МИНОБРНАУКИРОССИИ
ФЕДЕРАЛЬНОЕГОСУДАРСТВЕННОЕБЮДЖЕТНОЕОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ»(ФГБОУВО«ВГУ»)

Факультет компьютерных наук

Кафедра программирования и информационных технологий

Веб-сайты для спортзала "GG Gym "

Курсовой проект

09.03.04 Программная инженерия

Допущено к защите в ГЭК..... 2024

Зав.кафедрой _____ С.Д. Махортов д.ф.-м.н.,профессор __.__.2023

Обучающийся _____ О.Р.Х.Якуб

Руководитель _____ В.С. Тарасов ст. преподаватель

Воронеж 2023

Содержание

Введение

Современное общество сталкивается с постоянными изменениями и вызовами, включая технологические инновации, изменения в поведении потребителей и развитие новых отраслей экономики. В контексте этой динамичной среды фитнес-индустрия занимает особое место, предлагая решения для поддержания здорового образа жизни и физической активности. Однако, для эффективного функционирования и успешного конкурирования на рынке, фитнес-центры должны постоянно совершенствовать свои процессы управления, включая организацию тренировок, расписание занятий и взаимодействие с клиентами;

Цель настоящей курсовой работы заключается в разработке и реализации программного решения для управления тренировочным процессом в фитнес-центре. Это решение предполагает создание инструмента, который позволит эффективно планировать, контролировать и администрировать тренировки, а также управлять персоналом и взаимодействовать с клиентами;

Прежде чем приступить к разработке программного продукта, необходимо провести анализ существующих методов управления тренировочным процессом в фитнес-индустрии. Этот анализ позволит выявить проблемные области и определить требования к программному решению. Кроме того, будет проанализировано текущее состояние фитнес-рынка, его тенденции и особенности, чтобы разработать программу, соответствующую современным требованиям и ожиданиям клиентов;

В ходе работы будет разработана концепция программного решения, включающая в себя функциональные и нефункциональные требования, а также архитектуру системы. Далее будет реализован прототип программы, который будет протестирован на реальных данных и взаимодействии с пользователями. Результаты тестирования и обратная связь пользователей будут использованы для доработки и улучшения программного решения;

Данное исследование является актуальным в контексте современных тенденций в области фитнеса и управления бизнесом. Разработанное программное решение может стать ценным инструментом для фитнес-центров, помогая им оптимизировать свои операции, повысить

удовлетворенность клиентов и улучшить их результаты.

1. Постановка задачи

Для достижения цели разработки программного решения для управления тренировочным процессом в фитнес-центре необходимо решить следующие задачи

- Анализ существующих методов управления тренировочным процессом: Провести обзор существующих подходов к управлению тренировочными процессами в фитнес-индустрии. Изучить основные проблемы и недостатки существующих систем управления;

- Определение требований к программному решению: Выявить основные требования, которые должно удовлетворять программное решение. Определить функциональные и нефункциональные требования к системе;

- Разработка архитектуры программного решения: Разработать общую архитектуру программного продукта, включая выбор технологий, архитектурных шаблонов и инструментов;

- Реализация программного решения на “Java, JavaScript, HTML, CSS” Написать код программы, используя языки программирования Java для бэкэнда и JavaScript, HTML, CSS для фронтэнда. Обеспечить взаимодействие между серверной и клиентской частями приложения;

- Разработка базы данных: Создать и настроить базу данных MySQL для хранения информации о тренировках, тренерах, клиентах и других сущностях, необходимых для функционирования системы.

Решение каждой из этих задач позволит разработать функциональное, надежное и эффективное программное решение для управления тренировочным процессом в фитнес-центре.

2. Анализ предметной области

2.1 Терминология предметной области

В предметной области управления тренировочным процессом в фитнес-центре используются следующие термины:

- **Фитнес-центр**: Специализированное учреждение, предоставляющее услуги по физической активности, тренировкам и занятиям спортом;
- **Тренировка**: Сессия физических упражнений, проводимых с целью улучшения физической формы, развития определенных мышечных групп или достижения конкретных спортивных целей;
- **Тренер**: Специалист по физической подготовке, проводящий индивидуальные или групповые тренировки, разрабатывающий программы тренировок, следящий за выполнением упражнений и корректирующий их;
- **Расписание**: График проведения тренировок в фитнес-центре, включающий в себя информацию о времени начала и окончания занятий, их продолжительности, а также информацию о тренерах, проводящих занятия;
- **Клиент**: Лицо, занимающееся в фитнес-центре, посещающее тренировки и использующее услуги центра для достижения своих спортивных и физических целей;
- **База данных**: Структурированное хранилище данных, используемое для хранения информации о гостях, номерах, бронированиях, услугах и других аспектах управления ;

- **Интерфейс пользователя**: Средство взаимодействия между пользователем и приложением, которое облегчает выполнение определенных задач. В случае приложения для управления , интерфейс пользователя может включать веб-страницы, мобильные приложения, терминалы самообслуживания и другие формы;

- **Авторизация**: Процесс предоставления прав доступа пользователю к определенным ресурсам или функциям системы после успешной аутентификации;

- **API (Application Programming Interface**): Набор правил и инструкций, определяющих способы взаимодействия между различными компонентами программного обеспечения. API позволяет приложениям обмениваться данными и вызывать функции друг друга.

2.2 Средства реализации

Для успешной реализации разработки спорт зал " GG Gym " приложения был задействован целый ряд инструментов и ресурсов. В процессе создания данного приложения использовались следующие технологии, программные средства и библиотеки, которые играли важную роль в достижении целей:

- Java “Spring boot”
- JavaScript
- HTML, CSS
- MySQL

2.3 Обзор средств

Обзор основных средств, используемых в разработке веб-приложений для управления тренировочным процессом в фитнес-центре:

- Java с фреймворком Spring Boot: Java - это популярный язык программирования, широко используемый для создания серверных приложений. Spring Boot - это фреймворк для создания веб-приложений на Java, который облегчает разработку благодаря своей модульной структуре и автоматической настройке. Spring Boot позволяет быстро создавать веб-сервисы и RESTful API;
- JavaScript: Язык программирования, применяемый для создания интерактивных элементов на веб-страницах. JavaScript широко используется для разработки фронтенд-части веб-приложений, обеспечивая динамическое обновление содержимого страницы без перезагрузки. В контексте фитнес-центра, JavaScript может быть использован для создания интерфейса пользователя, включая динамическую загрузку расписания тренировок и взаимодействие с тренерами;
- HTML, CSS: HTML (HyperText Markup Language) и CSS (Cascading Style Sheets) используются для создания структуры и внешнего вида веб-страниц соответственно. HTML определяет содержимое страницы, в то время как CSS применяется для стилизации и оформления элементов. В разработке веб-приложения для управления тренировочным процессом, HTML и CSS используются для создания пользовательского интерфейса с поддержкой интерактивности и привлекательного внешнего вида;
- MySQL: Реляционная система управления базами данных, используемая для хранения, управления и извлечения данных. MySQL часто применяется в веб-разработке для сохранения информации о клиентах, тренерах, расписании тренировок и других сущностях, необходимых для функционирования приложения. В контексте управления тренировочным процессом, MySQL может быть использована для хранения данных о клиентах, их тренировках, расписании занятий и других важных аспектах работы фитнес-центра;

3. Реализация

3.1 диаграмма состояний (Diagram State)

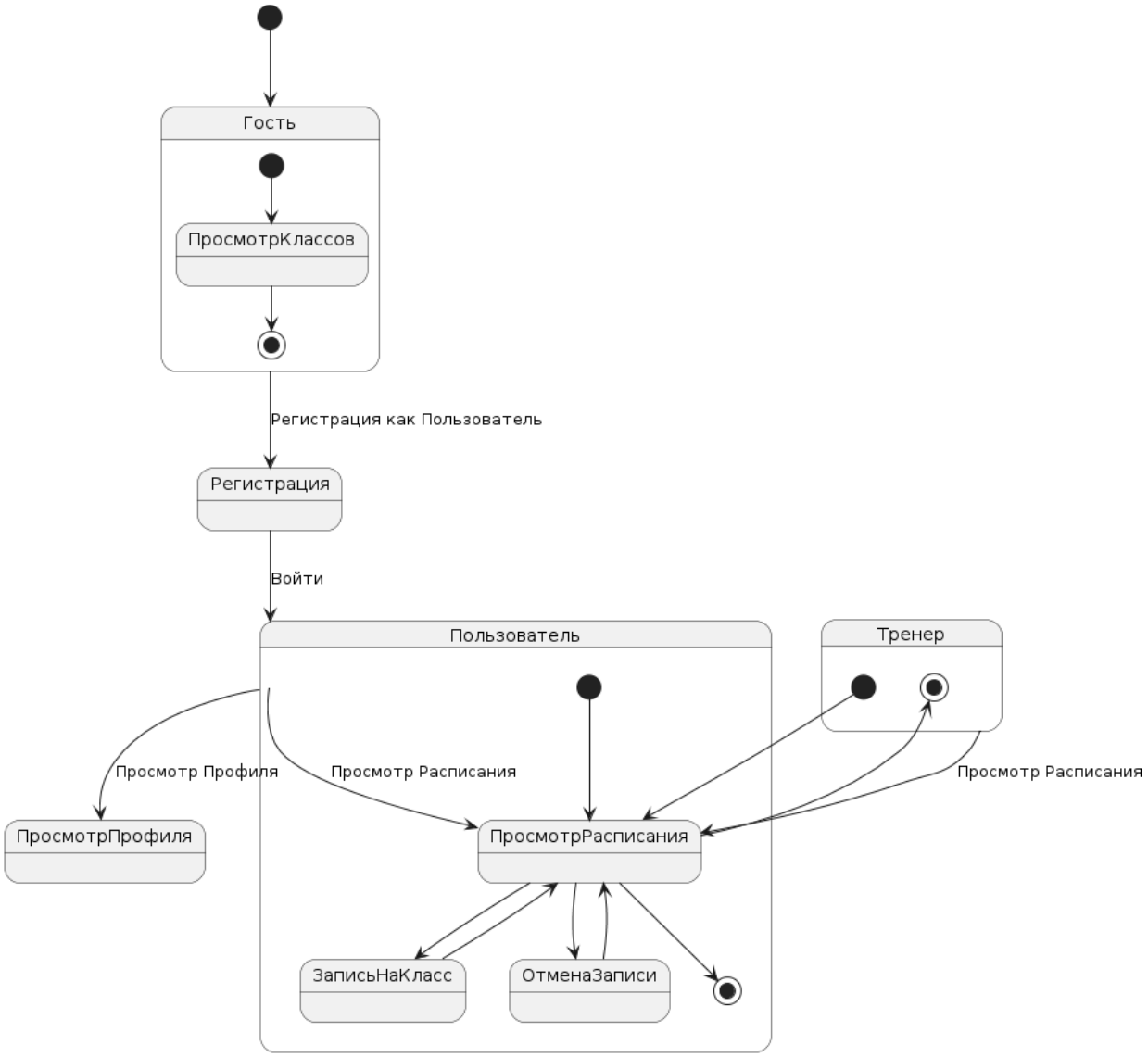

Рис .1 – диаграмма состояний (Diagram State)

Диаграмма состояний моделирует различные состояния для гостя, пользователя и тренера в системе управления тренировочным процессом в фитнес-центре. Вот краткий обзор состояний и переходов:

1. Гость: Начальное состояние, когда пользователь еще не зарегистрирован. Отсюда можно перейти к просмотру классов.
2. Пользователь: Состояние после регистрации или входа в систему. Пользователь может просматривать расписание, записываться на классы и отменять свои записи.
3. Тренер: Состояние тренера, который может просматривать расписание занятий.

3.2 Диаграммы вариантов использования (use case diagrams)

Рис .2- Диаграммы вариантов использования (use case diagrams)

Диаграмма включает актеров и функциональности системы управления спортивным залом.

Вот краткое описание каждого элемента:

1. Admin (Администратор): Может управлять факультетом (тренерами), тренерами, расписанием и членами зала.
2. Trainer (Тренер): Может просматривать расписание и управлять членством.
3. Member (Участник): Может просматривать расписание, искать по расписанию, просматривать информацию о тренере и членстве.

У администратора есть полный доступ ко всем функциям системы, в то время как тренер и член зала имеют ограниченный набор функций, специфичных для их роли. Это хорошо структурирует систему и определяет, какие действия доступны каждому типу пользователя

3.3 Диаграммы классов (Class Diagrams)

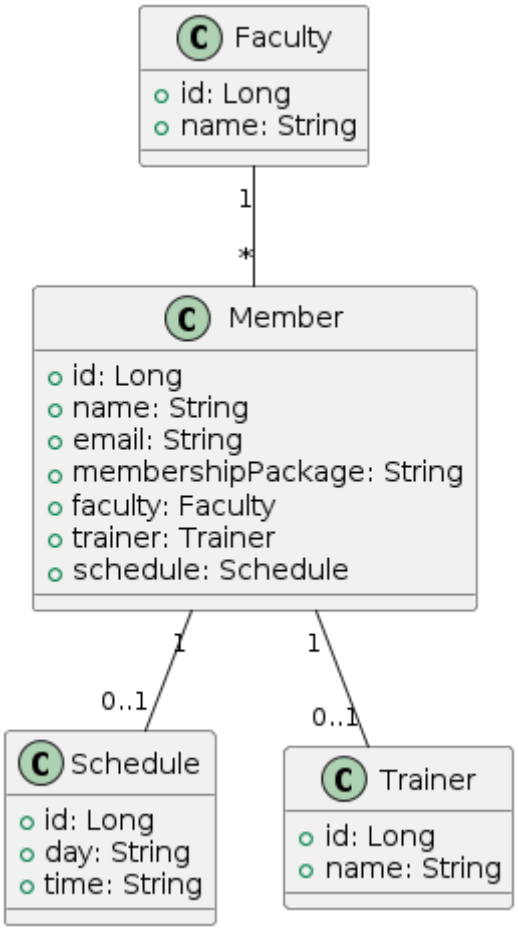

Рис .3- Диаграммы классов (Class Diagrams)

Диаграмма классов моделирует структуру данных для системы управления спортивным залом. Вот краткое описание каждого класса:

1. Faculty (Факультет): Представляет различные факультеты или отделения в спортивном зале. Каждый факультет имеет уникальный идентификатор (id) и

название (name).

2. Member (Участник): Представляет участников зала. Участник имеет уникальный идентификатор (id), имя (name), адрес электронной почты (email), пакет членства (membershipPackage), факультет (faculty), тренера (trainer) и расписание (schedule). Отношение "1" к "0..1" между участником и расписанием означает, что участник может иметь не более одного расписания. Отношение "1" к "0..1" между участником и тренером означает, что участник может иметь не более одного тренера.
3. Schedule (Расписание): Представляет расписание тренировок. Каждое расписание имеет уникальный идентификатор (id), день недели (day) и время (time).
4. Trainer (Тренер): Представляет тренеров, работающих в спортивном зале. Каждый тренер имеет уникальный идентификатор (id) и имя (name).

Отношения между классами показывают, как они связаны друг с другом:

- У каждого факультета может быть множество участников, но у каждого участника может быть только один факультет.
- У каждого участника может быть одно или ни одного расписания.
- У каждого участника может быть один или ни одного тренера.

Эта структура данных помогает организовать информацию о факультетах, участниках, расписаниях и тренерах, что позволяет системе эффективно управлять спортивным залом.

3.4 Реализация интерфейса

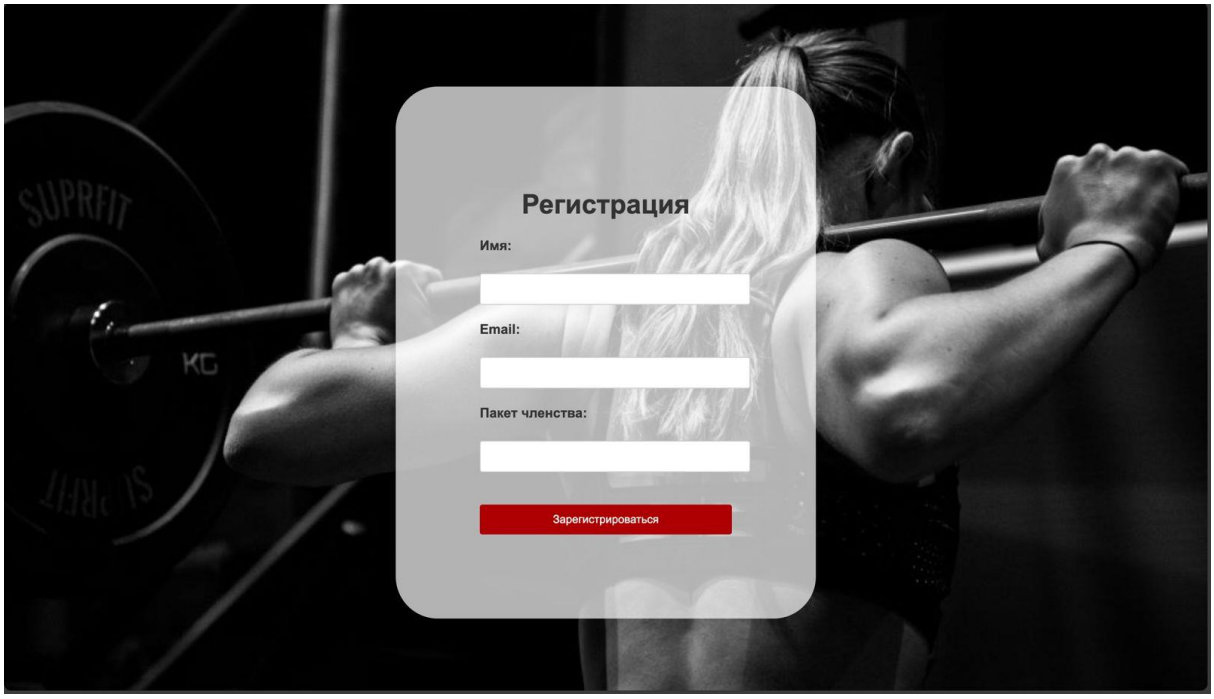

Рис .5 – Registration

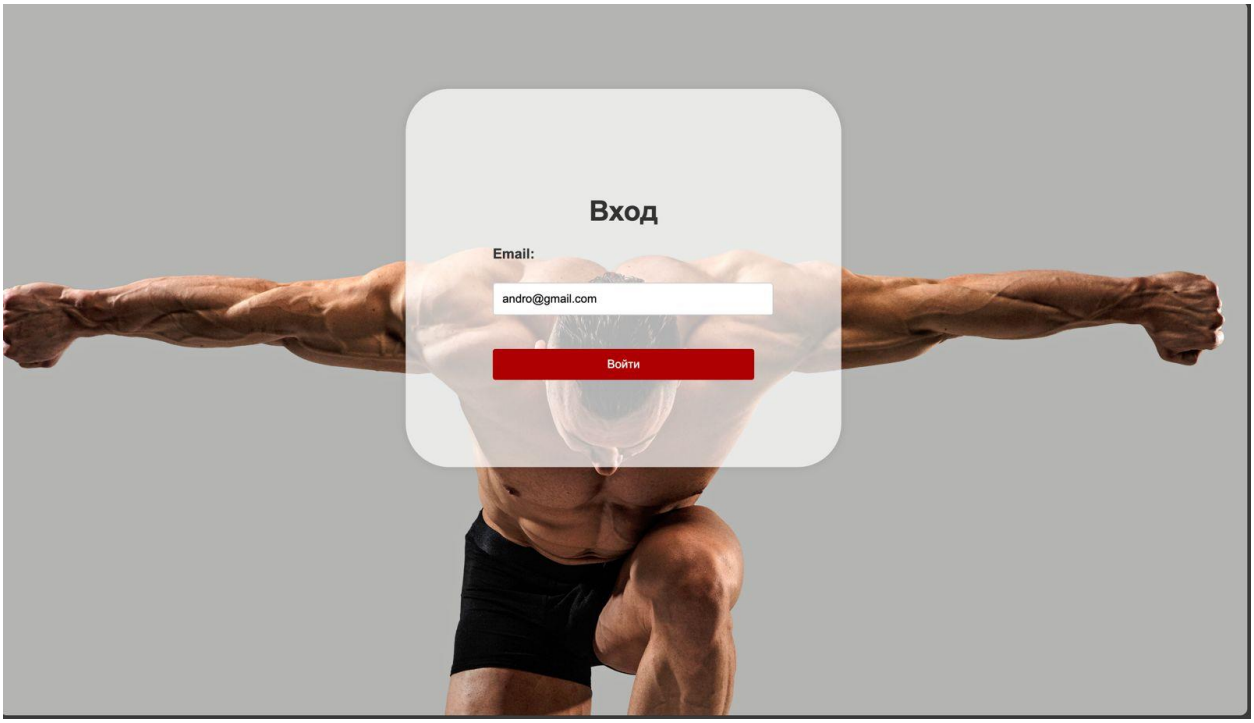

Рис .6 – Login

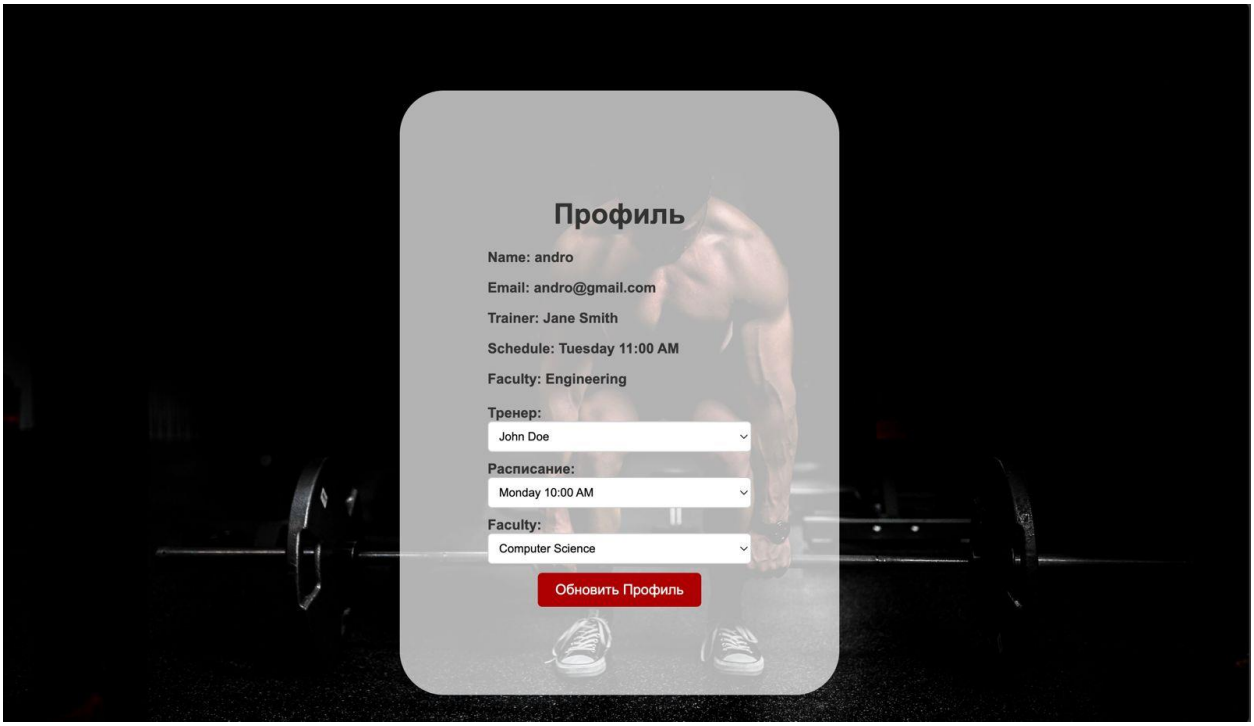

Рис .6 – Profile

Рис .7-Admin

Заключение

В заключение, разработка программного решения для управления тренировочным процессом в фитнес-центре "GG Gym" представляет собой важный шаг в современной фитнес-индустрии. Наше решение не только упростит процессы управления и повысит удовлетворенность клиентов, но и поможет "GG Gym" оставаться конкурентоспособным в быстро меняющейся среде, Эффективное управление тренировочным процессом играет ключевую роль в успехе фитнес-центра. Оно включает в себя планирование тренировок, контроль за выполнением упражнений, назначение тренеров, взаимодействие с клиентами и многое другое. Наше программное решение позволит "GG Gym" автоматизировать и оптимизировать эти процессы, что приведет к улучшению качества обслуживания и удовлетворенности клиентов, Кроме того, наше решение предлагает инновационные функции, такие как персонализированные тренировочные программы, отслеживание прогресса клиентов и удобный интерфейс взаимодействия с тренерами. Это поможет клиентам "GG Gym" достигать своих фитнес-целей более эффективно и мотивированно.
Разработка программного решения также сопровождается обучением персонала и внедрением системы. Мы предоставим всестороннюю поддержку "GG Gym" на всех этапах внедрения, чтобы гарантировать успешное использование нашего решения и достижение поставленных целей, Таким образом, наше программное решение является не просто инструментом управления тренировочным процессом, но и ключевым элементом в стратегии развития "GG Gym". Мы уверены, что его внедрение принесет значительные выгоды для фитнес-центра, его клиентов и всей команды.

Список используемых источников

В данной курсовой работе использовались следующие источники

- Oracle Corporation. “MySQL Documentation” https://dev.mysql.com/doc/
- Spring Boot. “Spring Boot Reference Guide” https://docs.spring.io/spring-boot/docs/current/reference/html/
- Mozilla Developer Network. “JavaScript Documentation” https://developer.mozilla.org/en-US/docs/Web/JavaScript
- Mozilla Developer Network. “HTML Documentation” https://developer.mozilla.org/en-US/docs/Web/HTML
- Mozilla Developer Network. “CSS Documentation” https://developer.mozilla.org/en-US/docs/Web/CSS

Приложение

```
package com.example.demo.config;

import com.example.demo.repositories.MemberRepository;
import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
```

```
import org.springframework.security.core.userdetails.UserDetailsService;
import
org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

import java.util.ArrayList;

@Service
public class CustomUserDetailsService  implements UserDetailsService {

    private MemberRepository MemberRepository;

    @Autowired
    public CustomUserDetailsService(MemberRepository  userRepository) {
        this.MemberRepository = userRepository;
    }

    @Override
    public UserDetails loadUserByUsername(String email) throws
UsernameNotFoundException {
        com.example.demo.model.Member member =
MemberRepository.findByEmail(email)
                .orElseThrow(() -> new UsernameNotFoundException("User not
found with email: " + email));

        return new User(member.getEmail(), member.getEmail(), new
ArrayList<>());
    }



//    private Collection<GrantedAuthority> mapRolesToAuthorities(List<Role>
roles) {
//        return roles.stream().map(role -> new
SimpleGrantedAuthority(role.getName())).collect(Collectors.toList());
//    }
}


package com.example.demo.config;
import org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationTok
en;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import
org.springframework.security.web.authentication.WebAuthenticationDetailsSourc
e;
import org.springframework.util.StringUtils;
import org.springframework.web.filter.OncePerRequestFilter;
import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
```

```
public class JWTAuthenticationFilter extends OncePerRequestFilter {

    @Autowired
    private JWTGenerator tokenGenerator;
    @Autowired
    private CustomUserDetailsService customUserDetailsService;


//    @Override
//    protected void doFilterInternal(HttpServletRequest request,
//                                    HttpServletResponse response,
//                                    FilterChain filterChain) throws
ServletException, IOException {
//        String token = getJWTFromRequest(request);
//        if(StringUtils.hasText(token) &&
tokenGenerator.validateToken(token)) {
//            String username = tokenGenerator.getUsernameFromJWT(token);
//
//            UserDetails userDetails =
customUserDetailsService.loadUserByUsername(username);
//            UsernamePasswordAuthenticationToken authenticationToken = new
UsernamePasswordAuthenticationToken(userDetails, null,
//                    userDetails.getAuthorities());
//            authenticationToken.setDetails(new
WebAuthenticationDetailsSource().buildDetails(request));
//
SecurityContextHolder.getContext().setAuthentication(authenticationToken);
//        }
//        filterChain.doFilter(request, response);
//    }

    private String getJWTFromRequest(HttpServletRequest request) {
        String bearerToken = request.getHeader("Authorization");
        if(StringUtils.hasText(bearerToken) && bearerToken.startsWith("Bearer
")) {
            return bearerToken.substring(7, bearerToken.length());
        }
        return null;
    }

    // Ensure JWT is parsed and user is authenticated for each request that
requires it
    @Override
    protected void doFilterInternal(HttpServletRequest request,
HttpServletResponse response, FilterChain filterChain) throws
ServletException, IOException {
        String token = getJWTFromRequest(request);
        if (StringUtils.hasText(token) &&
tokenGenerator.validateToken(token)) {
            String username = tokenGenerator.getUsernameFromJWT(token);
            UserDetails userDetails =
customUserDetailsService.loadUserByUsername(username);
            UsernamePasswordAuthenticationToken authentication = new
UsernamePasswordAuthenticationToken(userDetails, null,
userDetails.getAuthorities());
            authentication.setDetails(new
WebAuthenticationDetailsSource().buildDetails(request));
```

```
SecurityContextHolder.getContext().setAuthentication(authentication);
        }
        filterChain.doFilter(request, response);
    }



}
```

```
package com.example.demo.config;

import org.springframework.security.core.AuthenticationException;
import org.springframework.security.web.AuthenticationEntryPoint;
import org.springframework.stereotype.Component;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@Component
public class JwtAuthEntryPoint implements AuthenticationEntryPoint {
    @Override
    public void commence(HttpServletRequest request, HttpServletResponse
response, AuthenticationException authException) throws IOException,
ServletException {
        response.sendError(HttpServletResponse.SC_UNAUTHORIZED,
authException.getMessage());
    }
}
```

```
package com.example.demo.config;

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import io.jsonwebtoken.security.Keys;
import
org.springframework.security.authentication.AuthenticationCredentialsNotFound
Exception;
import org.springframework.security.core.Authentication;
import org.springframework.stereotype.Component;

import java.security.Key;
import java.util.Date;

@Component
public class JWTGenerator {
    //private static final KeyPair keyPair =
Keys.keyPairFor(SignatureAlgorithm.RS256);
    private static final Key key =
Keys.secretKeyFor(SignatureAlgorithm.HS512);
    private static final long EXPIRATION_TIME = 86400000; // 24 hours
```

```
    Date now = new Date();

//    public String generateToken(Authentication authentication) {
//        String username = authentication.getName();
//        Date currentDate = new Date();
//        Date expireDate = new Date(now.getTime() +
SecurityConstants.JWT_EXPIRATION);
//
//        String token = Jwts.builder()
//                .setSubject(username)
//                .setIssuedAt(new Date())
//                .setExpiration(expireDate)
//                .signWith(key, SignatureAlgorithm.HS512)
//                .compact();
//        System.out.println("New token :");
//        System.out.println(token);
//        return token;
//    }

    public String generateToken(String email) {
        Date now = new Date();
        Date expiryDate = new Date(now.getTime() + EXPIRATION_TIME);

        return Jwts.builder()
                .setSubject(email)
                .setIssuedAt(now)
                .setExpiration(expiryDate)
                .signWith(key, SignatureAlgorithm.HS512)
                .compact();
    }
    public String getUsernameFromJWT(String token){
        Claims claims = Jwts.parser()
                .setSigningKey(key)
                .build()
                .parseClaimsJws(token)
                .getBody();
        return claims.getSubject();
    }

    public boolean validateToken(String token) {
        try {
            Jwts.parser()
                    .setSigningKey(key)
                    .build()
                    .parseClaimsJws(token);
            return true;
        } catch (Exception ex) {
            throw new AuthenticationCredentialsNotFoundException("JWT was
exprired or incorrect",ex.fillInStackTrace());
        }
    }

}
```

```
package com.example.demo.config;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationManager;
import
org.springframework.security.config.annotation.authentication.configuration.A
uthenticationConfiguration;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSec
urity;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import
org.springframework.security.web.authentication.UsernamePasswordAuthenticatio
nFilter;

@Configuration
@EnableWebSecurity
public class SecurityConfig {

    private JwtAuthEntryPoint authEntryPoint;
    private CustomUserDetailsService userDetailsService;
    @Autowired
    public SecurityConfig(CustomUserDetailsService userDetailsService,
JwtAuthEntryPoint authEntryPoint) {
        this.userDetailsService = userDetailsService;
        this.authEntryPoint = authEntryPoint;
    }


//    public SecurityFilterChain filterChain(HttpSecurity http) throws
Exception {
//        http
//                .cors().and()
//                .csrf().disable()
//                .exceptionHandling()
//                .authenticationEntryPoint(authEntryPoint)
//                .and()
//                .sessionManagement()
//                .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
//                .and()
//                .authorizeRequests()
////                  .antMatchers("/specialties/**").permitAll()
////                  .antMatchers("/images/**").permitAll()
////                  .antMatchers("/appointments/**").permitAll()
////                  .antMatchers("/clinics/search/**").permitAll()// Allow
public access to search clinics
////                  .antMatchers("/doctors/**").permitAll() // Allow public
access to fetch doctors
////                  .antMatchers("/api/auth/**").permitAll()
////                  .antMatchers("/swagger-ui/**", "/v3/api-
```

```
docs/**").permitAll()
//
//                .anyRequest().permitAll()
//                .and()
//                .httpBasic();
//        http.addFilterBefore(jwtAuthenticationFilter(),
UsernamePasswordAuthenticationFilter.class);
//        return http.build();
//    }

    // Configure security to ensure JWT is used where needed
    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws
Exception {
        http.cors().and().csrf().disable()

.exceptionHandling().authenticationEntryPoint(authEntryPoint).and()

.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS).a
nd()
                .authorizeRequests()

.antMatchers("/","/login.html","/profile.html","/register.html",
"/api/members/login",
"/api/members/register","/api/members/profile").permitAll() // Ensure these
paths are publicly accessible
                .anyRequest().permitAll()
                .and()
                .addFilterBefore(jwtAuthenticationFilter(),
UsernamePasswordAuthenticationFilter.class);
        return http.build();
    }


    @Bean
    public AuthenticationManager authenticationManager(
            AuthenticationConfiguration authenticationConfiguration) throws
Exception {
        return authenticationConfiguration.getAuthenticationManager();
    }

    @Bean
    PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Bean
    public  JWTAuthenticationFilter jwtAuthenticationFilter() {
        return new JWTAuthenticationFilter();
    }
}



package com.example.demo.config;
```

```
public class SecurityConstants {
    public static final long JWT_EXPIRATION = 7000000;
}
```

```
package com.example.demo.config;

import org.modelmapper.ModelMapper;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import
org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class WebConfig implements WebMvcConfigurer {

    @Bean
    public ModelMapper modelMapper() {
        return new ModelMapper();
    }


    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/**")
                .allowedOrigins("http://localhost:3000")
                .allowedMethods("GET", "POST", "PUT", "DELETE", "OPTIONS")
                .allowedHeaders("*")
                .allowCredentials(true);
    }

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        // Map the URL path "/images/**" to the physical directory path
"file:///path_to_user-storage/"

registry.addResourceHandler("/images/**").addResourceLocations("f/Users/lutfi
/Documents/sem6/TP-HealHubProject/Program/BackEndHealHubV3/user-storage");
    }
}
```

```
package com.example.demo.controller;

import com.example.demo.model.Faculty;
import com.example.demo.services.FacultyService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
```

```
@RequestMapping("/api/faculties")
public class FacultyController {

    private final FacultyService facultyService;

    @Autowired
    public FacultyController(FacultyService facultyService) {
        this.facultyService = facultyService;
    }

    @GetMapping("/all")
    public ResponseEntity<List<Faculty>> getAllFaculties() {
        List<Faculty> faculties = facultyService.getAllFaculties();
        return ResponseEntity.ok(faculties);
    }

    @PostMapping("/")
    public ResponseEntity<Faculty> createOrUpdateFaculty(@RequestBody Faculty
faculty) {
        Faculty savedFaculty = facultyService.saveOrUpdate(faculty);
        return ResponseEntity.ok(savedFaculty);
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteFaculty(@PathVariable Long id) {
        facultyService.deleteFaculty(id);  // Ensure the service method
correctly handles the deletion.
        return ResponseEntity.noContent().build();
    }
    @PutMapping("/{id}")
    public ResponseEntity<Faculty> updateFaculty(@PathVariable Long id,
@RequestBody Faculty facultyDetails) {
        return facultyService.getFacultyById(id)
                .map(faculty -> {
                    faculty.setName(facultyDetails.getName());
                    // Update other fields as needed
                    Faculty updatedFaculty =
facultyService.saveOrUpdate(faculty);
                    return ResponseEntity.ok(updatedFaculty);
                })
                .orElseGet(() -> ResponseEntity.notFound().build());
    }

}
```

```
package com.example.demo.controller;

import com.example.demo.config.JWTGenerator;
import com.example.demo.dto.MemberDTO;
import com.example.demo.model.Faculty;
import com.example.demo.model.Member;
import com.example.demo.model.Schedule;
import com.example.demo.model.Trainer;
import com.example.demo.services.*;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.security.Principal;
import java.util.Collections;
import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("/api/members")
public class MemberController {

    private final MemberServiceImpl memberService;
    private final ProfileService profileService;

    private final TrainerService trainerService;

    private final ScheduleService scheduleService;

    private final FacultyService facultyService;

    private final JWTGenerator tokenGenerator;

    @Autowired
    public MemberController(MemberServiceImpl memberService, ProfileService
profileService, TrainerService trainerService, ScheduleService
scheduleService, FacultyService facultyService, JWTGenerator tokenGenerator)
{
        this.memberService = memberService;
        this.profileService = profileService;
        this.trainerService = trainerService;
        this.scheduleService = scheduleService;
        this.facultyService = facultyService;
        this.tokenGenerator = tokenGenerator;
    }

    @GetMapping
    public ResponseEntity<List<Member>> getAllMembers() {
        List<Member> members = memberService.getAllMembers();
        return ResponseEntity.ok(members);
    }

    @GetMapping("/{id}")
    public ResponseEntity<Member> getMemberById(@PathVariable Long id) {
        Optional<Member> member = memberService.getMemberById(id);
        return member.map(ResponseEntity::ok).orElseGet(() ->
ResponseEntity.notFound().build());
    }

    @PostMapping
    public ResponseEntity<Member> createMember(@RequestBody MemberDTO
memberDTO) {
        Member member = new Member();
        member.setName(memberDTO.getName());
        member.setEmail(memberDTO.getEmail());
```

```
        member.setMembershipPackage(memberDTO.getMembershipPackage());

        Member savedMember = memberService.saveMember(member);
        return new ResponseEntity<>(savedMember, HttpStatus.CREATED);
    }

    @PutMapping("/{id}")
    public ResponseEntity<Member> updateMember(@PathVariable Long id,
@RequestBody MemberDTO memberDTO) {
        Optional<Member> optionalMember = memberService.getMemberById(id);
        if (optionalMember.isPresent()) {
            Member member = optionalMember.get();
            member.setName(memberDTO.getName());
            member.setEmail(memberDTO.getEmail());
            member.setMembershipPackage(memberDTO.getMembershipPackage());

            Member updatedMember = memberService.saveMember(member);
            return ResponseEntity.ok(updatedMember);
        } else {
            return ResponseEntity.notFound().build();
        }
    }


    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteMember(@PathVariable Long id) {
        memberService.deleteMember(id);
        return ResponseEntity.noContent().build();
    }

    @PostMapping("/register")
    public ResponseEntity<Member> register(@RequestBody MemberDTO memberDTO)
{
        Member member = new Member();
        member.setName(memberDTO.getName());
        member.setEmail(memberDTO.getEmail());
        member.setMembershipPackage(memberDTO.getMembershipPackage());

        Member savedMember = memberService.saveMember(member);
        return new ResponseEntity<>(savedMember, HttpStatus.CREATED);
    }


    @PostMapping("/login")
    public ResponseEntity<?> login(@RequestBody MemberDTO memberDTO) {
        Optional<Member> member = memberService.login(memberDTO.getEmail());
        if (member.isPresent()) {
            String token =
tokenGenerator.generateToken(member.get().getEmail()); // Use the updated
method
            return ResponseEntity.ok(Collections.singletonMap("token",
token));
        } else {
            return ResponseEntity.status(HttpStatus.UNAUTHORIZED).build();
        }
    }
```

```
//    @GetMapping("/profile/{email}")
//    public ResponseEntity<Member> getProfile(@PathVariable String email) {
//        Optional<Member> member = profileService.getMemberProfile(email);
//        return member.map(ResponseEntity::ok)
//                .orElse(ResponseEntity.notFound().build());
//    }
//
//    @PutMapping("/profile/{email}")
//    public ResponseEntity<Member> updateProfile(@PathVariable String email,
@RequestBody MemberDTO memberDTO) {
//        Optional<Member> optionalMember = memberService.login(email);
//        if (optionalMember.isPresent()) {
//            Member member = optionalMember.get();
//            member.setName(memberDTO.getName());
//            member.setEmail(memberDTO.getEmail());
//            member.setMembershipPackage(memberDTO.getMembershipPackage());
//            // Set associations with Schedule, Faculty, Trainer here
//
//            Member updatedMember = memberService.saveMember(member);
//            return ResponseEntity.ok(updatedMember);
//        } else {
//            return ResponseEntity.notFound().build();
//        }
//    }

    @GetMapping("/profile/{email}")
    public ResponseEntity<Member> getProfile(@PathVariable String email) {
        Optional<Member> member = memberService.getMemberByEmail(email);
        return member.map(ResponseEntity::ok)
                .orElse(ResponseEntity.notFound().build());
    }

    @PutMapping("/profile/{email}")
    public ResponseEntity<Member> updateProfile(@PathVariable String email,
@RequestBody MemberDTO memberDTO) {
        Optional<Member> optionalMember =
memberService.getMemberByEmail(email);
        if (optionalMember.isPresent()) {
            Member member = optionalMember.get();
            member.setName(memberDTO.getName());
            member.setEmail(memberDTO.getEmail());
            member.setMembershipPackage(memberDTO.getMembershipPackage());
            // Set associations with Schedule, Faculty, Trainer here

            Member updatedMember = memberService.saveMember(member);
            return ResponseEntity.ok(updatedMember);
        } else {
            return ResponseEntity.notFound().build();
        }
    }

//    @GetMapping("/profile")
//    public ResponseEntity<MemberDTO> getProfile(Principal principal) {
//        String email = principal.getName(); // Email is the username in
this context
//        Optional<Member> member = memberService.getMemberByEmail(email);
```

```
//        return member.map(value -> ResponseEntity.ok(new
MemberDTO(value.getName(), value.getEmail(), value.getMembershipPackage())))
//                .orElseGet(() -> ResponseEntity.notFound().build());
//    }

    @PutMapping("/updateProfile")
    public ResponseEntity<?> updateProfile(Principal principal, @RequestBody
MemberDTO memberDTO) {
        Optional<Member> memberOpt =
memberService.getMemberByEmail(principal.getName());
        if (!memberOpt.isPresent()) {
            return ResponseEntity.notFound().build();
        }

        Member member = memberOpt.get();

        // Check if the trainerId is not null and then set the new trainer
        if (memberDTO.getTrainerId() != null) {
            Trainer trainer =
trainerService.getTrainerById(memberDTO.getTrainerId())
                    .orElse(null); // handle case where trainer might not
exist
            member.setTrainer(trainer);
        }

        // Check if the scheduleId is not null and then set the new schedule
        if (memberDTO.getScheduleId() != null) {
            Schedule schedule =
scheduleService.getScheduleById(memberDTO.getScheduleId())
                    .orElse(null); // handle case where schedule might not
exist
            member.setSchedule(schedule);
        }

        // Check if the facultyId is not null and then set the new faculty
        if (memberDTO.getFacultyId() != null) {
            Faculty faculty =
facultyService.getFacultyById(memberDTO.getFacultyId())
                    .orElse(null); // handle case where faculty might not
exist
            member.setFaculty(faculty);
        }

        // Save the updated member
        Member updatedMember = memberService.saveMember(member);
        return ResponseEntity.ok(updatedMember);
    }
    @GetMapping("/profile")
    public ResponseEntity<MemberDTO> getProfile(Principal principal) {
        String email = principal.getName(); // Email is the username in this
context
        Optional<Member> member = memberService.getMemberByEmail(email);
        return member.map(m -> {
            MemberDTO dto = new MemberDTO(m.getName(), m.getEmail(),
m.getMembershipPackage());
            dto.setTrainerName(m.getTrainer() != null ?
m.getTrainer().getName() : null);
```

```
            dto.setScheduleDetails(m.getSchedule() != null ?
String.valueOf(m.getSchedule()) : null); // Make sure toString() provides
meaningful information
            dto.setFacultyName(m.getFaculty() != null ?
m.getFaculty().getName() : null);
            return ResponseEntity.ok(dto);
        }).orElseGet(() -> ResponseEntity.notFound().build());
    }

}
```

```
package com.example.demo.controller;

import com.example.demo.model.Schedule;
import com.example.demo.services.ScheduleService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("/api/schedules")
public class ScheduleController {

    private final ScheduleService scheduleService;

    @Autowired
    public ScheduleController(ScheduleService scheduleService) {
        this.scheduleService = scheduleService;
    }

//    @GetMapping
//    public ResponseEntity<List<Schedule>> getAllSchedules() {
//        List<Schedule> schedules = scheduleService.getAllSchedules();
//        return ResponseEntity.ok(schedules);
//    }

    @GetMapping("/{id}")
    public ResponseEntity<Schedule> getScheduleById(@PathVariable Long id) {
        Optional<Schedule> schedule = scheduleService.getScheduleById(id);
        return schedule.map(ResponseEntity::ok).orElseGet(() ->
ResponseEntity.notFound().build());
    }

    @PostMapping
    public ResponseEntity<Schedule> createSchedule(@RequestBody Schedule
schedule) {
        Schedule savedSchedule = scheduleService.saveSchedule(schedule);
        return ResponseEntity.ok(savedSchedule);
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteSchedule(@PathVariable Long id) {
        scheduleService.deleteSchedule(id);
```

```
        return ResponseEntity.noContent().build();
    }

    @GetMapping("/all")
    public ResponseEntity<List<Schedule>> getAllSchedules() {
        List<Schedule> schedules = scheduleService.getAllSchedules();
        return ResponseEntity.ok(schedules);
    }

    @PutMapping("/{id}")
    public ResponseEntity<Schedule> updateSchedule(@PathVariable Long id,
@RequestBody Schedule scheduleDetails) {
        return scheduleService.getScheduleById(id)
                .map(schedule -> {
                    schedule.setDay(scheduleDetails.getDay());
                    schedule.setTime(scheduleDetails.getTime());
                    Schedule updatedSchedule =
scheduleService.saveSchedule(schedule);
                    return ResponseEntity.ok(updatedSchedule);
                })
                .orElseGet(() -> ResponseEntity.notFound().build());
    }

}
```

```
package com.example.demo.controller;

import com.example.demo.dto.MemberDTO;
import com.example.demo.model.Member;
import com.example.demo.model.Trainer;
import com.example.demo.services.TrainerService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("/api/trainers")
public class TrainerController {

    private final TrainerService trainerService;

    @Autowired
    public TrainerController(TrainerService trainerService) {
        this.trainerService = trainerService;
    }

//    @GetMapping
//    public ResponseEntity<List<Trainer>> getAllTrainers() {
//        List<Trainer> trainers = trainerService.getAllTrainers();
//        return ResponseEntity.ok(trainers);
//    }
```

```
    @GetMapping("/{id}")
    public ResponseEntity<Trainer> getTrainerById(@PathVariable Long id) {
        Optional<Trainer> trainer = trainerService.getTrainerById(id);
        return trainer.map(ResponseEntity::ok).orElseGet(() ->
ResponseEntity.notFound().build());
    }

    @PostMapping("/")
    public ResponseEntity<Trainer> createTrainer(@RequestBody Trainer
trainer) {
        try {
            Trainer savedTrainer = trainerService.saveTrainer(trainer);
            return
ResponseEntity.status(HttpStatus.CREATED).body(savedTrainer);
        } catch (Exception e) {
            System.out.println("Failed to create trainer: " +
e.getMessage());
            return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
        }
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteTrainer(@PathVariable Long id) {
        trainerService.deleteTrainer(id);
        return ResponseEntity.noContent().build();
    }


    @GetMapping("/all")
    public ResponseEntity<List<Trainer>> getAllTrainers() {
        List<Trainer> trainers = trainerService.getAllTrainers();
        return ResponseEntity.ok(trainers);
    }

    @PutMapping("/{id}")
    public ResponseEntity<Trainer> updateTrainer(@PathVariable Long id,
@RequestBody Trainer updatedTrainer) {
        return trainerService.getTrainerById(id)
                .map(trainer -> {
                    trainer.setName(updatedTrainer.getName());
                    Trainer savedTrainer =
trainerService.saveTrainer(trainer);
                    return ResponseEntity.ok(savedTrainer);
                })
                .orElseGet(() -> ResponseEntity.notFound().build());
    }

}

package com.example.demo.dto;

public class FacultyDTO {
    private String name;

    public String getName() {
        return name;
```

```
    }

    public void setName(String name) {
        this.name = name;
    }
}


package com.example.demo.dto;

public class JoinRequestDTO {
    private String name;
    private String email;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getMembershipPackage() {
        return membershipPackage;
    }

    public void setMembershipPackage(String membershipPackage) {
        this.membershipPackage = membershipPackage;
    }

    private String membershipPackage;

    // Getters and setters
}

package com.example.demo.dto;

import com.example.demo.model.Faculty;
import com.example.demo.model.Schedule;
import com.example.demo.model.Trainer;

import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;

public class MemberDTO {


    private Long id;
```

```
    public MemberDTO(Long id, String name, String email, String
membershipPackage, Long trainerId, Long scheduleId, Long facultyId, String
trainerName, String scheduleDetails, String facultyName, Schedule schedule) {
        this.id = id;
        this.name = name;
        this.email = email;
        this.membershipPackage = membershipPackage;
        this.trainerId = trainerId;
        this.scheduleId = scheduleId;
        this.facultyId = facultyId;
        TrainerName = trainerName;
        ScheduleDetails = scheduleDetails;
        FacultyName = facultyName;
        this.schedule = schedule;
    }
    public MemberDTO(Long id, String name, String email, String
membershipPackage) {
        this.id = id;
        this.name = name;
        this.email = email;
        this.membershipPackage = membershipPackage;
    }

    public MemberDTO() {
    }

    private String name;
    private String email;
    private String membershipPackage;

    private Long trainerId;
    private Long scheduleId;
    private Long facultyId;

    private String TrainerName;

    private String ScheduleDetails;

    private String FacultyName;

    public String getTrainerName() {
        return TrainerName;
    }

    public void setTrainerName(String trainerName) {
        TrainerName = trainerName;
    }

    public String getScheduleDetails() {
        return ScheduleDetails;
    }

    public void setScheduleDetails(String scheduleDetails) {
        ScheduleDetails = scheduleDetails;
    }

    public Long getId() {
```

```
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public String getFacultyName() {
    return FacultyName;
}

public void setFacultyName(String facultyName) {
    FacultyName = facultyName;
}

public Long getTrainerId() {
    return trainerId;
}

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

public Long getScheduleId() {
    return scheduleId;
}

public void setScheduleId(Long scheduleId) {
    this.scheduleId = scheduleId;
}

public Long getFacultyId() {
    return facultyId;
}

public void setFacultyId(Long facultyId) {
    this.facultyId = facultyId;
}

public Schedule getSchedule() {
    return schedule;
}

public void setSchedule(Schedule schedule) {
    this.schedule = schedule;
}

private Schedule schedule;

public MemberDTO(String name, String email, String membershipPackage) {
    this.name = name;
    this.email = email;
    this.membershipPackage = membershipPackage;
}

public String getName() {
    return name;
```

```
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getMembershipPackage() {
        return membershipPackage;
    }

    public void setMembershipPackage(String membershipPackage) {
        this.membershipPackage = membershipPackage;
    }
}
```

```
package com.example.demo.dto;

public class ScheduleDTO {
    private String day;
    private String time;

    public String getDay() {
        return day;
    }

    public void setDay(String day) {
        this.day = day;
    }

    public String getTime() {
        return time;
    }

    public void setTime(String time) {
        this.time = time;
    }
}
```

```
package com.example.demo.dto;

public class TrainerDTO {
    private String name;

    public String getName() {
        return name;
```

```
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

```
package com.example.demo.model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Faculty {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;

    // Default constructor required by JPA
    public Faculty() {}

    // Constructor for setting the ID (used elsewhere in your code)
    public Faculty(Long id) {
        this.id = id;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

```
package com.example.demo.model;


import javax.persistence.*;

@Entity
public class Member {
    @Id
```

```
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;

private String name;

@ManyToOne
@JoinColumn(name = "faculty_id")
private Faculty faculty;

@ManyToOne
@JoinColumn(name = "trainer_id")
private Trainer trainer;

@ManyToOne
@JoinColumn(name = "schedule_id")
private Schedule schedule;

public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getMembershipPackage() {
    return membershipPackage;
}

public void setMembershipPackage(String membershipPackage) {
    this.membershipPackage = membershipPackage;
}

private String email;

public Faculty getFaculty() {
    return faculty;
}

public void setFaculty(Faculty faculty) {
    this.faculty = faculty;
```

```
    }

    public Trainer getTrainer() {
        return trainer;
    }

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

    public Schedule getSchedule() {
        return schedule;
    }

    public void setSchedule(Schedule schedule) {
        this.schedule = schedule;
    }

    private String membershipPackage;



}
```

```
package com.example.demo.model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Schedule {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String day;
    private String time;

    // Add other fields if necessary

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getDay() {
        return day;
    }

    public void setDay(String day) {
        this.day = day;
```

```
    }

    public String getTime() {
        return time;
    }

    public void setTime(String time) {
        this.time = time;
    }

    // Override the toString method to provide a more informative output
    @Override
    public String toString() {
        return day + " " + time; // Customize this format as needed
    }
}
```

```
package com.example.demo.model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Trainer {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;

    public Trainer(Long id) {
        this.id = id;
    }

    public Trainer() {

    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

```java
package com.example.demo.repositories;

import com.example.demo.model.Faculty;
import org.springframework.data.jpa.repository.JpaRepository;

public interface FacultyRepository extends JpaRepository<Faculty, Long> {
}
```

```java
package com.example.demo.repositories;

import com.example.demo.model.Member;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.Optional;

public interface MemberRepository extends JpaRepository<Member, Long> {
    boolean existsByEmail(String email);

    Optional<Member> findByEmail(String email);
}
```

```java
package com.example.demo.repositories;

import com.example.demo.model.Schedule;
import org.springframework.data.jpa.repository.JpaRepository;

public interface ScheduleRepository extends JpaRepository<Schedule, Long> {
}
```

```java
package com.example.demo.repositories;

import com.example.demo.model.Trainer;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.Optional;

public interface TrainerRepository extends JpaRepository<Trainer, Long> {
}
```

```java
package com.example.demo.services;

import com.example.demo.model.Faculty;
import com.example.demo.repositories.FacultyRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.transaction.Transactional;
import java.util.List;
import java.util.Optional;

@Service
public class FacultyService {
    private final FacultyRepository facultyRepository;
```

```
    @Autowired
    public FacultyService(FacultyRepository facultyRepository) {
        this.facultyRepository = facultyRepository;
    }

    public List<Faculty> getAllFaculties() {
        return facultyRepository.findAll();
    }

    public Optional<Faculty> getFacultyById(Long id) {
        return facultyRepository.findById(id);
    }

    public Faculty saveFaculty(Faculty faculty) {
        return facultyRepository.save(faculty);
    }

    public void deleteFaculty(Long id) {
        facultyRepository.deleteById(id);
    }

    @Transactional
    public Faculty saveOrUpdate(Faculty faculty) {
        return facultyRepository.save(faculty);
    }

    @Transactional
    public void deleteTrainer(Long id) {

           facultyRepository.deleteById(id);

    }
}



package com.example.demo.services;

import com.example.demo.dto.JoinRequestDTO;
import com.example.demo.model.Member;
import com.example.demo.repositories.MemberRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
public class MemberServiceImpl {

    private final MemberRepository memberRepository;

    @Autowired
    public MemberServiceImpl(MemberRepository memberRepository) {
        this.memberRepository = memberRepository;
    }
```

```
    public Member join(JoinRequestDTO request) {
        Member member = new Member();
        member.setName(request.getName());
        member.setEmail(request.getEmail());
        member.setMembershipPackage(request.getMembershipPackage());
        // Save the member to the database
        return memberRepository.save(member);
    }

    public boolean emailExists(String email) {
        return memberRepository.existsByEmail(email);
    }

    public List<Member> getAllMembers() {
        return memberRepository.findAll();
    }

    public void deleteMember(Long id) {
        memberRepository.deleteById(id);
    }

    public Member saveMember(Member member) {
        return memberRepository.save(member);
    }


    public Optional<Member> login(String email) {
        return memberRepository.findByEmail(email);
    }

    public Optional<Member> getMemberByEmail(String email) {
        return memberRepository.findByEmail(email);
    }

    public Optional<Member> getMemberById(Long id) {
        return memberRepository.findById(id);
    }
}
```

```
package com.example.demo.services;

import com.example.demo.model.Member;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Optional;

@Service
public class ProfileService {
    private final MemberServiceImpl memberService;

    @Autowired
    public ProfileService(MemberServiceImpl memberService) {
```

```
        this.memberService = memberService;
    }

    public Optional<Member> getMemberProfile(String email) {
        return memberService.login(email);
    }
}
```

```
package com.example.demo.services;

import com.example.demo.model.Schedule;
import com.example.demo.repositories.ScheduleRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
public class ScheduleService {
    private final ScheduleRepository scheduleRepository;

    @Autowired
    public ScheduleService(ScheduleRepository scheduleRepository) {
        this.scheduleRepository = scheduleRepository;
    }

    public List<Schedule> getAllSchedules() {
        return scheduleRepository.findAll();
    }

    public Optional<Schedule> getScheduleById(Long id) {
        return scheduleRepository.findById(id);
    }

    public Schedule saveSchedule(Schedule schedule) {
        return scheduleRepository.save(schedule);
    }

    public void deleteSchedule(Long id) {
        scheduleRepository.deleteById(id);
    }
}
```

```
package com.example.demo.services;

import com.example.demo.model.Member;
import com.example.demo.model.Trainer;
import com.example.demo.repositories.TrainerRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;
```

```
@Service
public class TrainerService {
    private final TrainerRepository trainerRepository;

    @Autowired
    public TrainerService(TrainerRepository trainerRepository) {
        this.trainerRepository = trainerRepository;
    }

    public List<Trainer> getAllTrainers() {
        return trainerRepository.findAll();
    }

    public Optional<Trainer> getTrainerById(Long id) {
        return trainerRepository.findById(id);
    }

    public Trainer saveTrainer(Trainer trainer) {
        return trainerRepository.save(trainer);
    }

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


}
```

```
package com.example.demo;

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

@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
       SpringApplication.run(DemoApplication.class, args);
    }

}
```

```
function fetchInitialData() {
    fetchTrainers();
    fetchSchedules();
    fetchMembers();
    fetchFaculties();
}

function updateList(data, listId, formSetter) {
    const tableBody = document.getElementById(listId);
    tableBody.innerHTML = '';
```

```
    data.forEach(item => {
        const row = tableBody.insertRow();
        for (const key in item) {
            const cell = row.insertCell();
            cell.textContent = item[key];
        }
        const editCell = row.insertCell();
        editCell.appendChild(createEditButton(item, formSetter)); // Ensure
formSetter is being used here

        const deleteCell = row.insertCell();
        deleteCell.appendChild(createDeleteButton(item, listId, () =>
fetchInitialData()));
    });
}



function createEditButton(item, setForm) {
    const button = document.createElement('button');
    button.textContent = 'Edit';
    button.onclick = () => setForm(item);
    return button;
}

// function createDeleteButton(item, listId, fetchFunction) {
//     const modelType = listId.replace('-list', 's'); // Append 's' to make
it plural correctly
//     const button = document.createElement('button');
//     button.textContent = 'Delete';
//     button.onclick = () => {
//         const url = `/api/${modelType}/${item.id}`; // Correctly forming
the URL here
//         fetch(url, { method: 'DELETE' })
//             .then(response => {
//                 if (!response.ok) throw new Error(`Failed to delete
${modelType.slice(0, -1)} with ID ${item.id}`);
//                 fetchFunction();
//             })
//             .catch(error => console.error('Error:', error));
//     };
//     return button;
// }

function getPluralModelType(modelType) {
    if (modelType === 'faculty') {
        return 'faculties';
    }
    return modelType + 's';  // Default to just adding 's'
}

function createDeleteButton(item, listId, fetchFunction) {
    const singularModelType = listId.replace('-list', '');  // Strip '-list'
    const modelType = getPluralModelType(singularModelType);  // Correctly
handle irregular plurals
    const button = document.createElement('button');
    button.textContent = 'Delete';
```

```
    button.onclick = () => {
        const url = `/api/${modelType}/${item.id}`;
        fetch(url, { method: 'DELETE' })
            .then(response => {
                if (!response.ok) throw new Error(`Failed to delete
${singularModelType} with ID ${item.id}`);
                fetchFunction();
            })
            .catch(error => console.error('Error:', error));
    };
    return button;
}


function setTrainerForm(trainer) {
    document.getElementById('trainer-id').value = trainer.id;
    document.getElementById('trainer-name').value = trainer.name;
}

function setScheduleForm(schedule) {
    document.getElementById('schedule-id').value = schedule.id;
    document.getElementById('schedule-day').value = schedule.day;
    document.getElementById('schedule-time').value = schedule.time;
}

function setMemberForm(member) {
    document.getElementById('member-id').value = member.id;
    document.getElementById('member-name').value = member.name;
    document.getElementById('member-email').value = member.email;
    document.getElementById('member-package').value =
member.membership_package;
}

function setFacultyForm(faculty) {
    document.getElementById('faculty-id').value = faculty.id;
    document.getElementById('faculty-name').value = faculty.name;
}

function resetTrainerForm() {
    document.getElementById('trainer-id').value = '';
    document.getElementById('trainer-name').value = '';
}

function resetScheduleForm() {
    document.getElementById('schedule-id').value = '';
    document.getElementById('schedule-day').value = '';
    document.getElementById('schedule-time').value = '';
}

function resetMemberForm() {
    document.getElementById('member-id').value = '';
    document.getElementById('member-name').value = '';
    document.getElementById('member-email').value = '';
    document.getElementById('member-package').value = '';
}

function resetFacultyForm() {
```

```
    document.getElementById('faculty-id').value = '';
    document.getElementById('faculty-name').value = '';
}
function fetchTrainers() {
    fetch('/api/trainers/all')
        .then(response => {
            if (!response.ok) throw new Error('Failed to fetch trainers');
            return response.json();
        })
        .then(data => updateList(data, 'trainer-list', setTrainerForm)) //
Ensure setTrainerForm is passed here
        .catch(error => console.error('Error fetching trainers:', error));
}


// Fetch Schedules
function fetchSchedules() {
    fetch('/api/schedules/all')
        .then(response => {
            if (!response.ok) throw new Error('Failed to fetch schedules');
            return response.json();
        })
        .then(data => updateList(data, 'schedule-list', setScheduleForm))
        .catch(error => console.error('Error fetching schedules:', error));
}

// Fetch Members
function fetchMembers() {
    fetch('/api/members')
        .then(response => {
            if (!response.ok) throw new Error('Failed to fetch members');
            return response.json();
        })
        .then(data => updateList(data, 'member-list', setMemberForm))
        .catch(error => console.error('Error fetching members:', error));
}

// Fetch Faculties
function fetchFaculties() {
    fetch('/api/faculties/all')
        .then(response => {
            if (!response.ok) throw new Error('Failed to fetch faculties');
            return response.json();
        })
        .then(data => updateList(data, 'faculty-list', setFacultyForm))
        .catch(error => console.error('Error fetching faculties:', error));
}

function submitTrainerForm() {
    const id = document.getElementById('trainer-id').value;
    const name = document.getElementById('trainer-name').value;
    const method = id ? 'PUT' : 'POST';
    const url = id ? `/api/trainers/${id}` : '/api/trainers/';

    fetch(url, {
        method: method,
        headers: { 'Content-Type': 'application/json' },
```

```
        body: JSON.stringify({ name })
    })
        .then(response => {
            if (!response.ok) throw new Error('Failed to submit trainer
data');
            return response.json();
        })
        .then(() => {
            resetTrainerForm();
            fetchTrainers(); // Reload the list
        })
        .catch(error => console.error('Error:', error));
    return false; // Prevent default form submission
}
// Submit Schedule Form
function submitScheduleForm() {
    const id = document.getElementById('schedule-id').value;
    const day = document.getElementById('schedule-day').value;
    const time = document.getElementById('schedule-time').value;
    const method = id ? 'PUT' : 'POST';
    const url = id ? `/api/schedules/${id}` : '/api/schedules/';

    fetch(url, {
        method: method,
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ day, time })
    })
        .then(response => response.ok ? response.json() :
Promise.reject('Failed to submit schedule data'))
        .then(() => {
            resetScheduleForm();
            fetchSchedules(); // Reload the list
        })
        .catch(error => console.error('Error:', error));

    return false;
}

// Submit Member Form
function submitMemberForm() {
    const id = document.getElementById('member-id').value;
    const name = document.getElementById('member-name').value;
    const email = document.getElementById('member-email').value;
    const membership_package = document.getElementById('member-
package').value;
    const method = id ? 'PUT' : 'POST';
    const url = id ? `/api/members/${id}` : '/api/members/';

    fetch(url, {
        method: method,
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ name, email, membership_package })
    })
        .then(response => response.ok ? response.json() :
Promise.reject('Failed to submit member data'))
        .then(() => {
            resetMemberForm();
```

```
            fetchMembers(); // Reload the list
        })
        .catch(error => console.error('Error:', error));

    return false;
}

// Submit Faculty Form
function submitFacultyForm() {
    const id = document.getElementById('faculty-id').value;
    const name = document.getElementById('faculty-name').value;
    const method = id ? 'PUT' : 'POST';
    const url = id ? `/api/faculties/${id}` : '/api/faculties/';

    fetch(url, {
        method: method,
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ name })
    })
        .then(response => response.ok ? response.json() :
Promise.reject('Failed to submit faculty data'))
        .then(() => {
            resetFacultyForm();
            fetchFaculties(); // Reload the list
        })
        .catch(error => console.error('Error:', error));

    return false;
}


var btn_up = document.querySelector(".up");

function animate() {
    if (window.scrollY > 0) {
        scrollBy(0, -40);
        setTimeout(animate, 1);
    }
}

btn_up.addEventListener("click", function(e) {
    e.preventDefault();
    animate();
}, false);

window.addEventListener("scroll", function() {
    if (this.scrollY >= 600) {
        btn_up.classList.remove("up-opa");
    } else {
        btn_up.classList.add("up-opa");
    }
}, false);

// Menu
var btnMenu = document.getElementById("nav-menu");
btnMenu.addEventListener("click", toggleMenu);
```

```
function toggleMenu(e) {
    var divOverlay = document.getElementsByClassName("overlay")[0];
    var speed = 10;
    var eventTarget = e.target;

    if (eventTarget.className == "btn-open") {
        fadeIn(divOverlay, speed);
        eventTarget.className = "btn-close";
    } else if (eventTarget.className == "btn-close") {
        fadeOut(divOverlay, speed);
        eventTarget.className = "btn-open";
    }
}

function fadeIn(elem, speed) {
    var inInterval = setInterval(function() {
        elem.style.opacity = Number(elem.style.opacity) + 0.02;
        if (elem.style.opacity >= 1) {
            elem.style.opacity = 1;
            clearInterval(inInterval);
        }
    }, speed); // 10ms == .01s
}

function fadeOut(elem, speed) {
    var outInterval = setInterval(function() {
        elem.style.opacity = Number(elem.style.opacity) - 0.02;
        if (elem.style.opacity <= 0) {
            elem.style.opacity = 0;
            clearInterval(outInterval);
        }
    }, speed); // 10ms == .01s
}


// Smooth scrolling to section
(function($) {
    'use strict';

    $('.scrollTo').on('click', function(e) {
        e.preventDefault();
        var href = $(this).attr('href');
        $('html, body').animate({
            scrollTop: $(href).offset().top + 'px'
        }, 1500, function() {
            location.hash = href;
        });
    });

}(jQuery));



document.addEventListener("DOMContentLoaded", function() {
    const form = document.getElementById('join-form');
    const emailInput = form.elements['email'];
```

```
    const emailError = document.createElement('div');
    emailError.className = 'error-message';
    emailInput.parentNode.insertBefore(emailError, emailInput.nextSibling);

    // Email validation function
    function validateEmail() {
        const email = emailInput.value;
        const emailPattern = /[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}/i;
        if (email === '') {
            emailError.textContent = 'Email is required.';
            return false;
        } else if (!emailPattern.test(email)) {
            emailError.textContent = 'Please enter a valid email address.';
            return false;
        } else {
            emailError.textContent = '';
            return true;
        }
    }

    // Validate email in real-time
    emailInput.addEventListener('input', function() {
        validateEmail();
    });

    // Form submission event listener
    form.addEventListener('submit', function(event) {
        event.preventDefault();

        // Validate email before submission
        if (!validateEmail()) {
            alert('Please correct the errors in the form before
submitting.');
            return false;
        }

        // Gather form data
        const formData = {
            name: form.elements['name'].value,
            email: form.elements['email'].value,
            membershipPackage: form.elements['membershipPackage'].value
        };

        // Send the form data using Fetch API
        fetch('/join', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify(formData)
        })
            .then(response => {
                if (!response.ok) {
                    throw response;
                }
                return response.json();
            })
```

```
            .then(data => {
                console.log('Success:', data);
                alert('You have successfully joined!');
            })
            .catch((error) => {
                error.text().then(errorMessage => {
                    console.error('Error:', errorMessage);
                    emailError.textContent = errorMessage;
                    emailInput.focus();
                });
            });
    });
});

document.getElementById('loginForm').addEventListener('submit',
function(event) {
    event.preventDefault();
    const email = document.getElementById('email').value;
    fetch('/api/members/login', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({ email: email })
    })
    .then(response => response.json())
    .then(data => {
        if (data.token) {
            sessionStorage.setItem('token', data.token); // Сохранение токена
            window.location.href = '/profile.html'; // Перенаправление
        } else {
            console.error('Ошибка входа');
        }
    })
    .catch(error => console.error('Ошибка во время входа:', error));
});



<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Admin Dashboard</title>
    <link rel="stylesheet" href="css/admin.css">
    <script src="/js/admin.js"></script>
</head>
<body>
<h1>Панель администратора</h1>

<!-- Trainers Section -->
<div id="trainers">
    <h2>Тренер</h2>
    <form id="trainer-form" onsubmit="return submitTrainerForm()">
        <input type="hidden" id="trainer-id">
```

```
        <input type="text" id="trainer-name" placeholder="Trainer Name">
        <button type="submit">Save</button>
    </form>
    <table>
        <thead>
        <tr>
            <th>Имя</th>
            <th>Специальность</th>
            <th>Действия</th>
        </tr>
        </thead>
        <tbody id="trainer-list"></tbody>
    </table>
</div>

<!-- Schedules Section -->
<div id="schedules">
    <h2>Расписания</h2>
    <form id="schedule-form" onsubmit="return submitScheduleForm()">
        <input type="hidden" id="schedule-id">
        <input type="text" id="schedule-day" placeholder="Day">
        <input type="text" id="schedule-time" placeholder="Time">
        <button type="submit">Save</button>
    </form>
    <table>
        <thead>
        <tr>
            <th>День</th>
            <th>Время</th>
            <th>Действия</th>
        </tr>
        </thead>
        <tbody id="schedule-list"></tbody>
    </table>
</div>

<!-- Members Section -->
<div id="members">
    <h2>Пользователи</h2>
    <form id="member-form" onsubmit="return submitMemberForm()">
        <input type="hidden" id="member-id">
        <input type="text" id="member-name" placeholder="Member Name">
        <input type="text" id="member-email" placeholder="Email">
        <input type="text" id="member-package" placeholder="Membership
Package">
        <button type="submit">Save</button>
    </form>
    <table>
        <thead>
        <tr>
            <th>Имя</th>
            <th>Электронная почта</th>
            <th>Пакет</th>
            <th>Действия</th>
        </tr>
        </thead>
        <tbody id="member-list"></tbody>
```

```
    </table>
</div>

<!-- Faculties Section -->
<div id="faculties">
    <h2>отдел обучения</h2>
    <form id="faculty-form" onsubmit="return submitFacultyForm()">
        <input type="hidden" id="faculty-id">
        <input type="text" id="faculty-name" placeholder="Faculty Name">
        <button type="submit">Save</button>
    </form>
    <table>
        <thead>
        <tr>
            <th>Имя</th>
            <th>Действия</th>
        </tr>
        </thead>
        <tbody id="faculty-list"></tbody>
    </table>
</div>

<script>
    document.addEventListener("DOMContentLoaded", fetchInitialData);
</script>
</body>
</html>


<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>Фитнес-клуб</title>


    <link href="css/bootstrap.min.css" rel="stylesheet">
    <link href="css/style.css" rel="stylesheet">
    <link href="css/media-queries.css" rel="stylesheet">


    <script src="https://use.fontawesome.com/0e46e58ec0.js"></script>

</head>
<body>

<section id="main">
    <div class="container">
        <div class="button">
            <a href="#" id="nav-menu" class="btn-open"></a>
        </div>
        <div class="overlay">
            <div class="nav-container">
                <div class="nav-overlay">
```

```
                    <ul>
                        <li><a href="#" class="scrollTo">Главная</a></li>
                        <li><a href="#perform"
class="scrollTo">Производительность</a></li>
                        <li><a href="#trainers"
class="scrollTo">Тренеры</a></li>
                        <li><a href="#result"
class="scrollTo">Результаты</a></li>
                        <li><a href="register.html" class="scotollt">
Регистрация</a> </li>
                        <li><a href="login.html" class="scotollt"> Вход</a>
</li>
                        <li><a href="profile.html" class="scotollt">
Профиль</a></li>
                    </ul>
                </div>
            </div>
        </div>
        <header>
            <div class="row">
                <div class="col-md-10 col-md-offset-1 col-sm-12 col-xs-12
text-center">
                    <p class="main-text">Без боли нет пользы</p>
                    <p class="main-text-1">Мотивация - это то, что заставляет
вас начать. Привычка - это то, что заставляет вас двигаться вперед.</p>
                    <div class="main-btn">
                        <a href="#world"><button class="btn-
join">Присоединяйтесь сегодня</button></a>
                    </div>
                </div>
            </div>
        </header>
    </div>
</section>


<section id="world">
    <div class="container">
        <div class="row">
            <div class="col-md-8 col-md-offset-2 col-sm-6 col-sm-offset-3
col-xs-12 text-center">
                <div class="world-main">
                    <p class="title">Фитнес мирового класса</p>
                    <p class="world-text">В GG Gym всё, что мы делаем от
начала и до конца, имеет самое высокое качество, чтобы мы были готовы помочь
вам достичь ваших целей здорового и подтянутого образа жизни.</p>
                </div>
            </div>
        </div>
    </div>
</section>


<section id="fusion">
    <div class="container">
        <div class="row fusion-row">
            <div class="col-md-3 col-sm-6 col-xs-12 fusion-col">
```

```
                <div class="fusion-icon">
                    <div class="icon">
                        <i class="fa fa-chain-broken" aria-hidden="true"></i>
                    </div>
                    <h4>Силовые тренировки и кондиционирование</h4>
                </div>
                <div class="fusion-text">
                    <p class="fusion-text-2">
                        Клиент очень важен, за клиентом пойдет клиент. В
продвижении игроков, для загрузки карьеры.
                    </p>
                </div>
            </div>
            <div class="col-md-3 col-sm-6 col-xs-12 fusion-col">
                <div class="fusion-icon">
                    <div class="icon">
                        <i class="fa fa-bicycle" aria-hidden="true"></i>
                    </div>
                    <h4>Фитнес и кардио</h4>
                </div>
                <div class="fusion-text">
                    <p class="fusion-text-2">
                        Клиент очень важен, за клиентом пойдет клиент. В
продвижении игроков, для загрузки карьеры.
                    </p>
                </div>
            </div>
            <div class="col-md-3 col-sm-6 col-xs-12 fusion-col">
                <div class="fusion-icon">
                    <div class="icon">
                        <i class="fa fa-skyatlas" aria-hidden="true"></i>
                    </div>
                    <h4>Гибкость и отдых</h4>
                </div>
                <div class="fusion-text">
                    <p class="fusion-text-2">
                        Клиент очень важен, за клиентом пойдет клиент. В
продвижении игроков, для загрузки карьеры.
                    </p>
                </div>
            </div>
            <div class="col-md-3 col-sm-6 col-xs-12 fusion-col">
                <div class="fusion-icon">
                    <div class="icon">
                        <i class="fa fa-heartbeat" aria-hidden="true"></i>
                    </div>
                    <h4>Здоровье и диета</h4>
                </div>
                <div class="fusion-text">
                    <p class="fusion-text-2">
                        Клиент очень важен, за клиентом пойдет клиент. В
продвижении игроков, для загрузки карьеры.
                    </p>
                </div>
            </div>
        </div>
    </div>
```

```
</section>


<section id="perform">
    <div class="container">
        <div class="row">
            <div class="col-md-8 col-md-offset-2 text-center perf-col-title">
                <div class="title">Высокотехнологичные объекты</div>
            </div>
        </div>
        <div class="row">
            <div class="col-md-4 col-sm-12 col-xs-12 perf-col">
                <img src="img/perfomance/per1.jpg" alt="per1" >
            </div>
            <div class="col-md-4 col-sm-12 col-xs-12 perf-col">
                <img src="img/perfomance/per2.jpg" alt="per2" >
            </div>
            <div class="col-md-4 col-sm-12 col-xs-12 perf-col">
                <img src="img/perfomance/per3.jpg" alt="per3" >
            </div>
            <div class="col-md-4 col-sm-12 col-xs-12 perf-col">
                <img src="img/perfomance/per4.jpg" alt="per4" >
            </div>
            <div class="col-md-4 col-sm-12 col-xs-12 perf-col">
                <img src="img/perfomance/per5.jpg" alt="per5" >
            </div>
            <div class="col-md-4 col-sm-12 col-xs-12 perf-col">
                <img src="img/perfomance/per6.jpg" alt="per6" >
            </div>
            <div class="col-md-6 col-md-offset-3 text-center">
                <div class="perform-btn">
                    <button type="button" class="btn-per">Просмотреть все
объекты</button>
                </div>
            </div>
        </div>
    </div>
</section>


<section id="streng">
    <div class="container">
        <div class="row streng-con">
            <div class="col-md-6 col-sm-12 col-xs-12">
                <div class="streng-1">
                    <div class="streng-text">
                        <p class="text-1">Фитнес-центр</p>
                        <p class="text-2">Безупречная техника и стратегия —
залог успеха на спортивной арене. </p>
                    </div>
                </div>
            </div>
            <div class="col-md-6 col-sm-12 col-xs-12">
                <div class="streng-2">
                    <div class="streng-text">
                        <p class="text-1">Силовые тренировки и
кондиционирование</p>
```

```
                        <p class="text-2">Тренировка — это ключ к
совершенству. Чем больше усилий вы вкладываете, тем ближе к победе.</p>
                    </div>
                </div>
            </div>
            <div class="col-md-6 col-sm-12 col-xs-12">
                <div class="streng-3">
                    <div class="streng-text">
                        <p class="text-1">Бассейн</p>
                        <p class="text-2">Победа не приходит к тем, кто ждет,
а к тем, кто действует. Успех — это результат постоянной работы и
самоотдачи.</p>
                    </div>
                </div>
            </div>
            <div class="col-md-6 col-sm-12 col-xs-12">
                <div class="streng-4">
                    <div class="streng-text">
                        <p class="text-1">Спортивные занятия</p>
                        <p class="text-2">Спорт — это не просто игра, это
образ жизни. Он формирует характер, научит вас преодолевать препятствия и
стремиться к своим целям.</p>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>

<section id="trainers">
    <div class="container">
        <div class="title">Обучайтесь рядом с лучшими</div>
        <div class="row-t">
            <div class="trainers-col">
                <div class="trainers-img">
                    <img src="img/trainer/trainer1.jpg" alt="trn1" >
                </div>
                <div class="trainers-down">
                    <p class="trainers-text">
                        Анна Бейкер<br>
                        Личный тренер
                    </p>
                    <p class="trainers-text-2">
                        Клиент очень важен, за клиентом пойдет клиент. В
продвижении игроков, для загрузки карьеры.
                    </p>
                    <div class="trainers-social">
                        <i class="fa fa-facebook" aria-hidden="true"></i>
                        <i class="fa fa-twitter" aria-hidden="true"></i>
                        <i class="fa fa-instagram" aria-hidden="true"></i>
                    </div>
                </div>
            </div>
            <div class="trainers-col">
                <div class="trainers-img">
                    <img src="img/trainer/trainer2.jpg" alt="trn2" >
                </div>
```

```
                <div class="trainers-down">
                    <p class="trainers-text">
                        Питер Райс<br>
                        Личный тренер
                    </p>
                    <p class="trainers-text-2">
                        Клиент очень важен, за клиентом пойдет клиент. В
продвижении игроков, для загрузки карьеры.
                    </p>
                    <div class="trainers-social">
                        <i class="fa fa-facebook" aria-hidden="true"></i>
                        <i class="fa fa-twitter" aria-hidden="true"></i>
                        <i class="fa fa-instagram" aria-hidden="true"></i>
                    </div>
                </div>
            </div>
            <div class="trainers-col">
                <div class="trainers-img">
                    <img src="img/trainer/trainer3.jpg" alt="trn3" >
                </div>
                <div class="trainers-down">
                    <p class="trainers-text">
                        Анна Уоррен<br>
                        Личный тренер
                    </p>
                    <p class="trainers-text-2">
                        Клиент очень важен, за клиентом пойдет клиент. В
продвижении игроков, для загрузки карьеры.
                    </p>
                    <div class="trainers-social">
                        <i class="fa fa-facebook" aria-hidden="true"></i>
                        <i class="fa fa-twitter" aria-hidden="true"></i>
                        <i class="fa fa-instagram" aria-hidden="true"></i>
                    </div>
                </div>
            </div>
            <div class="trainers-col">
                <div class="trainers-img">
                    <img src="img/trainer/trainer4.jpg" alt="trn4" >
                </div>
                <div class="trainers-down">
                    <p class="trainers-text">
                        Алекс Сильва<br>
                        Личный тренер
                    </p>
                    <p class="trainers-text-2">
                        Клиент очень важен, за клиентом пойдет клиент. В
продвижении игроков, для загрузки карьеры.
                    </p>
                    <div class="trainers-social">
                        <i class="fa fa-facebook" aria-hidden="true"></i>
                        <i class="fa fa-twitter" aria-hidden="true"></i>
                        <i class="fa fa-instagram" aria-hidden="true"></i>
                    </div>
                </div>
            </div>
        </div>
```

```
    </div>
</section>

<section id="ali">
    <div class="container">
        <div class="row">
            <div class="col-md-8 col-md-offset-2 text-center">
                <div class="ali-text">
                    <p class="ali-text-1">"Бой выигрывается или проигрывается
далеко от глаз свидетелей - за линиями, в тренажерном зале и там, на дороге,
задолго до того, как я буду танцевать под этими огнями."</p>
                    <p class="ali-text-2">Мухаммед Али   Цитата дня</p>
                </div>
            </div>
        </div>
    </div>
</section>


<section id="result">
    <div class="container">
        <div class="title title-1">
            РЕАЛЬНЫЕ РЕЗУЛЬТАТЫ В РЕАЛЬНОМ МИРЕ
        </div>
        <div class="row-r">
            <div class="result-left-1">
                <img src="img/result/testimonial_member_2.jpg" alt="res1" >
            </div>
            <div class="result-right-1">
                <div class="result-text">
                    <p class="result-text-1">"Атмосфера в GG Gym потрясающая,
люди еще лучше. В целом, это отличное место для тренировок."</p>
                    <p class="result-text-2">Известно, что читатель будет
отвлечен читаемым содержанием страницы, когда будет рассматривать ее макет.
Lorem Ipsum имеет нормальное распределение букв. Nulla odio nunc, lobortis id
maximus ut, sodales eget erat. Proin sit amet lectus odio nascetur
ridiculus.</p>
                    <p class="result-text-3">Наталя Фоциле</p>
                </div>
            </div>
        </div>
        <div class="row-r">
            <div class="result-left-2">
                <div class="result-text">
                    <p class="result-text-1">"Атмосфера в GG Gym потрясающая,
люди еще лучше. В целом, это отличное место для тренировок."</p>
                    <p class="result-text-2">Известно, что читатель будет
отвлечен читаемым содержанием страницы, когда будет рассматривать ее макет.
Lorem Ipsum имеет нормальное распределение букв. Nulla odio nunc, lobortis id
maximus ut, sodales eget erat. Proin sit amet lectus odio nascetur
ridiculus.</p>
                    <p class="result-text-3">Андрей Владимирович</p>
                </div>
            </div>
            <div class="result-right-2">
                <img src="img/result/testimonial_member_1.jpg" alt="res2" >
            </div>
```

```
        </div>
    </div>
</section>



<section id="goals">
    <div class="container">
        <div class="row row-g">
            <div class="col-md-9 col-sm-6 col-xs-12">
                <div class="goals-text">
                    <p class="goals-text-1">Достигните своих фитнес-целей</p>
                    <p class="goals-text-2">Гибкие пакеты членства,
подходящие для всех уровней тренировок, чтобы помочь вам достичь своих
фитнес-целей</p>
                </div>
            </div>
            <div class="col-md-3 col-sm-6 col-xs-12 text-center">
                <button type="button" class="btn-goals">Присоединяйтесь к
нам</button>
            </div>
        </div>
    </div>
</section>

<section id="ftr">
    <div class="container">
        <div class="row row-f">
            <div class="col-md-3 col-sm-6 col-xs-12 col-f">
                <p class="ftr-title">Приходите к нам</p>
                <div class="ftr-text">
                    <p>Улица,фридреха Энеглься</p>
                    <p>Город, Воронеж</p>
                    <p>Телефон: 8.9081429063</p>
                </div>
            </div>
            <div class="col-md-3 col-sm-6 col-xs-12 col-f">
                <p class="ftr-title">Последние публикации</p>
                <div class="ftr-text">
                    <p class="ftr-bott">Тренируйтесь с весами или собственным
весом?</p>
                    <p class="ftr-bott">Пищевые советы, которые помогут вам
продолжать тренировки</p>
                </div>
            </div>
            <div class="col-md-3 col-sm-6 col-xs-12 col-f">
                <p class="ftr-title">Часы работы</p>
                <div class="ftr-text">
                    <p>Будние дни Понедельник - Пятница</p>
                    <p class="ftr-hours">09:00 - 19:00</p>
                    <p>Выходные Суббота - Воскресенье</p>
                    <p class="ftr-hours">09:00 - 21:00</p>
                </div>
            </div>
            <div class="col-md-3 col-sm-6 col-xs-12 col-f">
                <div class="ftr-img">
```

```
                    <img src="img/result/ftr.jpg" alt="ftr" >
                </div>
            </div>
        </div>
    </div>
</section>


<footer>
    <div class="container">
        <div class="row row-footer">
            <div class="col-md-10 col-sm-6 col-xs-12">
                <p> © Авторские права защищены 2024   | Тема от Якуб Осама  |
Все права защищены   |</p>
            </div>
            <div class="col-md-2 col-sm-6 col-xs-12">
                <div class="foot-social">
                    <i class="fa fa-facebook fa-facebook1" aria-
hidden="true"></i>
                    <i class="fa fa-twitter fa-twitter1" aria-
hidden="true"></i>
                    <i class="fa fa-instagram fa-instagram1" aria-
hidden="true"></i>
                    <div class="up up-opa"></div>
                </div>
            </div>
        </div>
    </div>
</footer>




    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></scr
ipt>
    <script src="js/bootstrap.min.js"></script>
    <script src="js/main.js"></script>
    <script src="js/news.js"></script>
  </body>
</html>



<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
    <link href="css/login.css" rel="stylesheet">
</head>
<body>
<form id="loginForm">
    <h1>Вход</h1>
```

```
    <label for="email">Email:</label><br>
    <input type="email" id="email" name="email"><br><br>
    <button type="submit">Войти</button>
</form>

<script>
    document.getElementById('loginForm').addEventListener('submit',
function(event) {
        event.preventDefault();
        const email = document.getElementById('email').value;
        fetch('/api/members/login', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({ email: email })
        })
            .then(response => response.json())
            .then(data => {
                if (data.token) {
                    sessionStorage.setItem('token', data.token); //
Сохранение токена
                    window.location.href = '/profile.html'; //
Перенаправление
                } else {
                    console.error('Ошибка входа');
                }
            })
            .catch(error => console.error('Ошибка во время входа:', error));
    });
</script>
</body>
</html>



<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>User Profile</title>
    <link href="css/profile.css" rel="stylesheet"></head>
<body>
<form>
    <h1>Профиль</h1>
    <div id="profileInfo">Загрузка...</div>
    <div>
        <label for="trainerSelect">Тренер:</label>
        <select id="trainerSelect"></select>
    </div>
    <div>
        <label for="scheduleSelect">Расписание:</label>
        <select id="scheduleSelect"></select>
    </div>
    <div>
```

```
        <label for="facultySelect">Faculty:</label>
        <select id="facultySelect"></select>
    </div>
    <button onclick="updateProfile()">Обновить Профиль</button>
    <li><a href="index.html" class="scotollt"> Главная странеца </a> </li>
</button>
    <script src="js/profile.js"></script>
    <script>


    const token = sessionStorage.getItem('token');

    window.onload = function() {
        if (!token) {
            window.location.href = '/login.html';  // Redirect to log in if
no token
        } else {
            fetchProfile();
            fetchDropdownData('/api/trainers/all', 'trainerSelect');
            fetchDropdownData('/api/schedules/all', 'scheduleSelect');
            fetchDropdownData('/api/faculties/all', 'facultySelect');
        }
    };

    function fetchProfile() {
        fetch('/api/members/profile', {
            headers: {'Authorization': 'Bearer ' + token}
        })
            .then(response => response.json())
            .then(data => {
                const profileInfo = `<p>Name: ${data.name}</p><p>Email:
${data.email}</p><p>Trainer: ${data.trainerName || 'Not
assigned'}</p><p>Schedule: ${data.scheduleDetails || 'Not
set'}</p><p>Faculty: ${data.facultyName || 'Not set'}</p>`;
                document.getElementById('profileInfo').innerHTML =
profileInfo;
                document.getElementById('trainerSelect').value =
data.trainerId || '';
                document.getElementById('scheduleSelect').value =
data.scheduleId || '';
                document.getElementById('facultySelect').value =
data.facultyId || '';
            })
            .catch(error => console.error('Error fetching profile data:',
error));
    }

    function fetchDropdownData(url, selectId) {
        fetch(url, {
            headers: {'Authorization': 'Bearer ' + token}
        })
            .then(response => response.json())
            .then(data => {
                const select = document.getElementById(selectId);
                select.innerHTML = '';  // Clear previous options
                data.forEach(item => {
                    const option = document.createElement('option');
```

```
                    option.value = item.id;
                    option.textContent = item.name || item.day + ' ' +
item.time; // Adjust based on what property names are returned
                    select.appendChild(option);
                });
            })
            .catch(error => console.error('Error fetching data for:',
selectId, error));
    }

    function updateProfile() {
        const trainerId = document.getElementById('trainerSelect').value;
        const scheduleId = document.getElementById('scheduleSelect').value;
        const facultyId = document.getElementById('facultySelect').value;

        fetch('/api/members/updateProfile', {
            method: 'PUT',
            headers: {
                'Authorization': 'Bearer ' + token,
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({ trainerId, scheduleId, facultyId })
        })
            .then(response => {
                if (response.ok) {
                    fetchProfile();
                    alert('Profile updated successfully!');
                } else {
                    alert('Failed to update profile.');
                }
            })
            .catch(error => console.error('Error updating profile:', error));
    }
</script>
</form>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Регистрация</title>
  <link href="css/register.css" rel="stylesheet">
</head>
<body>

<form id="registerForm">
  <h1>Регистрация</h1>
  <label for="name">Имя:</label><br>
  <input type="text" id="name" name="name"><br>
  <label for="email">Email:</label><br>
  <input type="email" id="email" name="email"><br>
```

```
  <label for="membershipPackage">Пакет членства:</label><br>
  <input type="text" id="membershipPackage" name="membershipPackage"><br><br>
  <button type="submit">Зарегистрироваться</button>
</form>
<script>
  document.getElementById('registerForm').addEventListener('submit',
function(event) {
    event.preventDefault();
    const formData = new FormData(this);
    const data = {};
    formData.forEach((value, key) => {
      data[key] = value;
    });
    fetch('/api/members/register', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify(data)
    })
            .then(response => response.json())
            .then(data => {
              console.log('Успешно зарегистрирован:', data);
              window.location.href = '/login.html'; // Перенаправление на
страницу входа после регистрации
            })
            .catch(error => console.error('Ошибка во время регистрации:',
error));
  });
</script>
</body>
</html>
```