Ryerson University

Department of Electrical & Computer Engineering
Faculty of Engineering & Architectural Science

Course Title:	Software Design Architecture
Course Number:	COE692
Semester/Year:	W2021

Instructor:	Dr. Faezeh Ensan

Assignment/Lab Number:	Lab 5
Assignment/Lab Title:	Lab 5 Part 1

Submission Date:	April 15th, 2021
Due Date:	April 15th, 2021

Last Name:	First Name:	Student ID:	Section:	*Signature:
Samaroo	Anand Alexander	xxxxxx021	03	AAS

*By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: http://www.ryerson.ca/senate/current/pol60.pdf

Dependent Microservices Description:

In the Car Rental Service project, there are 2 microservices HoldCar, and RentCar that need to communicate asynchronously. RentCar allows users to rent cars given that it is not on hold for another user. With that said RentCar needs to know which cars are on hold, and which users are holding said cars. Whenever the HoldCar microservice creates a new hold there must be a new message published to all users that are interested. The RentCar microservice is subscribed to the HoldCar Channel, and is able to receive messages from the HoldCar service as well as act accordingly.

Database Description:

The HoldCar microservice has the hold_CARS db which contains the Car_Hold table. This table includes a unique ID for each hold (id) along with the ID of the car (carid), the username of the holder (username), and the start date for each hold (startdate). When a user wants to hold a car there will be a new row created in this table. When the new row is created a message is published in the channel named “hold_car_channel”.

The RentCar microservice has the rent_CARS db which contains the Car_Hold, and the User_Car_Rent tables. Car_Hold retains all of the information from the Car_Hold table associated with the hold_CARS db obviously received from the HoldCar microservice. The User_Car_Rent table is related to the cars that users actually rent. It includes the ID of the rented car (carid), the username of the user renting the car (username), and the date at which the car was rented (rentDate).

Screenshots:

```java
package ryerson.ca.holdcar.business;


import io.kubemq.sdk.basic.ServerAddressNotSuppliedException;
import java.io.IOException;
import java.sql.SQLException;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import ryerson.ca.holdcar.helper.*;
import ryerson.ca.holdcar.persistence.*;
/**
 *
 * @author student
 */
public class HoldBusiness {

    public HoldBusiness(){

    }

    public CarHold getCar(String carID){

        CarHold cs = Car_Hold_CRUD.getHoldCar(carID);
        return cs;
    }

    public boolean hold(String carID, String username) throws ClassNotFoundException, SQLException, ServerAddressNotSuppliedException, IOException, InterruptedException{
        boolean success = false;

        success = Car_Hold_CRUD.addHold(carID, username);
        if(success){
            DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
            LocalDate date = LocalDate.now();

            LocalDate exDate = date.plusDays(3);

            Messaging.sendmessage("HOLD: "+carID+":"+username+":"+exDate.format(formatter));
        }
        return success;
    }
}

```

```java

package ryerson.ca.holdcar.business;

import io.kubemq.sdk.basic.ServerAddressNotSuppliedException;
import io.kubemq.sdk.event.Event;
import io.kubemq.sdk.tools.Converter;
import javax.net.ssl.SSLException;
import java.io.IOException;


public class Messaging {



    public static void sendmessage(String message) throws IOException{
    String channelName = "hold_car_channel",
            clientID = "hold-car-subscriber";
              String kubeMQAddress = System.getenv("kubeMQAddress");

  io.kubemq.sdk.event.Channel channel = new io.kubemq.sdk.event.Channel(channelName, clientID, false,
        kubeMQAddress);

    channel.setStore(true);
     Event event = new Event();
     event.setBody(Converter.ToByteArray(message));
     event.setEventId("event-Store-" );
     try {
         channel.SendEvent(event);
     } catch (SSLException e) {
         System.out.printf("SSLException: %s", e.getMessage());
         e.printStackTrace();
     } catch (ServerAddressNotSuppliedException e) {
         System.out.printf("ServerAddressNotSuppliedException: %s", e.getMessage());
         e.printStackTrace();
     }

    }

}

```

The first set of screenshots are from the HoldCar microservice. It has a Messaging class which creates the “hold_car_channel” to send messages to the RentCar microservice. In the HoldBusiness class whenever a car is put on hold, a message is created and sent over to any service that is subscribed to the hold_car_channel.

The next set of screenshots are from the RentCar microservice. It also has a Messaging class that receives the messages sent from the HoldCar microservice via the hold_car_channel. The messages are interpreted with the MyServletContextListener class. The MyServletContextListener gets notified about ServletContext lifecycle change.

```java
package ryerson.ca.rentcar.business;

import io.grpc.stub.StreamObserver;
import io.kubemq.sdk.basic.ServerAddressNotSuppliedException;
import io.kubemq.sdk.event.EventReceive;
import io.kubemq.sdk.event.Subscriber;
import io.kubemq.sdk.subscription.EventsStoreType;
import io.kubemq.sdk.subscription.SubscribeRequest;
import io.kubemq.sdk.subscription.SubscribeType;
import io.kubemq.sdk.tools.Converter;
import java.io.IOException;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.net.ssl.SSLException;
import ryerson.ca.rentcar.endpoint.MyAppServletContextListener;
import ryerson.ca.rentcar.persistence.*;

public class Messaging {
    public static void Receiving_Events_Store(String cname) throws SSLException, ServerAddressNotSuppliedException {
        String ChannelName = cname, ClientID = "hello-world-subscribe1r";
                String kubeMQAddress = System.getenv("kubeMQAddress");
        Subscriber subscriber = new Subscriber(kubeMQAddress);
        SubscribeRequest subscribeRequest = new SubscribeRequest();
        subscribeRequest.setChannel(ChannelName);
        subscribeRequest.setClientID(ClientID);
        subscribeRequest.setSubscribeType(SubscribeType.EventsStore);
        subscribeRequest.setEventsStoreType(EventsStoreType.StartAtSequence);
        subscribeRequest.setEventsStoreTypeValue(1);

        StreamObserver<EventReceive> streamObserver = new StreamObserver<EventReceive>() {

            @Override
            public void onNext(EventReceive value) {
                try {
                    String val=(String) Converter.FromByteArray(value.getBody());
                    System.out.printf("Event Received: EventID: %s, Channel: %s, Metadata: %s, Body: %s",
                            value.getEventId(), value.getChannel(), value.getMetadata(),
                            Converter.FromByteArray(value.getBody()));
                    String[] msgParts = val.split(":");
                    if(msgParts.length==4){
                        if(msgParts[0].equals("HOLD")){

                            String carid=msgParts[1];
                            String username=msgParts[2];
                            String date=msgParts[3];
                              Car_Hold_CRUD.addHold(carid, username, date);
                          }
                    }
                } catch (ClassNotFoundException e) {
                    System.out.printf("ClassNotFoundException: %s", e.getMessage());
                    e.printStackTrace();
                } catch (IOException e) {
                    System.out.printf("IOException: %s", e.getMessage());
                    e.printStackTrace();
                } catch (SQLException ex) {
                    Logger.getLogger(MyAppServletContextListener.class.getName()).log(Level.SEVERE, null, ex);
                }
            }

            @Override
            public void onError(Throwable t) {
                System.out.printf("onError:  %s", t.getMessage());
            }

            @Override
            public void onCompleted() {

            }

        };
        subscriber.SubscribeToEvents(subscribeRequest, streamObserver);

    }

}
```

```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ryerson.ca.rentcar.endpoint;

import io.kubemq.sdk.basic.ServerAddressNotSuppliedException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.net.ssl.SSLException;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

import ryerson.ca.rentcar.business.Messaging;


public class MyAppServletContextListener
                implements ServletContextListener{

    @Override
    public void contextDestroyed(ServletContextEvent arg0) {
        System.out.println("ServletContextListener destroyed");
    }

    @Override
    public void contextInitialized(ServletContextEvent arg0) {
     Runnable r = new Runnable() {
         public void run() {

             try {
                 Messaging.Receiving_Events_Store("hold_car_channel");
             } catch (SSLException ex) {
                 Logger.getLogger(MyAppServletContextListener.class.getName()).log(Level.SEVERE, null, ex);
             } catch (ServerAddressNotSuppliedException ex) {
                 Logger.getLogger(MyAppServletContextListener.class.getName()).log(Level.SEVERE, null, ex);
             }
         }
     };

     new Thread(r).start();
    }
}

```

Here are screenshots of the Databases: