



**CS 353**

**2024-2025 Fall Semester**

**Final Report**

Group 8

Umay Dünder 22202573

Murathan Işık 22103416

Ali Deniz Sözer 22202218

Damla İmre 22001640

Emre Uçar 22203675

# TABLE OF CONTENTS

|  |          |
|--|----------|
| <b>1. Introduction.....</b>                          | <b>3</b> |
| <b>2. Description of The Application.....</b>        | <b>4</b> |
| <b>3. Contribution of Each Member.....</b>           | <b>5</b> |
| 3.1 Emre UÇAR.....                                   | 5        |
| 3.1.1 Proposal Report.....                           | 5        |
| 3.1.2 Design Report.....                             | 5        |
| 3.1.3 Implementation.....                            | 5        |
| 3.1.4 Final Report.....                              | 5        |
| 3.2 Murathan IŞIK.....                               | 5        |
| 3.2.1 Proposal Report.....                           | 5        |
| 3.2.2 Design Report.....                             | 5        |
| 3.2.3 Implementation.....                            | 5        |
| 3.3 Umay DÜNDAR.....                                 | 5        |
| 3.3.1 Proposal Report.....                           | 5        |
| 3.3.2 Design Report.....                             | 5        |
| 3.3.3 Implementation.....                            | 6        |
| 3.3.4 Final Report.....                              | 6        |
| 3.4 Damla İMRE.....                                  | 6        |
| 3.4.1 Proposal Report.....                           | 6        |
| 3.4.2 Design Report.....                             | 6        |
| Table Schemas.....                                   | 6        |
| 3.4.3 Implementation.....                            | 6        |
| 3.4.4 Final Report.....                              | 6        |
| 3.5 Ali Deniz SÖZER.....                             | 6        |
| 3.5.1 Proposal Report.....                           | 6        |
| 3.5.2 Design Report.....                             | 6        |
| 3.5.3 Implementation.....                            | 6        |
| 3.5.4 Final Report.....                              | 6        |
| <b>4. ER Diagram and Database Table Schemas.....</b> | <b>7</b> |
| 4.1 ER Diagram.....                                  | 7        |
| 4.1.1 ER Diagram Changes from Design Report.....     | 7        |
| 4.1.2 Revised ER Diagram.....                        | 8        |
| 4.2 Table Schemas.....                               | 9        |
| 4.2.1 all_users.....                                 | 9        |
| 4.2.2 swimmer.....                                   | 10       |
| 4.2.3 member_swimmer.....                            | 11       |
| 4.2.4 nonmember_swimmer.....                         | 12       |
| 4.2.5 swimming_pool.....                             | 13       |
| 4.2.6 worker.....                                    | 14       |
| 4.2.7 coach.....                                     | 15       |

|   |           |
|---|-----------|
| 4.2.8 lifeguard.....                      | 16        |
| 4.2.9 administrator.....                  | 17        |
| 4.2.10 lane.....                          | 18        |
| 4.2.11 report.....                        | 18        |
| 4.2.12 course.....                        | 20        |
| 4.2.13 course_schedule.....               | 21        |
| 4.2.14 personal_training.....             | 21        |
| 4.2.15 swimming_lesson.....               | 23        |
| 4.2.16 cafe.....                          | 24        |
| 4.2.17 cafe_item.....                     | 25        |
| 4.2.18 rating.....                        | 26        |
| 4.2.19 comment.....                       | 27        |
| 4.2.20 cart.....                          | 28        |
| 4.2.21 private_booking.....               | 29        |
| <b>5. Implementation Details.....</b>     | <b>32</b> |
| 6.1 Constraints:.....                     | 33        |
| <b>7. Users Manual.....</b>               | <b>37</b> |
| 7.3 Register Page.....                    | 38        |
| 7.4 Change Password Page.....             | 39        |
| 7.5 Member Sidebar.....                   | 39        |
| 7.6 Nonmember Sidebar.....                | 40        |
| 7.7 Lifeguard Sidebar.....                | 40        |
| 7.8 Coach Sidebar.....                    | 41        |
| 7.9 Admin Sidebar.....                    | 41        |
| 7.10 Add Balance Pop-Up.....              | 42        |
| 7.11 Profile Page.....                    | 42        |
| 7.12 My Courses Page.....                 | 43        |
| 7.13 Upcoming Bookings Page.....          | 44        |
| 7.14 Member/Non Member Schedule Page..... | 44        |
| 7.15 All Courses Page.....                | 45        |
| 7.16 Book Pool Lane Page.....             | 46        |
| 7.17 My Cart Page.....                    | 47        |
| 7.18 Cafe Page.....                       | 47        |
| 7.19 Become a Member Page.....            | 48        |
| 7.20 Create Course Page.....              | 48        |
| 7.21 Coach Schedule Page.....             | 49        |
| 7.22 View Courses Page.....               | 49        |
| 7.23 Withdraw Money Page.....             | 50        |
| 7.24 Work Hours Page.....                 | 50        |
| 7.25 Upcoming Hours Page.....             | 51        |
| 7.26 Manage Users Page.....               | 51        |
| 7.27 Manage Courses Page.....             | 51        |
| 7.28 Manage Pool Bookings Pages.....      | 52        |
| 7.29 Reports Page.....                    | 52        |

|                           |           |
|---------------------------|-----------|
| <b>8. References.....</b> | <b>53</b> |
|---------------------------|-----------|

# **1. Introduction**

Many use public pools, just like any facility that works with an entry registration system. Many people are registered to a pool system: Regulars, irregulars, people working in the facility, and admins. Managing a pool comes with many responsibilities, varying from registering users to the system, managing bookings of the lanes and pools, scheduling lessons, and generating reports to see details of this information. However, these attributes are hard to manage manually, as many active pool systems are currently used, posing a challenge. These challenges highlight the need for a robust, automated system to streamline pool activity management, helping users and administrators achieve smooth operations.

## **2. Description of The Application**

In this project, we will create a pool facility system that serves many different users, such as member and nonmember swimmers, pool administrators, lifeguards, and coaches. In this application, all the swimmers can register for pool access and select lanes, book sessions, and register for swimming classes according to their level. Swimmers can also gain points that can be used to buy some items if they have a membership and they keep booking for the pool or registering for swimming classes. Swimmers can see their rating ranking among other swimmers in the system. Every swimmer can rate the swimming class coaches' instructor and see the coaches' overall rating. They can also comment on the swimming coaches' past performance. Swimmers and swimming coaches can see their swimming lesson schedules and available/unavailable daily time slots for accessing pools. Coaches can also see their ratings in the system. Administrators should be able to see coach ratings, feedback, swimmer points and rankings, available courses, and their course capacities. Admins can also generate reports for the demand of the pools with respect to their hours, most active members, and coach performances.

In this project, database usage is crucial since our application requires storing many different user types, their profile and lesson information and other necessary information like comments about coaches and courses or rating systems for member swimmers. Therefore, a database system is needed to handle all these data management tasks, including insertions, deletions, and modifications, according to the interaction of the user with the frontend part of our system.

## **3. Contribution of Each Member**

### **3.1 Emre UÇAR**

#### **3.1.1 Proposal Report**

- Have implemented introductions, limitations and partly ER diagram.
- Overall worked on the organization of the paper, which includes titles, spacing, justifying.

#### **3.1.2 Design Report**

- Redesigned ER diagram.
- Overall worked on the organization of the paper, which includes titles, spacing, justifying.

#### **3.1.3 Implementation**

- Have designed frontend first.
- Implemented the business logic of the project in general.
- Added Django views and connected the front and backend of chiefly swimmer user pages.

#### **3.1.4 Final Report**

- Revised ER diagram according to our implementation of the project and the feedback from the design report.
- Overall worked on the organization of the paper, which includes titles, spacing, justifying.

### **3.2 Murathan IŞIK**

#### **3.2.1 Proposal Report**

- Helped other group members to draw first E/R design diagram

#### **3.2.2 Design Report**

- Helped others to revise E/R design diagram and UI design in Canva

#### **3.2.3 Implementation**

- I worked on the frontend design and connected the frontend to the backend for functionality on some pages, especially for admin pages for managing the database.

### **3.3 Umay DÜNDAR**

#### **3.3.1 Proposal Report**

- I helped other group members to draw an E/R Design diagram.

- I wrote the functional requirements and limitations part of the proposal report.

### **3.3.2 Design Report**

- I helped other group members to draw an E/R Design diagram.
- I created the UI design of the pages using Canva.
- I wrote the schema definitions according to our current database tables.

### **3.3.3 Implementation**

- I mainly worked on the backend part of the project, and I worked on the connections between the front and back parts of the project. I wrote and fixed some axios queries of other frontend pages.

### **3.3.4 Final Report**

- I mostly worked on implementation details and the user manual part of the final report.

## **3.4 Damla İMRE**

### **3.4.1 Proposal Report**

- Non-functional Requirements
- Functional Requirements
- E/R Design

### **3.4.2 Design Report**

- E/R Design
- Table Schemas

### **3.4.3 Implementation**

- I worked on the backend part of the project. I implemented some backend functions and worked on the connections between frontend and backend of the project.

### **3.4.4 Final Report**

- Table Schemas

## **3.5 Ali Deniz SÖZER**

### **3.5.1 Proposal Report**

- I helped with creating the initial E/R diagram

### **3.5.2 Design Report**

- I helped with revising the E/R diagram according to the feedback we received



### **3.5.3 Implementation**

- I worked on the backend-frontend connections of the project. Mainly, I worked on the register, login and coach-related pages. I also worked on fixing the backend issues and adding new features.

### **3.5.4 Final Report**

- I added the table schemas and created the user manual.

## 4. ER Diagram and Database Table Schemas

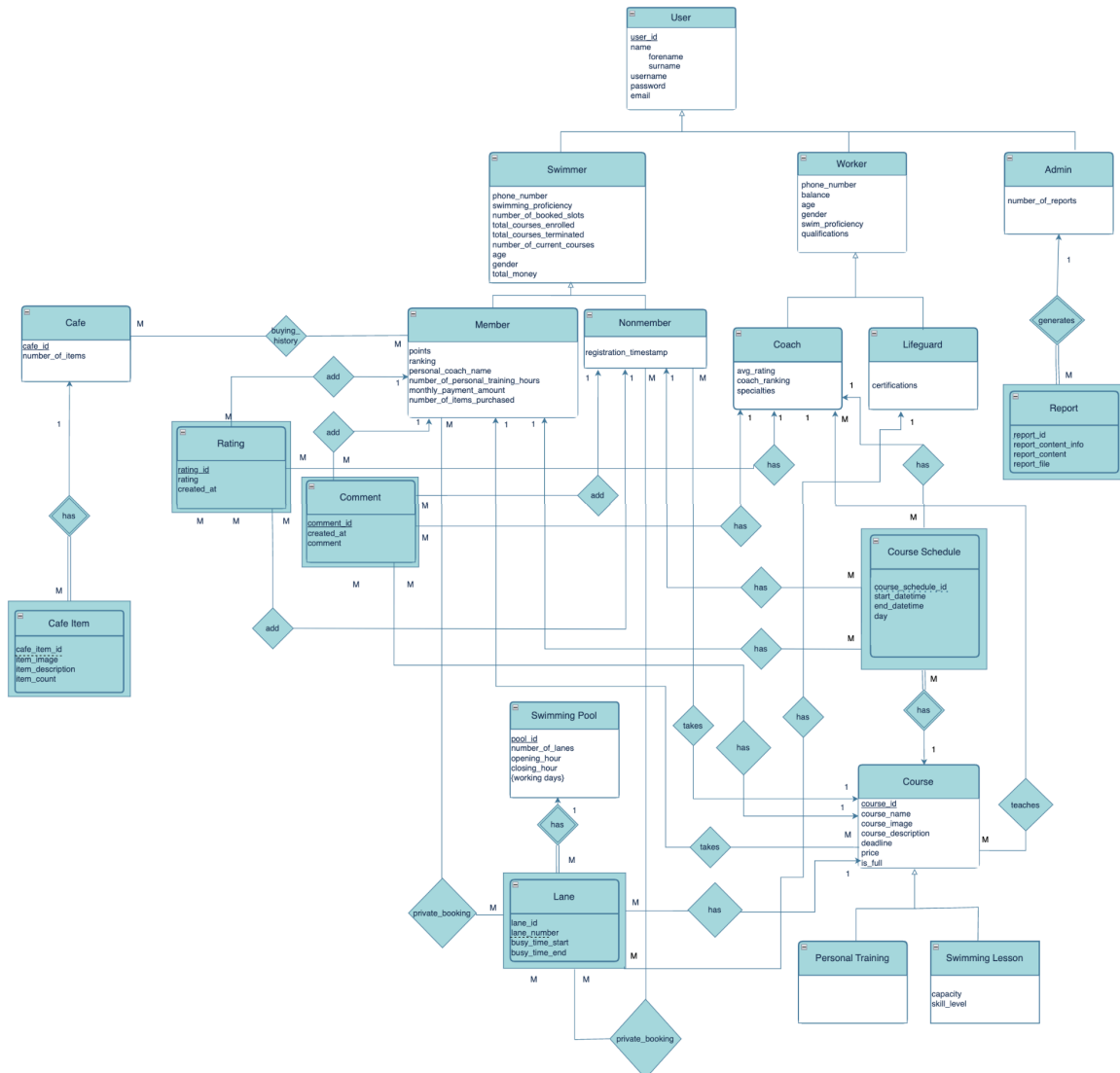
### 4.1 ER Diagram

#### 4.1.1 ER Diagram Changes from Design Report

Here is a list of changes in our diagram from the previous version

- We decided to delete the time slot entity as it is redundant and storing a DATETIME attribute in the lane weak entity.
- *phone\_number* attribute in Swimmer and Worker users are changed to single-valued attribute from multivariabled attribute.
- Worker's *qualifications* attribute is no longer a multivariabled attribute, changed to single-valued.
- Lifeguard's *certifications* attribute is no longer a multivariabled attribute, changed to single-valued.
- Rating and command entities are weak entities now.

## 4.1.2 Revised ER Diagram<sup>1</sup>



<sup>1</sup> [https://drive.google.com/file/d/19T17ddcRAOtJJUwzh8Hsv1P1V\\_jbrDnp/view?usp=sharing](https://drive.google.com/file/d/19T17ddcRAOtJJUwzh8Hsv1P1V_jbrDnp/view?usp=sharing)

## 4.2 Table Schemas

### 4.2.1 all\_users

**Table Name:** all\_users

**Relational Model:** all\_users(user\_id, user\_image, forename, surname, username, password, user\_type, email)

**Functional Dependencies:** user\_id -> user\_image, forename, surname, username, password, user\_type, email

**Candidate Keys:** {user\_id}

**Primary Key:** user\_id

**Foreign Keys:** none

**Normal Form:** BCNF

**Table Definition:**

```
CREATE TABLE all_users (  
    user_id SERIAL PRIMARY KEY,  
    user_image BYTEA,  
    forename VARCHAR(255),  
    surname VARCHAR(255),  
    username VARCHAR(255),  
    password VARCHAR(255),  
    user_type VARCHAR(255),  
    email VARCHAR(255) NOT NULL  
);
```

### 4.2.2 swimmer

**Table Name:** swimmer

**Relational Model:** swimmer(swimmer\_id, phone\_number, age, gender, swimming\_proficiency, number\_of\_booked\_slots, total\_courses\_enrolled, total\_courses\_terminated, membership\_status, total\_money)

**Functional Dependencies:** swimmer\_id → phone\_number, age, gender, swimming\_proficiency, number\_of\_booked\_slots, total\_courses\_enrolled, total\_courses\_terminated, membership\_status, total\_money

**Candidate Keys:** {swimmer\_id}

**Primary Key:** swimmer\_id

**Foreign Keys:** swimmer\_id → all\_users(user\_id)

**Normal Form:** BCNF

**Table Definition:**

```
CREATE TABLE swimmer (  
    swimmer_id SERIAL PRIMARY KEY,  
    phone_number VARCHAR(15),  
    age INT,  
    gender VARCHAR(100),  
    swimming_proficiency VARCHAR(100),  
    number_of_booked_slots INT,  
    total_courses_enrolled INT,  
    total_courses_terminated INT,  
    membership_status VARCHAR(255),  
    total_money INT,  
    FOREIGN KEY (swimmer_id) REFERENCES all_users(user_id),  
    check(swimming_proficiency in ('Beginner', 'Intermediate', 'Advanced'))  
);
```

### 4.2.3 member\_swimmer

**Table Name:** member\_swimmer

**Relational Model:** member\_swimmer(swimmer\_id, points, monthly\_payment\_amount, number\_of\_personal\_training\_hours, ranking, number\_of\_items\_purchased, personal\_coach\_id)

**Functional Dependencies:** swimmer\_id → points, monthly\_payment\_amount, number\_of\_personal\_training\_hours, ranking, number\_of\_items\_purchased, personal\_coach\_id

**Candidate Keys:** {swimmer\_id}

**Primary Key:** swimmer\_id

**Foreign Keys:** swimmer\_id → swimmer(swimmer\_id)  
personal\_coach\_id → coach(coach\_id)

**Normal Form:** BCNF

**Table Definition:**

```
CREATE TABLE member_swimmer(  
    swimmer_id SERIAL PRIMARY KEY,  
    points INT,  
    monthly_payment_amount INT,  
    number_of_personal_training_hours INT,  
    ranking INT,  
    number_of_items_purchased INT,  
    personal_coach_id INT,  
    FOREIGN KEY (swimmer_id) REFERENCES swimmer(swimmer_id),  
    FOREIGN KEY (personal_coach_id) REFERENCES coach(coach_id)  
);
```

#### 4.2.4 nonmember\_swimmer

**Table Name:** nonmember\_swimmer

**Relational Model:** nonmember\_swimmer(swimmer\_id, registration\_timestamp)

**Functional Dependencies:** swimmer\_id  $\rightarrow$  registration\_timestamp

**Candidate Keys:** {swimmer\_id}, {registration\_timestamp}

**Primary Key:** {swimmer\_id}

**Foreign Keys:** swimmer\_id  $\rightarrow$  swimmer(swimmer\_id)

**Normal Form:** BCNF

**Table Definition:**

```
CREATE TABLE nonmember_swimmer (  
    swimmer_id SERIAL PRIMARY KEY,  
    registration_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
    FOREIGN KEY (swimmer_id) REFERENCES swimmer(swimmer_id)  
);
```

### 4.2.5 swimming\_pool

**Table Name:** swimming\_pool

**Relational Model:** swimming\_pool(pool\_id, number\_of\_lanes, opening\_hour, closing\_hour, working\_days, location)

**Functional Dependencies:** pool\_id  $\rightarrow$  number\_of\_lanes, opening\_hour, closing\_hour, working\_days, location

**Candidate Keys:** {pool\_id}

**Primary Key:** pool\_id

**Foreign Keys:**

**Normal Form:** BCNF

**Table Definition:**

```
CREATE TABLE swimming_pool (  
    pool_id SERIAL PRIMARY KEY,  
    number_of_lanes INT NOT NULL,  
    opening_hour TIME NOT NULL,  
    closing_hour TIME NOT NULL,  
    working_days TEXT NOT NULL,  
    location TEXT  
);
```



## 4.2.6 worker

**Table Name:** worker

**Relational Model:** worker(worker\_id, pool\_id, age, gender, phone\_number, qualifications, balance)

**Functional Dependencies:** worker\_id  $\rightarrow$  pool\_id, age, gender, phone\_number, qualifications, balance

**Candidate Keys:** {worker\_id}

**Primary Key:** worker\_id

**Foreign Keys:** worker\_id  $\rightarrow$  all\_users(user\_id)  
pool\_id  $\rightarrow$  swimming\_pool(pool\_id)

**Normal Form:** BCNF

### Table Definition:

```
CREATE TABLE worker (  
    worker_id SERIAL PRIMARY KEY,  
    pool_id INT,  
    age INT,  
    gender VARCHAR(100),  
    phone_number VARCHAR(15),  
    qualifications TEXT,  
    balance INT,  
    FOREIGN KEY (worker_id) REFERENCES all_users(user_id),  
    FOREIGN KEY (pool_id) REFERENCES swimming_pool(pool_id),  
    check(gender in ('Male', 'Female'))  
);
```

### 4.2.7 coach

**Table Name:** coach

**Relational Model:** coach(coach\_id, avg\_rating, coach\_ranking, specialties)

**Functional Dependencies:** coach\_id  $\rightarrow$  avg\_rating, coach\_ranking, specialties

**Candidate Keys:** {coach\_id}

**Primary Key:** coach\_id

**Foreign Keys:** coach\_id  $\rightarrow$  worker(worker\_id)

**Normal Form:** BCNF

**Table Definition:**

```
CREATE TABLE coach (  
    coach_id SERIAL PRIMARY KEY,  
    avg_rating FLOAT,  
    coach_ranking INT,  
    specialties TEXT,  
    FOREIGN KEY (coach_id) REFERENCES worker(worker_id)  
);
```

### 4.2.8 lifeguard

**Table Name:** lifeguard

**Relational Model:** lifeguard(lifeguard\_id, certifications)

**Functional Dependencies:** lifeguard\_id  $\rightarrow$  certifications

**Candidate Keys:** {lifeguard\_id}

**Primary Key:** lifeguard\_id

**Foreign Keys:** lifeguard\_id  $\rightarrow$  worker(worker\_id)

**Normal Form:** BCNF

**Table Definition:**

```
CREATE TABLE lifeguard (  
    lifeguard_id SERIAL PRIMARY KEY,  
    certifications TEXT,  
    FOREIGN KEY (lifeguard_id) REFERENCES worker(worker_id)  
);
```

### 4.2.9 administrator

**Table Name:** administrator

**Relational Model:** administrator(administrator\_id, number\_of\_reports)

**Functional Dependencies:** administrator\_id  $\rightarrow$  number\_of\_reports

**Candidate Keys:** {administrator\_id}

**Primary Key:** administrator\_id

**Foreign Keys:** administrator\_id  $\rightarrow$  all\_users(user\_id)

**Normal Form:** BCNF

**Table Definition:**

```
CREATE TABLE administrator (  
    administrator_id SERIAL PRIMARY KEY,  
    number_of_reports INT,  
    FOREIGN KEY (administrator_id) REFERENCES all_users(user_id)  
);
```

### 4.2.10 lane

**Table Name:** lane

**Relational Model:** lane(lane\_id, pool\_id, lane\_number, lifeguard\_id, start\_time, end\_time, booking\_price, start\_date, end\_date, availability)

**Functional Dependencies:** lane\_id  $\rightarrow$  pool\_id, lane\_number, lifeguard\_id, start\_time, end\_time, booking\_price, start\_date, end\_date, availability

**Candidate Keys:** {lane\_id}

**Primary Key:** lane\_id

**Foreign Keys:** pool\_id  $\rightarrow$  swimming\_pool(pool\_id)  
lifeguard\_id  $\rightarrow$  lifeguard(lifeguard\_id)

**Normal Form:** BCNF

**Table Definition:**

```
CREATE TABLE lane(  
    lane_id SERIAL,  
    pool_id INT,  
    lane_number INT,  
    lifeguard_id INT,  
    start_time TIME,  
    end_time TIME,  
    booking_price INT,  
    start_date DATE,  
    end_date DATE,  
    availability VARCHAR(255),  
    FOREIGN KEY (pool_id) REFERENCES swimming_pool(pool_id),  
    FOREIGN KEY (lifeguard_id) REFERENCES lifeguard(lifeguard_id),  
    UNIQUE (lane_id),  
    PRIMARY KEY (lane_id, pool_id),  
    CHECK (availability IN ('available', 'added-to-cart', 'in-use'))  
);
```

### 4.2.11 report

**Table Name:** report

**Relational Model:** report(report\_id, admin\_id, report\_content\_info, report\_content, report\_file)

**Functional Dependencies:** report\_id  $\rightarrow$  admin\_id, report\_content\_info, report\_content, report\_file

**Candidate Keys:** {report\_id}

**Primary Key:** report\_id

**Foreign Keys:** admin\_id  $\rightarrow$  administration(admin\_id)

**Normal Form:** BCNF

**Table Definition:**

```
CREATE TABLE report (  
    report_id SERIAL PRIMARY KEY,  
    admin_id INT,  
    report_content_info VARCHAR(255),  
    report_content TEXT,  
    report_file BYTEA,  
    FOREIGN KEY (admin_id) REFERENCES administrator(administrator_id)  
);
```

### 4.2.12 course

**Table Name:** course

**Relational Model:** course(course\_id, course\_name, coach\_id, course\_description, date, start\_time, end\_time, restrictions, pool\_id, lane\_id, price, capacity)

**Functional Dependencies:** course\_id  $\rightarrow$  course\_name, coach\_id, course\_description, date, start\_time, end\_time, restrictions, pool\_id, lane\_id, price, capacity

**Candidate Keys:** {course\_id }

**Primary Key:** course\_id

**Foreign Keys:** coach\_id  $\rightarrow$  coach(coach\_id)

**Normal Form:** BCNF

**Table Definition:**

```
CREATE TABLE course (  
    course_id SERIAL PRIMARY KEY,  
    course_name VARCHAR(255) NOT NULL,  
    coach_id INT NOT NULL,  
    course_description TEXT,  
    date DATE NOT NULL,  
    start_time TIME NOT NULL,  
    end_time TIME NOT NULL,  
    restrictions VARCHAR(255),  
    pool_id INT NOT NULL,  
    lane_id INT NOT NULL,  
    price INT NOT NULL,  
    capacity INT NOT NULL,  
    FOREIGN KEY (coach_id) REFERENCES coach(coach_id)  
);
```

### 4.2.13 course\_schedule

**Table Name:** course\_schedule

**Relational Model:** course(course\_schedule\_id, course\_id, swimmer\_id, coach\_id, start\_time, end\_time, status, day)

**Functional Dependencies:** course\_schedule\_id → course\_id, swimmer\_id, coach\_id, start\_time, end\_time, status, day

**Candidate Keys:** {course\_schedule\_id}

**Primary Key:** course\_schedule\_id, course\_id

**Foreign Keys:** coach\_id → coach(coach\_id)  
swimmer\_id → swimmer(swimmer\_id)  
course\_id → course(course\_id)

**Normal Form:** BCNF

#### Table Definition:

```
CREATE TABLE course_schedule (  
    course_schedule_id SERIAL,  
    course_id INT,  
    swimmer_id INT,  
    coach_id INT,  
    start_time TIME NOT NULL,  
    end_time TIME NOT NULL,  
    status TEXT,  
    day TEXT NOT NULL,  
    FOREIGN KEY (swimmer_id) REFERENCES swimmer(swimmer_id),  
    FOREIGN KEY (coach_id) REFERENCES coach(coach_id),  
    FOREIGN KEY (course_id) REFERENCES course(course_id),  
    PRIMARY KEY (course_schedule_id, course_id),  
    CHECK (status IN ('not-enrolled', 'in-progress', 'withdrawn', 'finished', 'cancelled')),  
    CHECK (day IN ('Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',  
    'Saturday'))  
);
```

### 4.2.14 personal\_training

**Table Name:** personal\_training



**Relational Model:** personal\_training(training\_id)

**Functional Dependencies:** training\_id  $\rightarrow$  training\_id

**Candidate Keys:** {training\_id}

**Primary Key:** training\_id

**Foreign Keys:** training\_id  $\rightarrow$  course(course\_id)

**Normal Form:** BCNF

**Table Definition:**

```
CREATE TABLE personal_training (  
    training_id SERIAL PRIMARY KEY,  
    FOREIGN KEY (training_id) REFERENCES course(course_id)  
);
```

#### 4.2.15 swimming\_lesson

**Table Name:** swimming\_lesson

**Relational Model:** swimming\_lesson(lesson\_id, capacity, is\_full, skill\_level)

**Functional Dependencies:** lesson\_id  $\rightarrow$  capacity, is\_full, skill\_level

**Candidate Keys:** {lesson\_id}

**Primary Key:** lesson\_id

**Foreign Keys:** lesson\_id  $\rightarrow$  course(course\_id)

**Normal Form:** BCNF

**Table Definition:**

```
CREATE TABLE swimming_lesson (  
    lesson_id SERIAL PRIMARY KEY,  
    capacity INT NOT NULL,  
    is_full BOOLEAN NOT NULL,  
    skill_level TEXT,  
    FOREIGN KEY (lesson_id) REFERENCES course(course_id),  
    CHECK (skill_level IN ('beginner', 'intermediate', 'advanced'))  
);
```

#### 4.2.16 cafe

**Table Name:** cafe

**Relational Model:** cafe(cafe\_id, number\_of\_items, pool\_id)

**Functional Dependencies:**  $\text{cafe\_id} \rightarrow \text{number\_of\_items}, \text{pool\_id}$

**Candidate Keys:** {cafe\_id}

**Primary Key:** cafe\_id

**Foreign Keys:**  $\text{pool\_id} \rightarrow \text{swimming\_pool}(\text{pool\_id})$

**Normal Form:** BCNF

**Table Definition:**

```
CREATE TABLE cafe (  
    cafe_id SERIAL PRIMARY KEY,  
    number_of_items INT,  
    pool_id INT,  
    FOREIGN KEY (pool_id) REFERENCES swimming_pool(pool_id)  
);
```

### 4.2.17 cafe\_item

**Table Name:** cafe\_item

**Relational Model:** cafe\_item(cafe\_item\_id, cafe\_id, item\_image, item\_name, item\_description, item\_count, price)

**Functional Dependencies:** cafe\_item\_id, cafe\_id → item\_image, item\_name, item\_description, item\_count, price

**Candidate Keys:** {(cafe\_item\_id, cafe\_id)}

**Primary Key:** cafe\_item\_id, cafe\_id

**Foreign Keys:** cafe\_id → cafe(cafe\_id)

**Normal Form:** BCNF

**Table Definition:**

```
CREATE TABLE cafe_item (  
    cafe_item_id SERIAL,  
    cafe_id INT NOT NULL,  
    item_image BYTEA,  
    item_name VARCHAR(255),  
    item_description VARCHAR(255),  
    item_count INT,  
    price INT,  
    FOREIGN KEY (cafe_id) REFERENCES cafe(cafe_id),  
    UNIQUE(cafe_item_id),  
    PRIMARY KEY (cafe_item_id, cafe_id)  
);
```

### 4.2.18 rating

**Table Name:** rating

**Relational Model:** rating(rating\_id, course\_id, swimmer\_id, coach\_id, rating, created\_at)

**Functional Dependencies:** rating\_id, course\_id  $\rightarrow$  swimmer\_id, coach\_id, rating, created\_at

**Candidate Keys:** {(rating\_id, course\_id)}

**Primary Key:** rating\_id, course\_id

**Foreign Keys:** swimmer\_id  $\rightarrow$  swimmer(swimmer\_id)  
coach\_id  $\rightarrow$  coach(coach\_id)  
course\_id  $\rightarrow$  course(course\_id)

**Normal Form:** BCNF

**Table Definition:**

```
CREATE TABLE rating(  
    rating_id SERIAL,  
    swimmer_id INT NOT NULL,  
    coach_id INT NOT NULL,  
    course_id INT NOT NULL,  
    rating INT CHECK (rating BETWEEN 1 AND 5),  
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
    FOREIGN KEY (swimmer_id) REFERENCES swimmer(swimmer_id),  
    FOREIGN KEY (coach_id) REFERENCES coach(coach_id),  
    FOREIGN KEY (course_id) REFERENCES course(course_id),  
    PRIMARY KEY (rating_id, course_id)  
);
```

## 4.2.19 comment

**Table Name:** comment

**Relational Model:** comment(comment\_id, course\_id, swimmer\_id, coach\_id, comment, created\_at)

**Functional Dependencies:** comment\_id, course\_id  $\rightarrow$  swimmer\_id, coach\_id, comment, created\_at

**Candidate Keys:** {(comment\_id, course\_id)}

**Primary Key:** comment\_id, course\_id

**Foreign Keys:** swimmer\_id  $\rightarrow$  swimmer(swimmer\_id)  
coach\_id  $\rightarrow$  coach(coach\_id)  
course\_id  $\rightarrow$  course(course\_id)

**Normal Form:** BCNF

### Table Definition:

```
CREATE TABLE comment(  
    comment_id SERIAL,  
    swimmer_id INT NOT NULL,  
    coach_id INT NOT NULL,  
    course_id INT NOT NULL,  
    comment VARCHAR(255),  
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
    FOREIGN KEY (swimmer_id) REFERENCES swimmer(swimmer_id),  
    FOREIGN KEY (coach_id) REFERENCES coach(coach_id),  
    FOREIGN KEY (course_id) REFERENCES course(course_id),  
    PRIMARY KEY (comment_id, course_id)  
);
```

#### 4.2.20 cart

**Table Name:** cart

**Relational Model:** cart(cart\_id, purchaser\_id, course\_id, cafe\_item\_id, cafe\_id, lane\_id)

**Functional Dependencies:** comment\_id, course\_id  $\rightarrow$  swimmer\_id, coach\_id,  
comment, created\_at

**Candidate Keys:** {(comment\_id, course\_id)}

**Primary Key:** comment\_id, course\_id

**Foreign Keys:** swimmer\_id  $\rightarrow$  swimmer(swimmer\_id)  
coach\_id  $\rightarrow$  coach(coach\_id)  
course\_id  $\rightarrow$  coach(course\_id)

**Normal Form:** BCNF

**Table Definition:**

```
CREATE TABLE cart (  
    cart_id SERIAL PRIMARY KEY,  
    purchaser_id INT NOT NULL,  
    course_id INT,  
    cafe_item_id INT,  
    cafe_id INT,  
    lane_id INT,  
    FOREIGN KEY (purchaser_id) REFERENCES swimmer(swimmer_id),  
    FOREIGN KEY (course_id) REFERENCES course(course_id),  
    FOREIGN KEY (cafe_item_id) REFERENCES cafe_item(cafe_item_id),  
    FOREIGN KEY (cafe_id) REFERENCES cafe(cafe_id),  
    FOREIGN KEY (lane_id) REFERENCES lane(lane_id)  
);
```

#### 4.2.21 private\_booking

**Table Name:** private\_booking

**Relational Model:** private\_booking(private\_booking\_id, swimmer\_id, lane\_id, booking\_date, start\_time, end\_time, status)

**Functional Dependencies:** (private\_booking\_id, swimmer\_id)  
-> lane\_id, booking\_date, start\_time, end\_time, status

**Candidate Keys:** { (private\_booking\_id, swimmer\_id) }

**Primary Key:**(private\_booking\_id, swimmer\_id)

**Foreign Keys:** swimmer\_id → swimmer(swimmer\_id)  
lane\_id → lane(lane\_id)

**Normal Form:** BCNF

#### Table Definition:

```
CREATE TABLE private_booking (  
    private_booking_id SERIAL,  
    swimmer_id INT,  
    lane_id INT,  
    booking_date DATE NOT NULL,  
    start_time TIME,  
    end_time TIME,  
    status VARCHAR(50) DEFAULT 'active',  
    FOREIGN KEY (swimmer_id) REFERENCES swimmer(swimmer_id),  
    FOREIGN KEY (lane_id) REFERENCES lane(lane_id),  
    UNIQUE (swimmer_id, lane_id, start_time, end_time),  
    PRIMARY KEY (private_booking_id, swimmer_id)  
);
```



#### 4.2.22 teaches

**Table Name:** teaches

**Relational Model:** teaches(teaches\_id, lesson\_id, coach\_id)

**Functional Dependencies:** teaches\_id  $\rightarrow$  lesson\_id, coach\_id

**Candidate Keys:** {teaches\_id}

**Primary Key:** teaches\_id

**Foreign Keys:** lesson\_id  $\rightarrow$  swimming\_lesson(lesson\_id)  
coach\_id  $\rightarrow$  coach(coach\_id)

**Normal Form:** BCNF

**Table Definition:**

```
CREATE TABLE teaches (  
    teaches_id SERIAL PRIMARY KEY,  
    lesson_id INT,  
    coach_id INT,  
    FOREIGN KEY (lesson_id) REFERENCES swimming_lesson(lesson_id),  
    FOREIGN KEY (coach_id) REFERENCES coach(coach_id)  
);
```

### 4.2.23 buying\_history

**Table Name:** buying\_history

**Relational Model:** buying\_history(history\_id, purchaser\_id, course\_id, cafe\_item\_id, cafe\_id, lane\_id)

**Functional Dependencies:** history\_id  $\rightarrow$  purchaser\_id, course\_id, cafe\_item\_id, cafe\_id, lane\_id

**Candidate Keys:** {history\_id}

**Primary Key:** history\_id

**Foreign Keys:** purchaser\_id  $\rightarrow$  swimmer(swimmer\_id)  
course\_id  $\rightarrow$  course(course\_id)  
cafe\_item\_id  $\rightarrow$  cafe\_item(cafe\_item\_id)  
cafe\_id  $\rightarrow$  cafe(cafe\_id)  
lane\_id  $\rightarrow$  lane(lane\_id)

**Normal Form:** BCNF

**Table Definition:**

```
CREATE TABLE buying_history (  
    history_id SERIAL PRIMARY KEY,  
    purchaser_id INT NOT NULL,  
    course_id INT,  
    cafe_item_id INT,  
    cafe_id INT,  
    lane_id INT,  
    purchased_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
    FOREIGN KEY (purchaser_id) REFERENCES swimmer(swimmer_id),  
    FOREIGN KEY (course_id) REFERENCES course(course_id),  
    FOREIGN KEY (cafe_item_id) REFERENCES cafe_item(cafe_item_id),  
    FOREIGN KEY (cafe_id) REFERENCES cafe(cafe_id),  
    FOREIGN KEY (lane_id) REFERENCES lane(lane_id)  
);
```

## 5. Implementation Details

This project implementation is based on **React** for the frontend, **Django** for the backend, and **PostgreSQL** for the database. Our project's backend runs at port 8000, while our project's frontend runs from port 3000 and our database server runs on port 5432. Between these two components, we send queries by using Axios get and post requests, that return JSON responses. To configure these, we changed the database settings in django backend applications settings.py file. To create tables, we create management commands and run `python manage.py create_tables`. In `python manage.py create_tables`, we execute the `create_tables.sql` file, which contains table definitions. Nevertheless, to drop tables, we also create the `delete_tables` management command, which runs the `delete_tables.sql` file through Python code and drops all tables. We use Django views and `cursor.execute()` to run raw SQL queries in order to manipulate tables. Our queries contain select, aggregation functions, group by, cases, join, insert, delete, and update statements.

## 6.1 Constraints:

We use check statements in our SQL table definitions and make sure the right value is inserted into our tables. Also, we use NOT NULL and UNIQUE constraints to make sure

Example SQL check statements from our source code ( NOT NULL, UNIQUE and CHECK statements are marked as bold)

```
CREATE TABLE worker (  
    worker_id SERIAL PRIMARY KEY,  
    pool_id INT,  
    age INT,  
    gender VARCHAR(100),  
    phone_number VARCHAR(15),  
    qualifications TEXT,  
    balance INT,  
    FOREIGN KEY (worker_id) REFERENCES all_users(user_id),  
    FOREIGN KEY (pool_id) REFERENCES swimming_pool(pool_id),  
    check(gender in ('Male', 'Female'))  
);  
  
CREATE SEQUENCE lane_id_seq START 1;  
  
CREATE TABLE lane(  
    lane_id INT DEFAULT nextval('lane_id_seq'),  
    pool_id INT,  
    lane_number INT,  
    lifeguard_id INT,  
    start_time TIME,  
    end_time TIME,  
    booking_price INT,  
    start_date DATE,
```

```

end_date DATE,
availability VARCHAR(255),
FOREIGN KEY (pool_id) REFERENCES swimming_pool(pool_id),
FOREIGN KEY (lifeguard_id) REFERENCES lifeguard(lifeguard_id),
UNIQUE (lane_id),
PRIMARY KEY (lane_id, pool_id),
CHECK (availability IN ('available', 'added-to-cart', 'in-use'))
);

```

```

CREATE TABLE swimmer (
    swimmer_id SERIAL PRIMARY KEY,
    phone_number VARCHAR(15),
    age INT,
    gender VARCHAR(100),
    swimming_proficiency VARCHAR(100),
    number_of_booked_slots INT,
    total_courses_enrolled INT,
    total_courses_terminated INT,
    membership_status VARCHAR(255),
    total_money INT,
    FOREIGN KEY (swimmer_id) REFERENCES all_users(user_id),
    check(swimming_proficiency in ('Beginner', 'Intermediate', 'Advanced'))
);

```

```

CREATE TABLE course (
    course_id SERIAL PRIMARY KEY,
    course_name VARCHAR(255) NOT NULL,

```

```
coach_id INT NOT NULL,
course_description TEXT,
date DATE NOT NULL,
start_time TIME NOT NULL,
end_time TIME NOT NULL,
restrictions VARCHAR(255),
pool_id INT NOT NULL,
lane_id INT NOT NULL,
price INT NOT NULL,
capacity INT NOT NULL,
FOREIGN KEY (coach_id) REFERENCES coach(coach_id)
);
```

```
CREATE TABLE course_schedule(
course_schedule_id SERIAL,
course_id INT,
swimmer_id INT,
coach_id INT,
start_time TIME NOT NULL,
end_time TIME NOT NULL,
status VARCHAR(255),
day VARCHAR(255),
FOREIGN KEY (swimmer_id) REFERENCES swimmer(swimmer_id),
FOREIGN KEY (coach_id) REFERENCES coach(coach_id),
FOREIGN KEY (course_id) REFERENCES course(course_id),
PRIMARY KEY (course_schedule_id, course_id),
```

```

    CHECK (status IN ('not-enrolled', 'in-progress', 'withdrawn', 'finished', 'cancelled'))
);

CREATE TABLE swimming_lesson (
    lesson_id SERIAL PRIMARY KEY,
    capacity INT NOT NULL,
    is_full BOOLEAN NOT NULL,
    skill_level TEXT,
    FOREIGN KEY (lesson_id) REFERENCES course(course_id),
    CHECK (skill_level IN ('beginner', 'intermediate', 'advanced'))
);

CREATE TABLE private_booking (
    private_booking_id SERIAL,
    swimmer_id INT,
    lane_id INT,
    booking_date DATE NOT NULL,
    start_time TIME,
    end_time TIME,
    status VARCHAR(50) DEFAULT 'active',
    FOREIGN KEY (swimmer_id) REFERENCES swimmer(swimmer_id),
    FOREIGN KEY (lane_id) REFERENCES lane(lane_id),
    UNIQUE (swimmer_id, lane_id, start_time, end_time),
    PRIMARY KEY (private_booking_id, swimmer_id)
);

```

## 7. Users Manual

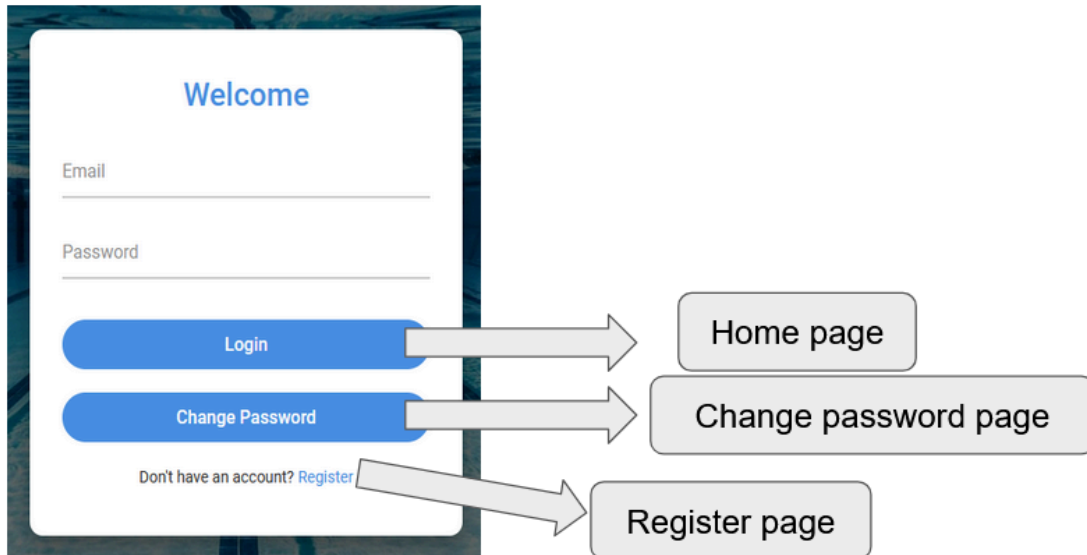
### 7.1 Build Instructions

- You can clone our repository manually through the command line or by using GitHub Desktop or gitKraken in your local folder.
- After cloning the repository, create a python venv. virtual environment using `python -m venv venv` command
- Activate your virtual environment using `source venv/bin/activate` # On Windows, use `venv\Scripts\activate`
- Make sure you have pip, python, Django, django-rest-framework, psycopg or psycopg2-binary installed in your virtual environment.
- If these dependencies are not installed, you can install them using `pip install` command.
- Then open the frontend folder of the project and make sure you have npm and npx on your environment.
- Then, install all the dependencies by using `npm install` command.
- To create tables, you can use `python manage.py create_tables` command.
- Then, to make migrations, run `python manage.py makemigrations` and `python manage.py migrate` commands to be able to use Django sessions.
- If you want to drop tables to recreate again you can use `python manage.py delete_tables` command.
- After that, you can start your backend server by running `python manage.py runserver` and you can start the frontend of your application by running `npm start` at your frontend folder. Make sure you do these things on different terminals so they can run together.



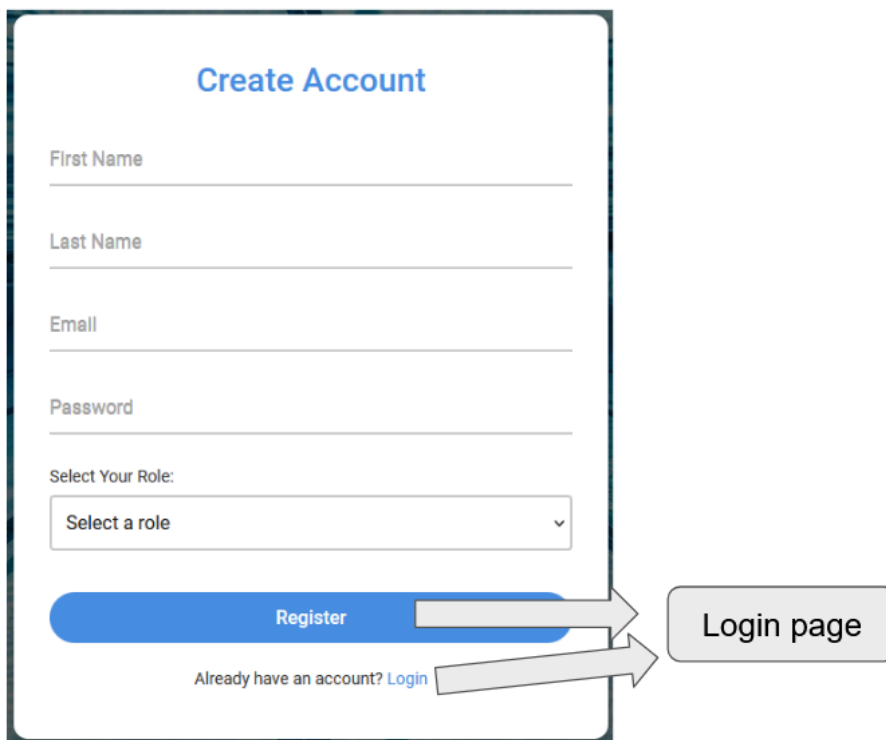
## 7.2 Login Page

All users use the same login page. They can use the email and password they used while registering to log in to the website. When logged in successfully, the user will go to a home page determined by their role.



## 7.3 Register Page

Users can register by entering their information and selecting their role (swimmer, coach, and lifeguard)



## 7.4 Change Password Page

Users can change their password by filling out the form on the left.

**Change Password**

Email

Current Password

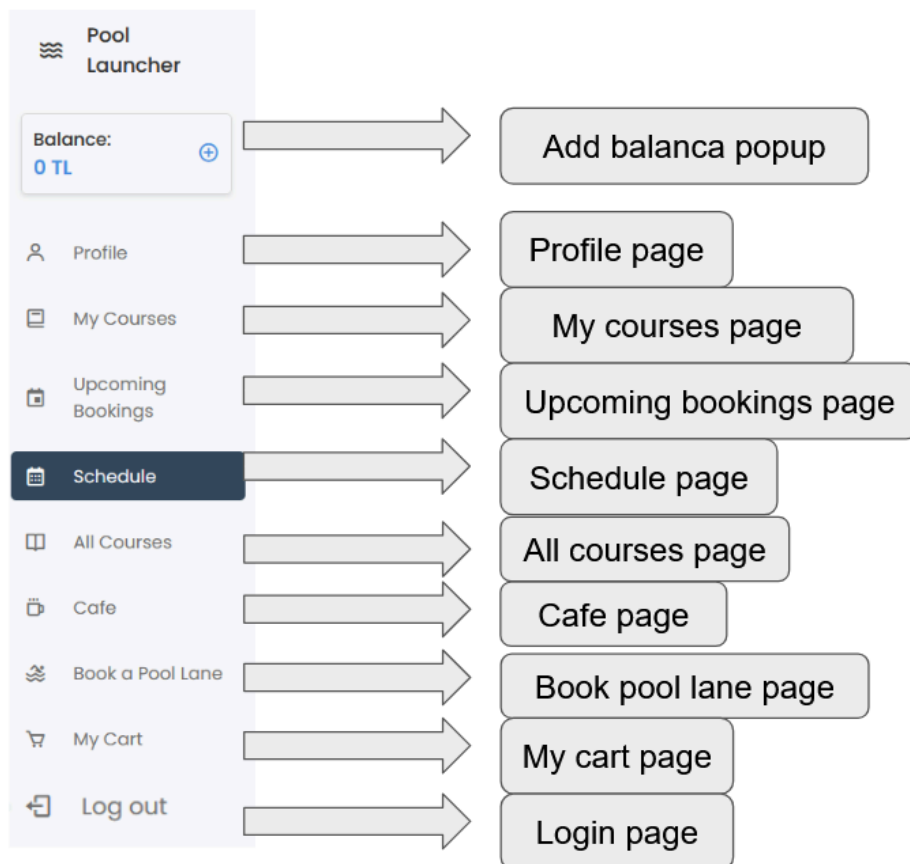
New Password

**Change Password**

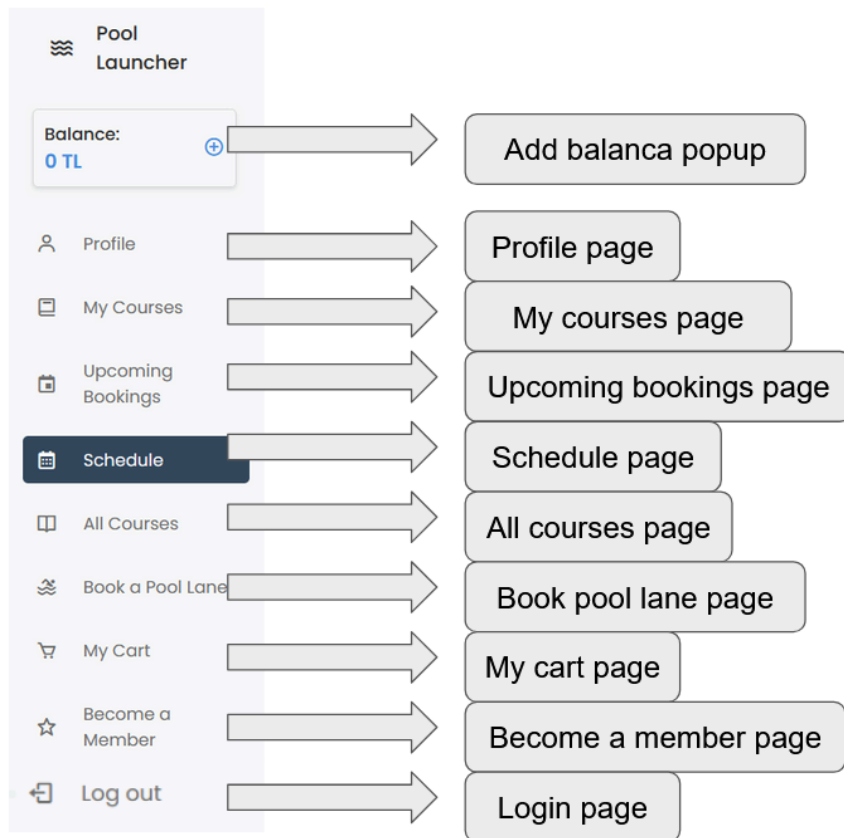
Remembered your credentials? [Login](#)

Login page

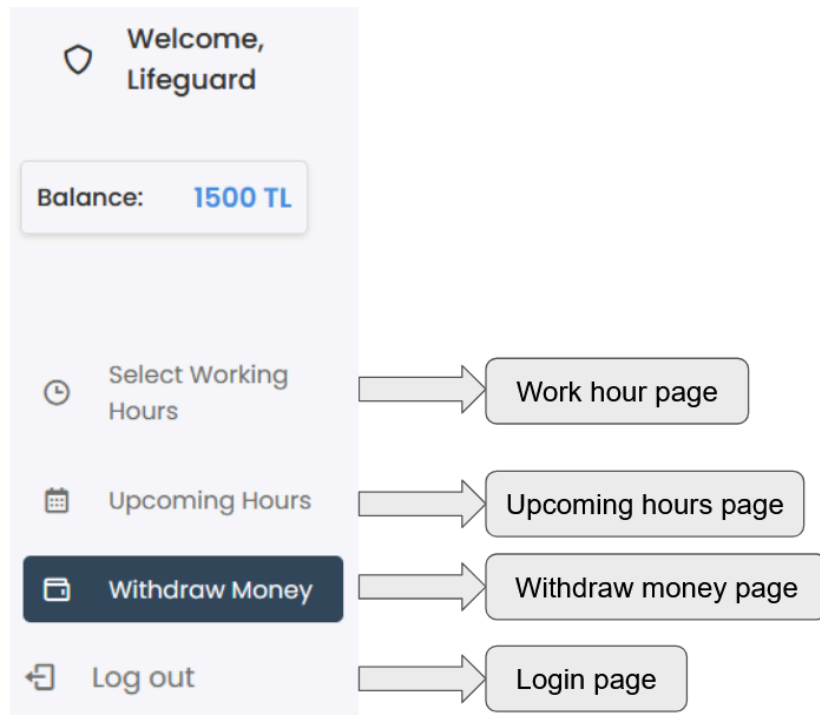
## 7.5 Member Sidebar



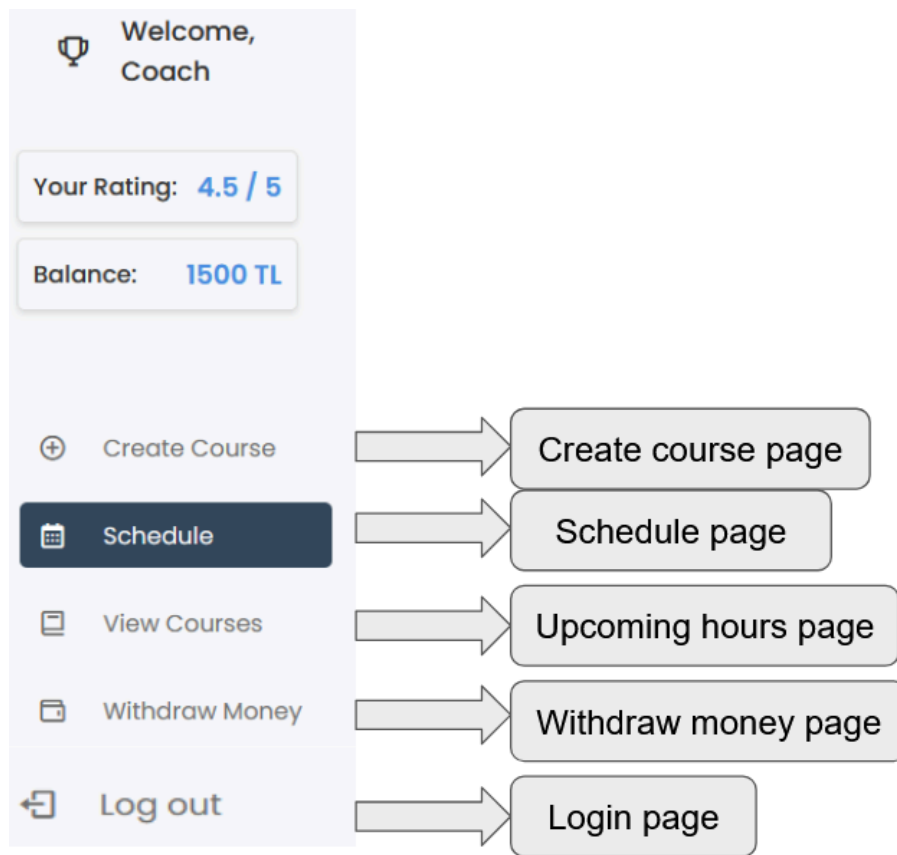
## 7.6 Nonmember Sidebar



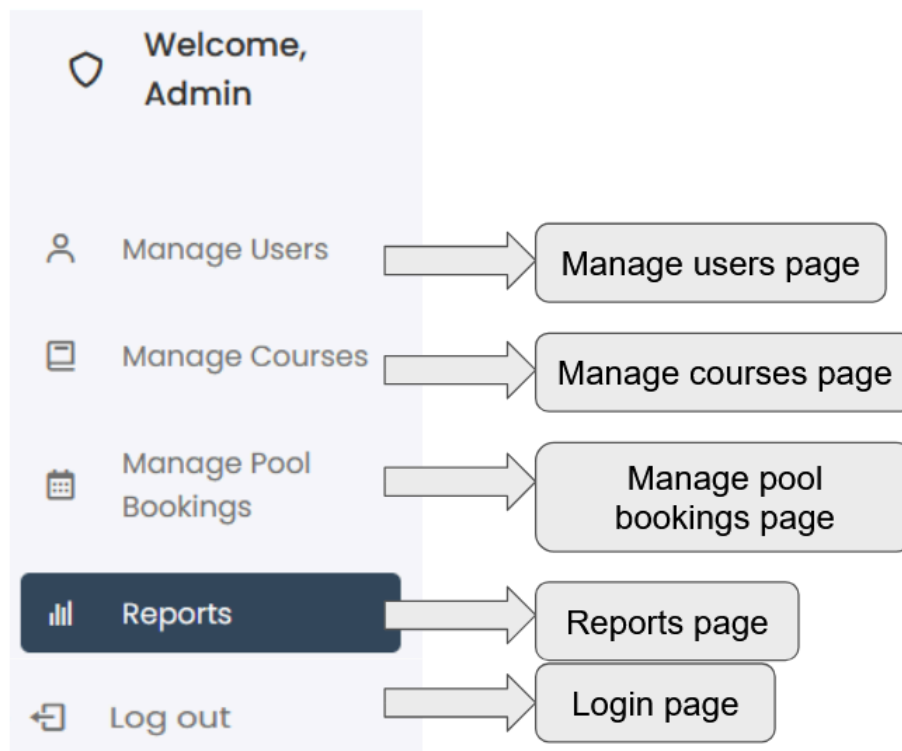
## 7.7 Lifeguard Sidebar



## 7.8 Coach Sidebar

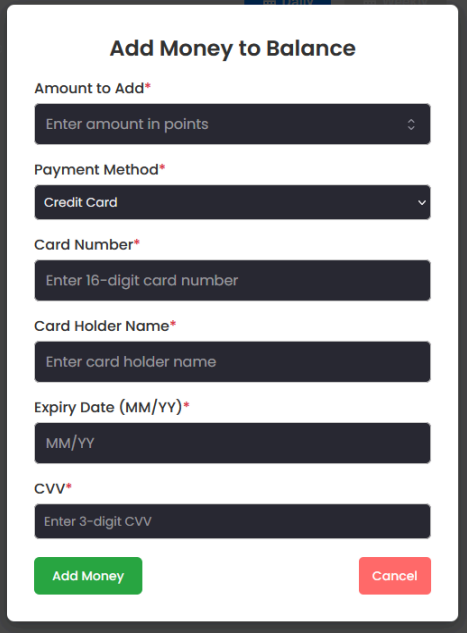


## 7.9 Admin Sidebar



## 7.10 Add Balance Pop-Up

Members and nonmembers can add balance to their account after clicking the plus sign next to their balance. This will open up a popup form that when entered the correct information withdraws money from the users bank account and adds it as balance to their account

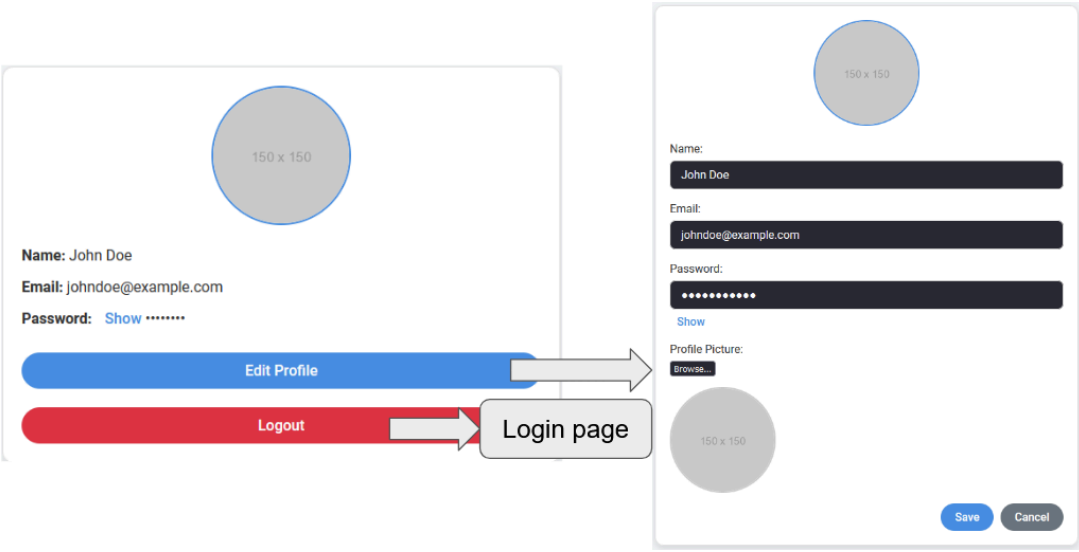


The form is titled "Add Money to Balance". It contains the following fields and buttons:

- Amount to Add\***: A text input field with the placeholder "Enter amount in points" and a small downward arrow icon on the right.
- Payment Method\***: A dropdown menu currently showing "Credit Card".
- Card Number\***: A text input field with the placeholder "Enter 16-digit card number".
- Card Holder Name\***: A text input field with the placeholder "Enter card holder name".
- Expiry Date (MM/YY)\***: A text input field with the placeholder "MM/YY".
- CVV\***: A text input field with the placeholder "Enter 3-digit CVV".
- Buttons**: A green "Add Money" button and a red "Cancel" button.

## 7.11 Profile Page

Users can see and update their information through the profile page. Edit profile button lets them edit their information and upload a profile picture.



The image shows a profile page layout with a flow diagram. The profile page includes:

- A circular profile picture placeholder labeled "150 x 150".
- User information: Name: John Doe, Email: johndoe@example.com, Password: Show .....
- Buttons: "Edit Profile" (blue) and "Logout" (red).

The flow diagram shows:

- An arrow from the "Edit Profile" button pointing to a detailed edit form.
- An arrow from the "Logout" button pointing to a box labeled "Login page".

The detailed edit form includes:

- A circular profile picture placeholder labeled "150 x 150".
- Form fields: Name (John Doe), Email (johndoe@example.com), Password (masked with dots, with a "Show" link), and Profile Picture (with a "Browse" button).
- Buttons: "Save" and "Cancel".

## 7.12 My Courses Page

Members and nonmembers can see their current and previous courses. They can withdraw from the current ones and rate the previous ones.

### My Courses

[Current Courses](#)[Previous Courses](#)

#### Beginner Swimming

**Description:** Learn the basics of swimming, including freestyle, backstroke, and water safety.

**Instructor:** John Doe

**Duration:** 8 weeks

**Pool Location:** Main Pool Area

**Schedule:** Mondays and Wednesdays, 6:00 PM - 7:30 PM

[View](#)

#### Intermediate Swimming

**Description:** Improve your swimming techniques and endurance with advanced drills and workouts.

**Instructor:** Jane Smith

**Duration:** 6 weeks

**Pool Location:** Main Pool Area

**Schedule:** Tuesdays and Thursdays, 5:00 PM - 6:30 PM

[View](#)

#### Beginner Swimming

**Instructor:** John Doe

**Description:** Learn the basics of swimming, including freestyle, backstroke, and water safety.

**Restrictions:** Must be able to swim at least 25 meters.

**Capacity:** 20 participants

**Announcements:** New swimming goggles available for purchase.

**Schedule:** Mondays and Wednesdays, 6:00 PM - 7:30 PM

**Termination Reason:**

Enter reason for termination

[Withdraw Course](#)[Mark as Done](#)[Close](#)

### 7.13 Upcoming Bookings Page

Members and nonmembers can see their booked lanes and cancel their bookings.

Upcoming Bookings

| Date       | Time Slot           | Lane   | Action |
|------------|---------------------|--------|--------|
| 2025-01-04 | 06:00 PM - 07:00 PM | Lane 5 | Cancel |
| 2025-01-05 | 08:00 AM - 09:00 AM | Lane 2 | Cancel |

### 7.14 Member/Non Member Schedule Page

Members and nonmembers can see their weekly and daily schedule consisting of the courses they enrolled in.

Schedule

Daily

Weekly

Wednesday (2024-12-25)

9:00 AM - 10:30 AM: Aqua Aerobics at Main Pool Area

5:30 PM - 7:00 PM: Meditation at Yoga Room

Thursday (2024-12-26)

6:00 AM - 7:30 AM: Beginner Swimming at Main Pool Area

5:00 PM - 6:30 PM: Intermediate Swimming at Main Pool Area

Friday (2024-12-27)

6:00 AM - 7:30 AM: Yoga Class at Fitness Room

6:00 PM - 7:30 PM: Advanced Swimming at Main Pool Area

Saturday (2024-12-28)

8:00 AM - 9:30 AM: Cardio Training at Gym

4:00 PM - 5:30 PM: Kids Swimming at Main Pool Area

## 7.15 All Courses Page

Members and nonmembers can see all the courses offered by the swimming pool. They can see their full details and enroll in the course.

### All Courses

#### Advanced Swimming Techniques

**Instructor:** Alice Johnson (Rating: 4.8)  
**Capacity:** 2/10  
**Deadline:** 2024-12-23

[Details](#) [Enroll](#)

#### Intermediate Swimming

**Instructor:** Jane Smith (Rating: 4.7)  
**Capacity:** 3/15  
**Deadline:** 2024-12-22

[Details](#) [Enroll](#)

#### Water Aerobics

**Instructor:** Bob Brown (Rating: 4.3)  
**Capacity:** 5/25  
**Deadline:** 2024-12-20

[Details](#) [Enroll](#)

#### Beginner Swimming

**Instructor:** John Doe (Rating: 4.5)  
**Capacity:** 0/20  
**Deadline:** 2024-12-21

[Details](#) [Full](#)

### Water Aerobics

**Instructor:** Bob Brown (Rating: 4.3)

**Description:** Engage in low-impact, high-efficiency workouts in the water to improve fitness and flexibility.

**Duration:** 12 weeks

**Pool Location:** Fitness Pool

**Schedule:** Saturdays, 9:00 AM - 10:30 AM

**Restrictions:** Open to all fitness levels.

**Capacity:** 5/25

**Announcements:** Bring your own water bottle.

[Close](#)



## 7.16 Book Pool Lane Page

Members and nonmembers can book a lane by entering the date, the lane, and the time slot they want.


### Book a Pool Lane

Select Date:  
04 / 01 / 2025

Select Lane:  
Lane 5

Select Time Slot:  
06:00 PM - 07:00 PM

Book Lane



#### Booking Confirmed!

**Date:** 2025-01-04  
**Time Slot:** 06:00 PM - 07:00 PM  
**Lane:** Lane 5

## 7.17 My Cart Page

Members and nonmembers can pay for the courses they enrolled in. They can decide to not take a course and remove it from their cart.

### My Cart

**Beginner Swimming**  
Instructor: John Doe  
Duration: 8 weeks  
Price: 100TL

Remove

**Advanced Swimming Techniques**  
Instructor: Alice Johnson  
Duration: 10 weeks  
Price: 150TL

Remove

Total: 250TL

Select Payment Method:  
Credit Card

Purchase

## 7.18 Cafe Page

Members can redeem items from the cafe using the points they earned. They can decide what to add to their cart and remove the items in their cart.

### Cafe

Your Points Balance: 0 Points  
Total Points Required: 80 Points  
You do not have enough points to complete this purchase.

Available Items

150 x 150

**Fresh Juice**  
A refreshing blend of seasonal fruits.  
Price: 50 Points  
Add to Cart

150 x 150

**Protein Bar**  
High-protein snack for energy.  
Price: 30 Points  
Add to Cart

150 x 150

**Sandwich**  
Delicious sandwich with fresh ingredients.  
Price: 40 Points  
Add to Cart

#### Your Cart

Fresh Juice - 50 Points

Remove

Protein Bar - 30 Points

Remove

Total: 80 Points

Purchase

## 7.19 Become a Member Page

Nonmembers can become a member by subscribing to the membership plan. They can fill out the payment form, and on success, their account will be switched to a member account.

**Become a Member**

Full Name\*:

Email Address\*:

Phone Number\*:

Select Membership Plan\*:  

Basic - \$50/month

Card Number\*:

Expiry Date\*:

CVV\*:

Register

## 7.20 Create Course Page

Coaches can create their own courses by filling a form. The form includes information such as the course date, hour, age and weight range, gender, capacity. They can also check if the time is available by clicking the check availability button.

**Create Course**

Start Date:

Start Time:

End Time:

Check Availability

Pool Facility:  

--Select Facility--

Age Range:

Weight Range:

Sex:  

--Select--

Price (TL):

Capacity:

Explanation:  

Add additional details about the course

Create Course

## 7.21 Coach Schedule Page

Coaches can see their daily and weekly schedules. The schedule includes the date, time, the course name, and the pool name.

Schedule

Daily

Weekly

2024-12-22

6:00 AM - 7:30 AM: Beginner Swimming - Main Pool Area

5:00 PM - 6:30 PM: Intermediate Swimming - Main Pool Area

## 7.22 View Courses Page

Coaches can view their upcoming and past courses and check their details.

View Courses

Upcoming Courses

Past Courses

Beginner Swimming

2024-12-20 | 6:00 PM - 7:30 PM

Main Pool Area

Capacity: 2/10

See Details

Intermediate Swimming

2024-12-25 | 5:00 PM - 6:30 PM

Main Pool Area

Capacity: 1/8

See Details

Advanced Swimming

2024-12-30 | 7:00 PM - 8:30 PM

Main Pool Area

Capacity: 0/5

See Details

Course Details

Title: Beginner Swimming

Date: 2024-12-20

Time: 6:00 PM - 7:30 PM

Location: Main Pool Area

Capacity: 2/10

Students

50 x 50

John Doe

50 x 50

Jane Smith

Close

51

## 7.23 Withdraw Money Page

Coaches and lifeguards can withdraw money from their account to their bank account by entering the amount and the IBAN of their bank account.

### Withdraw Money

Your Current Balance: **1500 TL**

Amount to Withdraw:

Enter amount in TL

IBAN:

Enter your IBAN

Withdraw

## 7.24 Work Hours Page

Lifeguards can enter the days they are working by selecting the time, the pool, and the hours.

Select Working Hours

16 / 01 / 2025

Facility 1

Facility 2

8:00 AM - 10:00 AM

10:00 AM - 12:00 PM

12:00 PM - 2:00 PM

2:00 PM - 4:00 PM

4:00 PM - 6:00 PM

6:00 PM - 8:00 PM

Save Working Hours

## 7.25 Upcoming Hours Page

Lifeguards can see and cancel the working hours they registered to the system.

| Upcoming Working Hours  |        |
|---|--------|
| Date: 2024-12-28<br>Time: 8:00 AM - 10:00 AM<br>Facility: Facility 1  | Cancel |
| Date: 2024-12-28<br>Time: 10:00 AM - 12:00 PM<br>Facility: Facility 2 | Cancel |
| Date: 2024-12-29<br>Time: 4:00 PM - 6:00 PM<br>Facility: Facility 1   | Cancel |

## 7.26 Manage Users Page

Admin can see the users registered in the system and their information. They can also delete accounts.

| Manage Users |               |                           |        |        |   |
|--------------|---------------|---------------------------|--------|--------|---|
| All Users    |               |                           |        |        |   |
| ID           | Name          | Email                     | Role   | Points | Actions   |
| 1            | John Doe      | john.doe@example.com      | Member | 150    | <button>View Profile</button> <button>Delete</button> |
| 2            | Jane Smith    | jane.smith@example.com    | Admin  | 300    | <button>View Profile</button> <button>Delete</button> |
| 3            | Alice Johnson | alice.johnson@example.com | Member | 200    | <button>View Profile</button> <button>Delete</button> |
| 4            | Bob Brown     | bob.brown@example.com     | Member | 120    | <button>View Profile</button> <button>Delete</button> |
| 5            | Sara Lee      | sara.lee@example.com      | Admin  | 450    | <button>View Profile</button> <button>Delete</button> |

## 7.27 Manage Courses Page

Admin can see the courses opened and their information. They can also delete courses.

| Manage Swimming Lessons |                     |           |        |   |
|-------------------------|---------------------|-----------|--------|---|
| All Swimming Lessons    |                     |           |        |   |
| Lesson Name             | Coach               | Age Range | Gender | Actions                                       |
| Beginner Swimming       | Coach Emily Brown   | 5-10      | Any    | <button>View</button> <button>Delete</button> |
| Intermediate Swimming   | Coach Mark Wilson   | 11-15     | Male   | <button>View</button> <button>Delete</button> |
| Advanced Swimming       | Coach Sarah Johnson | 16-20     | Female | <button>View</button> <button>Delete</button> |
| Recreational Swimming   | Coach Daniel Lee    | All Ages  | Any    | <button>View</button> <button>Delete</button> |
| Competitive Training    | Coach Anna Smith    | 10-18     | Any    | <button>View</button> <button>Delete</button> |

## 7.28 Manage Pool Bookings Pages

Admin can see the lane bookings and their date and location information. They can also delete bookings.

| Manage Pool Bookings |               |            |                     |           |           |   |
|----------------------|---------------|------------|---------------------|-----------|-----------|---|
| All Bookings         |               |            |                     |           |           |   |
| ID                   | User          | Date       | Time                | Pool      | Status    | Actions   |
| 1                    | John Doe      | 2024-12-30 | 10:00 AM - 12:00 PM | Main Pool | Confirmed | <a href="#">View Booking</a> <a href="#">Delete</a> |
| 2                    | Jane Smith    | 2024-12-31 | 2:00 PM - 4:00 PM   | Lap Pool  | Pending   | <a href="#">View Booking</a> <a href="#">Delete</a> |
| 3                    | Alice Johnson | 2025-01-01 | 8:00 AM - 10:00 AM  | Kids Pool | Confirmed | <a href="#">View Booking</a> <a href="#">Delete</a> |
| 4                    | Bob Brown     | 2025-01-02 | 6:00 PM - 8:00 PM   | Main Pool | Cancelled | <a href="#">View Booking</a> <a href="#">Delete</a> |

## 7.29 Reports Page

Admin can generate system reports which can include user, transaction, and course information.

Generate Report

☒ Include Users Information

☒ Include Transactions Information

☒ Include Courses Information

Generate Report

### Users Report

| ID | Name          | Email             |
|----|---------------|-------------------|
| 1  | Alice Johnson | alice@example.com |
| 2  | Bob Smith     | bob@example.com   |

### Transactions Report

| Transaction ID | User          | Amount | Date       |
|----------------|---------------|--------|------------|
| 1001           | Alice Johnson | \$50   | 2024-12-01 |
| 1002           | Bob Smith     | \$75   | 2024-12-05 |

### Swimming Courses Report

| Course ID | Lesson Title          | Coach               | Enrollment |
|-----------|-----------------------|---------------------|------------|
| SWIM101   | Beginner Swimming     | Coach Emily Brown   | 50         |
| SWIM202   | Intermediate Swimming | Coach Mark Wilson   | 30         |
| SWIM303   | Advanced Swimming     | Coach Sarah Johnson | 20         |

## 8. References

- [1] Canva, “Canva: Herkes için Görsel Çalışma Seti,” *Canva*. [https://www.canva.com/tr\\_tr](https://www.canva.com/tr_tr)
- [2] “Draw.io - free flowchart maker and diagrams online,” Flowchart Maker & Online Diagram Software.  
[https://app.diagrams.net/#G19T17ddcRAOtlJUwzh8Hsv1P1V\\_jbrDnp#%7B%22pageId%22%3A%22R2IEEEUBdFMjLhIrx00%22%7D](https://app.diagrams.net/#G19T17ddcRAOtlJUwzh8Hsv1P1V_jbrDnp#%7B%22pageId%22%3A%22R2IEEEUBdFMjLhIrx00%22%7D)