

**Sri Lanka Institute of Information Technology**

**Distributed Systems (SE3020)**

**Assignment 02**

**Train Ticket Reservation Application**

**Assignment Report**

Submitted By: IT17056212 (P.M.C.P.Paththinisekara)

**Table of Contents**

**1. Introduction 3**

**2. Architecture 4**

**2.1 High level architectural diagram 5**

**3. Interfaces & Services 6**

**4. Workflow (Backend) 8**

**5. Workflow (Frontend) 10**

**6. Appendix 12**

**6.1 Backend 12**

**6.2 Frontend 38**

1. **Introduction**

This Train Ticket Reservation application is developed for the users who try to book tickets online. This application let the customer reserve any number of train ticket for a specific day in a specific month and the user can do all the payment as a dialog user or Sampath bank user so they can use credit cards and debit cards as well. This application mainly consists of two parts. It’s the web client (Frontend) and Restful services (Backend). And this Train Ticket Reservation web application uses NoSQL MongoDB database as the main database.

1. Web Client: -

Web client is mainly developed using ReactJS and it’s generally accompanied with HTML, JavaScript, CSS. But the designing was mainly done using the Bootstrap for better customer experience. Web client uses Axios Library to send request to the restful services and receive necessary responses.

1. Restful Services: -

The backend which is consist of Restful services are mainly developed using Spring Boot framework and java. And also, these services are developed according SOA principles. There are mainly 4 Restful Services implemented in this application. Each of them controls specific service in the system.

1. **Architecture**

Overall architecture of this Ticket Reservation web application Mainly divided to 2 main parts it’s the Frontend and Backend. Frontend is mainly consisting of ReactJS components. Axios library was used to Send and Receive request and responses. If we consider about functionality there are 2 main functions in this application. They are,

1.Ticket Booking

2.Payment

But If we considering all the main Services Payment also can be divided in to 2 sub services according to the payment method selected by the user. So, in the backend they have implemented as two separate Restful services. And when the user has done the payment successfully an email should be sent to their email address confirming the reservation. So there are 4 RESTful services has been implemented.

1. Ticket Booking-

All the ticket information entered through the web client are send to Ticket booking service.

1. Sampath Payment-

If the user selects the Sampath as the Payment method from list box in the interface all the payment information including Credit or Debit Card Holders name, Card Number, CVC send through Sampath Payment Service to complete the payment through the payment gateway.

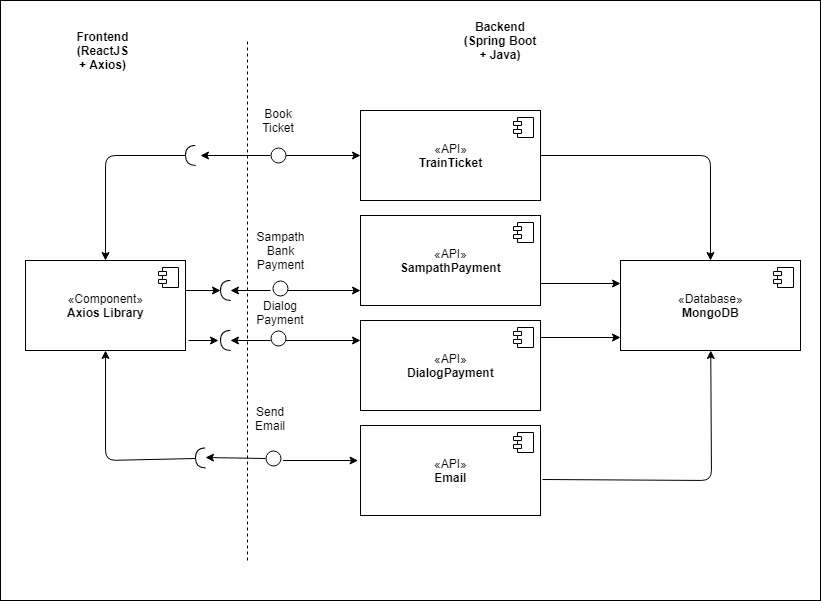
1. Dialog Payment-

If the user selects the Dialog as the Payment method from list box in the interface all the payment information including Phone number and PIN number send through Dialog Payment Service to complete the payment through the payment gateway.

1. Email-

When the Payment is successfully completed through the payment gateways An email will be sent to the users email address which provided in the ticket booking process. The contents of the email and email address send to the Email service for the task of sending the email.

**2.1 High level architectural diagram**



*High level architectural diagram of the overall system.*

When implementing the architecture and design SOA principles were mainly used to make the application better. Most of the services have been design to have less dependency in each other. So even if the service functionality changes at any point in time, it should not break the web client application or stop it from working. And the services have been developed to have maximum reusability we can take email service and payment gateways as an example. So, we wouldn’t have to spend time and effort building the same code again and again across multiple applications which require them.

Each Backend Service has 5 main parts: -

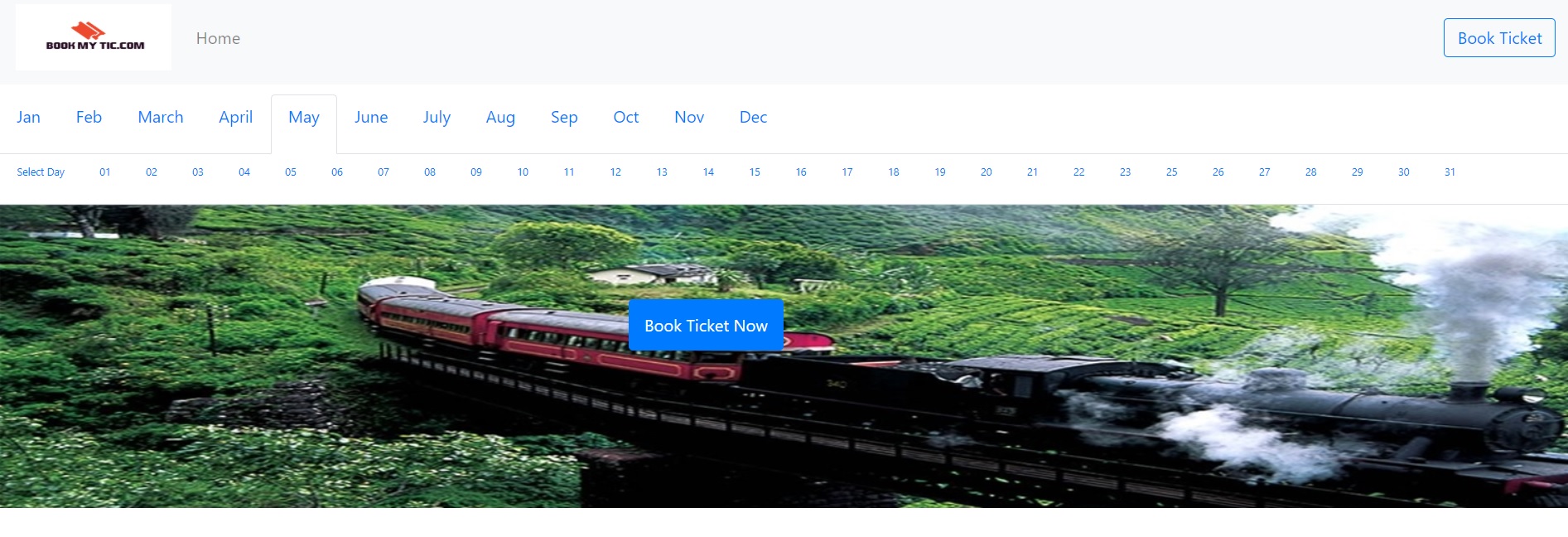
1.Controller

2.Model

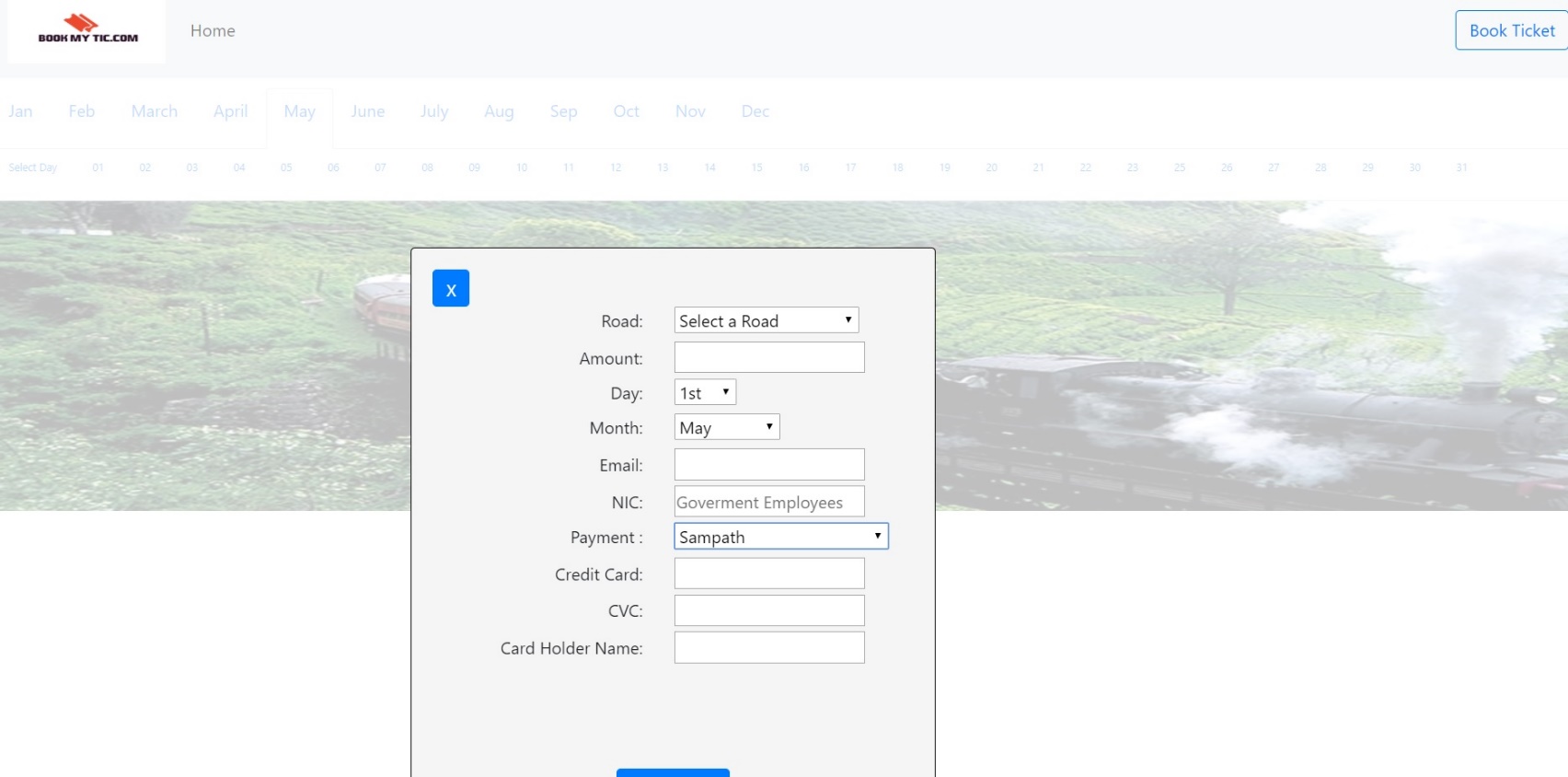
3.Repository

4.Service

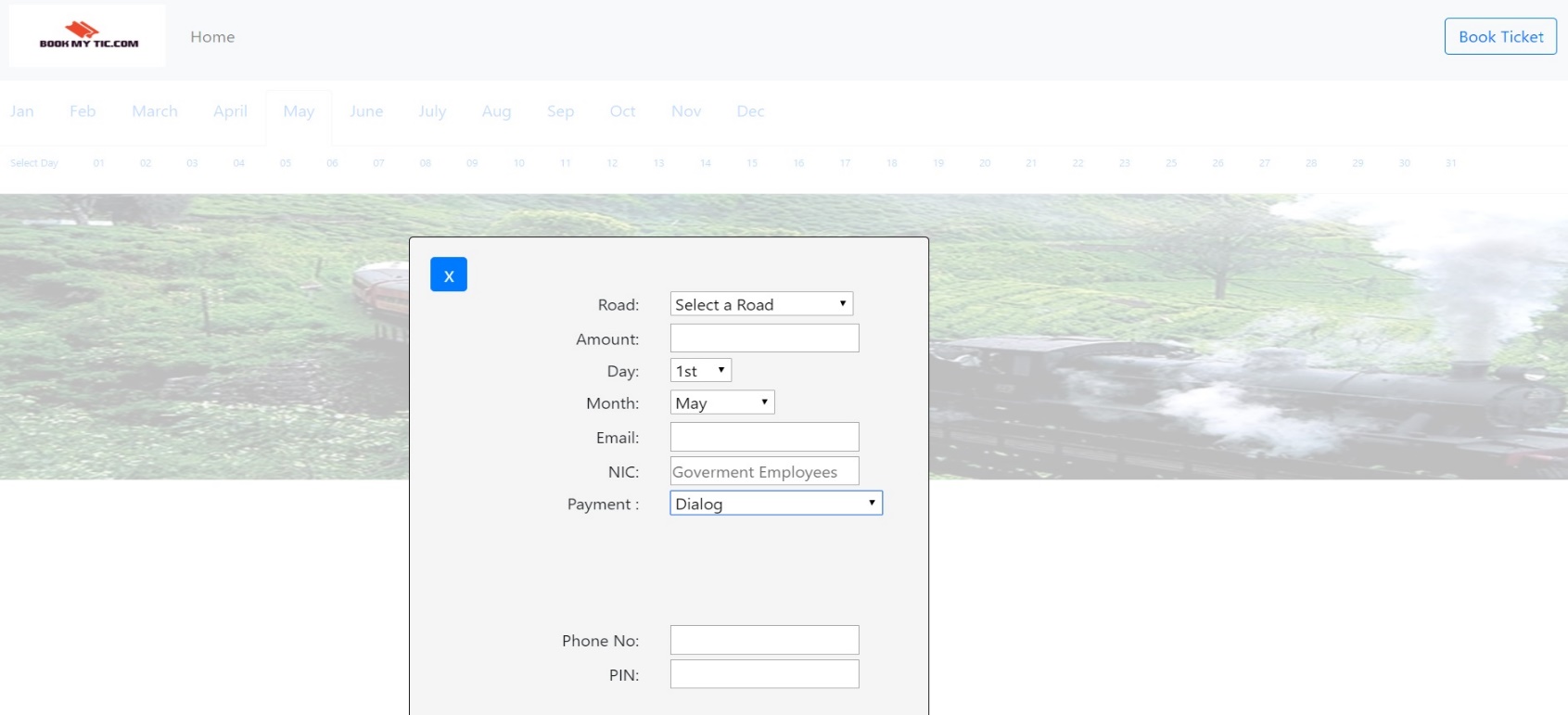
5.Service Implementation

1. **Interfaces & Services**

*Select a month and date*

* In above Interface the customer can select the month and the date for book a ticket after that user can click book now button to fill rest of the information.

*Ticket booking & Making the Sampth Payment.*

**

*Ticket booking & Making the Dialog Payment*.

* Above interface is used to enter the ticket information and ticket booking For this ticket reservation task it uses HTTP requests to GET and POST data. So, It connects to the Train Ticket Service through the Axios library.
* And for making the payment, When the Ticket book button is clicked the Axios library send HTTP requests to the SampathPamentService interface or DialogPaymentService Interface.
* When sending the email to the customer Email Service receive the email info through HTTP requests and send the email to the customer through javaMailSender confirming the reservation.

1. **Workflow (Backend)**

Generally, all the backend services are assembled in the same way. They consist of 4 main components,

1. Controller

2. Service

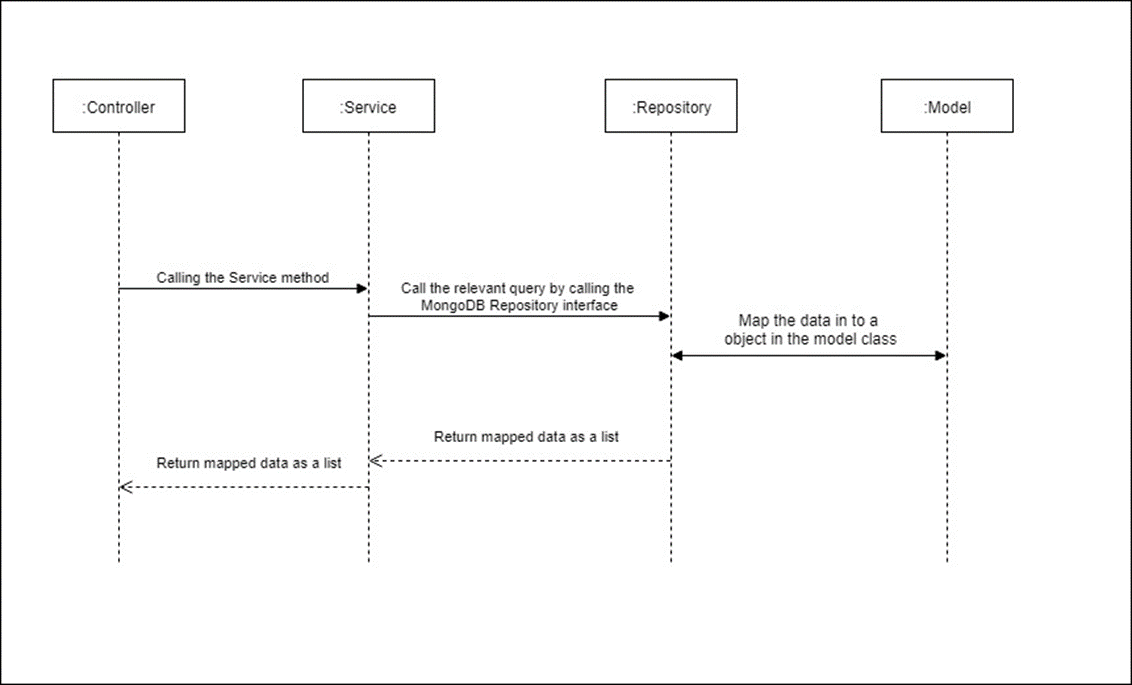
3. Repository

4. Model

All the request that will come through HTTP request such as GET, POST, UPDATE, DELETE first come through Controller class. All the requests that are mapped for that specific URL are handled by a controller class. Then the controller class calls the relevant methods from the service interface. Then the Service interface call the relevant query by calling the MongoDB Repository interface. Then the data get mapped in to an instance of the Model class. If it’s GET request it will return the mapped data as a list.

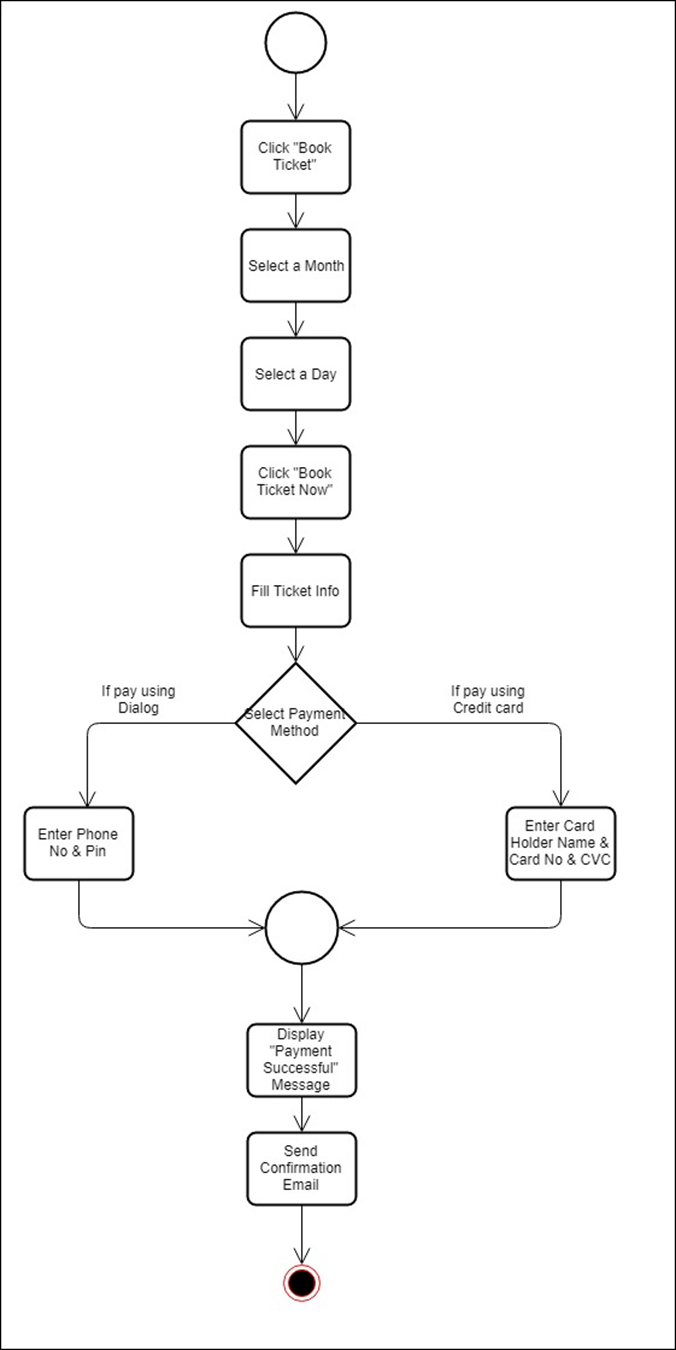
As an example, when a request is mapped for /roads it will be managed by PaymentController. So, when the GET request is made from the web client that the controller receives the HTTP request through the /roads URL then the controller call the

**“*public*** *ResponseEntity<?> getByroad(@PathVariable("road") String road)*” method. The inside that method controller calls the “*List<roads> findByroad(String road); “* method in PaymentService inerface method through PaymentImpl class which has implemented the PaymentService. And the PaymentRepository will be used to handle the MongoDB instances. And also the PaymentRepository extends the MongoRepository interface. MongoRepository implements a defined method in runtime. Finally, the fetch roads and price data from the MongoDB database mapped in to an instance of the roads class. The mapped roads model class data will be stored in a Array List. And the PaymentRepository returns the that list to the PaymentService class. And PaymentService class returns the that list to the PaymentController and the PaymentController can send that list as a response to the web client.



*Generic Backend Service workflow*

1. **Workflow (Frontend)**



Generic Frontend workflow

The Frontend web client is website that has implemented using mainly from ReactJS. JavaScript and CSS were used to designing purposes. Mainly the bootstrap framework was used to get an attractive and responsive user interface. Axios library was used to control all the HTTP requests which are sent to backend and the receiving responses.

1. **Appendix**

**6.1 Backend**

**TrainTicket Service**

**TrainTicketController.java**

**package** com.IT17056212.TrainTicket.controller;

**import** java.util.ArrayList;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.CrossOrigin;

**import** org.springframework.web.bind.annotation.DeleteMapping;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.IT17056212.TrainTicket.model.Ticket;

**import** com.IT17056212.TrainTicket.service.TrainTicketService;

@CrossOrigin(origins = "http://localhost:3000")

@RestController

@RequestMapping("/ticket")

**public** **class** TrainTicketController {

@Autowired

TrainTicketService trainticketservice;

//GET all

@GetMapping

**public** ResponseEntity<?> getAll() {

List<Ticket> result = trainticketservice.findAll();

**return** **new** ResponseEntity(result, HttpStatus.***OK***);

}

//GET by year and month

@GetMapping("/{year}/{month}")

**public** ResponseEntity<?> getBydaymon(@PathVariable("mon") String mon, @PathVariable("day") String day) {

List<Ticket> result = **new** ArrayList<>();

**if**("All".equals(day)) {

result = trainticketservice.findBymon(mon);

} **else** {

result = trainticketservice.findBydayAndMon(day, mon);

}

**return** **new** ResponseEntity(result, HttpStatus.***OK***);

}

//Add Ticket details

@PostMapping

**public** ResponseEntity<?> addorUpdateExpense(@RequestBody Ticket expense) {

trainticketservice.saveOrUpdateExpense(expense);

**return** **new** ResponseEntity("Ticket Booked succcessfully", HttpStatus.***OK***);

}

}

**Ticket.java**

**package** com.IT17056212.TrainTicket.model;

**import** org.springframework.data.annotation.Id;

**import** org.springframework.data.mongodb.core.mapping.Document;

@Document

**public** **class** Ticket {

@Id

String id;

String road;

Integer amount;

String day;

String mon;

String pay;

String nic;

String card;

String cvc;

String phone;

String pin;

String email;

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** String getroad() {

**return** road;

}

**public** **void** setroad(String road) {

**this**.road = road;

}

**public** Integer getAmount() {

**return** amount;

}

**public** **void** setAmount(Integer amount) {

**this**.amount = amount;

}

**public** String getday() {

**return** day;

}

**public** **void** setday(String day) {

**this**.day = day;

}

**public** String getmon() {

**return** mon;

}

**public** **void** setmon(String mon) {

**this**.mon = mon;

}

**public** String getPay() {

**return** pay;

}

**public** **void** setPay(String pay) {

**this**.pay = pay;

}

**public** String getNic() {

**return** nic;

}

**public** **void** setNic(String nic) {

**this**.nic = nic;

}

**public** String getCard() {

**return** card;

}

**public** **void** setCard(String card) {

**this**.card = card;

}

**public** String getCvc() {

**return** cvc;

}

**public** **void** setCvc(String cvc) {

**this**.cvc = cvc;

}

**public** String getPhone() {

**return** phone;

}

**public** **void** setPhone(String phone) {

**this**.phone = phone;

}

**public** String getPin() {

**return** pin;

}

**public** **void** setPin(String pin) {

**this**.pin = pin;

}

}

**TrainTicketRepository.java**

**package** com.IT17056212.TrainTicket.repositories;

**import** java.util.List;

**import** org.springframework.data.mongodb.repository.MongoRepository;

**import** com.IT17056212.TrainTicket.model.Ticket;

**public** **interface** TrainTicketRepository **extends** MongoRepository<Ticket, String>{

//find by day and month

List<Ticket> findBydayAndMon(String day, String mon);

//find by month

List<Ticket> findBymon(String mon);

}

**TrainTicketImpl.java**

**package** com.IT17056212.TrainTicket.service;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** com.IT17056212.TrainTicket.model.Ticket;

**import** com.IT17056212.TrainTicket.repositories.TrainTicketRepository;

@Service

**public** **class** TrainTicketImpl **implements** TrainTicketService {

@Autowired

TrainTicketRepository trainticketRepository;

//find all

@Override

**public** List<Ticket> findAll() {

**return** trainticketRepository.findAll();

}

//find by day and month

@Override

**public** List<Ticket> findBydayAndMon(String day, String mon) {

**return** trainticketRepository.findBydayAndMon(day, mon);

}

//add ticket

@Override

**public** **void** saveOrUpdateExpense(Ticket ticket) {

trainticketRepository.save(ticket);

}

//find by month

@Override

**public** List<Ticket> findBymon(String mon) {

**return** trainticketRepository.findBymon(mon);

}

}

**TrainTicketService.java**

**package** com.IT17056212.TrainTicket.service;

**import** java.util.List;

**import** com.IT17056212.TrainTicket.model.Ticket;

**public** **interface** TrainTicketService {

//find all

List<Ticket> findAll();

//find by day and month

List<Ticket> findBydayAndMon(String day, String mon);

//find by month

List<Ticket> findBymon(String mon);

//add ticket

**void** saveOrUpdateExpense(Ticket ticket);

}

**SampathPayment Service**

**NicController.java**

**package** com.IT17056212.SampathPayment.controller;

**import** java.util.ArrayList;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.CrossOrigin;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.IT17056212.SampathPayment.model.govermentnic;

**import** com.IT17056212.SampathPayment.model.roads;

**import** com.IT17056212.SampathPayment.model.sampathpayment;

**import** com.IT17056212.SampathPayment.service.NicService;

**import** com.IT17056212.SampathPayment.service.PaymentService;

@CrossOrigin(origins = "http://localhost:3000")

@RestController

@RequestMapping("/govermentnic")

**public** **class** NicController {

@Autowired

NicService nicservice;

//GET by nic if to check whether it's a goverment nic

@GetMapping("/{govermentnic}")

**public** ResponseEntity<?> getBynic(@PathVariable("govermentnic") String nic) {

List<govermentnic> result = **new** ArrayList<>();

sampathpayment n=**new** sampathpayment();

result = nicservice.findBynic(nic);

**for** (**int** i=0; i<result.size(); i++){

System.***out***.println(result.get(i).getStatus());

String p=result.get(i).getStatus();

**if**(p.equalsIgnoreCase("yes")) {

n.setDiscount((0.9\*roads.*getFullprice*()));

}**else** {

n.setDiscount((2\*roads.*getFullprice*()));

}

}

**return** **new** ResponseEntity(result, HttpStatus.***OK***);

}

}

**PaymentController.java**

**package** com.IT17056212.SampathPayment.controller;

**import** java.util.ArrayList;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.CrossOrigin;

**import** org.springframework.web.bind.annotation.DeleteMapping;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.IT17056212.SampathPayment.model.roads;

**import** com.IT17056212.SampathPayment.service.PaymentService;

@CrossOrigin(origins = "http://localhost:3000")

@RestController

@RequestMapping("/roads")

**public** **class** PaymentController {

@Autowired

PaymentService payment;

//GET price by road

@GetMapping("/{road}")

**public** ResponseEntity<?> getByroad(@PathVariable("road") String road) {

List<roads> result = **new** ArrayList<>();

result = payment.findByroad(road);

**for** (**int** i=0; i<result.size(); i++){

System.***out***.println(result.get(i).getPrice());

**double** p=result.get(i).getPrice();

roads.*setFullprice*(p);

System.***out***.println(roads.*getFullprice*());

}

**return** **new** ResponseEntity(result, HttpStatus.***OK***);

}

@PostMapping

**public** ResponseEntity<?> addorUpdateExpense(@RequestBody roads expense) {

payment.saveOrUpdateExpense(expense);

**return** **new** ResponseEntity("Payment Successfull", HttpStatus.***OK***);

}

}

**SampathPaymentController.java**

**package** com.IT17056212.SampathPayment.controller;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.CrossOrigin;

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.IT17056212.SampathPayment.model.roads;

**import** com.IT17056212.SampathPayment.model.sampathpayment;

**import** com.IT17056212.SampathPayment.service.PaymentService;

**import** com.IT17056212.SampathPayment.service.SampathPaymentService;

@CrossOrigin(origins = "http://localhost:3000")

@RestController

@RequestMapping("/sampathpayment")

**public** **class** SampathPaymentController {

@Autowired

SampathPaymentService payment;

//Add payment

@PostMapping

**public** ResponseEntity<?> addorUpdateExpense(@RequestBody sampathpayment s) {

payment.saveOrUpdateExpense(s);

**return** **new** ResponseEntity("Payment Successfull", HttpStatus.***OK***);

}

}

**govermentnic.java**

**package** com.IT17056212.SampathPayment.model;

**import** org.springframework.data.annotation.Id;

**public** **class** govermentnic {

@Id

String id;

String nic;

String status;

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** String getNic() {

**return** nic;

}

**public** **void** setNic(String nic) {

**this**.nic = nic;

}

**public** String getStatus() {

**return** status;

}

**public** **void** setStatus(String status) {

**this**.status = status;

}

**public** govermentnic(String id, String nic, String status) {

**this**.id = id;

**this**.nic = nic;

**this**.status = status;

}

}

**roads.java**

**package** com.IT17056212.SampathPayment.model;

**import** org.springframework.data.annotation.Id;

**import** org.springframework.data.mongodb.core.mapping.Document;

@Document

**public** **class** roads {

@Id

String id;

String road;

Integer amount;

**double** price;

**static** **double** *fullprice*;

**public** roads(String id, String road, Integer amount, **double** price) {

**super**();

**this**.id = id;

**this**.road = road;

**this**.amount = amount;

**this**.price = price;

}

**public** **static** **double** getFullprice() {

**return** *fullprice*;

}

**public** **static** **void** setFullprice(**double** fullprice) {

roads.*fullprice* = fullprice;

}

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** Integer getAmount() {

**return** amount;

}

**public** **void** setAmount(Integer amount) {

**this**.amount = amount;

}

**public** String getRoad() {

**return** road;

}

**public** **void** setRoad(String road) {

**this**.road = road;

}

**public** **double** getPrice() {

**return** price;

}

**public** **void** setPrice(**double** price) {

**this**.price = price;

}

}

**sampathpayment.java**

**package** com.IT17056212.SampathPayment.model;

**import** org.springframework.data.annotation.Id;

**import** org.springframework.data.mongodb.core.mapping.Document;

@Document

**public** **class** sampathpayment {

@Id

String id;

String road;

Integer amount;

Integer card;

**double** discount;

**double** price=roads.*getFullprice*();

**public** **double** getDiscount() {

**return** discount;

}

**public** **void** setDiscount(**double** discount) {

**this**.discount = discount;

System.***out***.println("full "+roads.*getFullprice*());

System.***out***.println("discount "+**this**.discount);

}

**public** Integer getCard() {

**return** card;

}

**public** **void** setCard(Integer card) {

**this**.card = card;

}

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** Integer getAmount() {

**return** amount;

}

**public** **void** setAmount(Integer amount) {

**this**.amount = amount;

}

**public** String getRoad() {

**return** road;

}

**public** **void** setRoad(String road) {

**this**.road = road;

}

**public** **double** getPrice() {

**return** price;

}

**public** **void** setPrice(**double** price) {

System.***out***.println(price);

**this**.price = price;

System.***out***.println(price);

}

}

**NicRepository.java**

**package** com.IT17056212.SampathPayment.repositories;

**import** java.util.List;

**import** org.springframework.data.mongodb.repository.MongoRepository;

**import** com.IT17056212.SampathPayment.model.govermentnic;

**public** **interface** NicRepository **extends** MongoRepository<govermentnic, String> {

//find by nic

List<govermentnic> findBynic(String nic);

}

**PaymentRepository.java**

**package** com.IT17056212.SampathPayment.repositories;

**import** java.util.List;

**import** org.springframework.data.mongodb.repository.MongoRepository;

**import** com.IT17056212.SampathPayment.model.roads;

**public** **interface** PaymentRepository **extends** MongoRepository<roads, String> {

//find by road

List<roads> findByroad(String road);

}

**NicImpl.java**

**package** com.IT17056212.SampathPayment.service;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** com.IT17056212.SampathPayment.model.govermentnic;

**import** com.IT17056212.SampathPayment.repositories.NicRepository;

@Service

**public** **class** NicImpl **implements** NicService{

@Autowired

NicRepository nicrep;

//find by nic

@Override

**public** List<govermentnic> findBynic(String nic) {

**return** nicrep.findBynic(nic);

}

}

**NicService.java**

**package** com.IT17056212.SampathPayment.service;

**import** java.util.List;

**import** com.IT17056212.SampathPayment.model.govermentnic;

**public** **interface** NicService {

//find by nic

List<govermentnic> findBynic(String nic);

}

**PaymentImpl.java**

**package** com.IT17056212.SampathPayment.service;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** com.IT17056212.SampathPayment.model.roads;

**import** com.IT17056212.SampathPayment.repositories.PaymentRepository;

@Service

**public** **class** PaymentImpl **implements** PaymentService {

@Autowired

PaymentRepository paymentRepository;

//add payment

@Override

**public** **void** saveOrUpdateExpense(roads r) {

paymentRepository.save(r);

}

//find by road

@Override

**public** List<roads> findByroad(String road) {

**return** paymentRepository.findByroad(road);

}

}

**PaymentService.java**

**package** com.IT17056212.SampathPayment.service;

**import** java.util.List;

**import** com.IT17056212.SampathPayment.model.roads;

**public** **interface** PaymentService {

//find by road

List<roads> findByroad(String road);

//add payment

**void** saveOrUpdateExpense(roads payment);

}

**SampathPaymentImpl.java**

**package** com.IT17056212.SampathPayment.service;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** com.IT17056212.SampathPayment.model.roads;

**import** com.IT17056212.SampathPayment.model.sampathpayment;

**import** com.IT17056212.SampathPayment.repositories.PaymentRepository;

**import** com.IT17056212.SampathPayment.repositories.SampathPaymentRepository;

@Service

**public** **class** SampathPaymentImpl **implements** SampathPaymentService {

@Autowired

SampathPaymentRepository paymentRepository;

//add payment

@Override

**public** **void** saveOrUpdateExpense(sampathpayment s) {

paymentRepository.save(s);

}

}

**SampathPaymentService.java**

**package** com.IT17056212.SampathPayment.service;

**import** com.IT17056212.SampathPayment.model.roads;

**import** com.IT17056212.SampathPayment.model.sampathpayment;

**public** **interface** SampathPaymentService {

//add payment

**void** saveOrUpdateExpense(sampathpayment payment);

}

**DialogPayment Service**

**DialogPaymentController.java**

**package** com.IT17056212.DialogPayment.controller;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.CrossOrigin;

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.IT17056212.DialogPayment.model.dialogpayment;

**import** com.IT17056212.DialogPayment.service.DialogPaymentService;

@CrossOrigin(origins = "http://localhost:3000")

@RestController

@RequestMapping("/dialogpayment")

**public** **class** DialogPaymentController {

@Autowired

DialogPaymentService payment;

//add payment

@PostMapping

**public** ResponseEntity<?> addorUpdateExpense(@RequestBody dialogpayment s) {

payment.saveOrUpdateExpense(s);

**return** **new** ResponseEntity("Dialog Payment Successfull", HttpStatus.***OK***);

}

}

**NicController.java**

**package** com.IT17056212.DialogPayment.controller;

**import** java.util.ArrayList;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.CrossOrigin;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.IT17056212.DialogPayment.model.govermentnic;

**import** com.IT17056212.DialogPayment.model.roads;

**import** com.IT17056212.DialogPayment.model.dialogpayment;

**import** com.IT17056212.DialogPayment.service.NicService;

**import** com.IT17056212.DialogPayment.service.PaymentService;

@CrossOrigin(origins = "http://localhost:3000")

@RestController

@RequestMapping("/govermentnic")

**public** **class** NicController {

@Autowired

NicService nicservice;

//GET by nic if to check whether it's a goverment nic

@GetMapping("/{govermentnic}")

**public** ResponseEntity<?> getBynic(@PathVariable("govermentnic") String nic) {

List<govermentnic> result = **new** ArrayList<>();

dialogpayment n=**new** dialogpayment();

result = nicservice.findBynic(nic);

**for** (**int** i=0; i<result.size(); i++){

System.***out***.println(result.get(i).getStatus());

String p=result.get(i).getStatus();

**if**(p.equalsIgnoreCase("yes")) {

n.setDiscount((0.9\*roads.*getFullprice*()));

}**else** {

n.setDiscount((2\*roads.*getFullprice*()));

}

}

**return** **new** ResponseEntity(result, HttpStatus.***OK***);

}

}

**PaymentController.java**

**package** com.IT17056212.DialogPayment.controller;

**import** java.util.ArrayList;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.CrossOrigin;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.IT17056212.DialogPayment.model.roads;

**import** com.IT17056212.DialogPayment.service.PaymentService;

@CrossOrigin(origins = "http://localhost:3000")

@RestController

@RequestMapping("/roads")

**public** **class** PaymentController {

@Autowired

PaymentService payment;

//GET price by road

@GetMapping("/{road}")

**public** ResponseEntity<?> getByroad(@PathVariable("road") String road) {

List<roads> result = **new** ArrayList<>();

result = payment.findByroad(road);

**for** (**int** i=0; i<result.size(); i++){

System.***out***.println(result.get(i).getPrice());

**double** p=result.get(i).getPrice();

roads.*setFullprice*(p);

System.***out***.println(roads.*getFullprice*());

}

**return** **new** ResponseEntity(result, HttpStatus.***OK***);

}

//add payment

@PostMapping

**public** ResponseEntity<?> addorUpdateExpense(@RequestBody roads expense) {

payment.saveOrUpdateExpense(expense);

**return** **new** ResponseEntity("Payment Successfull", HttpStatus.***OK***);

}

}

**dialogpayment.java**

**package** com.IT17056212.DialogPayment.model;

**import** org.springframework.data.annotation.Id;

**import** org.springframework.data.mongodb.core.mapping.Document;

**import** com.IT17056212.DialogPayment.model.roads;

@Document

**public** **class** dialogpayment {

@Id

String id;

String road;

Integer amount;

Integer phone;

**double** price=roads.*getFullprice*();

**double** discount;

**public** **double** getDiscount() {

**return** discount;

}

**public** **void** setDiscount(**double** discount) {

**this**.discount = discount;

System.***out***.println("full "+roads.*getFullprice*());

System.***out***.println("discount "+**this**.discount);

}

**public** Integer getPhone() {

**return** phone;

}

**public** **void** setPhone(Integer phone) {

**this**.phone = phone;

}

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** Integer getAmount() {

**return** amount;

}

**public** **void** setAmount(Integer amount) {

**this**.amount = amount;

}

**public** String getRoad() {

**return** road;

}

**public** **void** setRoad(String road) {

**this**.road = road;

}

**public** **double** getPrice() {

**return** price;

}

**public** **void** setPrice(**double** price) {

System.***out***.println(price);

**this**.price = price;

System.***out***.println(price);

}

}

**govermentnic.java**

**package** com.IT17056212.DialogPayment.model;

**import** org.springframework.data.annotation.Id;

**public** **class** govermentnic {

@Id

String id;

String nic;

String status;

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** String getNic() {

**return** nic;

}

**public** **void** setNic(String nic) {

**this**.nic = nic;

}

**public** String getStatus() {

**return** status;

}

**public** **void** setStatus(String status) {

**this**.status = status;

}

**public** govermentnic(String id, String nic, String status) {

**this**.id = id;

**this**.nic = nic;

**this**.status = status;

}

}

**roads.java**

**package** com.IT17056212.DialogPayment.model;

**import** org.springframework.data.annotation.Id;

**import** org.springframework.data.mongodb.core.mapping.Document;

@Document

**public** **class** roads {

@Id

String id;

String road;

Integer amount;

**double** price;

**static** **double** *fullprice*;

**public** **static** **double** getFullprice() {

**return** *fullprice*;

}

**public** **static** **void** setFullprice(**double** fullprice) {

roads.*fullprice* = fullprice;

}

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** Integer getAmount() {

**return** amount;

}

**public** **void** setAmount(Integer amount) {

**this**.amount = amount;

}

**public** String getRoad() {

**return** road;

}

**public** **void** setRoad(String road) {

**this**.road = road;

}

**public** **double** getPrice() {

**return** price;

}

**public** **void** setPrice(**double** price) {

**this**.price = price;

}

}

**NicRepository.java**

**package** com.IT17056212.DialogPayment.repositories;

**import** java.util.List;

**import** org.springframework.data.mongodb.repository.MongoRepository;

**import** com.IT17056212.DialogPayment.model.govermentnic;

**public** **interface** NicRepository **extends** MongoRepository<govermentnic, String> {

//find by nic

List<govermentnic> findBynic(String nic);

}

**PaymentRepository.java**

**package** com.IT17056212.DialogPayment.repositories;

**import** java.util.List;

**import** org.springframework.data.mongodb.repository.MongoRepository;

**import** com.IT17056212.DialogPayment.model.roads;

**public** **interface** PaymentRepository **extends** MongoRepository<roads, String> {

//find by road

List<roads> findByroad(String road);

}

**DialogPaymentImpl.java**

**package** com.IT17056212.DialogPayment.service;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** com.IT17056212.DialogPayment.model.dialogpayment;

**import** com.IT17056212.DialogPayment.repositories.DialogPaymentRepository;

@Service

**public** **class** DialogPaymentImpl **implements** DialogPaymentService {

@Autowired

DialogPaymentRepository paymentRepository;

//add payment

@Override

**public** **void** saveOrUpdateExpense(dialogpayment s) {

paymentRepository.save(s);

}

}

**DialogPaymentService.java**

**package** com.IT17056212.DialogPayment.service;

**import** com.IT17056212.DialogPayment.model.dialogpayment;

**public** **interface** DialogPaymentService {

//add payment

**void** saveOrUpdateExpense(dialogpayment payment);

}

**NicImpl.java**

**package** com.IT17056212.DialogPayment.service;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** com.IT17056212.DialogPayment.model.govermentnic;

**import** com.IT17056212.DialogPayment.repositories.NicRepository;

@Service

**public** **class** NicImpl **implements** NicService{

@Autowired

NicRepository nicrep;

//find by nic

@Override

**public** List<govermentnic> findBynic(String nic) {

**return** nicrep.findBynic(nic);

}

}

**NicService.java**

**package** com.IT17056212.DialogPayment.service;

**import** java.util.List;

**import** com.IT17056212.DialogPayment.model.govermentnic;

**public** **interface** NicService {

//find by nic

List<govermentnic> findBynic(String nic);

}

**PaymentImpl.java**

**package** com.IT17056212.DialogPayment.service;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** com.IT17056212.DialogPayment.model.roads;

**import** com.IT17056212.DialogPayment.repositories.PaymentRepository;

@Service

**public** **class** PaymentImpl **implements** PaymentService {

@Autowired

PaymentRepository paymentRepository;

//add payment

@Override

**public** **void** saveOrUpdateExpense(roads r) {

paymentRepository.save(r);

}

//find by road

@Override

**public** List<roads> findByroad(String road) {

**return** paymentRepository.findByroad(road);

}

}

**PaymentService.java**

**package** com.IT17056212.DialogPayment.service;

**import** java.util.List;

**import** com.IT17056212.DialogPayment.model.roads;

**public** **interface** PaymentService {

//find by road

List<roads> findByroad(String road);

//add payment

**void** saveOrUpdateExpense(roads payment);

}

**email Service**

**EmailApplication.java**

**package** com.mail.email;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** EmailApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(EmailApplication.**class**, args);

}

}

**Emailcontroller.java**

**package** com.mail.email.controller;

**import** javax.mail.MessagingException;

**import** javax.mail.internet.MimeMessage;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.mail.SimpleMailMessage;

**import** org.springframework.mail.javamail.JavaMailSender;

**import** org.springframework.mail.javamail.MimeMessageHelper;

**import** org.springframework.web.bind.annotation.CrossOrigin;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.mail.email.domain.Email;

@RestController

@RequestMapping(value = "/email")

@CrossOrigin(origins="\*")

**public** **class** Emailcontroller {

@Autowired

**public** JavaMailSender javaMailSender;

//recieve email (to,body,sub) & send email

@RequestMapping(value = "/send",method=RequestMethod.***POST***)

**public** ResponseEntity<?> addPayment(@RequestBody Email email) **throws** MessagingException{

MimeMessage mime=javaMailSender.createMimeMessage();

MimeMessageHelper me= **new** MimeMessageHelper(mime,**true**);

me.setTo(email.getTo());

me.setSubject(email.getSub());

me.setText(email.getBody(),**true**);

javaMailSender.send(mime);

System.***out***.println(email.getTo());

**return** **new** ResponseEntity<>(**null**, HttpStatus.***OK***);

}

}

**Email.java**

**package** com.mail.email.domain;

**public** **class** Email {

**private** String to;

**private** String sub;

**private** String body;

**public** String getTo() {

**return** to;

}

**public** **void** setTo(String to) {

**this**.to = to;

}

**public** String getSub() {

**return** sub;

}

**public** **void** setSub(String sub) {

**this**.sub = sub;

}

**public** String getBody() {

**return** body;

}

**public** **void** setBody(String body) {

**this**.body = body;

}

}

**6.2 Frontend**

**Index.html**

<!DOCTYPE html>  
<html lang="en">  
 <head>  
 <meta charset="utf-8" />  
 <link rel="shortcut icon" href="" />  
 <meta  
 name="viewport"  
 content="width=device-width, initial-scale=1, shrink-to-fit=no"  
 />  
 <meta name="theme-color" content="#000000" />ke  
 <!--  
 manifest.json provides metadata used when your web app is installed on a  
 user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/  
 -->  
 <link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />  
 <!--  
 Notice the use of %PUBLIC\_URL% in the tags above.  
 It will be replaced with the URL of the `public` folder during the build.  
 Only files inside the `public` folder can be referenced from the HTML.  
  
 Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC\_URL%/favicon.ico" will  
 work correctly both with client-side routing and a non-root public URL.  
 Learn how to configure a non-root public URL by running `npm run build`.  
 -->  
 <title>CourseWeb</title>  
  
  
 <script>  
 src="%PUBLIC\_URL%/bootstrap/react-bootstrap.min.js"  
 </script>  
 <link rel="stylesheet" href="%PUBLIC\_URL%/bootstrap/bootstrap.min.css">  
 <script>  
 src="%PUBLIC\_URL%/it17152938/js/tabmove.js"  
 </script>  
  
 <script>var Alert = ReactBootstrap.Alert;</script>  
 </head>  
 <body>  
 <noscript>You need to enable JavaScript to run this app.</noscript>  
 <div id="root"></div>  
 <!--  
 This HTML file is a template.  
 If you open it directly in the browser, you will see an empty page.  
  
 You can add webfonts, meta tags, or analytics to this file.  
 The build step will place the bundled scripts into the <body> tag.  
  
 To begin the development, run `npm start` or `yarn start`.  
 To create a production bundle, use `npm run build` or `yarn build`.  
 -->  
 </body>  
</html>

**tabmove.js**

function openPage(pageName, elmnt, color) {  
 // Hide all elements with class="tabcontent" by default \*/  
 var i, tabcontent, tablinks;  
 tabcontent = document.getElementsByClassName("tabcontent");  
 for (i = 0; i < tabcontent.length; i++) {  
 tabcontent[i].style.display = "none";  
 }  
  
 // Remove the background color of all tablinks/buttons  
 tablinks = document.getElementsByClassName("tablink");  
 for (i = 0; i < tablinks.length; i++) {  
 tablinks[i].style.backgroundColor = "";  
 }  
  
 // Show the specific tab content  
 document.getElementById(pageName).style.display = "block";  
  
 // Add the specific color to the button used to open the tab content  
 elmnt.style.backgroundColor = color;  
}  
  
// Get the element with id="defaultOpen" and click on it  
document.getElementById("defaultOpen").click();

**dayTabs.js**

import React from 'react';  
import ReactDOM from 'react-dom';  
import { Tab, Tabs } from 'react-bootstrap'  
import DayTabsRouter from './dayTabsRouter'  
import MonthTabsRouter from './monthTabsRouter'  
  
class DayTabs extends React.Component {  
 constructor(){  
 super();  
 this.state = {activeTab:''};  
 this.handleSelect = this.handleSelect.bind(this);  
}  
componentWillReceiveProps(nextProps) {  
 this.setState({activeTab:this.props.mon+'-'+nextProps.monthlyActiveTab});  
 }  
handleSelect(selectedTab) {  
 this.setState({  
 activeTab: selectedTab  
 });  
 }  
 //rendering the day tabs  
render(){  
 return <Tabs activeKey={this.state.activeTab} onSelect={this.handleSelect}>  
 <Tab eventKey={this.props.mon+'-All'} title={<DayTabsRouter tabId='Select Day' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-01'} title={<DayTabsRouter tabId='01' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-02'} title={<DayTabsRouter tabId='02' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-03'} title={<DayTabsRouter tabId='03' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-04'} title={<DayTabsRouter tabId='04' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-05'} title={<DayTabsRouter tabId='05' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-06'} title={<DayTabsRouter tabId='06' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-07'} title={<DayTabsRouter tabId='07' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-08'} title={<DayTabsRouter tabId='08' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-09'} title={<DayTabsRouter tabId='09' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-10'} title={<DayTabsRouter tabId='10' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-11'} title={<DayTabsRouter tabId='11' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-12'} title={<DayTabsRouter tabId='12' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-13'} title={<DayTabsRouter tabId='13' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-14'} title={<DayTabsRouter tabId='14' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-15'} title={<DayTabsRouter tabId='15' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-16'} title={<DayTabsRouter tabId='16' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-17'} title={<DayTabsRouter tabId='17' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-18'} title={<DayTabsRouter tabId='18' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-19'} title={<DayTabsRouter tabId='19' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-20'} title={<DayTabsRouter tabId='20' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-21'} title={<DayTabsRouter tabId='21' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-21'} title={<DayTabsRouter tabId='22' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-23'} title={<DayTabsRouter tabId='23' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-25'} title={<DayTabsRouter tabId='25' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-26'} title={<DayTabsRouter tabId='26' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-27'} title={<DayTabsRouter tabId='27' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-28'} title={<DayTabsRouter tabId='28' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-29'} title={<DayTabsRouter tabId='29' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-30'} title={<DayTabsRouter tabId='30' year={this.props.mon}/>}></Tab>  
 <Tab eventKey={this.props.mon+'-31'} title={<DayTabsRouter tabId='31' year={this.props.mon}/>}></Tab>  
 </Tabs>  
}  
}  
export default DayTabs;

**dayTabsRouter.js**

import React from 'react';  
import ReactDOM from 'react-dom';  
import { Link } from 'react-router-dom';  
class DayTabsRouter extends React.Component {  
 constructor(){  
 super();  
 this.state={style:{'font-size': '10px'}}  
 }  
render(){  
 if(this.props.tabId == 'All'){  
 return <Link to={{pathname: '/', search: '?day=All&mon='+this.props.mon}} >  
 <p style={this.state.style}>Show All</p>  
 </Link>  
 }  
else{  
 return <Link to={{pathname: '/', search: '?day='+this.props.tabId + '&mon='+this.props.mon }} >  
 <p style={this.state.style}>{this.props.tabId} {this.props.mon}</p>  
 </Link>  
 }  
}  
}  
export default DayTabsRouter;

**monthTabRouter.js**

import React from 'react';  
import ReactDOM from 'react-dom';  
import { Tab, Tabs } from 'react-bootstrap'  
import { Link } from 'react-router-dom';  
class MonthTabsRouter extends React.Component {  
constructor(){  
 super();  
 this.state={style:{'font-size': '16px'}}  
 }  
 render(){  
 return <Link to={{pathname: '/', search: '?day=All&mon='+this.props.mon }} >  
 <p style={this.state.style}>{this.props.mon}</p>  
 </Link>  
 }  
}  
export default MonthTabsRouter;

**Add.js**

import React from 'react';  
import {Button} from 'react-bootstrap';  
import Modal from 'react-modal';  
import axios from 'axios';  
import background1 from '../img/logo/addbackground.jpg';  
import {Link} from 'react-router-dom';  
var querystring = require('querystring');  
class Add extends React.Component {  
  
  
 spay;  
 dpay;  
  
 constructor() {  
 super();  
  
  
this.state = {  
 road: '',  
 amount: '',  
 day: '',  
 mon: '',  
 pay: '',  
 card: '',  
 cvc: '',  
 holdername: '',  
 phone:'',  
 pin: '',  
 nic:'',  
 email:'',  
 price:'',  
 to:'',  
 sub:'',  
 body:'',  
 discount:'',  
 messageFromServer: '',  
 modalIsOpen: false  
 }  
this.handleSelectChange = this.handleSelectChange.bind(this);  
 this.onClick = this.onClick.bind(this);  
 this.handleTextChange = this.handleTextChange.bind(this);  
 this.insertNewExpense = this.insertNewExpense.bind(this);  
 this.openModal = this.openModal.bind(this);  
 this.closeModal = this.closeModal.bind(this);  
 }  
openModal() {  
 this.setState({  
 modalIsOpen: true  
 });  
 }  
closeModal() {  
 this.setState({  
 modalIsOpen: false,  
 road: 'Negombo-Colombo',  
 amount: '',  
 day: '01',  
 mon: 'january',  
 pay: 'Sampath',  
 card: '',  
 cvc: '',  
 holdername: '',  
 phone:'',  
 pin: '',  
 nic:'',  
 email:'',  
 price:'1',  
 discount:'0',  
 to:'',  
 sub:'Train Ticket Payment Confirmation',  
 body:'Payment Successfull',  
 messageFromServer: ''  
 });  
 }  
componentDidMount() {  
 if(this.props.selectedday == 'All'){  
 this.setState({  
 day: '01'  
 });  
 }else{  
 this.setState({  
 day: this.props.selectedday  
 });  
 }  
  
 this.setState({  
 mon: this.props.selectedmon  
 });  
 }  
componentWillReceiveProps(nextProps){  
 if(this.props.selectedday == 'All'){  
 this.setState({  
 day: '01'  
 });  
 }else{  
 this.setState({  
 day: this.props.selectedday  
 });  
 }  
  
 this.setState({  
 mon:nextProps.selectedmon  
 })  
 }  
 //handle select box changes  
handleSelectChange(e) {  
 if (e.target.name == 'day') {  
 this.setState({  
 day: e.target.value  
 });  
 }  
 if (e.target.name == 'mon') {  
 this.setState({  
 mon: e.target.value  
 });  
 }  
 if (e.target.name == "road") {  
 this.setState({  
 road: e.target.value  
 });  
 }  
 if (e.target.name == "pay") {  
 this.setState({  
 pay: e.target.value  
 });  
  
  
 var optionvalue = document.getElementById("pay").value;  
 if (optionvalue == "Sampath") {  
 document.getElementById('ifSampath').style.visibility = 'visible';  
 document.getElementById('ifDialog').style.visibility = 'hidden';  
  
 this.spay = {  
 price:this.pri,  
 amount: this.state.amount,  
 road: this.state.road,  
 card: this.state.card,  
 discount: this.state.discount,  
  
 }  
  
 } else if (optionvalue == "Dialog") {  
 document.getElementById('ifDialog').style.visibility = 'visible';  
 document.getElementById('ifSampath').style.visibility = 'hidden';  
  
 this.dpay = {  
 price:this.pri,  
 amount: this.state.amount,  
 road: this.state.road,  
 phone: this.state.phone,  
  
 }  
 }else{  
 document.getElementById('ifDialog').style.visibility = 'hidden';  
 document.getElementById('ifSampath').style.visibility = 'hidden';  
 }  
  
 }  
 }  
 //handle onclick  
onClick(e) {  
 this.insertNewExpense(this);  
 }  
insertNewExpense(e) {  
 var ticket = {  
 road: e.state.road,  
 amount: e.state.amount,  
 day: e.state.day,  
 mon: e.state.mon,  
 pay: e.state.pay,  
 card: e.state.card,  
 cvc: e.state.cvc,  
 holdername: e.state.holdername,  
 phone: e.state.phone,  
 pin: e.state.pin,  
 nic: e.state.nic,  
 email: e.state.email,  
 price:'1',  
 }  
 axios.post('http://localhost:8080/ticket', ticket).then(function(response) {  
 e.setState({  
 messageFromServer: response.data  
 });  
 });  
  
 axios.get('http://localhost:8081/roads/'+ e.state.road) .then(response => {  
 this.pri= response.data.price  
  
  
 console.log(response.data.pri)  
  
  
 })  
 axios.get('http://localhost:8082/roads/'+ e.state.road) .then(response => {  
 this.pri= response.data.price  
  
  
 console.log(response.data.pri)  
  
  
 })  
  
 axios.get('http://localhost:8081/govermentnic/'+ e.state.nic) .then(response => {  
  
 console.log(response.data)  
  
  
 })  
  
  
  
  
 axios.post('http://localhost:8081/sampathpayment', this.spay).then(function(response) {  
 e.setState({  
 messageFromServer: response.data  
 });  
  
 });  
  
  
 axios.post('http://localhost:8082/dialogpayment', this.dpay).then(function(response) {  
 e.setState({  
 messageFromServer: response.data  
 });  
 });  
  
  
 var mail = {  
 to: e.state.email,  
 sub: 'Train Ticket Reservation',  
 body: 'Payment Successfull',  
 }  
 var headers = {  
 'Content-Type': 'application/json',  
 }  
 axios.post('http://localhost:8188/email/send', mail,{headers: headers}).then(function(response) {  
 e.setState({  
 messageFromServer: response.data  
 });  
 });  
  
 }  
 //handle text box changes  
handleTextChange(e) {  
  
if (e.target.name == "amount") {  
 this.setState({  
 amount: e.target.value  
 });  
 }  
 if (e.target.name == "nic") {  
 this.setState({  
 nic: e.target.value  
 });  
 }  
 if (e.target.name == "pin") {  
 this.setState({  
 pin: e.target.value  
 });  
 }  
 if (e.target.name == "phone") {  
 this.setState({  
 phone: e.target.value  
 });  
 }  
 if (e.target.name == "card") {  
 this.setState({  
 card: e.target.value  
 });  
 }  
 if (e.target.name == "cvc") {  
 this.setState({  
 cvc: e.target.value  
 });  
 }  
 if (e.target.name == "holdername") {  
 this.setState({  
 holdername: e.target.value  
 });  
 }  
 if (e.target.name == "email") {  
 this.setState({  
 email: e.target.value  
 });  
 }  
 }  
   
 //rendering the ticket booking form  
render() {  
 if(this.state.messageFromServer == ''){  
 return (  
 <div style={{backgroundImage: "url(" + background1 + ")"}}>  
  
 <Button style={{height: '50px',width: '150px', color: 'white', marginLeft:'40%',marginTop:'6%',marginBottom:'10%'} } onClick={this.openModal}>Book Ticket Now<span className="glyphicon glyphicon-plus"></span></Button>  
 <Modal  
 isOpen={this.state.modalIsOpen}  
 onRequestClose={this.closeModal}  
 contentLabel="Ticket Book"  
 className="Modal">  
<Link to={{pathname: '/', search: '?day='+this.state.day+'&mon='+this.state.mon }} style={{ textDecoration: 'none' }}>  
 <Button style={{height: '35px',width: '35px', color: 'white'} } onClick={this.closeModal}>X<span className="closebtn glyphicon glyphicon-remove"></span></Button>  
 </Link><br/>  
<fieldset>  
 <label for="road">Road:</label><select id="road" name="road" value={this.state.road} onChange={this.handleSelectChange} required>  
 <option value="Select a Road">Select a Road</option>  
 <option value="Negombo-Colombo" id="Negombo-Colombo" >Negombo-Colombo</option>  
 <option value="Negombo-Alawwa" id="Negombo-Alawwa">Negombo-Alawwa</option>  
 <option value="Abewela-Dehiwala" id="Abewela-Dehiwala">Abewela-Dehiwala</option>  
 <option value="Awissawells-Dehiwala" id="Awissawells-Dehiwala">Awissawells-Dehiwala</option>  
 <option value="Colombo-Negombo" id="Colombo-Negombo">Colombo-Negombo</option>  
 <option value="Colombo-Kollupitiya" id="Colombo-Kollupitiya">Colombo-Kollupitiya</option>  
 <option value="Trinkomalee-Colombo" id="Trinkomalee-Colombo">Trinkomalee-Colombo</option>  
  
  
 </select>  
 <label for="amount">Amount:</label><input type="number" id="amount" name="amount" value={this.state.amount} onChange={this.handleTextChange}></input>  
 <label for="day">Day:</label><select id="day" name="day" value={this.state.day} onChange={this.handleSelectChange}>  
 <option value="01" id="01">1st</option>  
 <option value="02" id="02">2nd</option>  
 <option value="03" id="03">3rd</option>  
 <option value="04" id="04">4th</option>  
 <option value="05" id="05">5th</option>  
 <option value="06" id="06">6th</option>  
 <option value="07" id="07">7th</option>  
 <option value="08" id="08">8th</option>  
 <option value="09" id="09">9th</option>  
 <option value="10" id="10">10th</option>  
 <option value="11" id="11">11th</option>  
 <option value="12" id="12">12th</option>  
 <option value="13" id="13">13th</option>  
 <option value="14" id="14">14th</option>  
 <option value="15" id="15">15th</option>  
 <option value="16" id="16">16th</option>  
 <option value="17" id="17">17th</option>  
 <option value="18" id="18">18th</option>  
 <option value="19" id="19">19th</option>  
 <option value="20" id="20">20th</option>  
 <option value="21" id="21">21st</option>  
 <option value="22" id="22">22nd</option>  
 <option value="23" id="23">23rd</option>  
 <option value="24" id="24">24th</option>  
 <option value="25" id="25">25th</option>  
 <option value="26" id="26">26th</option>  
 <option value="27" id="27">27th</option>  
 <option value="28" id="28">28th</option>  
 <option value="29" id="29">29th</option>  
 <option value="30" id="30">30th</option>  
 <option value="31" id="31">31st</option>  
  
 </select>  
 <label for="mon">Month:</label><select id="mon" name="mon" value={this.state.mon} onChange={this.handleSelectChange}>  
 <option value="1" id="1">January</option>  
 <option value="2" id="2">February</option>  
 <option value="3" id="3">March</option>  
 <option value="4" id="4">April</option>  
 <option value="5" id="5">May</option>  
 <option value="6" id="6">June</option>  
 <option value="7" id="7">July</option>  
 <option value="8" id="8">August</option>  
 <option value="9" id="9">September</option>  
 <option value="10" id="10">October</option>  
 <option value="11" id="11">November</option>  
 <option value="12" id="12">December</option>  
  
 </select>  
 <label htmlFor="email">Email:</label><input type="email" id="email" name="email" value={this.state.email}  
 onChange={this.handleTextChange}></input>  
 <label htmlFor="NIC">NIC:</label><input type="text" id="nic" name="nic" placeholder="Goverment Employees" value={this.state.nic}  
 onChange={this.handleTextChange}></input>  
 <label htmlFor="pay">Payment :</label><select id="pay" name="pay" value={this.state.pay}  
 onChange={this.handleSelectChange} required>  
 <option value="Select a Payment Method">Select a Payment Method</option>  
 <option value="Sampath" id="Sampath">Sampath</option>  
 <option value="Dialog" id="Dialog">Dialog</option>  
  
</select>  
  
 <div id="ifSampath" style={{visibility:"hidden"}}>  
 <label htmlFor="card"> Credit Card: </label><input type='text' id='card' name='card' value={this.state.card} onChange={this.handleTextChange}></input>  
 <label htmlFor="cvc"> CVC: </label><input type='text' id='cvc' name='cvc' value={this.state.cvc} onChange={this.handleTextChange}></input>  
 <label htmlFor="holdername"> Card Holder Name: </label><input type='text' id='holdername' name='holdername' value={this.state.holdername} onChange={this.handleTextChange}></input>  
 </div>  
 <div id="ifDialog" style={{visibility:"hidden"}}>  
 <label htmlFor="phone"> Phone No: </label><input type='text' id='phone' name='phone' value={this.state.phoneno} onChange={this.handleTextChange}></input>  
 <label htmlFor="pin"> PIN: </label><input type='text' id='pin' name='pin' value={this.state.pin} onChange={this.handleTextChange}></input>  
 </div>  
</fieldset>  
<div className='button-center'>  
 <br/>  
 <Button bsStyle="success" bsSize="small" onClick={this.onClick}>Ticket Book</Button>  
 </div>  
 </Modal>  
 </div>  
 )  
 }  
 else{  
 return (  
 <div>  
 <Button bsStyle="success" bsSize="small" onClick={this.openModal}><span className="glyphicon glyphicon-plus"></span></Button>  
 <Modal  
 isOpen={this.state.modalIsOpen}  
 onAfterOpen={this.afterOpenModal}  
 onRequestClose={this.closeModal}  
 contentLabel="Add ticket"  
 className="Modal">  
<div className='button-center'>  
 <h3>{this.state.messageFromServer}</h3>  
 <Link to={{pathname: '/', search: '?day='+this.state.day+'&mon='+this.state.mon}} style={{ textDecoration: 'none' }}>  
 <Button bsStyle="success" bsSize="mini" onClick={this.closeModal}>Close the Dialog</Button>  
 </Link>  
 </div>  
 </Modal>  
 </div>  
 )  
 }  
 }  
}  
export default Add;

**App.js**

import React from 'react';  
import ReactDOM from 'react-dom';  
import axios from 'axios';  
import Add from './Add';  
import { Tab, Tabs } from 'react-bootstrap';  
import MonthTabsRouter from './tabs/monthTabsRouter';  
import DayTabs from './tabs/dayTabs';  
import styles from '../CSS/App.css';  
  
export default class App extends React.Component {  
constructor() {  
 super();  
 this.state = {selectedday:'All', selectedmon: 1, data: [], activeTab:2016};  
 this.getData = this.getData.bind(this);  
 this.handleSelect = this.handleSelect.bind(this);  
 }  
componentWillReceiveProps(nextProps) {  
 if(nextProps.history.location.search){  
 var search = nextProps.history.location.search;  
 search = search.substring(1);  
 var searchObj = JSON.parse('{"' + decodeURI(search).replace(/"/g, '\\"').replace(/&/g, '","').replace(/=/g,'":"') + '"}');  
 this.setState({activeTab: parseInt(searchObj.mon)});  
 this.setState({selectedmon: searchObj.mon});  
 this.setState({selectedday: searchObj.day});  
  
  
this.getData(this, searchObj.mon, searchObj.day);  
 }else{  
 this.getData(this, 1, 'All');  
 }  
 }  
componentDidMount(){  
 this.getData(this, 1, 'All');  
 }  
handleSelect(selectedTab) {  
 this.setState({  
 activeTab: selectedTab,  
 selectedmon: selectedTab  
 });  
 }  
getData(ev, mon, day){  
 axios.get('http://localhost:8080/expense/'+mon+'/'+day)  
 .then(function(response) {  
 ev.setState({data: response.data});  
 ev.setState({selectedmon: mon});  
 ev.setState({selectedday: day});  
 });  
  
  
 }  
   
 //rendering the month tab  
render() {  
 return (  
 <div>  
 <div style={{marginTop:"5%",marginBottom:"18%"}}>  
 <Tabs activeKey={this.state.activeTab} onSelect={this.handleSelect}>  
 <Tab eventKey={1} title={<MonthTabsRouter mon='Jan' />}><DayTabs mon='1' dailyActiveTab={this.state.selectedday}/></Tab>  
 <Tab eventKey={2} title={<MonthTabsRouter mon='Feb' />}><DayTabs mon='2' dailyActiveTab={this.state.selectedday}/></Tab>  
 <Tab eventKey={3} title={<MonthTabsRouter mon='March'/>}><DayTabs mon='3' dailyActiveTab={this.state.selectedday}/></Tab>  
 <Tab eventKey={4} title={<MonthTabsRouter mon='April'/>}><DayTabs mon='4' dailyActiveTab={this.state.selectedday}/></Tab>  
 <Tab eventKey={5} title={<MonthTabsRouter mon='May'/>}><DayTabs mon='5' dailyActiveTab={this.state.selectedday}/></Tab>  
 <Tab eventKey={6} title={<MonthTabsRouter mon='June'/>}><DayTabs mon='6' dailyActiveTab={this.state.selectedday}/></Tab>  
 <Tab eventKey={7} title={<MonthTabsRouter mon='July'/>}><DayTabs mon='7' dailyActiveTab={this.state.selectedday}/></Tab>  
 <Tab eventKey={8} title={<MonthTabsRouter mon='Aug'/>}><DayTabs mon='8' dailyActiveTab={this.state.selectedday}/></Tab>  
 <Tab eventKey={9} title={<MonthTabsRouter mon='Sep'/>}><DayTabs mon='9' dailyActiveTab={this.state.selectedday}/></Tab>  
 <Tab eventKey={10} title={<MonthTabsRouter mon='Oct'/>}><DayTabs mon='10' dailyActiveTab={this.state.selectedday}/></Tab>  
 <Tab eventKey={11} title={<MonthTabsRouter mon='Nov'/>}><DayTabs mon='11' dailyActiveTab={this.state.selectedday}/></Tab>  
 <Tab eventKey={12} title={<MonthTabsRouter mon='Dec'/>}><DayTabs mon='12' dailyActiveTab={this.state.selectedday}/></Tab>  
  
 </Tabs>  
 <Add selectedday={this.state.selectedday} selectedmon={this.state.selectedmon} />  
 <table>  
  
 <tbody>  
 {  
 this.state.data.map((exp) => {  
 return <tr><td className='counterCell'></td><td className='desc-col'>{exp.road}</td><td className='button-col'>{exp.amount}</td><td className='button-col'>{exp.day}</td><td className='button-col'>{exp.mon}</td></tr>  
 })  
 }  
 </tbody>  
</table>  
 </div>  
 </div>  
 );  
 }  
}

**App.css**

.button-col {  
 width:100px;  
 text-align:center;  
}  
.desc-col {  
 width:300px;  
 text-align:left;  
}  
table {  
 counter-reset: tableCount;  
}  
.counterCell:before {  
 content: counter(tableCount);  
 counter-increment: tableCount;  
}  
.counterCell {  
 text-align: center;  
 width:50px;  
}  
.button-center {  
 text-align: center;  
}  
.Modal {  
 position: relative;  
 top: 100px;  
 left: 400px;  
 right: 20px;  
 bottom: 20px;  
 background-color: #F5F5F5;  
 width:500px;  
 border: 1px solid #000;  
 border-radius: 4px;  
 padding: 20px;  
 }  
.Overlay {  
 position: fixed;  
 top: 0;  
 left: 0;  
 right: 0;  
 bottom: 0;  
 background-color: rebeccapurple;  
 }  
.button-center {  
 text-align: center;  
 }  
label{  
 display:inline-block;  
 width:200px;  
 margin-right:30px;  
 text-align:right;  
 }  
fieldset{  
 border:none;  
 margin:0px auto;  
 }  
.closebtn{  
 float:right;  
 }

**Detailsindexjs.css**

/\* Style tab links \*/  
.tablink {  
 background-color: #555;  
 color: white;  
 float: left;  
 border: none;  
 outline: none;  
 cursor: pointer;  
 padding: 14px 16px;  
 font-size: 17px;  
 width: 25%;  
}  
  
.tablink:hover {  
 background-color: #777;  
}  
  
/\* Style the tab content (and add height:100% for full page content) \*/  
.tabcontent {  
 color: white;  
 display: none;  
 padding: 100px 20px;  
 height: 100%;  
}  
  
#Home {background-color: red;}  
#News {background-color: green;}  
#Contact {background-color: blue;}  
#About {background-color: orange;}

**Detailsindexjs.jsx**

import React, { Component } from 'react';  
  
 import '../CSS/Detailsindexjs.css';  
import { CardDeck,Card} from 'react-bootstrap';  
  
import img1 from '../img/tabimg/003.png';  
import img2 from '../img/tabimg/001.png';  
import img3 from '../img/tabimg/002.png';  
import img4 from '../img/tabimg/005.png';  
import img5 from '../img/tabimg/004.png';  
  
export default class Detailsindexjs extends Component {  
 //cards on the home page  
 render() {  
 return (  
 <div className="DI" >  
 <br/>  
 <br/>  
 <CardDeck>  
 <Card>  
 <Card.Img variant="top" src={img1} />  
 <Card.Body>  
 <Card.Title>Easy Payment Methods</Card.Title>  
 <Card.Text>  
  
 </Card.Text>  
 </Card.Body>  
  
 </Card>  
 <Card>  
 <Card.Img variant="top" src={img2} />  
 <Card.Body>  
 <Card.Title>Secure Reservation</Card.Title>  
 <Card.Text>  
  
 </Card.Text>  
 </Card.Body>  
 </Card>  
 <Card>  
 <Card.Img variant="top" src={img3} />  
 <Card.Body>  
 <Card.Title>Fast and Reliable</Card.Title>  
 <Card.Text>  
  
 </Card.Text>  
 </Card.Body>  
 </Card>  
 <Card>  
 <Card.Img variant="top" src={img4}/>  
 <Card.Body>  
 <Card.Title>User Friendly</Card.Title>  
 <Card.Text>  
  
 </Card.Text>  
 </Card.Body>  
 </Card>  
 <Card>  
 <Card.Img variant="top" src={img5} />  
 <Card.Body>  
 <Card.Title>Mobile Payments</Card.Title>  
 <Card.Text>  
  
 </Card.Text>  
 </Card.Body>  
 </Card>  
 </CardDeck>  
 <br/>  
 <br/>  
 </div>  
 );  
  
  
  
 }  
  
  
  
}

**Footer.jsx**

import React, { Component } from 'react';  
  
  
import { Card} from 'react-bootstrap';  
  
export default class Footer extends Component {  
   
 //rendering the footer  
 render() {  
 return (  
 <div className="fo" >  
 <Card className="text-center" bg="secondary" text="white">  
  
 <Card.Body>  
 <Card.Text>  
 The Best Train Ticket Reservation Website.  
 </Card.Text>  
 </Card.Body>  
 <Card.Footer>&copy; {new Date().getFullYear()} Copyright: IT17056212</Card.Footer>  
 </Card>  
 </div>  
 );  
 }  
  
  
}

**ImageSlide.jsx**

import React, { Component } from 'react';  
import slide1 from '../img/slide/1.jpg';  
import slide2 from '../img/slide/2.jpg';  
import slide3 from '../img/slide/3.jpg';  
  
import { Carousel} from 'react-bootstrap';  
  
export default class ImageSlide extends Component {  
   
 //rendering the image slides on the home page  
 render() {  
 return (  
 <div className="IM" >  
 <Carousel>  
 <Carousel.Item>  
 <img  
 className="d-block w-100"  
 src={slide1}  
 alt="First slide"  
 />  
 <Carousel.Caption>  
 <h3>Book Your Ticket Today </h3>  
 <p>Reserve train tickets with convenience and ease any time.</p>  
 </Carousel.Caption>  
 </Carousel.Item>  
 <Carousel.Item>  
 <img  
 className="d-block w-100"  
 src={slide2}  
 alt="Third slide"  
 />  
  
 <Carousel.Caption>  
 <h3>Fast and Secure</h3>  
 <p>Best way to book train tickets in advance.</p>  
 </Carousel.Caption>  
 </Carousel.Item>  
 <Carousel.Item>  
 <img  
 className="d-block w-100"  
 src={slide3}  
 alt="Third slide"  
 />  
  
 <Carousel.Caption>  
 <h3>User Friendly</h3>  
 <p>Safe, friendly and remarkably hassle-free.</p>  
 </Carousel.Caption>  
 </Carousel.Item>  
 </Carousel>  
 </div>  
 );  
 }  
}

**Navbarjsx.jsx**

import React, { Component } from 'react';  
import icon from '../img/logo/logo.jpg';  
import { Button,Navbar,Nav ,Form,FormControl,NavDropdown} from 'react-bootstrap';  
import { Link } from 'react-router-dom';  
  
  
export default class Navbarjsx extends Component {  
   
 //rendering the navigation bar of the home page  
 render() {  
 return (  
 <div className="Na" >  
 <Navbar bg="light" variant="light" fixed="top">  
 <Navbar.Brand href="#home">  
 <img  
 width={150}  
 height={64}  
 className="d-inline-block align-top"  
 src={icon}  
 alt="Logo ICON"/>  
  
 </Navbar.Brand>  
 <Nav className="mr-auto">  
 <Link to="/"> <Nav.Link href="#home">Home</Nav.Link></Link>  
  
 </Nav>  
 <Form inline>  
  
 <Link to="/App"> <Button variant="outline-primary">Book Ticket</Button></Link>  
 </Form>  
 </Navbar>  
 </div>  
 );  
 }  
  
  
}

**App.js**

import React, { Component } from 'react';  
import './App.css';  
import { Button,ButtonToolbar } from 'react-bootstrap';  
import RouterRoutes from "./RouterRoutes";  
import Footer from "./JSX/Footer"  
import { BrowserRouter } from 'react-router-dom';  
import Navbarjsx from "./JSX/Navbarjsx";  
  
  
class App extends Component {  
 render() {  
 return (  
 <div className="App" >  
  
 <Navbarjsx/>  
 <RouterRoutes />  
  
 <Footer/>  
  
 </div>  
 );  
 }  
}  
  
export default App;

**Home.jsx**

import React, { Component } from 'react';  
  
  
import { Card} from 'react-bootstrap';  
  
import ImageSlide from "./JSX/ImageSlide";  
import Detailsindexjs from "./JSX/Detailsindexjs";  
  
export default class Home extends Component {  
 render() {  
 return (  
 <div>  
 <ImageSlide/>  
 <Detailsindexjs/>  
 </div>  
 );  
 }  
  
  
}

**index.js**

import React from 'react';  
import ReactDOM from 'react-dom';  
import './index.css';  
import App from './App';  
import \* as serviceWorker from './serviceWorker';  
import {BrowserRouter} from "react-router-dom";  
  
ReactDOM.render( <BrowserRouter><App /></BrowserRouter>, document.getElementById('root'));  
  
serviceWorker.unregister();

**RouterRoutes.jsx**

import React from 'react';  
import { BrowserRouter, Route, Switch, Redirect, withRouter } from 'react-router-dom';  
  
import Home from './Home';  
import App from './components/App';  
  
//Handles the page routing  
const RouterRoutes = () => (  
 <Switch>  
 <Route exact path="/" component={withRouter(Home)}/>  
 <Route exact path="/App" component={withRouter(App)} />  
  
 </Switch>  
  
)  
  
export default RouterRoutes;

**serviceWorker.js**

// This optional code is used to register a service worker.  
// register() is not called by default.  
  
// This lets the app load faster on subsequent visits in production, and gives  
// it offline capabilities. However, it also means that developers (and users)  
// will only see deployed updates on subsequent visits to a page, after all the  
// existing tabs open on the page have been closed, since previously cached  
// resources are updated in the background.  
  
// To learn more about the benefits of this model and instructions on how to  
// opt-in, read https://bit.ly/CRA-PWA  
  
const isLocalhost = Boolean(  
 window.location.hostname === 'localhost' ||  
 // [::1] is the IPv6 localhost address.  
 window.location.hostname === '[::1]' ||  
 // 127.0.0.1/8 is considered localhost for IPv4.  
 window.location.hostname.match(  
 /^127(?:\.(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)){3}$/  
 )  
);  
  
export function register(config) {  
 if (process.env.NODE\_ENV === 'production' && 'serviceWorker' in navigator) {  
 // The URL constructor is available in all browsers that support SW.  
 const publicUrl = new URL(process.env.PUBLIC\_URL, window.location.href);  
 if (publicUrl.origin !== window.location.origin) {  
 // Our service worker won't work if PUBLIC\_URL is on a different origin  
 // from what our page is served on. This might happen if a CDN is used to  
 // serve assets; see https://github.com/facebook/create-react-app/issues/2374  
 return;  
 }  
  
 window.addEventListener('load', () => {  
 const swUrl = `${process.env.PUBLIC\_URL}/service-worker.js`;  
  
 if (isLocalhost) {  
 // This is running on localhost. Let's check if a service worker still exists or not.  
 checkValidServiceWorker(swUrl, config);  
  
 // Add some additional logging to localhost, pointing developers to the  
 // service worker/PWA documentation.  
 navigator.serviceWorker.ready.then(() => {  
 console.log(  
 'This web app is being served cache-first by a service ' +  
 'worker. To learn more, visit https://bit.ly/CRA-PWA'  
 );  
 });  
 } else {  
 // Is not localhost. Just register service worker  
 registerValidSW(swUrl, config);  
 }  
 });  
 }  
}  
  
function registerValidSW(swUrl, config) {  
 navigator.serviceWorker  
 .register(swUrl)  
 .then(registration => {  
 registration.onupdatefound = () => {  
 const installingWorker = registration.installing;  
 if (installingWorker == null) {  
 return;  
 }  
 installingWorker.onstatechange = () => {  
 if (installingWorker.state === 'installed') {  
 if (navigator.serviceWorker.controller) {  
 // At this point, the updated precached content has been fetched,  
 // but the previous service worker will still serve the older  
 // content until all client tabs are closed.  
 console.log(  
 'New content is available and will be used when all ' +  
 'tabs for this page are closed. See https://bit.ly/CRA-PWA.'  
 );  
  
 // Execute callback  
 if (config && config.onUpdate) {  
 config.onUpdate(registration);  
 }  
 } else {  
 // At this point, everything has been precached.  
 // It's the perfect time to display a  
 // "Content is cached for offline use." message.  
 console.log('Content is cached for offline use.');  
  
 // Execute callback  
 if (config && config.onSuccess) {  
 config.onSuccess(registration);  
 }  
 }  
 }  
 };  
 };  
 })  
 .catch(error => {  
 console.error('Error during service worker registration:', error);  
 });  
}  
  
function checkValidServiceWorker(swUrl, config) {  
 // Check if the service worker can be found. If it can't reload the page.  
 fetch(swUrl)  
 .then(response => {  
 // Ensure service worker exists, and that we really are getting a JS file.  
 const contentType = response.headers.get('content-type');  
 if (  
 response.status === 404 ||  
 (contentType != null && contentType.indexOf('javascript') === -1)  
 ) {  
 // No service worker found. Probably a different app. Reload the page.  
 navigator.serviceWorker.ready.then(registration => {  
 registration.unregister().then(() => {  
 window.location.reload();  
 });  
 });  
 } else {  
 // Service worker found. Proceed as normal.  
 registerValidSW(swUrl, config);  
 }  
 })  
 .catch(() => {  
 console.log(  
 'No internet connection found. App is running in offline mode.'  
 );  
 });  
}  
  
export function unregister() {  
 if ('serviceWorker' in navigator) {  
 navigator.serviceWorker.ready.then(registration => {  
 registration.unregister();  
 });  
 }  
}