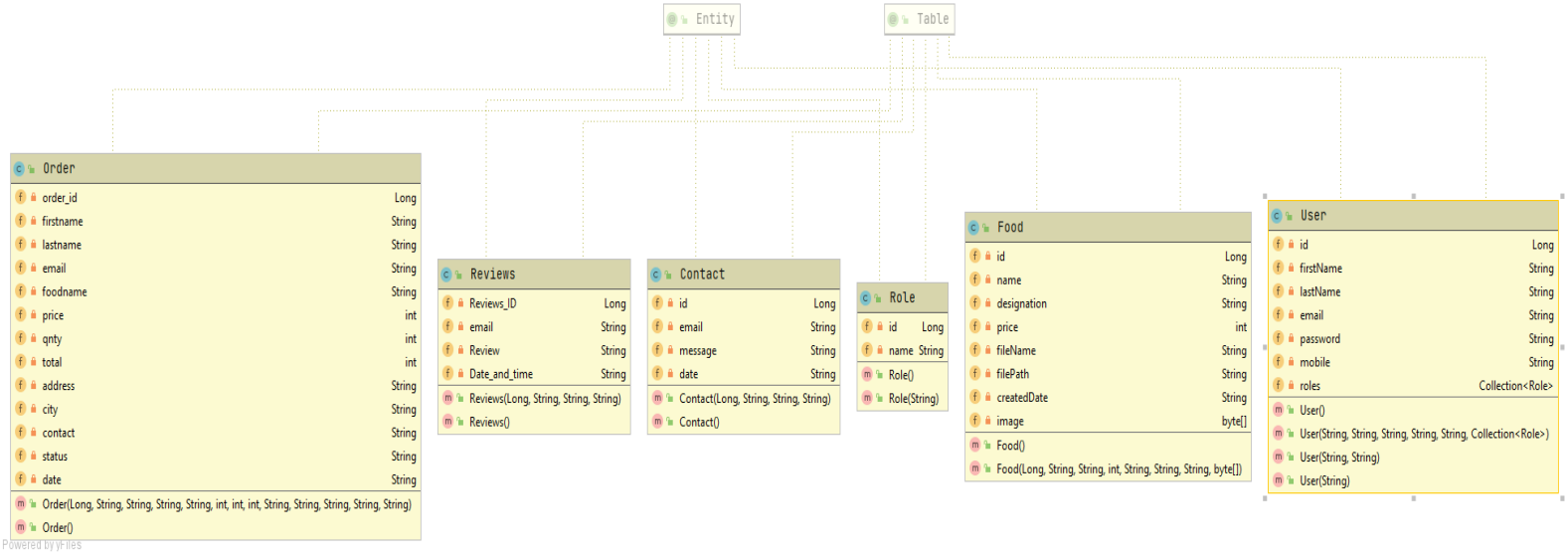
- Operating System – Windows 10
- IntelliJ IDEA
- XAMPP Control Panel

System Design

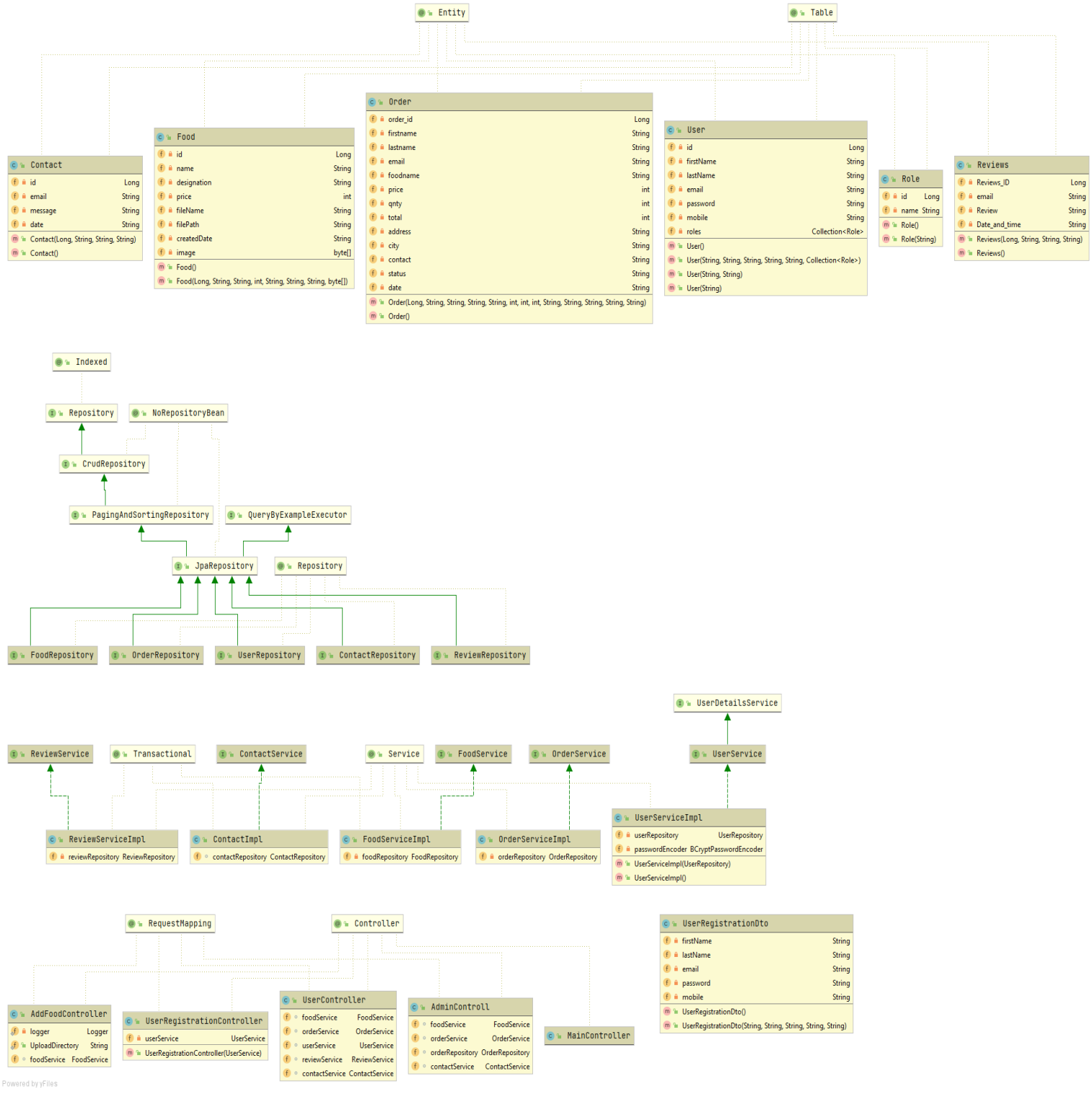
High Level Use Case Diagram



Entity Class Diagram (ER Diagram)

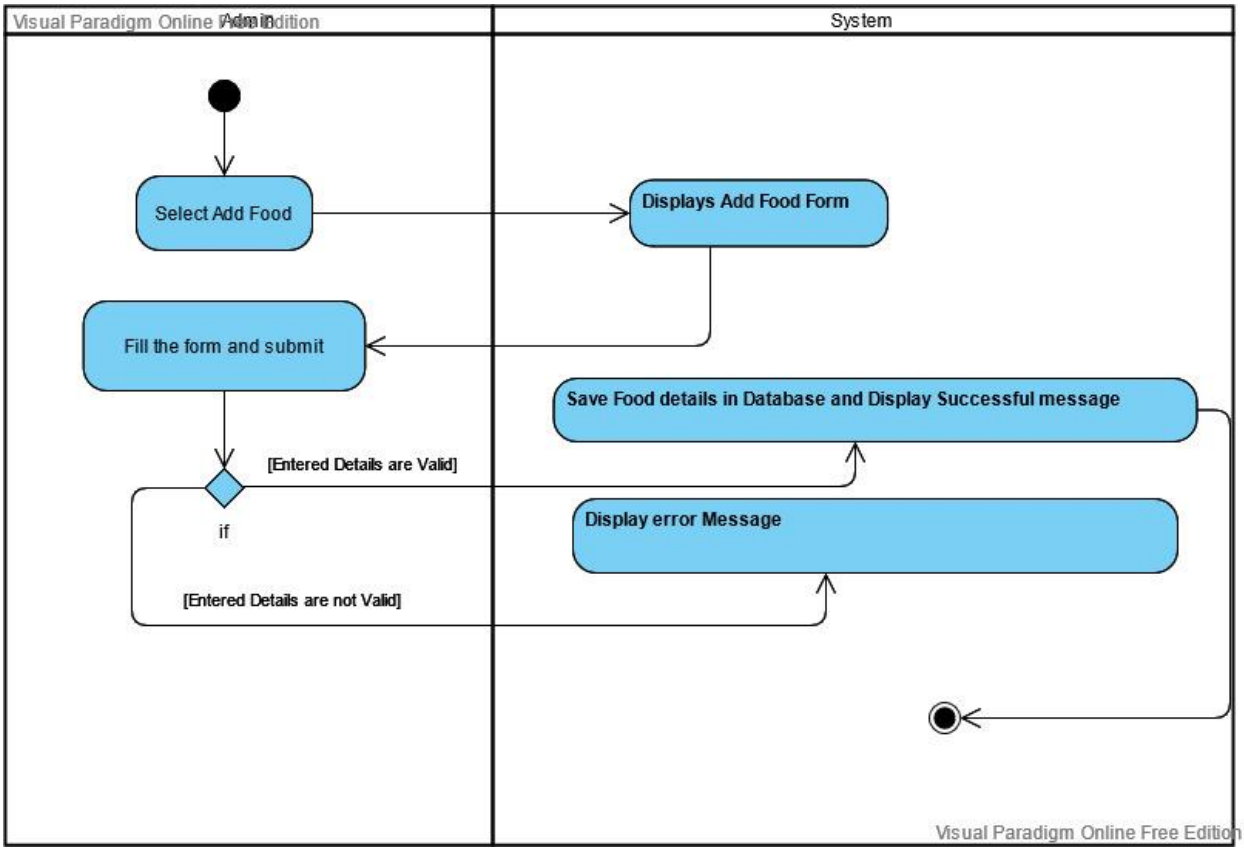


Class Diagram

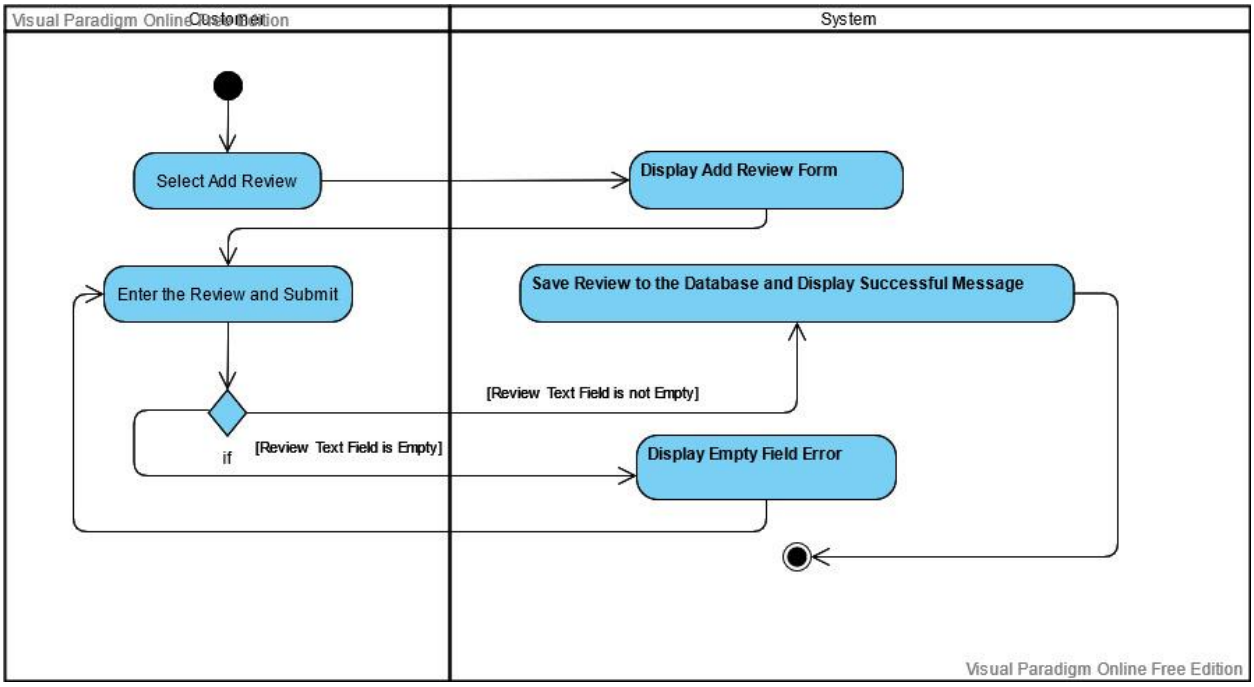


Activity Diagrams

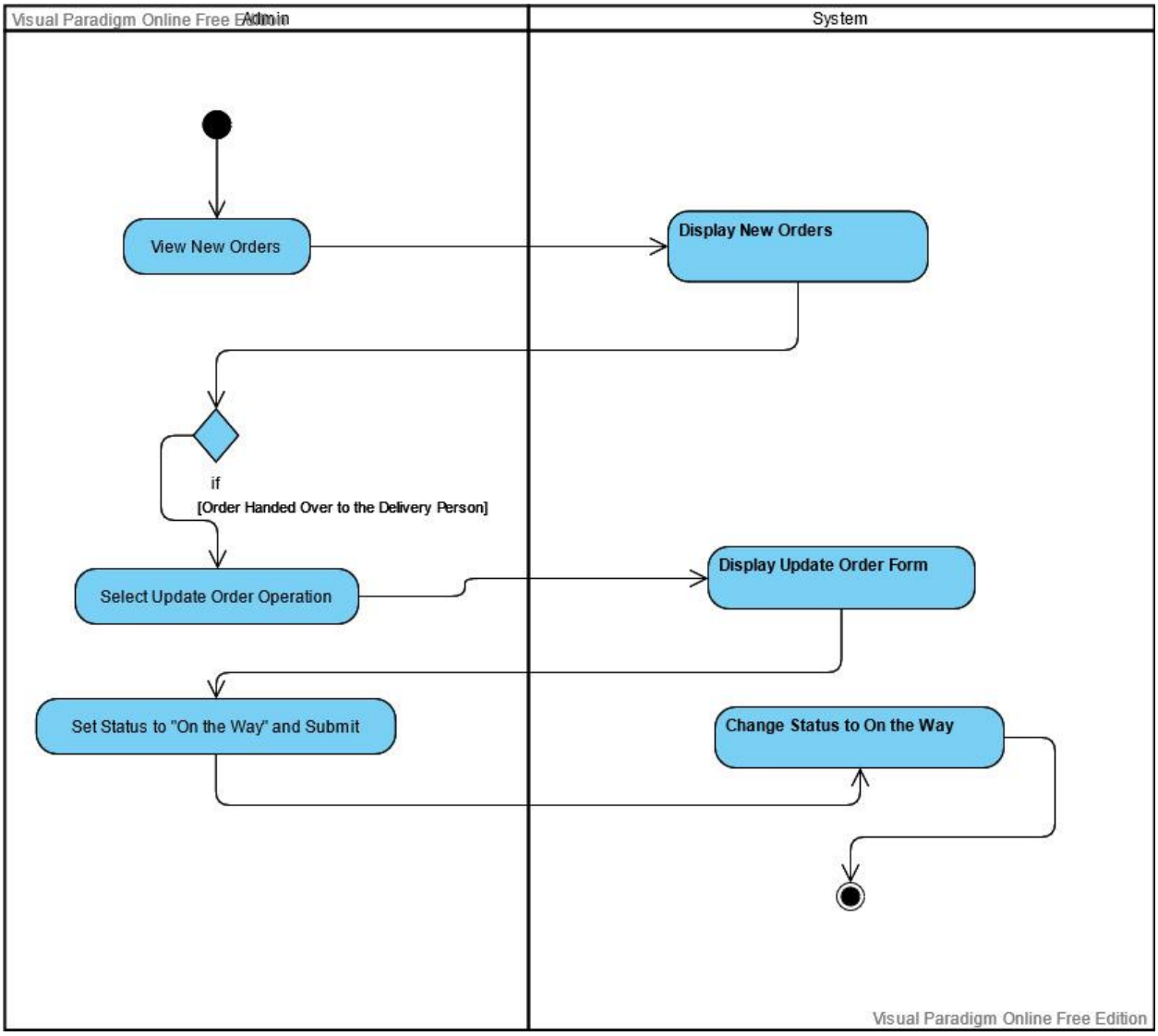
Add Food Activity Diagram



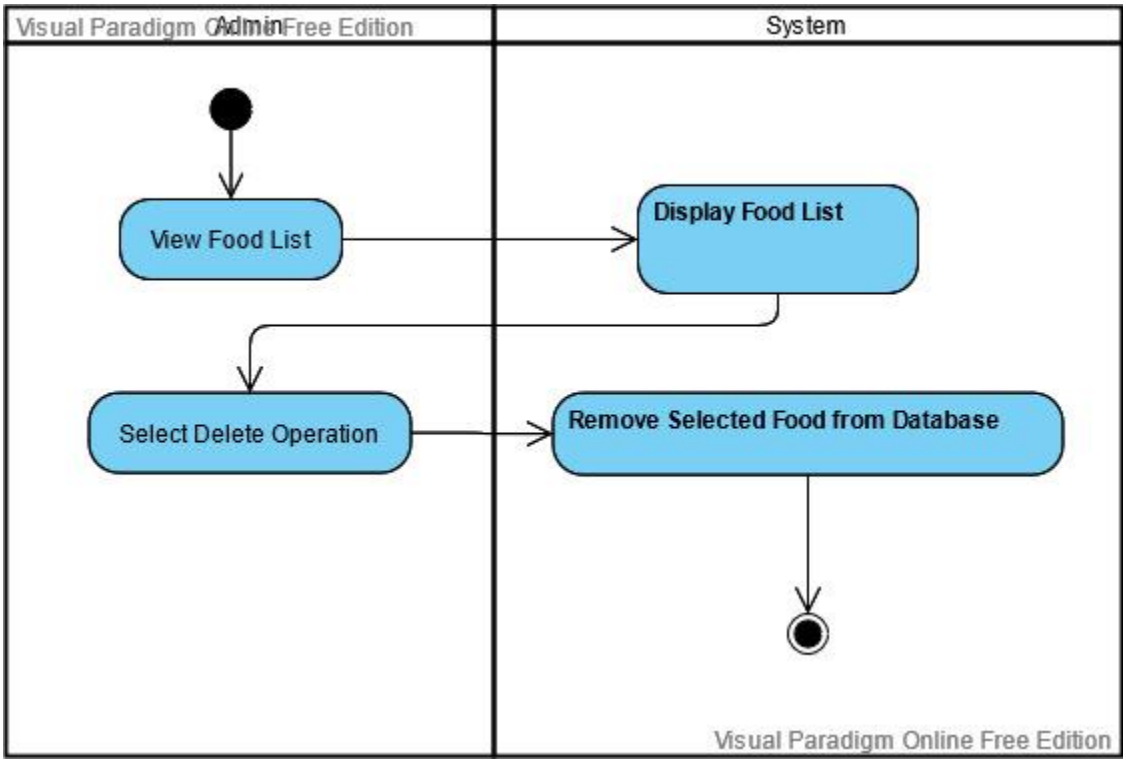
Add Review Activity Diagram



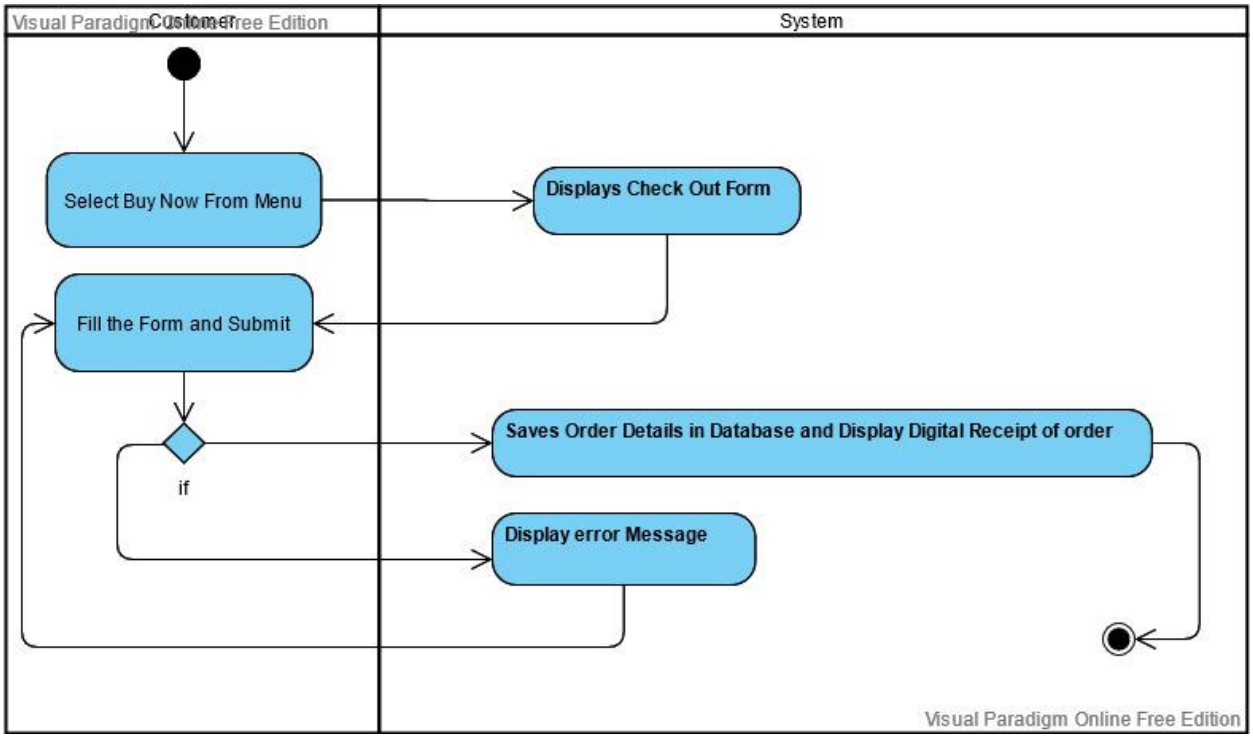
Admin Change Status Activity Diagram



Admin Delete Food Activity Diagram



Customer Order Food Activity Diagram



Implementation

Backend of the Foody Web Application includes Controllers Models, Repositories and Service classes. Frontend of the System based on HTML5 and Bootstrap 4.

Controllers

When the Java class contains the @Controller Annotation, Then the System will Detect it as a Controller class (Bodnar,2020).

Main Controller (MainController.java)

```
@Controller
public class MainController
{
    @GetMapping("/login")
    public String login() { return "login"; }

    @GetMapping("/")
    public String home(Model model)
    {
        ModelAndView modelAndView=new ModelAndView();
        modelAndView.setViewName("home");
        Authentication authentication= SecurityContextHolder.getContext().getAuthentication();
        UserDetails userDetails=(UserDetails)authentication.getPrincipal();
        model.addAttribute( s: "useremail",userDetails);
        return "home";
    }
}
```

@GetMapping annotation used for mapping HTTP GET requests to unique handler methods.

User Registration Controller (UserRegistrationController.java)

```java
@Controller
@RequestMapping("/registration")
public class UserRegistrationController {

    private UserService userService;

    public UserRegistrationController(UserService userService) {
        super();
        this.userService = userService;
    }

    @ModelAttribute("user")
    public UserRegistrationDto userRegistrationDto() { return new UserRegistrationDto(); }

    @GetMapping
    public String showRegistrationForm() { return "registration"; }

    @PostMapping
    public String registerUserAccount(@ModelAttribute("user") UserRegistrationDto registrationDto) {
        userService.save(registrationDto);
        return "redirect:/registration?success";
    }
}
```

@PostMapping annotation used for mapping HTTP POST requests to individual handler methods.

Add Food Controller (AddFoodController.java)

```java
@Controller
@RequestMapping("/add")
public class AddFoodController
{
    private static Logger logger=LoggerFactory.getLogger(AddFoodController.class);
    public static String UploadDirectory=System.getProperty("user.dir")+"/uploads/";

    @Autowired
    FoodService foodService;

    @GetMapping
    public String add()
    {
        return "addfood";
    }
    @PostMapping
    public String createFood(@Valid Food food, @RequestParam("fname")final String name,
                             @RequestParam("description")final String desc, final @RequestParam("file") MultipartFile file,
                             @RequestParam("price")final int price)
    {
        try
        {
            if(food==null)
            {
                return "redirect:/add?unsuccess";
            }
            String filename=file.getOriginalFilename();
            String filepath=Paths.get(UploadDirectory,filename).toString();
            LocalDateTime localDateTime=LocalDateTime.now();
            DateTimeFormatter formatter=DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");
            String created_date=formatter.format(localDateTime);

            BufferedOutputStream stream=new BufferedOutputStream(new FileOutputStream(new File(filepath)));
            stream.write(file.getBytes());
            stream.close();

             byte[] imagedata=file.getBytes();
             String base64EncodedImage = Base64.encodeBase64String(imagedata);

             food.setName(name);
             food.setPrice(price);
             food.setDesignation(desc);
             food.setFileName(filename);
             food.setImage(base64EncodedImage.getBytes(StandardCharsets.UTF_8));
             food.setFilePath(filepath);
             food.setCreatedDate(created_date);

             boolean status=foodService.saveFood(food);
             if(status)
             {
                 return "redirect:/add?success";
             }

         }
         catch (Exception e)
         {
             e.printStackTrace();
             logger.info("Exception :"+e);
         }
         return "redirect:/add?unsuccess";
     }
}
```

User Controller (UserController.java)

```java
@Controller
@RequestMapping("/user")
public class UserController
{
    @Autowired
    FoodService foodService;

    @Autowired
    OrderService orderService;

    @Autowired
    UserService userService;

    @Autowired
    ReviewService reviewService;

    @Autowired
    ContactService contactService;

    @GetMapping(value = "/userfoodlist")
    public String viewfooduser(Model model)
    {
        ModelAndView modelAndView=new ModelAndView();
        modelAndView.setViewName("viewfoodlistuser");
        List<Food> foodList=foodService.getAllFoods();
        model.addAttribute( s: "userfoodlist",foodList);
        return "viewfoodlistuser";
    }
```

```java
    @GetMapping(value = "/checkout/{id}")
    public String Chekout(@PathVariable("id") Long id,Model model)
    {
        ModelAndView modelAndView=new ModelAndView();
        modelAndView.setViewName("CheckOut");
        Optional<Food>foodOptional=foodService.getFoodByID(id);
        model.addAttribute( s: "id",foodOptional.get().getId());
        model.addAttribute( s: "name",foodOptional.get().getName());
        model.addAttribute( s: "details",foodOptional.get().getDesignation());
        model.addAttribute( s: "img",foodOptional.get().getFileName());
        model.addAttribute( s: "price",foodOptional.get().getPrice());
        return "CheckOut";

    }
```

A Spring annotation called **@PathVariable** specifies that a method parameter should be attached to a URI template variable.

```
@PostMapping(value = "/placeorder")
public String make_order(@Valid Order order, @RequestParam("firstname")String firstname, @RequestParam("lastname") String lastname,
                         @RequestParam("address") String address, @RequestParam("city") String city,
                         @RequestParam("contact") String contact, @RequestParam("price")int price, @RequestParam("qty")int qty,
                         @RequestParam("foodname") String foodname,Model model)
{
    try
    {
        order.setFirstname(firstname);
        order.setLastname(lastname);
        order.setAddress(address);
        order.setCity(city);

        Authentication authentication= SecurityContextHolder.getContext().getAuthentication();
        UserDetails userDetails=(UserDetails)authentication.getPrincipal();
        order.setEmail(userDetails.getUsername());

        order.setContact(contact);
        order.setStatus("Preparing");
        order.setPrice(price);
        order.setQnty(qty);
        int total= (price*qty)+100;
        order.setTotal(total);
        order.setFoodname(foodname);

        DateTimeFormatter dateTimeFormatter=DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");
        LocalDateTime localDateTime=LocalDateTime.now();
        order.setDate(dateTimeFormatter.format(localDateTime));

        orderService.saveOrder(order);
        model.addAttribute( s: "foodname",order.getFoodname());

    } catch (Exception e) {
        e.printStackTrace();
    }
     model.addAttribute( s: "foodname",foodname);
     model.addAttribute( s: "qty",qty);
     model.addAttribute( s: "address",address);
     model.addAttribute( s: "price",price);
     model.addAttribute( s: "total", o: (price*qty)+100);
     model.addAttribute( s: "firstname",firstname);
     model.addAttribute( s: "lastname",lastname);
    return "OrderConfirmation";
```

```java
@GetMapping(value = "/viewprofile/{email}")
public String ViewUserDetails(@PathVariable String email,Model model)
{
    ModelAndView detailsview=new ModelAndView();
    detailsview.setViewName("ViewCustomerProfile");
    User userdetails=userService.getUserByEmail(email);
    model.addAttribute( s: "userdetails",userdetails);
    return "ViewCustomerProfile";
}

@PostMapping(value = "/addreview")
public String Add_Reviews(@Valid Reviews reviews,@RequestParam ("review") String review)
{
    try
    {
        if (reviews==null)
        {
            return "redirect:/user/addreviewpage?reviewunsuccess";
        }
        reviews.setReview(review);

        Authentication authentication= SecurityContextHolder.getContext().getAuthentication();
        UserDetails userDetails=(UserDetails)authentication.getPrincipal();
        reviews.setEmail(userDetails.getUsername());

        DateTimeFormatter dateTimeFormatter=DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");
        LocalDateTime localDateTime=LocalDateTime.now();
        reviews.setDate_and_time(dateTimeFormatter.format(localDateTime));

        boolean status=reviewService.AddReviews(reviews);
        if(status)
        {
            return "redirect:/user/addreviewpage?reviewsuccess";
        }
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
    return "redirect:/user/addreviewpage?reviewunsuccess";
}
```

```java
@GetMapping(value = "/addreviewpage")
public String AddReviewsPage() { return "addReviews"; }

@RequestMapping(value = "/viewallreviews")
public String viewAllReviews(Model model)
{
    ModelAndView modelAndView=new ModelAndView();
    modelAndView.setViewName("ViewReviewsUser");
    List<Reviews> reviewsList=reviewService.getAllReviews();
    model.addAttribute( s: "reviewsList",reviewsList);
    return "ViewReviewsUser";
}

@GetMapping(value = "/contactpage")
public String ContactPage() { return "UserContact"; }

@PostMapping(value = "/contact")
public String contact(@Valid Contact contact, @RequestParam ("msge") String msge) {
    try {
        if (contact == null) {
            return "redirect:/user/contactpage?contactunsuccess";
        }
        contact.setMessage(msge);

        Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
        UserDetails userDetails = (UserDetails) authentication.getPrincipal();
        contact.setEmail(userDetails.getUsername());

        DateTimeFormatter dateTimeFormatter = DateTimeFormatter.ofPattern("yyyy/MM/dd HH:mm:ss");
        LocalDateTime localDateTime = LocalDateTime.now();
        contact.setDate(dateTimeFormatter.format(localDateTime));

        boolean message = contactService.AddReviews(contact);
        if (message) {
            return "redirect:/user/contactpage?contactsuccess";
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return "redirect:/user/contactpage?contactunsuccess";
}
```

```java
@GetMapping(value = "/update/{id}")
public String UpdateStatus(@PathVariable("id") Long id,Model model)
{
    Optional<Order> orderdetails=orderService.getOrderByID(id);
    model.addAttribute( s: "id",orderdetails.get().getOrder_id());
    model.addAttribute( s: "address",orderdetails.get().getAddress());
    model.addAttribute( s: "city",orderdetails.get().getCity());
    model.addAttribute( s: "contact",orderdetails.get().getContact());
    model.addAttribute( s: "email",orderdetails.get().getEmail());
    model.addAttribute( s: "firstname",orderdetails.get().getFirstname());
    model.addAttribute( s: "foodname",orderdetails.get().getFoodname());
    model.addAttribute( s: "lastname",orderdetails.get().getLastname());
    model.addAttribute( s: "price",orderdetails.get().getPrice());
    model.addAttribute( s: "quantity",orderdetails.get().getQnty());
    model.addAttribute( s: "status",orderdetails.get().getStatus());
    model.addAttribute( s: "total",orderdetails.get().getTotal());
    model.addAttribute( s: "date",orderdetails.get().getDate());
    return "OrderDetailsUser";
}
```

```java
}

@PostMapping(value = "/updateorder")
public String Update_Order(@Valid Order order, @RequestParam("order_id")Long id,@RequestParam("firstname")String firstname, @RequestParam("lastname") String lastname,
                           @RequestParam("address") String address, @RequestParam("city") String city,
                           @RequestParam("contact") String contact, @RequestParam("price")int price, @RequestParam("qnty")int qty,
                           @RequestParam("foodname") String foodname,@RequestParam("email")String email,@RequestParam("updatestatus")String status,@RequestParam("total")int total,
                           @RequestParam("date")String date, Model model)
{
    try
    {
        order.setOrder_id(id);
        order.setFirstname(firstname);
        order.setLastname(lastname);
        order.setAddress(address);
        order.setCity(city);
        order.setContact(contact);
        order.setStatus(status);
        order.setPrice(price);
        order.setQnty(qty);
        order.setTotal(total);
        order.setFoodname(foodname);
        order.setDate(date);
        order.setEmail(email);

        orderService.saveOrder(order);

    } catch (Exception e) {
        e.printStackTrace();
    }
    return "ViewCustomerOrders";
}
@RequestMapping(value = "/viewneworders/{email}/{status}")
public String viewneworderlist(@PathVariable ("email")String email, @PathVariable("status") String status, Model model)
{
    ModelAndView orderview=new ModelAndView();
    orderview.setViewName("ViewCustomerOrders");
    List<Order> customerorderlist=orderService.getOrderByEmailAndStatus(email,status);
    model.addAttribute( "customerorderlist",customerorderlist);
    return "ViewCustomerOrders";
}
```

```java
    @RequestMapping(value = "/viewonthewayorders/{email}/{status}")
    public String viewonthewayorderlist(@PathVariable ("email")String email, @PathVariable("status") String status, Model model)
    {
        List<Order> customerorderlist=orderService.getOrderByEmailAndStatus(email,status);
        model.addAttribute( s: "customerorderlist",customerorderlist);
        return "OnTheWayOrdersUser";
    }

    @RequestMapping(value = "/viewcompletedorders/{email}/{status}")
    public String viewcompletedorderlist(@PathVariable ("email")String email, @PathVariable("status") String status, Model model)
    {
        List<Order> customerorderlist=orderService.getOrderByEmailAndStatus(email,status);
        model.addAttribute( s: "customerorderlist",customerorderlist);
        return "CompletedOrdersUser";
    }
}
```

Admin Controller (AdminController.java)

```java
@Controller
@RequestMapping("/admin")
public class AdminControll
{
    @Autowired
    FoodService foodService;

    @Autowired
    OrderService orderService;

    @Autowired
    OrderRepository orderRepository;

    @Autowired
    ContactService contactService;

    @GetMapping
    public String home() { return "home"; }

    @GetMapping(value = "/viewfoodlist")
    public String viewallfoods(Model model)
    {
        ModelAndView modelAndView=new ModelAndView();
        modelAndView.setViewName("viewfoodlistadmin");
        List<Food> foodList=foodService.getAllFoods();
        model.addAttribute( s: "adminfoodlist",foodList);
        return "viewfoodlistadmin";
    }

    @GetMapping(value = "/deletefood/{id}")
    public String deletefood(@PathVariable("id") Long id)
    {
        foodService.deleteFile(id);
        return  "redirect:/admin/viewfoodlist";
    }
```

```java
@GetMapping(value = "/viewneworders/{status}")
public String viewneworderssadmin(@PathVariable("status") String status, Model model)
{
    ModelAndView modelAndView=new ModelAndView();
    modelAndView.setViewName("NewOrdersAdmin");
    List<Order> orderList=orderService.getOrderByStatus(status);
    model.addAttribute( s: "orderlist",orderList);
    return "NewOrdersAdmin";
}

@GetMapping(value = "/update/{id}")
public String UpdateStatus(@PathVariable("id") Long id,Model model)
{
    ModelAndView modelAndView=new ModelAndView();
    modelAndView.setViewName("OrderDetailsAdmin");
    Optional<Order> orderdetails=orderService.getOrderByID(id);
    model.addAttribute( s: "id",orderdetails.get().getOrder_id());
    model.addAttribute( s: "address",orderdetails.get().getAddress());
    model.addAttribute( s: "city",orderdetails.get().getCity());
    model.addAttribute( s: "contact",orderdetails.get().getContact());
    model.addAttribute( s: "email",orderdetails.get().getEmail());
    model.addAttribute( s: "firstname",orderdetails.get().getFirstname());
    model.addAttribute( s: "foodname",orderdetails.get().getFoodname());
    model.addAttribute( s: "lastname",orderdetails.get().getLastname());
    model.addAttribute( s: "price",orderdetails.get().getPrice());
    model.addAttribute( s: "quantity",orderdetails.get().getQnty());
    model.addAttribute( s: "status",orderdetails.get().getStatus());
    model.addAttribute( s: "total",orderdetails.get().getTotal());
    model.addAttribute( s: "date",orderdetails.get().getDate());
    return "OrderDetailsAdmin";
}

@GetMapping(value = "/viewcontacts")
public String ViewContactAdmin(Model model)
{
    ModelAndView modelAndView=new ModelAndView();
    modelAndView.setViewName("ViewMessageAdmin");
    List<Contact> contactList=contactService.getAllMessages();
    model.addAttribute( s: "msgelist",contactList);
    return "ViewMessageAdmin";
}
```

```java
@PostMapping(value = "/updateorder")
public String Update_Order(@Valid Order order, @RequestParam("order_id")Long id,@RequestParam("firstname")String firstname, @RequestParam("lastname") String lastname,
                           @RequestParam("address") String address, @RequestParam("city") String city,
                           @RequestParam("contact") String contact, @RequestParam("price")int price, @RequestParam("qnty")int qty,
                           @RequestParam("foodname") String foodname,@RequestParam("email")String email,@RequestParam("updatestatus")String status,@RequestParam("total")int total,
                             @RequestParam("date")String date, Model model)
{
    try
    {
        order.setOrder_id(id);
        order.setFirstname(firstname);
        order.setLastname(lastname);
        order.setAddress(address);
        order.setCity(city);
        order.setContact(contact);
        order.setStatus(status);
        order.setPrice(price);
        order.setQnty(qty);
        order.setTotal(total);
        order.setFoodname(foodname);
        order.setDate(date);
        order.setEmail(email);

        orderService.saveOrder(order);

    } catch (Exception e) {
        e.printStackTrace();
    }
    return "NewOrdersAdmin";
}

@GetMapping(value = "/onthewayorders/{status}")
public String viewonthewayordersadmin(@PathVariable("status") String status, Model model)
{
    ModelAndView modelAndView=new ModelAndView();
    modelAndView.setViewName("OnTheWayOrdersAdmin");
    List<Order> orderList=orderService.getOrderByStatus(status);
    model.addAttribute( s: "orderlist",orderList);
    return "OnTheWayOrdersAdmin";
}
```

```java
@GetMapping(value = "/completedorders/{status}")
public String completedordersadmin(@PathVariable("status") String status, Model model)
{
    ModelAndView modelAndView=new ModelAndView();
    modelAndView.setViewName("CompletedOrdersAdmin");
    List<Order> orderList=orderService.getOrderByStatus(status);
    model.addAttribute( s: "orderlist",orderList);
    return "CompletedOrdersAdmin";
}
```

Model/Entity Classes

The model in Spring MVC is a container that holds the application's data. Model Classes also known as Entity Classes.

At least two annotations must be specified for each entity: **@Entity** and **@Id.** The **@Entity** annotation indicates that the class is an entity with a database table mapped to it. The name of the database table to be used for mapping is specified by the **@Table** annotation (Bodnar,2020).

Food.java

```java
@Entity
@Table (name = "products")
public class Food
{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Long id;

    @Column(name = "name")
    private String name;

    @Column(name = "description")
    private String designation;

    @Column(name = "price")
    private int price;

    @Column(name = "file_name")
    private String fileName;

    @Column(name = "file_path")
    private String filePath;

    @Column(name = "created_date")
    private String createdDate;

    private byte[] image;

    public Food()
    {

    }

    public Food(Long id, String name, String designation, int price, String fileName, String filePath,String createdDate, byte[] image) {
        this.id = id;
        this.name = name;
        this.designation = designation;
        this.price = price;
        this.fileName = fileName;
        this.filePath = filePath;
        this.createdDate = createdDate;
        this.image = image;
    }

    public Long getId() { return id; }

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

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public String getDesignation() { return designation; }

    public void setDesignation(String designation) { this.designation = designation; }

    public int getPrice() { return price; }

    public void setPrice(int price) { this.price = price; }

    public String getFileName() {
        return fileName;
    }

    public void setFileName(String fileName) { this.fileName = fileName; }

    public String getFilePath() {
        return filePath;
    }

    public void setFilePath(String filePath) { this.filePath = filePath; }

    public String getCreatedDate() { return createdDate; }

    public void setCreatedDate(String createdDate) { this.createdDate = createdDate; }

    public byte[] getImage() { return image; }

    public void setImage(byte[] image) { this.image = image; }

    @Override
    public String toString() {
        return "Food{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", designation='" + designation + '\'' +
                ", price=" + price +
                ", fileName='" + fileName + '\'' +
                ", filePath='" + filePath + '\'' +
                ", createdDate='" + createdDate + '\'' +
                ", image=" + Arrays.toString(image) +
                '}';
    }
}
```

When the Java class contains the @Enitity Annotation, Then the System will Detect it as an Entity class. The @Id annotation specifies an entity's primary key, while the @GeneratedValue annotation specifies generation strategies for primary key values (Bodnar,2020).

Contact.java

```java
@Entity
@Table(name = "contact")
public class Contact
{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String email;

    @Column(length = 1000)
    private String message;

    private String date;

    public Contact(Long id, String email, String message, String date) {
        this.id = id;
        this.email = email;
        this.message = message;
        this.date = date;
    }

    public Contact() {
    }

    public Long getId() { return id; }

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

    public String getEmail() { return email; }

    public void setEmail(String email) { this.email = email; }

    public String getMessage() { return message; }

    public void setMessage(String message) { this.message = message; }

    public String getDate() {
        return date;
    }

    public void setDate(String date) { this.date = date; }
}
```

Order.java

```java
@Entity
@Table(name = "order_info")
public class Order
{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "order_id")
    private Long order_id;

    private String firstname;

    private String lastname;

    private String email;

    private String foodname;

    private int price;

    @Column(name = "quantity")
    private int qnty;

    private int total;

    private String address;

    private String city;

    private String contact;

    private String status;

    private String date;

    public Order(Long order_id, String firstname, String lastname, String email, String foodname, int price, int qnty, int total, String address,
                 String city, String contact, String status,String date) {
        this.order_id = order_id;
        this.firstname = firstname;
        this.lastname = lastname;
        this.email = email;
        this.foodname = foodname;
        this.price = price;
        this.qnty = qnty;
        this.total = total;
        this.address = address;
        this.city = city;
        this.contact = contact;
        this.status = status;
        this.date=date;
    }

    public Order() {

    }

    public Long getOrder_id() { return order_id; }

    public void setOrder_id(Long order_id) { this.order_id = order_id; }

    public String getFirstname() { return firstname; }

    public void setFirstname(String firstname) { this.firstname = firstname; }

    public String getLastname() { return lastname; }

    public void setLastname(String lastname) { this.lastname = lastname; }

    public String getEmail() { return email; }

    public void setEmail(String email) { this.email = email; }

    public String getFoodname() { return foodname; }

    public void setFoodname(String foodname) { this.foodname = foodname; }

    public int getPrice() { return price; }

    public void setPrice(int price) { this.price = price; }

    public int getQnty() { return qnty; }

    public void setQnty(int qnty) { this.qnty = qnty; }

    public String getAddress() { return address; }

    public void setAddress(String address) { this.address = address; }

    public String getCity() { return city; }

    public void setCity(String city) { this.city = city; }

    public String getContact() { return contact; }

    public void setContact(String contact) { this.contact = contact; }

    public String getStatus() { return status; }

    public void setStatus(String status) {
        this.status = status;
    }

    public int getTotal() { return total; }

    public void setTotal(int total) { this.total = total; }

    public String getDate() { return date; }

    public void setDate(String date) { this.date = date; }
}
```

Review.java

```java
@Entity
@Table(name = "reviews")
public class Reviews
{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long Reviews_ID;

    private String email;

    @Column(length = 1000)
    private String Review;

    private String Date_and_time;

    public Reviews(Long reviews_ID, String email, String review, String date_and_time) {
        Reviews_ID = reviews_ID;
        this.email = email;
        Review = review;
        Date_and_time = date_and_time;
    }

    public Reviews() {
    }

    public Long getReviews_ID() { return Reviews_ID; }

    public void setReviews_ID(Long reviews_ID) { Reviews_ID = reviews_ID; }

    public String getEmail() { return email; }

    public void setEmail(String email) { this.email = email; }

    public String getReview() { return Review; }

    public void setReview(String review) { Review = review; }

    public String getDate_and_time() { return Date_and_time; }

    public void setDate_and_time(String date_and_time) { Date_and_time = date_and_time; }
}
```

User.java

```java
@Entity
@Table(name =  "user", uniqueConstraints = @UniqueConstraint(columnNames = "email"))
public class User {

    @Id
    @GeneratedValue(strategy =  GenerationType.IDENTITY)
    private Long id;

    @Column(name = "first_name")
    private String firstName;

    @Column(name = "last_name")
    private String lastName;

    private String email;

    private String password;

    @Column(name = "mobile")
    private String mobile;

    @ManyToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
    @JoinTable(
            name = "users_roles",
            joinColumns = @JoinColumn(
                    name = "user_id", referencedColumnName = "id"),
            inverseJoinColumns = @JoinColumn(
                        name = "role_id", referencedColumnName = "id"))

    private Collection<Role> roles;

    public User() {

    }

    public User(String firstName, String lastName, String email, String password,String mobile, Collection<Role> roles) {
        super();
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
        this.password = password;
        this.roles = roles;
        this.mobile=mobile;
    }

    public User(String email, String firstName) {
    }

    public User(String email) {
    }


    public Long getId() { return id; }
    public void setId(Long id) { this.id = id; }
    public String getFirstName() { return firstName; }
    public void setFirstName(String firstName) { this.firstName = firstName; }
    public String getLastName() { return lastName; }
    public void setLastName(String lastName) { this.lastName = lastName; }
    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }
    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }
    public Collection<Role> getRoles() { return roles; }
    public void setRoles(Collection<Role> roles) { this.roles = roles; }

    public String getMobile() { return mobile; }

    public void setMobile(String mobile) { this.mobile = mobile; }
}
```

To establish a many-to-many relationship between two individuals, the system uses the @ManyToMany annotation.

An intermediate join table is often used to store the association that connects two individuals in a many-to-many association. The @JoinTable annotation is used to describe the join table.

Role.java

```java
@Entity
@Table(name = "role")
public class Role {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;

    public Role() {

    }

    public Role(String name) {
        super();
        this.name = name;
    }

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

UserRegistartionDTO.java

This Class uses Data Transfer Object (DTO) Design Pattern. This Class will be Used to User Registration Process.

Data Transfer Object (DTO) Design Pattern

One of the enterprise application architecture trends, the Data Transfer Object Design Pattern, calls for the use of objects that aggregate and encapsulate data for transfer (Seniuk,2020).

```java
public class UserRegistrationDto {
    private String firstName;
    private String lastName;
    private String email;
    private String password;
    private String mobile;

    public UserRegistrationDto(){

    }

    public UserRegistrationDto(String firstName, String lastName, String email, String password, String mobile) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
        this.password = password;
        this.mobile = mobile;
    }

    public String getFirstName() { return firstName; }

    public void setFirstName(String firstName) { this.firstName = firstName; }

    public String getLastName() { return lastName; }

    public void setLastName(String lastName) { this.lastName = lastName; }

    public String getEmail() { return email; }

    public void setEmail(String email) { this.email = email; }

    public String getPassword() { return password; }

    public void setPassword(String password) { this.password = password; }

    public String getMobile() { return mobile; }

    public void setMobile(String mobile) { this.mobile = mobile; }
}
```

Repository

The annotation @Repository in Spring shows that the decorated class is a repository. A repository is a mechanism that emulates a set of objects and encapsulates storage, retrieval, and search activity.

Food Repository (FoodRepository.java)

```
@Repository
public interface FoodRepository extends JpaRepository<Food,Long>
{

}
```

Contact Repository (ContactRepository.java)

```
@Repository
public interface ContactRepository extends JpaRepository<Contact,Long>
{

}
```

Order Repository (OrderReposiotry.java)

```
@Repository
public interface OrderRepository extends JpaRepository<Order,Long>
{
    List<Order> findByEmail(String email);

    List<Order> findByStatus(String status);

    List<Order> findByEmailAndStatus(String email,String status);


}
```

Review Repository (ReviewRepository.java)

```java
@Repository
public interface ReviewRepository extends JpaRepository<Reviews,Long>
{

}
```

User Repository (UserRepository.java)

```java
@Repository
public interface UserRepository extends JpaRepository<User, Long>{
    User findByEmail(String email);

    List<User> findDetailsByEmail(String email);

}
```

Service

The @Service annotation in Spring is a subset of the @Component annotation. Only classes can be annotated with the Spring Service annotation. It is used to designate a class as a supplier of services.

To Implement the Service Class, System have used Proxy Design Pattern.

Proxy Design Pattern

When the System wants to provide managed access to a functionality, the proxy architecture pattern is used.

Contact Service Interface (ContactService.java)

```
public interface ContactService
{
    boolean AddReviews(Contact contact) throws IOException;

    List<Contact> getAllMessages();
}
```

Contact Service Implementation (ContactServiceImpl.java)

```java
@Service
@Transactional
public class ContactImpl implements ContactService
{
    @Autowired
    ContactRepository contactRepository;

    @Override
    public boolean AddReviews(Contact contact) throws IOException
    {
        try
        {
            if(contact!=null)
            {
                contactRepository.save(contact);
                return true;
            }
        }
        catch (Exception e)
        {
            e.printStackTrace();
            return false;
        }
        return false;
    }

    @Override
    public List<Contact> getAllMessages() { return contactRepository.findAll(); }
}
```

Review Interface (ReviewService.java)

```java
public interface ReviewService
{
    boolean AddReviews(Reviews reviews) throws IOException;

    List<Reviews> getAllReviews();

}
```

Review Service Implementation (ReviewServiceImpl.java)

```java
@Service
@Transactional
public class ReviewServiceImpl implements ReviewService
{
    @Autowired
    private ReviewRepository reviewRepository;

    @Override
    public boolean AddReviews(Reviews reviews)
    {
        try
        {
            if(reviews!=null)
            {
                reviewRepository.save(reviews);
                return true;
            }
        }
        catch (Exception e)
        {
            e.printStackTrace();
            return false;
        }
        return false;

    }

    @Override
    public List<Reviews> getAllReviews() { return reviewRepository.findAll(); }
}
```

Food Interface (FoodService.java)

```java
public interface FoodService
{
    boolean saveFood(Food food) throws IOException;

    List<Food> getAllFoods();

    Optional<Food> getFoodByID(Long id);

    void deleteFile(Long id);
}
```

Food Service Implementation (FoodServiceImpl.java)

```java
@Service
@Transactional
public class FoodServiceImpl implements FoodService
{
    @Autowired
    private FoodRepository foodRepository;

    @Override
    public boolean saveFood(Food food) throws IOException
    {
        try
        {
            if(food!=null)
            {
                foodRepository.save(food);
                return true;
            }
        }
        catch (Exception e)
        {
            e.printStackTrace();
            return false;
        }
        return false;
    }

    @Override
    public List<Food> getAllFoods() { return foodRepository.findAll(); }

    @Override
    public Optional<Food> getFoodByID(Long id) {
        return foodRepository.findById(id);
    }

    @Override
    public void deleteFile(Long id) { foodRepository.deleteById(id); }

}
```

Order Interface (OrderService.java)

```java
public interface OrderService
{
    String saveOrder(Order order);

    Optional<Order> getOrderByID(Long id);

    List<Order> getOrderByStatus(String status);

    List<Order> getOrderByEmailAndStatus(String email,String status);
}
```

Order Service Implementation (OrderServiceImpl.java)

```java
@Service
public class OrderServiceImpl implements OrderService
{
    @Autowired
    private OrderRepository orderRepository;

    @Override
    public String saveOrder(Order order)
    {
        if(order!=null)
        {
            orderRepository.save(order);
        }
        else
        {
            return "Error";
        }
        return "Error";
    }

    @Override
    public Optional<Order> getOrderByID(Long id) { return orderRepository.findById(id); }

    @Override
    public List<Order> getOrderByStatus(String status) { return orderRepository.findByStatus(status); }

    @Override
    public List<Order> getOrderByEmailAndStatus(String email, String status) {
        return orderRepository.findByEmailAndStatus(email,status);
    }

}
```

User Interface (UserService.java)

```java
public interface UserService extends UserDetailsService{
    User getUserByEmail(String email);

    User save(UserRegistrationDto registrationDto);

    boolean passwordCheck(String password, String password1);
}
```

User Service Implementation (UserServiceImpl.java)

```java
@Service
public class UserServiceImpl implements UserService{

    @Autowired
    private UserRepository userRepository;

    @Autowired
    private BCryptPasswordEncoder passwordEncoder;

    public UserServiceImpl(UserRepository userRepository) {
        super();
        this.userRepository = userRepository;
    }

    public UserServiceImpl() {

    }

    @Override
    public User getUserByEmail(String email) {
        return userRepository.findByEmail(email);
    }

    @Override
    public User save(UserRegistrationDto registrationDto) {
        User user = new User(registrationDto.getFirstName(),
                registrationDto.getLastName(), registrationDto.getEmail(),
                passwordEncoder.encode(registrationDto.getPassword()),registrationDto.getMobile(), Arrays.asList(new Role( name: "ROLE_USER")));
        userRepository.save(user);
        return userRepository.findByEmail(user.getEmail());
    }

    @Override
    public boolean passwordCheck(String password, String password1) {
        return passwordEncoder.matches(password, password1);
    }

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

        User user = userRepository.findByEmail(username);
        if(user == null) {
            throw new UsernameNotFoundException("Invalid username or password.");
        }
        return new org.springframework.security.core.userdetails.User(user.getEmail(), user.getPassword(), mapRolesToAuthorities(user.getRoles()));
    }

    private Collection<? extends GrantedAuthority> mapRolesToAuthorities(Collection<Role> roles){
        return roles.stream().map(role -> new SimpleGrantedAuthority(role.getName())).collect(Collectors.toList());
    }


}
```

View

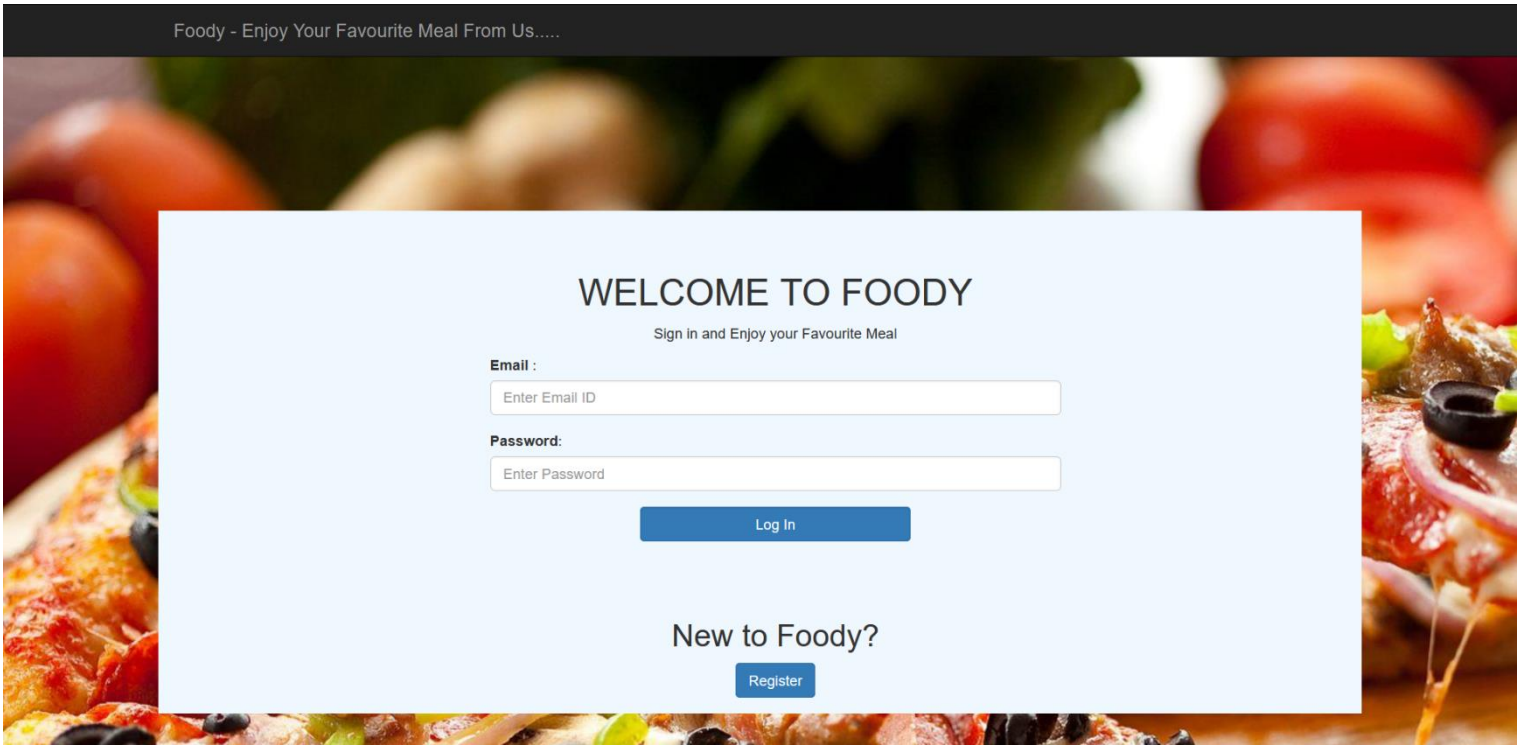
The View oversees rendering the model data, and it produces HTML output that the client's browser can understand.

In this Part, document will discuss about the HTML Pages.

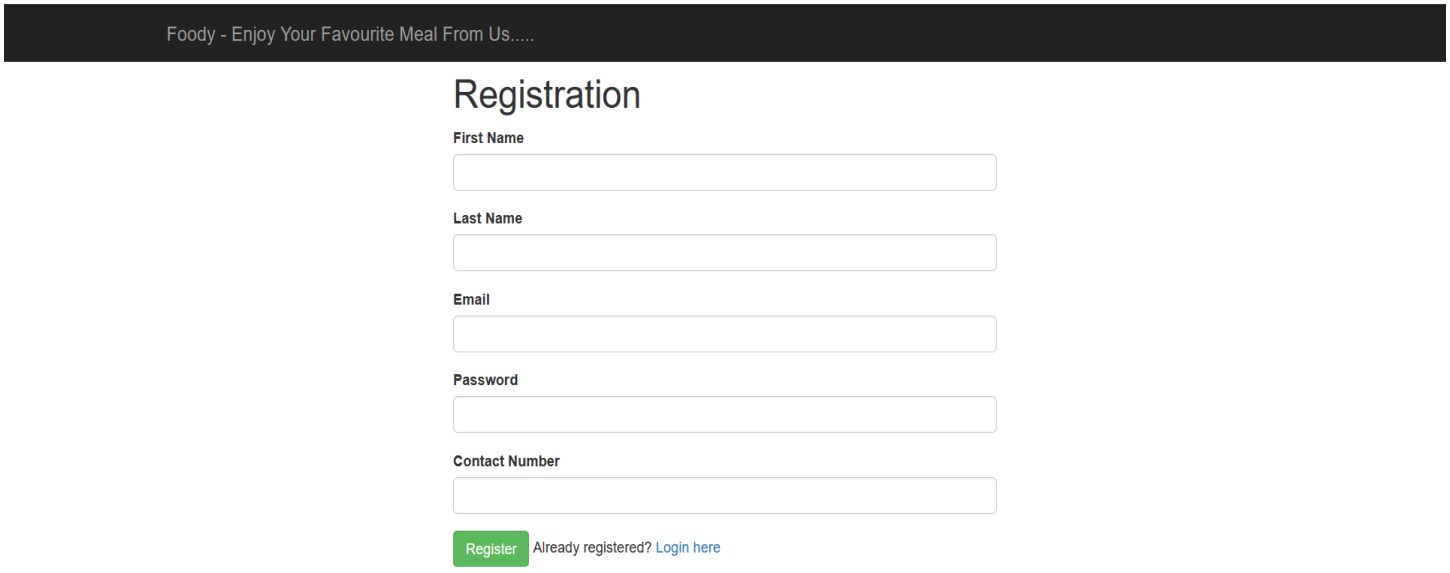
Thymleaf

Thymeleaf is a web application development library based on Java. It offers excellent support for serving XHTML/HTML5 in web applications. It is a server-side Java template engine that can be used in both web (servlet-based) and offline (non-web) environments. It is a JSP replacement.

Login.html

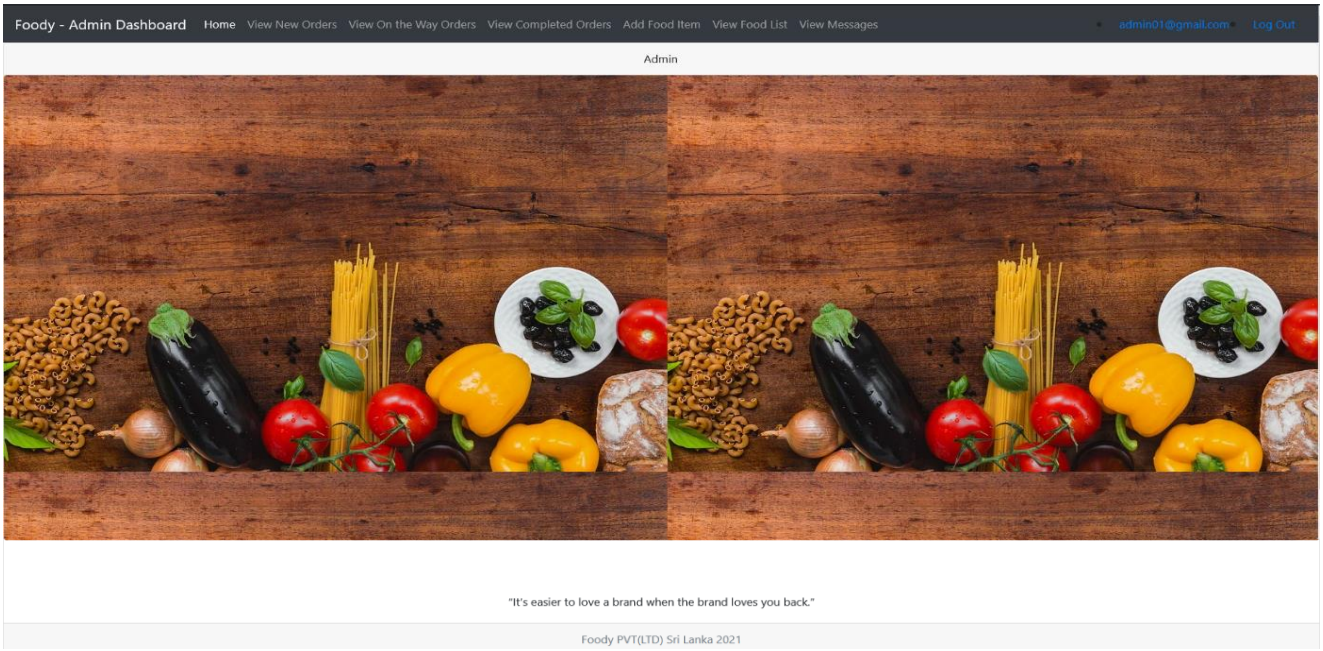

Foody - Enjoy Your Favourite Meal From Us.....
WELCOME TO FOODY
Sign in and Enjoy your Favourite Meal
Email :
Enter Email ID
Password:
Enter Password
Log In
New to Foody?
Register

registration.html

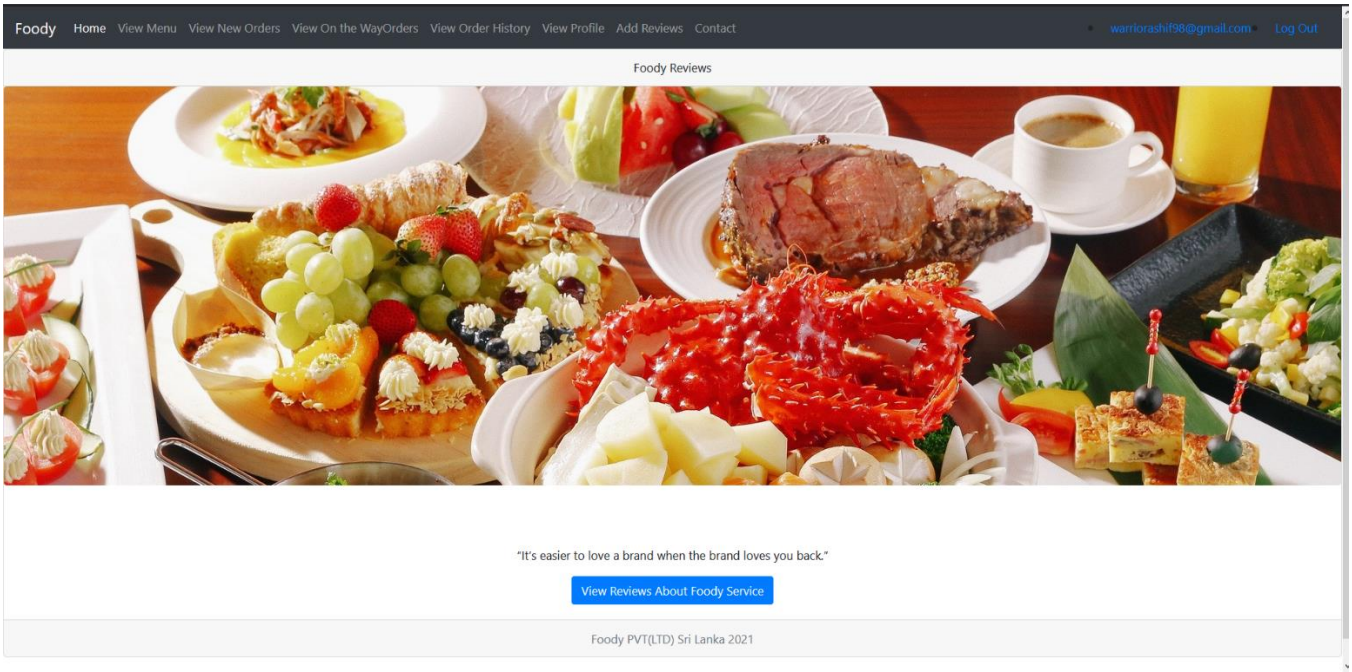


home.html

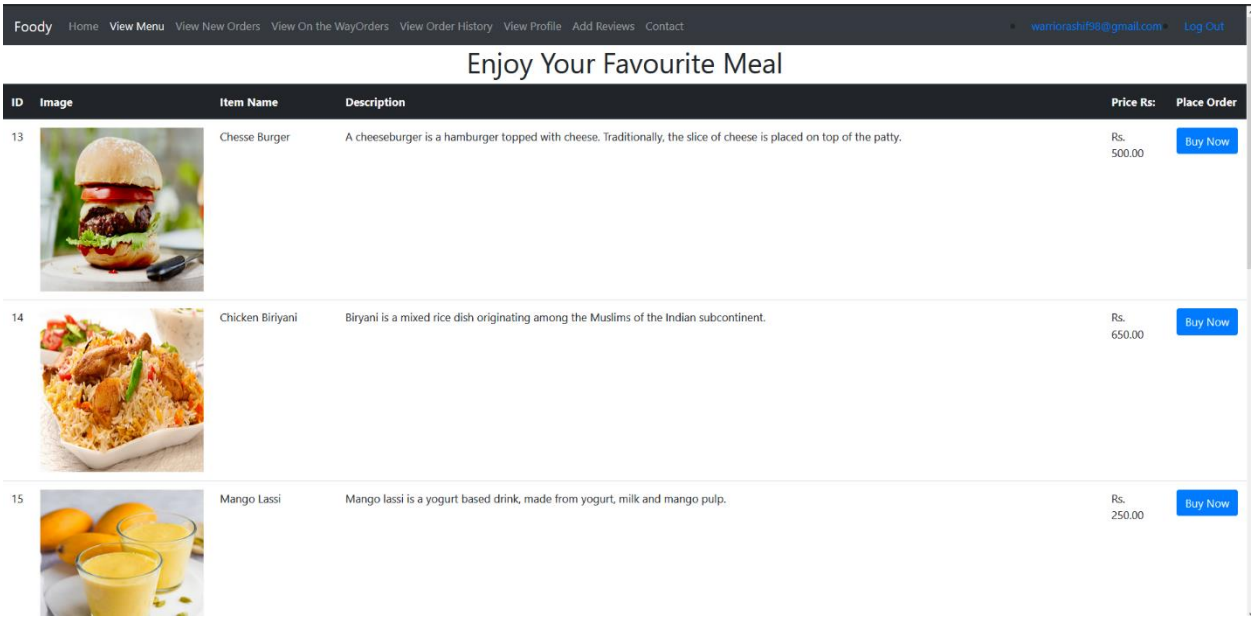
Admin login Preview



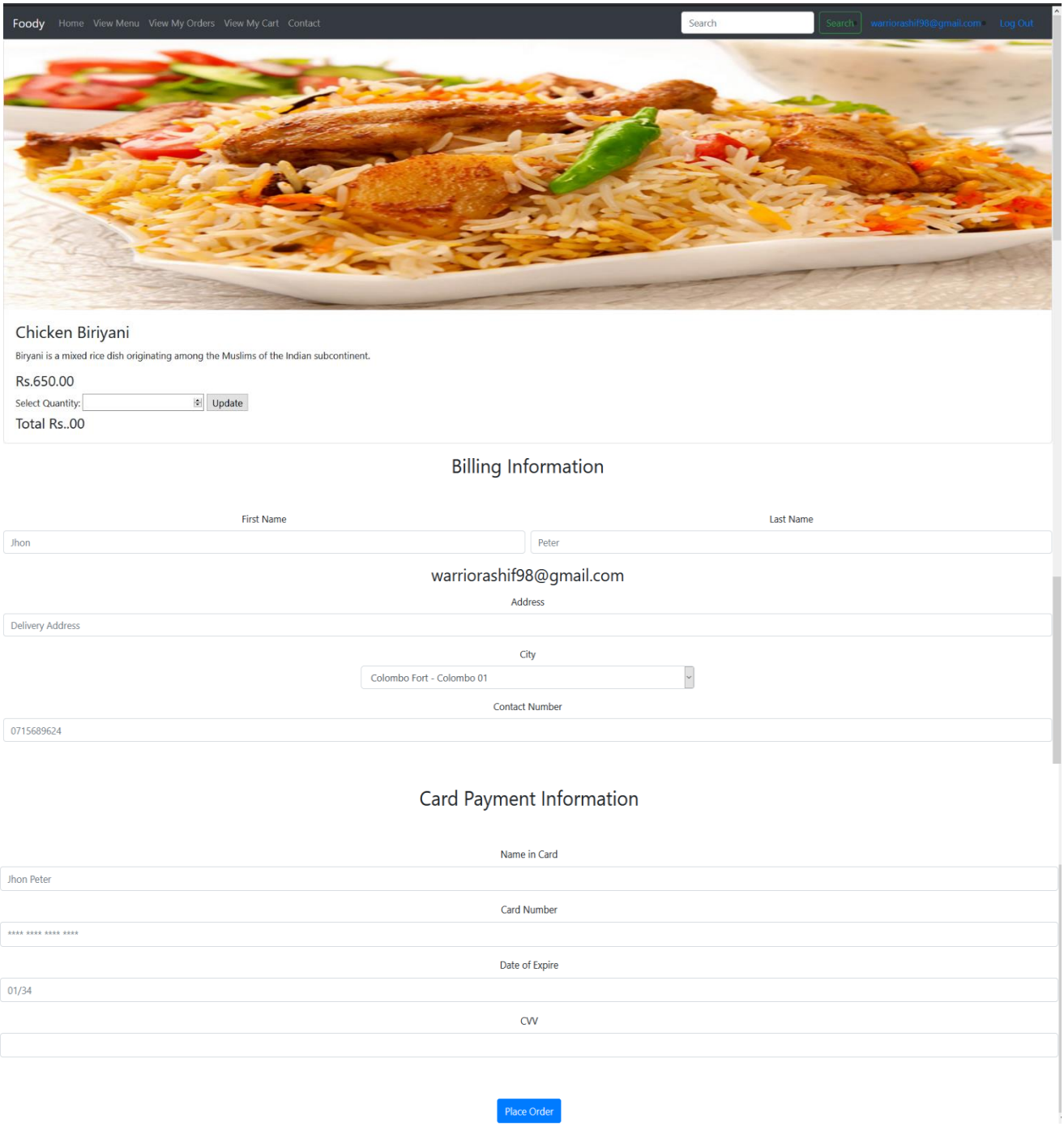
Customer login Preview



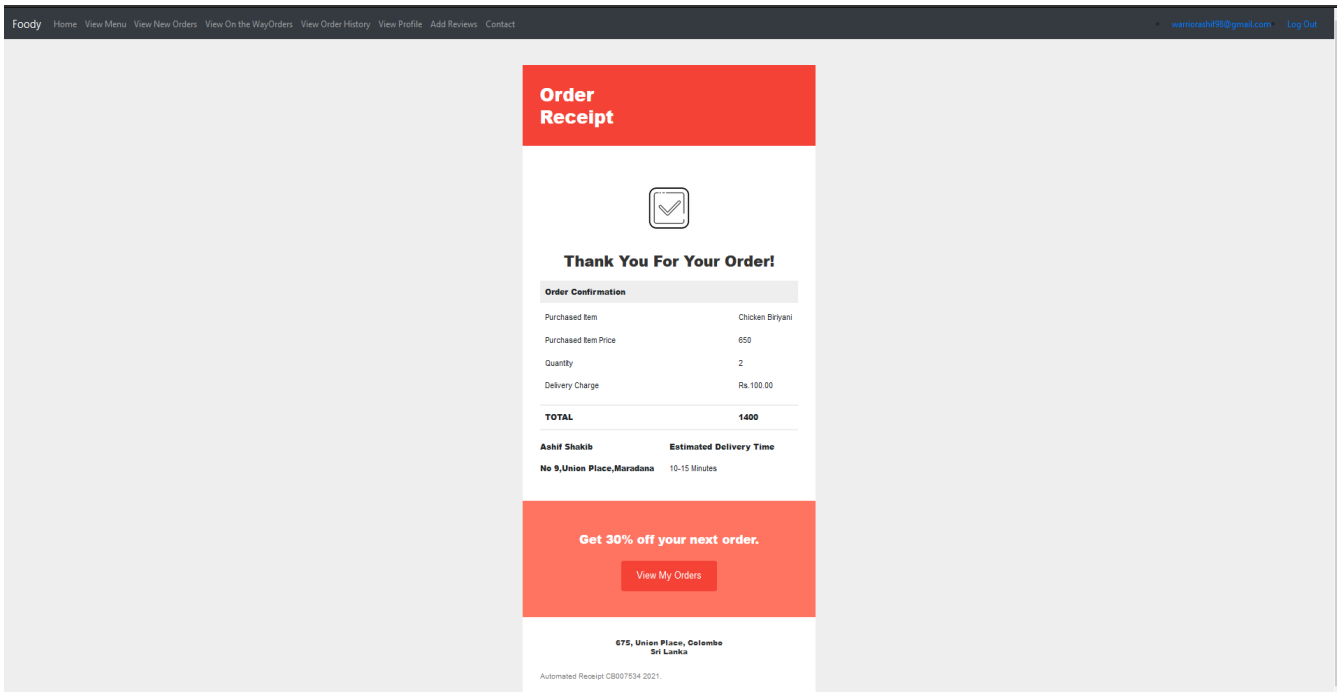
viewfoodlistuser.html



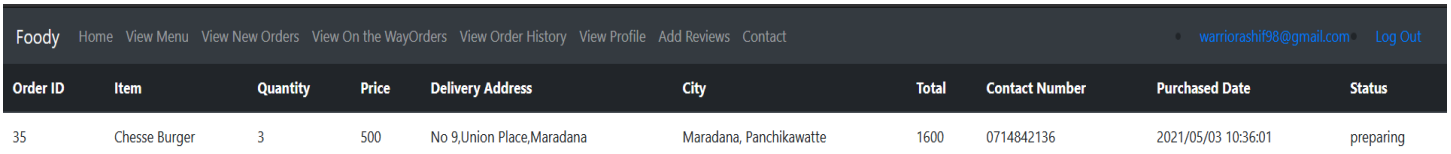
CheckOut.html



OrderConfirmation.html

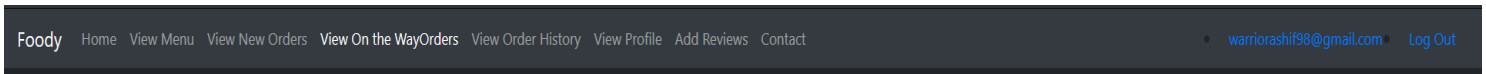


ViewCustomerOrders.html



Order ID	Item	Quantity	Price	Delivery Address	City	Total	Contact Number	Purchased Date	Status
35	Chesse Burger	3	500	No 9,Union Place,Maradana	Maradana, Panchikawatte	1600	0714842136	2021/05/03 10:36:01	preparing

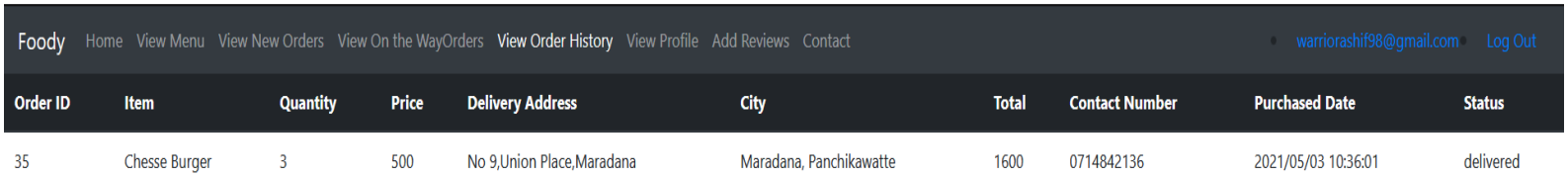
OnTheWayOrdersUser.html



Order ID	Item	Quantity	Price	Delivery Address	City	Total	Contact Number	Purchased Date	Status	Action
35	Chesse Burger	3	500	No 9,Union Place,Maradana	Maradana, Panchikawatte	1600	0714842136	2021/05/03 10:36:01	on the way	UPDATE ORDER

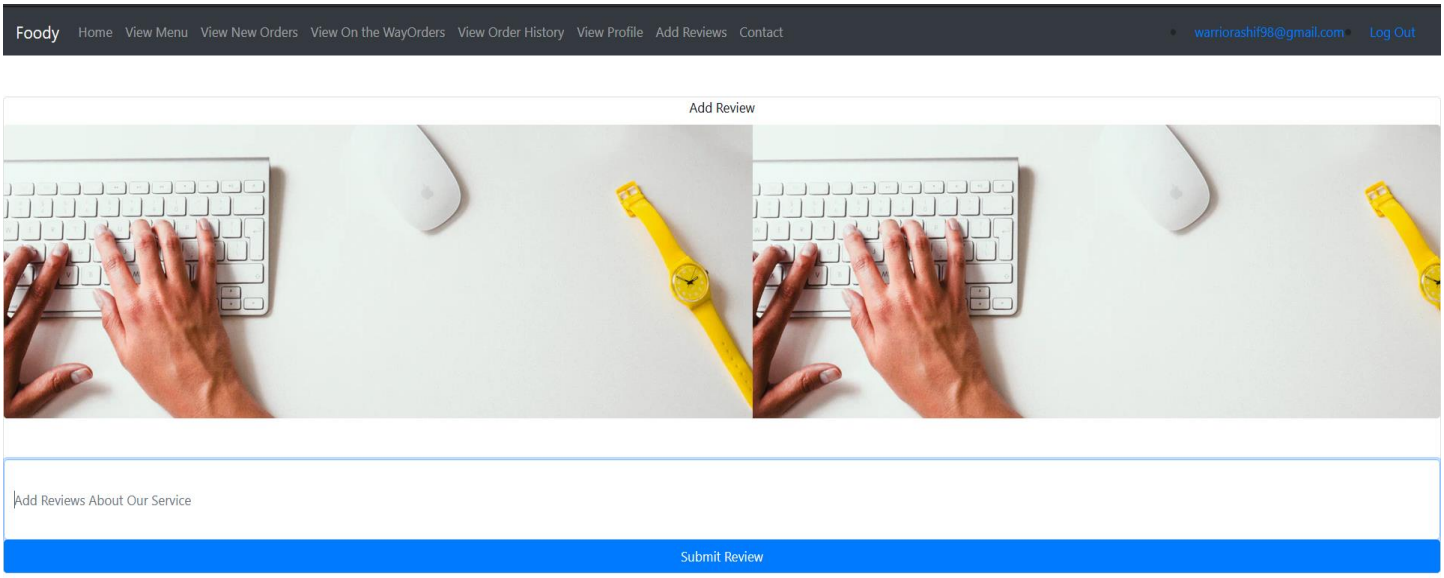
OrderDetailsUser.html

CompletedOrdersUser.html

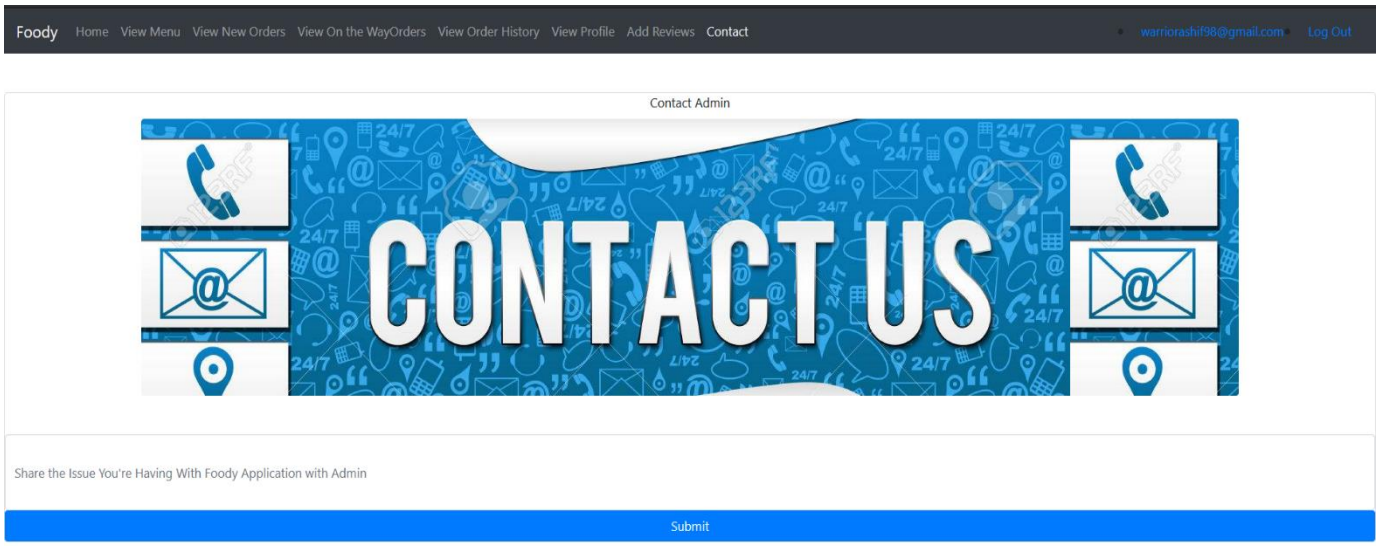


Order ID	Item	Quantity	Price	Delivery Address	City	Total	Contact Number	Purchased Date	Status
35	Chesse Burger	3	500	No 9,Union Place,Maradana	Maradana, Panchikawatte	1600	0714842136	2021/05/03 10:36:01	delivered

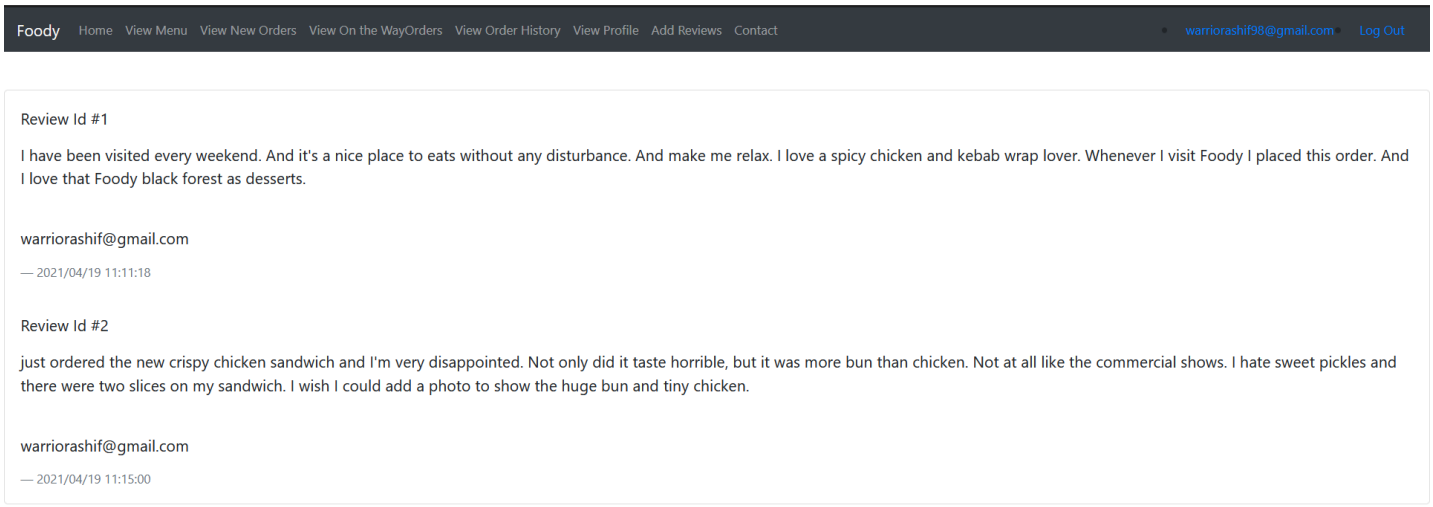
addReviews.html



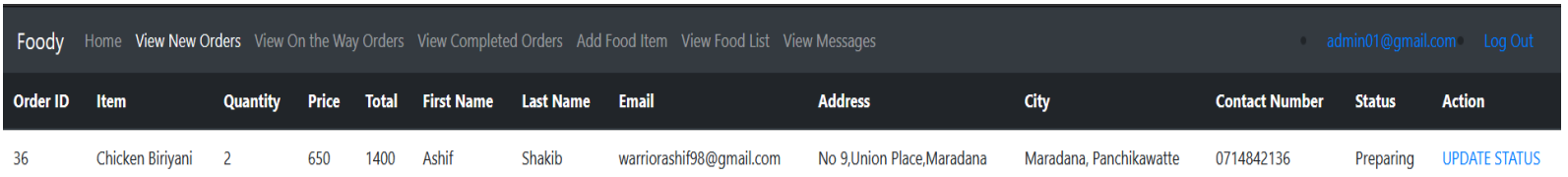
UserContact.html



ViewReviewsUser.html



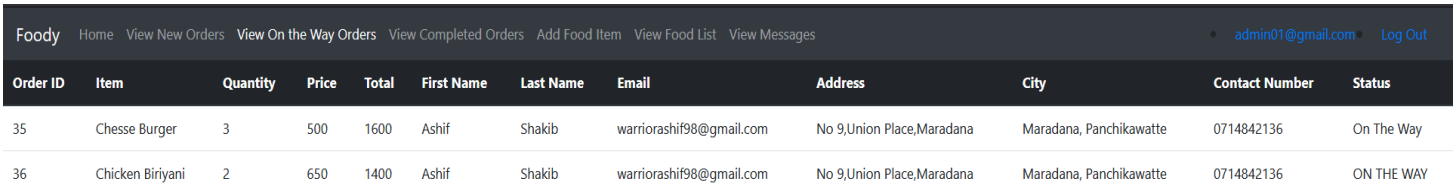
NewOrderAdmin.html



Order ID	Item	Quantity	Price	Total	First Name	Last Name	Email	Address	City	Contact Number	Status	Action
36	Chicken Biriyani	2	650	1400	Ashif	Shakib	warriorashif98@gmail.com	No 9,Union Place,Maradana	Maradana, Panchikawatte	0714842136	Preparing	UPDATE STATUS

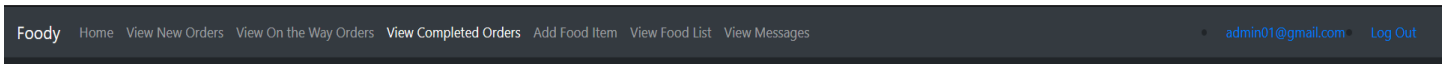
OrderDetailsAdmin.html

OnTheWayOrdersAdmin.html



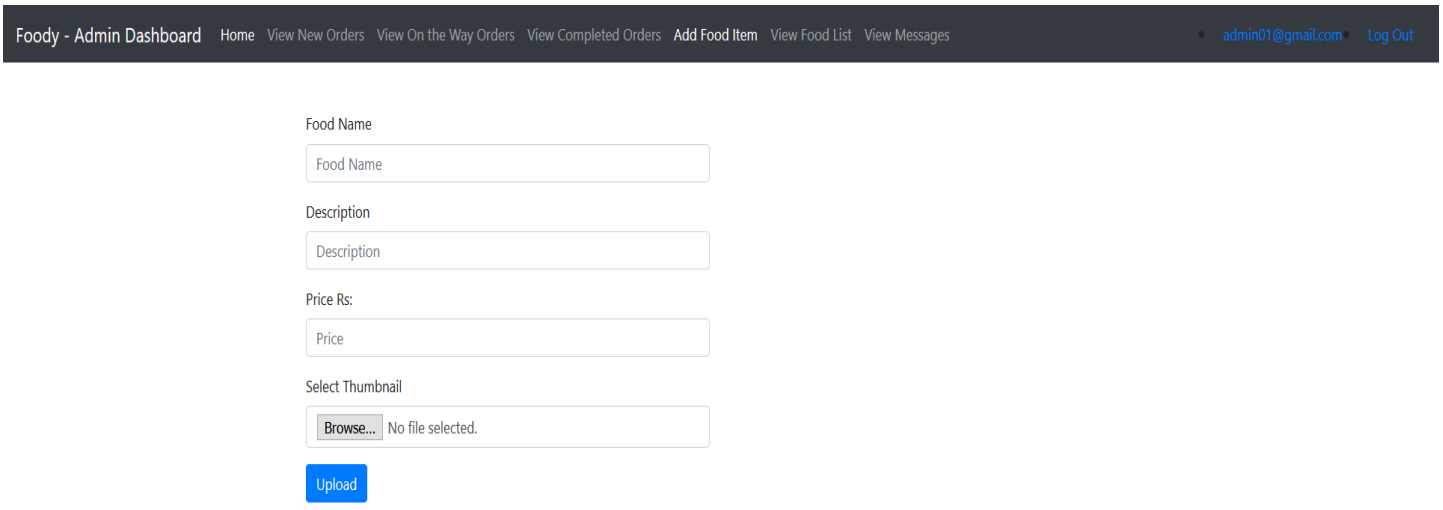
Order ID	Item	Quantity	Price	Total	First Name	Last Name	Email	Address	City	Contact Number	Status
35	Chesse Burger	3	500	1600	Ashif	Shakib	warriorashif98@gmail.com	No 9,Union Place,Maradana	Maradana, Panchikawatte	0714842136	On The Way
36	Chicken Biriyani	2	650	1400	Ashif	Shakib	warriorashif98@gmail.com	No 9,Union Place,Maradana	Maradana, Panchikawatte	0714842136	ON THE WAY

CompletedOrdersAdmin.html

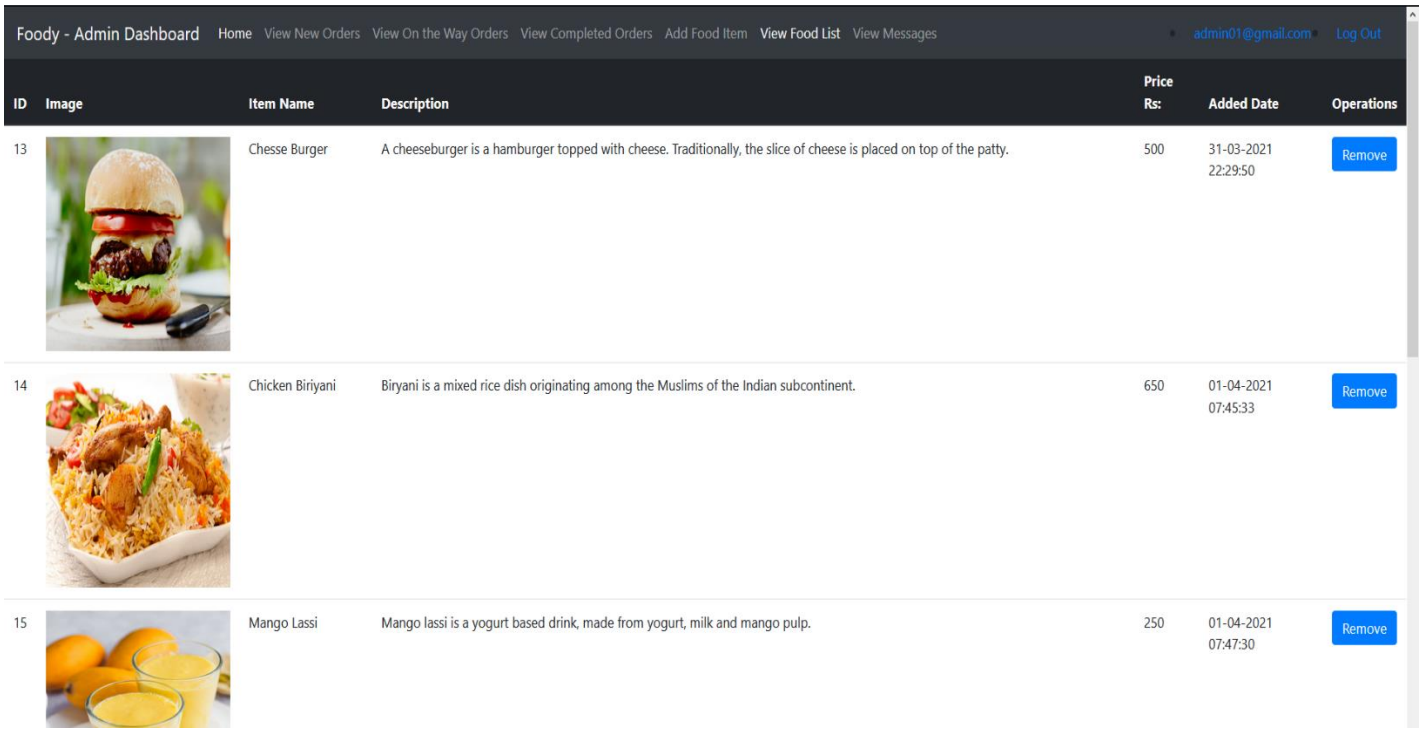


Order ID	Item	Quantity	Price	Total	First Name	Last Name	Email	Address	City	Contact Number	Status	Action
35	Chesse Burger	3	500	1600	Ashif	Shakib	warriorashif98@gmail.com	No 9,Union Place,Maradana	Maradana, Panchikawatte	0714842136	Delivered	
36	Chicken Biriyani	2	650	1400	Ashif	Shakib	warriorashif98@gmail.com	No 9,Union Place,Maradana	Maradana, Panchikawatte	0714842136	Delivered	

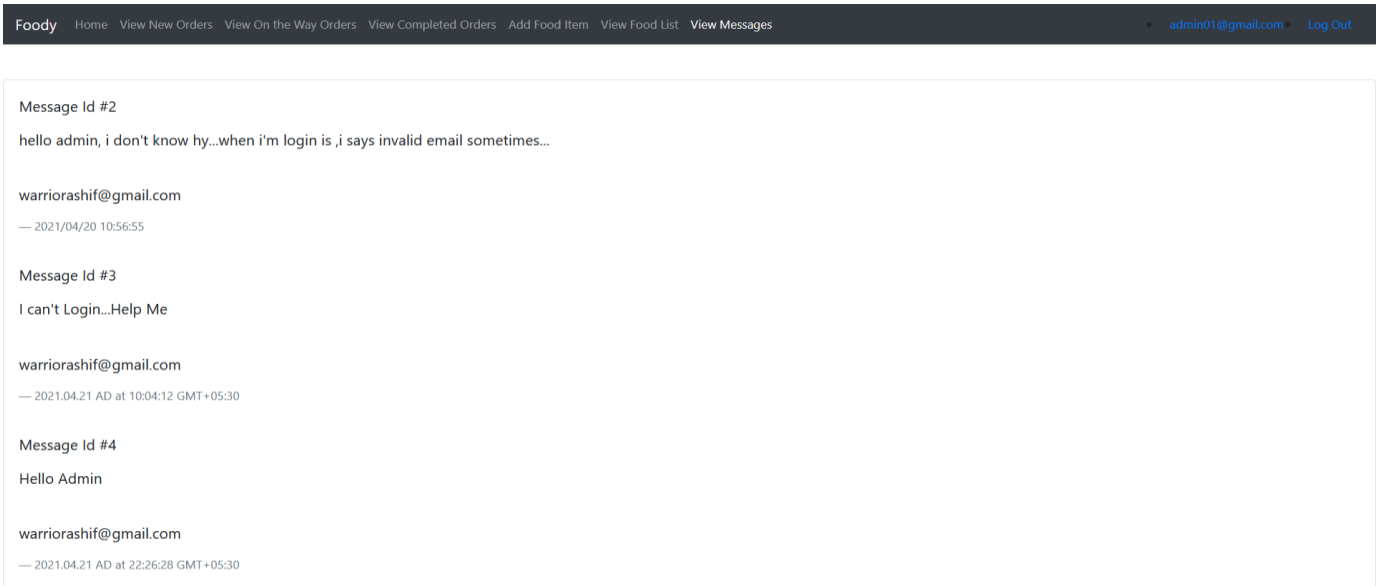
addfood.html



Viewfoodlistadmin.html



ViewMessageAdmin.html



Test Cases

Test ID	Purpose	prerequisite	Steps	Expected Result	Actual Result	Result
001	Selecting Login without entering email and password	System up and Running	01.Run the Foody Web URL 02.Select Login Button	Display an Error message	Displays “Invalid Username or Password “message	Pass
002	Entering invalid Email and Password for Login	System up and Running	01.Run the Foody Web URL 02.Enter invalid email and password. 03.Select Login Button	Display an Error message	Displays “Invalid Username or Password “message	Pass
003	Login as Customer	System up and running	01.Run the Foody Web URL 02.Enter Customer email and password. 03.Select Login Button	Login to Customer home page.	Logged in to Customer Home page.	Pass
004	Login as Admin	System up and running	01.Run the Foody Web URL 02.Enter Admin email and password. 03.Select Login Button	Login to Admin Home page.	Logged into Admin Home Page.	Pass

005	Selecting Register without filling the Register form.	System up and running	01.Run the Foody Web URL 02.Select Register. 03.Select Register Button.	Display empty field error message.	Displayed empty field error message.	Pass
006	Selecting Register with valid details.	System up and running	01.Run the Foody Web URL 02.select Register. 03.Fill the form 04.Select Login Button	Display registration successful message.	Display registration successful message.	Pass
007	View All Reviews for customer.	System up and running	01.Select Read all reviews Button in customer home page.	Display all reviews.	Displayed all the reviews.	Pass
008	View New Orders of Customers	System up and running	01.Select New orders from Navigation Bar in Customer Home Page.	Display orders where status equals to preparing	Displayed all new orders	Pass
009	View On the way Orders of Customer.	System up and running	01.Select on the Way orders from Navigation Bar in Customer Home Page.	Display orders where status equals to on the way.	Displayed all on the way orders.	Pass
010	View Customer Order History	System up and running	01.Select Completed orders from Navigation Bar in Customer Home Page.	Display orders where status equals to Delivered.	Displayed all the completed orders.	Pass

011	View customer Profile.	System up and running	01.Select view Profile from Navigation Bar in Customer Home Page.	Display the profile details of the customer	Displayed the customer profile.	Pass
012	View Add Reviews page	System up and running	01.Select Add Review from Navigation Bar in Customer Home Page.	Display Add review page.	Displayed add review page.	Pass
013	Add review with empty field.	System up and running	01.Select Add Review from Navigation Bar in Customer Home Page. 02.Select add review button with Empty field.	Display empty field error message.	Displayed empty field error message.	Pass
014	Add Reviews without empty field.	System up and running	01.Select Add Review from Navigation Bar in Customer Home Page. 02.Select Add review without empty field.	Display review added successful message.	Displayed review added successful message.	Pass
015	View Contact page	System up and running	01.Select Contact from Navigation Bar in Customer Home Page.	Display contact page.	Displayed contact page.	Pass
016	Add message to admin with empty field.	System up and running	01.Select Contact from Navigation Bar in	Display empty field error message.	Displayed empty field error message.	Pass

			Customer Home Page. 02.Select add contact button with Empty field.			
017	Add message to admin without empty field.	System up and running	01.Select Contact from Navigation Bar in Customer Home Page. 02.Select add contact button without Empty field.	Display message sent successful message.	Displayed message sent successful message.	Pass
018	Customer Change Order Status to Delivered.	System up and running	01.select update status button in on the way orders list. 02.System displays Order details page. 03.select order status to Delivered and update order.	Updates the order status to Delivered.	Updated the order status to Delivered.	Pass
019	Admin Change Order Status to On the Way.	System up and Running	01.select update status button in New Orders list. 02.System displays Order details page. 03.select order status to On the Way and update order.	Updates the order status to On the Way.	Updated the order status to On the Way.	Pass

020	View Menu for Customer	System up and running	01.select View Menu in navigation Bar.	Display Menu for customer.	Displayed Menu for customer.	Pass
021	Admin Delete Food from List	System up and running	01.select food list from navigation bar. 02.Select delete button.	Food item will be removed from the List.	Food item removed from the list.	Pass
022	View food list for admin	System up and running	01.select View food list in navigation Bar.	Displays Food List.	Displayed food list.	Pass
023	Customer Place order with empty fields.	System up and running	01.select Buy now from Menu 02.displays checkout page. 03.select place order	Display empty field error message.	Displayed empty field error message.	Pass
024	Place order with valid details.	System up and running	01.select Buy now from Menu 02.displays checkout page. 03.fill the form with valid details. 04.select place order	Display Order confirmation Page.	Displayed Order confirmation Page.	Pass.

Chapter 02: Mobile Application

Introduction

A well-designed mobile app is much faster than a website at performing tasks. In comparison to Web Applications, which typically use web servers, apps typically store their data locally on mobile devices.

Requirement Analysis

Functional Requirements

Login

Foody mobile app can be used by only customer. Admin has no authority to use the mobile app.

Register

If Customer not registered with foody, customer could register to the System.

View New Order

Customer can view preparing orders.

View on the Way Orders

Customer can View on the way orders.

View order history

Customer can view all the completed orders.

Add Reviews

Customer can add reviews about the foody service.

Place Order

Customer can order food items.

Update Order Status

When order is delivered, Customer can change the status to “Delivered”.

View Reviews

Customer can view all reviews which are posted by all customers.

Contact Admin

Customer can send messages to admin.

Logout

Customer can logout from the system.

Non – functional requirements

- User Friendly GUI – System will have a User friendly and eye-catching Web Application.
- Reliability - Reliability is the degree to which the specified functions are continuously executed by the software system without loss.
- Maintainability - Maintainability is the simplicity with which bugs can be detected and corrected in the software code.
- Usability – user will be a very easy one for use. System will have an instruction page where every step will be displayed.
- Modifiability - Modifiability is the degree to which improvements to a software framework can be developed and executed reliably and cost-effectively.

Hardware requirements

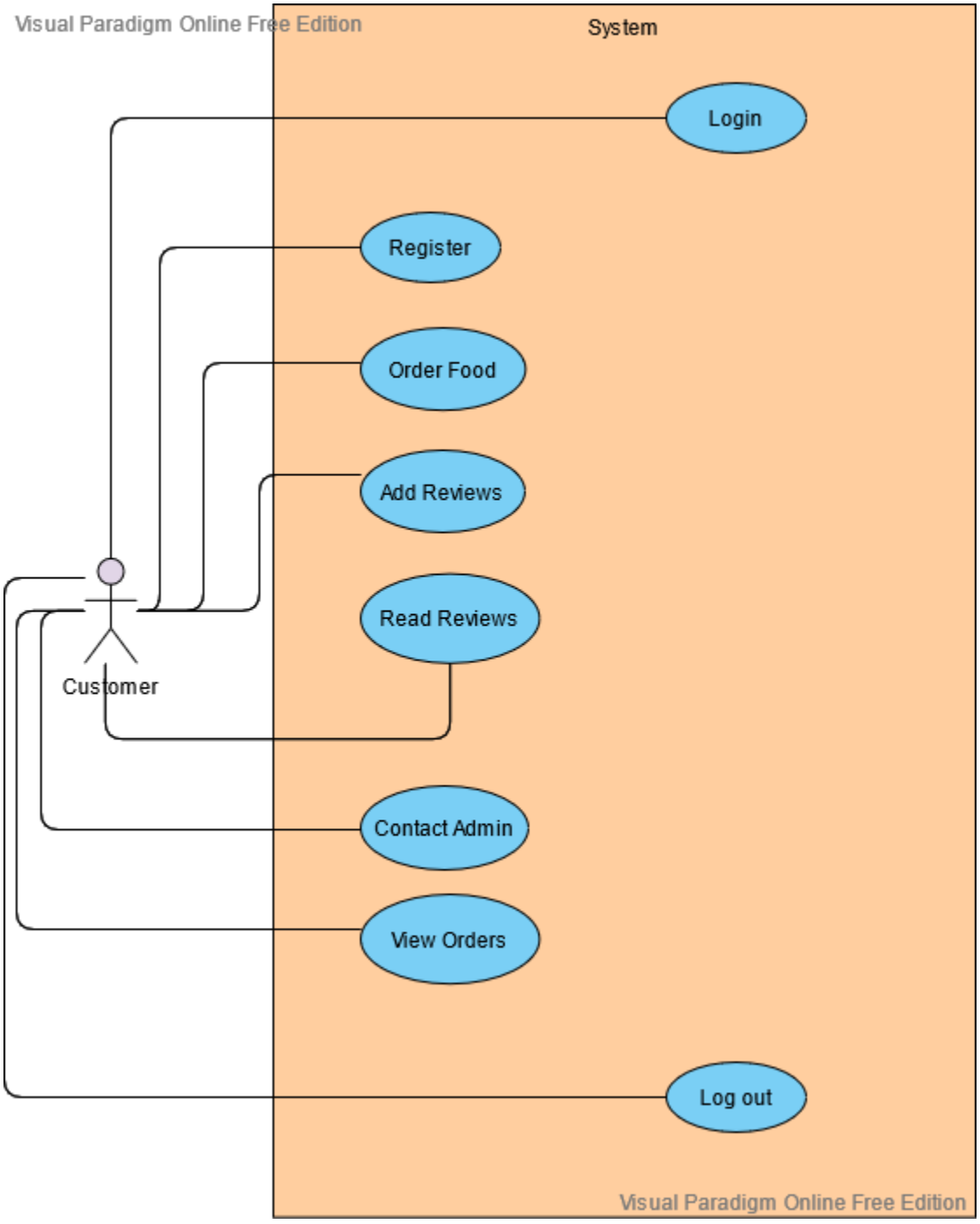
- Device Mechanism - MSI GF63 Thin 95c
- RAM – 8GB
- VGA- 4GB NVIDIA GEFORCE GTX 1650
- Core i5 9th Gen Device Mechanism - MSI GF63 Thin 95c

Software Requirements

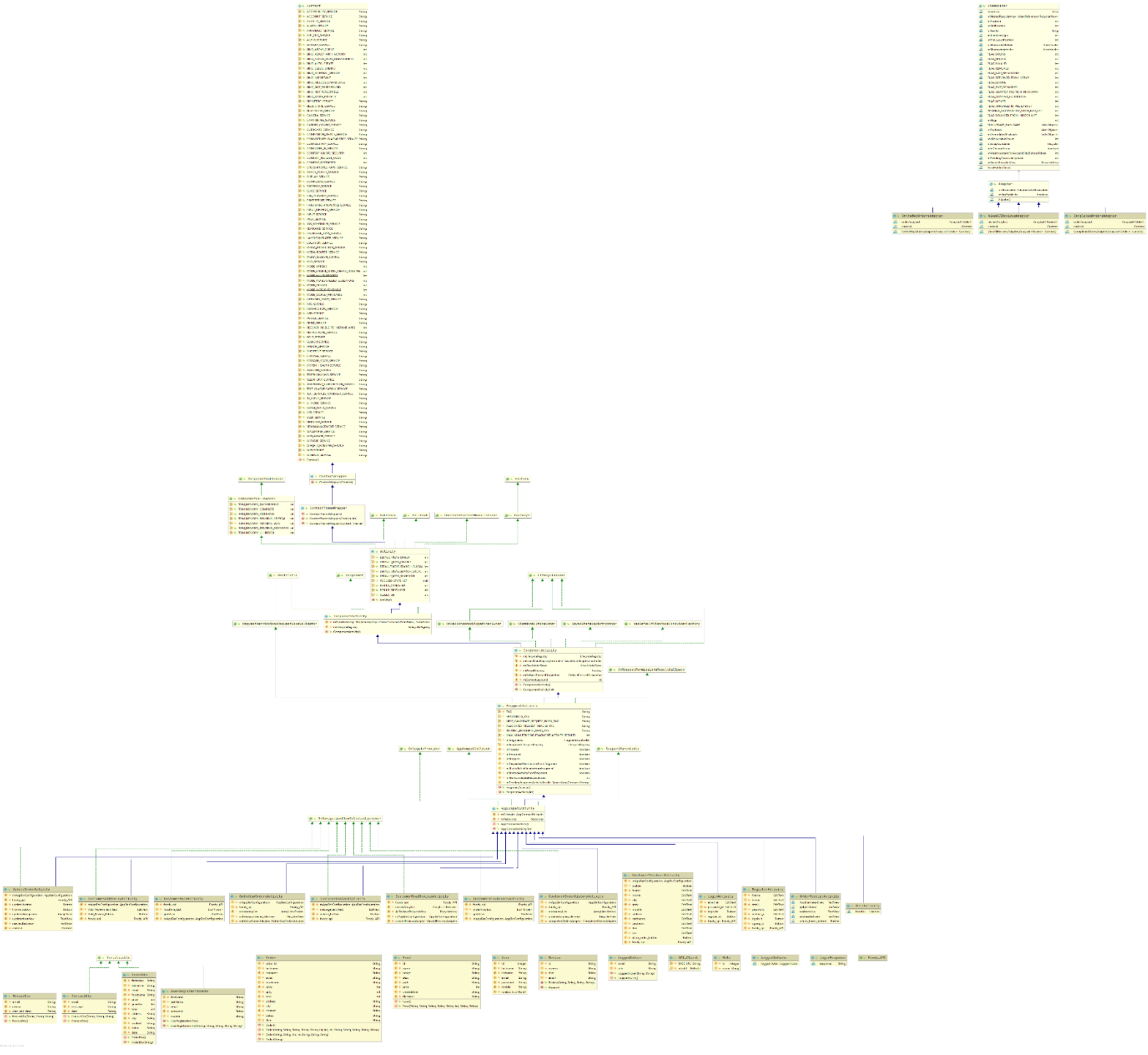
- Operating System – Windows 10
- IntelliJ IDEA
- XAMPP Control Panel
- Android Studio

System Design

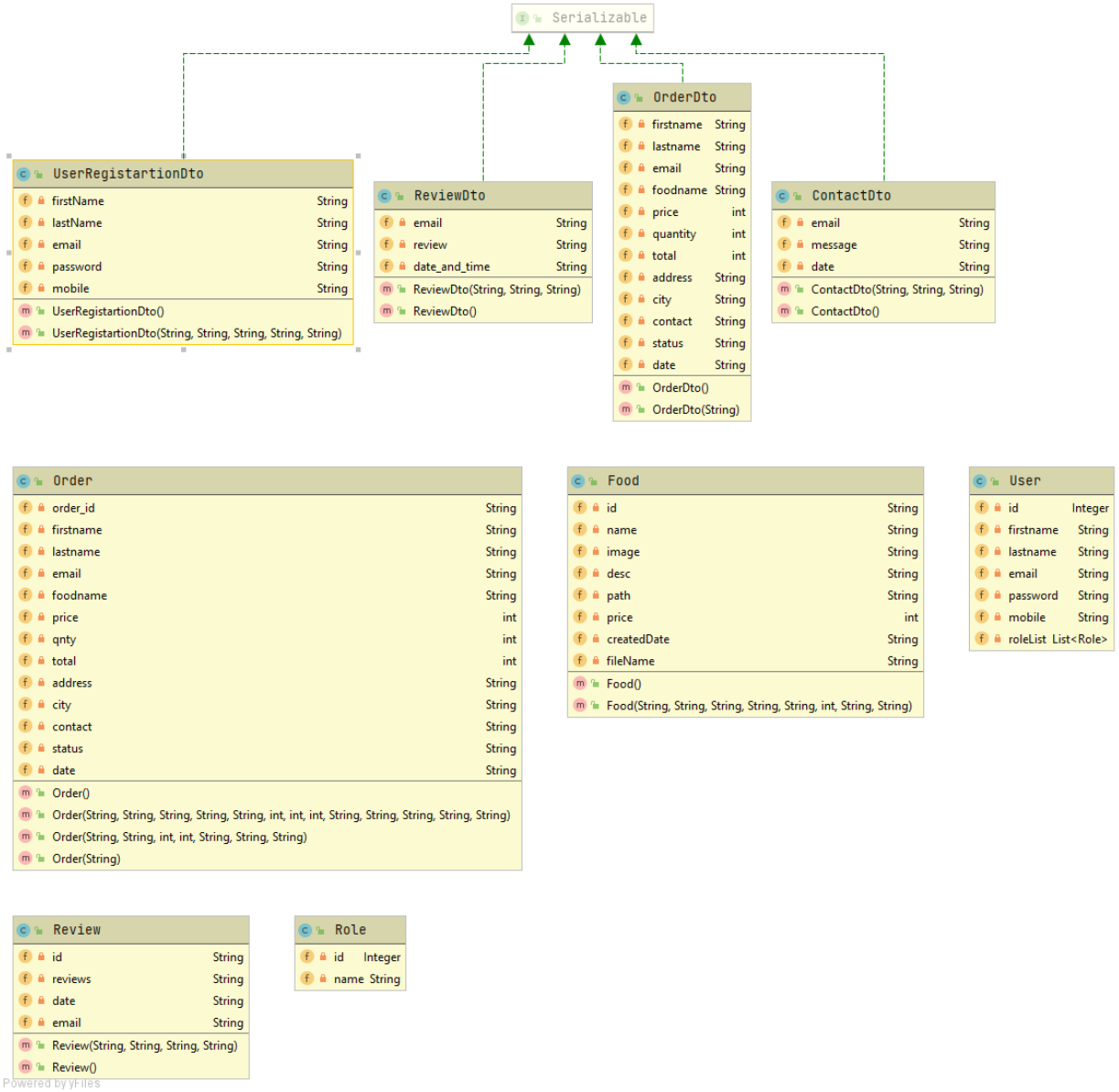
High Level Use Case Diagram



Class Diagram

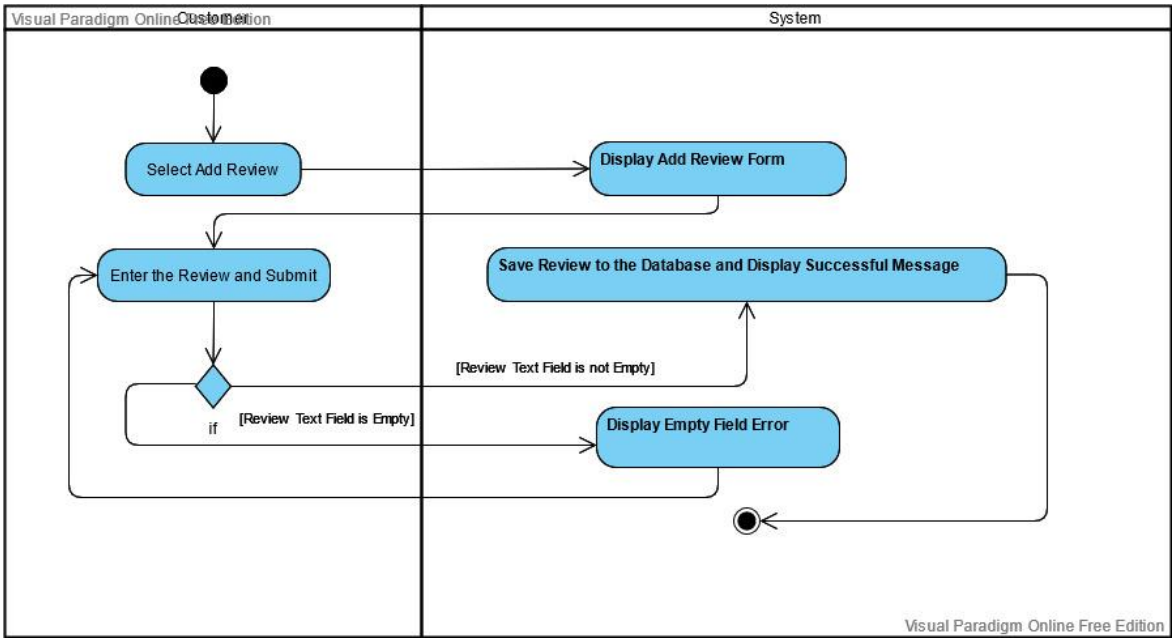


Entity Class Diagram

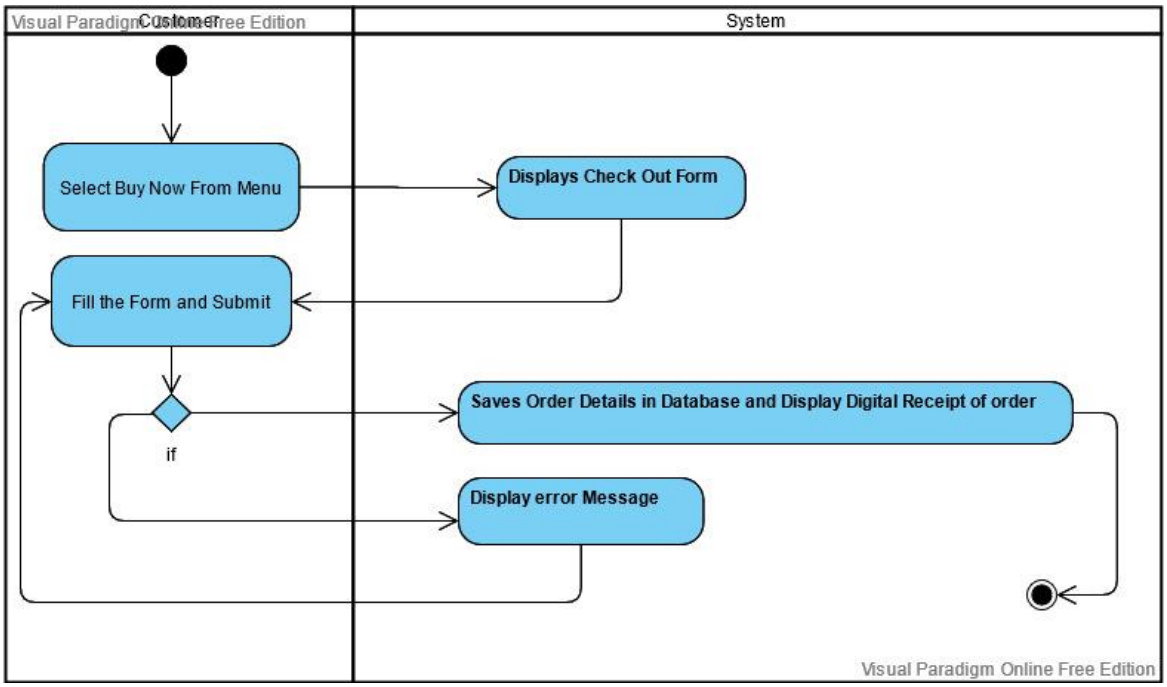


Activity Diagrams

Add Review Activity



Order Activity



Implementation

Retrofit

Retrofit is a type-safe REST client for Android and Java that aims to make RESTful web service consumption easier. Retrofit uses a POJO (Plain Old Java Object) to automatically serialize the JSON response, which must be specified in advance for the JSON Structure. To serialize JSON, the system first needs to convert it to Gson (Bodnar,2020).

```
public class API_Client
{
    private static final String BASE_URL="http://192.168.8.106:8080/api/v1/";
    private static Retrofit retrofit;

    public static Retrofit getRetrofit()
    {
        if(retrofit==null)
        {
            Gson gson=new GsonBuilder()
                    .setLenient()
                    .create();

            retrofit=new Retrofit.Builder()
                    .baseUrl(BASE_URL)
                    .addConverterFactory(GsonConverterFactory.create(gson))
                    .build();
        }
        return retrofit;
    }
}
```

API

API stands for Application Programming Interface. API used for Communication between two systems. In this Case, System uses API to communicate between Web Application and Mobile Application.

The @RestController annotation is useful for creating Restful controllers (Bodnar,2020).

```java
@RestController
@RequestMapping("/api/v1/")
public class API
{
    @Autowired
    private UserRepository userRepository;

    @Autowired
    private OrderRepository orderRepository;

    @Autowired
    private FoodRepository foodRepository;

    @Autowired
    private OrderService orderService;

    @Autowired
    private UserService userService;

    @Autowired
    private ContactService contactService;

    @Autowired
    private ReviewService reviewService;

    @Autowired
    private ReviewRepository reviewRepository;

    @PostMapping("User/Register")
    public JSONObject user_register(@RequestBody UserRegistrationDto userRegistrationDto){
        JSONObject jsonObject=new JSONObject();
        userService.save(userRegistrationDto);

        jsonObject.put("Response","User Saved");
        return jsonObject;
    }
    @GetMapping("login/{username}/{password}")
    public JSONObject loginResponse(@PathVariable(value = "username")String username, @PathVariable(value = "password")String password){
        JSONObject obj = new JSONObject();
        User currentUser = userRepository.findByEmail(username);
        if(currentUser !=null){
            if(userService.passwordCheck(password, currentUser.getPassword())){
                obj.put("Response", "Correct");
            }else {
                obj.put("Response", "Password Incorrect");
            }
        }
        else {
            obj.put("Response", "User does not exist");
        }
        return obj;
    }

    @GetMapping ( "User/foodlist")
    public List<Food> getAllFoods() { return foodRepository.findAll(); }

    @GetMapping("User/vieworders/{email}/{status}")
    public List<Order> getCustomerOrders(@PathVariable(value = "email")String username,@PathVariable(value = "status")String status)
    {
        return orderRepository.findByEmailAndStatus(username,status);
    }
    @GetMapping("User/userdetails/{email}")
    public List<User> getUserDetails(@PathVariable(value = "email") String email)
    {
        return userRepository.findDetailsByEmail(email);
    }

    @PostMapping("User/addcontact")
    public JSONObject user_contact(@RequestBody Contact contact) throws IOException {
        JSONObject jsonObject=new JSONObject();
        contactService.AddReviews(contact);
        jsonObject.put("Response","Message Sent");
        return jsonObject;
    }
    @PostMapping("User/addreviews")
    public JSONObject user_add_review(@RequestBody Reviews reviews) throws IOException {
        JSONObject jsonObject=new JSONObject();
        reviewService.AddReviews(reviews);
        jsonObject.put("Response","Review Added");
        return jsonObject;
    }
    @GetMapping ( "User/reviewlist")
    public List<Reviews> getAllReviews() { return reviewRepository.findAll(); }

    @PostMapping("User/placeorder")
    public JSONObject user_place_order(@RequestBody Order order) throws IOException {
        JSONObject jsonObject=new JSONObject();
        orderService.saveOrder(order);
        jsonObject.put("Response","Order Added");
        return jsonObject;
    }
    @GetMapping("User/onthewayorders/{email}/{status}")
    public List<Order> getOnTheWayCustomerOrders(@PathVariable(value = "email")String username,@PathVariable(value = "status")String status)
    {
        return orderRepository.findByEmailAndStatus(username,status);
    }
    @GetMapping("User/completedorders/{email}/{status}")
    public List<Order> getCompletedCustomerOrders(@PathVariable(value = "email")String username,@PathVariable(value = "status")String status)
    {
        return orderRepository.findByEmailAndStatus(username,status);
    }
    @PostMapping("User/update")
    public JSONObject update_order(@RequestBody Order order) throws IOException {
        JSONObject jsonObject=new JSONObject();
        orderService.saveOrder(order);
        jsonObject.put("Response","Updated");
        return jsonObject;
    }
```

Below figure display the API interface class for mobile Application operations.

```
public interface Foody_API
{
    @POST("User/Register")
    Call<LoginResponse> UserSave(@Body UserRegistartionDto userRegistartionDto);

    @GET("login/{username}/{password}")
    Call<LoginResponse> getLoginResponse(@Path("username")String username, @Path("password")String password);

    @GET("User/foodlist")
    Call<List<Food>> getAllFoods();

    @GET("User/vieworders/{email}/{status}")
    Call<List<Order>> ViewCustomerOrders(@Path("email")String username,@Path("status")String status);

    @POST("User/addcontact")
    Call<Void> ComtactAdmin(@Body ContactDto contactDto);

    @POST("User/addreviews")
    Call<Void> AddReview(@Body ReviewDto reviewDto);

    @GET("User/reviewlist")
    Call<List<Review>> getAllReviews();

    @POST("User/placeorder")
    Call<Void> PlaceOrder(@Body OrderDto order);

    @GET("User/onthewayorders/{email}/{status}")
    Call<List<Order>> ViewOnTheWayOrders(@Path("email")String username,@Path("status")String status);

    @GET("User/completedorders/{email}/{status}")
    Call<List<Order>> ViewCompletedOrders(@Path("email")String username,@Path("status")String status);

    @POST("User/update")
    Call<Void> update_Order(@Body Order order);

}
```

Model/Entity

Instead of using the actual field name, the @SerializedName annotation can be used to serialize a field with a different name. @SerializedName annotation specifies that the annotated member should be serialized to JSON with the name value given as the field name.

The Gson @Expose annotation can be used to indicate whether a field should be exposed when serialized or deserialized.

ContactDto.java

This Entity class uses DTO Design Pattern.

```
public class ContactDto implements Serializable
{
    private String email;
    private String message;
    private String date;

    public ContactDto(String email, String message, String date) {
        this.email = email;
        this.message = message;
        this.date = date;
    }

    public ContactDto() {
    }

    public String getEmail() { return email; }

    public void setEmail(String email) { this.email = email; }

    public String getMessage() { return message; }

    public void setMessage(String message) { this.message = message; }

    public String getDate() { return date; }

    public void setDate(String date) { this.date = date; }
}
```

Food.java

```java
public class Food {
    @SerializedName("id")
    @Expose
    private String id;

    @SerializedName("name")
    @Expose
    private String name;

    @SerializedName("image")
    @Expose
    private String image;

    @SerializedName("designation")
    @Expose
    private String desc;

    @SerializedName("filePath")
    @Expose
    private String path;

    @SerializedName("price")
    @Expose
    private int price;

    @SerializedName("createdDate")
    @Expose
    private String createdDate;

    @SerializedName("fileName")
    @Expose
    private String fileName;

    public Food()
    {

    }
    public Food(String id, String name, String image, String desc, String path, int price, String createdDate, String fileName) {
        this.id = id;
        this.name = name;
        this.image = image;
        this.desc = desc;
        this.path = path;
        this.price = price;
        this.createdDate = createdDate;
        this.fileName = fileName;
    }

    public String getId() { return id; }

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

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public String  getImage() { return image; }

    public void setImage(String image) { this.image = image; }

    public String getDesc() { return desc; }

    public void setDesc(String desc) { this.desc = desc; }

    public String getPath() { return path; }

    public void setPath(String path) { this.path = path; }

    public int getPrice() { return price; }

    public void setPrice(int price) { this.price = price; }

    public String getCreatedDate() { return createdDate; }

    public void setCreatedDate(String createdDate) { this.createdDate = createdDate; }

    public String getFileName() { return fileName; }

    public void setFileName(String fileName) { this.fileName = fileName; }
}
```

Order.java

```java
public class Order
{
    private String order_id;

    private String firstname;

    private String lastname;

    private String email;

    private String foodname;

    private int price;

    @SerializedName("quantity")
    @Expose
    private int qnty;

    private int total;

    private String address;

    private String city;

    private String contact;

    private String status;

    private String date;

    public Order()
    {

    }

    public Order(String order_id, String firstname, String lastname, String email, String foodname, int price, int qnty, int total, String address, String city, String contact, String status, String date) {
        this.order_id = order_id;
        this.firstname = firstname;
        this.lastname = lastname;
        this.email = email;
        this.foodname = foodname;
        this.price = price;
        this.qnty = qnty;
        this.total = total;
        this.address = address;
        this.city = city;
        this.contact = contact;
        this.status = status;
        this.date = date;
    }

    public Order(String order_id, String foodname, int qnty, int total, String address, String status, String date) {
        this.order_id = order_id;
        this.foodname = foodname;
        this.qnty = qnty;
        this.total = total;
        this.address = address;
        this.status = status;
        this.date = date;
    }

    public Order(String email) { this.email = email; }

    public String getOrder_id() { return order_id; }

    public void setOrder_id(String order_id) { this.order_id = order_id; }

    public String getFirstname() { return firstname; }

    public void setFirstname(String firstname) { this.firstname = firstname; }

    public String getLastname() { return lastname; }

    public void setLastname(String lastname) { this.lastname = lastname; }

    public String getEmail() { return email; }

    public void setEmail(String email) { this.email = email; }

    public String getFoodname() { return foodname; }

    public void setFoodname(String foodname) { this.foodname = foodname; }

    public int getPrice() { return price; }

    public void setPrice(int price) { this.price = price; }

    public int getQnty() { return qnty; }

    public void setQnty(int qnty) { this.qnty = qnty; }

    public int getTotal() { return total; }

    public void setTotal(int total) { this.total = total; }

    public String getAddress() { return address; }

    public void setAddress(String address) { this.address = address; }

    public String getCity() { return city; }

    public void setCity(String city) { this.city = city; }

    public String getContact() { return contact; }

    public void setContact(String contact) { this.contact = contact; }

    public String getStatus() { return status; }

    public void setStatus(String status) { this.status = status; }

    public String getDate() { return date; }

    public void setDate(String date) { this.date = date; }
}
```

OrderDto.java

```java
public class OrderDto implements Serializable
{
    private String firstname;

    private String lastname;

    private String email;

    private String foodname;

    private int price;

    private int quantity;

    private int total;

    private String address;

    private String city;

    private String contact;

    private String status;

    private String date;

    public OrderDto()
    {

    }
    public OrderDto(String email) { this.email = email; }

    public String getFirstname() { return firstname; }

    public void setFirstname(String firstname) { this.firstname = firstname; }

    public String getLastname() { return lastname; }

    public void setLastname(String lastname) { this.lastname = lastname; }

    public String getEmail() { return email; }

    public void setEmail(String email) { this.email = email; }

    public String getFoodname() { return foodname; }

    public void setFoodname(String foodname) { this.foodname = foodname; }

    public int getPrice() { return price; }

    public void setPrice(int price) { this.price = price; }

    public int getTotal() { return total; }

    public void setTotal(int total) { this.total = total; }

    public String getAddress() { return address; }

    public void setAddress(String address) { this.address = address; }

    public String getCity() { return city; }

    public void setCity(String city) { this.city = city; }

    public String getContact() { return contact; }

    public void setContact(String contact) { this.contact = contact; }
    public void setContact(String contact) { this.contact = contact; }

    public String getStatus() { return status; }

    public void setStatus(String status) { this.status = status; }

    public String getDate() { return date; }

    public void setDate(String date) { this.date = date; }

    public int getQuantity() { return quantity; }

    public void setQuantity(int quantity) { this.quantity = quantity; }
}
```

Review.java

```java
public class Review
{
    @SerializedName("reviews_ID")
    @Expose
    private String id;

    @SerializedName("review")
    @Expose
    private String reviews;

    @SerializedName("date_and_time")
    @Expose
    private String date;

    @SerializedName("email")
    @Expose
    private String email;

    public Review(String id, String reviews, String date, String email) {
        this.id = id;
        this.reviews = reviews;
        this.date = date;
        this.email = email;
    }

    public Review() {
    }

    public String getId() { return id; }

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

    public String getReviews() { return reviews; }

    public void setReviews(String reviews) { this.reviews = reviews; }

    public String getDate() { return date; }

    public void setDate(String date) { this.date = date; }

    public String getEmail() { return email; }

    public void setEmail(String email) { this.email = email; }
}
```

ReviewDto.java

```java
public class ReviewDto implements Serializable
{
    private String email;
    private String review;
    private String date_and_time;

    public ReviewDto(String email, String review, String date_and_time) {
        this.email = email;
        this.review = review;
        this.date_and_time = date_and_time;
    }

    public ReviewDto() {
    }

    public String getEmail() { return email; }

    public void setEmail(String email) { this.email = email; }

    public String getReview() { return review; }

    public void setReview(String review) { this.review = review; }

    public String getDate_and_time() { return date_and_time; }

    public void setDate_and_time(String date_and_time) { this.date_and_time = date_and_time; }
}
```

Role.java

```java
public class Role
{
    @SerializedName("id")
    @Expose
    private Integer id;
    @SerializedName("name")
    @Expose
    private String name;

    public Integer getId() { return id; }

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

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }
}
```

User.java

```java
public class User
{
    @SerializedName("id")
    @Expose
    private Integer id;
    @SerializedName("firstName")
    @Expose
    private String firstname;
    @SerializedName("lastName")
    @Expose
    private String lastname;
    @SerializedName("email")
    @Expose
    private String email;
    @SerializedName("password")
    @Expose
    private String  password;
    @SerializedName("mobile")
    @Expose
    private String mobile;
    @SerializedName("roles")
    @Expose
    private List<Role> roleList=null;

    public Integer getId() { return id; }

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

    public String getFirstname() { return firstname; }

    public void setFirstname(String firstname) { this.firstname = firstname; }

    public String getLastname() { return lastname; }

    public void setLastname(String lastname) { this.lastname = lastname; }

    public String getEmail() { return email; }

    public void setEmail(String email) { this.email = email; }

    public String getPassword() { return password; }

    public void setEmail(String email) { this.email = email; }

    public String getPassword() { return password; }

    public void setPassword(String password) { this.password = password; }

    public String getMobile() { return mobile; }

    public void setMobile(String mobile) { this.mobile = mobile; }

    public List<Role> getRoleList() { return roleList; }

    public void setRoleList(List<Role> roleList) { this.roleList = roleList; }
}
```

UserRegistrationDto.java

```java
public class UserRegistartionDto implements Serializable
{
    private String firstName;
    private String lastName;
    private String email;
    private String  password;
    private String mobile;

    public UserRegistartionDto() {
    }

    public UserRegistartionDto(String firstName, String lastName, String email, String password, String mobile) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
        this.password = password;
        this.mobile = mobile;
    }

    public String getFirstName() { return firstName; }

    public void setFirstName(String firstName) { this.firstName = firstName; }

    public String getLastName() { return lastName; }

    public void setLastName(String lastName) { this.lastName = lastName; }

    public String getEmail() { return email; }

    public void setEmail(String email) { this.email = email; }

    public String getPassword() { return password; }

    public void setPassword(String password) { this.password = password; }

    public String getMobile() { return mobile; }

    public void setMobile(String mobile) { this.mobile = mobile; }
}
```

Activities

MainActivity.java

```java
public class MainActivity extends AppCompatActivity
{
    Handler handler;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        if (getSupportActionBar() != null) {
            getSupportActionBar().hide();
        }
        handler=new Handler();
        handler.postDelayed(() -> {
                Intent intent=new Intent( packageContext: MainActivity.this,LoginActivity.class);
                startActivity(intent);
                finish();
        }, delayMillis: 3000);

    }
}
```

LoginActivity.java

```java
public class LoginActivity extends AppCompatActivity
{
    private EditText email_et,password_et;
    private Button login_bt,register_bt;
    private Foody_API foody_api;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        if (getSupportActionBar() != null) {
            getSupportActionBar().hide();
        }

        email_et=(EditText)findViewById(R.id.login_email);
        password_et=(EditText)findViewById(R.id.login_password);

        login_bt=(Button)findViewById(R.id.login_button);
        login_bt.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view)
            {
                if(email_et.getText().toString().isEmpty())
                {
                    Toast.makeText( context: LoginActivity.this,  text: "No Email Detected", Toast.LENGTH_SHORT).show();
                }
                else if(password_et.getText().toString().isEmpty())
                {
                    Toast.makeText( context: LoginActivity.this,  text: "No Password Detected", Toast.LENGTH_SHORT).show();
                }
                else {
                    Login();
                }
            }
        });
        register_bt=(Button)findViewById(R.id.register_button_login);
        register_bt.setOnClickListener((view) -> {
                Intent intent=new Intent( packageContext: LoginActivity.this,RegisterActivity.class);
                startActivity(intent);
        });
    private void Login()
    {
        foody_api=API_Client.getRetrofit().create(Foody_API.class);

        final String email=email_et.getText().toString();
        String password=password_et.getText().toString();

        Call<LoginResponse> loginResponseCall=foody_api.getLoginResponse(email,password);
        loginResponseCall.enqueue(new Callback<LoginResponse>() {
            @Override
            public void onResponse(Call<LoginResponse> call, Response<LoginResponse> response)
            {
                LoginResponse loginResponse=response.body();
                if(loginResponse.getResponse().matches( regex: "Correct"))
                {
                    LoggedInCache.LoggedInUser=new LoggedInUser(email,  role: "");
                    Toast.makeText( context: LoginActivity.this,  text: "login success", Toast.LENGTH_SHORT).show();
                    Intent myIntent = new Intent( packageContext: LoginActivity.this, CustomerHomeActivity.class);
                    myIntent.putExtra( name: "email",email);
                    LoginActivity.this.startActivity(myIntent);
                }
                else if (loginResponse.getResponse().matches( regex: "Password Incorrect"))
                {
                    Toast.makeText( context: LoginActivity.this,  text: "Password Incorrect", Toast.LENGTH_SHORT).show();
                }
                else if(loginResponse.getResponse().matches( regex: "User does not exist"))
                {
                    Toast.makeText( context: LoginActivity.this,  text: "User Does not exist", Toast.LENGTH_SHORT).show();
                }
            }

            @Override
            public void onFailure(Call<LoginResponse> call, Throwable t)
            {
                Toast.makeText( context: LoginActivity.this,  text: "Login Error: "+t, Toast.LENGTH_SHORT).show();

            }
        });
    }
}
```

RegisterActivity.java

```java
public class RegisterActivity extends AppCompatActivity
{
    private EditText fname,lname,email_,password_,contact_n;
    private Button signin_b,signup_b;
    private Foody_API foody_api;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);

        if (getSupportActionBar() != null) {
            getSupportActionBar().hide();
        }

        fname=findViewById(R.id.firstname);
        lname=findViewById(R.id.lastname);
        email_=findViewById(R.id.email);
        password_=findViewById(R.id.password);
        contact_n=findViewById(R.id.mobile);

        signup_b=findViewById(R.id.signup_reg);
        signup_b.setOnClickListener((view) -> {

                UserRegistartionDto userRegistartionDto= new UserRegistartionDto();
                userRegistartionDto.setFirstName(fname.getText().toString());
                userRegistartionDto.setLastName(lname.getText().toString());
                userRegistartionDto.setEmail(email_.getText().toString());
                userRegistartionDto.setPassword(password_.getText().toString());
                userRegistartionDto.setMobile(contact_n.getText().toString());

                foody_api= API_Client.getRetrofit().create(Foody_API.class);
                Call<LoginResponse> call = foody_api.UserSave(userRegistartionDto);
                call.enqueue(new Callback<LoginResponse>()
                {
                    @Override
                    public void onResponse(Call<LoginResponse> call, Response<LoginResponse> response) {
                        Toast.makeText( context: RegisterActivity.this,  text: "Success", Toast.LENGTH_SHORT).show();
                    }

                    @Override
                    public void onFailure(Call<LoginResponse> call, Throwable t) {
                        Toast.makeText( context: RegisterActivity.this,t.getMessage(),Toast.LENGTH_SHORT).show();
                    }
                });
        });

        signin_b=findViewById(R.id.signin_reg);
        signin_b.setOnClickListener((view) -> {
                Intent intent=new Intent( packageContext: RegisterActivity.this,LoginActivity.class);
                startActivity(intent);
        });


    }
}
```

CustomerHomeActivity.java

```
public class CustomerHomeActivity extends AppCompatActivity implements NavigationView.OnNavigationItemSelectedListener
{
    private Foody_API foody_api;
    private List<Food> foodArrayList=new ArrayList<>();
    GridView gridView;
    private AppBarConfiguration mAppBarConfiguration;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_customer_home);

        gridView=findViewById(R.id.customer_menu_gridview);
        getFoodList();

        Intent receive_email= getIntent();
        String email=receive_email.getStringExtra( name: "email");

        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        DrawerLayout drawer = findViewById(R.id.drawer_layout);
        NavigationView navigationView = findViewById(R.id.nav_view);
        drawer = findViewById(R.id.drawer_layout);
        navigationView.setNavigationItemSelectedListener(this);

        TextView txtProfileName = (TextView) navigationView.getHeaderView( index: 0).findViewById(R.id.textView_cus_home);
        txtProfileName.setText(email);

        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle( activity: this, drawer, toolbar, "Open navigation drawer", "Close navigation drawer");
        drawer.addDrawerListener(toggle);
        toggle.syncState();
    }
```

```
    private void getFoodList()
    {
        foody_api= API_Client.getRetrofit().create(Foody_API.class);
        Call<List<Food>> call=foody_api.getAllFoods();
        call.enqueue(new Callback<List<Food>>() {
            @Override
            public void onResponse(Call<List<Food>> call, Response<List<Food>> response)
            {
                if(response.isSuccessful())
                {
                    String message="successful";
                    Toast.makeText( context: CustomerHomeActivity.this,message,Toast.LENGTH_SHORT).show();
                    foodArrayList=response.body();
                    CustomerHomeActivity.FoodViewAdapter foodViewAdapter=new CustomerHomeActivity.FoodViewAdapter(foodArrayList, context: CustomerHomeActivity.this);
                    gridView.setAdapter(foodViewAdapter);
                }
            }

            @Override
            public void onFailure(Call<List<Food>> call, Throwable t)
            {
                Toast.makeText( context: CustomerHomeActivity.this, text: "Inquiry List Failed",Toast.LENGTH_SHORT).show();

            }
        });
    }
    public class FoodViewAdapter extends BaseAdapter
    {
        private List<Food> foodList;
        private Context context;
        private LayoutInflater layoutInflater;

        public FoodViewAdapter(List<Food> foodList, Context context) {
            this.foodList = foodList;
            this.context = context;
            this.layoutInflater = (LayoutInflater) context.getSystemService(LAYOUT_INFLATER_SERVICE);
        }

        @Override
        public int getCount() { return foodList.size(); }

        @Override
        public Object getItem(int i) { return null; }
```

```java
    @Override
    public long getItemId(int i) { return 0; }

    @Override
    public View getView(int i, View view, ViewGroup viewGroup)
    {
        if(view==null)
        {
            view=layoutInflater.inflate(R.layout.list_item,viewGroup, attachToRoot: false);

        }
        ImageView imageView=view.findViewById(R.id.image_view_home);
        final TextView textView=view.findViewById(R.id.food_name);
        final TextView priceview=view.findViewById(R.id.food_price);
        final TextView desc=view.findViewById(R.id.desc_food);

        Intent receive_email= getIntent();
        final String email=receive_email.getStringExtra( name: "email");

        textView.setText(foodList.get(i).getName());
        priceview.setText("Rs."+foodList.get(i).getPrice()+".00");
        final int price=foodList.get(i).getPrice();
        desc.setText(foodList.get(i).getDesc());

        Picasso.Builder builder=new Picasso.Builder(context);
        builder.downloader(new OkHttp3Downloader(context));
        builder.build().load(foodList.get(i).getImage()).into(imageView);

        Button buy_now_button=view.findViewById(R.id.buy_now_button);
        buy_now_button.setOnClickListener((view) -> {
                Intent intent=new Intent(context,CustomerCheckoutActivity.class);
                intent.putExtra( name: "foodname",textView.getText().toString());
                intent.putExtra( name: "foodprice",priceview.getText().toString());
                intent.putExtra( name: "price",price);
                intent.putExtra( name: "desc",desc.getText().toString());
                intent.putExtra( name: "email",email);
                context.startActivity(intent);


        });
        return view;
    }
}
```

CustomerCheckoutActivity.java

```
public class CustomerCheckoutActivity extends AppCompatActivity implements NavigationView.OnNavigationItemSelectedListener {

    private AppBarConfiguration mAppBarConfiguration;
    AlertDialog.Builder builder;
    private EditText fname,lname,city,qnty,mobile,address,cardname,cardnum,doe,cvv;
    private Button place_order_button;
    private Foody_API foody_api;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_customer_checkout);

        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        DrawerLayout drawer = findViewById(R.id.drawer_layout);
        NavigationView navigationView = findViewById(R.id.nav_view);
        drawer = findViewById(R.id.drawer_layout);
        navigationView.setNavigationItemSelectedListener(this);

        Intent receive_email= getIntent();
        final String email=receive_email.getStringExtra( name: "email");

        TextView txtProfileName = (TextView) navigationView.getHeaderView( index: 0).findViewById(R.id.textView_checkout);
        txtProfileName.setText(email);

        Intent get_foodd_details= getIntent();
        final String food_name=get_foodd_details.getStringExtra( name: "foodname");
        final String food_price=get_foodd_details.getStringExtra( name: "foodprice");
        String food_desc=get_foodd_details.getStringExtra( name: "desc");
        final int price=get_foodd_details.getIntExtra( name: "price", defaultValue: 0);

        TextView foodname_view=(TextView)findViewById(R.id.food_name_checkout);
        TextView fooddesc_view=(TextView)findViewById(R.id.food_desc_checkout);
        TextView foodprice_view=(TextView)findViewById(R.id.food_price_checkout);
        foodname_view.setText(food_name);
        fooddesc_view.setText(food_desc);
        foodprice_view.setText(food_price);

        place_order_button=(Button)findViewById(R.id.place_order_btn);
        fname=(EditText)findViewById(R.id.user_fname_edit_text);
        lname=(EditText)findViewById(R.id.user_lname_edit_text);
        city=(EditText)findViewById(R.id.user_city_edit_text);
        address=(EditText)findViewById(R.id.user_address_edit_text);
        mobile=(EditText)findViewById(R.id.user_mobile_edit_text);
        qnty=(EditText)findViewById(R.id.user_qnty_edit_text);
        cardname=(EditText)findViewById(R.id.user_cardname_edit_text);
        cardnum=(EditText)findViewById(R.id.user_cardnumber_edit_text);
        doe=(EditText)findViewById(R.id.user_doe_edit_text);
        cvv=(EditText)findViewById(R.id.user_cvv_edit_text);

        SimpleDateFormat sdf = new SimpleDateFormat( pattern: "yyyy.MM.dd G 'at' HH:mm:ss z");
        final String currentDateandTime = sdf.format(new Date());

        place_order_button.setOnClickListener((view) -> {
                if(cardname.getText().toString().isEmpty() || cardnum.getText().toString().isEmpty() || doe.getText().toString().isEmpty() ||
                        cvv.getText().toString().isEmpty() || fname.getText().toString().isEmpty() || lname.getText().toString().isEmpty() ||
                city.getText().toString().isEmpty() || address.getText().toString().isEmpty() || mobile.getText().toString().isEmpty() || qnty.getText().toString().isEmpty())
                {
                    Toast.makeText( context: CustomerCheckoutActivity.this, text: "Empty Fields Detected", Toast.LENGTH_SHORT).show();
                }
                else if(cardnum.length()<16 || doe.length()<5 || cvv.length()<3)
                {
                    Toast.makeText( context: CustomerCheckoutActivity.this, text: "No Valid Card Details Detected", Toast.LENGTH_SHORT).show();
                }
                else
                {
                    OrderDto order= new OrderDto();
                    order.setFirstname(fname.getText().toString());
                    order.setLastname(fname.getText().toString());
                    order.setEmail(email);
                    order.setFoodname(food_name);
                    order.setAddress(address.getText().toString());
                    order.setCity(city.getText().toString());
                    order.setDate(currentDateandTime);
                    order.setStatus("Preparing");
                    order.setContact(mobile.getText().toString());

                    final int food_qnty=Integer.parseInt(qnty.getText().toString());
                    order.setQuantity(food_qnty);
```

```
            final int Total= (price*food_qnty)+100;
            order.setTotal(Total);
            order.setPrice(price);

            foody_api= API_Client.getRetrofit().create(Foody_API.class);
            Call<Void> call =foody_api.PlaceOrder(order);
            call.enqueue(new Callback<Void>() {
                @Override
                public void onResponse(Call<Void> call, Response<Void> response)
                {
                    Toast.makeText( context: CustomerCheckoutActivity.this,  text: "Order Placed", Toast.LENGTH_SHORT).show();
                    Intent intent= new Intent( packageContext: CustomerCheckoutActivity.this,OrderReceiptActivity.class);
                    intent.putExtra( name: "foodname",food_name);
                    intent.putExtra( name: "qnty",food_qnty);
                    intent.putExtra( name: "total",Total);
                    intent.putExtra( name: "address",address.getText().toString());
                    startActivity(intent);
                }

                @Override
                public void onFailure(Call<Void> call, Throwable t)
                {
                    Toast.makeText( context: CustomerCheckoutActivity.this,  text: "Something went wrong", Toast.LENGTH_SHORT).show();

                }
            });
        }
    });

    ActionBarDrawerToggle toggle = new ActionBarDrawerToggle( activity: this, drawer, toolbar, "Open navigation drawer", "Close navigation drawer");
    drawer.addDrawerListener(toggle);
    toggle.syncState();
}
```

CustomerContactActivity.java

```
public class CustomerContactActivity extends AppCompatActivity implements NavigationView.OnNavigationItemSelectedListener{

    private AppBarConfiguration mAppBarConfiguration;
    private EditText message_edit_text;
    private Button submit_button;
    private Foody_API foody_api;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_customer_contact);


        Intent receive_email= getIntent();
        final String email=receive_email.getStringExtra( name: "email");

        message_edit_text=findViewById(R.id.user_contact_edit_text);
        submit_button=findViewById(R.id.contact_submit_button);


        SimpleDateFormat sdf = new SimpleDateFormat( pattern: "yyyy.MM.dd G 'at' HH:mm:ss z");
        final String currentDateandTime = sdf.format(new Date());

        submit_button.setOnClickListener((view) -> {
                ContactDto contactDto=new ContactDto();
                contactDto.setMessage(message_edit_text.getText().toString());
                contactDto.setEmail(email);
                contactDto.setDate(currentDateandTime);
                foody_api= API_Client.getRetrofit().create(Foody_API.class);
                Call<Void> call =foody_api.ComtactAdmin(contactDto);
                call.enqueue(new Callback<Void>() {
                    @Override
                    public void onResponse(Call<Void> call, Response<Void> response) {
                        Toast.makeText( context: CustomerContactActivity.this,  text: "Success", Toast.LENGTH_SHORT).show();
                    }
                    @Override
                    public void onFailure(Call<Void> call, Throwable t)
                    {
                        Toast.makeText( context: CustomerContactActivity.this,  text: "Something went Wrong", Toast.LENGTH_SHORT).show();
                    }
                });

        });
```

CustomerOrderHistory.java

```
public class CustomerOrderHistoryActivity extends AppCompatActivity implements NavigationView.OnNavigationItemSelectedListener {

    private AppBarConfiguration mAppBarConfiguration;
    private Foody_API foody_api;
    ArrayList<Order> orderArrayList=new ArrayList<>();
    private RecyclerView orderhistoryrecyclerview;
    private CompletedOrdersAdapter completedOrdersAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_customer_order_history);
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        DrawerLayout drawer = findViewById(R.id.drawer_layout);
        NavigationView navigationView = findViewById(R.id.nav_view);
        drawer = findViewById(R.id.drawer_layout);
        navigationView.setNavigationItemSelectedListener(this);

        orderhistoryrecyclerview=findViewById(R.id.customer_completed_orders_view);
        orderhistoryrecyclerview.setLayoutManager(new LinearLayoutManager( context: this));

        Intent intent=getIntent();
        String email=intent.getStringExtra( name: "email");
        String status= "Delivered";

        TextView txtProfileName = (TextView) navigationView.getHeaderView( index: 0).findViewById(R.id.textView_order_history);
        txtProfileName.setText(email);

        foody_api= API_Client.getRetrofit().create(Foody_API.class);
        Call<List<Order>> call=foody_api.ViewCompletedOrders(email,status);
        call.enqueue(new Callback<List<Order>>() {
            @Override
            public void onResponse(Call<List<Order>> call, Response<List<Order>> response)
            {
                if(response.isSuccessful())
                {
                    String message="successful";
                    Toast.makeText( context: CustomerOrderHistoryActivity.this,message,Toast.LENGTH_SHORT).show();
                    orderArrayList= new ArrayList<>(response.body());
                    completedOrdersAdapter=new CompletedOrdersAdapter(orderArrayList, context: CustomerOrderHistoryActivity.this);
                    orderhistoryrecyclerview.setAdapter(completedOrdersAdapter);
                }
            }

            @Override
            public void onFailure(Call<List<Order>> call, Throwable t)
            {
                Toast.makeText( context: CustomerOrderHistoryActivity.this, text: "Order List Failed",Toast.LENGTH_SHORT).show();

            }
        });

        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle( activity: this, drawer, toolbar, "Open navigation drawer", "Close navigation drawer");
        drawer.addDrawerListener(toggle);
        toggle.syncState();
    }
```

CustomerReadReviewsActivity.java

```
public class CustomerReadReviewsActivity extends AppCompatActivity implements NavigationView.OnNavigationItemSelectedListener
{
    private Foody_API foody_api;
    ArrayList<Review> reviewArrayList=new ArrayList<>();
    private RecyclerView AllReviewsRecyclerView;
    private AppBarConfiguration mAppBarConfiguration;
    private ViewAllReviewsAdapter viewAllReviewsAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_customer_read_reviews);

       AllReviewsRecyclerView=findViewById(R.id.customer_read_review_view);
       AllReviewsRecyclerView.setLayoutManager(new LinearLayoutManager( context: this));

       getReviewList();

       Intent receive_email= getIntent();
       String email=receive_email.getStringExtra( name: "email");

       Toolbar toolbar = findViewById(R.id.toolbar);
       setSupportActionBar(toolbar);
       DrawerLayout drawer = findViewById(R.id.drawer_layout);
       NavigationView navigationView = findViewById(R.id.nav_view);
       drawer = findViewById(R.id.drawer_layout);
       navigationView.setNavigationItemSelectedListener(this);

        TextView txtProfileName = (TextView) navigationView.getHeaderView( index: 0).findViewById(R.id.textView_read_reviews);
        txtProfileName.setText(email);

        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle( activity: this, drawer, toolbar, "Open navigation drawer", "Close navigation drawer");
        drawer.addDrawerListener(toggle);
        toggle.syncState();
    }

    private void getReviewList()
    {
        foody_api= API_Client.getRetrofit().create(Foody_API.class);
        Call<List<Review>> call=foody_api.getAllReviews();
        call.enqueue(new Callback<List<Review>>() {
            @Override
            public void onResponse(Call<List<Review>> call, Response<List<Review>> response)
            {
                if(response.isSuccessful())
                {
                    String message="successful";
                    Toast.makeText( context: CustomerReadReviewsActivity.this,message,Toast.LENGTH_SHORT).show();
                    reviewArrayList= new ArrayList<>(response.body());
                    viewAllReviewsAdapter=new ViewAllReviewsAdapter(reviewArrayList, context: CustomerReadReviewsActivity.this);
                    AllReviewsRecyclerView.setAdapter(viewAllReviewsAdapter);
                }
            }

            @Override
            public void onFailure(Call<List<Review>> call, Throwable t)
            {
                Toast.makeText( context: CustomerReadReviewsActivity.this, text: "List Loading Failed",Toast.LENGTH_SHORT).show();

            }
        });
    }
```

CustomerViewOrdersActivity.java

```java
public class CustomerViewOrdersActivity extends AppCompatActivity implements NavigationView.OnNavigationItemSelectedListener
{
    private Foody_API foody_api;
    private List<Order> orderArrayList=new ArrayList<>();
    GridView gridView;
    private AppBarConfiguration mAppBarConfiguration;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_customer_view_orders);

        gridView=findViewById(R.id.customer_order_gridview);
       // getOrdersByEmail();

        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        DrawerLayout drawer = findViewById(R.id.drawer_layout);
        NavigationView navigationView = findViewById(R.id.nav_view);
        drawer = findViewById(R.id.drawer_layout);
        navigationView.setNavigationItemSelectedListener(this);

        Intent intent=getIntent();
        String email=intent.getStringExtra( name: "email");
        String status= "Preparing";

        foody_api= API_Client.getRetrofit().create(Foody_API.class);
        Call<List<Order>> call=foody_api.ViewCustomerOrders(email,status);
        call.enqueue(new Callback<List<Order>>() {
            @Override
            public void onResponse(Call<List<Order>> call, Response<List<Order>> response)
            {
                if(response.isSuccessful())
                {
                    String message="successful";
                    Toast.makeText( context: CustomerViewOrdersActivity.this,message,Toast.LENGTH_SHORT).show();

                    orderArrayList=response.body();
                    CustomerViewOrdersActivity.OrderViewAdapter orderViewAdapter=new CustomerViewOrdersActivity.OrderViewAdapter(orderArrayList, context: CustomerViewOrdersActivity.this);
                    gridView.setAdapter(orderViewAdapter);
                }
            }

            @Override
            public void onFailure(Call<List<Order>> call, Throwable t)
            {
                Toast.makeText( context: CustomerViewOrdersActivity.this, text: "Order List Failed",Toast.LENGTH_SHORT).show();

            }
        });

        TextView txtProfileName = (TextView) navigationView.getHeaderView( index: 0).findViewById(R.id.textView_cus_order);
        txtProfileName.setText(email);

        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle( activity: this, drawer, toolbar, "Open navigation drawer", "Close navigation drawer");
        drawer.addDrawerListener(toggle);
        toggle.syncState();

    }

    public class OrderViewAdapter extends BaseAdapter
    {
        private List<Order> orderList;
        private Context context;
        private LayoutInflater layoutInflater;

        public OrderViewAdapter(List<Order> orderList, Context context) {
            this.orderList=orderList;
            this.context = context;
            this.layoutInflater = (LayoutInflater) context.getSystemService(LAYOUT_INFLATER_SERVICE);
        }

        @Override
        public int getCount() { return orderList.size(); }

        @Override
        public Object getItem(int i) { return null; }

        @Override
        public long getItemId(int i) { return 0; }

        @Override
        public View getView(int i, View view, ViewGroup viewGroup)
        {
```

```
        if(view==null)
        {
            view=layoutInflater.inflate(R.layout.customer_order_list_item,viewGroup, attachToRoot: false);

        }
        TextView order_id_View=view.findViewById(R.id.new_order_id);
        TextView food_name_view=view.findViewById(R.id.food_name_order);
        TextView date_view=view.findViewById(R.id.date_order);
        TextView qnty_viewview=view.findViewById(R.id.food_qnty_order);
        TextView total_view=view.findViewById(R.id.food_total_order);
        TextView status_view=view.findViewById(R.id.food_status_order);
        TextView address_view=view.findViewById(R.id.food_dev_address_order);

        String string_order_id=String.valueOf(orderArrayList.get(i).getOrder_id());
        String string_qunatity=String.valueOf(orderArrayList.get(i).getQnty());

        order_id_View.setText(string_order_id);
        total_view.setText("Rs."+orderList.get(i).getTotal()+".00");
        food_name_view.setText(orderList.get(i).getFoodname());
        date_view.setText(orderList.get(i).getDate());
        qnty_viewview.setText(string_qunatity);
        status_view.setText(orderList.get(i).getStatus());
        address_view.setText(orderList.get(i).getAddress());
        return view;
    }
}
```

OntheWayOrdersActivity.java

```
public class OntheWayOrdersActivity extends AppCompatActivity implements NavigationView.OnNavigationItemSelectedListener {

    private AppBarConfiguration mAppBarConfiguration;
    private Foody_API foody_api;
    ArrayList<Order> orderArrayList=new ArrayList<>();
    private RecyclerView onthewayorderrecyclerview;
    private OntheWayOrdersAdapter onthWayOrdersAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_onthe_way_orders);
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        DrawerLayout drawer = findViewById(R.id.drawer_layout);
        NavigationView navigationView = findViewById(R.id.nav_view);
        drawer = findViewById(R.id.drawer_layout);
        navigationView.setNavigationItemSelectedListener(this);


        onthewayorderrecyclerview=findViewById(R.id.customer_onthe_way_orders_view);
        onthewayorderrecyclerview.setLayoutManager(new LinearLayoutManager( context: this));

        Intent intent=getIntent();
        String email=intent.getStringExtra( name: "email");
        String status= "On the Way";

        foody_api= API_Client.getRetrofit().create(Foody_API.class);
        Call<List<Order>> call=foody_api.ViewOnTheWayOrders(email,status);
        call.enqueue(new Callback<List<Order>>() {
            @Override
            public void onResponse(Call<List<Order>> call, Response<List<Order>> response)
            {
                if(response.isSuccessful())
                {
                    String message="successful";
                    Toast.makeText( context: OntheWayOrdersActivity.this,message,Toast.LENGTH_SHORT).show();
                    orderArrayList= new ArrayList<>(response.body());
                    onthWayOrdersAdapter=new OntheWayOrdersAdapter(orderArrayList, context: OntheWayOrdersActivity.this);
                    onthewayorderrecyclerview.setAdapter(onthWayOrdersAdapter);
                }
            }

            @Override
            public void onFailure(Call<List<Order>> call, Throwable t)
            {
                Toast.makeText( context: OntheWayOrdersActivity.this, text: "Order List Failed",Toast.LENGTH_SHORT).show();
            }
        });

        TextView txtProfileName = (TextView) navigationView.getHeaderView( index: 0).findViewById(R.id.textView_onthe_way);
        txtProfileName.setText(email);

        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle( activity: this, drawer, toolbar, "Open navigation drawer", "Close navigation drawer");
        drawer.addDrawerListener(toggle);
        toggle.syncState();
    }
```

OrderReceiptActitvity.java

```java
public class OrderReceiptActivity extends AppCompatActivity
{
    private TextView foodnametextview,qntytextview,toaltextview,addresstextview;
    private Button return_home_button;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_order_receipt);

        final Intent intent=getIntent();
        final String food_name=intent.getStringExtra( name: "foodname");
        final String address=intent.getStringExtra( name: "address");
        final int quantity=intent.getIntExtra( name: "qnty", defaultValue: 0);
        final  int toal=intent.getIntExtra( name: "total", defaultValue: 0);

        foodnametextview=(TextView) findViewById(R.id.order_details_foodname);
        qntytextview=(TextView) findViewById(R.id.order_details_food_qnty);
        toaltextview=(TextView) findViewById(R.id.order_details_food_total);
        addresstextview=(TextView) findViewById(R.id.order_details_food_address);

        return_home_button=(Button) findViewById(R.id.order_details_return_home);

        String food_price=String.valueOf(quantity);
        String food_total=String.valueOf(toal);

        foodnametextview.setText(food_name);
        addresstextview.setText(address);
        qntytextview.setText(food_price);
        toaltextview.setText(food_total);

        return_home_button.setOnClickListener((view) → {
                Intent home_intent=new Intent( packageContext: OrderReceiptActivity.this,CustomerHomeActivity.class);
                startActivity(home_intent);
        });
    }
}
```

UpdateOrderActivity.java

```
public class UpdateOrderActivity extends AppCompatActivity implements NavigationView.OnNavigationItemSelectedListener {

    private AppBarConfiguration mAppBarConfiguration;
    private Foody_API foody_api;
    private Button update_button,Home_button;
    private ImageView updatedimageview;
    private TextView updatedtextview,reachedtextview;
    private Context context;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_update_order);

        Intent get_order_details= getIntent();
        final String food_name=get_order_details.getStringExtra( name: "foodname");
        final int food_price=get_order_details.getIntExtra( name: "foodprice", defaultValue: 0);
        final String city=get_order_details.getStringExtra( name: "city");
        final String address=get_order_details.getStringExtra( name: "address");
        final String contact=get_order_details.getStringExtra( name: "contact");
        final int qnty=get_order_details.getIntExtra( name: "qnty", defaultValue: 0);
        final String order_id=get_order_details.getStringExtra( name: "id");
        final String fname=get_order_details.getStringExtra( name: "fname");
        final String lname=get_order_details.getStringExtra( name: "lname");
        final String email=get_order_details.getStringExtra( name: "email");
        final int total=get_order_details.getIntExtra( name: "total", defaultValue: 0);
        final String date=get_order_details.getStringExtra( name: "date");

        updatedimageview=findViewById(R.id.correct_image_view);
        updatedtextview=findViewById(R.id.updated_text_view);
        reachedtextview=findViewById(R.id.reachedtextview);
        update_button=findViewById(R.id.update_order_button);
        update_button.setOnClickListener((view) -> {
                Order order=new Order();
                order.setDate(date);
                order.setCity(city);
                order.setAddress(address);
                order.setFoodname(food_name);
                order.setLastname(lname);
                order.setEmail(email);
                order.setContact(contact);
                order.setQnty(qnty);
                order.setTotal(total);
                order.setOrder_id(order_id);
                order.setFirstname(fname);
                order.setPrice(food_price);
                String status="Delivered";
                order.setStatus(status);
                foody_api= API_Client.getRetrofit().create(Foody_API.class);
                Call<Void> call =foody_api.update_Order(order);
                call.enqueue(new Callback<Void>() {
                    @Override
                    public void onResponse(Call<Void> call, Response<Void> response) {
                        Toast.makeText( context: UpdateOrderActivity.this, text: "Success", Toast.LENGTH_SHORT).show();
                        updatedimageview.setImageResource(R.drawable.correctlogo);
                        updatedtextview.setText("Enjoy your Meal !");
                        update_button.setVisibility(View.GONE);
                        reachedtextview.setText("Thank you for Choosing Foody!");

                    }
                    @Override
                    public void onFailure(Call<Void> call, Throwable t)
                    {
                        Toast.makeText( context: UpdateOrderActivity.this, text: "Something went Wrong", Toast.LENGTH_SHORT).show();
                    }
                });


        });
```

Adapter

The adapter is the component that connects the data to RecyclerView and determines which ViewHolder will be used to display it.

CompleteOrdersAdapter.java

```
public class CompletedOrdersAdapter extends RecyclerView.Adapter<CompletedOrdersAdapter.ViewHolder>
{
    private ArrayList<Order> orderArrayList=new ArrayList<>();
    private Context context;

    public CompletedOrdersAdapter(ArrayList<Order> orderArrayList, Context context) {
        this.orderArrayList = orderArrayList;
        this.context = context;
    }

    @NonNull
    @Override
    public CompletedOrdersAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType)
    {
        View view;
        LayoutInflater layoutInflater=LayoutInflater.from(context);
        view=layoutInflater.inflate(R.layout.order_history_list,parent, attachToRoot: false);
        CompletedOrdersAdapter.ViewHolder viewHolder=new CompletedOrdersAdapter.ViewHolder(view);
        return viewHolder;
    }

    @Override
    public void onBindViewHolder(@NonNull CompletedOrdersAdapter.ViewHolder holder, int position)
    {
        String string_order_id=String.valueOf(orderArrayList.get(position).getOrder_id());
        String string_qunatity=String.valueOf(orderArrayList.get(position).getQnty());
        String string_total=String.valueOf(orderArrayList.get(position).getTotal());


        holder.order_id_View.setText(string_order_id);
        holder.address_view.setText(orderArrayList.get(position).getAddress());
        holder.date_view.setText(orderArrayList.get(position).getDate());
        holder.food_name_view.setText(orderArrayList.get(position).getFoodname());
        holder.qnty_viewview.setText(string_qunatity);
        holder.status_view.setText(orderArrayList.get(position).getStatus());
        holder.total_view.setText(string_total);

    }
    @Override
    public int getItemCount() { return orderArrayList.size(); }

    public class ViewHolder extends RecyclerView.ViewHolder
    {
        private TextView order_id_View;
        private TextView food_name_view;
        private TextView date_view;
        private TextView qnty_viewview;
        private TextView total_view;
        private TextView status_view;
        private TextView address_view;
        public ViewHolder(@NonNull View itemView)
        {
            super(itemView);
            order_id_View=itemView.findViewById(R.id.new_order_id_history);
            food_name_view=itemView.findViewById(R.id.food_name_order_history);
            date_view=itemView.findViewById(R.id.date_order_history);
            qnty_viewview=itemView.findViewById(R.id.food_qnty_order_history);
            total_view=itemView.findViewById(R.id.food_total_order_history);
            status_view=itemView.findViewById(R.id.food_status_order_history);
            address_view=itemView.findViewById(R.id.food_dev_address_order_history);
        }
    }
}
```

OnthewayOrdersAdapter.java

```
public class OntheWayOrdersAdapter extends RecyclerView.Adapter<OntheWayOrdersAdapter.ViewHolder>
{
    private ArrayList<Order> orderArrayList=new ArrayList<>();
    private Context context;

    public OntheWayOrdersAdapter(ArrayList<Order> orderArrayList, Context context) {
        this.orderArrayList = orderArrayList;
        this.context = context;
    }

    @NonNull
    @Override
    public OntheWayOrdersAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view;
        LayoutInflater layoutInflater=LayoutInflater.from(context);
        view=layoutInflater.inflate(R.layout.customer_ontheway_orders_list,parent, attachToRoot: false);
        OntheWayOrdersAdapter.ViewHolder viewHolder=new OntheWayOrdersAdapter.ViewHolder(view);
        return viewHolder;
    }

    @Override
    public void onBindViewHolder(@NonNull OntheWayOrdersAdapter.ViewHolder holder, final int position)
    {
        final String string_order_id=String.valueOf(orderArrayList.get(position).getOrder_id());
        final String string_qunatity=String.valueOf(orderArrayList.get(position).getQnty());
        final String string_total=String.valueOf(orderArrayList.get(position).getTotal());
        final String item_price=String.valueOf(orderArrayList.get(position).getPrice());
        final String email_customer= orderArrayList.get(position).getEmail();
        final String fname_customer= orderArrayList.get(position).getFirstname();
        final String lname_customer= orderArrayList.get(position).getLastname();
        final String city= orderArrayList.get(position).getCity();
        final String contact= orderArrayList.get(position).getContact();



        holder.order_id_View.setText(string_order_id);
        holder.address_view.setText(orderArrayList.get(position).getAddress());
        holder.date_view.setText(orderArrayList.get(position).getDate());
        holder.food_name_view.setText(orderArrayList.get(position).getFoodname());
        holder.qnty_viewview.setText(string_qunatity);
        holder.status_view.setText(orderArrayList.get(position).getStatus());
        holder.total_view.setText(string_total);
        holder.update_button.setOnClickListener((view) -> {
                Intent intent=new Intent(context, UpdateOrderActivity.class);
                intent.putExtra( name: "foodname",orderArrayList.get(position).getFoodname());
                intent.putExtra( name: "foodprice",orderArrayList.get(position).getPrice());
                intent.putExtra( name: "total",orderArrayList.get(position).getTotal());
                intent.putExtra( name: "qnty",orderArrayList.get(position).getQnty());
                intent.putExtra( name: "email",orderArrayList.get(position).getEmail());
                intent.putExtra( name: "fname",orderArrayList.get(position).getFirstname());
                intent.putExtra( name: "lname",orderArrayList.get(position).getLastname());
                intent.putExtra( name: "city",orderArrayList.get(position).getCity());
                intent.putExtra( name: "address",orderArrayList.get(position).getAddress());
                intent.putExtra( name: "contact",orderArrayList.get(position).getContact());
                intent.putExtra( name: "id",orderArrayList.get(position).getOrder_id());
                intent.putExtra( name: "status",orderArrayList.get(position).getStatus());
                intent.putExtra( name: "date",orderArrayList.get(position).getDate());
                context.startActivity(intent);

        });

    }

    @Override
    public int getItemCount() { return orderArrayList.size(); }

    public class ViewHolder extends RecyclerView.ViewHolder
    {
        private TextView order_id_View;
        private TextView food_name_view;
        private TextView date_view;
        private TextView qnty_viewview;
        private TextView total_view;
        private TextView status_view;
        private TextView address_view;
        private Button update_button;
```

```
        public ViewHolder(@NonNull View itemView)
        {
            super(itemView);
            order_id_View=itemView.findViewById(R.id.new_order_id_onthe_way);
            food_name_view=itemView.findViewById(R.id.food_name_order_ontheway);
            date_view=itemView.findViewById(R.id.date_order_onthe_way);
            qnty_viewview=itemView.findViewById(R.id.food_qnty_order_onthe_way);
            total_view=itemView.findViewById(R.id.food_total_order_onthe_way);
            status_view=itemView.findViewById(R.id.food_status_order_onthe_way);
            address_view=itemView.findViewById(R.id.food_dev_address_order_onthe_way);
            update_button=itemView.findViewById(R.id.update_order_button_onthe_way);
        }
    }
}
```

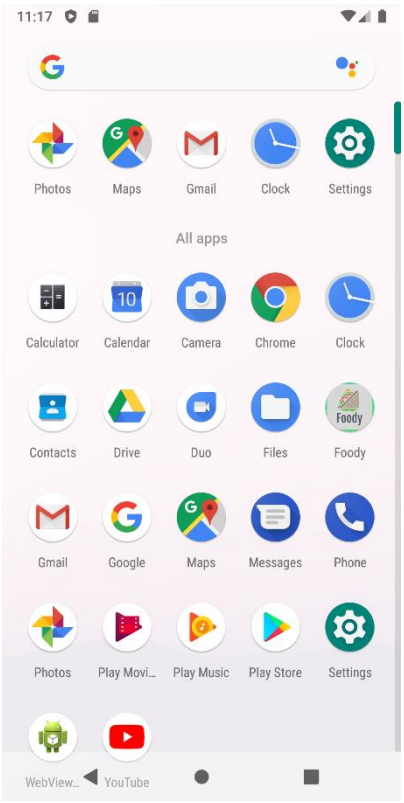
ViewAllReviewsAdapter.java

```
public class ViewAllReviewsAdapter extends RecyclerView.Adapter<ViewAllReviewsAdapter.ViewHolder>
{
    private ArrayList<Review> reviewArrayList=new ArrayList<>();
    private Context context;

    public ViewAllReviewsAdapter(ArrayList<Review> reviewArrayList, Context context) {
        this.reviewArrayList = reviewArrayList;
        this.context = context;
    }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType)
    {
        View view;
        LayoutInflater layoutInflater=LayoutInflater.from(context);
        view=layoutInflater.inflate(R.layout.review_list,parent, attachToRoot: false);
        ViewHolder viewHolder=new ViewHolder(view);
        return viewHolder;
    }

    @Override
    public void onBindViewHolder(@NonNull ViewAllReviewsAdapter.ViewHolder holder, int position)
    {
        holder.review_view.setText(reviewArrayList.get(position).getReviews());
        holder.reviewer_view.setText(reviewArrayList.get(position).getEmail());
        holder.date_view.setText(reviewArrayList.get(position).getDate());
    }

    @Override
    public int getItemCount() { return reviewArrayList.size(); }

    public class ViewHolder  extends RecyclerView.ViewHolder
    {

        private TextView review_view;
        private TextView reviewer_view;
        private TextView date_view;

        public ViewHolder(@NonNull View itemView) {
            super(itemView);
            review_view=itemView.findViewById(R.id.view_review_text);
            reviewer_view=itemView.findViewById(R.id.view_reviewer_text);
            date_view=itemView.findViewById(R.id.view_review_date_text);
        }
    }
}
```

View

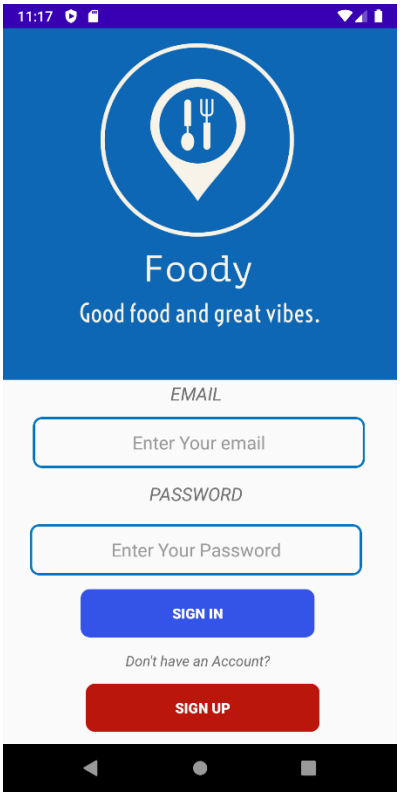
This part of the document contains the Interface images of the mobile Application.

App Icon

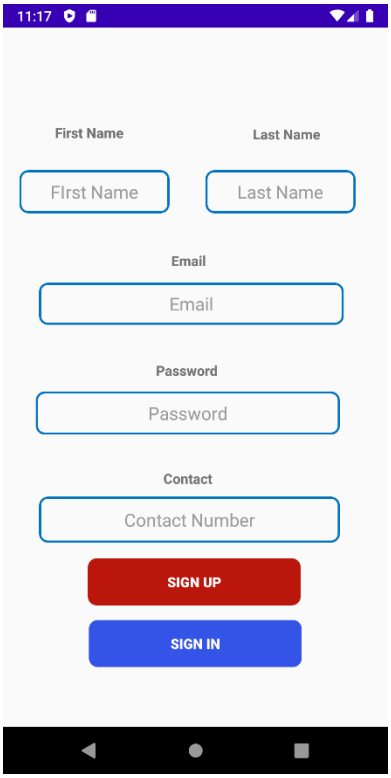


Flash Screen

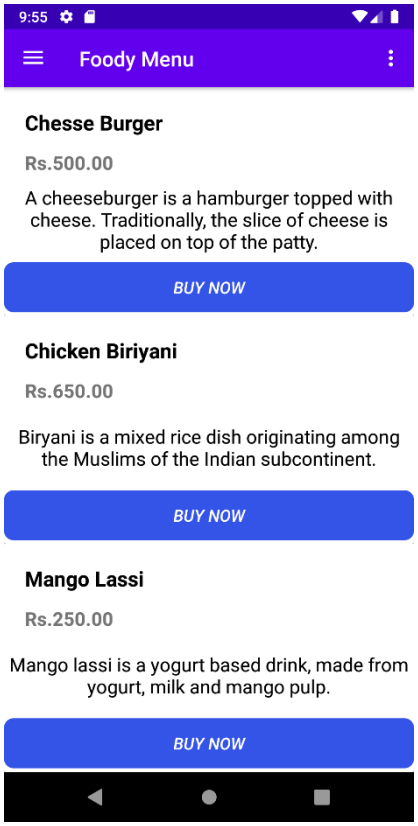
Login Screen



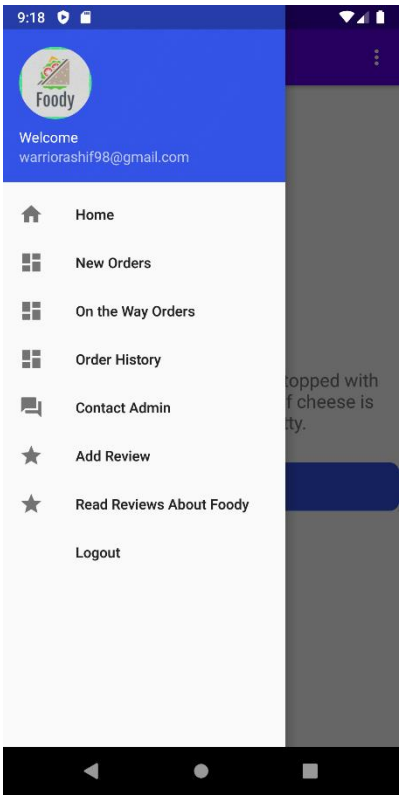
Register Screen



Home Screen

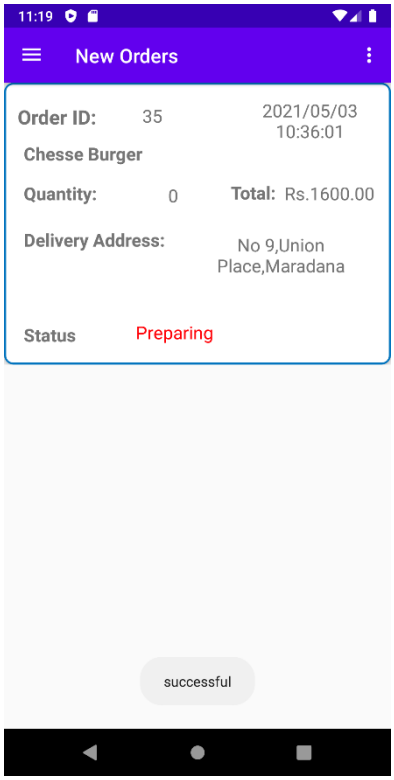


Navigation Drawer Menu

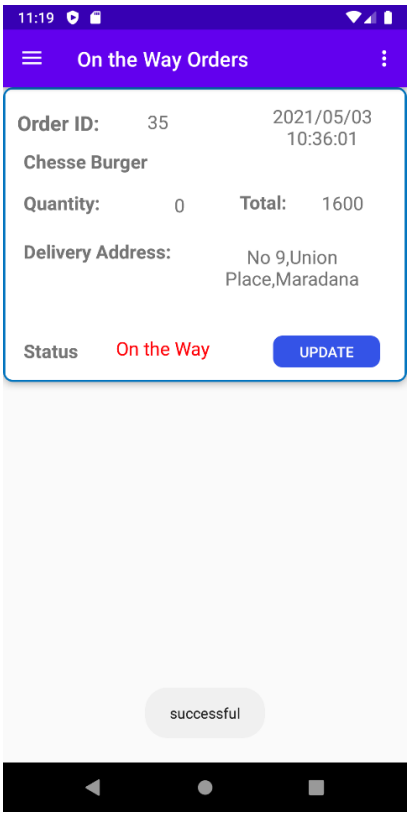


View All Reviews Screen

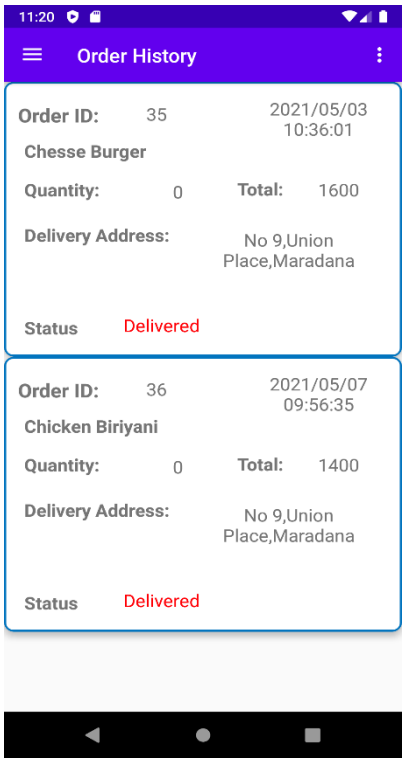
View Preparing Orders Screen



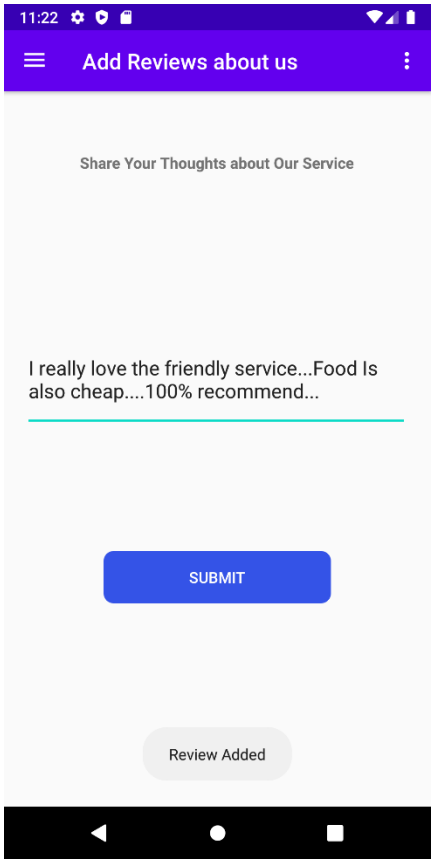
View On the way Orders



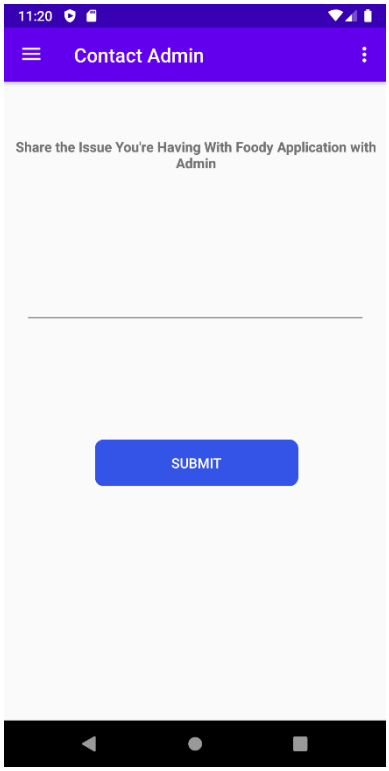
View Order History Screen



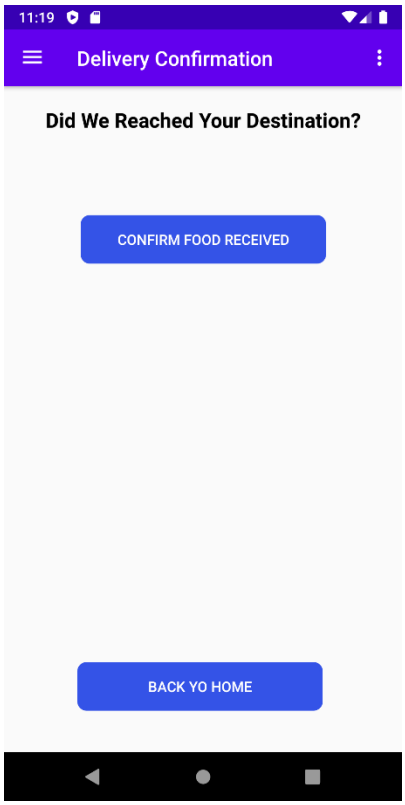
Add Reviews Screen



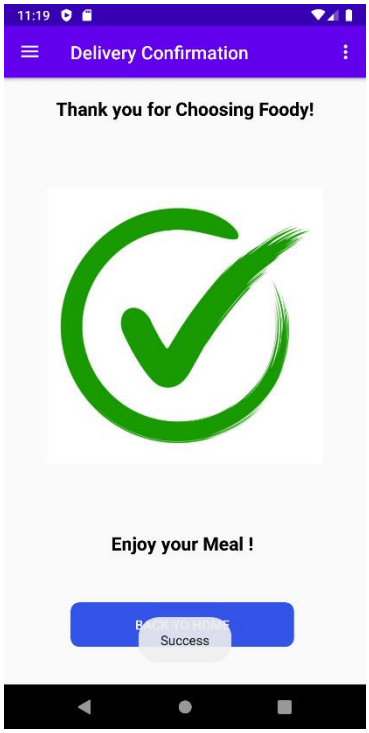
Contact Admin Screen



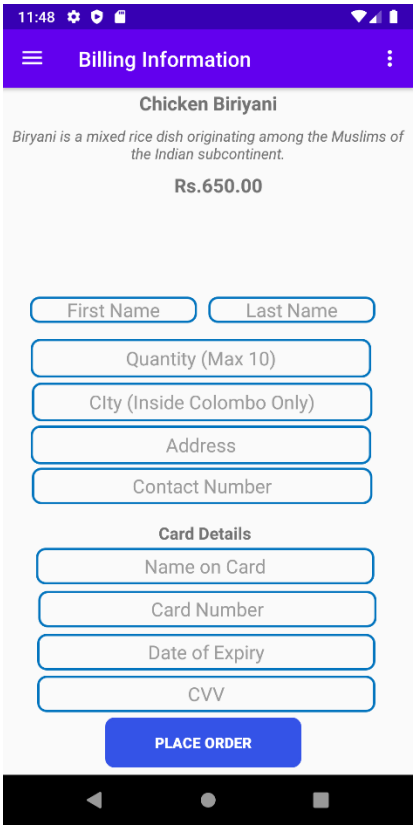
Update Order Screen



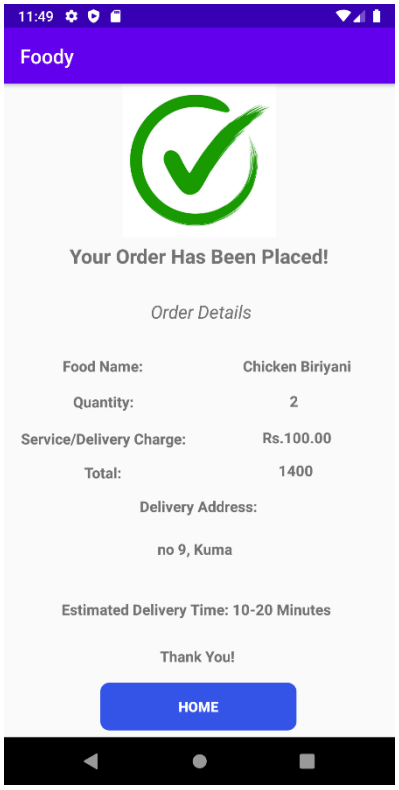
Order status update successful screen



Checkout Screen



Order Confirmation Screen



Test Case

Test ID	Purpose	prerequisite	Steps	Expected Result	Actual Result	Result
001	Selecting Login without entering email and password	System up and Running	01Open the App. 02.Select Login Button.	Display an Empty field message.	Displays "No Email Detected" message	Pass
002	Entering invalid Email for Login	System up and Running	01.Enter invalid email. 02.Select Login Button	Display an Error message	Displays "User does not exist" message	Pass
003	Entering invalid Password for Login	System up and running	01.Enter invalid Password. 02.Select Login Button	Display an Error message	Displays "Password Incorrect" message	Pass
004	Login with valid credentials.	System up and running	01.Enter valid email and Password. 02.Select Login Button	Login to Home page.	Logged into Home Page.	Pass
005	Selecting Register without filling the Register form.	System up and running	01.Open App. 02.Select Register. 03.Select Register Button.	Display empty field error message.	Displayed empty field error message.	Pass
006	Selecting Register with valid details.	System up and running	01.select Register. 02.Fill the form	Display registration successful message.	Display registration successful message.	Pass

			03.Select Login Button			
007	View All Reviews for customer.	System up and running	01.Select "Read Reviews about foody" in Navigation Drawer Menu.	Display all reviews.	Displayed all the reviews.	Pass
008	View New Orders of Customers	System up and running	01.Select "New Orders" in Navigation Drawer Menu.	Display orders where status equals to preparing	Displayed all new orders	Pass
009	View On the way Orders of Customer.	System up and running	01.Select "On the Way Orders" in Navigation Drawer Menu.	Display orders where status equals to on the way.	Displayed all on the way orders.	Pass
010	View Customer Order History	System up and running	01.Select "Order History" in Navigation Drawer Menu.	Display orders where status equals to Delivered.	Displayed all the completed orders.	Pass
011	View customer Profile.	System up and running	01.Select view Profile from Navigation Bar in Customer Home Page.	Display the profile details of the customer	Displayed the customer profile.	Pass
012	View Add Reviews page	System up and running	01.Select "Add Reviews" in Navigation Drawer Menu.	Display Add review page.	Displayed add review page.	Pass
013	Add review with empty field.	System up and running	01.Select "Add Reviews" in Navigation	Display empty field error message.	Displayed empty field error message.	Pass

			Drawer Menu. 02.Select Submit with Empty field.			
014	Add Reviews without empty field.	System up and running	01.Select "Add Reviews" in Navigation Drawer Menu. 02.Select Submit without empty field.	Display review added successful message.	Displayed review added successful message.	Pass
015	View Contact page	System up and running	01.Select "Contact Admin" in Navigation Drawer Menu.	Display contact page.	Displayed contact page.	Pass
016	Add message to admin with empty field.	System up and running	01.Select "Contact Admin" in Navigation Drawer Menu. 02.Select submit button with Empty field.	Display empty field error message.	Displayed empty field error message.	Pass
017	Add message to admin without empty field.	System up and running	01.Select "Contact Admin" in Navigation Drawer Menu. 02.Select submit button with Empty field.	Display message sent successful message.	Displayed message sent successful message.	Pass
018	Customer Change Order Status to Delivered.	System up and running	01.Select "On the Way Orders" in Navigation	Updates the order status to Delivered.	Updated the order status to Delivered.	Pass

			Drawer Menu. 02.select Update. 03.select Confirm food received.			
019	Customer Place order with empty fields.	System up and running	01.select Buy now from Menu 02.displays checkout page. 03.select place order	Display empty field error message.	Displayed empty field error message.	Pass
020	Place order with valid details.	System up and running	01.select Buy now from Menu 02.displays checkout page. 03.fill the form with valid details. 04.select place order	Display Order confirmation Page.	Displayed Order confirmation Page.	Pass.

Conclusion

Reflection

Foody web and Mobile Application help people to order food from home. This will help them to reduce the caution of people gathering due to COVID-19. Beside that, I have gathered knowledge about Restful Application Development.

Strength and Weakness

Strengths	Weakness
User friendly GUI	Cannot Add foods to cart
Track the order via Web and mobile application	Cannot Edit the Customer profile

Future Enhancement

Future of the application will be focused on Add to cart function and Profile Editing function.

References

Bodnar, J (2020). Spring Boot @Controller. [Online] Available from: https://zetcode.com/springboot/controller/ . [Accessed: 6th July 2020]

Seniuk,V (2020). Data Transfer Object Pattern in Java - Implementation and Mapping. [Online] Available from: https://stackabuse.com/data-transfer-object-pattern-in-java-implementation-and-mapping/ .[Accessed: 2020]