BILKENT UNIVERSITY
CS353 DATABASE SYSTEMS

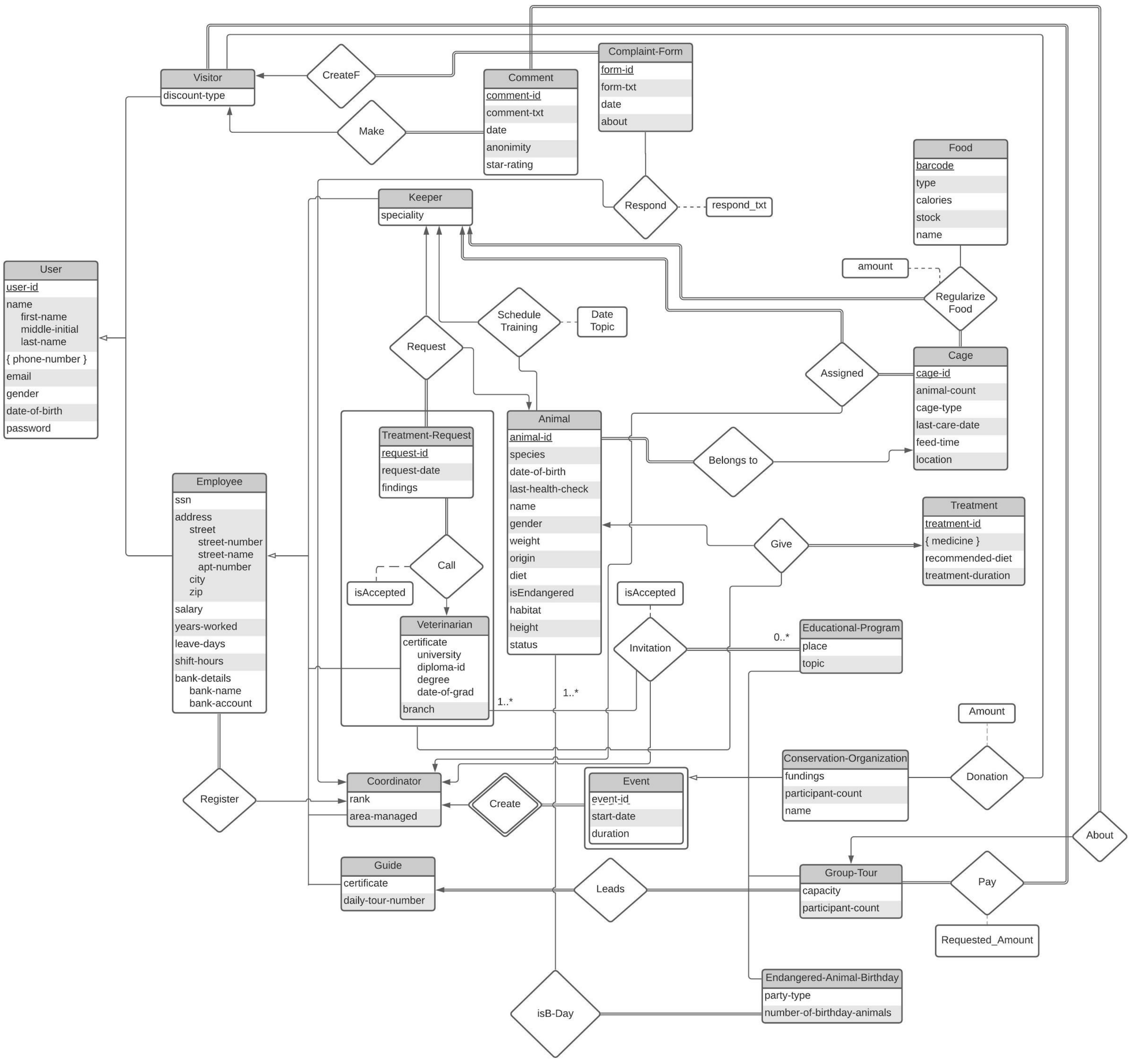
DESIGN REPORT

GROUP 33

Asya Doğa Özer 21803479
Can Kılıç 21703333
Ege Çetin 21802305
Ufuk Palpas 21702958

Table of Contents

1. Revised E/R Diagram



- Security and Researcher entity removed.
- Date and Topic added to the Schedule Training
- Password added to the User.
- About relation added between the group tour and comment.

- Aggregation between comment and complaint-form removed.
- Animal separated from the aggregation of veterinarian and treatment-request.
- From employee and veterinarians primary keys removed.
- Respond_txt added to the Respond relation.
- Amount added to the Regularize Food.
- Register relation added between Employee and Coordinator.
- Name added to the Conservation Organizations.

2. E/R Diagram as Relation Tables

2.1 User

Model:
user(<u>user-id</u>, name, phone-number, email, gender, date-of-birth, password)

Candidate Key:
user-id, email, phone-number

Primary Key:
user-id

Table Declaration:
```
CREATE TABLE USER(
        user-id int PRIMARY KEY AUTO_INCREMENT,
        name VARCHAR(32) NOT NULL,
        phone-number NVARCHAR(32) NOT NULL UNIQUE,
        email VARCHAR(32) NOT NULL UNIQUE,
        gender VARCHAR(20) DEFAULT NULL,
        date-of-birth DATE NOT NULL,
        password VARCHAR(32) NOT NULL
);
```

2.2 Visitor

Model:
visitor(<u>user-id</u>, discount-type)

Foreign Key:
user-id

Candidate Key:
user-id

Primary Key:
user-id

Table Declaration:
```
CREATE TABLE VISITOR(
        user-id int PRIMARY KEY,
        discount-type boolean DEFAULT 1,
        FOREIGN KEY(user-id) REFERENCES User(user-id) ON DELETE
        CASCADE
);
```

2.3 Employee

Model:
employee(<u>user-id</u>, ssn, address, salary, years-worked, leave-days, shift-hours, bank-details, createdBy)

Foreign Key:
user-id

Candidate Key:
user-id, ssn
Primary Key:
user-id

Table Declaration:
```
CREATE TABLE EMPLOYEE(
        user-id int PRIMARY KEY,
        ssn int NOT NULL UNIQUE,
        address varchar(50) DEFAULT NULL,
        salary int NOT NULL,
        years-worked int DEFAULT NULL,
        leave-days varchar(20) DEFAULT NULL,
        shift-hours varchar(10) DEFAULT NULL,
        bank-details varchar(50) NOT NULL DEFAULT ' ',
        createdBy int NOT NULL,
        FOREIGN KEY(user-id) REFERENCES User(user-id) ON DELETE
        CASCADE
        FOREIGN KEY(createdBy-id) REFERENCES coordinator(user-id) ON
        DELETE SET NULL
);
```

2.4 Keeper

Model:
keeper(<u>user-id</u>, speciality)

Foreign Key:
user-id

Candidate Key:
user-id

Primary Key:
user-id

Table Declaration:

```
CREATE TABLE KEEPER(
        user-id int PRIMARY KEY,
        specialty varchar(20) DEFAULT NULL,
        FOREIGN KEY(user-id) REFERENCES User(user-id) ON DELETE
        CASCADE
);
```

2.5. Veterinarian

Model:
veterinarian(user-id, certificate, branch)

Foreign Key:
user-id

Candidate Key:
user-id, certificate

Primary Key:
user-id

Table Declaration:
```
CREATE TABLE VETERINARIAN(
        user-id int PRIMARY KEY,
        certificate varchar(50) NOT NULL,
        branch varchar(20) DEFAULT NULL,
        FOREIGN KEY(user-id) REFERENCES user(user-id) ON DELETE
        CASCADE
);
```

2.6 Treatment Request

Model:
treatment-request(request-id, vet-id, request-date, findings, isAccepted)

Candidate Key:
request-id

Primary Key:
request-id

Table Declaration:

```
CREATE TABLE TREATMENT-REQUEST(
        request-id int PRIMARY KEY AUTO_INCREMENT,
        vet-id int DEFAULT NULL,
        request-date date DEFAULT CURRENT_DATE,
        findings varchar(120) NOT NULL,
        isAccepted boolean DEFAULT NULL,
        FOREIGN KEY(vet-id) REFERENCES veterinarian(user-id) ON DELETE
        CASCADE
);
```

2.7 Coordinator

Model:
coordinator(user-id, rank, area-managed)

Foreign Key:
user-id

Candidate Key:
user-id

Primary Key:
user-id

Table Declaration:

```
CREATE TABLE COORDINATOR(
        user-id int PRIMARY KEY,
        rank varchar(20) NOT NULL,
        area-managed(20) NOT NULL,
        FOREIGN KEY(user-id) REFERENCES User(user-id) ON DELETE
        CASCADE
);
```

2.8 Event

Model:
event(user-id, event-id, start-date, duration)

Foreign Key:
user-id

Candidate Key:
{user-id, event-id}

Primary Key:
{user-id, event-id}

Table Declaration:

```
CREATE TABLE EVENT(
        user-id int PRIMARY KEY NOT NULL,
        event-id int NOT NULL AUTO_INCREMENT,
        start-date date NOT NULL,
        duration varchar(10) NOT NULL,
        FOREIGN KEY(user-id) REFERENCES coordinator(user-id) ON DELETE
        CASCADE
);
```

2.9 Guide

Model:
guide(<u>user-id</u>, certificate, daily-tour-number)

Foreign Key:
user-id

Candidate Key:
user-id, certificate

Primary Key:
user-id

Table Declaration:

```
CREATE TABLE GUIDE(
        user-id int PRIMARY KEY,
        certificate varchar(50) NOT NULL,
        daily-tour-number int DEFAULT NULL,
        FOREIGN KEY(user-id) REFERENCES User(user-id) ON DELETE
        CASCADE
);
```

2.10 Educational Program

Model:
educational-program(event-id, place, topic)

Foreign Key:
event-id

Candidate Key:
event-id

Primary Key:
event-id

Table Declaration:

```
CREATE TABLE EDUCATIONAL-PROGRAM(
        event-id int PRIMARY KEY,
        place varchar(20) NOT NULL,
        topic varchar(10) NOT NULL,
        FOREIGN KEY(event-id) REFERENCES event(event-id) ON DELETE
        CASCADE
);
```

2.11 Conservation Organization

Model:
conservation-organization(event-id, fundings, participant-count, name)

Foreign Key:
event-id

Candidate Key:
event-id

Primary Key:
event-id

Table Declaration:

```
CREATE TABLE CONSERVATION-ORGANIZATION(
        event-id int PRIMARY KEY,
        fundings decimal(19,4) DEFAULT 0,
        participant-count int DEFAULT NULL,
        name varchar(50) NOT NULL,
        FOREIGN KEY(event-id) REFERENCES event(event-id) ON DELETE
        CASCADE
);
```

2.12 Group Tour

Model:
group-tour(<u>event-id</u>, guide-id, capacity, participant-count)

Foreign Key:
event-id

Candidate Key:
event-id

Primary Key:
event-id

Table Declaration:

```
CREATE TABLE GROUP-TOUR(
        event-id int PRIMARY KEY,
        guide-id int NOT NULL,
        capacity int NOT NULL,
        participant-count int DEFAULT NULL,
        FOREIGN KEY(event-id) REFERENCES event(event-id) ON DELETE
        CASCADE
        FOREIGN KEY(user-id) REFERENCES guide(user-id) ON DELETE
        CASCADE
);
```

2.13 Endangered Animal Birthday

Model:
endangered-animal-birthday(event-id, party-type, number-of-birthday-animals)

Foreign Key:
event-id
Candidate Key:
event-id
Primary Key:
Event-id

Table Declaration:
```
CREATE TABLE ENDANGERED-ANIMAL-BIRTHDAY(
        event-id int PRIMARY KEY,
        party-type varchar(20) NOT NULL,
        number-of-birthday-animals SMALLINT DEFAULT NULL,
        FOREIGN KEY(event-id) REFERENCES event(event-id) ON DELETE
        CASCADE
);
```

2.14 Treatment

Model:

treatment(treatment-id, medicine, recommended-diet, treatment-duration)

Candidate Key:

treatment-id

Primary Key:

treatment-id

Table Declaration:

```
CREATE TABLE TREATMENT(
        treatment-id int PRIMARY KEY AUTO_INCREMENT ,
        medicine varchar(120) NOT NULL,
        medicine varchar(20) DEFAULT NULL,
        treatment-duration varchar(20) NOT NULL
);
```

2.15 Cage

Model:
cage(cage-id, animal-count, cage-type, last-care-date, feed-time, location)

Candidate Key:
cage-id

Primary Key:
cage-id

Table Declaration:

```
CREATE TABLE CAGE(
        cage-id int PRIMARY KEY AUTO_INCREMENT ,
        animal-count SMALLINT DEFAULT NULL,
        cage-type varchar(20) DEFAULT NULL,
        last-care-date date DEFAULT NULL,
        feed-time time DEFAULT NULL,
        location varchar(20) NOT NULL
);
```

2.16 Food

Model:
food(<u>barcode</u>, type, calories, stock, name)

Candidate Key:
barcode

Primary Key:
barcode

Table Declaration:

```
CREATE TABLE FOOD(
        barcode int PRIMARY KEY NOT NULL UNIQUE ,
        type varchar(20) NOT NULL,
        calories varchar(20) NOT NULL,
        stock int DEFAULT NULL,
        name varchar(20) NOT NULL
);
```

2.17 Comment

Model:
comment(comment-id, user-id, event-id, comment-text, date, anonymity, star-rate)

Foreign Key:
user-id, event-id

Candidate Key:
comment-id

Primary Key:
comment-id

Table Declaration:

```
CREATE TABLE COMMENT(
        comment-id int PRIMARY KEY AUTO_INCREMENT ,
        comment-text nvarchar(200) DEFAULT NULL,
        date date DEFAULT CURRENT_DATE,
        anonymity boolean NOT NULL DEFAULT 1,
        star-rate int NOT NULL DEFAULT 5,
        user-id int NOT NULL,
        event-id int NOT NULL,
        FOREIGN KEY(event-id) REFERENCES group-tour(event-id) ON DELETE
        CASCADE,
        FOREIGN KEY(user-id) REFERENCES visitor(user-id) ON DELETE
        CASCADE
);
```

2.18 Complaint

Model:

complaint(form-id, coor-id, vis-id, form-text, date, about, respond_txt)

Candidate Key:

form-id

Primary Key:

form-id

Table Declaration:

```
CREATE TABLE COMPLAINT(
        form-id int PRIMARY KEY AUTO_INCREMENT ,
        form-text nvarchar(200) DEFAULT ' ',
        date date DEFAULT CURRENT_DATE,
        about varchar(20) DEFAULT ' ',
        coor-id int DEFAULT NULL,
        vis-id int NOT NULL,
        respond-text nvarchar(200) DEFAULT ' ',
        FOREIGN KEY(coor-id) REFERENCES coordinator(user-id) ON DELETE
        CASCADE,
        FOREIGN KEY(vis-id) REFERENCES visitor(user-id) ON DELETE
        CASCADE
);
```

2.19 Animal

Model:

animal(animal-id, trainer-id, cage-id, training-date, training-topic, species, date-of-birth, last-health-check, name, gender, weight, origin, diet, isEndangered, habitat, height, status)

Foreign Key:

Trainer-id, cage-id

Candidate Key:

animal-id

Primary Key:

animal-id

Table Declaration:

```
CREATE TABLE ANIMAL(
        animal-id int PRIMARY KEY AUTO_INCREMENT,
        user-id int NOT NULL,
        cage-id int NOT NULL,
        training-date date DEFAULT NULL,
        training-topic varchar(40) DEFAULT NULL,
        species varchar(20) DEFAULT NULL,
        date-of-birth date DEFAULT CURRENT_DATE,
        last-health-check date DEFAULT CURRENT_DATE,
        name varchar(20) DEFAULT NULL,
        gender varchar(10) DEFAULT NULL,
        weight decimal(6,2) DEFAULT NULL,
        origin varchar(20) DEFAULT NULL,
        diet varchar(20) DEFAULT NULL,
        isEndangered boolean DEFAULT FALSE,
        habitat varchar(20) DEFAULT NULL,
        height decimal(4,2) DEFAULT NULL,
        status varchar(20) DEFAULT NULL,
        FOREIGN KEY(trainer-id) REFERENCES keeper(user-id) ON DELETE
        CASCADE,
        FOREIGN KEY(cage-id) REFERENCES cage(cege-id) ON DELETE
        CASCADE
);
```

2.20 Donation

Model:

donation(user-id, event-id, Amount)

Foreign key:

user-id, event-id

Table Declaration:

```
CREATE TABLE DONATION(
        user-id int NOT NULL,
        event-id int NOT NULL,
        amount decimal(19,4) DEFAULT 0,
        FOREIGN KEY(user-id) REFERENCES visitor(user-id) ON DELETE SET
        NULL,
        FOREIGN KEY(event-id) REFERENCES conservation-organization(barcode)
        ON DELETE SET NULL
);
```

2.21 Regularize Food

Model:

Regularize-Food(<u>cage-id</u>, <u>feed-date</u>, barcode, user-id, amount)

Foreign Key:

cage-id, barcode-id, user-id

Candidate Key:

{cage-id, feed-date}

Primary Key:

{cage-id, feed-date}

Table Declaration:

```
CREATE TABLE REGULARIZE-FOOD(
        cage-id int NOT NULL,
        barcode int NOT NULL,
        feed-date date DEFAULT CURRENT_DATE,
        amount decimal(5,2) DEFAULT 0,
        user-id int NOT NULL,
        FOREIGN KEY(cage-id) REFERENCES cage(user-id)ON DELETE SET
        NULL,
        FOREIGN KEY(user-id) REFERENCES keeper(user-id) ON DELETE SET
        NULL,
        FOREIGN KEY(barcode-id) REFERENCES food(barcode) ON DELETE
        CASCADE
);
```

2.22 Request

Model:

Request(animal-id, request-id, user-id)

Foreign Key:

animal-id, request-id, user-id

Candidate Key:

request-id

Primary Key:

request-id

Table Declaration:

```
CREATE TABLE REQUEST(
        animal-id int NOT NULL,
        request-id int NOT NULL,
        user-id int NOT NULL,
        FOREIGN KEY(animal-id) REFERENCES animal(animal-id) ON DELETE
        CASCADE,
        FOREIGN KEY(request-id) REFERENCES Treatment-Request(request-id)
        ON DELETE CASCADE,
        FOREIGN KEY(user-id) REFERENCES keeper(user-id) ON DELETE SET
        NULL
);
```

2.23 Assigned

Model:
Assigned(keep-id, <u>cage-id</u>, coor-id)

Foreign Key:

keep-id, cage-id, coor-id

Candidate Key:
cage-id

Primary Key:

cage-id

Table Declaration:

```
CREATE TABLE ASSIGNED(
        keep-id int NOT NULL,
        coor-id int NOT NULL,
        cage-id int NOT NULL,
        FOREIGN KEY(keep-id) REFERENCES Keeper(user-id)ON DELETE
        CASCADE,
        FOREIGN KEY(coor-id) REFERENCES Coordinator(user-id) ON DELETE
        NULL,
        FOREIGN KEY(cage-id) REFERENCES Cage(cage-id) ON DELETE
        CASCADE
);
```

2.24 Invitation

Model:

Invitation(<u>vet-id</u>, coor-id, <u>event-id</u>, isAccepted)

Foreign Key:

vet-id, event-id, coor-id

Candidate Key:

{vet-id, event-id}

Primary Key:

{vet-id, event-id}

Table Declaration:

```
CREATE TABLE INVITATION(
        vet-id int NOT NULL,
        coor-id int NOT NULL,
        event-id int NOT NULL,
        isAccepted boolean DEFAULT NULL,
        FOREIGN KEY(vet-id) REFERENCES veterinarian(user-id)ON DELETE
        CASCADE,
        FOREIGN KEY(coor-id) REFERENCES Coordinator(user-id) ON DELETE
        CASCADE,
        FOREIGN KEY(event-id) REFERENCES educational-program(event-id) ON
        DELETE CASCADE
);
```

2.25 Give

Model:

Give(animal-id, treatment-id, request-id, vet-id)

Foreign Key:

animal-id, treatment-id, request-id, vet-id

Candidate Key:

{request-id, vet-id}, treatment-id

Primary Key:

treatment-id

Table Declaration:

```
CREATE TABLE GIVE(
        animal-id int NOT NULL,
        treatment-id int NOT NULL,
        request-id int NOT NULL,
        vet-id int NOT NULL,
        FOREIGN KEY(animal-id) REFERENCES Animal(animal-id) ON DELETE
        CASCADE,
        FOREIGN KEY(treatment-id) REFERENCES Coordinator(treatment-id) ON
        DELETE CASCADE,
        FOREIGN KEY(request-id) REFERENCES Treatment-Request(request-id)
        ON DELETE SET NULL,
        FOREIGN KEY(vet-id) REFERENCES Veterinarian(user-id) ON DELETE
        SET NULL
);
```

2.26 Pay

Model:
Pay(event-id, user-id, requested-amount)

Foreign Key:

event-id, user-id

Table Declaration:

```
CREATE TABLE PAY(
          event-id int NOT NULL,
          user-id int NOT NULL,
          requested-amount decimal(19,4) DEFAULT 0,
          FOREIGN KEY(event-id) REFERENCES
          conservation-organization(event-id) ON DELETE SET NULL,
          FOREIGN KEY(user-id) REFERENCES visitor(user-id) ON DELETE SET
          NULL,
);
```

2.27 is Birthday

Model:
isB-Day(animal-id,event-id)

Foreign Key:

animal-id, event-id

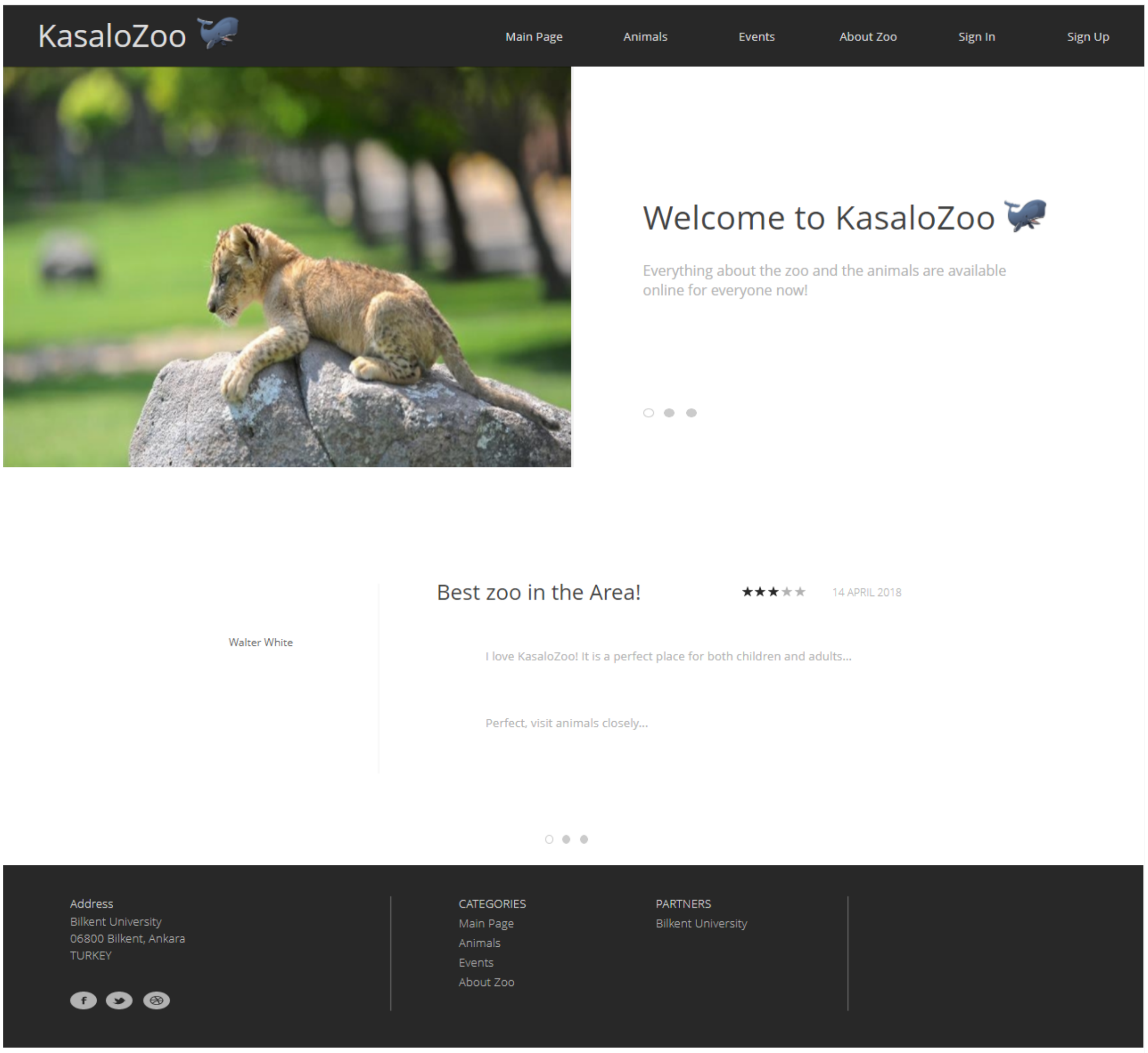
Table Declaration:

```
CREATE TABLE ISB-DAY(
        animal-id int NOT NULL,
        event-id int NOT NULL,
        FOREIGN KEY(animal-id) REFERENCES Animal(animal-id) ON DELETE
        CASCADE,
        FOREIGN KEY(event-id) REFERENCES
        Endangered-Animal-Birthday(event-id) ON DELETE CASCADE
);
```

3. Software Design Specifications

3.1 User Interface Design and Corresponding SQL Statements

3.1.1 Homepage

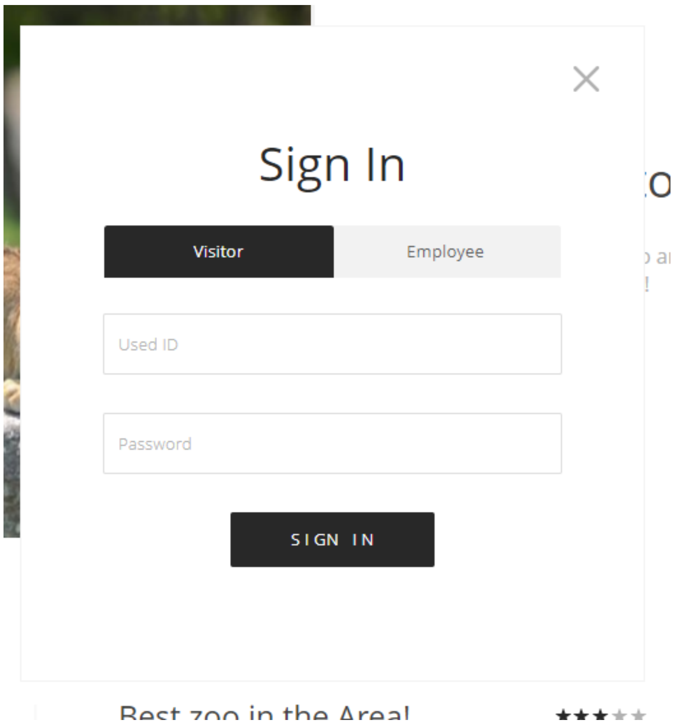


Inputs: None

Process: Users who have not logged in or registered can access the sign in or sign up screens by using the buttons at the top right of the screen. Furthermore, they can access some information about the zoo which does not require any registration.

SQL Statements: These operations do not require any SQL operation.

3.1.2 Login Screen

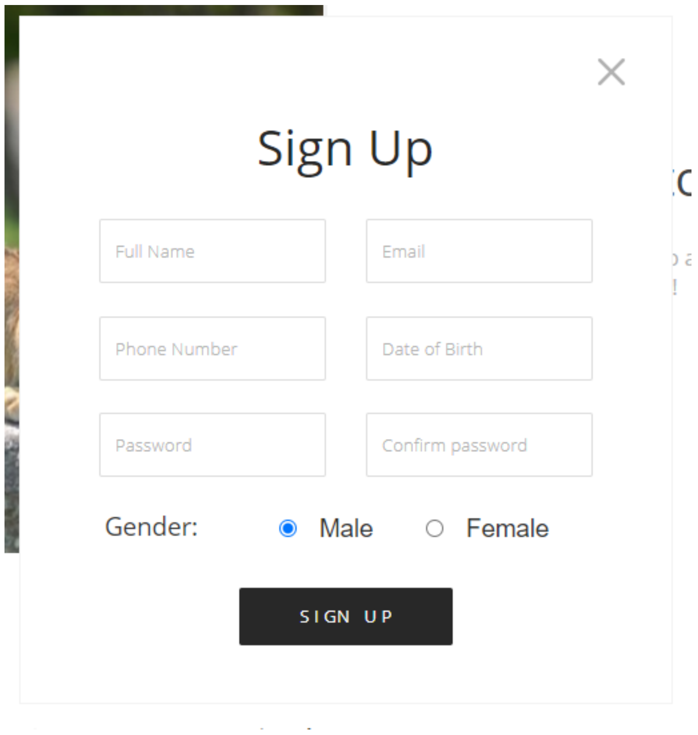


Input: @user_id, @password

Process: Employees and visitors have two different signs in screens which are combined in one screen. Users can select their input areas from the tab menu on top of the input areas and both can enter by using their user id and passwords.

SQL Statements:
SELECT user-id, password
FROM User
Where user-id = @user_id and password = @password

3.1.3 Sign Up Screen



Gender: [✓ Male] Female

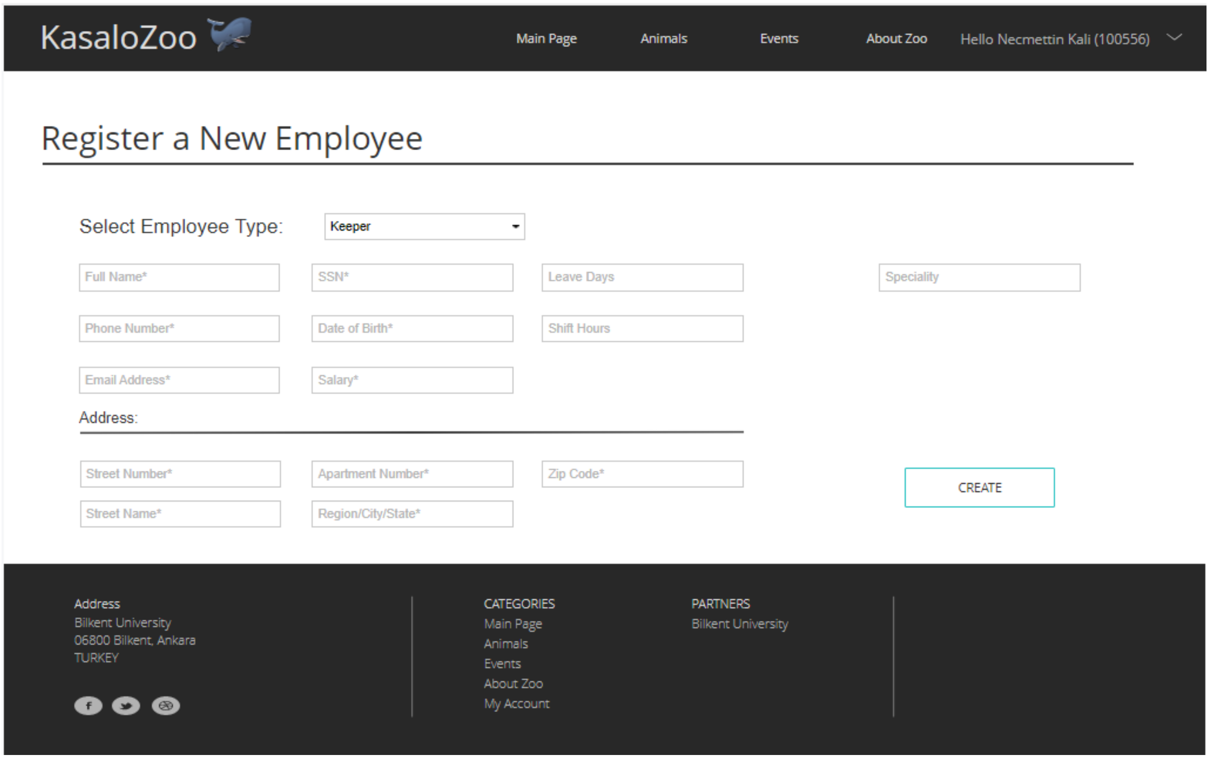
Input: @Full_Name, @Email, @PhoneNumber, @Date_of_Birth, @Password, @ConPass, @Gender

Process: New visitors can use this page to register an account. In order to complete the creation, they need to specify their names, emails, phone numbers, date of births, gender and their password. After pressing the sign up button, an unique user-id is given to the user.

SQL Statements:

```
insert into user( name, phone-number, email, gender,
        date-of-birth, password)
        values(@Full_Name, @PhoneNumber, @Email,
        @Gender, @Date_of_Birth, @Password);
Select last_insert_id() as user;
insert into visitor( user-id)
        values(user);
```

3.1.4 Employee Sign Up



Select Employee Type: Keeper

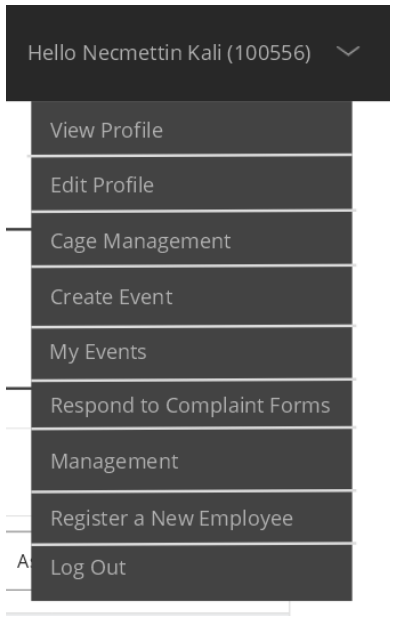
Input: @Full_Name, @SSN, @LeaveDays, @ShiftHours, @Email, @PhoneNumber, @Date_of_Birth, @Password, @ConPass, @Gender, @Address, @Salary, @Speciality

Process: Coordinator can add new employees to the system. After selecting the type of the employee, coordinators must specify the important information about the employees. After specifying this information, they can add the employee to the system by pressing the create button.

SQL Statements:

```
insert into user( name, phone-number, email, gender,
                    date-of-birth, password)
                    values(@Full_Name, @PhoneNumber, @Email,
                    @Gender, @Date_of_Birth, @Password);
select last_insert_id() as user;
insert into employee( user-id, ssn, address, salary, leave-days,
          shift-hours)
          values(user, @SSN, @Address, @Salary, @LeaveDays,
          @ShiftHours);
insert into keeper(user-id, specialty)
          values(user, @Speciality);
```

3.1.5 Coordinator Menu



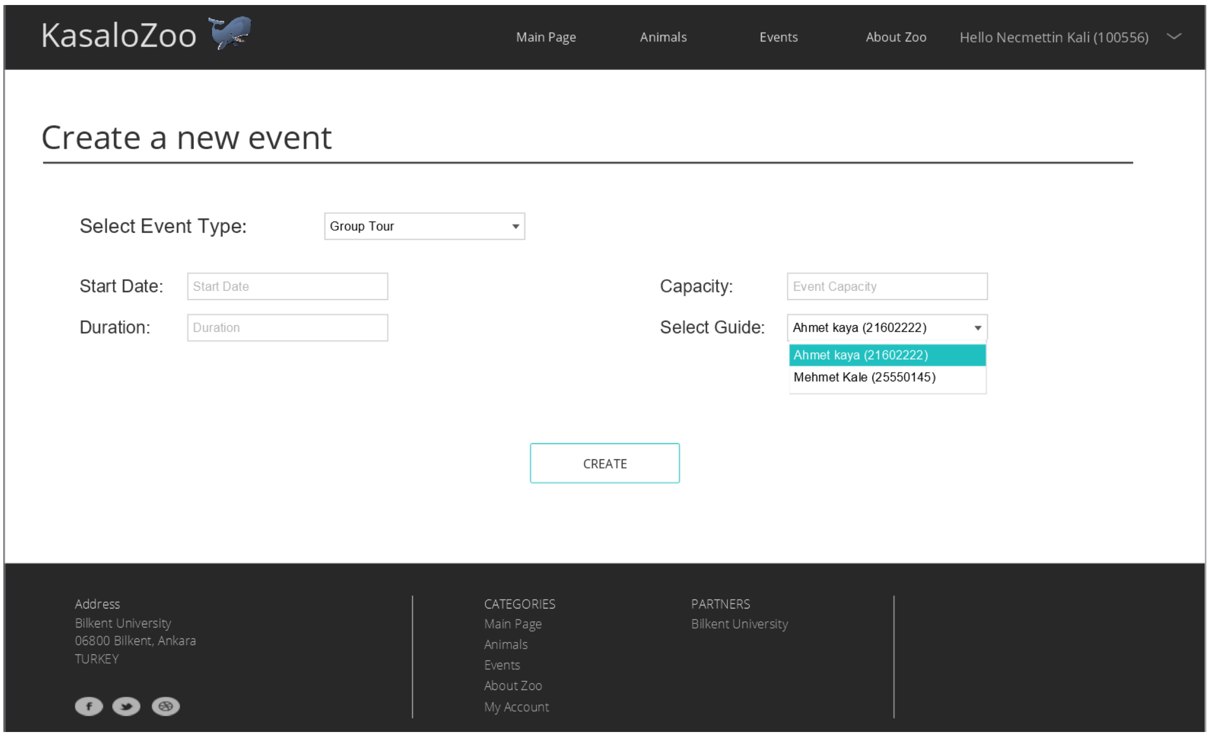
Input: None

Process: The coordinators can select various operations from this screen. From this screen, they can be redirected to the pages that enables them to view or edit their profiles, manage the zoo's cages where they can assign keepers to unattended cages or change the attended cages' keepers, create new events, view ongoing event, see and respond to complaint forms, add new employees to the system and log out from the system.

SQL Statements: These operations do not require any SQL operation.

3.1.6 Create a New Event

3.1.6.1 Create Group Tour



Create a new event

Select Event Type: Group Tour

Start Date: Start Date

Duration: Duration

Capacity: Event Capacity

Select Guide: Ahmet kaya (21602222)

Ahmet kaya (21602222)
Mehmet Kale (25550145)

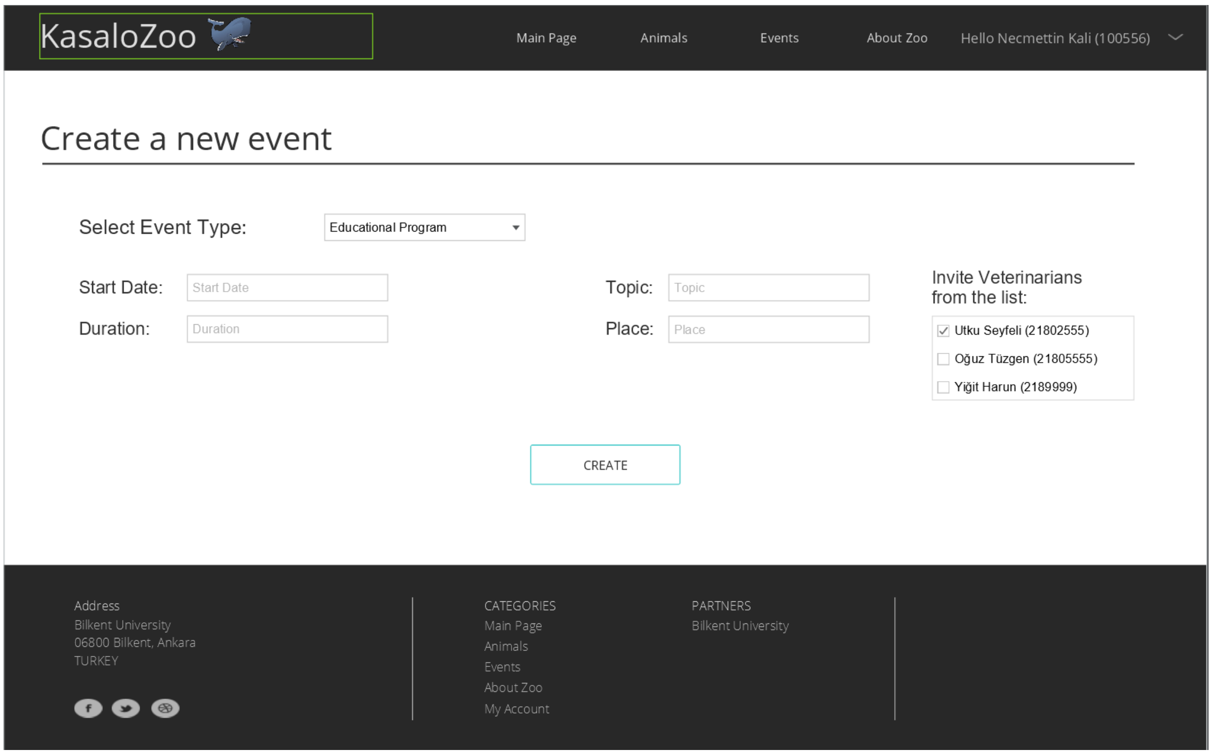
Input: @StartDate, @Duration, @Capacity, @SelectGuide

Process: The coordinators can create a group tour for the visitors and specify the information about these tours. They specify the start date, duration, capacity and the guide that is going to be in charge of the group tour. After all input fields are filled out, by pressing the create button a new group tour is created in the system.

SQL Statements:

```
select name, user-id from user u, guide g where u.user-id = g.user-id
insert into event(user-id, start-date, duration)
        values(@user-id, @StartDate, @Duration);
select last_insert_id() as event;
select event-id from event;
insert into group-tour(event-id, guide-id, capacity)
        values(event-id, @SelectGuide, @Capacity);
```

3.1.6.2 Create Educational Program



Create a new event

Select Event Type: Educational Program

Start Date: Start Date

Duration: Duration

Topic: Topic

Place: Place

Invite Veterinarians from the list:

- [x] Utku Seyfeli (21802555)
- [ ] Oğuz Tüzgen (21805555)
- [ ] Yiğit Harun (2189999)

CREATE

Address
Bilkent University
06800 Bilkent, Ankara
TURKEY

CATEGORIES
Main Page
Animals
Events
About Zoo
My Account

PARTNERS
Bilkent University

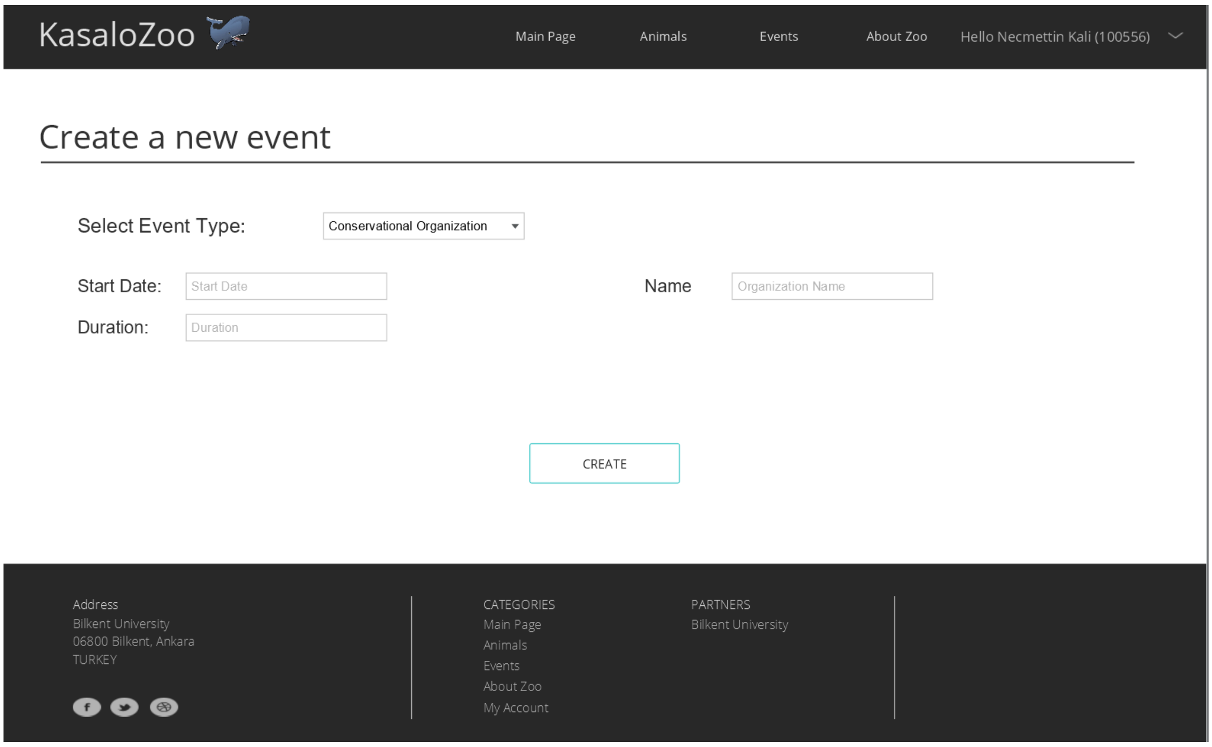
Input: @StartDate, @Duration, @Topic, @Place @Vet-id

Process: The coordinators can create an educational program for veterinarians and specify the information about these programs. They specify the start date, duration, topic and the place the program is going to take place. They can also select the veterinarians to invite the program on this page. After all input fields are filled out, by pressing the create button a new educational program is created in the system and the invitations are sent to the selected veterinarians.

SQL Statements:

```
select name, user-id from user u, veterinarian v where
        u.user-id = v.user-id
insert into event(user-id, start-date, duration)
        values(@user-id, @StartDate, @Duration);
select last_insert_id() as event;
select event-id from event;
insert into educational-program( event-id, place, topic)
        values(event-id, @Place, @Topic);
insert into invitation(vet-id, coor-id, event-id)
        values(@Vet-id, @user-id, event-id);
```

3.1.6.3 Create Conservation Organization



Create a new event

Select Event Type:	Conservational Organization		
Start Date:	Start Date	Name	Organization Name
Duration:	Duration		

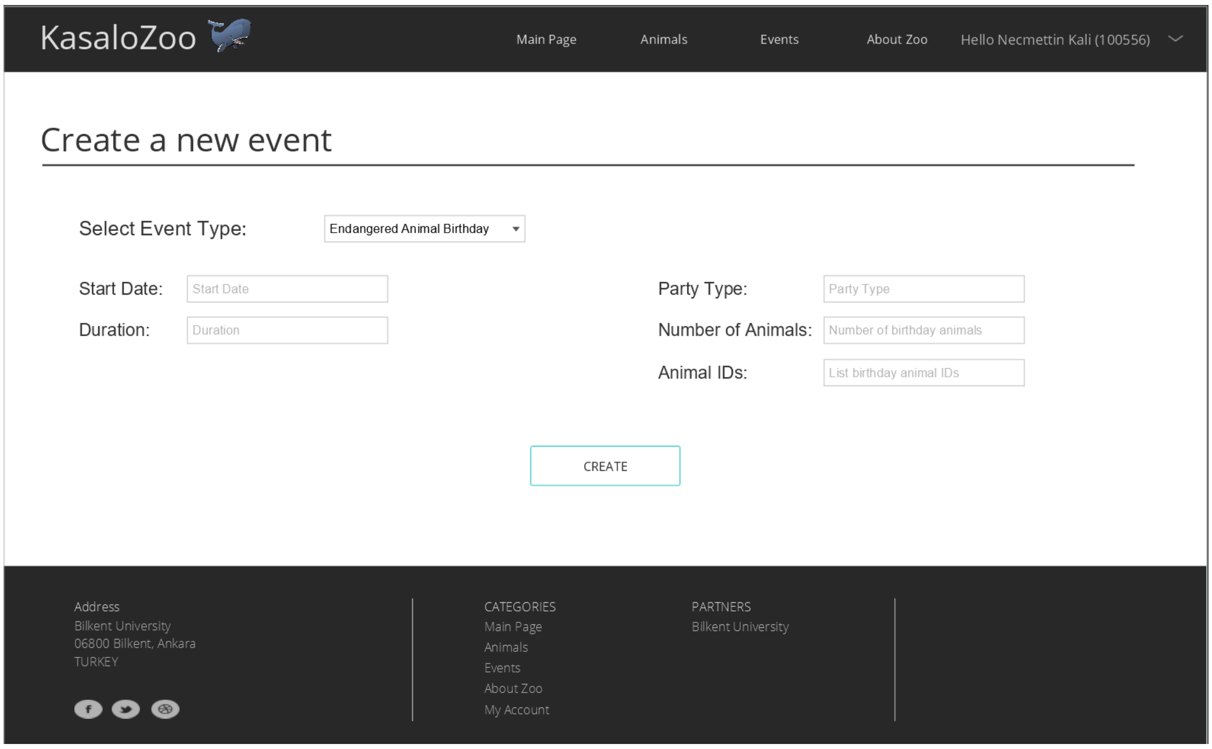
Input: @StartDate, @Duration, @Name

Process: The coordinators can create conservation organizations. They specify the start date, duration, name of the organizations on this page. After all input fields are filled out, by pressing the create button a new organization is added to the system.

SQL Statements:

```
insert into event(user-id, start-date, duration)
        values(@user-id, @StartDate, @Duration);
select last_insert_id() as event;
select event-id from event;
insert into conservation-group(event-id, name)
        values(event-id, @Name);
```

3.1.6.4 Create Endangered Animal Birthday



Create a new event

Select Event Type:	Endangered Animal Birthday		
Start Date:	Start Date	Party Type:	Party Type
Duration:	Duration	Number of Animals:	Number of birthday animals
		Animal IDs:	List birthday animal IDs

Address
Bilkent University
06800 Bilkent, Ankara
TURKEY

CATEGORIES
Main Page
Animals
Events
About Zoo
My Account

PARTNERS
Bilkent University

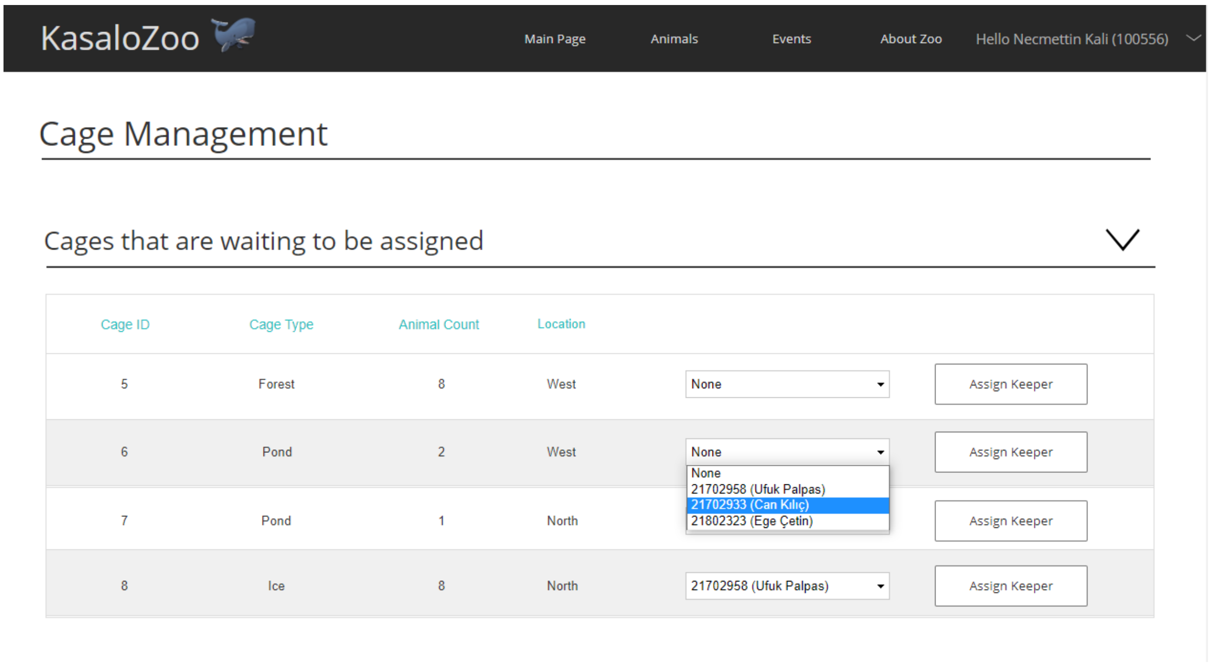
Input: @StartDate, @Duration, @PartyType, @NumOfAni, @AnimalID

Process: The coordinators can create birthday events for endangered animals. They specify the start date, duration, party type, number of animals that have a birthday and the ids' of these animals on this page. After all input fields are filled out, by pressing the create button a new birthday event is added to the system.

SQL Statements:

```
insert into event(user-id, start-date, duration)
        values(@user-id, @StartDate, @Duration);
select last_insert_id() as event;
select event-id from event;
endangered-animal-birthday( event-id, party-type,
number-of-birthday-animals)
        values(event-id, @PartyType, @NumOfAni);
insert into isb-day(animal-id, event-id)
        values(@AnimalID, event-id)
```

3.1.7 List Unassigned Cages and Assign Keeper to a Cage


KasaloZoo
Main Page
Animals
Events
About Zoo
Hello Necmettin Kali (100556)
Cage Management
Cages that are waiting to be assigned

Cage ID	Cage Type	Animal Count	Location		
5	Forest	8	West	None	Assign Keeper
6	Pond	2	West	None None 21702958 (Ufuk Palpas) 21702933 (Can Kılıç) 21802323 (Ege Çetin)	Assign Keeper
7	Pond	1	North		Assign Keeper
8	Ice	8	North	21702958 (Ufuk Palpas)	Assign Keeper

Input: @select_keeper

Process: From the cage management page coordinators can make operations about the cages and they can see the details of the cages. In the first part of the page system lists all the cages that are not assigned to any keeper and therefore, coordinators can select one of them and can see the available keepers that can be assigned to the selected cage. So that coordinators can assign keepers to the cages.

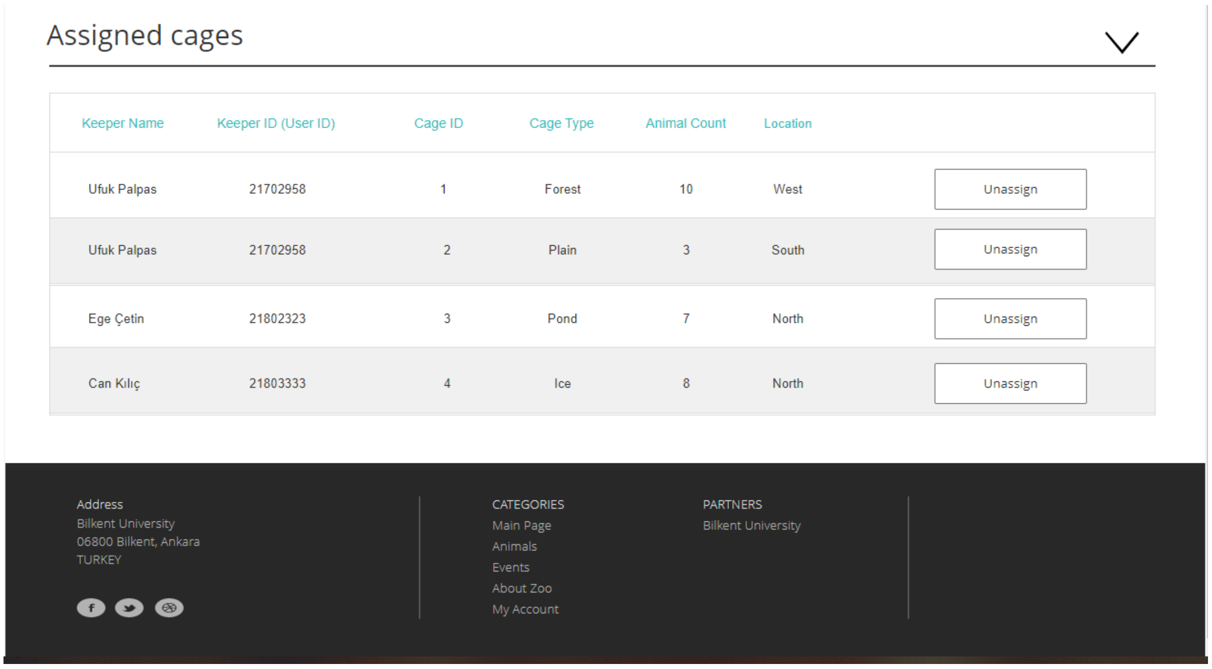
SQL Statements:

```
select C.cage-id, C.cage-type, C.animal-count, C.location
from Cage C
where C.cage-id not in (select A.cage-id
  from assigned A)

select u.name, k.user-id
from Keeper k, User u
where k.user-id = u.user-id and specialty = @cage-type

insert into assigned(keep-id, cage-id, coor-id)
  values(@select_keeper, @cage-id, @user-id)
```

3.1.8 List All Assigned Cages


Assigned cages

Keeper Name	Keeper ID (User ID)	Cage ID	Cage Type	Animal Count	Location	
Ufuk Palpas	21702958	1	Forest	10	West	Unassign
Ufuk Palpas	21702958	2	Plain	3	South	Unassign
Ege Çetin	21802323	3	Pond	7	North	Unassign
Can Kılıç	21803333	4	Ice	8	North	Unassign

Address
Bilkent University
06800 Bilkent, Ankara
TURKEY

CATEGORIES
Main Page
Animals
Events
About Zoo
My Account

PARTNERS
Bilkent University

Input: None

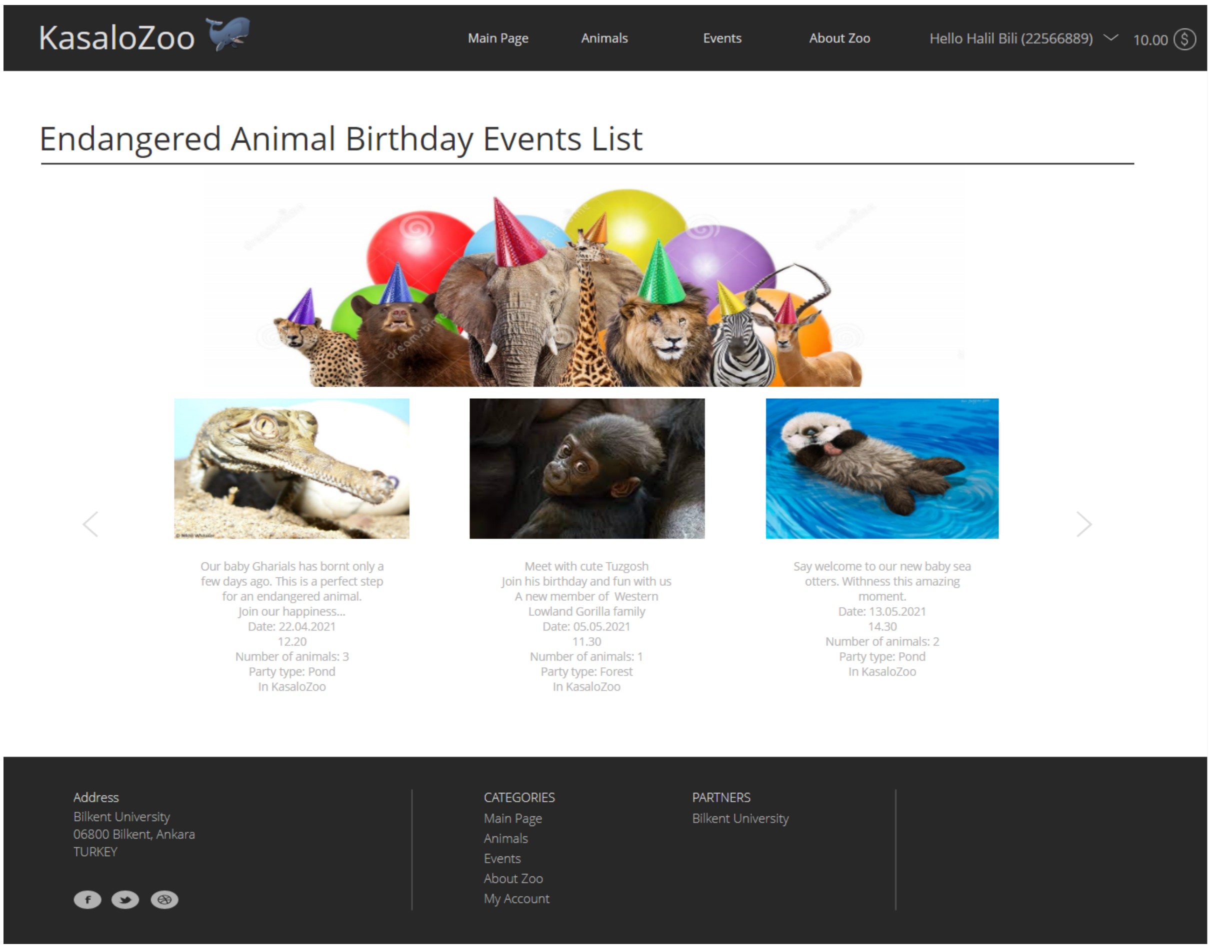
Process: After the unassigned cages part of the cage management page there is also a list of assigned cages so that coordinators can see which cage is assigned to whom. And if it's needed the coordinators can unassign the keeper from the selected/listed cage with the aid of the "Unassign" button at the right handside.

SQL Statements:

```
select u.name, k.user-id, c.cage-id, c.cage-type, c.animal-count,
c.location
from User u, Keeper k, Assigned a, Cage c
where k.user-id = u.user-id and k.user-id = a.user.id
                        and c.cage-id = a.cage.id

delete from assigned(@keep-id, @cage-id, @user-id)
```

3.1.9 Endangered Animal Birthday Events List



Endangered Animal Birthday Events List

Our baby Gharials has bornt only a
few days ago. This is a perfect step
for an endangered animal.
Join our happiness...
Date: 22.04.2021
12.20
Number of animals: 3
Party type: Pond
In KasaloZoo

Meet with cute Tuzgosh
Join his birthday and fun with us
A new member of Western
Lowland Gorilla family
Date: 05.05.2021
11.30
Number of animals: 1
Party type: Forest
In KasaloZoo

Say welcome to our new baby sea
otters. Withness this amazing
moment.
Date: 13.05.2021
14.30
Number of animals: 2
Party type: Pond
In KasaloZoo

Address
Bilkent University
06800 Bilkent, Ankara
TURKEY

CATEGORIES
Main Page
Animals
Events
About Zoo
My Account

PARTNERS
Bilkent University

Input: None

Process: Visitors can check the coming endangered animal birthdays from the endangered animals birthday page that can be accessed from the events page. All the endangered animal birthday announcements can be accessed by users, with the photos of the newborn animals.

SQL Statements:

```
select E.start-date, P.party-type, P.number-of-birthday-animals, A.name
from Event E, Endangered-Animal-Birthday P, isB-Day B, Animal A
where E.event-id = P.event-id
        and B.event-id = E.event-id
        and B.animal-id = A.animal-id
```

4. Implementation Plan

To implement the zoo management system and its functionalities and interfaces we have intention to use JavaScript, PHP, CSS and HTML. In detail to JS, we have a plan to use JQuery to handle our website's traversal, manipulation of event handlings, animation etc. Also, to sustain the data flow and management of it in our system we will use MySQL server.

5. Website

https://cs353group33.tk/