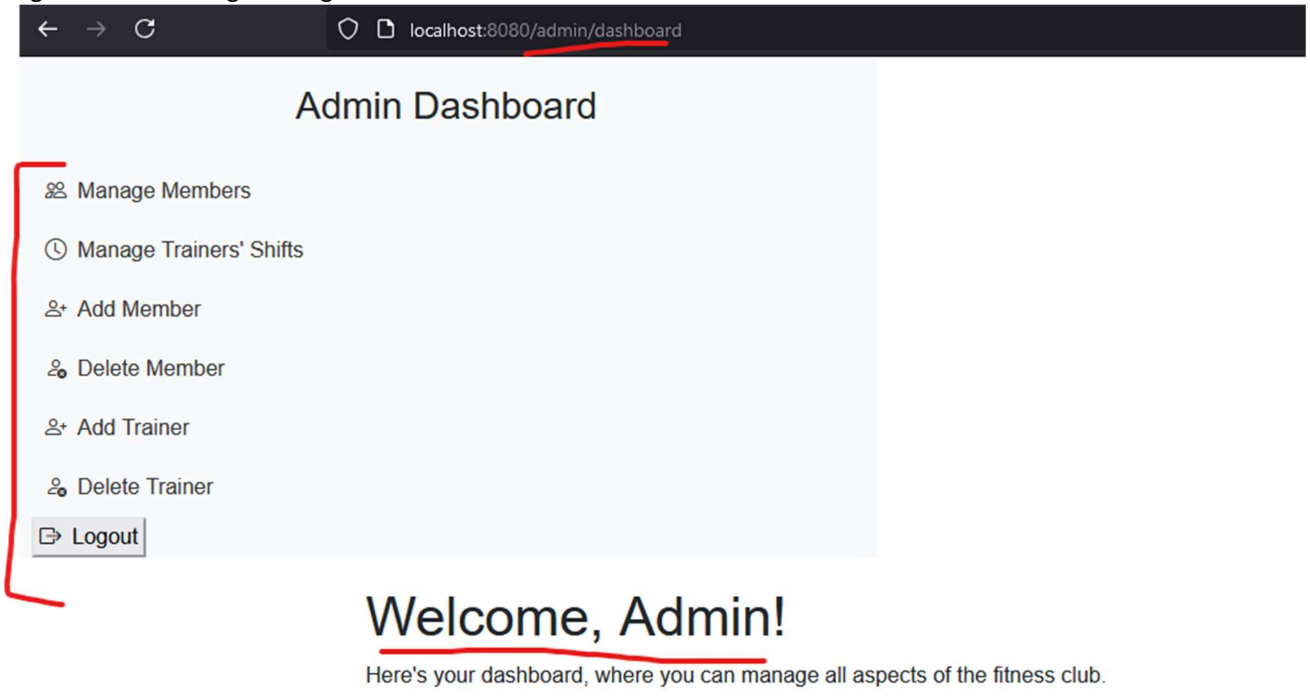
Marc Remillard

Final Project: Fitness Club Management System

Requirements: -

1. Admin:
   - Login: Admin can log in using credentials.

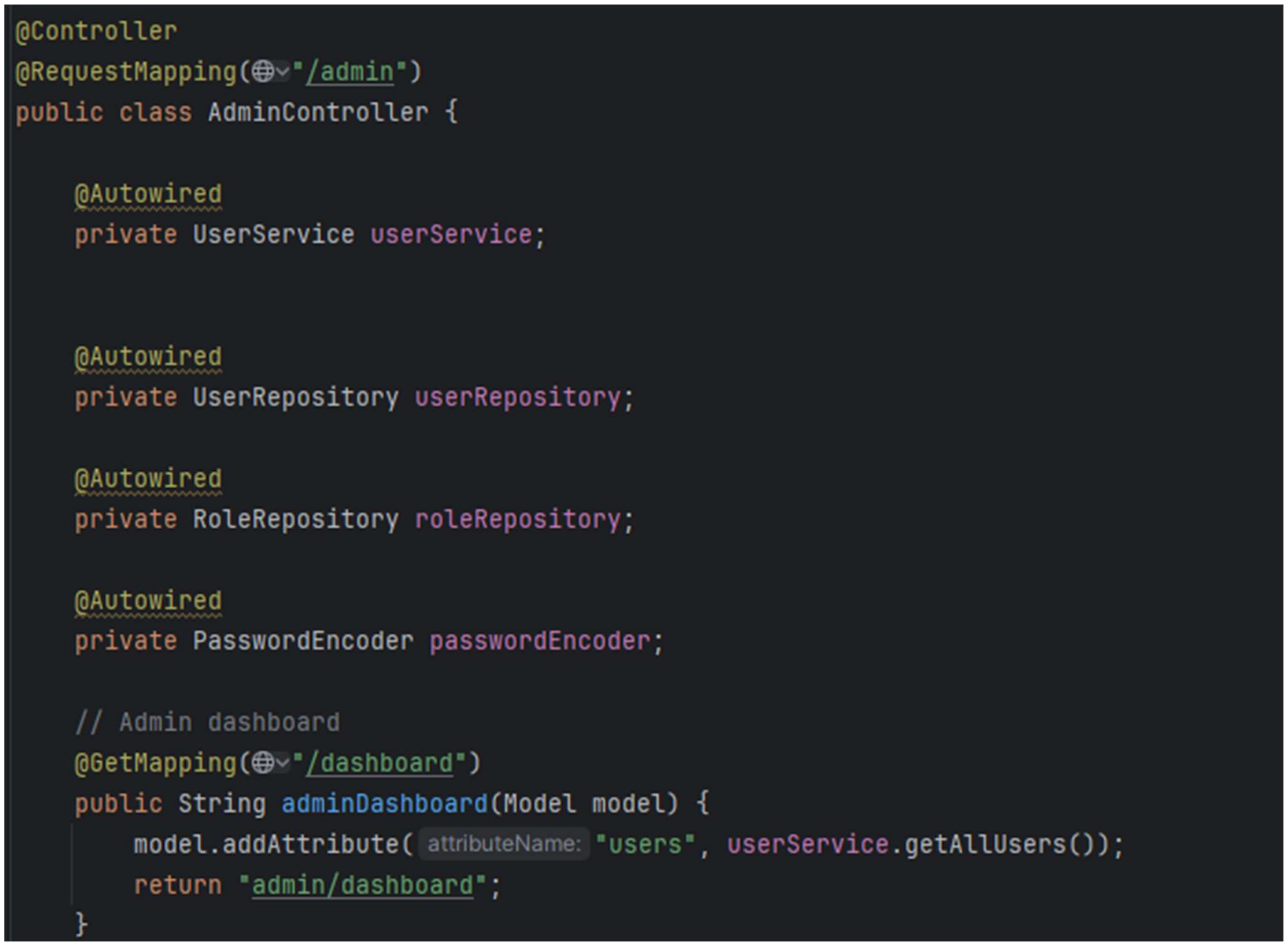


localhost:8080/admin/dashboard

Admin Dashboard

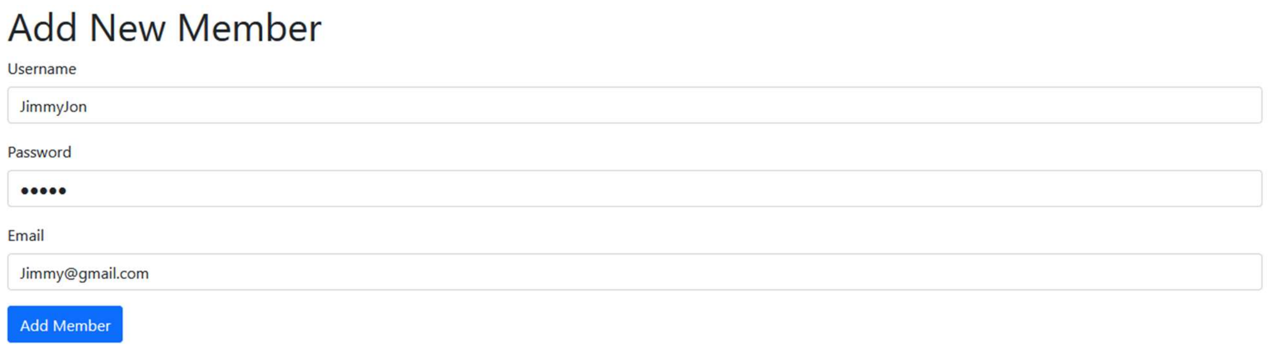
Welcome, Admin!

Here's your dashboard, where you can manage all aspects of the fitness club.



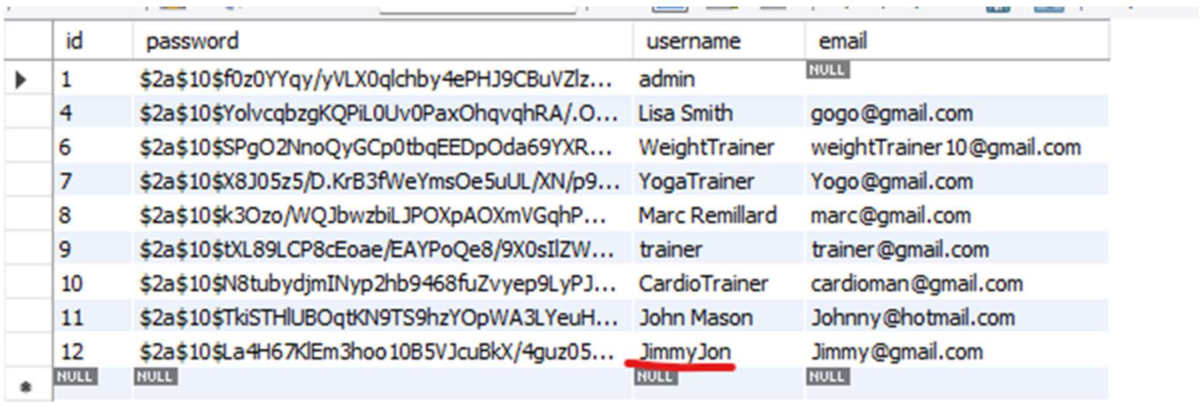
```java
@Controller
@RequestMapping("/admin")
public class AdminController {

    @Autowired
    private UserService userService;


    @Autowired
    private UserRepository userRepository;

    @Autowired
    private RoleRepository roleRepository;

    @Autowired
    private PasswordEncoder passwordEncoder;

    // Admin dashboard
    @GetMapping("/dashboard")
    public String adminDashboard(Model model) {
        model.addAttribute( attributeName: "users", userService.getAllUsers());
        return "admin/dashboard";
    }
```

- Add member: Admin can add a new member.

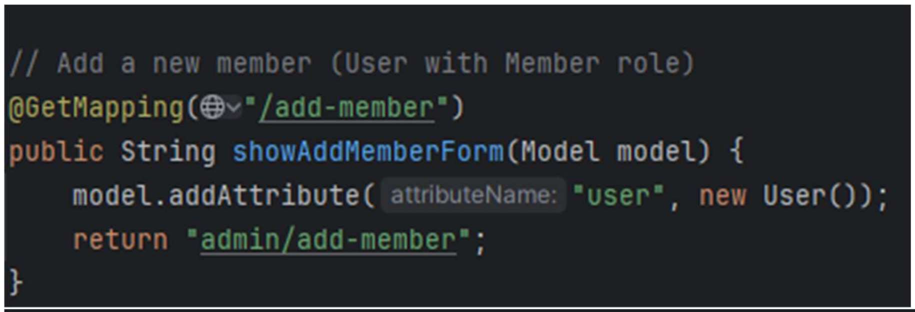


Add New Member

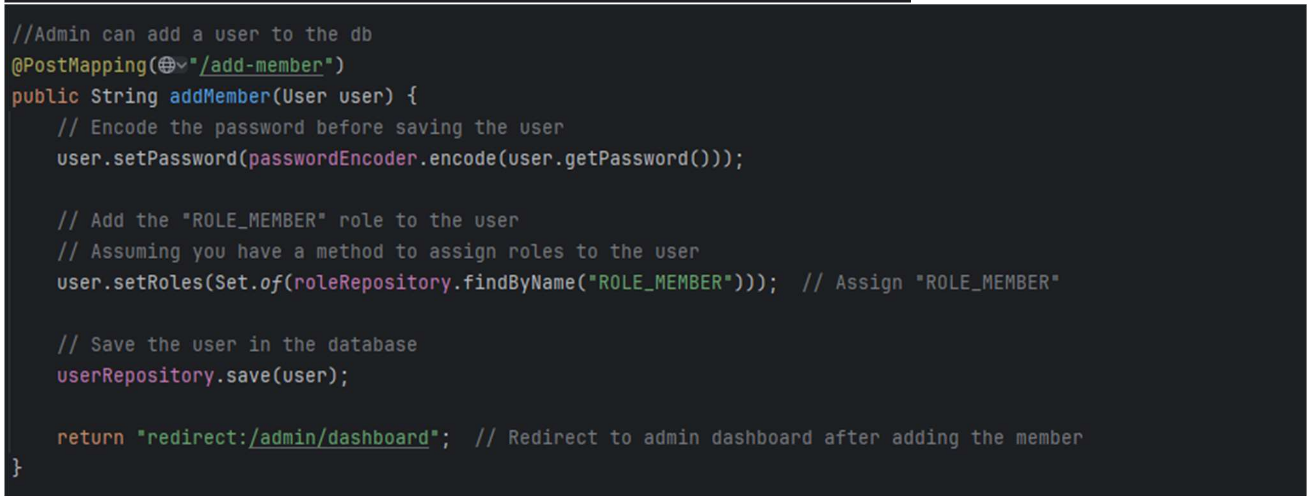
Username	JimmyJon
Password	•••••
Email	Jimmy@gmail.com



id	password	username	email
1	$2a$10$f0z0YYqy/yVLX0qlchby4ePHJ9CBuVZlz...	admin	NULL
4	$2a$10$YolvcqbzgKQPiL0Uv0PaxOhqvqhRA/.O...	Lisa Smith	gogo@gmail.com
6	$2a$10$SPgO2NnoQyGCp0tbqEEDpOda69YXR...	WeightTrainer	weightTrainer10@gmail.com
7	$2a$10$X8J05z5/D.KrB3fWeYmsOe5uUL/XN/p9...	YogaTrainer	Yogo@gmail.com
8	$2a$10$k3Ozo/WQJbwzbiLJPOXpAOXmVGqhP...	Marc Remillard	marc@gmail.com
9	$2a$10$tXL89LCP8cEoae/EAYPoQe8/9X0sIlZW...	trainer	trainer@gmail.com
10	$2a$10$N8tubydjmINyp2hb9468fuZvyep9LyPJ...	CardioTrainer	cardioman@gmail.com
11	$2a$10$TkiSTHlUBOqtKN9TS9hzYOpWA3LYeuH...	John Mason	Johnny@hotmail.com
12	$2a$10$La4H67KlEm3hoo10B5VJcuBkX/4guz05...	JimmyJon	Jimmy@gmail.com
NULL	NULL	NULL	NULL

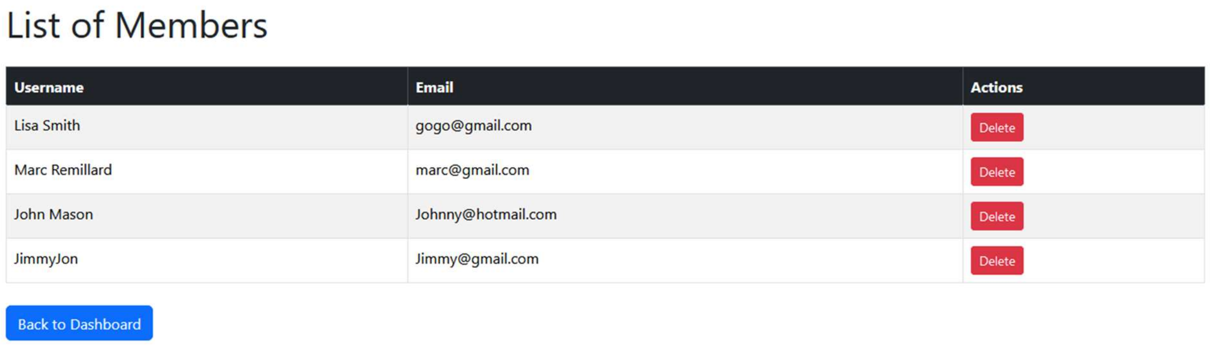


```java
// Add a new member (User with Member role)
@GetMapping("/add-member")
public String showAddMemberForm(Model model) {
    model.addAttribute( attributeName: "user", new User());
    return "admin/add-member";
}
```



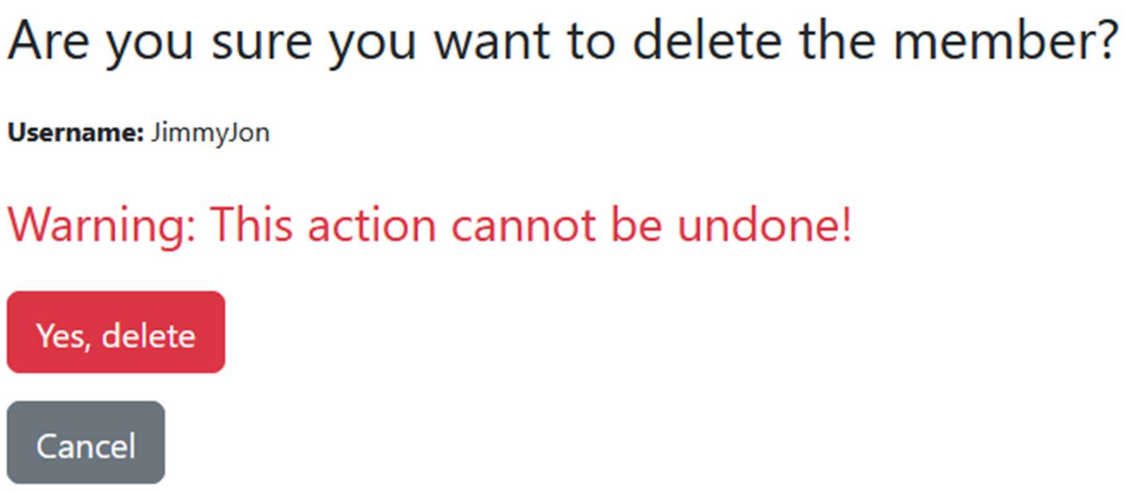
```java
//Admin can add a user to the db
@PostMapping("/add-member")
public String addMember(User user) {
    // Encode the password before saving the user
    user.setPassword(passwordEncoder.encode(user.getPassword()));

    // Add the "ROLE_MEMBER" role to the user
    // Assuming you have a method to assign roles to the user
    user.setRoles(Set.of(roleRepository.findByName("ROLE_MEMBER")));  // Assign "ROLE_MEMBER"

    // Save the user in the database
    userRepository.save(user);

    return "redirect:/admin/dashboard";  // Redirect to admin dashboard after adding the member
}
```

- Delete member: Admin can delete unwanted members.



List of Members

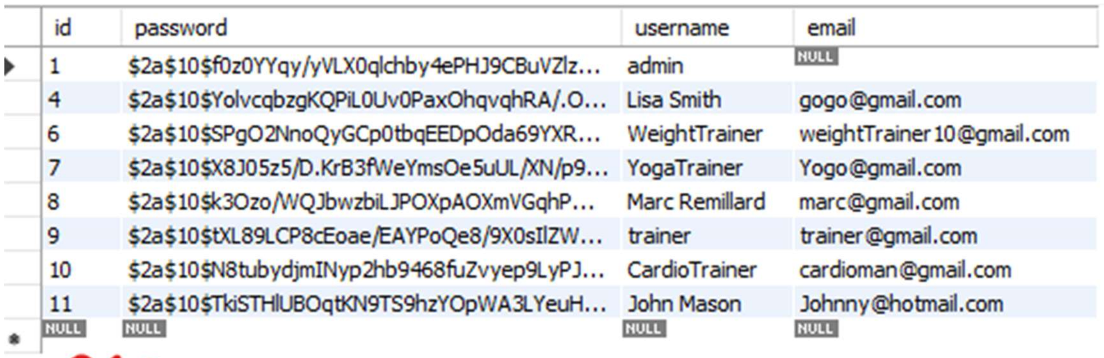
Username	Email	Actions
Lisa Smith	gogo@gmail.com	Delete
Marc Remillard	marc@gmail.com	Delete
John Mason	Johnny@hotmail.com	Delete
JimmyJon	Jimmy@gmail.com	Delete



Are you sure you want to delete the member?

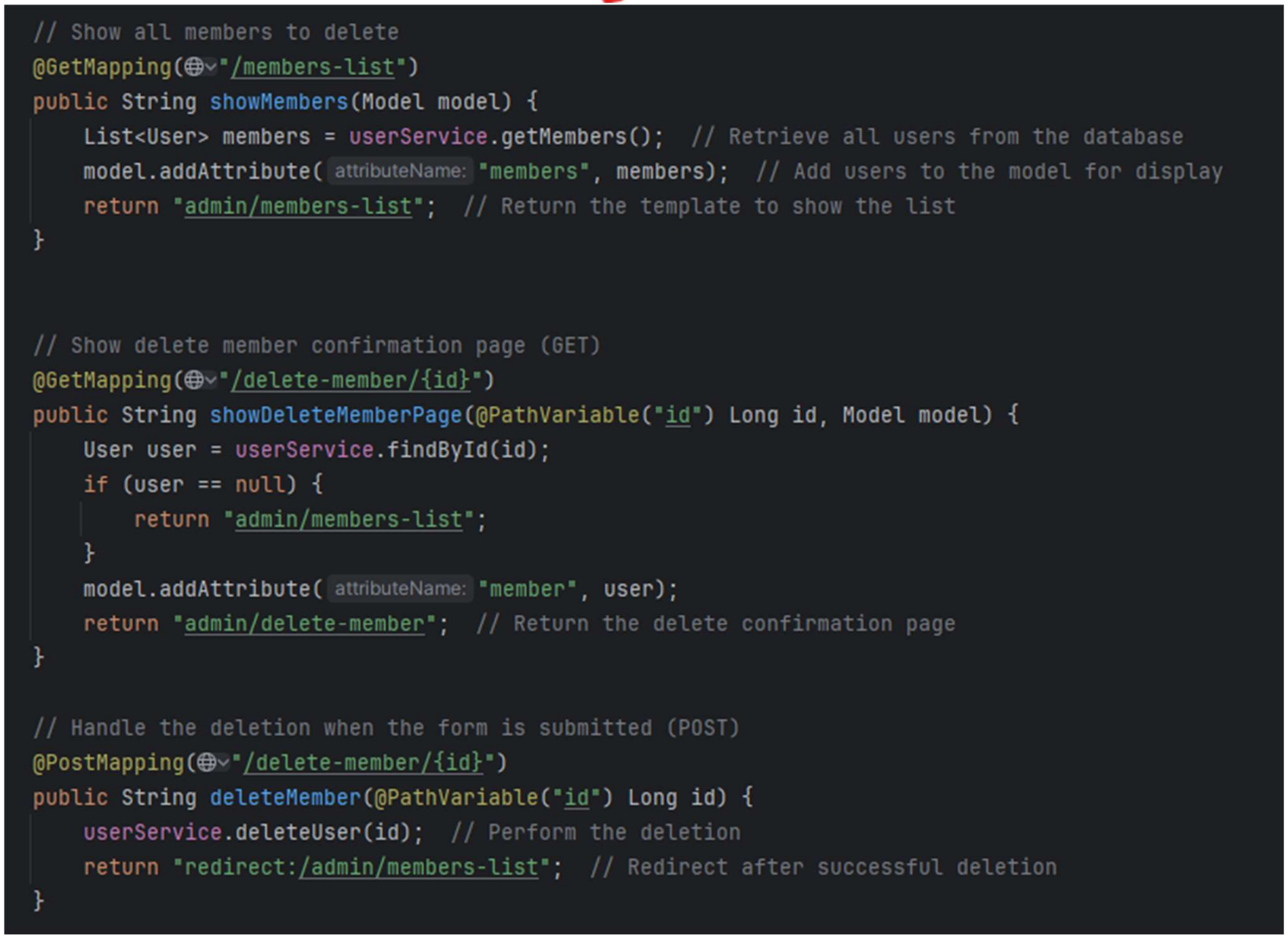
Username: JimmyJon

Warning: This action cannot be undone!



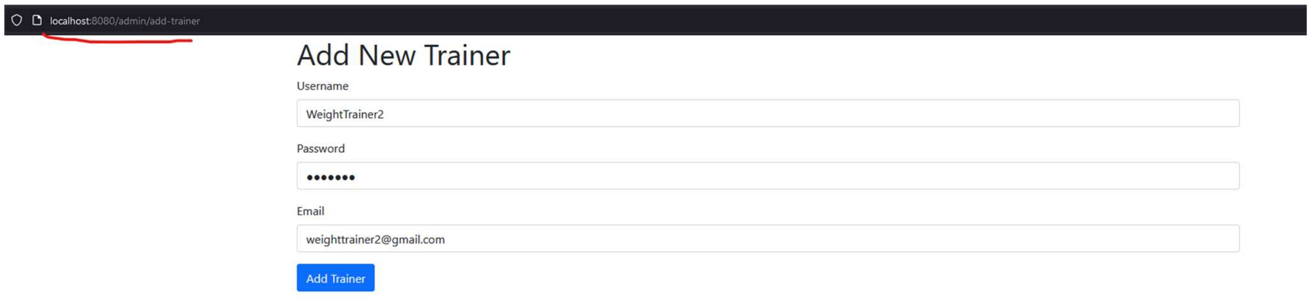
id	password	username	email
1	$2a$10$f0z0YYqy/yVLX0qlchby4ePHJ9CBuVZlz...	admin	NULL
4	$2a$10$YolvcqbzgKQPiL0Uv0PaxOhqvqhRA/.O...	Lisa Smith	gogo@gmail.com
6	$2a$10$SPgO2NnoQyGCp0tbqEEDpOda69YXR...	WeightTrainer	weightTrainer10@gmail.com
7	$2a$10$X8J05z5/D.KrB3fWeYmsOe5uUL/XN/p9...	YogaTrainer	Yogo@gmail.com
8	$2a$10$k3Ozo/WQJbwzbiLJPOXpAOXmVGqhP...	Marc Remillard	marc@gmail.com
9	$2a$10$tXL89LCP8cEoae/EAYPoQe8/9X0sIlZW...	trainer	trainer@gmail.com
10	$2a$10$N8tubydjmINyp2hb9468fuZvyep9LyPJ...	CardioTrainer	cardioman@gmail.com
11	$2a$10$TkiSTHlUBOqtKN9TS9hzYOpWA3LYeuH...	John Mason	Johnny@hotmail.com
NULL	NULL	NULL	NULL

no Jimmy



```
// Show all members to delete
@GetMapping("/members-list")
public String showMembers(Model model) {
    List<User> members = userService.getMembers();  // Retrieve all users from the database
    model.addAttribute( attributeName: "members", members);  // Add users to the model for display
    return "admin/members-list";  // Return the template to show the list
}


// Show delete member confirmation page (GET)
@GetMapping("/delete-member/{id}")
public String showDeleteMemberPage(@PathVariable("id") Long id, Model model) {
    User user = userService.findById(id);
    if (user == null) {
        return "admin/members-list";
    }
    model.addAttribute( attributeName: "member", user);
    return "admin/delete-member";  // Return the delete confirmation page
}

// Handle the deletion when the form is submitted (POST)
@PostMapping("/delete-member/{id}")
public String deleteMember(@PathVariable("id") Long id) {
    userService.deleteUser(id);  // Perform the deletion
    return "redirect:/admin/members-list";  // Redirect after successful deletion
}
```

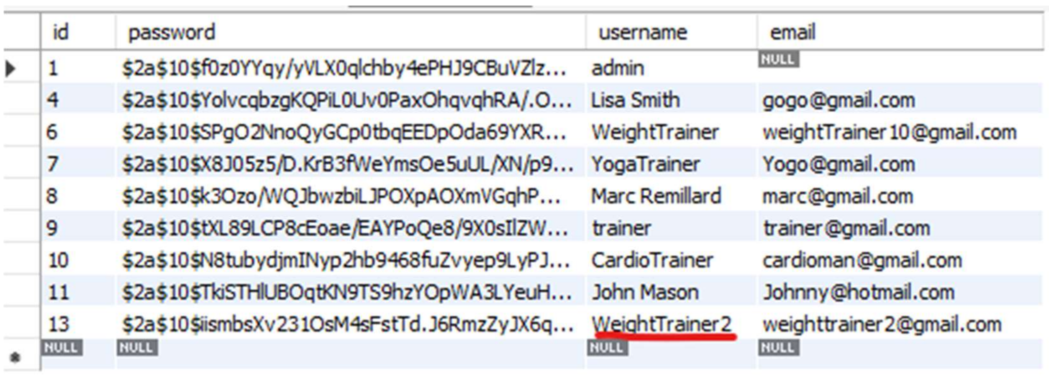
- Add Trainer: A new trainer can be added.



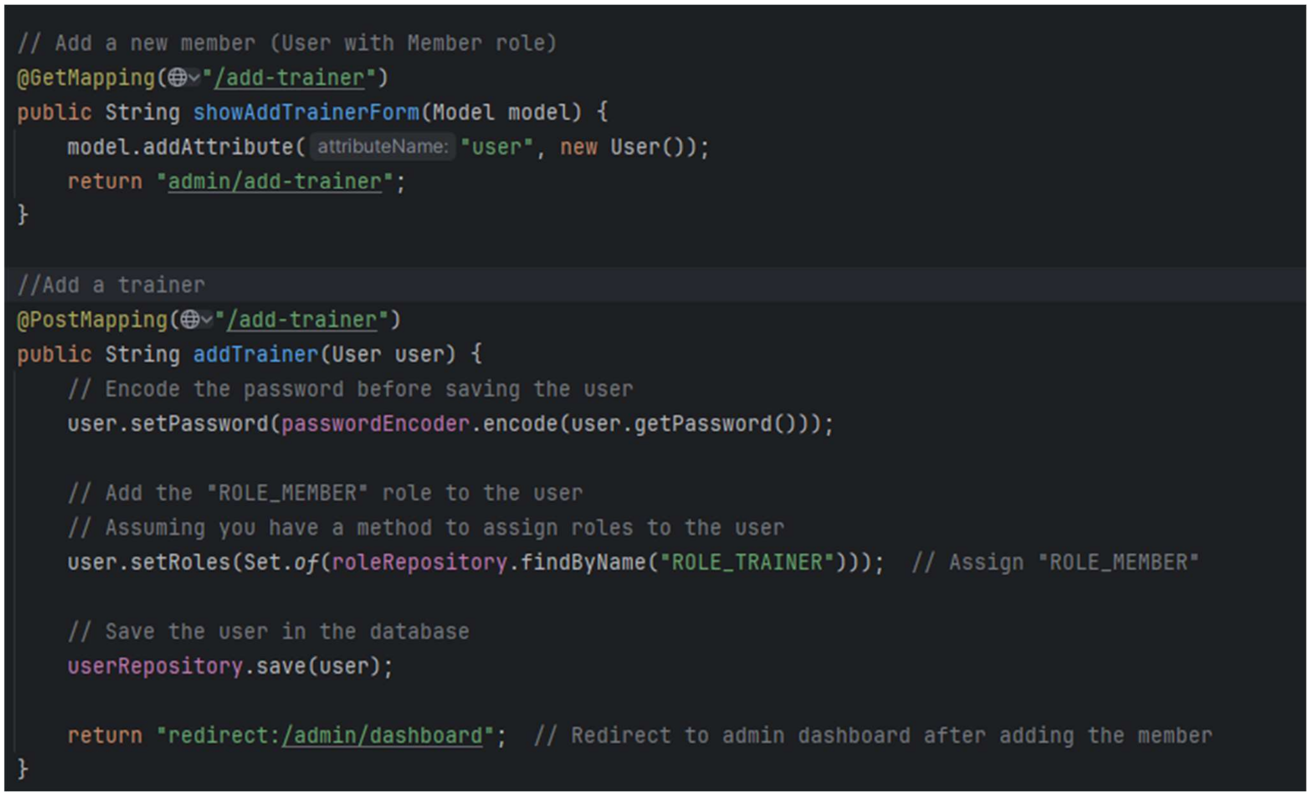
localhost:8080/admin/add-trainer

Add New Trainer

Username	WeightTrainer2
Password	•••••••
Email	weighttrainer2@gmail.com

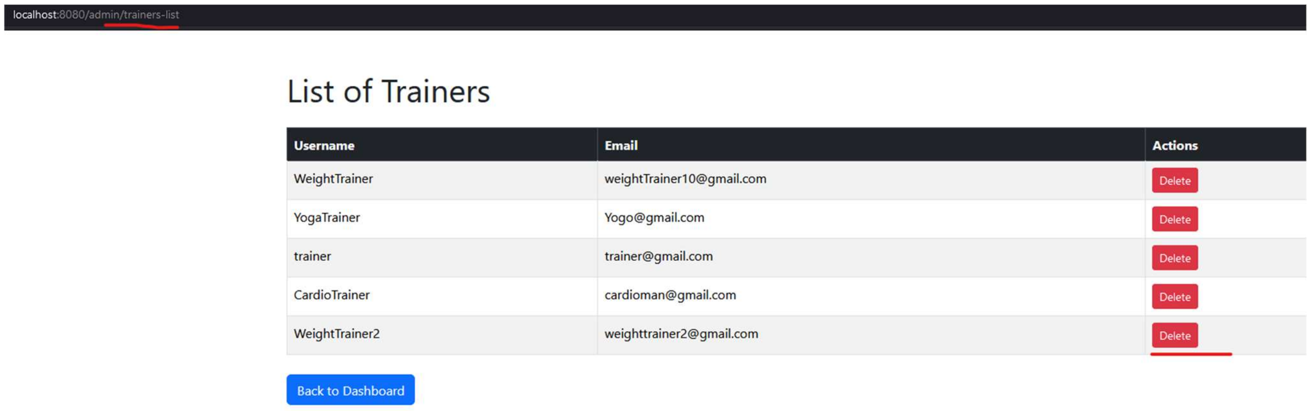


id	password	username	email
1	$2a$10$f0z0YYqy/yVLX0qlchby4ePHJ9CBuVZlz...	admin	NULL
4	$2a$10$YolvcqbzgKQPiL0Uv0PaxOhqvqhRA/.O...	Lisa Smith	gogo@gmail.com
6	$2a$10$SPgO2NnoQyGCp0tbqEEDpOda69YXR...	WeightTrainer	weightTrainer10@gmail.com
7	$2a$10$X8J05z5/D.KrB3fWeYmsOe5uUL/XN/p9...	YogaTrainer	Yogo@gmail.com
8	$2a$10$k3Ozo/WQJbwzbiLJPOXpAOXmVGqhP...	Marc Remillard	marc@gmail.com
9	$2a$10$tXL89LCP8cEoae/EAYPoQe8/9X0sIlZW...	trainer	trainer@gmail.com
10	$2a$10$N8tubydjmINyp2hb9468fuZvyep9LyPJ...	CardioTrainer	cardioman@gmail.com
11	$2a$10$TkiSTHlUBOqtKN9TS9hzYOpWA3LYeuH...	John Mason	Johnny@hotmail.com
13	$2a$10$iismbsXv231OsM4sFstTd.J6RmzZyJX6q...	WeightTrainer2	weighttrainer2@gmail.com
NULL	NULL	NULL	NULL



```java
// Add a new member (User with Member role)
@GetMapping("/add-trainer")
public String showAddTrainerForm(Model model) {
    model.addAttribute( attributeName: "user", new User());
    return "admin/add-trainer";
}

//Add a trainer
@PostMapping("/add-trainer")
public String addTrainer(User user) {
    // Encode the password before saving the user
    user.setPassword(passwordEncoder.encode(user.getPassword()));

    // Add the "ROLE_MEMBER" role to the user
    // Assuming you have a method to assign roles to the user
    user.setRoles(Set.of(roleRepository.findByName("ROLE_TRAINER")));  // Assign "ROLE_MEMBER"

    // Save the user in the database
    userRepository.save(user);

    return "redirect:/admin/dashboard";  // Redirect to admin dashboard after adding the member
}
```

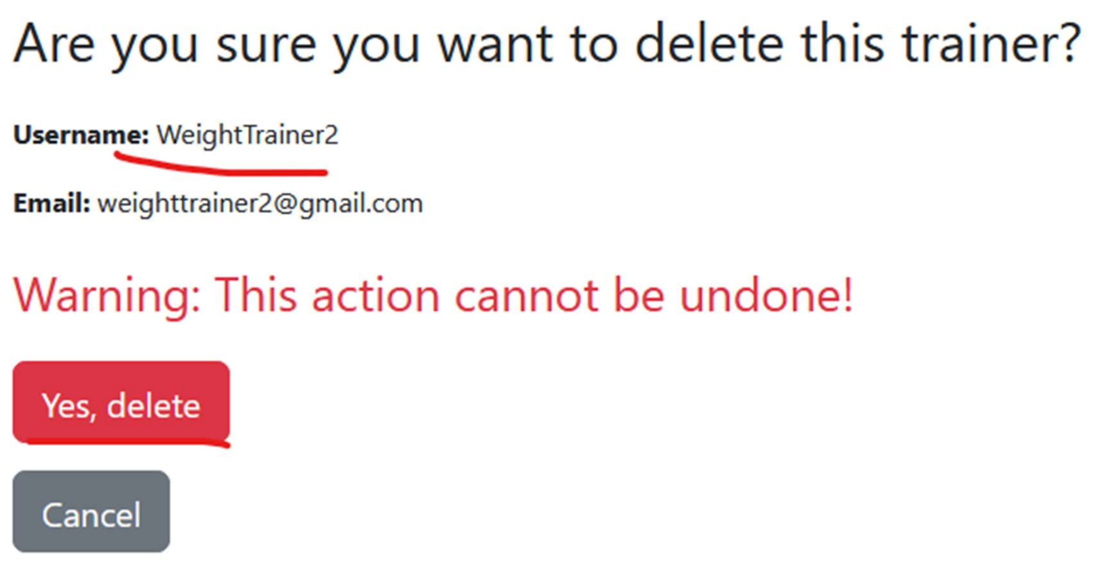
- Delete trainer: Unwanted trainers can be deleted.



localhost:8080/admin/trainers-list

List of Trainers

Username	Email	Actions
WeightTrainer	weightTrainer10@gmail.com	Delete
YogaTrainer	Yogo@gmail.com	Delete
trainer	trainer@gmail.com	Delete
CardioTrainer	cardioman@gmail.com	Delete
WeightTrainer2	weighttrainer2@gmail.com	Delete

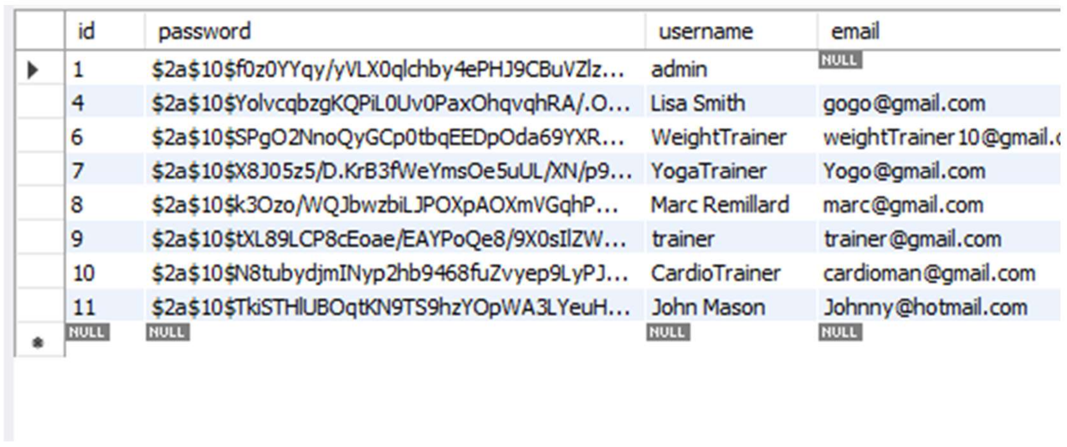


Are you sure you want to delete this trainer?

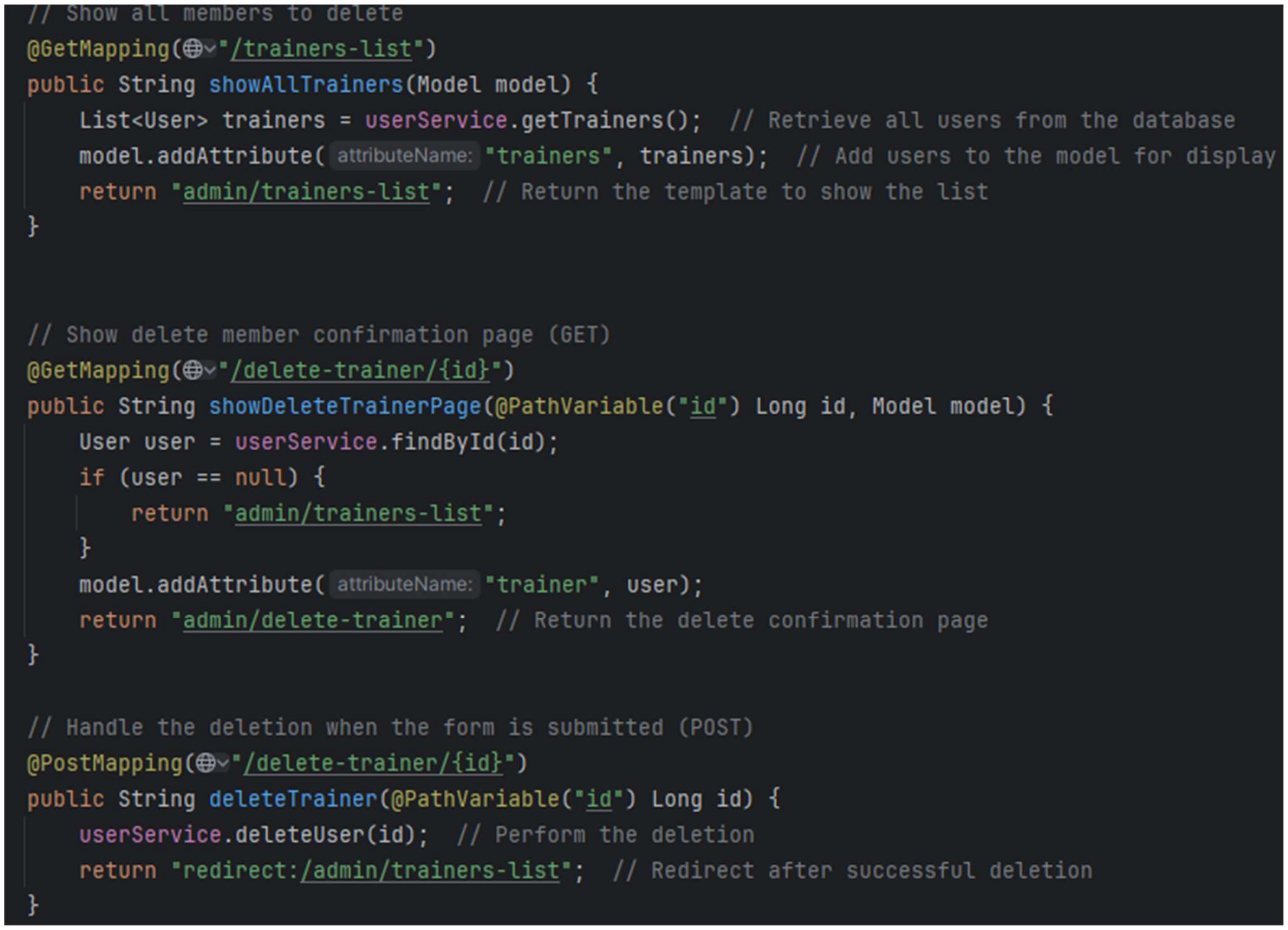
Username: WeightTrainer2

Email: weighttrainer2@gmail.com

Warning: This action cannot be undone!

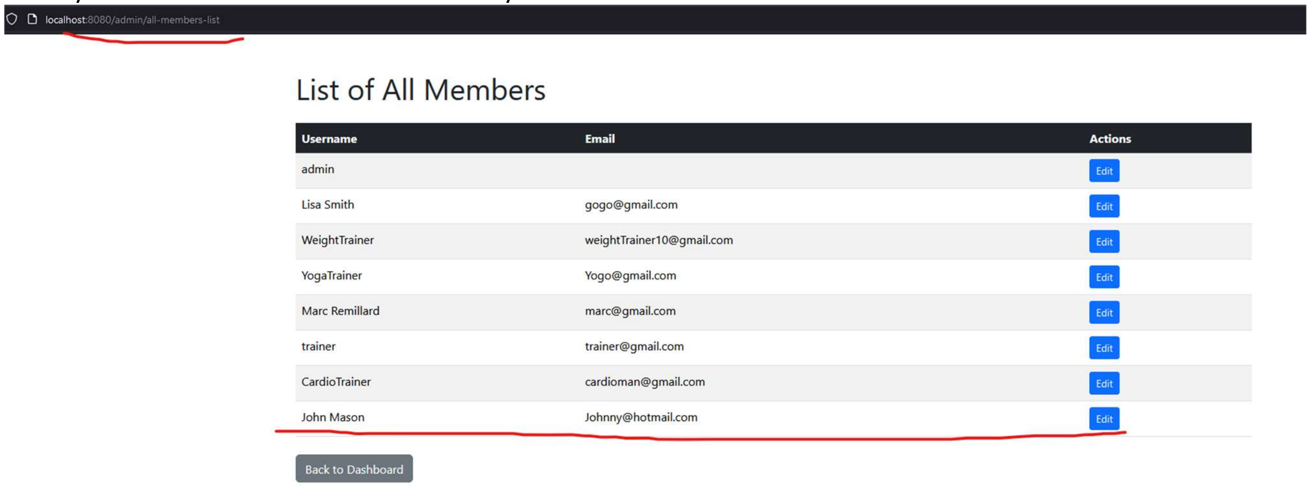


id	password	username	email
1	$2a$10$f0z0YYqy/yVLX0qlchby4ePHJ9CBuVZlz...	admin	NULL
4	$2a$10$YolvcqbzgKQPiL0Uv0PaxOhqvqhRA/.O...	Lisa Smith	gogo@gmail.com
6	$2a$10$SPgO2NnoQyGCp0tbqEEDpOda69YXR...	WeightTrainer	weightTrainer10@gmail.c
7	$2a$10$X8J05z5/D.KrB3fWeYmsOe5uUL/XN/p9...	YogaTrainer	Yogo@gmail.com
8	$2a$10$k3Ozo/WQJbwzbiLJPOXpAOXmVGqhP...	Marc Remillard	marc@gmail.com
9	$2a$10$tXL89LCP8cEoae/EAYPoQe8/9X0sIlZW...	trainer	trainer@gmail.com
10	$2a$10$N8tubydjmINyp2hb9468fuZvyep9LyPJ...	CardioTrainer	cardioman@gmail.com
11	$2a$10$TkiSTHlUBOqtKN9TS9hzYOpWA3LYeuH...	John Mason	Johnny@hotmail.com
NULL	NULL	NULL	NULL



```
// Show all members to delete
@GetMapping("/trainers-list")
public String showAllTrainers(Model model) {
    List<User> trainers = userService.getTrainers();  // Retrieve all users from the database
    model.addAttribute( attributeName: "trainers", trainers);  // Add users to the model for display
    return "admin/trainers-list";  // Return the template to show the list
}


// Show delete member confirmation page (GET)
@GetMapping("/delete-trainer/{id}")
public String showDeleteTrainerPage(@PathVariable("id") Long id, Model model) {
    User user = userService.findById(id);
    if (user == null) {
        return "admin/trainers-list";
    }
    model.addAttribute( attributeName: "trainer", user);
    return "admin/delete-trainer";  // Return the delete confirmation page
}

// Handle the deletion when the form is submitted (POST)
@PostMapping("/delete-trainer/{id}")
public String deleteTrainer(@PathVariable("id") Long id) {
    userService.deleteUser(id);  // Perform the deletion
    return "redirect:/admin/trainers-list";  // Redirect after successful deletion
}
```

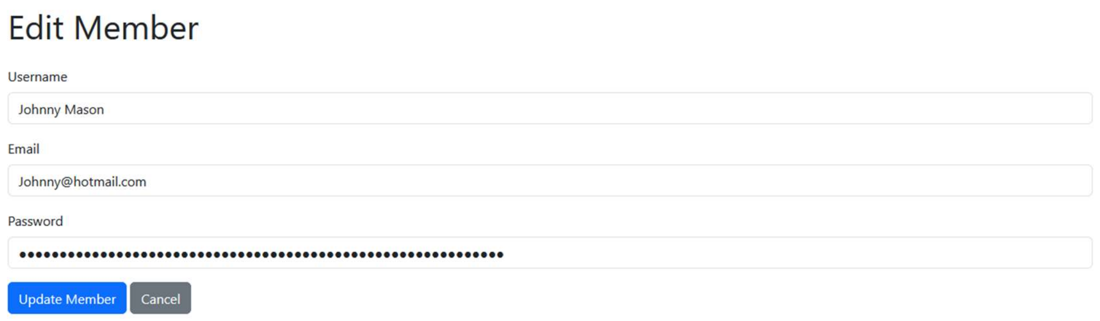
- Modify Member Data: The admin will modify the data of members.



localhost:8080/admin/all-members-list

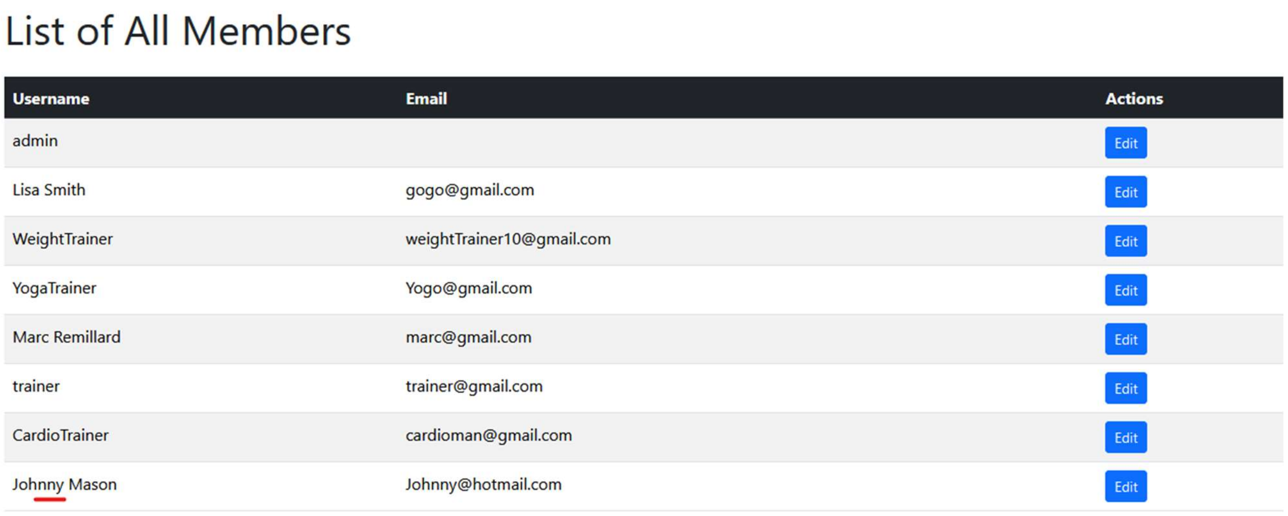
List of All Members

Username	Email	Actions
admin		Edit
Lisa Smith	gogo@gmail.com	Edit
WeightTrainer	weightTrainer10@gmail.com	Edit
YogaTrainer	Yogo@gmail.com	Edit
Marc Remillard	marc@gmail.com	Edit
trainer	trainer@gmail.com	Edit
CardioTrainer	cardioman@gmail.com	Edit
John Mason	Johnny@hotmail.com	Edit



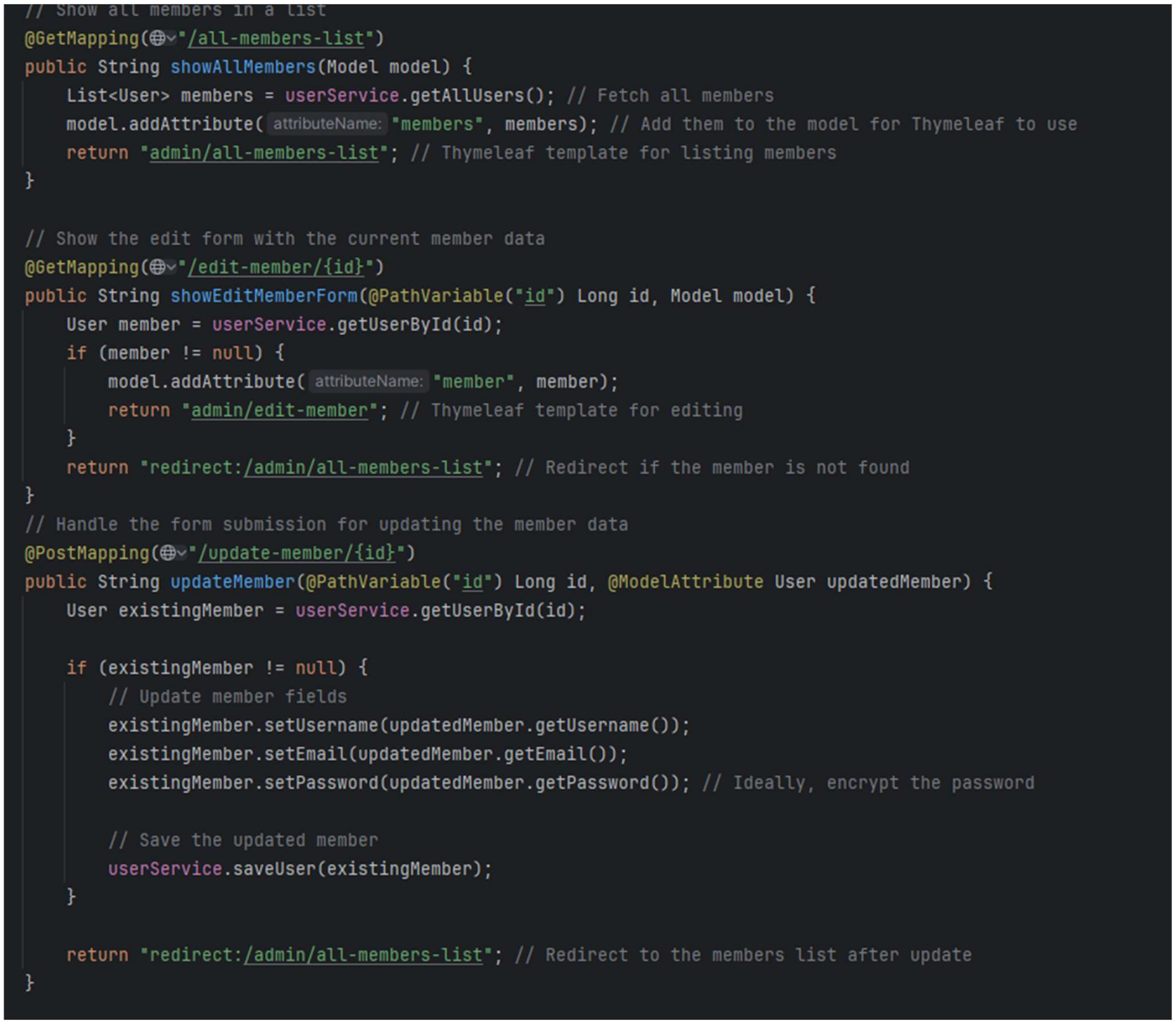
Edit Member

Username	Johnny Mason
Email	Johnny@hotmail.com
Password	••



List of All Members

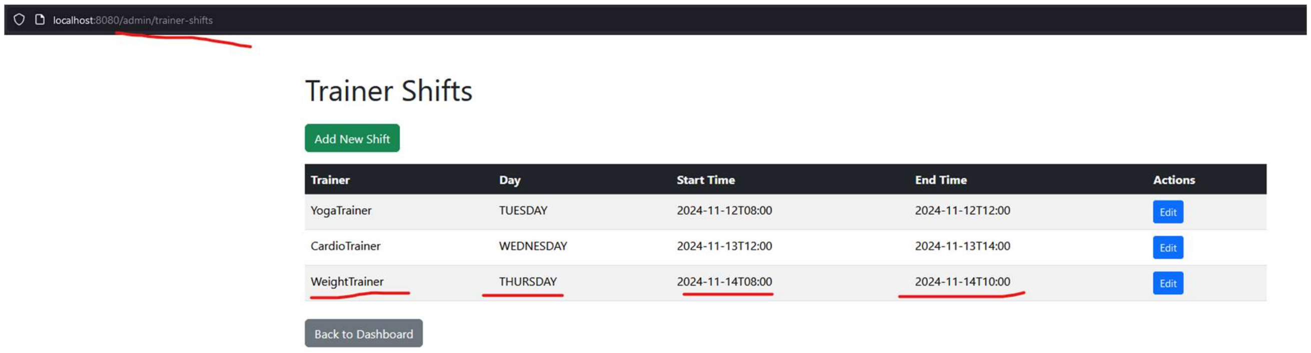
Username	Email	Actions
admin		Edit
Lisa Smith	gogo@gmail.com	Edit
WeightTrainer	weightTrainer10@gmail.com	Edit
YogaTrainer	Yogo@gmail.com	Edit
Marc Remillard	marc@gmail.com	Edit
trainer	trainer@gmail.com	Edit
CardioTrainer	cardioman@gmail.com	Edit
Johnny Mason	Johnny@hotmail.com	Edit



```
// Show all members in a list
@GetMapping("/all-members-list")
public String showAllMembers(Model model) {
    List<User> members = userService.getAllUsers(); // Fetch all members
    model.addAttribute( attributeName: "members", members); // Add them to the model for Thymeleaf to use
    return "admin/all-members-list"; // Thymeleaf template for listing members
}

// Show the edit form with the current member data
@GetMapping("/edit-member/{id}")
public String showEditMemberForm(@PathVariable("id") Long id, Model model) {
    User member = userService.getUserById(id);
    if (member != null) {
        model.addAttribute( attributeName: "member", member);
        return "admin/edit-member"; // Thymeleaf template for editing
    }
    return "redirect:/admin/all-members-list"; // Redirect if the member is not found
}
// Handle the form submission for updating the member data
@PostMapping("/update-member/{id}")
public String updateMember(@PathVariable("id") Long id, @ModelAttribute User updatedMember) {
    User existingMember = userService.getUserById(id);

    if (existingMember != null) {
        // Update member fields
        existingMember.setUsername(updatedMember.getUsername());
        existingMember.setEmail(updatedMember.getEmail());
        existingMember.setPassword(updatedMember.getPassword()); // Ideally, encrypt the password

        // Save the updated member
        userService.saveUser(existingMember);
    }

    return "redirect:/admin/all-members-list"; // Redirect to the members list after update
}
```

- Modify Trainers' Shift: The admin will modify trainers’ shifts.

Marc Remillard

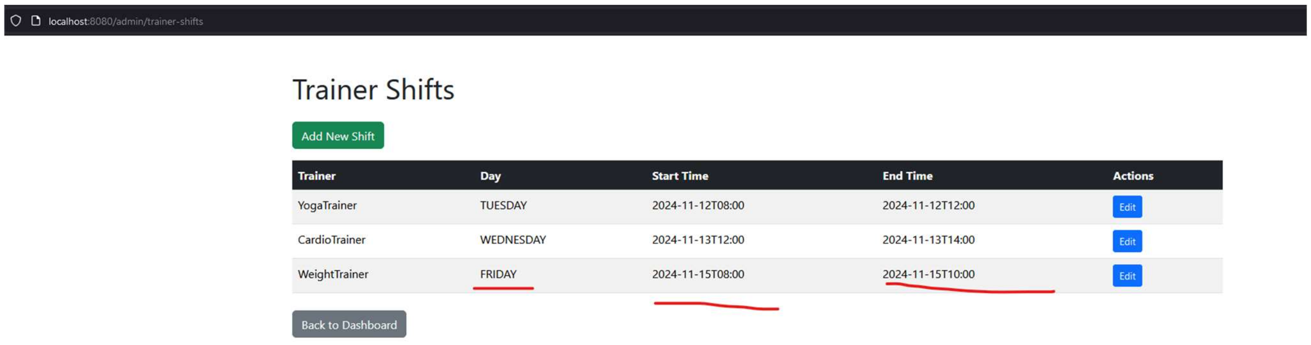

localhost:8080/admin/trainer-shifts

Trainer Shifts

Add New Shift

Trainer	Day	Start Time	End Time	Actions
YogaTrainer	TUESDAY	2024-11-12T08:00	2024-11-12T12:00	Edit
CardioTrainer	WEDNESDAY	2024-11-13T12:00	2024-11-13T14:00	Edit
WeightTrainer	THURSDAY	2024-11-14T08:00	2024-11-14T10:00	Edit

Back to Dashboard

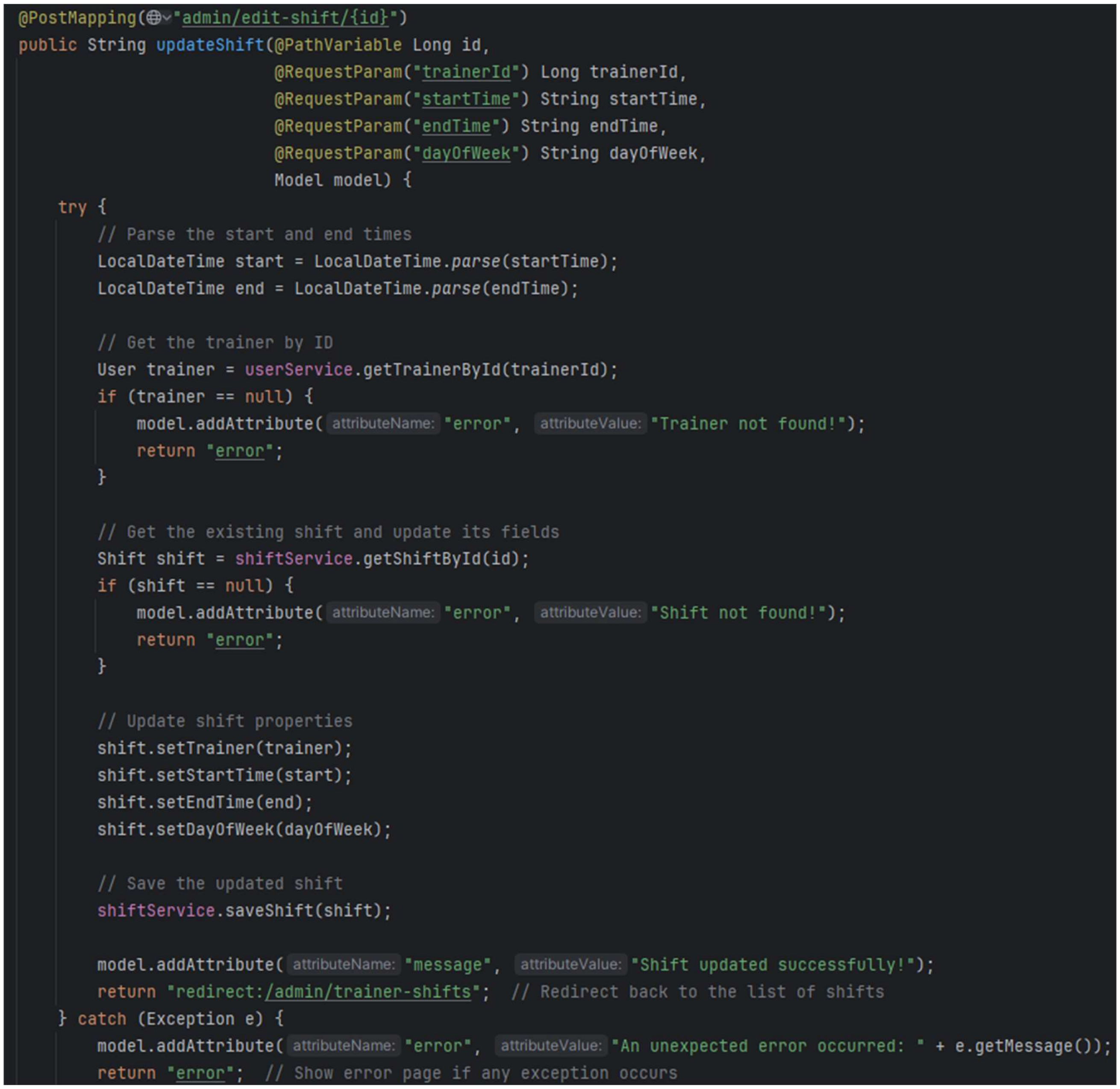

localhost:8080/admin/trainer-shifts

Trainer Shifts

Add New Shift

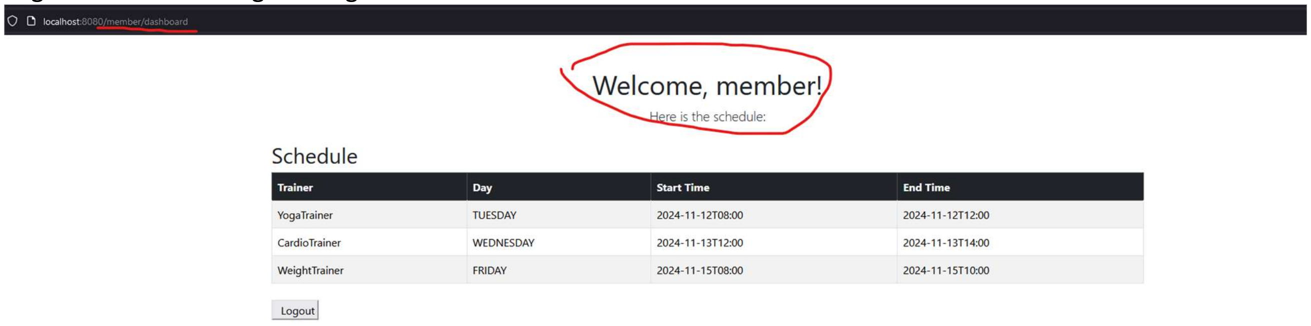
Trainer	Day	Start Time	End Time	Actions
YogaTrainer	TUESDAY	2024-11-12T08:00	2024-11-12T12:00	Edit
CardioTrainer	WEDNESDAY	2024-11-13T12:00	2024-11-13T14:00	Edit
WeightTrainer	FRIDAY	2024-11-15T08:00	2024-11-15T10:00	Edit

Back to Dashboard



```java
@PostMapping("admin/edit-shift/{id}")
public String updateShift(@PathVariable Long id,
                          @RequestParam("trainerId") Long trainerId,
                          @RequestParam("startTime") String startTime,
                          @RequestParam("endTime") String endTime,
                          @RequestParam("dayOfWeek") String dayOfWeek,
                          Model model) {
    try {
        // Parse the start and end times
        LocalDateTime start = LocalDateTime.parse(startTime);
        LocalDateTime end = LocalDateTime.parse(endTime);

        // Get the trainer by ID
        User trainer = userService.getTrainerById(trainerId);
        if (trainer == null) {
            model.addAttribute( attributeName: "error",  attributeValue: "Trainer not found!");
            return "error";
        }

        // Get the existing shift and update its fields
        Shift shift = shiftService.getShiftById(id);
        if (shift == null) {
            model.addAttribute( attributeName: "error",  attributeValue: "Shift not found!");
            return "error";
        }

        // Update shift properties
        shift.setTrainer(trainer);
        shift.setStartTime(start);
        shift.setEndTime(end);
        shift.setDayOfWeek(dayOfWeek);

        // Save the updated shift
        shiftService.saveShift(shift);

        model.addAttribute( attributeName: "message",  attributeValue: "Shift updated successfully!");
        return "redirect:/admin/trainer-shifts";  // Redirect back to the list of shifts
    } catch (Exception e) {
        model.addAttribute( attributeName: "error",  attributeValue: "An unexpected error occurred: " + e.getMessage());
        return "error";  // Show error page if any exception occurs
```

2. Member:
   - Login: Member can log in using credentials.



localhost:8080/member/dashboard

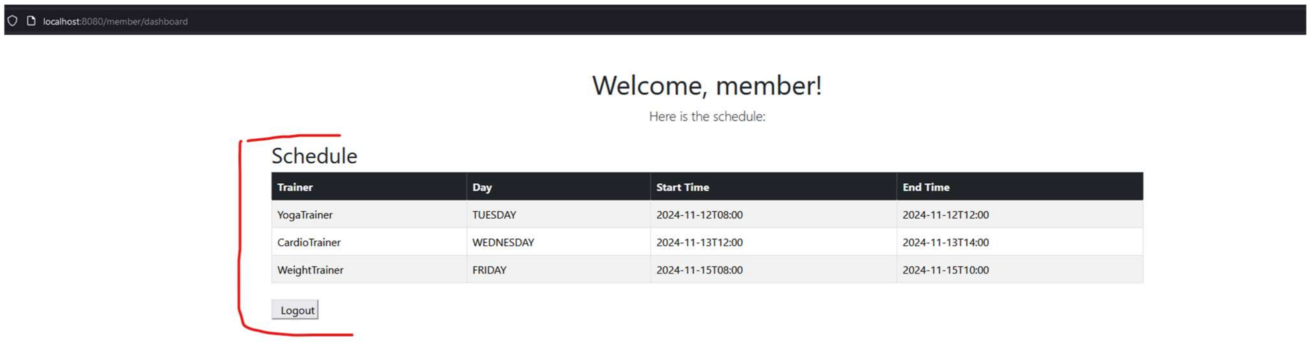
Welcome, member!

Here is the schedule:

Schedule

Trainer	Day	Start Time	End Time
YogaTrainer	TUESDAY	2024-11-12T08:00	2024-11-12T12:00
CardioTrainer	WEDNESDAY	2024-11-13T12:00	2024-11-13T14:00
WeightTrainer	FRIDAY	2024-11-15T08:00	2024-11-15T10:00

   - Schedule: Member can log in to check the schedule.



localhost:8080/member/dashboard

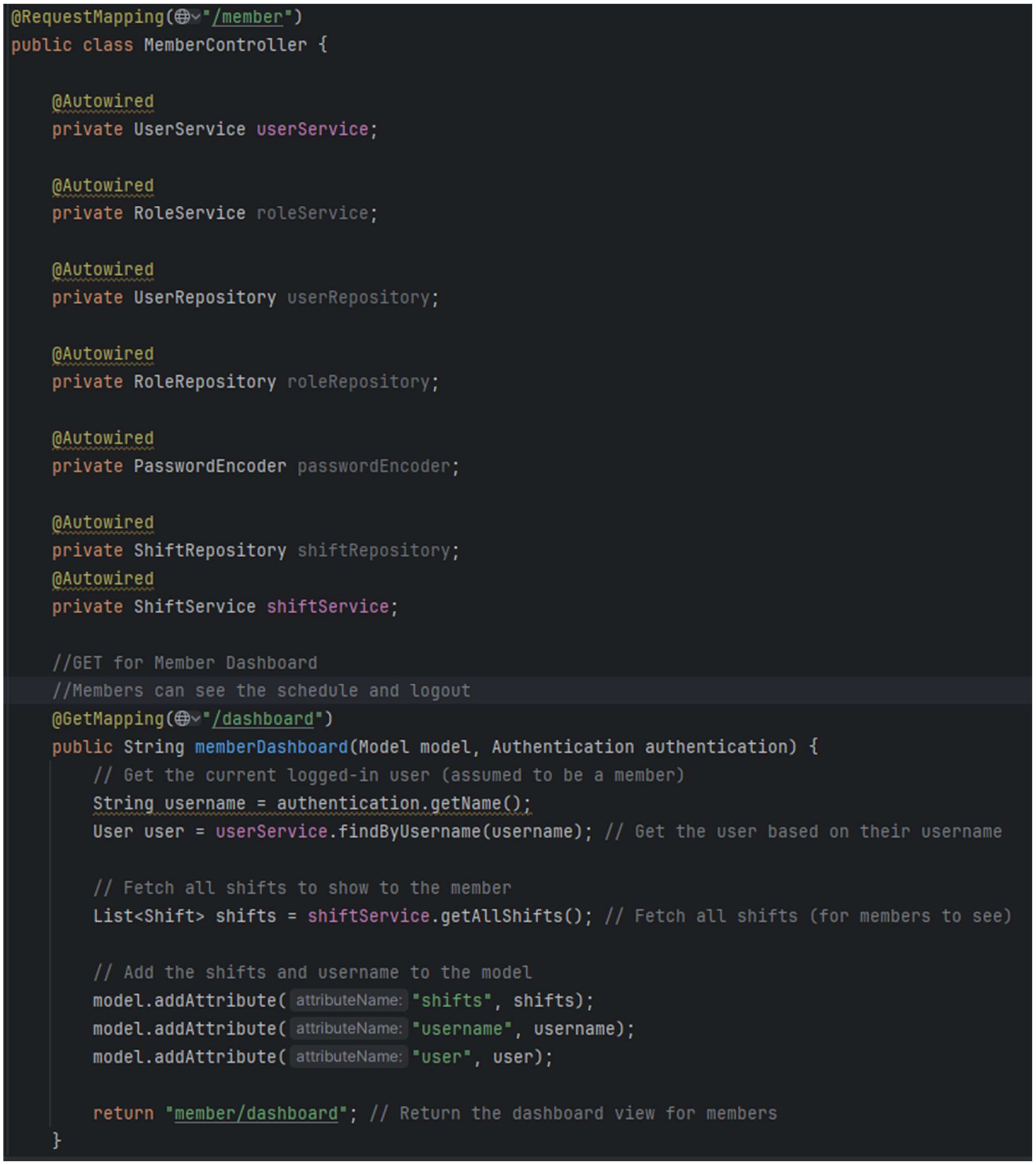
Welcome, member!

Here is the schedule:

Schedule

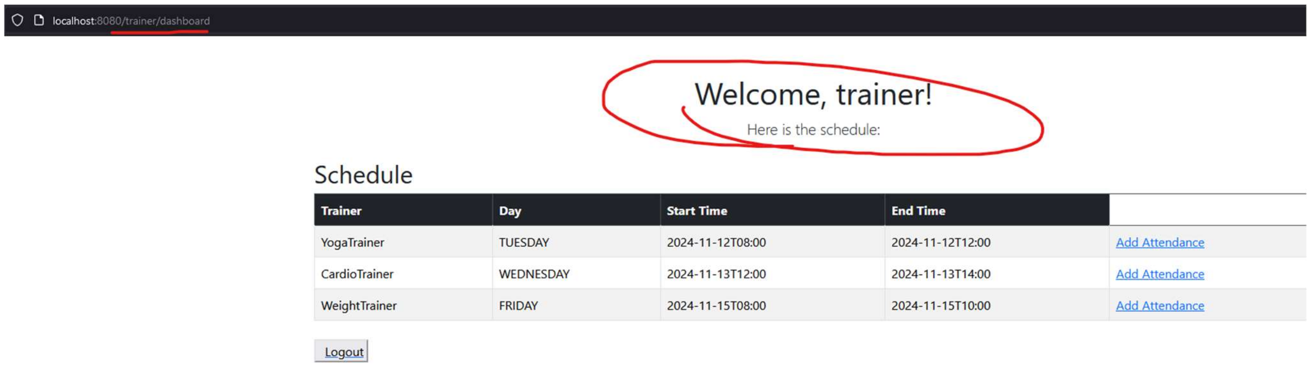
Trainer	Day	Start Time	End Time
YogaTrainer	TUESDAY	2024-11-12T08:00	2024-11-12T12:00
CardioTrainer	WEDNESDAY	2024-11-13T12:00	2024-11-13T14:00
WeightTrainer	FRIDAY	2024-11-15T08:00	2024-11-15T10:00

Logout



```java
@RequestMapping("/member")
public class MemberController {

    @Autowired
    private UserService userService;

    @Autowired
    private RoleService roleService;

    @Autowired
    private UserRepository userRepository;

    @Autowired
    private RoleRepository roleRepository;

    @Autowired
    private PasswordEncoder passwordEncoder;

    @Autowired
    private ShiftRepository shiftRepository;
    @Autowired
    private ShiftService shiftService;

    //GET for Member Dashboard
    //Members can see the schedule and logout
    @GetMapping("/dashboard")
    public String memberDashboard(Model model, Authentication authentication) {
        // Get the current logged-in user (assumed to be a member)
        String username = authentication.getName();
        User user = userService.findByUsername(username); // Get the user based on their username

        // Fetch all shifts to show to the member
        List<Shift> shifts = shiftService.getAllShifts(); // Fetch all shifts (for members to see)

        // Add the shifts and username to the model
        model.addAttribute( attributeName: "shifts", shifts);
        model.addAttribute( attributeName: "username", username);
        model.addAttribute( attributeName: "user", user);

        return "member/dashboard"; // Return the dashboard view for members
    }
```

3. Trainer:
    - Login: The trainer can log in using credentials.


localhost:8080/trainer/dashboard

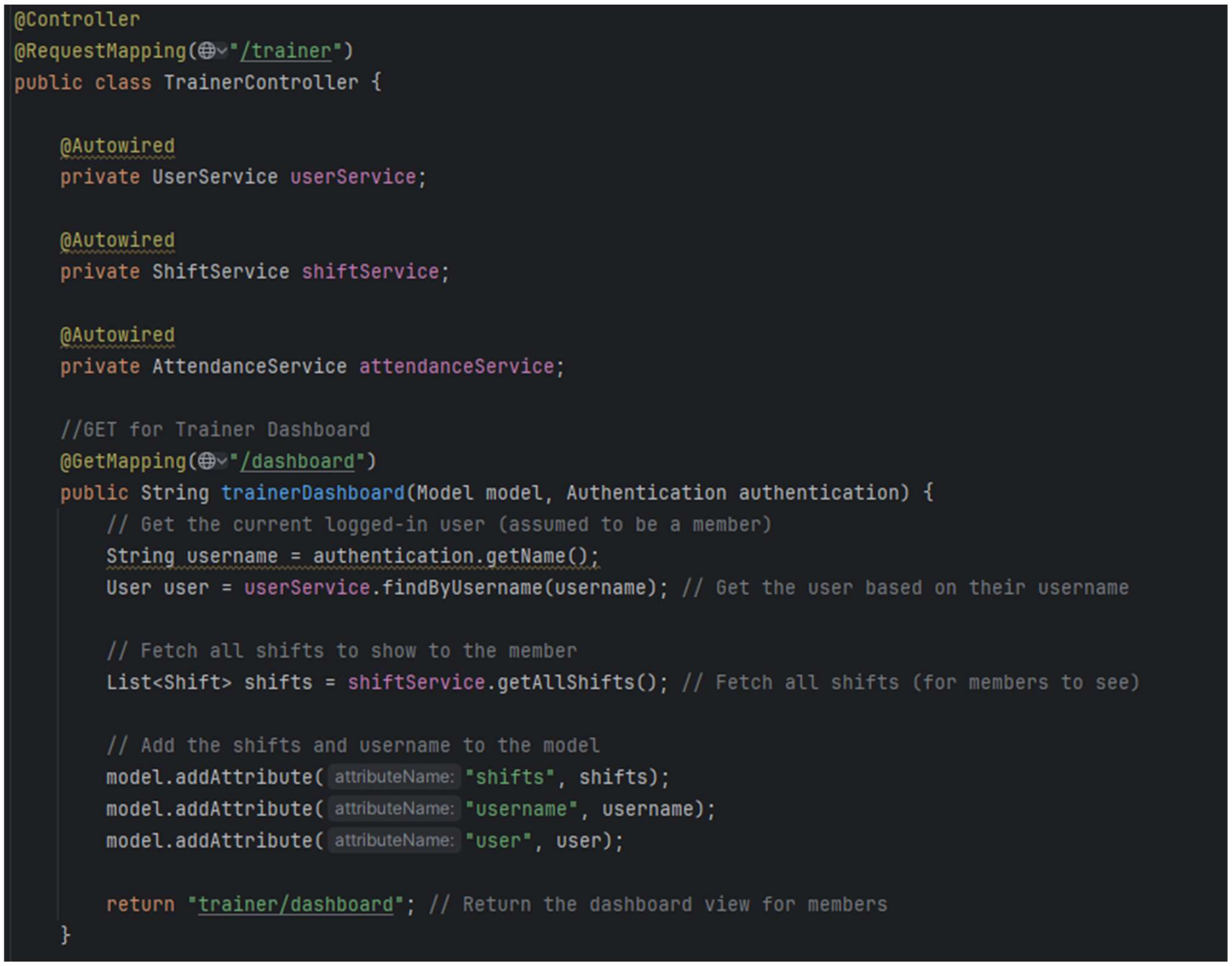
Welcome, trainer!

Here is the schedule:

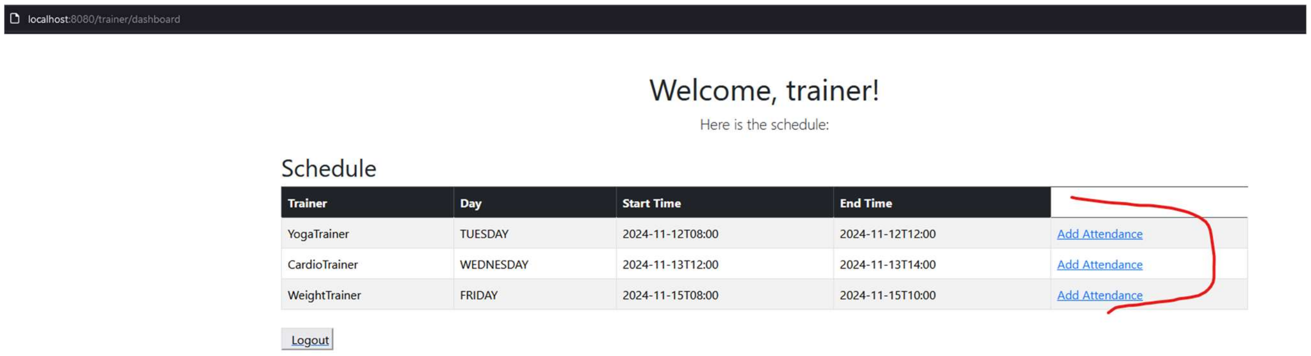
Schedule

Trainer	Day	Start Time	End Time	
YogaTrainer	TUESDAY	2024-11-12T08:00	2024-11-12T12:00	Add Attendance
CardioTrainer	WEDNESDAY	2024-11-13T12:00	2024-11-13T14:00	Add Attendance
WeightTrainer	FRIDAY	2024-11-15T08:00	2024-11-15T10:00	Add Attendance

Logout


```java
@Controller
@RequestMapping("/trainer")
public class TrainerController {

    @Autowired
    private UserService userService;

    @Autowired
    private ShiftService shiftService;

    @Autowired
    private AttendanceService attendanceService;

    //GET for Trainer Dashboard
    @GetMapping("/dashboard")
    public String trainerDashboard(Model model, Authentication authentication) {
        // Get the current logged-in user (assumed to be a member)
        String username = authentication.getName();
        User user = userService.findByUsername(username); // Get the user based on their username

        // Fetch all shifts to show to the member
        List<Shift> shifts = shiftService.getAllShifts(); // Fetch all shifts (for members to see)

        // Add the shifts and username to the model
        model.addAttribute( attributeName: "shifts", shifts);
        model.addAttribute( attributeName: "username", username);
        model.addAttribute( attributeName: "user", user);

        return "trainer/dashboard"; // Return the dashboard view for members
    }
```

- Attendance: The trainer can mark the attendance of the members.


localhost:8080/trainer/dashboard

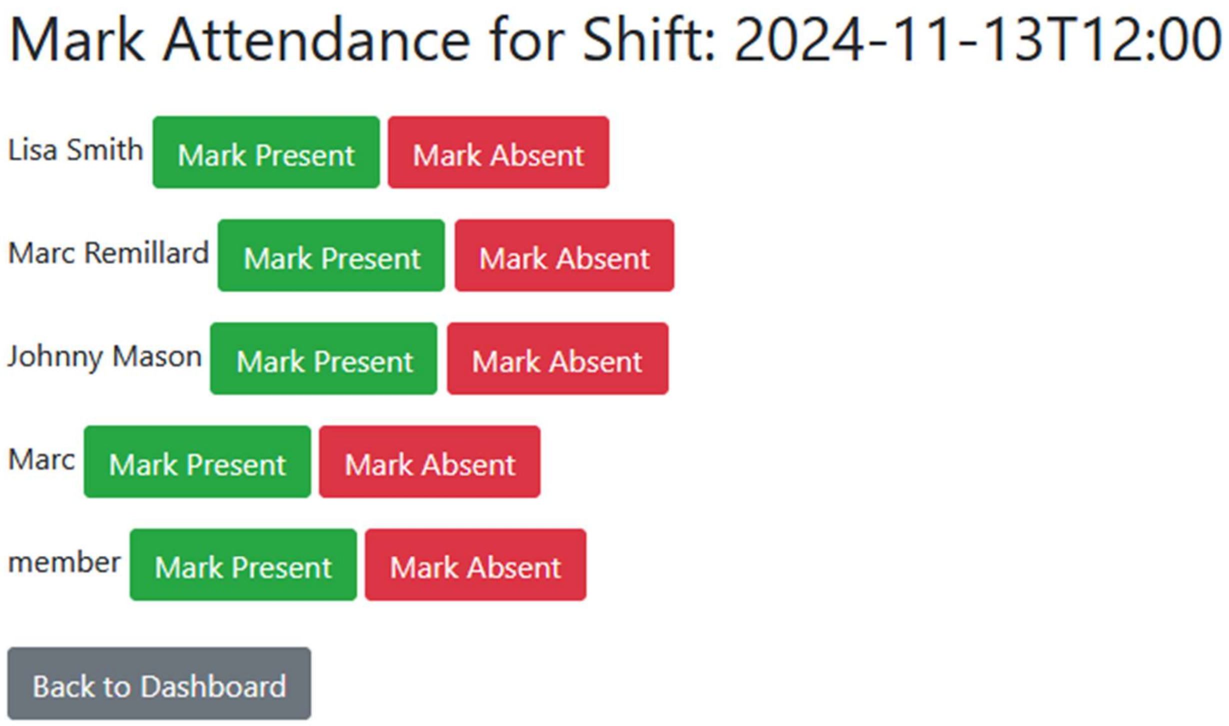
Welcome, trainer!

Here is the schedule:

Schedule

Trainer	Day	Start Time	End Time	
YogaTrainer	TUESDAY	2024-11-12T08:00	2024-11-12T12:00	Add Attendance
CardioTrainer	WEDNESDAY	2024-11-13T12:00	2024-11-13T14:00	Add Attendance
WeightTrainer	FRIDAY	2024-11-15T08:00	2024-11-15T10:00	Add Attendance

Logout



Mark Attendance for Shift: 2024-11-13T12:00

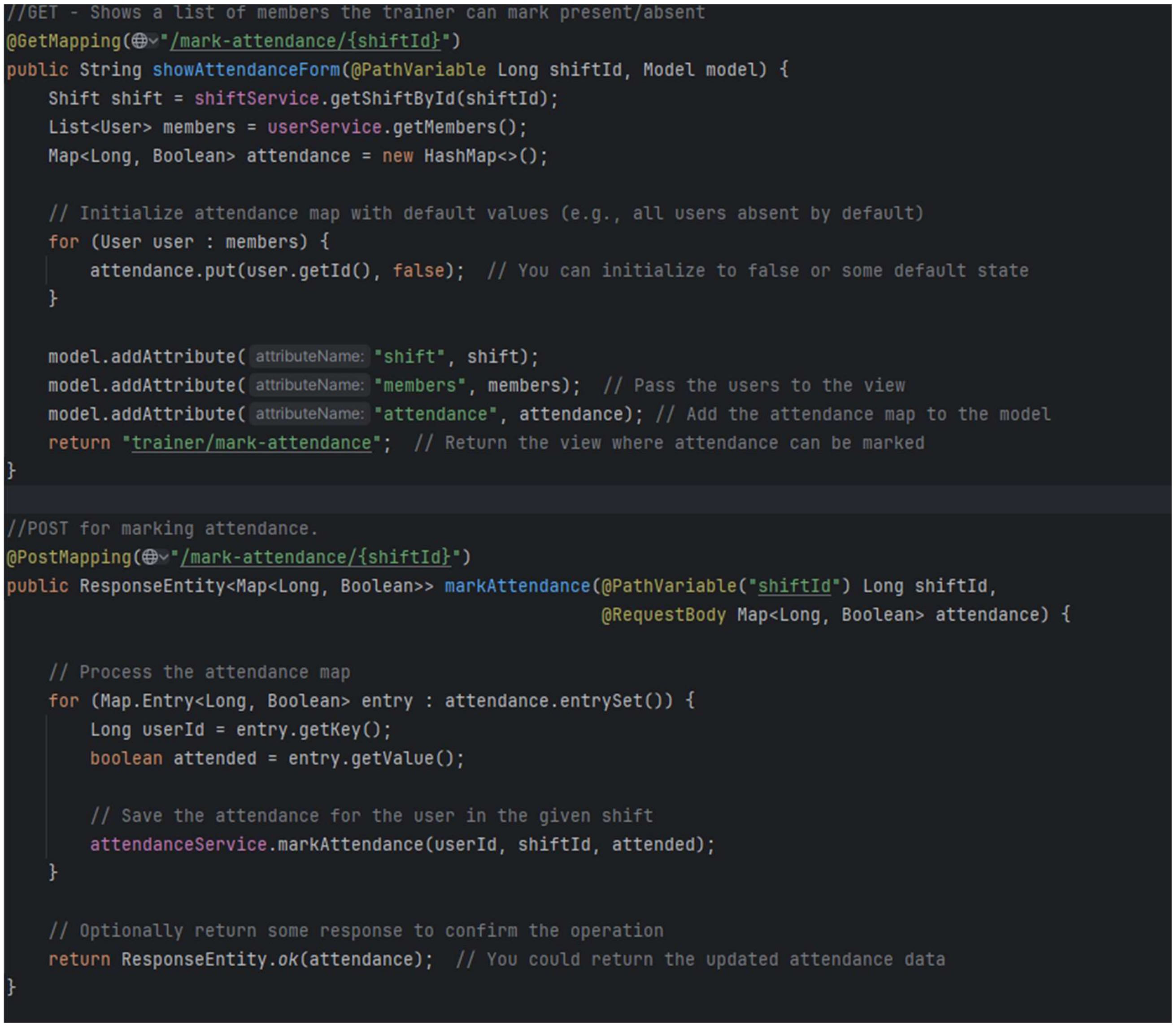
Lisa Smith

Marc Remillard

Johnny Mason

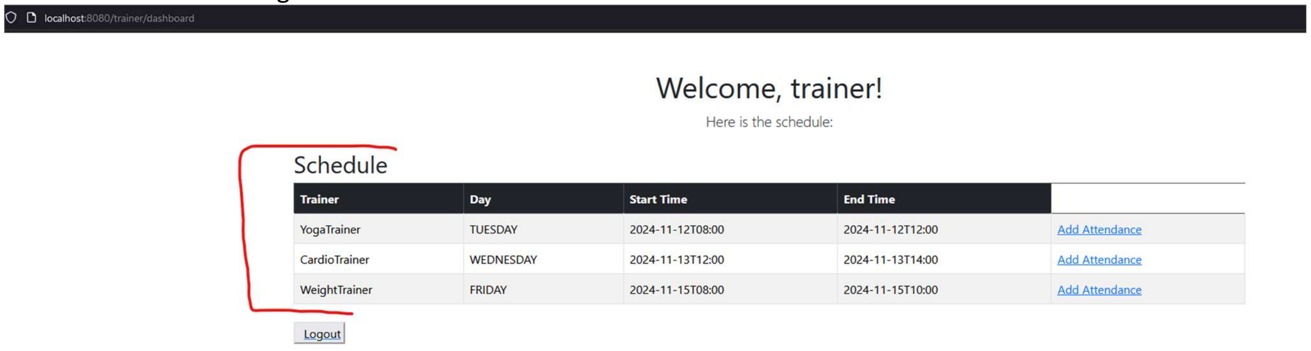
Marc

member



```java
//GET - Shows a list of members the trainer can mark present/absent
@GetMapping("/mark-attendance/{shiftId}")
public String showAttendanceForm(@PathVariable Long shiftId, Model model) {
    Shift shift = shiftService.getShiftById(shiftId);
    List<User> members = userService.getMembers();
    Map<Long, Boolean> attendance = new HashMap<>();

    // Initialize attendance map with default values (e.g., all users absent by default)
    for (User user : members) {
        attendance.put(user.getId(), false);  // You can initialize to false or some default state
    }

    model.addAttribute( attributeName: "shift", shift);
    model.addAttribute( attributeName: "members", members);  // Pass the users to the view
    model.addAttribute( attributeName: "attendance", attendance); // Add the attendance map to the model
    return "trainer/mark-attendance";  // Return the view where attendance can be marked
}

//POST for marking attendance.
@PostMapping("/mark-attendance/{shiftId}")
public ResponseEntity<Map<Long, Boolean>> markAttendance(@PathVariable("shiftId") Long shiftId,
                                                         @RequestBody Map<Long, Boolean> attendance) {

    // Process the attendance map
    for (Map.Entry<Long, Boolean> entry : attendance.entrySet()) {
        Long userId = entry.getKey();
        boolean attended = entry.getValue();

        // Save the attendance for the user in the given shift
        attendanceService.markAttendance(userId, shiftId, attended);
    }

    // Optionally return some response to confirm the operation
    return ResponseEntity.ok(attendance);  // You could return the updated attendance data
}
```

- Schedule: Trainer can log in to check the schedule.



localhost:8080/trainer/dashboard

Welcome, trainer!

Here is the schedule:

Schedule

Trainer	Day	Start Time	End Time	
YogaTrainer	TUESDAY	2024-11-12T08:00	2024-11-12T12:00	Add Attendance
CardioTrainer	WEDNESDAY	2024-11-13T12:00	2024-11-13T14:00	Add Attendance
WeightTrainer	FRIDAY	2024-11-15T08:00	2024-11-15T10:00	Add Attendance

Program 1 – Fitness Club Management System

Criteria	Insufficient (0 pts)	Needs Development (3-5 pts)	Sufficient (7 pts)	Excellent (10 pts)	Mark
Submissions: GitHub Source Code & Screen Recording	• Little to no effort was made or contains too many errors/omissions.	• A reasonable effort was made, but there are multiple areas for improvement.	• A good effort was made, but at least one error or omission exists.	• An extended effort was made, and go beyond the mentioned requirement.	
In-code Documentation & Code Quality	• Little to no effort was made or contains too many errors/omissions.	• A reasonable effort was made, but there are multiple areas for improvement.	• A good effort was made, but at least one error or omission exists.	• An extended effort was made and go beyond expectations. Also demonstrated a strong understanding of the in-code documentation and code quality.	
System Design & Solution: Dynamic Input/Output, Fulfill all the mentioned requirement	• Little to no effort was made or contains too many errors/omissions.	• A reasonable effort was made, but there are multiple areas for improvement.	• A good effort was made, but at least one error or omission exists.	• An extended effort was made and go beyond the mentioned requirement. Also demonstrated a strong understanding of the solution design.	
Java Language/Spring Concepts: Variables, Datatypes, Logic control statements, GET, POST, Forms, Functions and a lot.	• Little to no effort was made or contains too many errors/omissions.	• A reasonable effort was made, but there are multiple areas for improvement.	• A good effort was made, but at least one error or omission exists.	• An extended effort was made and demonstrated a strong understanding of the framework language concepts.	
				Total:	**/40**