```
create database mphdb1;
use mphdb1;
CREATE TABLE tbl_admin (
  admin_id int(11) NOT NULL,
  admin_name varchar(255) NOT NULL,
  admin_email varchar(255) NOT NULL,
  admin_password varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

INSERT INTO tbl_admin (`admin_id`, `admin_name`, `admin_email`,
`admin_password`) VALUES
(1, 'Admin', ganesh@gmail.com', ‘ganesh123’);

select * from tbl_admin;
```

```
package com.project.fitnessclubautomation.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

import com.project.fitnessclubautomation.model.Admin;
import com.project.fitnessclubautomation.service.AdminService;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

@Controller
public class AdminController {
  @Autowired
  AdminService adminService;

  @RequestMapping("/")
  private String home(ModelMap modelMap) {
    modelMap.addAttribute("pagetitle", "Login");
    return "login";
  }

  @RequestMapping("/login")
  private String login(ModelMap modelMap,
             HttpServletRequest request,
             @RequestParam(value = "user_email", required = true)
String user_email,
             @RequestParam(value = "user_password", required =
true) String user_password) {
    modelMap.addAttribute("pagetitle", "Login");
    Admin admin = adminService.login(user_email, user_password);
    if (admin == null) {
      modelMap.addAttribute("error", true);
      modelMap.addAttribute("message", "Invalid Credentials! Please
try again.");
```

```
            return "login";
        }
        HttpSession session = request.getSession();
        session.setAttribute("adminId", admin.getAdminId());
        session.setAttribute("adminName", admin.getAdminName());
        return "redirect:/subscriber";
    }

    @RequestMapping(value = "/logout", method = RequestMethod.GET)
    public String logout(ModelMap modelMap, HttpServletRequest
request) {
        HttpSession session = request.getSession();
        session.invalidate();
        return "redirect:/";
    }
}



package com.project.fitnessclubautomation.controller;


import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import com.project.fitnessclubautomation.model.Payment;
import com.project.fitnessclubautomation.model.Subscriber;
import com.project.fitnessclubautomation.service.PaymentService;
import com.project.fitnessclubautomation.service.SubscriberService;
import
com.project.fitnessclubautomation.service.SubscriptionPlanService;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

@Controller
public class PaymentController {
   @Autowired
   PaymentService paymentService;
   @Autowired
   SubscriberService subscriberService;
   @Autowired
   SubscriptionPlanService subscriptionPlanService;

   @RequestMapping(value = "/payment/add-new")
   private String addPayment(ModelMap modelMap,
HttpServletRequest request) {
      HttpSession session = request.getSession();
      if (session.getAttribute("adminId") == null) {
          return "redirect:/";
      }
      String[] payment_mode = {"Cash", "Credit/Debit Card", "UPI"};
      List<Subscriber> subscribers =
subscriberService.getAllSubscribers();
      if (subscribers.size() > 0) {
          modelMap.addAttribute("subscribers", subscribers);
          modelMap.addAttribute("payment_mode", payment_mode);
      } else {
          modelMap.addAttribute("error", true);
          modelMap.addAttribute("message", "No subscriber found for
initiating payment.<br>");
      }
      return "add-payment";
   }

   @RequestMapping(value = "/payment/add", method =
RequestMethod.POST)
   private String addPayment(ModelMap modelMap,
HttpServletRequest request, @RequestParam(value =
"payment_amount", required = true) int payment_amount,
@RequestParam(value = "payment_mode", required = true) String
payment_mode, @RequestParam(value = "payment_subscriber_id",
required = true) Long payment_subscriber_id) {
      HttpSession session = request.getSession();
      if (session.getAttribute("adminId") == null) {
          return "redirect:/";
      }
      Subscriber s =
subscriberService.getSubscriber(payment_subscriber_id);
      Long planId = s.getSubscriptionPlan().getPlanId();
      int planFees =
subscriptionPlanService.getSubscriptionPlan(planId).getPlanFees();
      int paid = s.getSubscriberFeesPaid();
      int total = paid + payment_amount;
      if (total > planFees) {
          modelMap.addAttribute("error", true);
          modelMap.addAttribute("message", "Payment amount
exceeding the actual fees limit.<br>");
          return "add-payment";
      }
      SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
      Date date = new Date();
      String payment_date = sdf.format(date);
      Payment payment = new Payment(payment_amount,
payment_date, payment_mode, s);
      paymentService.addPayment(payment);
      s.setSubscriberFeesPaid(total);
      subscriberService.addSubscriber(s);
      return "redirect:/payment";
   }

   @RequestMapping(value = "/payment/edit-payment/{payment_id}")
   private String editPayment(ModelMap modelMap,
HttpServletRequest request, @PathVariable Long payment_id) {
      HttpSession session = request.getSession();
      if (session.getAttribute("adminId") == null) {
          return "redirect:/";
      }
      Payment payment = new Payment();
      try {
          payment = paymentService.getPayment(payment_id);
          modelMap.addAttribute("payment", payment);
          String[] payment_mode = {"Cash", "Credit/Debit Card", "UPI"};
          modelMap.addAttribute("payment_mode", payment_mode);
          List<Subscriber> subscribers =
subscriberService.getAllSubscribers();
          modelMap.addAttribute("subscribers", subscribers);
          return "edit-payment";
      } catch (Exception ex) {
          modelMap.addAttribute("error", true);
          modelMap.addAttribute("message", ex.getMessage());
          return "redirect:/payment-list";
      }
   }

   @RequestMapping(value = "/payment/edit", method =
RequestMethod.POST)
   private String editPayment(ModelMap modelMap,
HttpServletRequest request, @RequestParam(value = "payment_id",
required = true) Long payment_id, @RequestParam(value =
"payment_amount", required = true) int payment_amount,
@RequestParam(value = "old_payment_amount", required = true) int
old_payment_amount, @RequestParam(value = "payment_date",
required = true) String payment_date, @RequestParam(value =
"payment_mode", required = true) String payment_mode,
@RequestParam(value = "payment_subscriber_id", required = true)
Long payment_subscriber_id) {
      HttpSession session = request.getSession();
      if (session.getAttribute("adminId") == null) {
          return "redirect:/";
      }
      Subscriber subscriber =
subscriberService.getSubscriber(payment_subscriber_id);
      int old_total = subscriber.getSubscriberFeesPaid() -
old_payment_amount;
      int new_total = old_total + payment_amount;
      int planFees = subscriber.getSubscriptionPlan().getPlanFees();



      /*System.out.println("FEES PAID - " +
subscriber.getSubscriberFeesPaid());
      System.out.println("PAYMENT AMOUNT - " + payment_amount);
      System.out.println("PLAN FEES - " + planFees);
      System.out.println("OLD TOTAL - " + old_total);
      System.out.println("NEW TOTAL - " + new_total);
*/
      if (new_total > planFees) {
          modelMap.addAttribute("error", true);
          modelMap.addAttribute("message", "Payment amount
exceeding the actual fees limit.<br>");
          return "edit-payment";
      }

      Payment payment = new Payment(payment_id, payment_amount,
payment_date, payment_mode, subscriber);
      paymentService.addPayment(payment);
      subscriber.setSubscriberFeesPaid(new_total);
      subscriberService.addSubscriber(subscriber);
      return "redirect:/payment";
  }

  @RequestMapping(value = "/payment")
  private String getAllPayments(ModelMap modelMap,
HttpServletRequest request) {
      HttpSession session = request.getSession();
      if (session.getAttribute("adminId") == null) {
          return "redirect:/";
      }
      List<Payment> payments = new ArrayList<>();
      try {
          payments = paymentService.getAllPayments();
          modelMap.addAttribute("payments", payments);
          modelMap.addAttribute("message", "Total <b>" +
payments.size() + "</b> payment records found.");
      } catch (Exception ex) {
          modelMap.addAttribute("error", true);
          modelMap.addAttribute("message", ex.getMessage() + "<br>");
      }
      return "payment-list";
  }

  @RequestMapping(value = "/payment/{payment_id}")
  private String getPayment(ModelMap modelMap, HttpServletRequest
request, @PathVariable Long payment_id) {
      HttpSession session = request.getSession();
      if (session.getAttribute("adminId") == null) {
          return "redirect:/";
      }
      Payment payment = new Payment();
      try {
          payment = paymentService.getPayment(payment_id);
          modelMap.addAttribute("payment", payment);
          return "payment-single";
      } catch (Exception ex) {
          modelMap.addAttribute("error", true);
          modelMap.addAttribute("message", ex.getMessage() + "<br>");
          return "redirect:/payment-list";
      }
  }
}


package com.project.fitnessclubautomation.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

import com.project.fitnessclubautomation.model.Subscriber;
import com.project.fitnessclubautomation.model.SubscriptionPlan;
import com.project.fitnessclubautomation.model.Trainer;
import com.project.fitnessclubautomation.service.SubscriberService;
import
com.project.fitnessclubautomation.service.SubscriptionPlanService;
import com.project.fitnessclubautomation.service.TrainerService;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import java.util.ArrayList;
import java.util.List;

@Controller
public class SubscriberController {
   @Autowired
   SubscriberService subscriberService;
   @Autowired
   private TrainerService trainerService;
   @Autowired
   private SubscriptionPlanService subscriptionPlanService;

   @RequestMapping(value = "/subscriber/add-new")
```

```
    private String addSubscriber(ModelMap modelMap,
HttpServletRequest request) {
        HttpSession session = request.getSession();
        if (session.getAttribute("adminId") == null) {
            return "redirect:/";
        }
        List<Trainer> trainers = trainerService.getAllTrainers();
        modelMap.addAttribute("trainers", trainers);

        List<SubscriptionPlan> subscriptionPlans =
subscriptionPlanService.getAllSubscriptionPlans();
        modelMap.addAttribute("subscriptionPlans", subscriptionPlans);
        return "add-subscriber";
    }

    @RequestMapping(value = "/subscriber/add", method =
RequestMethod.POST)
    private String addSubscriber(HttpServletRequest request,
                                 @RequestParam(value = "subscriber_name",
required = true) String subscriber_name,
                                 @RequestParam(value = "subscriber_age", required =
true) int subscriber_age,
                                 @RequestParam(value = "subscriber_gender",
required = true) String subscriber_gender,
                                 @RequestParam(value = "subscriber_address",
required = true) String subscriber_address,
                                 @RequestParam(value = "subscriber_trainer_id",
required = true) Long subscriber_trainer_id,
                         @RequestParam(value = "subscription_plan_id",
required = true) Long subscription_plan_id) {
        HttpSession session = request.getSession();
        if (session.getAttribute("adminId") == null) {
            return "redirect:/";
        }
        Trainer trainer = trainerService.getTrainer(subscriber_trainer_id);
        SubscriptionPlan subscriptionPlan =
subscriptionPlanService.getSubscriptionPlan(subscription_plan_id);
        Subscriber subscriber = new Subscriber(subscriber_name,
subscriber_age, subscriber_gender, subscriber_address, true, 0, trainer,
subscriptionPlan);
        if (subscriberService.addSubscriber(subscriber) != null) {
            return "redirect:/subscriber";
        } else {
            return "redirect:/subscriber";
        }
    }

    @RequestMapping(value = "/subscriber/edit-
subscriber/{subscriber_id}")
    private String editSubscriber(ModelMap modelMap,
HttpServletRequest request,
                         @PathVariable Long subscriber_id) {
        HttpSession session = request.getSession();
        if (session.getAttribute("adminId") == null) {
            return "redirect:/";
        }
        Subscriber subscriber =
subscriberService.getSubscriber(subscriber_id);
        modelMap.addAttribute("subscriber", subscriber);

        List<Trainer> trainers = trainerService.getAllTrainers();
        modelMap.addAttribute("trainers", trainers);

        List<SubscriptionPlan> subscriptionPlans =
subscriptionPlanService.getAllSubscriptionPlans();
        modelMap.addAttribute("subscriptionPlans", subscriptionPlans);
        return "edit-subscriber";
    }

    @RequestMapping(value = "/subscriber/edit", method =
RequestMethod.POST)
    private String editSubscriber(ModelMap modelMap,
HttpServletRequest request, @RequestParam(value = "subscriber_id",
required = true) Long subscriber_id,
                                  @RequestParam(value = "subscriber_name",
required = true) String subscriber_name,
                                  @RequestParam(value = "subscriber_age", required
= true) int subscriber_age,
                                  @RequestParam(value = "subscriber_gender",
required = true) String subscriber_gender,
                                  @RequestParam(value = "subscriber_status",
required = true) boolean subscriber_status,
                                  @RequestParam(value = "subscriber_address",
required = true) String subscriber_address,
                                @RequestParam(value = "subscriber_trainer_id",
required = true) Long subscriber_trainer_id,
                                @RequestParam(value = "subscription_plan_id",
required = true) Long subscription_plan_id) {
        HttpSession session = request.getSession();
        if (session.getAttribute("adminId") == null) {
            return "redirect:/";
        }
        if (subscriber_trainer_id == 0) {
            modelMap.addAttribute("error", true);
            modelMap.addAttribute("message", "Please select a valid
trainer");
            return "edit-subscriber";
        }
        if (subscription_plan_id == 0) {
            modelMap.addAttribute("error", true);
            modelMap.addAttribute("message", "Please select a valid
subscription plan");
            return "edit-subscriber";
        }
        Trainer trainer = trainerService.getTrainer(subscriber_trainer_id);
        SubscriptionPlan subscriptionPlan =
subscriptionPlanService.getSubscriptionPlan(subscription_plan_id);
        int paid_fees =
subscriberService.getSubscriber(subscriber_id).getSubscriberFeesPaid();
        Subscriber subscriber = new Subscriber(subscriber_id,
subscriber_name, subscriber_age, subscriber_gender,
subscriber_address, subscriber_status, paid_fees, trainer,
subscriptionPlan);
        subscriber = subscriberService.addSubscriber(subscriber);
        return "redirect:/subscriber";
    }

    @RequestMapping(value = "/subscriber/delete/{subscriber_id}")
    private String deleteSubscriber(@PathVariable Long subscriber_id,
HttpServletRequest request) {
        HttpSession session = request.getSession();
        if (session.getAttribute("adminId") == null) {
            return "redirect:/";
        }
        try {
            subscriberService.deleteSubscriber(subscriber_id);
            return "redirect:/subscriber";
        } catch (Exception ex) {
            return "redirect:/subscriber";
        }
    }

    @RequestMapping(value = "/subscriber")
    private String getAllSubscribers(ModelMap modelMap,
HttpServletRequest request) {
        HttpSession session = request.getSession();
        if (session.getAttribute("adminId") == null) {
            return "redirect:/";
        }
        List<Subscriber> subscribers = new ArrayList<>();
        try {
            subscribers = subscriberService.getAllSubscribers();
            modelMap.addAttribute("subscriber_list", subscribers);
            modelMap.addAttribute("message", "Total <b>" +
subscribers.size() + "</b> subscribers found.");
            return "subscriber-list";
        } catch (Exception ex) {
            modelMap.addAttribute("error", true);
            modelMap.addAttribute("message", ex.getMessage() + "");
            return "subscriber-list";
        }
    }

    @RequestMapping(value = "/subscriber/{subscriber_id}")
    private String getSubscriber(ModelMap modelMap,
HttpServletRequest request, @PathVariable Long subscriber_id) {
        Subscriber subscriber = new Subscriber();
        HttpSession session = request.getSession();
        if (session.getAttribute("adminId") == null) {
            return "redirect:/";
        }
        try {
            subscriber = subscriberService.getSubscriber(subscriber_id);
            modelMap.addAttribute("subscriber", subscriber);
            return "subscriber-single";
        } catch (Exception ex) {
            modelMap.addAttribute("error", true);
            modelMap.addAttribute("message", ex.getMessage());
            return "redirect:/subscriber/{subscriber_id}";
        }
    }
}

package com.project.fitnessclubautomation.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.*;

import com.project.fitnessclubautomation.model.SubscriptionPlan;
import
com.project.fitnessclubautomation.service.SubscriptionPlanService;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import java.util.ArrayList;
import java.util.List;

@Controller
public class SubscriptionController {
    @Autowired
    SubscriptionPlanService subscriptionPlanService;

    @RequestMapping(value = "/subscriptionplan/add-new")
    private String addSubscriptionPlan(ModelMap modelMap,
HttpServletRequest request) {
        modelMap.addAttribute("pageTitle", "Add new Plan");
        HttpSession session = request.getSession();
        if (session.getAttribute("adminId") == null) {
            return "redirect:/";
        }
        return "add-plan";
    }

    @RequestMapping(value = "/subscriptionplan/add", method =
RequestMethod.POST)
    private String addSubscriptionPlan(ModelMap modelMap,
HttpServletRequest request,
                                       @RequestParam(value = "plan_title", required =
true) String plan_title,
                                       @RequestParam(value = "plan_duration",
required = true) int plan_duration,
                                       @RequestParam(value = "plan_fees", required =
true) int plan_fees) {
        HttpSession session = request.getSession();
        if (session.getAttribute("adminId") == null) {
            return "redirect:/";
        }
        if (plan_duration <= 0) {
            modelMap.addAttribute("error", true);
            modelMap.addAttribute("message", "Invalid duration");
            return "add-plan";
        }
        if (plan_fees <= 0) {
            modelMap.addAttribute("error", true);
            modelMap.addAttribute("message", "Invalid fees");
            return "add-plan";
        }
        SubscriptionPlan subscriptionPlan = new
SubscriptionPlan(plan_title, plan_duration, plan_fees);
        subscriptionPlanService.addSubscriptionPlan(subscriptionPlan);
        modelMap.addAttribute("success", true);
        modelMap.addAttribute("message", "Plan added successfully.");
        return "redirect:/subscriptionplan";
    }

    @RequestMapping(value = "/subscriptionplan/edit-plan/{plan_id}")
    private String editSubscriptionPlan(ModelMap modelMap,
HttpServletRequest request, @PathVariable Long plan_id) {
        HttpSession session = request.getSession();
        if (session.getAttribute("adminId") == null) {
            return "redirect:/";
        }
        SubscriptionPlan plan = null;
        try {
            plan = subscriptionPlanService.getSubscriptionPlan(plan_id);
            if (plan != null) {
                modelMap.addAttribute("plan", plan);
                return "edit-plan";
            } else {
```

```
            return "redirect:/subscriptionplan";
          }
        } catch (Exception ex) {
          modelMap.addAttribute("error", true);
          modelMap.addAttribute("message", ex.getMessage());
          return "redirect:/subscriptionplan";
        }
    }

    @RequestMapping(value = "/subscriptionplan/edit", method =
RequestMethod.POST)
    private String editSubscriptionPlan(ModelMap modelMap,
HttpServletRequest request,
                                    @RequestParam(value = "plan_id", required =
true) Long plan_id,
                                    @RequestParam(value = "plan_title", required =
true) String plan_title,
                                    @RequestParam(value = "plan_duration",
required = true) int plan_duration,
                                    @RequestParam(value = "plan_fees", required =
true) int plan_fees) {
        HttpSession session = request.getSession();
        if (session.getAttribute("adminId") == null) {
          return "redirect:/";
        }
        if (plan_duration <= 0) {
          modelMap.addAttribute("error", true);
          modelMap.addAttribute("message", "Invalid duration");
```

```
            return "redirect:/subscriptionplan/edit-plan/" + plan_id;
        }
        if (plan_fees <= 0) {
            modelMap.addAttribute("error", true);
            modelMap.addAttribute("message", "Invalid fees");
            return "redirect:/subscriptionplan/edit-plan/" + plan_id;
        }
        SubscriptionPlan subscriptionPlan = new SubscriptionPlan(plan_id,
plan_title, plan_duration, plan_fees);
        subscriptionPlanService.addSubscriptionPlan(subscriptionPlan);
        modelMap.addAttribute("success", true);
        modelMap.addAttribute("message", "Plan data saved
successfully.");
        return "redirect:/subscriptionplan";
    }

    @RequestMapping(value = "/subscriptionplan/delete/{plan_id}")
    private String deleteSubscriptionPlan(ModelMap modelMap,
HttpServletRequest request, @PathVariable Long plan_id) {
        HttpSession session = request.getSession();
        if (session.getAttribute("adminId") == null) {
            return "redirect:/";
        }
        try {
            subscriptionPlanService.deleteSubscriptionPlan(plan_id);
            return "redirect:/subscriptionplan";
        } catch (Exception ex) {
            modelMap.addAttribute("error", true);
            modelMap.addAttribute("message", ex.getMessage());
            return "redirect:/subscriptionplan";
        }
    }

    @RequestMapping(value = "/subscriptionplan")
    private String getAllSubscriptionPlans(ModelMap modelMap,
HttpServletRequest request) {
        HttpSession session = request.getSession();
        if (session.getAttribute("adminId") == null) {
            return "redirect:/";
        }
        List<SubscriptionPlan> subscriptionPlans = new ArrayList<>();
        try {
            subscriptionPlans =
subscriptionPlanService.getAllSubscriptionPlans();
            modelMap.addAttribute("subscriptionPlans", subscriptionPlans);
            modelMap.addAttribute("success", true);
            modelMap.addAttribute("message", "Total <b>" +
subscriptionPlans.size() + "</b> subscription plans found.");
            return "plan-list";
        } catch (Exception ex) {
            modelMap.addAttribute("error", true);
            modelMap.addAttribute("message", ex.getMessage());
            return "plan-list";
        }
    }
    @RequestMapping(value = "/subscriptionplan/{plan_id}")
    private String getSubscriptionPlan(ModelMap modelMap,
HttpServletRequest request, @PathVariable Long plan_id) {
        HttpSession session = request.getSession();
        if (session.getAttribute("adminId") == null) {
            return "redirect:/";
        }
        SubscriptionPlan subscriptionPlan = new SubscriptionPlan();
        try {
            subscriptionPlan =
subscriptionPlanService.getSubscriptionPlan(plan_id);
            modelMap.addAttribute("plan", subscriptionPlan);
            return "plan-single";
        } catch (Exception ex) {
            modelMap.addAttribute("error", true);
            modelMap.addAttribute("message", ex.getMessage());
            return "redirect:/subscriptionplan";
        }
    }
}


package com.project.fitnessclubautomation.controller;


import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpRequest;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.*;

import com.project.fitnessclubautomation.model.Trainer;
import com.project.fitnessclubautomation.service.TrainerService;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import java.util.ArrayList;
import java.util.List;

@Controller
public class TrainerController {
   @Autowired
   TrainerService trainerService;

   @RequestMapping(value = "/trainer/add-new")
   private String addTrainer() {
      return "add-trainer";
   }

   @RequestMapping(value = "/trainer/add", method =
RequestMethod.POST)
   private String addTrainer(ModelMap modelMap, HttpServletRequest
request,
                  @RequestParam(value = "trainer_name", required =
true) String trainer_name,
                  @RequestParam(value = "trainer_age", required =
true) int trainer_age,
                    @RequestParam(value = "trainer_gender", required =
true) String trainer_gender,
                    @RequestParam(value = "trainer_experience",
required = true) int trainer_experience,
                    @RequestParam(value = "trainer_address", required =
true) String trainer_address) {
    HttpSession session = request.getSession();
    if (session.getAttribute("adminId") == null) {
      return "redirect:/";
    }
    Trainer tr = new Trainer(trainer_name, trainer_age, trainer_gender,
trainer_experience, trainer_address);
    tr = trainerService.addTrainer(tr);
    return "redirect:/trainer";
  }

  @RequestMapping(value = "/trainer/edit-trainer/{trainer_id}")
  private String editTrainer(ModelMap modelMap, HttpServletRequest
request,
                    @PathVariable Long trainer_id) {
    HttpSession session = request.getSession();
    if (session.getAttribute("adminId") == null) {
      return "redirect:/";
    }
    int[] experience = {1, 2, 3, 4, 5, 6, 7, 8, 9};
    modelMap.addAttribute("experience", experience);
    Trainer trainer = trainerService.getTrainer(trainer_id);
    modelMap.addAttribute("trainer", trainer);
```

```
        return "edit-trainer";
    }


    @RequestMapping(value = "/trainer/edit", method =
RequestMethod.POST)
    private String editTrainer(ModelMap modelMap, HttpServletRequest
request,
                            @RequestParam(value = "trainer_id", required = true)
Long trainer_id,
                            @RequestParam(value = "trainer_name", required =
true) String trainer_name,
                            @RequestParam(value = "trainer_age", required =
true) int trainer_age,
                            @RequestParam(value = "trainer_gender", required =
true) String trainer_gender,
                            @RequestParam(value = "trainer_experience",
required = true) int trainer_experience,
                            @RequestParam(value = "trainer_address", required =
true) String trainer_address) {
        HttpSession session = request.getSession();
        if (session.getAttribute("adminId") == null) {
            return "redirect:/";
        }
        Trainer tr = new Trainer(trainer_id, trainer_name, trainer_age,
trainer_gender, trainer_experience, trainer_address);
        trainerService.addTrainer(tr);
        return "redirect:/trainer";
    }
    @RequestMapping(value = "/trainer/delete/{trainer_id}", method =
RequestMethod.GET)
    private String deleteTrainer(ModelMap modelMap,
HttpServletRequest request,
                                 @PathVariable Long trainer_id) {
        HttpSession session = request.getSession();
        if (session.getAttribute("adminId") == null) {
            return "redirect:/";
        }
        try {
            trainerService.deleteTrainer(trainer_id);
            modelMap.addAttribute("success", true);
            modelMap.addAttribute("message", "Data deleted
successfully.");
            return "redirect:/trainer";
        } catch (Exception ex) {
            modelMap.addAttribute("error", true);
            modelMap.addAttribute("message", ex.getMessage());
            return "redirect:/trainer";
        }
    }

    @RequestMapping(value = "/trainer")
    private String getAllTrainers(ModelMap modelMap,
HttpServletRequest request) {
        HttpSession session = request.getSession();
        if (session.getAttribute("adminId") == null) {
            return "redirect:/";
        }
        List<Trainer> trainers = new ArrayList<>();
        try {
            trainers = trainerService.getAllTrainers();
            if (trainers.size() > 0) {
                modelMap.addAttribute("trainers", trainers);
                modelMap.addAttribute("message", "Total <b>" +
trainers.size() + "</b> trainers found.");
            } else {
                modelMap.addAttribute("message", "Total <b>" +
trainers.size() + "</b> trainers found.");
            }
            return "trainer-list";
        } catch (Exception ex) {
            modelMap.addAttribute("error", true);
            modelMap.addAttribute("message", ex.getMessage());
            return "trainer-list";
        }
    }

    @RequestMapping(value = "/trainer/{trainer_id}")
    private String getTrainer(ModelMap modelMap, HttpServletRequest
request, @PathVariable Long trainer_id) {
        HttpSession session = request.getSession();
        if (session.getAttribute("adminId") == null) {
            return "redirect:/";
        }
```

```
                Trainer trainer = new Trainer();

                try {

                    trainer = trainerService.getTrainer(trainer_id);

                    modelMap.addAttribute("trainer", trainer);

                    return "trainer-single";

                } catch (Exception ex) {

                    modelMap.addAttribute("error", true);

                    modelMap.addAttribute("message", ex.getMessage());

                    return "redirect:/trainer";

                }

            }

        }
```

```
package com.project.fitnessclubautomation.model;

import javax.persistence.*;

@Entity
@Table(name = "tbl_admin")
public class Admin {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "admin_id", nullable = false)
    private Long adminId;
    @Column(name = "admin_name", nullable = false)
    private String adminName;
    @Column(name = "admin_email", nullable = false)
    private String adminEmail;
    @Column(name = "admin_password", nullable = false)
    private String adminPassword;

    public Admin() {
    }

    public Admin(Long adminId, String adminName, String adminEmail, String
adminPassword) {
        this.adminId = adminId;
        this.adminName = adminName;
        this.adminEmail = adminEmail;
        this.adminPassword = adminPassword;
    }
    public Admin(String adminName, String adminEmail, String adminPassword) {
        this.adminName = adminName;
        this.adminEmail = adminEmail;
        this.adminPassword = adminPassword;
    }

    public Long getAdminId() {
        return adminId;
    }

    public void setAdminId(Long adminId) {
        this.adminId = adminId;
    }

    public String getAdminName() {
        return adminName;
    }

    public void setAdminName(String adminName) {
        this.adminName = adminName;
    }

    public String getAdminEmail() {
        return adminEmail;
    }

    public void setAdminEmail(String adminEmail) {
        this.adminEmail = adminEmail;
    }

    public String getAdminPassword() {
        return adminPassword;
    }

    public void setAdminPassword(String adminPassword) {
        this.adminPassword = adminPassword;
    }
}
```

```
package com.project.fitnessclubautomation.model;

import javax.persistence.*;

@Entity
@Table(name = "tbl_subscriberpayments")
public class Payment {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "payment_id", nullable = false)
    private Long paymentId;
    @Column(name = "payment_amount", nullable = false)
    private int paidAmount;
    @Column(name = "payment_date", nullable = false)
    private String paymentDate;
    @Column(name = "payment_mode", nullable = false)
    private String paymentMode;

    @ManyToOne
    @JoinColumn(name = "payment_subscriber_id")
    private Subscriber subscriber;

    public Subscriber getSubscriber() {
        return subscriber;
    }

    public void setSubscriber(Subscriber subscriber) {
        this.subscriber = subscriber;
    }

    public Payment() {
    }

    public Payment(Long paymentId, int paidAmount, String paymentDate, String
paymentMode, Subscriber subscriber) {
        this.paymentId = paymentId;
        this.paidAmount = paidAmount;
        this.paymentDate = paymentDate;
        this.paymentMode = paymentMode;
        this.subscriber = subscriber;
    }

    public Payment(int paidAmount, String paymentDate, String paymentMode, Subscriber
subscriber) {
        this.paidAmount = paidAmount;
        this.paymentDate = paymentDate;
        this.paymentMode = paymentMode;
        this.subscriber = subscriber;
    }

    public Long getPaymentId() {
        return paymentId;
    }

    public void setPaymentId(Long paymentId) {
        this.paymentId = paymentId;
    }

    public int getPaidAmount() {
        return paidAmount;
    }

    public void setPaidAmount(int paidAmount) {
        this.paidAmount = paidAmount;
    }

    public String getPaymentDate() {
        return paymentDate;
```

```java
    }

    public void setPaymentDate(String paymentDate) {
        this.paymentDate = paymentDate;
    }

    public String getPaymentMode() {
        return paymentMode;
    }

    public void setPaymentMode(String paymentMode) {
        this.paymentMode = paymentMode;
    }
}
```

```java
package com.project.fitnessclubautomation.model;

import javax.persistence.*;

@Entity
@Table(name = "tbl_subscriber")
public class Subscriber {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "subscriber_id", nullable = false)
    private Long subscriberId;
    @Column(name = "subscriber_name", nullable = false)
    private String subscriberName;
    @Column(name = "subscriber_age", nullable = false)
    private int subscriberAge;
    @Column(name = "subscriber_gender", nullable = false)
    private String subscriberGender;
    @Column(name = "subscriber_address", nullable = false)
    private String subscriberAddress;
    @Column(name = "subscriber_status", nullable = false)
    private boolean subscriberStatus;

    @Column(name = "subscriber_fees_paid", nullable = false)
    private int subscriberFeesPaid;

    @ManyToOne
    @JoinColumn(name = "subscriber_trainer_id")
    private Trainer trainer;
    @ManyToOne
    @JoinColumn(name = "subscription_plan_id")
    private SubscriptionPlan subscriptionPlan;

    public Subscriber() {
    }
    public Subscriber(Long subscriberId, String subscriberName, int subscriberAge,
String subscriberGender, String subscriberAddress, boolean subscriberStatus, int
subscriberFeesPaid, Trainer trainer, SubscriptionPlan subscriptionPlan) {
        this.subscriberId = subscriberId;
        this.subscriberName = subscriberName;
        this.subscriberAge = subscriberAge;
        this.subscriberGender = subscriberGender;
        this.subscriberAddress = subscriberAddress;
        this.subscriberStatus = subscriberStatus;
        this.subscriberFeesPaid = subscriberFeesPaid;
        this.trainer = trainer;
        this.subscriptionPlan = subscriptionPlan;
    }

    public Subscriber(String subscriberName, int subscriberAge, String
subscriberGender, String subscriberAddress, boolean subscriberStatus, int
subscriberFeesPaid, Trainer trainer, SubscriptionPlan subscriptionPlan) {
        this.subscriberName = subscriberName;
        this.subscriberAge = subscriberAge;
        this.subscriberGender = subscriberGender;
        this.subscriberAddress = subscriberAddress;
        this.subscriberStatus = subscriberStatus;
        this.subscriberFeesPaid = subscriberFeesPaid;
        this.trainer = trainer;
        this.subscriptionPlan = subscriptionPlan;
    }

    public Long getSubscriberId() {
        return subscriberId;
    }

    public void setSubscriberId(Long subscriberId) {
        this.subscriberId = subscriberId;
    }

    public String getSubscriberName() {
        return subscriberName;
    }

    public void setSubscriberName(String subscriberName) {
        this.subscriberName = subscriberName;
    }

    public int getSubscriberAge() {
        return subscriberAge;
    }

    public void setSubscriberAge(int subscriberAge) {
        this.subscriberAge = subscriberAge;
    }

    public String getSubscriberGender() {
        return subscriberGender;
    }
```

```java
    public void setSubscriberGender(String subscriberGender) {
        this.subscriberGender = subscriberGender;
    }

    public String getSubscriberAddress() {
        return subscriberAddress;
    }

    public void setSubscriberAddress(String subscriberAddress) {
        this.subscriberAddress = subscriberAddress;
    }

    public boolean isSubscriberStatus() {
        return subscriberStatus;
    }

    public void setSubscriberStatus(boolean subscriberStatus) {
        this.subscriberStatus = subscriberStatus;
    }

    public int getSubscriberFeesPaid() {
        return subscriberFeesPaid;
    }

    public void setSubscriberFeesPaid(int subscriberFeesPaid) {
        this.subscriberFeesPaid = subscriberFeesPaid;
    }

    public Trainer getTrainer() {
        return trainer;
    }

    public void setTrainer(Trainer trainer) {
        this.trainer = trainer;
    }

    public SubscriptionPlan getSubscriptionPlan() {
        return subscriptionPlan;
    }

    public void setSubscriptionPlan(SubscriptionPlan subscriptionPlan) {
        this.subscriptionPlan = subscriptionPlan;
    }
}
```

```java
package com.project.fitnessclubautomation.model;

import javax.persistence.*;
import java.util.List;


@Entity
@Table(name = "tbl_subscriptionplan")
public class SubscriptionPlan {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  @Column(name = "plan_id", nullable = false)
  private Long planId;
  @Column(name = "plan_title", nullable = false)
  private String planTitle;
  @Column(name = "plan_duration", nullable = false)
  private int planDuration;
  @Column(name = "plan_fees", nullable = false)
  private int planFees;


  public SubscriptionPlan() {
  }


  public SubscriptionPlan(Long planId, String planTitle, int planDuration,
int planFees) {
    this.planId = planId;
    this.planTitle = planTitle;
    this.planDuration = planDuration;
    this.planFees = planFees;
  }
```

```
    public SubscriptionPlan(String planTitle, int planDuration, int
planFees) {
        this.planTitle = planTitle;
        this.planDuration = planDuration;
        this.planFees = planFees;
    }

    public Long getPlanId() {
        return planId;
    }

    public void setPlanId(Long planId) {
        this.planId = planId;
    }

    public String getPlanTitle() {
        return planTitle;
    }

    public void setPlanTitle(String planTitle) {
        this.planTitle = planTitle;
    }

    public int getPlanDuration() {
        return planDuration;
    }

    public void setPlanDuration(int planDuration) {
```

```
                    this.planDuration = planDuration;
                }

                public int getPlanFees() {
                    return planFees;
                }

                public void setPlanFees(int planFees) {
                    this.planFees = planFees;
                }
            }
```

```
package com.project.fitnessclubautomation.model;

import javax.persistence.*;

@Entity
@Table(name = "tbl_trainer")
public class Trainer {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "trainer_id", nullable = false)
    private Long trainerId;
    @Column(name = "trainer_name", nullable = false)
    private String trainerName;
    @Column(name = "trainer_age", nullable = false)
    private int trainerAge;
    @Column(name = "trainer_gender", nullable = false)
    private String trainerGender;
    @Column(name = "trainer_experience", nullable = false)
    private int trainerExperience;
    @Column(name = "trainer_address", nullable = false)
    private String trainerAddress;

    public Trainer() {
    }

    public Trainer(Long trainerId, String trainerName, int trainerAge, String
trainerGender, int trainerExperience, String trainerAddress) {
        this.trainerId = trainerId;
        this.trainerName = trainerName;
        this.trainerAge = trainerAge;
        this.trainerGender = trainerGender;
        this.trainerExperience = trainerExperience;
        this.trainerAddress = trainerAddress;
    }

    public Trainer(String trainerName, int trainerAge, String trainerGender, int 
trainerExperience, String trainerAddress) {
        this.trainerName = trainerName;
        this.trainerAge = trainerAge;
        this.trainerGender = trainerGender;
        this.trainerExperience = trainerExperience;
        this.trainerAddress = trainerAddress;
    }

    public Long getTrainerId() {
        return trainerId;
    }

    public void setTrainerId(Long trainerId) {
        this.trainerId = trainerId;
    }

    public String getTrainerName() {
        return trainerName;
    }

    public void setTrainerName(String trainerName) {
        this.trainerName = trainerName;
    }

    public int getTrainerAge() {
        return trainerAge;
    }

    public void setTrainerAge(int trainerAge) {
        this.trainerAge = trainerAge;
    }

    public String getTrainerGender() {
        return trainerGender;
    }

    public void setTrainerGender(String trainerGender) {
        this.trainerGender = trainerGender;
    }

    public int getTrainerExperience() {
        return trainerExperience;
    }

    public void setTrainerExperience(int trainerExperience) {
        this.trainerExperience = trainerExperience;
    }
    public String getTrainerAddress() {
        return trainerAddress;
    }

    public void setTrainerAddress(String trainerAddress) {
        this.trainerAddress = trainerAddress;
    }
}
```

```
package com.project.fitnessclubautomation.repository;

import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;

import com.project.fitnessclubautomation.model.Admin;

public interface AdminRepository extends CrudRepository<Admin,
Long> {
   @Query("select a from Admin a where a.adminEmail = ?1 and
a.adminPassword = ?2")
   Admin login(String adminEmail, String adminPassword);



}
```

```
package com.project.fitnessclubautomation.repository;

import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;
import com.project.fitnessclubautomation.model.Payment;

@Repository
public interface PaymentRepository extends CrudRepository<Payment,
Long> {
}

package com.project.fitnessclubautomation.repository;

import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import com.project.fitnessclubautomation.model.Subscriber;

@Repository
public interface SubscriberRepository extends
CrudRepository<Subscriber, Long> {
}



package com.project.fitnessclubautomation.repository;

import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import com.project.fitnessclubautomation.model.SubscriptionPlan;
```

```
@Repository
public interface SubscriptionPlanRepository extends
CrudRepository<SubscriptionPlan, Long> {
}



package com.project.fitnessclubautomation.repository;

import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import com.project.fitnessclubautomation.model.Trainer;

@Repository
public interface TrainerRepository extends CrudRepository<Trainer,
Long> {
}




package com.project.fitnessclubautomation.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.project.fitnessclubautomation.model.Admin;
import com.project.fitnessclubautomation.repository.AdminRepository;
```

```
@Service
public class AdminService {
    @Autowired
    AdminRepository adminRepository;

    public Admin getAdmin(Long adminId) {
        return adminRepository.findById(adminId).get();
    }

    public Admin login(String adminEmail, String adminPassword) {
        Admin admin = adminRepository.login(adminEmail,
adminPassword);
        return admin;
    }
}



package com.project.fitnessclubautomation.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.project.fitnessclubautomation.model.Payment;
import
com.project.fitnessclubautomation.repository.PaymentRepository;

import java.util.ArrayList;
import java.util.List;


@Service
public class PaymentService {
   @Autowired
   PaymentRepository paymentRepository;


   public Payment addPayment(Payment p) {
      return paymentRepository.save(p);
   }


   public void deletePayment(Long payment_id) {
      paymentRepository.deleteById(payment_id);
   }


   public List<Payment> getAllPayments() {
      List<Payment> payments = new ArrayList<>();
      paymentRepository.findAll().forEach(payments::add);
      return payments;
   }


   public Payment getPayment(Long payment_id) {
      return paymentRepository.findById(payment_id).get();
   }
}



package com.project.fitnessclubautomation.service;
```

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

import com.project.fitnessclubautomation.model.Subscriber;
import
com.project.fitnessclubautomation.repository.SubscriberRepository;

import java.util.ArrayList;
import java.util.List;

@Service
public class SubscriberService {
  @Autowired
  SubscriberRepository subscriberRepository;

  public Subscriber addSubscriber(Subscriber subscriber) {
    return subscriberRepository.save(subscriber);
  }

  public void deleteSubscriber(Long subscriber_id) {
    subscriberRepository.deleteById(subscriber_id);
  }

  public List<Subscriber> getAllSubscribers() {
    List<Subscriber> subscribers = new ArrayList<>();
    subscriberRepository.findAll().forEach(subscribers::add);
    return subscribers;
    }

    public Subscriber getSubscriber(Long subscriber_id) {
        return subscriberRepository.findById(subscriber_id).get();
    }
}



package com.project.fitnessclubautomation.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.project.fitnessclubautomation.model.SubscriptionPlan;
import
com.project.fitnessclubautomation.repository.SubscriptionPlanReposito
ry;

import java.util.ArrayList;
import java.util.List;

@Service
public class SubscriptionPlanService {
    @Autowired
    SubscriptionPlanRepository subscriptionPlanRepository;

    public SubscriptionPlan addSubscriptionPlan(SubscriptionPlan
subscriptionPlan) {
        return subscriptionPlanRepository.save(subscriptionPlan);
    }

    public void deleteSubscriptionPlan(Long subscriptionPlan_id) {
        subscriptionPlanRepository.deleteById(subscriptionPlan_id);
    }

    public List<SubscriptionPlan> getAllSubscriptionPlans() {
        List<SubscriptionPlan> subscriptionPlans = new ArrayList<>();

subscriptionPlanRepository.findAll().forEach(subscriptionPlans::add);
        return subscriptionPlans;
    }

    public SubscriptionPlan getSubscriptionPlan(Long
subscriptionPlan_id) {
        return
subscriptionPlanRepository.findById(subscriptionPlan_id).get();
    }
}




package com.project.fitnessclubautomation.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.project.fitnessclubautomation.model.Trainer;
import
com.project.fitnessclubautomation.repository.TrainerRepository;

import java.util.ArrayList;
import java.util.List;

@Service
public class TrainerService {
  @Autowired
  TrainerRepository trainerRepository;

  public Trainer addTrainer(Trainer tr) {
    return trainerRepository.save(tr);
  }

  public void deleteTrainer(Long trainer_id) {
    trainerRepository.deleteById(trainer_id);
  }

  public List<Trainer> getAllTrainers() {
    List<Trainer> trainers = new ArrayList<>();
    trainerRepository.findAll().forEach(trainers::add);
    return trainers;
  }

  public Trainer getTrainer(Long trainer_id) {
    return trainerRepository.findById(trainer_id).get();
```

```
        }

      }
```

Application Properties.

```
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost:3306/mphdb1
spring.datasource.username=root
spring.datasource.password=18W91A0477
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.show-sql=true
#Server Configuration
server.port=8086
spring.mvc.view.prefix=/views/
spring.mvc.view.suffix=.jsp
```