



CS 353 - Database Systems Term Project

Project Name: PetLink

Group No: 3

Design Report

15.11.2023

Instructor: Prof. Özgür Ulusoy **TA:** Mousa Farshkar Azari

Group Members:

Ahmet Alperen Yılmazyıldız - 22002712

Zeynep Doğa Dellal - 22002572

Borga Haktan Bilen - 22002733

Kardelen Ceren - 22003017

Yusuf Şenyüz - 21903105

Table of Contents

1. Design of the Database	4
1.1 Revised E/R Diagram	4
1.2 Table Schemas	6
Post	6
Reply	7
User	8
Document	9
Shelter	10
Administrator	11
Adopter	12
Veterinarian	13
Pet_Care_Info	14
Pet	15
Photo	16
Medical_Record	17
Oversee_Record	18
Appointment	19
Apply_Adopt	20
Meet_Greet	21
1.3 Views	22
User Login View	22
Administrator with Least Number of Applications View	22
Document without Document ID View	22
1.4 Triggers	22
Delete Unadopted Pets When Shelter Is Deleted	22
Adoption Status Trigger:	23
Adoption Fee Payment Trigger	23
Admin Assignment To Adoption Requests Trigger	23
2. User Interface Design and Related SQL Statements	24
2.1 Common Functionalities	24
General Login Page	24
Forgot Password - E-Mail	25
Forgot Password - Verification Code	26
Forgot Password - Reset Password	26
Register - Adopter	27
Register - Shelter	28
Register - Administrator	29
Register - Veterinarian	30
General Home Page - User Specific	31
Adopter Home Page	31
Adopter Filtering Preferences	32
Administrator Home Page	33

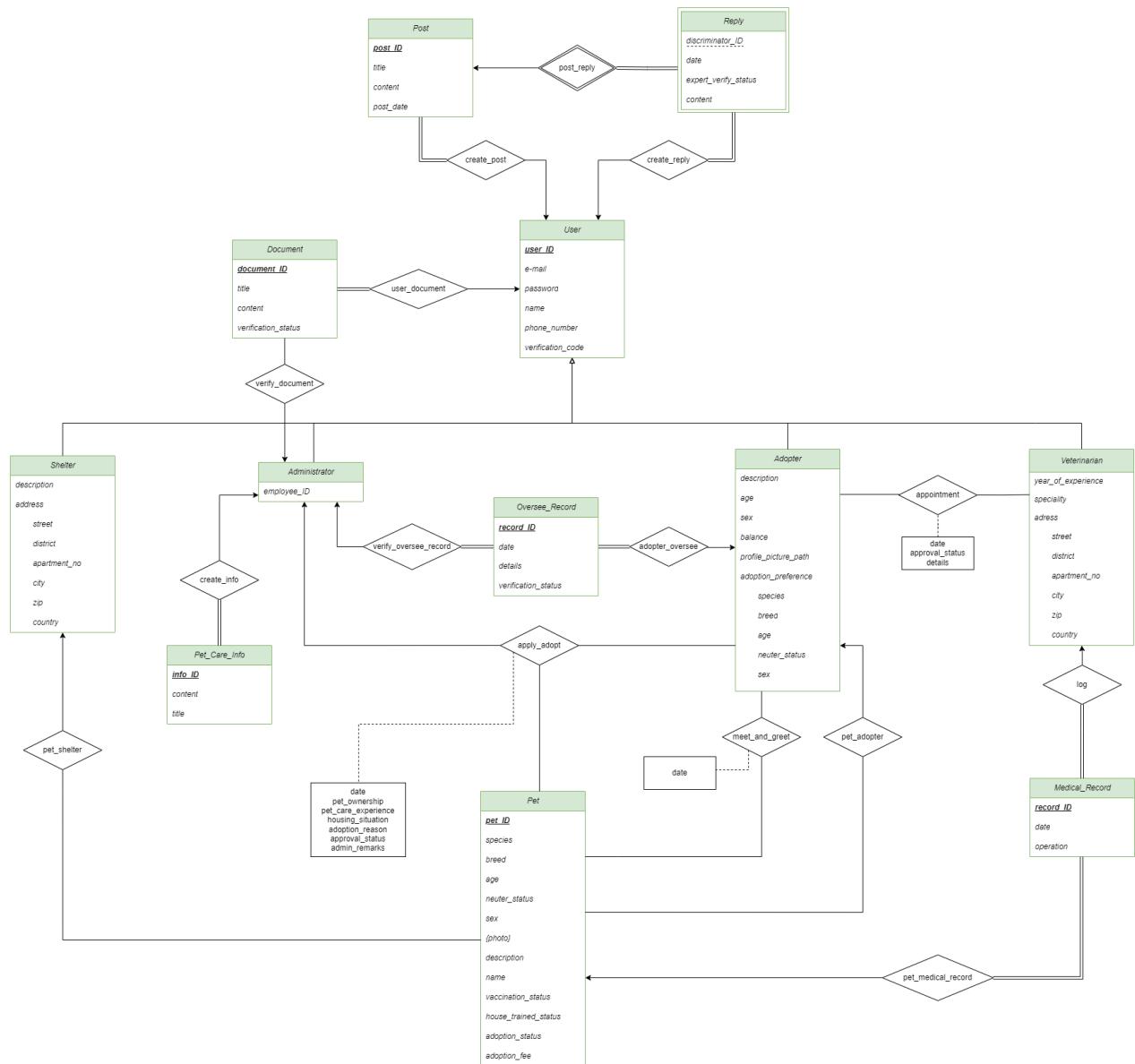
2.3 Topic Specific Functionality	34
2.3.1 Forum Pages	34
Post Page	34
Create Post Page	35
Forum Main Page	35
2.3.2 Pet Adoption Pages	36
Shelter Home Page	36
Animal Adding Page For Shelters	37
Animal Details Page	38
Application Submit Page For Adopters	39
Application List Page For Administrators	40
Application Details Page For Adopters	41
Add Balance Page	42
3. Implementation Plan	43

1. Design of the Database

1.1 Revised E/R Diagram

After our feedback session with our TA, we agreed that no changes were needed to our E/R diagram. However, given the additional features requested in the design report, we made these modifications:

- Added a balance attribute to Adopter and an adoption fee to Pet.
- Changed the apply_adopt relation to include the Administrator instead of Shelter, as specified in the instructions.
- Added a meet_and_greet relation between Adopter and Pet.
- Added more attributes to the apply_adopt relation representing adoption applications, such as the adoption_reason and housing_situation, to encourage responsible pet adoption.



Link:

https://viewer.diagrams.net/?highlight=0000ff&nav=1&title=DB-Project-ER-Diagram.drawio#Uhttps%3A%2F%2Fdrive.google.com%2Fuc%3Fid%3D1LFV8WULKTMciQ2_akvlhcV9X9ZRII274%26export%3Ddownload

1.2 Table Schemas

Post

Relational Model: Post(post_ID, poster_ID, title, content, post_date)

Functional Dependencies:

post_ID → poster_ID, title, content, post_date

Candidate Keys: {(post_ID)}

Primary Key: {(post_ID)}

Foreign Keys: poster_ID references User(user_ID)

Normal Form: BCNF

SQL Definition:

```
CREATE TABLE IF NOT EXISTS Post (
    post_ID          INT NOT NULL AUTO_INCREMENT,
    title            VARCHAR(255) NOT NULL,
    content          TEXT NOT NULL,
    post_date        DATE NOT NULL,
    FOREIGN KEY(poster_ID) REFERENCES User(user_ID)
        ON DELETE CASCADE,
    PRIMARY KEY(post_id)
);
```

Reply

Relational Model: Reply(post_ID, discriminator_ID, replier_ID, date, expert_verify_status, content)

Functional Dependencies:

post_ID, discriminator_ID → replier_ID, date, expert_verify_status, content

Candidate Keys: {(post_ID, discriminator_ID)}

Primary Key: {(post_ID, discriminator_ID)}

Foreign Keys: post_ID references Post(post_ID), replier_ID references

User(user_ID)

Normal Form: BCNF

SQL Definition:

```
CREATE TABLE IF NOT EXISTS Reply(
```

```
    post_ID              INT NOT NULL,
```

```
    discriminator_ID     INT NOT NULL AUTO_INCREMENT,
```

```
    date                DATE NOT NULL,
```

```
    expert_verify_status BOOLEAN,
```

```
    content              TEXT NOT NULL,
```

```
    FOREIGN KEY(post_ID) REFERENCES Post(post_ID)
```

```
        ON DELETE CASCADE,
```

```
    FOREIGN KEY(replier_ID) REFERENCES User(user_ID)
```

```
        ON DELETE CASCADE,
```

```
    PRIMARY KEY (post_ID, discriminator_ID),
```

```
);
```

User

Relational Model: User(user_ID, e-mail, password, name, phone_number, verification_code)

Functional Dependencies:

user_ID → e-mail, password, name, phone_number, verification_code

e-mail → user_ID, password, name, phone_number, verification_code

phone_number → user_ID, e-mail, password, name, verification_code

Candidate Keys: {(user_ID), (e-mail), (phone_number)}

Primary Key: {(user_ID)}

Foreign Keys: None

Normal Form: BCNF

SQL Definition:

```
CREATE TABLE IF NOT EXISTS User(
```

user_ID	INT NOT NULL AUTO_INCREMENT,
---------	------------------------------

password	VARCHAR(255) NOT NULL,
----------	------------------------

name	VARCHAR(255) NOT NULL,
------	------------------------

verification_code	VARCHAR(4),
-------------------	-------------

phone_number	VARCHAR(20) NOT NULL,
--------------	-----------------------

e-mail	VARCHAR(255) NOT NULL,
--------	------------------------

	UNIQUE(e-mail),
--	-----------------

	UNIQUE(phone_number),
--	-----------------------

	PRIMARY KEY (user_ID)
--	-----------------------

```
);
```

Document

Relational Model: Document(document_ID, user_ID, title, content, verification_status)

Functional Dependencies:

document_ID → user_ID, title, content, verification_status

Candidate Keys: {(document_ID)}

Primary Key: {(document_ID)}

Foreign Keys: user_ID references User(user_ID)

Normal Form: BCNF

SQL Definition:

```
CREATE TABLE IF NOT EXISTS Document(
    document_ID      INT NOT NULL AUTO_INCREMENT,
    title            VARCHAR(255) NOT NULL,
    content          TEXT NOT NULL,
    verification_status VARCHAR(50) NOT NULL,
    FOREIGN KEY(user_ID) REFERENCES User(user_ID)
        ON DELETE CASCADE,
    PRIMARY KEY (document_ID)
);
```

Shelter

Relational Model: Shelter(user_ID, description, street, district, apartment_no, city, zip, country)

Functional Dependencies:

user_ID → description, street, district, apartment_no, city, zip, country

Candidate Keys: {(user_ID)}

Primary Key: {(user_ID)}

Foreign Keys: user_ID references User(user_ID)

Normal Form: BCNF

SQL Definition:

```
CREATE TABLE IF NOT EXISTS Shelter(
    user_ID          INT NOT NULL,
    description      TEXT,
    street           VARCHAR(255) NOT NULL,
    district          VARCHAR(255) NOT NULL,
    apartment_no     VARCHAR(20) NOT NULL,
    city              VARCHAR(255) NOT NULL,
    zip               VARCHAR(20) NOT NULL,
    country           VARCHAR(255) NOT NULL,
    FOREIGN KEY(user_ID) REFERENCES User(user_ID)
        ON DELETE CASCADE,
    PRIMARY KEY (user_ID)
);
```

Administrator

Relational Model: Administrator(user_ID, employee_ID)

Functional Dependencies:

user_ID → employee_ID

Candidate Keys: {(user_ID)}

Primary Key: {(user_ID)}

Foreign Keys: user_ID references User(user_ID)

Normal Form: BCNF

SQL Definition:

```
CREATE TABLE IF NOT EXISTS Administrator(
```

```
    user_ID          INT NOT NULL,
```

```
    employee_ID      INT NOT NULL,
```

```
    FOREIGN KEY(user_ID) REFERENCES User(user_ID)
```

```
        ON DELETE CASCADE,
```

```
    PRIMARY KEY (user_ID)
```

```
);
```

Adopter

Relational Model: Adopter(user_ID, description, age, sex, balance, profile_picture_path, species, breed, adoption_age, neuter_status, adoption_sex)

Functional Dependencies:

$\text{user_ID} \rightarrow \text{description, age, sex, balance, profile_picture_path, species, breed, adoption_age, neuter_status, adoption_sex}$

Candidate Keys: { $\{\text{user_ID}\}$ }

Primary Key: { $\{\text{user_ID}\}$ }

Foreign Keys: user_ID references User(user_ID)

Normal Form: BCNF

SQL Definition:

```
CREATE TABLE IF NOT EXISTS Adopter(
    user_ID          INT NOT NULL,
    description      TEXT,
    age              INT,
    sex              VARCHAR(10),
    balance          DECIMAL(10, 2),
    profile_picture_path VARCHAR(300),
    species          VARCHAR(50),
    breed             VARCHAR(50),
    adoption_age     INT NOT NULL,
    neuter_status    VARCHAR(20),
    adoption_sex     VARCHAR(10),
    FOREIGN KEY(user_ID) REFERENCES User(user_ID)
        ON DELETE CASCADE,
    PRIMARY KEY (user_ID)
);
```

Veterinarian

Relational Model: Veterinarian(user_ID, year_of_experience, speciality, street, district, apartment_no, city, zip, country)

Functional Dependencies:

$\text{user_ID} \rightarrow \text{year_of_experience, speciality, street, district, apartment_no, city, zip, country}$

Candidate Keys: { (user_ID) }

Primary Key: { (user_ID) }

Foreign Keys: user_ID references User(user_ID)

Normal Form: BCNF

SQL Definition:

```
CREATE TABLE IF NOT EXISTS Veterinarian(
```

user_ID	INT NOT NULL,
year_of_experience	INT NOT NULL,
speciality	VARCHAR(100) NOT NULL,
street	VARCHAR(100) NOT NULL,
district	VARCHAR(100) NOT NULL,
apartment_no	VARCHAR(50) NOT NULL,
city	VARCHAR(50) NOT NULL,
zip	VARCHAR(50) NOT NULL,
country	VARCHAR(100) NOT NULL,

```
FOREIGN KEY(user_ID) REFERENCES User(user_ID)
ON DELETE CASCADE,
PRIMARY KEY (user_ID)
```

```
);
```

Pet_Care_Info

Relational Model: PetCareInfo(info_ID, administrator_ID, content, title)

Functional Dependencies:

info_ID → administrator_ID, content, title

Candidate Keys: {(info_ID)}

Primary Keys: {(info_ID)}

Foreign Keys: administrator_ID references Administrator(user_ID)

Normal Form: BCNF

SQL Definition:

```
CREATE TABLE IF NOT EXISTS PetCareInfo(
```

```
    info_ID          INT NOT NULL AUTO_INCREMENT,  

    content         TEXT NOT NULL,  

    title           VARCHAR(300) NOT NULL  

    FOREIGN KEY(administrator_ID) REFERENCES Administrator(user_ID)  

        ON DELETE CASCADE,  

    PRIMARY KEY (info_ID)
```

```
);
```

Pet

Relational Model: Pet(pet_ID, shelter_ID, adopter_ID, species, breed, age, neutered_status, sex, description, name, vaccination_status, house_trained_status, adoption_status, adoption_fee)

Functional Dependencies:

pet_ID → shelter_ID, adopter_ID, species, breed, age, neutered_status, sex, description, name, vaccination_status, house_trained_status, adoption_status, adoption_fee

Candidate Keys: {(pet_ID)}

Primary Keys: {(pet_ID)}

Foreign Keys: shelter_ID references Shelter(user_ID), adopter_ID references Adopter(user_ID)

Normal Form: BCNF

SQL Definition:

```

CREATE TABLE IF NOT EXISTS Pet(
    pet_ID          INT NOT NULL AUTO_INCREMENT,
    shelter_ID      INT,
    adopter_ID      INT,
    species         VARCHAR(150) NOT NULL,
    breed           VARCHAR(150) NOT NULL,
    age             INT NOT NULL,
    neutered_status BOOLEAN,
    sex             VARCHAR(150) NOT NULL,
    description     TEXT NOT NULL,
    name            VARCHAR(150) NOT NULL,
    vaccination_status BOOLEAN,
    house_trained_status BOOLEAN,
    adoption_status  BOOLEAN,
    adoption_fee     DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY(shelter_ID) REFERENCES Shelter(user_ID)
        ON DELETE SET NULL,
    FOREIGN KEY(adopter_ID) REFERENCES Adopter(user_ID)
        ON DELETE CASCADE,
    PRIMARY KEY (pet_ID)
);

```

Photo

Relational Model: Photo(pet_ID, photo_path)

Functional Dependencies:

none

Candidate Keys: {(pet_ID, photo_path)}

Primary Keys: {(pet_ID, photo_path)}

Foreign Keys: pet_ID references Pet(pet_ID)

Normal Form: BCNF

SQL Definition:

```
CREATE TABLE IF NOT EXISTS Photo(
    pet_ID          INT NOT NULL,
    photo_path      TEXT NOT NULL,
    FOREIGN KEY(pet_ID) REFERENCES Pet(pet_ID)
        ON DELETE CASCADE,
    PRIMARY KEY (pet_ID, photo_path)
);
```

Medical_Record

Relational Model: MedicalRecord(record_ID, pet_ID, veterinarian_ID, date, operation)

Functional Dependencies:

record_ID → pet_ID, veterinarian_ID, date, operation

Candidate Keys: {(record_ID)}

Primary Keys: {(record_ID)}

Foreign Keys: pet_ID references Pet(pet_ID), veterinarian_ID references

Veterinarian(user_ID)

Normal Form: BCNF

SQL Definition:

```
CREATE TABLE IF NOT EXISTS MedicalRecord(
    record_ID          INT NOT NULL AUTO_INCREMENT,
    pet_ID             INT NOT NULL,
    veterinarian_ID   INT NOT NULL,
    date               DATE NOT NULL,
    operation          TEXT,
    FOREIGN KEY(pet_ID) REFERENCES Pet(pet_ID)
        ON DELETE CASCADE,
    FOREIGN KEY(veterinarian_ID) REFERENCES Veterinarian(user_ID)
        ON DELETE CASCADE,
    PRIMARY KEY (record_ID)
);
```

Oversee_Record

Relational Model: OverseeRecord(record_ID, administrator_ID, adopter_ID, date, details, verification_status)

Functional Dependencies:

record_ID → administrator_ID, adopter_ID, date, details, verification_status

Candidate Keys: {(record_ID)}

Primary Keys: {(record_ID)}

Foreign Keys: administrator_ID references Administrator(user_ID), adopter_ID references Adopter(user_ID)

Normal Form: BCNF

SQL Definition:

```
CREATE TABLE IF NOT EXISTS OverseeRecord(
    record_ID          INT NOT NULL AUTO_INCREMENT,
    administrator_ID   INT NOT NULL,
    adopter_ID         INT NOT NULL,
    date               DATE NOT NULL,
    details             TEXT,
    verification_status BOOLEAN,
    FOREIGN KEY(administrator_ID) REFERENCES Administrator(user_ID)
        ON DELETE CASCADE,
    FOREIGN KEY(adopter_ID) REFERENCES Adopter(user_ID)
        ON DELETE CASCADE,
    PRIMARY KEY (record_ID)
);
```

Appointment

Relational Model: Appointment(adopter_ID, veterinarian_ID, date, approval_status, details)

Functional Dependencies:

adopter_ID, veterinarian_ID → date, approval_status, details

Candidate Keys: {(adopter_ID, veterinarian_ID)}

Primary Keys: {(adopter_ID, veterinarian_ID)}

Foreign Keys: adopter_ID references Adopter(user_ID), veterinarian_ID references Veterinarian(user_ID)

Normal Form: BCNF

SQL Definition:

```
CREATE TABLE IF NOT EXISTS Appointment(
```

adopter_ID	INT NOT NULL,
veterinarian_ID	INT NOT NULL,
date	DATE NOT NULL,
approval_status	BOOLEAN,
details	TEXT,

```
FOREIGN KEY(adopter_ID) REFERENCES Adopter(user_ID)
```

```
    ON DELETE CASCADE,
```

```
FOREIGN KEY(veterinarian_ID) REFERENCES Veterinarian(user_ID)
```

```
    ON DELETE CASCADE,
```

```
PRIMARY KEY (adopter_ID, veterinarian_ID)
```

```
);
```

Apply_Adopt

Relational Model: Apply_Adopt(adopter_ID, pet_ID, administrator_ID, date, pet_ownership, pet_care_experience, housing_situation, adoption_reason, approval_status, admin_remarks)

Functional Dependencies:

adopter_ID, pet_ID → date, pet_ownership, pet_care_experience, housing_situation, adoption_reason, approval_status, admin_remarks

Candidate Keys: {(adopter_ID, pet_ID)}

Primary Keys: {(adopter_ID, pet_ID)}

Foreign Keys: adopter_ID references Adopter(user_ID), pet_ID references Pet(pet_ID), administrator_ID references Administrator(user_ID)

Normal Form: BCNF

SQL Definition:

```

CREATE TABLE IF NOT EXISTS Apply_Adopt(
    adopter_ID          INT NOT NULL,
    pet_ID              INT NOT NULL,
    administrator_ID    INT,
    date                DATE NOT NULL,
    pet_ownership       BOOLEAN,
    pet_care_experience INT,
    housing_situation   TEXT NOT NULL,
    adoption_reason     TEXT NOT NULL,
    approval_status     BOOLEAN,
    admin_remarks        TEXT,
    FOREIGN KEY(adopter_ID) REFERENCES Adopter(user_ID)
        ON DELETE CASCADE,
    FOREIGN KEY(pet_ID) REFERENCES Pet(pet_ID)
        ON DELETE CASCADE,
    FOREIGN KEY(administrator_ID) REFERENCES Administrator(user_ID)
        ON DELETE CASCADE,
    PRIMARY KEY (adopter_ID, pet_ID)
);

```

Meet_Greet

Relational Model: Meet_Greet(adopter_ID, pet_ID, date)

Functional Dependencies:

adopter_ID, pet_ID → date

Candidate Keys: {(adopter_ID, pet_ID)}

Primary Keys: {(adopter_ID, pet_ID)}

Foreign Keys: adopter_ID references Adopter(user_ID), pet_ID references Pet(pet_ID)

Normal Form: BCNF

SQL Definition:

```
CREATE TABLE IF NOT EXISTS Meet_Greet(
    adopter_ID          INT NOT NULL,
    pet_ID              INT NOT NULL,
    date                DATE NOT NULL,
    FOREIGN KEY(adopter_ID) REFERENCES Adopter(user_ID)
        ON DELETE CASCADE,
    FOREIGN KEY(pet_ID) REFERENCES Pet(pet_ID)
        ON DELETE CASCADE,
    PRIMARY KEY (adopter_ID, pet_ID)
);
```

1.3 Views

User Login View

Purpose of this view is to store an instance of the User Login without displaying the sensitive data such as password and verification code

```
CREATE VIEW User_Login_View AS
SELECT user_ID, e-mail, name, phone_number
FROM User
```

Administrator with Least Number of Applications View

The view is used to select the administrator with the least number of applications. It is used when an administrator is assigned to a new application.

```
CREATE VIEW Administrator_with_Least_Applications AS
SELECT administrator_ID, COUNT(*) AS num_rows
FROM Apply_Adopt
GROUP BY administrator_ID
ORDER BY num_rows ASC
LIMIT 1;
```

Document without Document ID View

This view is used for concealing the private data of the user. For instance, we can get whether a user is verified or not without directly seeing the private document that he/she uploaded.

```
CREATE VIEW Document_without_Document_ID AS
SELECT user_ID, verification_status
FROM Document
```

1.4 Triggers

Delete Unadopted Pets When Shelter Is Deleted

```
CREATE TRIGGER delete_unadopted_pet
AFTER DELETE OF Shelter
REFERENCING OLD ROW AS orow
BEGIN ATOMIC
    DELETE FROM Pet
    WHERE Pet.adoption_status = false;
```

```
END;
```

Adoption Status Trigger:

```
CREATE TRIGGER change_adoption_status
AFTER UPDATE ON Apply_Adopt on (approval_status)
REFERENCING NEW ROW AS new_row
REFERENCING OLD ROW AS old_row
FOR EACH ROW
BEGIN
UPDATE Pet AS P
SET P.adoption_status = @approval_status
WHERE old_row.pet_ID = P.pet_ID
END;
```

Adoption Fee Payment Trigger

```
CREATE TRIGGER make_fee_payment
AFTER UPDATE ON Apply_Adopt on (approval_status)
REFERENCING NEW ROW AS new_row
REFERENCING OLD ROW AS old_row
FOR EACH ROW
BEGIN
UPDATE Adopter AS AD
SET AD.balance = AD.balance - (SELECT P.adoption_fee FROM Pet as P WHERE
P.pet_ID = new_row.pet_ID)
WHERE AD.user_ID = new_row.adopter_ID
END;
```

Admin Assignment To Adoption Requests Trigger

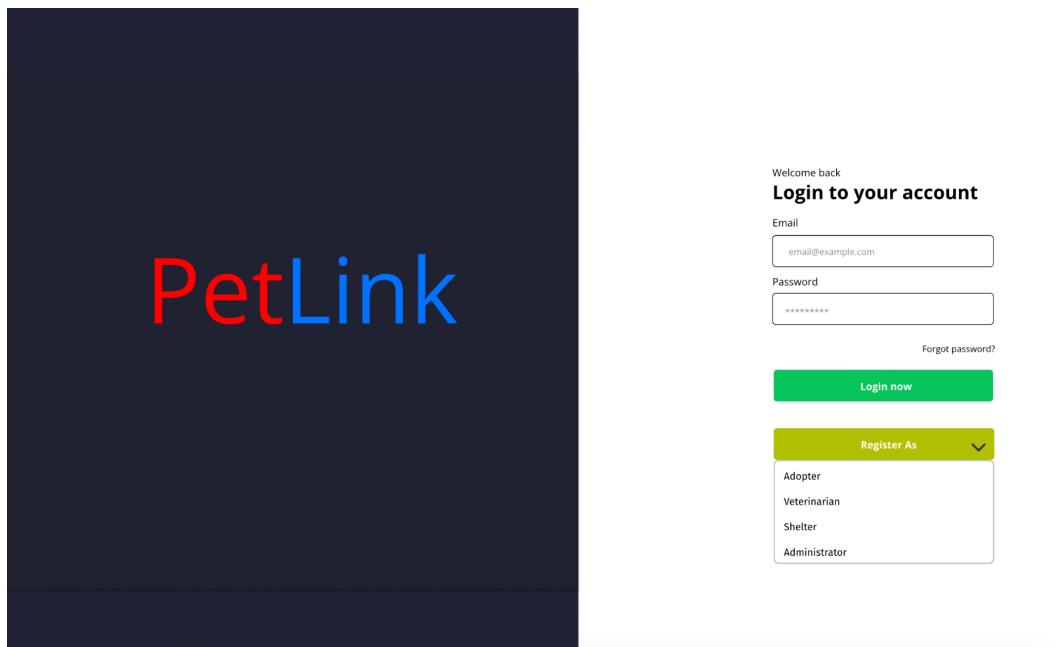
```
CREATE TRIGGER assign_admin
AFTER INSERT ON Apply_Adopt
REFERENCING NEW ROW AS nrow
REFERENCING OLD ROW AS orow
BEGIN ATOMIC
    UPDATE apply_adopt
    SET administrator_id = (SELECT administrator_ID FROM
        Administrator_with_Least_Applications)
    WHERE administrator_id = NULL;
END;
```

2. User Interface Design and Related SQL Statements

Throughout the UI sketches, we included profile pictures in all the sketches. However, it should be considered that only “Adopter” types of users have real profile pictures and remaining roles (Veterinarian, Administrator and Shelter) only have placeholders which aren't stored in the database. This design choice is made with the intention of making every scene more well-kept.

2.1 Common Functionalities

General Login Page

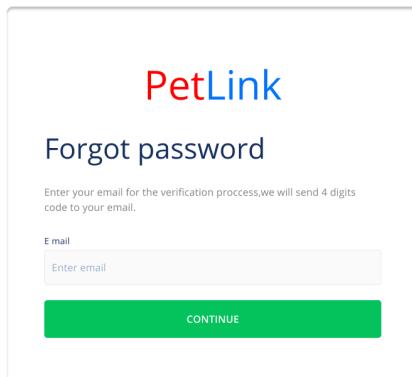


SQL Statements:

To login:

```
SELECT user_ID, name, e-mail, phone_number, verification_code
FROM User
WHERE e-mail = @email AND password = @password;
```

Forgot Password - E-Mail



SQL Statements:

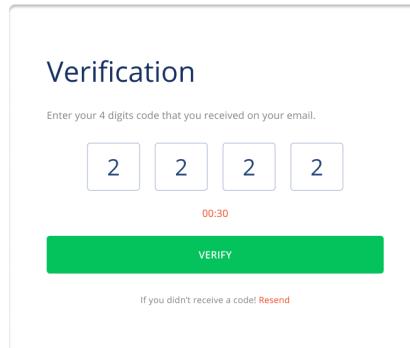
To check whether a user with the given email exists:

```
SELECT COUNT(user_ID)  
FROM user  
WHERE e-mail = @email;
```

To set the user's verification code to a code generated with internal logic:

```
UPDATE user  
SET verification_code = @generatedcode  
WHERE e-mail =@email;
```

Forgot Password - Verification Code



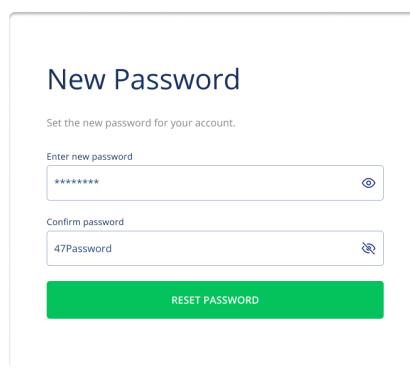
A screenshot of a verification code input interface. The title "Verification" is at the top. Below it is a placeholder text "Enter your 4 digits code that you received on your email." Underneath is a row of four input fields, each containing the digit "2". A red timer "00:30" is positioned between the second and third input fields. Below the inputs is a green "VERIFY" button. At the bottom, a small note says "If you didn't receive a code! Resend".

SQL Statements:

To verify whether the given code matches the code sent to user:

```
SELECT user_ID, e-mail
FROM user
WHERE e-mail = @email AND verification_code = @user_code;
```

Forgot Password - Reset Password



A screenshot of a new password reset form titled "New Password". It asks "Set the new password for your account." There are two input fields: "Enter new password" containing "*****" and "Confirm password" containing "47Password". Both fields have eye icon password reveal buttons. Below the inputs is a green "RESET PASSWORD" button.

SQL Statements:

To update user's password:

```
UPDATE user
SET password = @new_password
WHERE e-mail = @email;
```

To clear the user's verification code so it cannot be reused:

```
UPDATE user
SET verification_code = NULL
WHERE e-mail = @email;
```

Register - Adopter

Register - Adopter

Full Name

Email

Phone Number

Password

Re-enter Password

[Forgot password?](#)

Register

Already have an Account? [Login](#)

SQL Statements:

To add the User:

```
INSERT INTO User(user_ID, e-mail, password, name, phone_number, verification_code )
VALUES( @generatedID, @email, @password, @fullname, @phonenumber, NULL);
```

To add the Adopter:

```
INSERT INTO Adopter(user_ID, description, age, sex, balance, profile_picture_path,
species, breed, adoption_age, neuter_status, adoption_sex)
VALUES( @generatedID, NULL, NULL, NULL, 0, NULL, NULL, NULL, NULL, NULL );
```

Register - Shelter

Register - Shelter

Street

District

Building Number

Zip Code

City

Country

Shelter Name

Email

Phone Number

Password

Re-enter Password

Document for Verification
Permit.jpg

Upload

Register

Already have an Account? [Login!](#)

SQL Statements:

To add the User:

```
INSERT INTO User(user_ID, e-mail, password, name, phone_number, verification_code )
VALUES( @generatedID, @email, @password, @sheltername, @phonenumer, NULL);
```

To add the Shelter:

```
INSERT INTO Shelter(user_ID, description, street, district, apartment_no, city, zip, country)
VALUES(@generatedID, NULL, @street, @district, @buildingnumber, @city, @zipcode,
@country);
```

To update the Document:

```
INSERT INTO Document(document_ID, title, content, verification_status)
VALUES( @generatedID, @documenttitle, @documentcontent, NULL);
```

Register - Administrator

Register - Administrator

The form consists of several input fields:

- Full Name
- Email
- Employee ID
- Phone Number
- Password
- Re-enter Password

A "Document for Verification" section includes:

- File input for "ID Card.pdf"
- "Upload" button
- "Forgot password?" link

At the bottom:

- "Register" button
- "Already have an Account? [Login!](#)" link

SQL Statements:

To add the User:

```
INSERT INTO User(user_ID, e-mail, password, name, phone_number, verification_code )
VALUES( @generatedID, @email, @password, @fullname, @phonenumbers, NULL);
```

To add the Administrator:

```
INSERT INTO Administrator(user_ID, employee_ID)
VALUES(@generatedID, @employeeID);
```

To update the Document:

```
INSERT INTO Document(document_ID, title, content, verification_status)
VALUES( @generatedID, @documenttitle, @documentcontent, NULL);
```

Register - Veterinarian

Register - Veterinarian

<input type="text" value="Street"/>	<input type="text" value="Full Name"/>
<input type="text" value="District"/>	<input type="text" value="Email"/>
<input type="text" value="Building Number"/>	<input type="text" value="Year of Experience"/>
<input type="text" value="Zip Code"/>	<input type="text" value="Speciality"/> <input checked="" type="radio"/>
<input type="text" value="City"/>	<input type="text" value="Phone Number"/>
<input type="text" value="Country"/>	<input type="text" value="Password"/>
<input type="text" value="Re-enter Password"/> Forgot password?	
<input style="border: 1px solid #ccc; border-radius: 5px; padding: 2px 10px; margin-right: 10px;" type="button" value="Upload"/> <input type="file" value="Degree.pdf"/>	
<input style="background-color: #008000; color: white; border: none; border-radius: 5px; padding: 2px 10px;" type="button" value="Register"/>	
Already have an Account? Login!	

SQL Statements:

To add the User:

```
INSERT INTO User(user_ID, e-mail, password, name, phone_number, verification_code )
VALUES( @generatedID, @email, @password, @fullname, @phonenumbers, NULL);
```

To add the Veterinarian:

```
INSERT INTO Veterinarian(user_ID, year_of_experience, speciality, street, district,
apartment_no, city, zip, country)
VALUES( @generatedID, @yearofexperience, @speciality, @street, @district,
@buildingnumber, @city, @zipcode, @country);
```

To update the Document:

```
INSERT INTO Document(document_ID, title, content, verification_status)
VALUES( @generatedID, @documenttitle, @documentcontent, NULL);
```

General Home Page - User Specific

Adopter Home Page

Every Pet Deserves a Loving Home.
Adopt a Pet Today

Browse our available animals and learn more about the adoption process. Together, we can **rescue**, **rehabilitate**, and **rehome pets in need**. Thank you for supporting our mission to bring joy to families through pet adoption.

Dogs 137

Huskies 15	Labrador Retriever 20	German shepherd 25	Bull Dog 2

Cats 37

--	--	--	--

SQL Statements:

To list dog breeds available to adopt:

```
SELECT Ph.photo_path, P.breed, COUNT(DISTINCT P.pet_ID) AS count_in_website
FROM Pet P NATURAL JOIN Photo Ph
WHERE species = 'Dog' AND adoption_status = false
GROUP BY P.breed
ORDER BY P.breed
LIMIT 4;
```

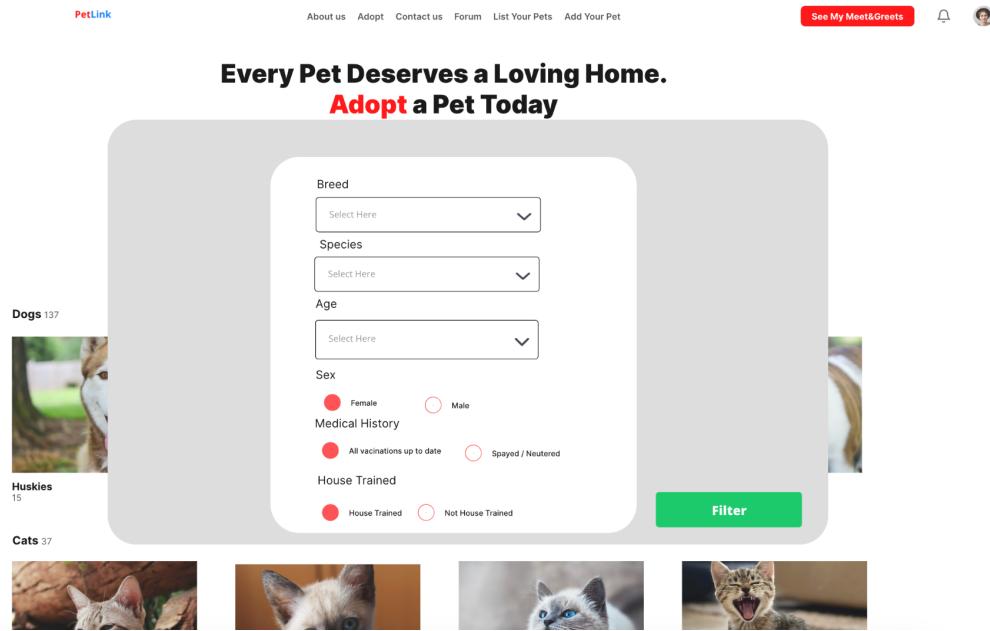
To list cat breeds available to adopt:

```
SELECT Ph.photo_path, P.species, COUNT(DISTINCT P.pet_ID) AS count_in_website
FROM Pet P NATURAL JOIN Photo Ph
WHERE species = 'Cat' AND adoption_status = false
GROUP BY P.breed
ORDER BY P.breed
LIMIT 4;
```

To show user's profile picture:

```
SELECT profile_picture_path
FROM adopter
WHERE user_ID = @user_ID
```

Adopter Filtering Preferences



SQL Statements:

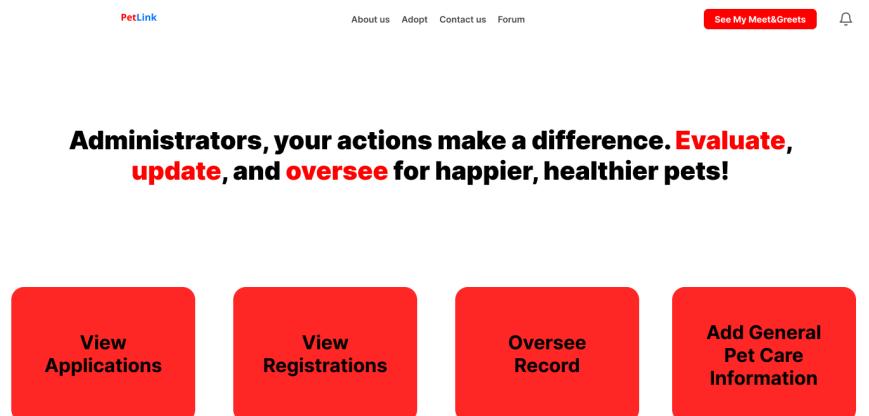
To list animal results based on search criteria:

```
SELECT pet_ID, species, breed, name, age
FROM Pet
WHERE (CASE WHEN @filter-gender = 1 THEN gender END = @chosen-gender
AND CASE WHEN @filter-neuter-status = 1 THEN neuter_status END =
@chosen-neuter
AND CASE WHEN @filter-vaccination-status = 1 THEN vaccination_status END =
@chosen-vaccination
AND CASE WHEN @filter-house-trained-status = 1 THEN house_trained_status
END = @chosen-house-trained
AND CASE WHEN @filter-sex = 1 THEN sex END = @chosen-sex);
```

To show user's profile picture:

```
SELECT profile_picture_path
FROM adopter
WHERE user_ID = @user_ID
```

Administrator Home Page



This page is designed for administrators to navigate to various sections. Therefore, no SQL statements are needed.

2.3 Topic Specific Functionality

2.3.1 Forum Pages

Post Page

The screenshot shows a forum post from 'PetLink'. The post title is 'Seeking Advice on Choosing the Perfect Furry Companion' and it is marked as 'Solved'. It was posted 23 hours ago by 'Kardelen Ceren'. The post content asks for advice on adopting a pet, mentioning the multitude of adorable faces on PetLink. Below the post, there are two bullet points: 'How did you decide on the right furry friend for your family?' and 'Did you have specific criteria or a checklist while browsing profiles?'. A reply from a 'Verified Veterinarian' provides advice on considering factors like size, energy level, and temperament. This reply has 17 comments. One comment from 'Totally.' is highlighted with a red box, stating: 'Consider factors such as size, energy level, and temperament, ensuring compatibility with your daily routine. Take into account potential allergies and the time you can dedicate to pet care. Evaluate the age of the pet and be realistic about your capacity for training and attention. Research the health considerations associated with different breeds and source your pet from reputable shelters or rescue organizations.' Other comments include responses from 'Yusuf Şenyürek', 'Zeynep Doğa Dilek', 'Kardelen Ceren', and 'Alperen Yılmazlıdız'. A 'Load more 12 comments...' link is visible at the bottom of the comment section.

SQL Statements:

To show the post's title, contents, replies, number of replies:

```
SELECT P.title, P.date, P.content, U.name, R.content, R.date, R.expert_verify_status
FROM (Post P
NATURAL JOIN Reply R)
JOIN User_Login_View U ON P.poster_ID = U.user_ID
WHERE P.post_ID = @post_ID
```

To show the number of replies:

```
SELECT COUNT(*) as reply_count
FROM Post P
NATURAL JOIN Reply R
WHERE P.post_ID = @post_ID
```

To reply to a post:

```
INSERT INTO Reply(date, expert_verify_status, content)
VALUES(GETDATE(), CASE WHEN EXISTS (SELECT 1 FROM Veterinarian WHERE user_ID = @user_ID) THEN 1 ELSE 0 END, @content)
```

Create Post Page

Create Post

Post Title
Seeking Advice on Choosing the Perfect Furry Companion

Post Details

I've been pondering the idea of adopting a pet lately, and the multitude of adorable faces on PetLink is making it a tough decision. Could use some guidance from those who've been through the adoption process.

- How did you decide on the right furry friend for your family?
- Did you have specific criteria or a checklist while browsing profiles?

Submit Post

SQL Statements:

To add a new post:

```
INSERT INTO Post(post_ID, poster_ID, title, content, post_date)
VALUES(@post_ID, @user_ID, @title, @content, GETDATE())
```

Forum Main Page

Forum

Posted by Kardelen Ceren 23 hours ago
Seeking Advice on Choosing the Perfect Furry Companion

I've been pondering the idea of adopting a pet lately, and the multitude of adorable faces on PetLink is making it a tough decision. Could use some guidance from those who've been through the adoption process. How did you decide on the right furry friend for your family? Did you have specific criteria or a checklist while browsing profiles?...

57 Comments

Posted by Alperen Yilmaz 6 hours ago
Pet Healthcare Real Talk: Navigating the Ups and Downs

I'm not gonna lie; this whole pet healthcare thing has me feeling like I'm on the struggle bus. From figuring out what's in their food to...

SQL Statements:

To list posts, poster's user name and their reply number:

```
SELECT P.title, P.date, P.content, U.name, COUNT(R.discriminator_ID) as reply_count
FROM (Post P
NATURAL JOIN Reply R)
JOIN User_Login_View U ON P.poster_ID = U.user_ID
GROUP BY P.post_ID, P.title, P.date, P.content, U.name
ORDER BY P.date DESC
```

2.3.2 Pet Adoption Pages

Shelter Home Page

Category	Name	Count
Dogs	Huskies	15
Dogs	Labrador Retriever	20
Dogs	German shepherd	25
Dogs	Bull Dog	2
Cats	Cats	37
Cats	Siamese Cat	1
Cats	Siamese Cat	1
Cats	Siamese Cat	1

SQL Statements:

To search animals based on filtering for shelters:

```
SELECT *
FROM Pet
WHERE (CASE WHEN @filter-gender = 1 THEN gender END = @chosen-gender
        AND CASE WHEN @filter-neuter-status = 1 THEN neuter_status END =
@chosen-neuter
        AND CASE WHEN @filter-vaccination-status = 1 THEN vaccination_status END =
@chosen-vaccination
        AND CASE WHEN @filter-house-trained-status = 1 THEN house_trained_status END =
@chosen-house-trained
        AND CASE WHEN @filter-adoption-status = 1 THEN adoption_status END =
@chosen-adoption)
```

```

AND CASE WHEN @filter-sex = 1 THEN adoption_status END =
@chosen-sex
AND CASE WHEN @filter-adoption-status = 1 THEN adoption_status END =
@chosen-adoption);

```

Animal Adding Page For Shelters

Enter the details of the animal

Name
Enter name

Breed
Select Here

Age
Select Here

Sex
 Female Male

Medical History
 All vaccinations up to date Spayed / Neutered

House Trained
 House Trained Not House Trained

Add details
Details about the animal

Adoption Fee
Adoption fee in dollars

Upload Photo
Choose a file or drag & drop it here
JPEG, PNG, PDF, and MP4 formats, up to 50MB
Browse File

Add Animal

SQL Statements:

To add new pet:

```

INSERT INTO Pet(pet_ID, shelter_ID, adopter_ID, species, breed, age, neutered_status,
sex, description, name, vaccination_status, house_trained_status, adoption_status,
adoption_fee)
VALUES (@pet_ID, @user_ID, NULL, @species, @breed, @age, @neuter_status, @sex,
@description, @name, @vaccination_status, @house_trained_status, 0, @adoption_fee);

```

To add pet's photos (one statement for each photo):

```

INSERT INTO Photo(pet_ID, photo_path)
VALUES(@pet_ID, @photo_path)

```

Animal Details Page

The screenshot shows a form for an animal named Luna. The fields include:

- Shelter:** Happy Homes
- Name:** Luna
- Breed - Species:** Labradoodle - Dog
- Age:** 1 years old
- Sex:** Female
- Neuter Status:** Neutered
- Details:** Luna is a very friendly dog. She loves little kids.
- House Trained?**: Yes (selected)
- Fully Vaccinated?**: No (selected)
- Photo of the Pet:** A photo of a brown Labradoodle.
- Apply for adoption!** button
- Adoption Fee:** 100 \$
- Arrange Meet & Greet!** button with a calendar overlay showing October 2023. The 17th is selected.
- Enter Time** input field showing 00:00 AM.

SQL Statements:

To arrange meet and greet:

```
INSERT INTO Meet_Greet(adopter_ID, pet_ID, date)
VALUES(@adopter_ID, @pet_ID, @meet_date)
```

To display pet information:

```
SELECT *
FROM Pet NATURAL JOIN Photo
WHERE pet_ID = @pet_ID
```

Application Submit Page For Adopters

The screenshot shows a web form for adopting a pet. At the top, there's a navigation bar with links for 'About us', 'Adopt', 'Contact us', 'Forum', 'List Your Pets', and 'Add Your Pet'. On the right, there are buttons for 'See My Meet&Greets' (red), a notification bell, and a user profile icon.

Application Detail

Age: A dropdown menu labeled 'Select Here' with a downward arrow icon.

Sex: Radio buttons for 'Female' (selected), 'Male', and 'Other'.

Do you have other pets? Radio buttons for 'Yes' (selected) and 'No'.

For how many years have you been a pet owner? An input field with placeholder text 'Enter a number'.

Can you share your current housing situation? A text area with placeholder text: 'Please write the type of housing you have (apartment, single family, farm etc.), how many adults and children reside, where will your pet sleep, etc.'.

Why do you want to adopt? A text area with placeholder text: 'Please explain why you want to adopt and who the primary caregivers of your pet will be.'

Submit Application (green button)

Note: Adoption Fee: 100 \$, Your Balance: 230 \$. Note: The adoption fee will be paid from your balance automatically after the acceptance of your application!

Enter your information to adopt
Your application will be evaluated and we will contact you.

1/5 >

You are adopting: Luna from HappyHomes shelter

SQL Statements:

To show user's age, sex and balance:

This is done in order to be updated if any of these attributes of an Adopter changes in any time.

```
SELECT age, sex, balance
FROM adopter
WHERE user_ID = @user_ID;
```

To show pet's name, adoption fee, shelter name, and photos:

```
SELECT P.name, P.adoption_fee, U.name, Ph.photo_path
FROM pet P
NATURAL JOIN photo Ph
JOIN User_Login_View U ON P.shelter_ID = U.user_ID
WHERE P.pet_ID = @pet_ID ;
```

To update user's age and sex:

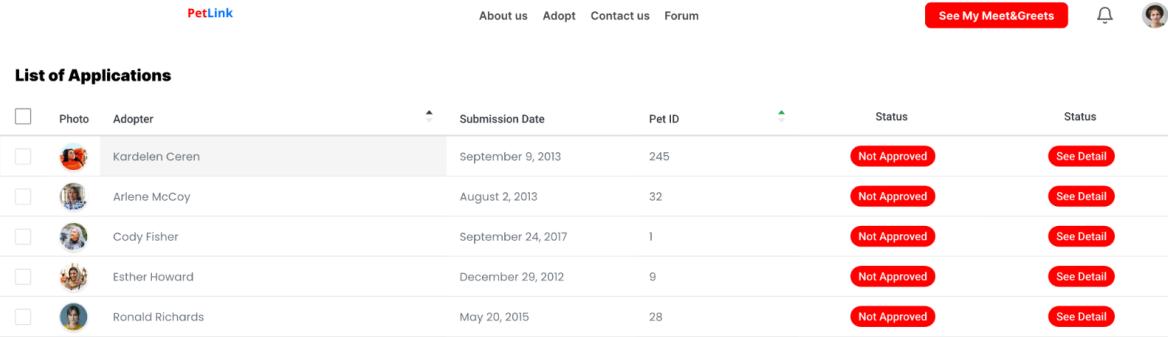
```
UPDATE user
SET age = @age AND sex = @sex
WHERE user_ID = @user_ID;
```

To submit an application:

```
INSERT INTO Apply_Adopt(adopter_ID, pet_ID, administrator_ID, date,
pet_ownership, pet_care_experience, housing_situation, adoption_reason,
approval_status, admin_remarks)
```

```
VALUES(@user_ID, @pet_ID, NULL, GETDATE(), @pet_ownership,
@pet_care_experience, @housing_situation, @adoption_reason, 0, NULL)
```

Application List Page For Administrators



<input type="checkbox"/> Photo	Adopter	Submission Date	Pet ID	Status	Status
<input type="checkbox"/>	Kardelen Ceren	September 9, 2013	245	Not Approved	See Detail
<input type="checkbox"/>	Arlene McCoy	August 2, 2013	32	Not Approved	See Detail
<input type="checkbox"/>	Cody Fisher	September 24, 2017	1	Not Approved	See Detail
<input type="checkbox"/>	Esther Howard	December 29, 2012	9	Not Approved	See Detail
<input type="checkbox"/>	Ronald Richards	May 20, 2015	28	Not Approved	See Detail

SQL Statements:

To list applications:

```
SELECT A.profile_picture_path, U.name, AA.date, AA.pet_ID, AA.approval_status
FROM Apply_Adopt AA
JOIN User_Login_View U ON U.user_ID = AA.adopter_ID
JOIN Adopter A ON U.user_ID = A.user_ID
WHERE AA.administrator_ID = @user_ID;
```

Application Details Page For Adopters

Application Detail

Name: Kardelen Ceren
Email: kardelenceren@testemail.com
Age: 21
Sex: Female
Other pets: Yes
Pet care experience: 3 Years
Housing situation: Single family home, 3 adults and no children
Adoption reason: I love dogs! I am looking for a companion for me and for my 5 year old dog. I will be the main caregiver.
Administrator's remarks: Please explain why you are accepting or declining this application.

Requested pet

Luna from HappyHomes Shelter

Go to Dog Detail Page

Accept | Decline | Delete Application!

SQL Statements:

To show user's name, email, age, sex, application date, other pets, pet care experience, housing situation and adoption reason:

```
SELECT U.name, U.email, A.age, A.sex, App.date, App.pet_ownership,
App.pet_care_experience, App.housing_situation, App.adoption_reason
FROM adopter A
NATURAL JOIN User_Login_View U
JOIN apply_adopt App ON U.user_ID = App.adopter_ID
WHERE App.adopter_ID = @adopter_ID AND App.pet_ID = @pet_ID;
```

To show pet's name, shelter name and photos:

```
SELECT P.name, U.name, Ph.photo_path
FROM pet P
NATURAL JOIN photo Ph
JOIN User_Login_View U ON P.shelter_ID = U.user_ID
WHERE P.pet_ID = @pet_ID ;
```

To update the application's status and remarks:

```
UPDATE apply_adopt
SET approval_status = @approval_status AND admin_remarks = @admin_remarks
WHERE App.adopter_ID = @adopter_ID AND App.pet_ID = @pet_ID;
```

Delete application:

```
DELETE from apply_adopt
WHERE adopter_ID = @adopter_ID AND pet_ID = @pet_ID;
```

Add Balance Page

PetLink

About us Adopt Contact us Forum List Your Pets Add Your Pet

See My Meet&Greets

Add Balance

Name on Card
Borga Haktan Bilen

Credit Card Number
1234-5678-9012-3456

Expire Date
29/02/2025

CVV

Top-up Amount
150 \$

Pay

SQL Statements:

To add balance:

```
UPDATE adopter
SET balance = balance + @top_up_amount
WHERE user_ID = @user_ID;
```

3. Implementation Plan

We will use the MySQL database system as our database management system, taking advantage of its broad support for the latest features required for our project. The backend of the application will be built with the Python Flask framework, while the frontend will be built with the React.JS framework and TypeScript. We'll integrate the Tailwind CSS framework and employ Material UI components to improve the appearance and user experience. Notably, all database-application interactions will be limited to SQL queries, with no automation tools or libraries such as ORM used.