

## Rent A Car

### SUMMARY

The systems consist of 2 user roles. In this application, user can look for a car based on the time and date he wants and can book a car on rent for that selected duration.

Admin can enter new car and can also track the online activity of each and every car since it's starting.

#### 1. **ADMIN:**

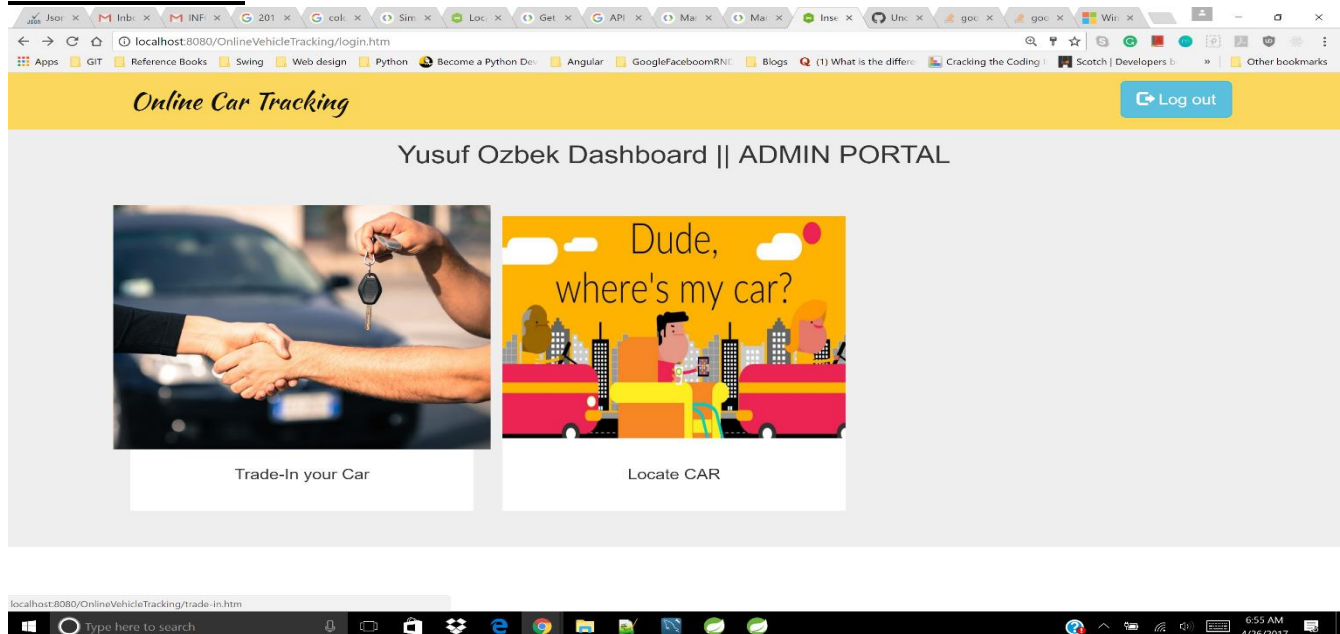
- Admin can enter new car into the system. He will enter car details and put the car on rent for customers.
- Admin can also locate any car on google maps.
- He can trace the most recent location of a car.
- He can also track all the location of the car since car was put into the system.

#### 2. **USER:**

- User can search for cars on rent and he can book the car based on the duration he wants and can search for appropriate car.
- He can see the details of the car.
- Customer can also see images of the car.
- On successful booking, booking reference number will be generated and will be emailed to the customer for booking a car on rent.

---

### SCREENSHOTS



Online Car Tracking [Log out](#)

### Vehicle Info

<b>Model:</b> CR-V	<b>Company:</b> Honda	<b>Color:</b> Matt Black	<b>Seater:</b> 7
<b>Mileage:</b> 1000	<b>Purchase Year:</b> 04/01/2017	<b>Per Hour Rate:</b> \$ 100	<b>Made in year:</b> 04/01/2017
<b>Number Plate:</b> CR-HO-4444	<b>Comments:</b> Excellent	<b>AC:</b> <input checked="" type="radio"/> Yes <input type="radio"/> No	<b>Gear type:</b> <input checked="" type="radio"/> Automatic <input type="radio"/> Manual
<b>Trade-In Type:</b> Sell	<b>Fuel Type:</b> <input checked="" type="radio"/> Gasoline <input type="radio"/> Diesel	<b>Assign station:</b> Northeastern University, 360 Huntli	

[Submit](#)

Online Car Tracking [Image uploaded successfully](#)




### Upload Images

Select files from your computer

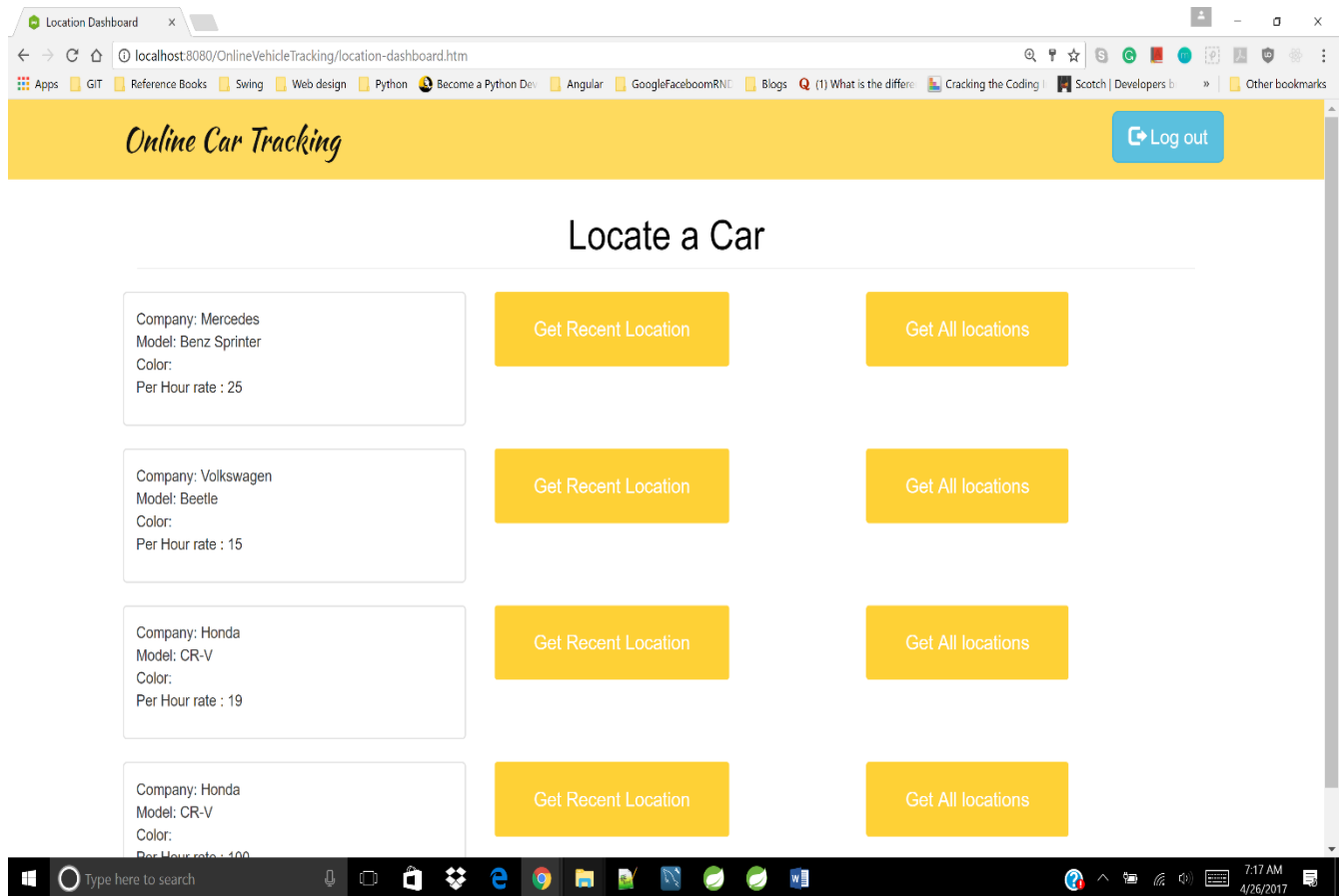
[Choose Files](#) No file chosen [Upload](#)

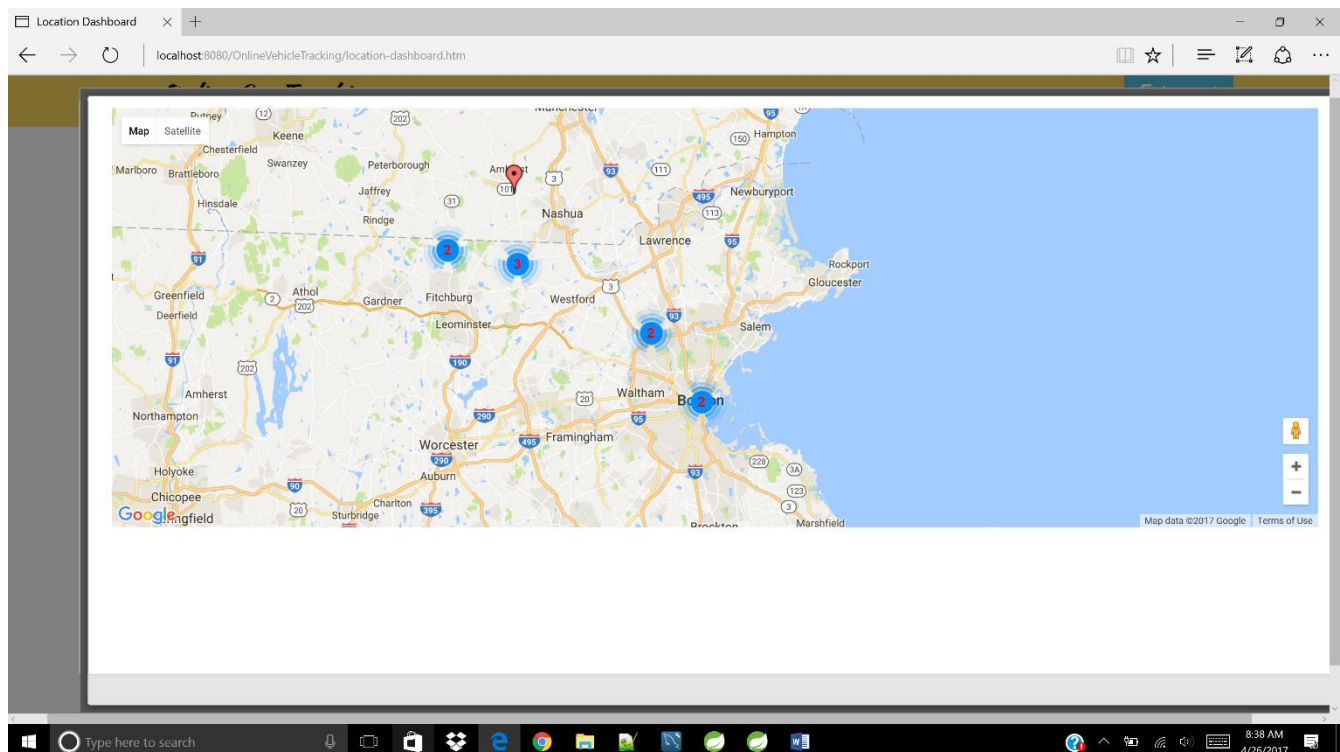
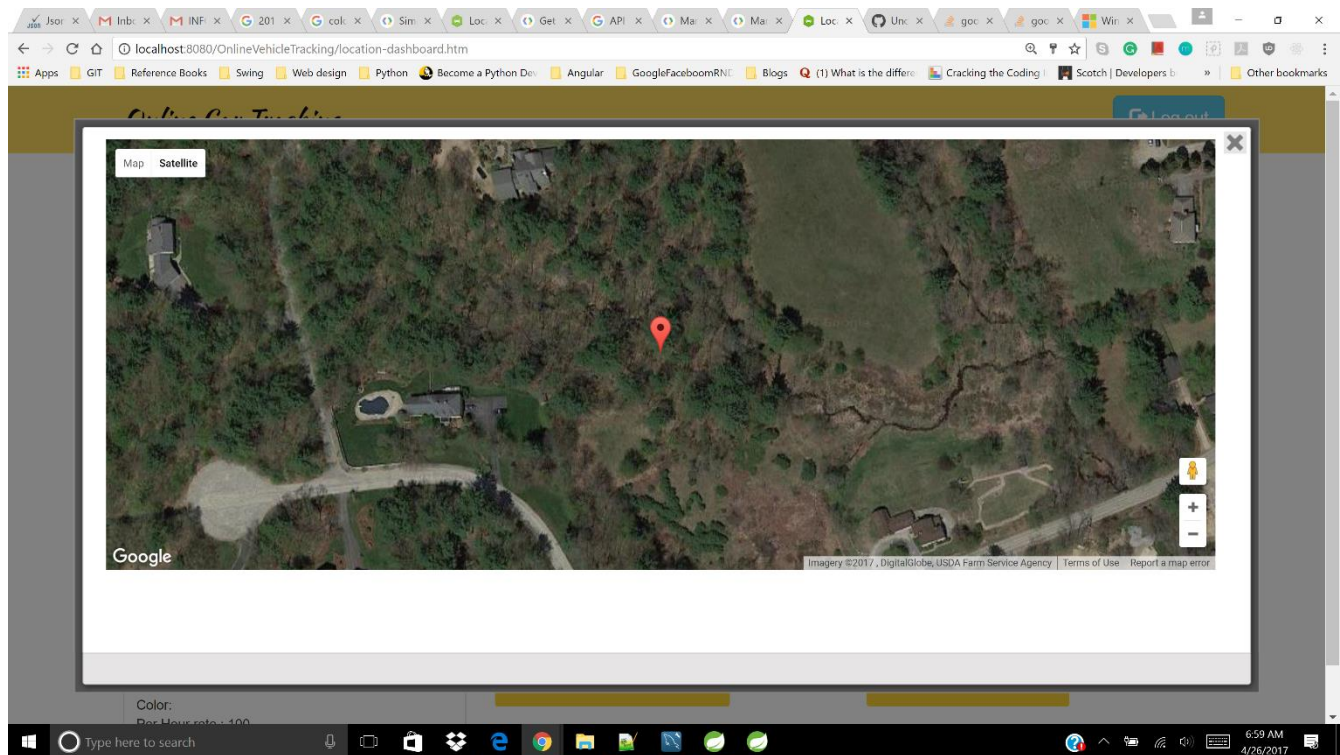
Or drag and drop files below

Just drag and drop files here

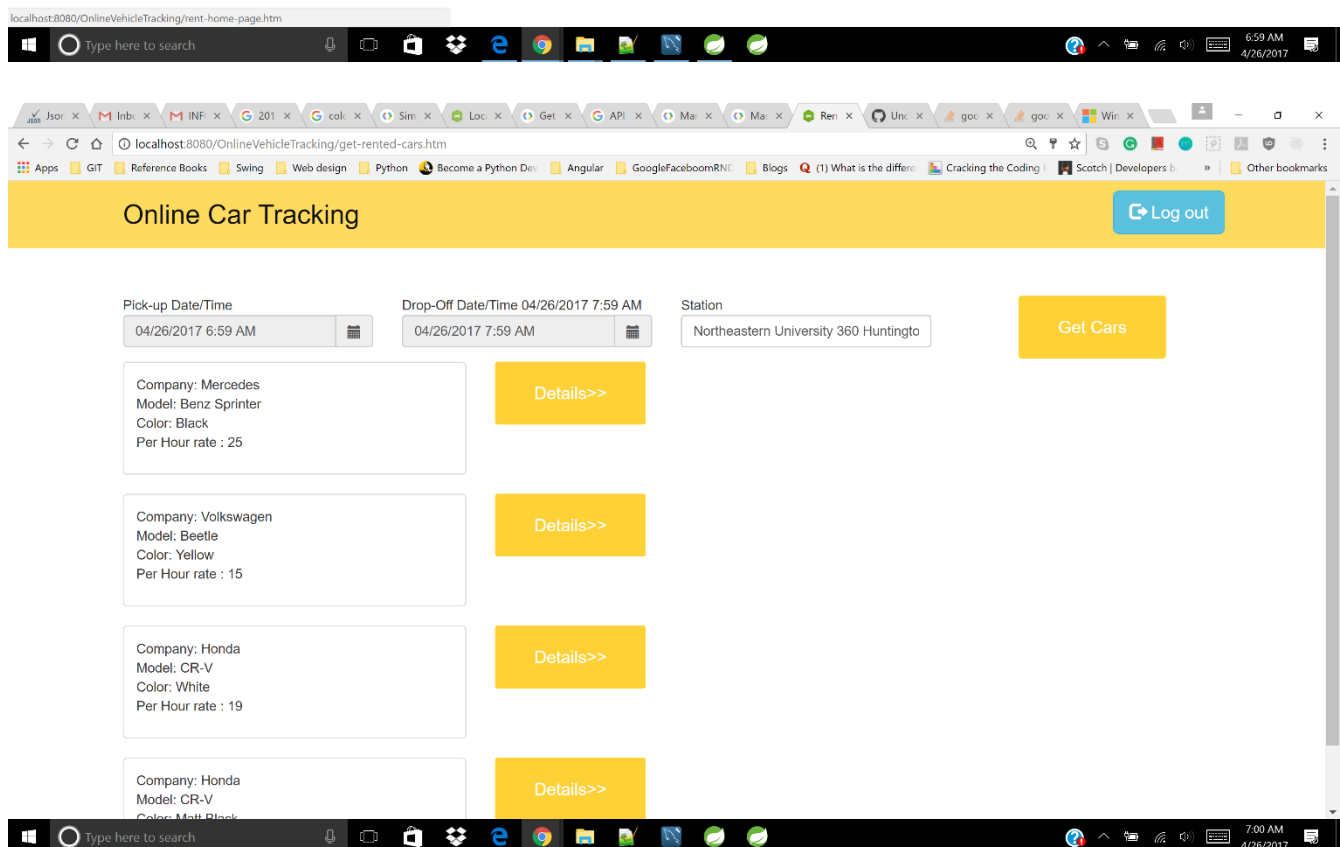
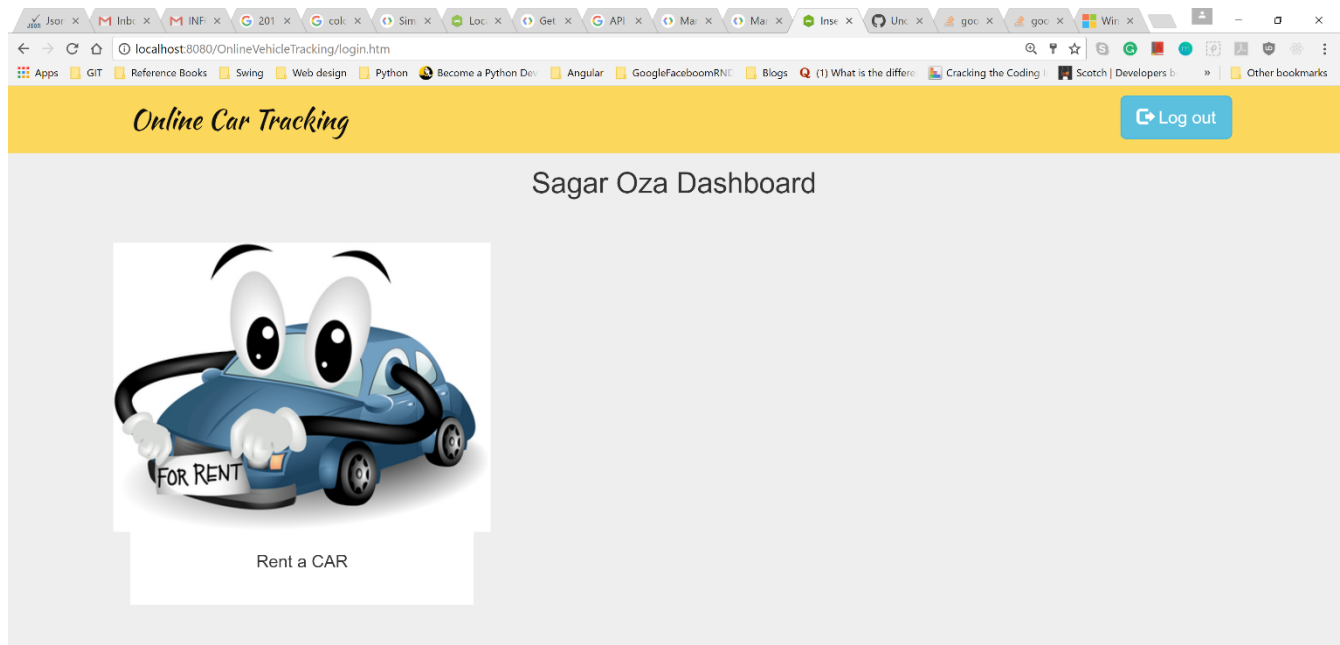


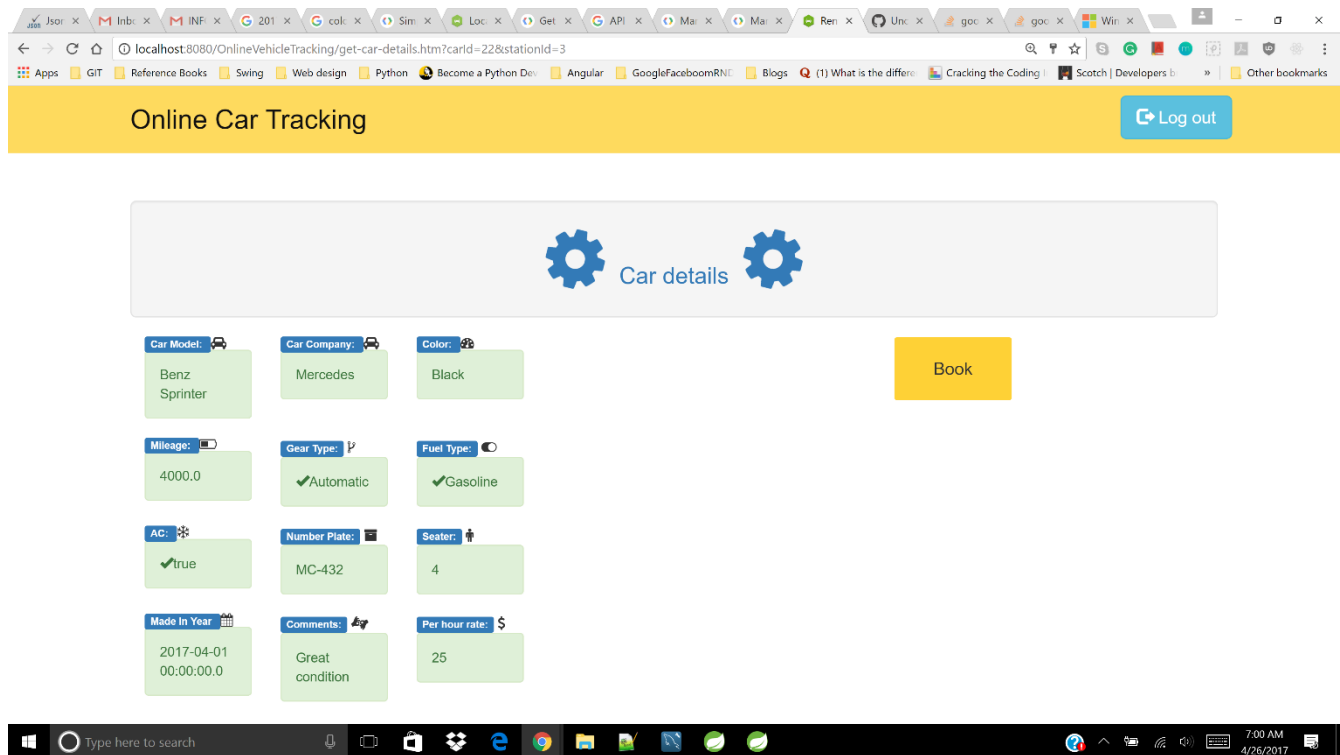
[Confirm Trade](#)





## USER VIEW





On Booking , a email will be triggered to user's emailId and a booking reference number will be emailed to him

### 3. CONTROLLER CODE STARTS HERE

#### 1. LOGINCONTROLLER

```
package com.neu.webtools.controller;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpSession;
```

```
import org.hibernate.SessionFactory;
```

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

```
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 org.springframework.web.servlet.ModelAndView;

import com.neu.webtools.beans.User;
import com.neu.webtools.constants.Constants;
import com.neu.webtools.dao.ILoginDao;

@RestController
public class LoginController {
    @Autowired
    ILoginDao loginDao;

    @RequestMapping(value = "/login.htm", method = RequestMethod.GET)
    public ModelAndView redirectLogin(HttpServletRequest request) {
        System.out.println("Inside redirectLogin");
        return new ModelAndView("login");
    }

    @RequestMapping(value = "/register.htm", method = RequestMethod.GET)
    public ModelAndView registerGet(HttpServletRequest request) {
        System.out.println("Inside register");
        return new ModelAndView("register");
    }

    @RequestMapping(value = "/add.htm", method = RequestMethod.POST)
    public ModelAndView registerUser(HttpServletRequest request) {
        System.out.println("Inside register Post");
        String firstName = request.getParameter("fname");
```

```
String lastName = request.getParameter("lname");
String username = request.getParameter("username");
String password = request.getParameter("password");
String emailId = request.getParameter("emailId");
String phoneNo = request.getParameter("phoneNo");

User user = new User();
user.setFirstName(firstName);
user.setLastName(lastName);
user.setUsername(username);
user.setPassword(password);
user.setEmailId(emailId);
user.setPhoneNo(phoneNo);
user.setRole(Constants.NORMAL_USER);

boolean isUser = loginDao.registerUser(user);
return new ModelAndView("login");
}

@RequestMapping(value = "/login.htm", method = RequestMethod.POST)
public ModelAndView checkUser(HttpServletRequest request) {
    System.out.println("Inside login Post");
    String username = request.getParameter("username");
    String password = request.getParameter("password");

    User user = loginDao.checkUser(username, password);
    System.out.println("User: " + user);
    if (user == null) {
```



```
        System.out.println("in user null");
        request.setAttribute("loginFailedMessage", "Username or password is
incorrect!!");
    }
    HttpSession session = request.getSession();
    session.setAttribute("userSession", user);
    return new ModelAndView("dashboard");
}

@RequestMapping(value = "/logout.htm", method = RequestMethod.GET)
public ModelAndView logout(HttpServletRequest request) {
    HttpSession session = request.getSession(false);
    if (session == null) {
        return new ModelAndView("login");
    } else {
        session.invalidate();
        return new ModelAndView("login");
    }
}
}
```

## 2. **TRADEIN CONTROLLER**

```
package com.neu.webtools.controller;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
```

```
import javax.servlet.http.HttpServletRequest;

import org.apache.commons.mail.DefaultAuthenticator;
import org.apache.commons.mail.Email;
import org.apache.commons.mail.EmailException;
import org.apache.commons.mail.SimpleEmail;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.beans.propertyeditors.CustomDateEditor;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.InitBinder;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.servlet.ModelAndView;

import com.google.gson.Gson;
import com.neu.webtools.beans.Car;
import com.neu.webtools.beans.Station;
import com.neu.webtools.beans.User;
import com.neu.webtools.dao.IRentDao;
import com.neu.webtools.dao.ITradeInDao;
import com.neu.webtools.validator.CarTradingValidator;

@RestController
public class TradeInController
{
    @Autowired
    @Qualifier("carValidator")
    CarTradingValidator carValidator;

    @Autowired
    public ITradeInDao tradeIndao;

    @Autowired
    public IRentDao rentDao;

    @InitBinder
    private void initBinder(WebDataBinder binder)
    {
        binder.setValidator(carValidator);
    }
}
```

```

        try
        {
            SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd"); //yyyy-MM-dd'THH:mm:ssZ example
            dateFormat.setLenient(true);
            binder.registerCustomEditor(Date.class, new CustomDateEditor(dateFormat, true));
            //true because it allows empty values
        }
        catch(Exception exp)
        {
            System.out.println("Dates cannot be empty");
        }
    }

    @RequestMapping(value="/trade-in.htm" , method=RequestMethod.GET)
    public ModelAndView getTradeInPage(HttpServletRequest request,
    @ModelAttribute("car") Car car, BindingResult result)
    {
        ArrayList<Station> stationList = (ArrayList<Station>)rentDao.getStations("");
        request.getSession().setAttribute("stationListDropDown", stationList);
        return new ModelAndView("trade-in");
    }

    @RequestMapping(value="/trade-in-car.htm" , method=RequestMethod.POST)
    public ModelAndView submitTradingDetails(HttpServletRequest request,
    @ModelAttribute("car") Car car, BindingResult result) throws ParseException
    {
        carValidator.validate(car, result);

        //System.out.println(imageFile.getSize());
        /*if(result.hasFieldErrors("madeInYear") ||
        result.hasFieldErrors("purchaseYear"))
        {
            FieldError a = result.getFieldError("madeInYear");
            boolean isError =
            result.getFieldError("madeInYear").getDefaultMessage().contains("Unparseable date");
            result.reject("madeInYear");
            String madeInYearJugaad = request.getParameter("madeInYear");
            Date madeInYearPared = new
            SimpleDateFormat("yyyyMMdd").parse(madeInYearJugaad);
            String formattedDate = new SimpleDateFormat("yyyy-MM-dd").format(madeInYearPared);

```

```
//car.setMadeInYear();

String purchaseYearJugaad = request.getParameter("purchaseYear");
Date purchaseYearParsed = new
SimpleDateFormat("yyyyMMdd").parse(purchaseYearJugaad);
String formattedDateP = new SimpleDateFormat("yyyy-MM-
dd").format(purchaseYearJugaad);
car.setPurchaseYear(new Date(formattedDateP));
}*/
if (result.hasErrors())
{
    //return new ModelAndView("uploadCarImage","car", car);
    return new ModelAndView("trade-in", "car", car);
}

else
{
    Car savedCar = tradeIndao.saveCarDetails(car);
    if(savedCar.getCarId() != 0)
    {
        request.getSession().setAttribute("savedCar", savedCar);
        return new ModelAndView("uploadCarImage","car", car);
    }
    else
    {
        request.setAttribute("persistingError", "There was an error while
saving car details!! Please try again");
        return new ModelAndView("trade-in","car", car);
    }
}

}

@RequestMapping(value="/trade-confirm.htm" , method=RequestMethod.GET)
public ModelAndView confirmTrade(HttpServletRequest request) throws
EmailException
{
    Car savedCar = (Car)request.getSession().getAttribute("savedCar");
    //User user = (User)request.getSession().getAttribute("userSession");
    try
    {
        Email email = new SimpleEmail();
```

email.setHostName("smtp.googlemail.com");//If a server is capable of sending email, then you don't need the authentication. In this case, an email server needs to be running on that machine. Since we are running this application on the localhost and we don't have a email server, we are simply asking gmail to relay this email.

```
email.setSmtpPort(465);
email.setAuthenticator(new
DefaultAuthenticator("webtools001222126Oza@gmail.com", "webtools"));
email.setSSLonConnect(true);
email.setFrom("oza.s@husky.neu.edu");//This email will appear in the from field of
the sending email. It doesn't have to be a real email address.This could be used for
phishing/spoofing!
email.setSubject("Car Traded | Car details");
email.setMsg("You have successfully registered a car into the system! \n Your car
number is " + savedCar.getNumberPlate());
```

```
email.addTo(((User)request.getSession().getAttribute("userSession")).getEmailId());//Will
come from the database
```

```
email.send();
System.out.println("Email sent succesfully!!");
return new ModelAndView("trade-success-page");
}
catch(Exception e)
{
    System.out.println("There was error while sending email!!");
    return new ModelAndView("trade-success-page");
}
```

```
}
```

```
@RequestMapping(value="/dashboard.htm" , method=RequestMethod.GET)
```

```
public ModelAndView redirectdashboard(HttpServletRequest request)
```

```
{
```

```
    System.out.println("Inside dashboard");
    return new ModelAndView("dashboard");
```

```
}
```

```
}
```

### 3. **IMAGECONTROLLER**

```
package com.neu.webtools.controller;
```

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.List;

import javax.servlet.ServletContext;
import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.util.FileCopyUtils;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.ModelAndView;

import com.neu.webtools.beans.Car;
import com.neu.webtools.beans.Images;
import com.neu.webtools.dao.IImageDao;

@RestController
public class ImageController
{
    @Autowired
    public IImageDao imageDao;

    @Autowired
    public ServletContext servletContext;

    @RequestMapping(value="/upload.htm" , method=RequestMethod.POST)
    public ModelAndView uploadImages(HttpServletRequest request,
    @RequestParam("images") MultipartFile[] files) throws IOException
    {
        int recordsInserted = 0;
        Car savedCar = (Car)request.getSession().getAttribute("savedCar");
        String contextPath = servletContext.getRealPath(File.separator);
        File carPhotosDirectory = new File(contextPath + "/car_photos" );//+
        savedCar.getCarId());
```

```

        carPhotosDirectory.mkdir();
        File carDirectory = new File(carPhotosDirectory.toString() + "/car_" +
savedCar.getCarId());
        ArrayList<Images> imageList = new ArrayList<Images>();

        if(!carDirectory.exists())
        {
            carDirectory.mkdir();
            System.out.println(carDirectory.mkdir() + " " +
carDirectory.getAbsolutePath());
        }
        for(MultipartFile file : files)
        {
            InputStream inputStream = file.getInputStream();
            OutputStream outputStream = new
FileOutputStream(carDirectory.toString().concat("/") + file.getOriginalFilename());

            FileCopyUtils.copy(inputStream, outputStream);
            Images image =
setImageObject(carDirectory.toString().concat(File.separator
file.getOriginalFilename()).toString(), savedCar, 6);//savedCar.getCarId());
            System.out.println("File name: " + image.getImagePath());
            imageList.add(image);
        }
        recordsInserted = imageDao.insertImages(imageList);
        System.out.println(recordsInserted + "images submitted successfully");
        request.setAttribute("imageUpload", "true");
        request.setAttribute("recordsInserted", recordsInserted );
        request.setAttribute("imageList", imageList);
        return new ModelAndView("uploadCarImage");
    }

    private Images setImageObject(String imagePath, Car savedcar, int savedCarId)
    {
        Images image = new Images();
        image.setImagePath(imagePath.substring(imagePath.indexOf("car_photos"),
imagePath.length()));
        image.setDeleteFlag("F");
        image.setModifiedOn(new Timestamp(System.currentTimeMillis()));
        image.setCreatedOn(new Timestamp(System.currentTimeMillis()));
        image.setCar(savedcar);
        return image;
    }

```

```
}
```

#### 4. LOCATION CONTROLLER

```
package com.neu.webtools.controller;

import java.util.ArrayList;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.servlet.ModelAndView;

import com.google.gson.Gson;
import com.neu.webtools.beans.Car;
import com.neu.webtools.beans.CarLocation;
import com.neu.webtools.dao.ILocationDao;

@RestController
public class LocationController
{
    @Autowired
    ILocationDao locationDao;

    @RequestMapping(value="/location-dashboard.htm" ,
method=RequestMethod.GET)
    public ModelAndView getLocationDashboard(HttpServletRequest request)
    {
        ArrayList<Car> cars = locationDao.getListOfCars();
        request.getSession().setAttribute("cars", cars);
        return new ModelAndView("location-dashboard");
    }

    @RequestMapping(value="/getRecentCarLocation.htm" ,
method=RequestMethod.GET)
    public String getRecentCarLocation(HttpServletRequest request)
    {
        String carId = request.getParameter("carId");
```



```
        CarLocation carLocation =
locationDao.getRecentCarLocation(carId);
        request.setAttribute("carLocation", carLocation);
        return new Gson().toJson(carLocation);
    }

    @RequestMapping(value="/map.htm" , method=RequestMethod.GET)
    public ModelAndView map(HttpServletRequest request)
    {
        return new ModelAndView("map");
    }

    @RequestMapping(value="/getAllLocations.htm" ,
method=RequestMethod.GET)
    public String getAllLocations(HttpServletRequest request)
    {
        String carId = request.getParameter("carId");
        ArrayList<CarLocation> carLocations =
locationDao.getAllLocations(carId);
        request.setAttribute("carLocations", carLocations);
        return new Gson().toJson(carLocations);
    }

    @RequestMapping(value="/mapForAllLocation.htm" ,
method=RequestMethod.GET)
    public ModelAndView mapForAllLocation(HttpServletRequest request)
    {
        return new ModelAndView("mapForAllLocation");
    }
}
```

## 5. RENT CONTROLLER

```
package com.neu.webtools.controller;

import java.sql.Timestamp;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
```

```
import org.apache.commons.mail.EmailException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.servlet.ModelAndView;

import com.google.gson.Gson;
import com.neu.webtools.beans.Car;
import com.neu.webtools.beans.RentalTransaction;
import com.neu.webtools.beans.Station;
import com.neu.webtools.beans.User;
import com.neu.webtools.dao.IRentDao;

@RestController
public class RentController
{
    @Autowired
    public IRentDao rentDao;

    @Autowired
    IEmailService emailService;

    @RequestMapping(value="/rent-home-page.htm" , method=RequestMethod.GET)
    public ModelAndView getCarsForRents(HttpServletRequest request)
    {
        return new ModelAndView("car-rent");
    }

    @RequestMapping(value="/getStations.htm" , method=RequestMethod.GET)
    public String getStations(HttpServletRequest request)
    {
        String stationReq = request.getParameter("station");
        List<Station> stationList = rentDao.getStations(stationReq);
        request.getSession().setAttribute("stationList", stationList);
        Gson json = new Gson();
        return json.toJson(stationList).toString();
    }

    @RequestMapping(value="/get-rented-cars.htm", method=RequestMethod.POST)
    public ModelAndView getCarForRent(HttpServletRequest request) throws
    ParseException
    {
```

```

        String pickUpDate = request.getParameter("pickUpDate");
        String dropOffdate = request.getParameter("dropOffDate");
        String station = request.getParameter("station");
        String stationId = request.getParameter("stationId");
        DateFormat inputFormat = new SimpleDateFormat("MM/dd/yyyy hh:mm a");
        DateFormat outputFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        String                startDateTime                =
outputFormat.format(inputFormat.parse(pickUpDate.concat(":00")));
        String                endDateTime                =
outputFormat.format(inputFormat.parse(dropOffdate.concat(":00")));
        ArrayList<Car> carList = rentDao.getRentedCars(startDateTime, endDateTime,
station);
        request.getSession().setAttribute("rentCarList", carList);
        request.getSession().setAttribute("pickUpDate", pickUpDate);
        request.getSession().setAttribute("dropOffdate", dropOffdate);
        request.getSession().setAttribute("station", station);
        request.getSession().setAttribute("startDateTime", startDateTime);
        request.getSession().setAttribute("enddateTime", endDateTime);
        request.setAttribute("stationId", stationId);
        //request.getSession().setAttribute("rentCarJson", new Gson().toJson(carList));
        System.out.println(carList);
        return new ModelAndView("car-rent");
    }

    @RequestMapping(value="get-car-details.htm", method=RequestMethod.GET)
    public ModelAndView getCarDetails(HttpServletRequest request)
    {
        String stationId = request.getParameter("stationId");
        String carId = request.getParameter("carId");
        Station stationBean = null;
        ArrayList<Car> carList =
        (ArrayList<Car>)request.getSession().getAttribute("rentCarList");
        for(Car car: carList)
        {
            if(car.getCarId() == Integer.parseInt(carId))
            {
                request.getSession().setAttribute("carSelectedDetails", car);
                break;
            }
        }

        for(Station st
        (ArrayList<Station>)request.getSession().getAttribute("stationList"))

```

```

        {
            if(st.getStationId() == Integer.parseInt(stationId))
            {
                stationBean = st;
                break;
            }
        }
        request.getSession().setAttribute("stationBean", stationBean);
        return new ModelAndView("car-details");
    }

    @RequestMapping(value="book-car.htm", method=RequestMethod.POST)
    public ModelAndView bookCar(HttpServletRequest request) throws EmailException
    {
        request.getSession().getAttribute("pickUpDate");
        request.getSession().getAttribute("dropOffdate");
        String station = (String)request.getSession().getAttribute("station");
        Car selectedCar = (Car)request.getSession().getAttribute("carSelectedDetails");
        String startdateTime = (String)request.getSession().getAttribute("startDateTime");
        String enddateTime = (String)request.getSession().getAttribute("endDateTime");
        Station stationBean = (Station)request.getSession().getAttribute("stationBean");
        //ArrayList<Station> stationList = (ArrayList<Station>)request.getSession().getAttribute("stationList");

        RentalTransaction rentalTransaction = new RentalTransaction();
        setRentalTransactionDetails(stationBean, startdateTime, enddateTime, selectedCar,
        (User)request.getSession().getAttribute("userSession"));
        RentalTransaction savedTransaction = new RentalTransaction();
        rentDao.saveTransaction(rentalTransaction);
        boolean emailed = emailService.sendEmail(savedTransaction,
        (User)request.getSession().getAttribute("userSession"));
        return new ModelAndView("successful-booking", "bookinReferenceNumber",
        rentalTransaction.getBookingReferenceNumber());
    }

    private RentalTransaction setRentalTransactionDetails(Station stationBean, String
    startdateTime, String enddateTime,
        Car selectedCar, User userSession)
    {
        RentalTransaction rentalTransaction = new RentalTransaction();
        rentalTransaction.setAvailability("true");
        rentalTransaction.setCar(selectedCar);
    }

```

```
        rentalTransaction.setCreatedOn(new Timestamp(System.currentTimeMillis()));
        rentalTransaction.setStartDateTime(Timestamp.valueOf(startdateTime) );
        rentalTransaction.setEndDateTime(Timestamp.valueOf(enddateTime));
        rentalTransaction.setDeleteFlag("F");
        rentalTransaction.setModifiedOn(new Timestamp(System.currentTimeMillis()));
        rentalTransaction.setPerHourRate(selectedCar.getPerHourrate());
        rentalTransaction.setStation(stationBean);
        rentalTransaction.setUser(userSession);

        rentalTransaction.setBookingReferenceNumber(userSession.getFirstName().substring(0,
2).concat("_").concat(userSession.getLastName().substring(0,
2)).concat(System.currentTimeMillis()+""));
        return rentalTransaction;
    }
}
```