PET SIGN IN WEB APPLICATION

CS-670-IA RESEARCH PROJECT SEMINAR

SACRED HEART UNIVERSITY

5151 PARK AVENUE, FAIRFIELD CT 06825-1000

FRIDAY, MARCH 18TH, 2016

ALI B. KABA

MASTER OF SCIENCE IN CYBERSECURITY

GREGORY KYRYTSCHENKO

Contents

Abstract

Pet Sign In is a web application to sign in pets at a work environment. The idea behind this web application is to do two things:

1. Anyone with a pet will be required to abide to the pet policy.
2. Keep an audit trail on the pets and their owners.

Introduction

2.1 Background

At my old job pet friendly, we were required do the following:

1. Fill in a form about our pets
2. Send the pet documentation to the Human Resource team for approval
3. Once approved, the front desk would require us to fill out a form requiring the following:
    a. Name
    b. Pet's Name
    c. Shots up to date check
    d. Understanding of the dog policy
    e. Signature

I got tired of doing the same process over and over so I thought of making a web application that requires minimum work from Human Resource, wasted paper and stress free.

2.2 Goals and Objectives

This project's goal is to simplify the process of getting your dog registered and signed in.

- An employee can:
    - Create an account
    - Sign into their account once activated
    - Sign in their pet(s)
    - View their account activities.
    - Change their password
    - Reset their password
- An employee administrator can:
    - Manage accounts
        - Enable an account
        - Disable an account
        - Enable a pet
        - Disable a pet
        - Update a pet's name
        - Update a pet's breed
        - Update a pet's gender
    - Manage breeds
        - Update a breed's name
        - Add a breed
    - View their account activities.
    - Change their password
    - Reset their password
- A super administrator can:
    - View the website's code errors
    - Add administrator
    - Upload pet policy

The following is a list of opportunities this product will bring to dog friendly places:

- Free to use
- Mobile friendly website
- Strong audit trail
- Fast pet sign in process
- Using latest security to protect data in transit and at rest

2.3 Scope
This website is a pet sign in application installed in a pet friendly local environment. It will only be accessible via the company's intranet and not made available on the internet. The idea is to facilitate the sign in process of bringing in your pet to a pet friendly environment.

2.4 Intended Audience
The audience for this document is anyone looking to understand this project and how it is designed.

2.5 Computer Application
- Agent Ransack
- Fiddler4
- FileZilla
- Google Chrome
- Internet Explorer 11
- Microsoft Excel, PowerPoint, Visio and Word
- MySQL Workbench
- Notepad ++
- PuTTY
- IntelliJ Idea

2.6 Web Application
- CPanel
- Dropbox
- GitHub
- JSFiddle
- Runnables

2.7 Languages
- HTML 5
- JavaScript
- PHP
- MySQL

2.8 Libraries
- Bootstrap
- jQuery
- Joyride

3 Analysis Overview

3.1 System Usage

3.2 Assumptions, Dependencies and Constraints

The availability of the website will depend on the service availability of the hosting company.

3.3 Development Methods

Machine	
OS Name	Microsoft Windows 10 Pro
Version	1511
OS Manufacturer	Microsoft Corporation
System Manufacturer	INTEL
System Model	DZ68BC
System Type	x64-based PC
Processor	Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz, 3401 Mhz, 4 Core(s), 8 Logical Processor(s)
BIOS Version/Date	Intel Corp. BCZ6810H.86A.0021.2011.0831.1555, 8/31/2011
SMBIOS Version	2.6
Total Physical Memory	16.0 GB
Total Virtual Memory	32.0 GB
C Drive	238 GB

4 Requirements

4.1 Inputs – Data

Visitors	
Email	Extension must match company email address
Password	Password must be between 8 and 15 characters long.
	Password must be different from your email.
	Password must contain at least one number (0-9).
	Password must contain at least one lowercase letter (a-z).
	Password must contain at least one uppercase letter (A-Z).
Users	
Email	Extension must match company email address
Password	Password must be between 8 and 15 characters long.
	Password must be different from your email.
	Password must contain at least one number (0-9).
	Password must contain at least one lowercase letter (a-z).
	Password must contain at least one uppercase letter (A-Z).
Pet's name	Pet name must not already exist.
Pet's document	Document must not already exist.
	Document must have .PDF extension.
	Document must be 500kb or lower.
	Pet name must exist.
	Email address must exist.

	Document must not already exist.
Administrators	
Email	Extension must match company email address
Password	Password must be between 8 and 15 characters long.
	Password must be different from your email.
	Password must contain at least one number (0-9).
	Password must contain at least one lowercase letter (a-z).
	Password must contain at least one uppercase letter (A-Z).
Pet's name	Pet name must not already exist.
Breed name	Breed name must not already exist.
Server Administrators	
Email	
Password	

4.2 Outputs – Information

Account Activity

Activity Messages

Error logging:

Error Messages	
Error 0	(PHP errors, varies)
Error 1	Oops, something went wrong. Contact an administrator with this error message.
Error 2	Oops, something went wrong. Contact an administrator with this error message.
Error 3	Oops, something went wrong. Contact an administrator with this error message.
Error 10	Please enter a valid GMAIL e-mail address (your.name@gmail.com).
Error 11	Not a valid e-mail address.
Error 15	Your account is not activate. Wait or contact an Admin.
Error 19	This account doesn't exist. Please click on "Register for a new account".
Error 20	Password must be between 8 and 15 characters long
Error 21	Password must be different from your email.
Error 22	Password must contain at least one number (0-9).
Error 23	Password must contain at least one lowercase letter (a-z).
Error 24	Password must contain at least one uppercase letter (A-Z).
Error 25	Invalid email and/or password. If you forgot your password, reset it.
Error 26	Please fill all of the fields.
Error 27	Your old passwords don't match and/or they match with your new password.
Error 28	Your old passwords don't match and/or they match with your new password.
Error 29	Your old passwords don't match and/or they match with your new password.
Error 50	This breed already exist.
Error 51	You already have a pet name (pet name). Please pick a different name.
Error 60	Sorry, file already exists. Ask an admin to remove it before uploading a new file.

Error 61	Sorry, your file is too large.
Error 62	Sorry, only PDF files are allowed.
Error 63	Sorry, you don't have a pet named (pet name).
Error 64	Sorry, your file was not uploaded.
Error 65	Sorry, there was an error uploading your file.
Error 87	Your account will be locked out soon.
Error 89	This account has been locked. Reset your account or contact the administrator.
Error 99	Your session expired, please sign in again.

4.3 Storage – Database

- All data will be stored in a MySQL Database.

4.4 Control – Interfaces

The website will consist of four roles:

- Visitor
- User.
- Administrator.
- Server administrator.

Each will have distinct rights and ability throughout the application.

4.5 Timelines and deadlines

Phase	Start Date	End Date
Iteration 1		
Phase	Start Date	End Data
Inception	05/17/2015	11/03/2015
Iteration 2		
Elaboration	11/04/2015	11/29/2015
Construction	11/30/2015	12/07/2015
Iteration 3		
Inception	12/08/2015	12/09/2015
Elaboration	12/09/2015	12/09/2015
Construction	12/09/2015	12/09/2015
Transition		
Iteration 4		
Inception	12/09/2015	12/15/2015
Elaboration	12/09/2015	12/30/2015
Construction	12/09/2015	12/30/2015
Transition	12/09/2015	12/30/2015
Iteration 5		
Inception	12/31/2015	12/31/2015
Elaboration	12/31/2015	2/25/2015
Construction	12/31/2015	2/25/2015
Transition	12/31/2015	2/25/2015
Iteration 6		
Construction	2/25/2015	3/11/2015

Transition	2/25/2015	3/11/2015

4.6 Use Cases

4.6.1 Register

Use Case Name	**Register 1.0**
Description	Create an account.
Actors	Visitors.
Pre-Conditions	Valid Gmail email account.
Basic Flow	1. Go to the website. 2. Click on the "Register for a new account" link. 3. Enter the email address in the "Email address" field. 4. Enter the password in the "Password" field. 5. Click on "Terms and Conditions" and read it. 6. Click on "I agree" radio button. 7. Click on the "Register" button.
Post Conditions	- A successful registration will send an email to the newly registered email, informing them that an admin will need to approve the account first.
Alternate Flows	- Clicking on the "I do not agree" radio button will disable the "Register" button. - Leaving empty text fields will give you Error 26. - Not using a gmail.com email address will give you Error 10. - Using an invalid email will give you Error 11. - Passwords shorter than 6 characters long will give you Error 20. - Password matching the email address will give you Error 21. - Password not containing one number (0-9) will give you Error 22. - Password not containing one lowercase letter (a-z) will give you Error 23. - Password not containing one uppercase letter (A-Z) will give you Error 24. - Registering with an already registered account will give you Error 87. - Registering with an already registered account will give you Error 89. - Any other problem you experience will give you Error 1 or Error 2.
Notes	None.

4.6.2 Sign in pet

Use Case Name	**Sign in pet 2.0**
Description	Sign in pet.
Actors	Users.
Pre-Conditions	Enabled account and pet.
Basic Flow	1. Go to the website. 2. Sign in. 3. Click on the pet's name.
Post Conditions	None.
Alternate Flows	- An expired session will give you Error 99. - Any other problem you experience will give you Error 1 or Error 2.
Notes	None.

4.6.3 Add pet

Use Case Name	**Add Pet 3.0**

Description	Add a pet.
Actors	Users.
Pre-Conditions	Enabled account.
Basic Flow	1. Go to the website. 2. Sign in. 3. Click on the “Add a pet” button. 4. Enter the pet’s name. 5. Select pet’s breed. 6. Select pet’s gender. 7. Click on “I agree” radio button. 8. Click on the “Add Pet” button.
Post Conditions	- A successful registration will send an email to the pet’s owner with information on use case name “Upload pet document”.
Alternate Flows	- Clicking on the “I do not agree” radio button will disable the “Add Pet” button. - Leaving empty text fields will give you Error 26. - An expired session will give you Error 99. - Any other problem you experience will give you Error 1, Error 2 or Error 3.
Notes	None.

4.6.4 Upload pet document

Use Case Name	**Upload pet document 4.0**
Description	Upload pet vaccination and rabies documentations.
Actors	Users.
Pre-Conditions	Enabled account.
Basic Flow	1. Go to the upload website. 2. Enter the email address in the “Email address” field. 3. Enter the pet’s name. 4. Click on the “Choose File” button. 5. Select pet’s PDF required document. 6. Click on the “Upload Document” button.
Post Conditions	None.
Alternate Flows	- Leaving empty text fields will give you Error 26. - Uploading any file after a successful upload will give you Error 60. - Uploading a file that is larger than 500kb will give you Error 61. - Uploading a file that isn’t a PDF will give you Error 62. - Uploading a file for a non-existent pet name will give you Error 63. - A failed upload will give you error 64. - Any other upload problems will give you Error 65. - Any other problem you experience will give you Error 1 or Error 2.
Notes	None.

4.6.5 Reset Password

Use Case Name	**Reset password 5.0**
Description	Reset password.
Actors	Users and administrators.
Pre-Conditions	Registered user or administrator email address
Basic Flow	1. Go to the website. 2. Click on the “Can’t access your account?” link.

	3. Enter the email address in the “Email address” field. 4. Click on the “Reset” button.
Post Conditions	- An email will be sent to the user with the new password.
Alternate Flows	- Leaving empty text fields will give you Error 26. - Entering a non-existent email will give you Error 19. - Any other upload problems will give you Error 65. - An expired session will give you Error 99. - Any other problem you experience will give you Error 1 or Error 2.
Notes	None.

4.6.6 Update password

Use Case Name	**Update password 6.0**
Description	Update your password.
Actors	Users and administrators.
Pre-Conditions	Enabled account.
Basic Flow	1. Go to the website. 2. Sign in. 3. Click on the “Account” menu. 4. Click on the “Change password” button. 5. Enter your current password in the “Old password” field. 6. Enter your current password in the “Old password again” field. 7. Enter your new password in the “New password” field. 8. Click on the “Change password” button.
Post Conditions	None.
Alternate Flows	- Leaving empty text fields will give you Error 26. - The two old passwords not matching will give you Error 27. - Old passwords not matching with the current password will give you Error 28. - Old password fields matching new password field will give you Error 27. - Passwords shorter than 6 characters long will give you Error 20. - Password matching the email address will give you Error 21. - Password not containing one number (0-9) will give you Error 22. - Password not containing one lowercase letter (a-z) will give you Error 23. - Password not containing one uppercase letter (A-Z) will give you Error 24. - Registering with an already registered account will give you Error 89. - Any other upload problems will give you Error 65. - An expired session will give you Error 99. - Any other problem you experience will give you Error 1 or Error 2.
Notes	None.

4.6.7 Sign in

Use Case Name	**Sign In 7.0**
Description	Sign into the website.
Actors	Users and administrators.
Pre-Conditions	Enabled account.
Basic Flow	1. Enter the email address in the “Email address” field. 2. Enter the password in the “Password” field. 3. Click on the “Sign in” button.
Post Conditions	None.

Alternate Flows	- Leaving empty text fields will give you Error 26. - Not using a gmail.com email address will give you Error 10. - Using an invalid email will give you Error 11. - Passwords shorter than 6 characters long will give you Error 20. - Password matching the email address will give you Error 21. - Password not containing one number (0-9) will give you Error 22. - Password not containing one lowercase letter (a-z) will give you Error 23. - Password not containing one uppercase letter (A-Z) will give you Error 24. - Signing in with an incorrect password will give you Error 87. - Signing in with an incorrect password will give you Error 89. - Any other upload problems will give you Error 65. - Any other problem you experience will give you Error 1 or Error 2.
Notes	None.

4.6.8 View activities

Use Case Name	**View activities 8.0**
Description	View account activities
Actors	Users and administrators.
Pre-Conditions	Enabled account.
Basic Flow	1. Sign into the application. 2. Click on the “Account” menu. 3. Click on the “View activities” button.
Post Conditions	None.
Alternate Flows	- An expired session will give you Error 99. - Any other problem you experience will give you Error 1 or Error 2.
Notes	None.

4.6.9 Sign out

Use Case Name	**Sign out 9.0**
Description	Sign out of the application.
Actors	Administrators.
Pre-Conditions	Already signed in.
Basic Flow	1. Click on the “Sign out” menu.
Post Conditions	None.
Alternate Flows	- Any other problem you experience will give you Error 1 or Error 2.
Notes	None.

4.6.10 Enable account

Use Case Name	**Enable account 9.0**
Description	Enable an account.
Actors	Administrators.
Pre-Conditions	Enabled account.
Basic Flow	1. Sign into the application. 2. Click on the “Account” menu. 3. Click on the “View accounts” button. 4. Click on the “Select an account” drop down menu and select an account.

	5. Check on the “Disable account” checkbox. 6. Click on the “Update” button.
Post Conditions	None.
Alternate Flows	- An expired session will give you Error 99. - Any other problem you experience will give you Error 1 or Error 2.
Notes	None.

4.6.11 Disable account

Use Case Name	**Disable account 10.0**
Description	Disable an account.
Actors	Administrators.
Pre-Conditions	Enabled account.
Basic Flow	1. Sign into the application. 2. Click on the “Account” menu. 3. Click on the “View accounts” button. 4. Click on the “Select an account” drop down menu and select an account. 5. Un-check on the “Disable account” checkbox. 6. Click on the “Update” button.
Post Conditions	None.
Alternate Flows	- An expired session will give you Error 99. - Any other problem you experience will give you Error 1 or Error 2.
Notes	None.

4.6.12 Enable pet

Use Case Name	**Enable pet 12.0**
Description	Enable a pet.
Actors	Administrators.
Pre-Conditions	Enabled account.
Basic Flow	1. Sign into the application. 2. Click on the “Account” menu. 3. Click on the “View accounts” button. 4. Click on the “Select an account” drop down menu and select an account. 5. Click on the “Select a pet” drop down menu and select a pet. 6. Check on the “Disable pet” checkbox. 7. Click on the “Update” button.
Post Conditions	None.
Alternate Flows	- An expired session will give you Error 99. - Any other problem you experience will give you Error 1, Error 2 or Error 3.
Notes	None.

4.6.13 Disable pet

Use Case Name	**Disable pet 13.0**
Description	Disable a pet.
Actors	Administrators.
Pre-Conditions	Enabled account.
Basic Flow	1. Sign into the application.

	2. Click on the “Account” menu. 3. Click on the “View accounts” button. 4. Click on the “Select an account” drop down menu and select an account. 5. Click on the “Select a pet” drop down menu and select a pet. 6. Un-check on the “Disable pet” checkbox. 7. Click on the “Update” button.
Post Conditions	None.
Alternate Flows	- An expired session will give you Error 99. - Any other problem you experience will give you Error 1, Error 2 or Error 3.
Notes	None.

4.6.14 Update pet’s name

Use Case Name	**Update pet’s name 14.0**
Description	Update a pet’s name.
Actors	Administrators.
Pre-Conditions	Enabled account.
Basic Flow	1. Sign into the application. 2. Click on the “Account” menu. 3. Click on the “View accounts” button. 4. Click on the “Select an account” drop down menu and select an account. 5. Click on the “Select a pet” drop down menu and select a pet. 6. Edit the Pet’s name in the “Pet’s name” field. 7. Click on the “Update” button.
Post Conditions	None.
Alternate Flows	- An expired session will give you Error 99. - Any other problem you experience will give you Error 1, Error 2 or Error 3.
Notes	None.

4.6.15 Update pet’s breed

Use Case Name	**Update Pet’s breed 15.0**
Description	Update a pet’s breed.
Actors	Administrators.
Pre-Conditions	Enabled account.
Basic Flow	1. Sign into the application. 2. Click on the “Account” menu. 3. Click on the “View accounts” button. 4. Click on the “Select an account” drop down menu and select an account. 5. Click on the “Select a pet” drop down menu and select a pet. 6. Click on the “Pet’s breed” drop down and select a breed. 7. Click on the “Update” button.
Post Conditions	None.
Alternate Flows	- An expired session will give you Error 99. - Any other problem you experience will give you Error 1, Error 2 or Error 3.
Notes	None.

4.6.16 Update pet’s gender

Use Case Name	Update Pet’s gender 16.0
Description	Update a pet’s gender.
Actors	Administrators.
Pre-Conditions	Enabled account.
Basic Flow	1. Sign into the application. 2. Click on the “Account” menu. 3. Click on the “View accounts” button. 4. Click on the “Select an account” drop down menu and select an account. 5. Click on the “Select a pet” drop down menu and select a pet. 6. Click on the “Pet’s gender” drop down menu and select a gender. 7. Click on the “Update” button.
Post Conditions	None.
Alternate Flows	- An expired session will give you Error 99. - Any other problem you experience will give you Error 1, Error 2 or Error 3.
Notes	None.

4.6.17 Add breed

Use Case Name	Add breed 17.0
Description	Add a breed.
Actors	Administrators.
Pre-Conditions	Enabled account.
Basic Flow	1. Sign into the application. 2. Click on the “Account” menu. 3. Click on the “View breeds” button. 4. Click on the “Select a breed” drop down menu and select a breed. 5. Edit the breed’s name in the “Breed’s name” field. 6. Click on the “Update” button.
Post Conditions	None.
Alternate Flows	- Adding a breed that already exists will give you Error 50. - An expired session will give you Error 99. - Any other problem you experience will give you Error 1, Error 2 or Error 3.
Notes	None.

4.6.18 Update breed

Use Case Name	Update breed 18.0
Description	Update the name of a breed.
Actors	Administrators.
Pre-Conditions	Enabled account.
Basic Flow	1. Sign into the application. 2. Click on the “Account” menu. 3. Click on the “View breeds” button. 4. Enter a breed name in the “Add a new breed” text field. 5. Click on the “Add” button.
Post Conditions	None.
Alternate Flows	- An expired session will give you Error 99. - Any other problem you experience will give you Error 1, Error 2 or Error 3.

Notes	None.

4.6.19 View errors

Use Case Name	View errors 19.0
Description	View website errors.
Actors	Administrators.
Pre-Conditions	Enabled account.
Basic Flow	1. Sign into the application. 2. Click on the “Account” menu. 3. Click on the “View errors” button.
Post Conditions	None.
Alternate Flows	- An expired session will give you Error 99. - Any other problem you experience will give you Error 1 or Error 2.
Notes	None.

4.6.20 Add administrator

Use Case Name	Add administrator 20.0
Description	Add administrators.
Actors	Server Administrators.
Pre-Conditions	Server Administrators.
Basic Flow	1. Go to the “..sa/admin.html” page. 2. Enter the user name in the “Email address” field. 3. Enter the password in the “Password” field. 4. Enter the email address in the “Email address” field. 5. Enter the password in the “Password” field.
Post Conditions	None.
Alternate Flows	None.
Notes	None.

4.6.21 Update pet policy

Use Case Name	Update pet policy 21.0
Description	Upload the pet policy to the home directory of the website. The file must be a PDF and must be named “petpolicy.pdf”.
Actors	Server Administrators.
Pre-Conditions	Server Administrators.
Basic Flow	None.
Post Conditions	None.
Alternate Flows	None.
Notes	None.

4.7 Browser Compatibility

Browser	Version
Chrome	
Firefox	
Internet Explorer	

Opera	
Safari	

4.8 Security

The only risk assessment if this application is to be stolen or disappear is as followed:

- Negligible
- Loss of personal information

4.8.1 Overview

A variety of Open Web Application Security Project's (OWASP) suggestions will be applied.

https://www.owasp.org/index.php/Session_Management_Cheat_Sheet

4.8.2 Information Gathering

Because I'm being hosting by a commercial company, it simple for someone to navigate the different services offered by the company and determine the sort of system I'm running on.

4.8.3 Business Logic, Authentication and Authorization

4.8.4 Session Management

4.8.5 Data Validation

4.8.6 Vulnerabilities

My system is vulnerable to Denial of Service.

4.8.8 Auditing

4.8.9 Risk Assessments

All these are in my book1 excel file. Will transfer it here later

Website is forced into a HTTPS/SSL connection therefore all sessions are protected

Every action taking on an account will be logged in the history tab of that particular account, as well as the pet.

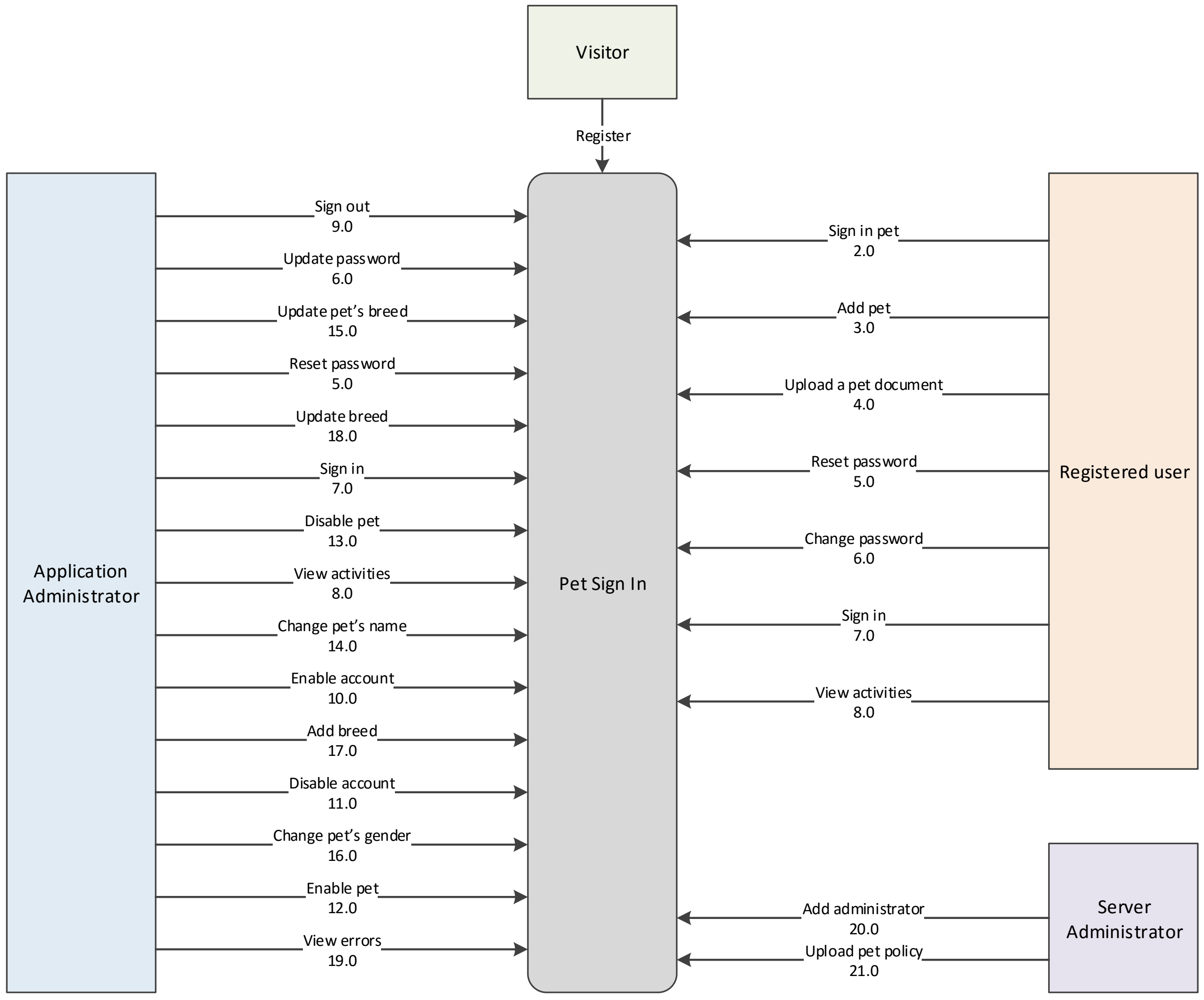
The dashboard will show a brief history of the user's account.

All actions can notify the account holder of a new log.

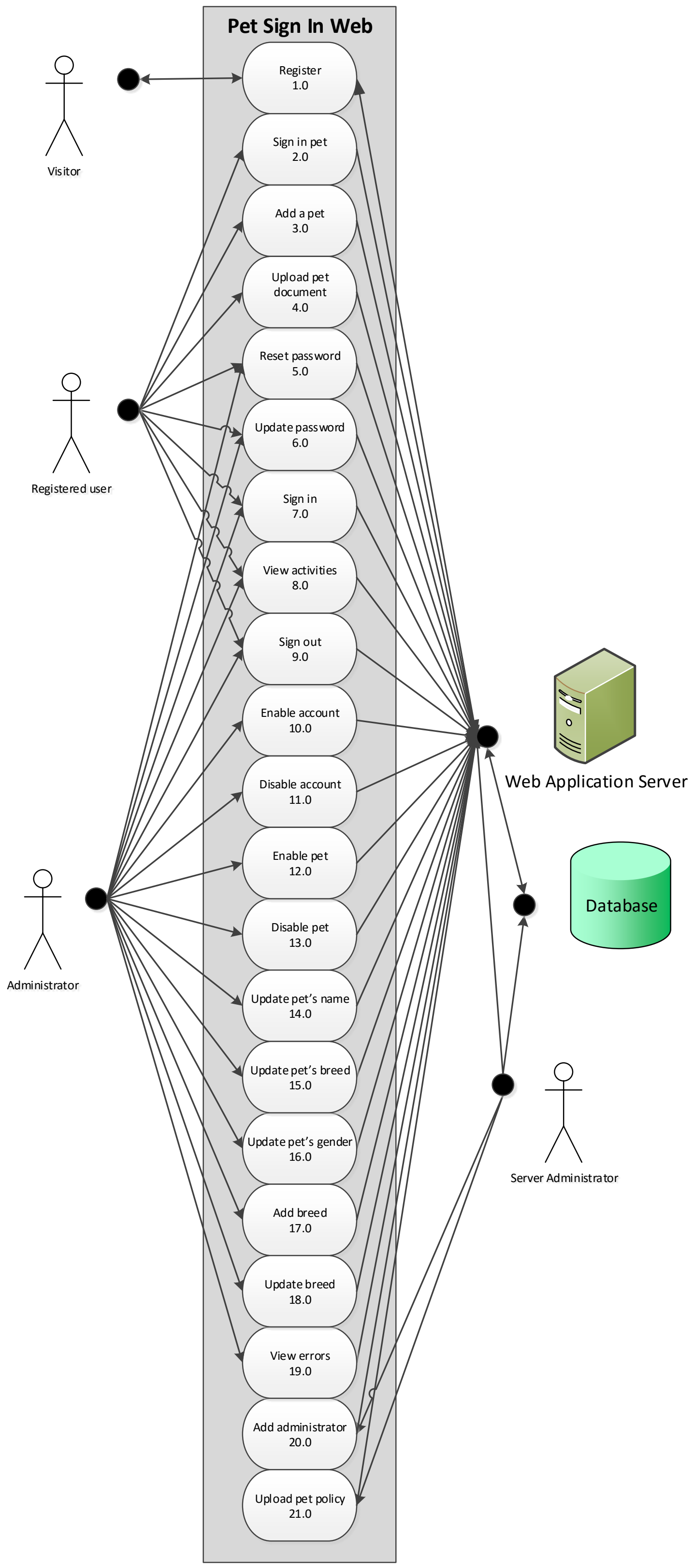
A two factor login will be optional (via email or possibly phone multimedia message).

5 Design

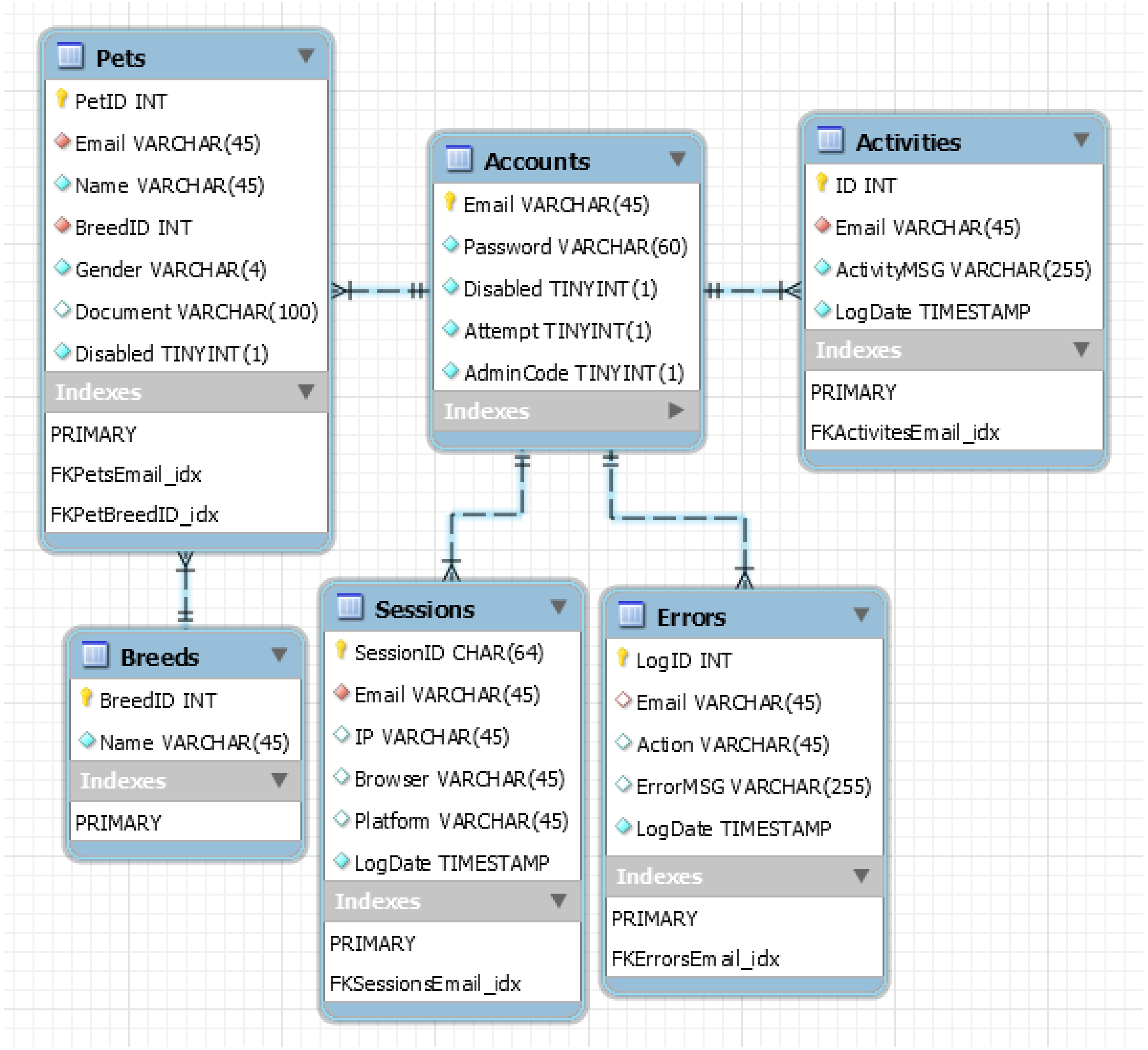
5.1 Context Dataflow Diagram



5.2 Use Case Diagram


Pet Sign In Web
Visitor
Registered user
Administrator
Server Administrator
Register 1.0
Sign in pet 2.0
Add a pet 3.0
Upload pet document 4.0
Reset password 5.0
Update password 6.0
Sign in 7.0
View activities 8.0
Sign out 9.0
Enable account 10.0
Disable account 11.0
Enable pet 12.0
Disable pet 13.0
Update pet's name 14.0
Update pet's breed 15.0
Update pet's gender 16.0
Add breed 17.0
Update breed 18.0
View errors 19.0
Add administrator 20.0
Upload pet policy 21.0
Web Application Server
Database

5.3 Entity Relationship Diagram



5.5 Dataflow Diagram

5.6 Activity Diagram

5.7 Conceptual Website Diagram

5.8 Database Script

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;

SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';


-- -----------------------------------------------------
-- Schema djkabau1_petsignin
-- -----------------------------------------------------
DROP SCHEMA IF EXISTS `djkabau1_petsignin` ;


-- -----------------------------------------------------
-- Schema djkabau1_petsignin
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `djkabau1_petsignin` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci ;
USE `djkabau1_petsignin` ;


-- -----------------------------------------------------
-- Table `djkabau1_petsignin`.`Accounts`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `djkabau1_petsignin`.`Accounts` ;


CREATE TABLE IF NOT EXISTS `djkabau1_petsignin`.`Accounts` (
  `Email` VARCHAR(45) NOT NULL COMMENT '',
  `Password` VARCHAR(60) NOT NULL COMMENT '',
  `Disabled` TINYINT(1) NOT NULL COMMENT '',
  `Attempt` TINYINT(1) NOT NULL COMMENT '',
  `AdminCode` TINYINT(1) NOT NULL COMMENT '',
  PRIMARY KEY (`Email`)  COMMENT '')
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `djkabau1_petsignin`.`Breeds`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `djkabau1_petsignin`.`Breeds` ;


CREATE TABLE IF NOT EXISTS `djkabau1_petsignin`.`Breeds` (
  `BreedID` INT NOT NULL AUTO_INCREMENT COMMENT '',
  `Name` VARCHAR(45) NOT NULL COMMENT '',
  PRIMARY KEY (`BreedID`)  COMMENT '')
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `djkabau1_petsignin`.`Pets`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `djkabau1_petsignin`.`Pets` ;

CREATE TABLE IF NOT EXISTS `djkabau1_petsignin`.`Pets` (
  `PetID` INT NOT NULL AUTO_INCREMENT COMMENT '',
  `Email` VARCHAR(45) NOT NULL COMMENT '',
  `Name` VARCHAR(45) NOT NULL COMMENT '',
  `BreedID` INT NOT NULL COMMENT '',
  `Gender` VARCHAR(4) NOT NULL COMMENT '',
  `Document` VARCHAR(100) NULL COMMENT '',
  `Disabled` TINYINT(1) NOT NULL COMMENT '',
  PRIMARY KEY (`PetID`)  COMMENT '',
  INDEX `FKPetsEmail_idx` (`Email` ASC)  COMMENT '',
  INDEX `FKPetBreedID_idx` (`BreedID` ASC)  COMMENT '',
  CONSTRAINT `FKPetsEmail`
    FOREIGN KEY (`Email`)
    REFERENCES `djkabau1_petsignin`.`Accounts` (`Email`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `FKPetBreedID`
    FOREIGN KEY (`BreedID`)
    REFERENCES `djkabau1_petsignin`.`Breeds` (`BreedID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `djkabau1_petsignin`.`Activities`
-- -----------------------------------------------------
```

```
DROP TABLE IF EXISTS `djkabau1_petsignin`.`Activities` ;

CREATE TABLE IF NOT EXISTS `djkabau1_petsignin`.`Activities` (
  `ID` INT NOT NULL AUTO_INCREMENT COMMENT '',
  `Email` VARCHAR(45) NOT NULL COMMENT '',
  `ActivityMSG` VARCHAR(255) NOT NULL COMMENT '',
  `LogDate` TIMESTAMP NOT NULL DEFAULT NOW() COMMENT '',
  PRIMARY KEY (`ID`)  COMMENT '',
  INDEX `FKActivitesEmail_idx` (`Email` ASC)  COMMENT '',
  CONSTRAINT `FKActivitesEmail`
    FOREIGN KEY (`Email`)
    REFERENCES `djkabau1_petsignin`.`Accounts` (`Email`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `djkabau1_petsignin`.`Errors`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `djkabau1_petsignin`.`Errors` ;

CREATE TABLE IF NOT EXISTS `djkabau1_petsignin`.`Errors` (
  `LogID` INT NOT NULL AUTO_INCREMENT COMMENT '',
  `Email` VARCHAR(45) NULL COMMENT '',
  `Action` VARCHAR(45) NULL COMMENT '',
  `ErrorMSG` VARCHAR(255) NULL COMMENT '',
  `LogDate` TIMESTAMP NOT NULL DEFAULT NOW() COMMENT '',
  PRIMARY KEY (`LogID`)  COMMENT '',
  INDEX `FKErrorsEmail_idx` (`Email` ASC)  COMMENT '',
  CONSTRAINT `FKErrorsEmail`
    FOREIGN KEY (`Email`)
    REFERENCES `djkabau1_petsignin`.`Accounts` (`Email`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `djkabau1_petsignin`.`Sessions`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `djkabau1_petsignin`.`Sessions` ;

CREATE TABLE IF NOT EXISTS `djkabau1_petsignin`.`Sessions` (
  `SessionID` CHAR(64) NOT NULL COMMENT '',
  `Email` VARCHAR(45) NOT NULL COMMENT '',
  `IP` VARCHAR(45) NULL COMMENT '',
  `Browser` VARCHAR(45) NULL COMMENT '',
  `Platform` VARCHAR(45) NULL COMMENT '',
  `LogDate` TIMESTAMP NOT NULL DEFAULT NOW() COMMENT '',
  PRIMARY KEY (`SessionID`)  COMMENT '',
  INDEX `FKSessionsEmail_idx` (`Email` ASC)  COMMENT '',
  CONSTRAINT `FKSessionsEmail`
    FOREIGN KEY (`Email`)
    REFERENCES `djkabau1_petsignin`.`Accounts` (`Email`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

USE `djkabau1_petsignin` ;

-- -----------------------------------------------------
-- procedure UTCreate
-- -----------------------------------------------------

USE `djkabau1_petsignin`;
DROP procedure IF EXISTS `djkabau1_petsignin`.`UTCreate`;

DELIMITER $$
USE `djkabau1_petsignin`$$
CREATE PROCEDURE `UTCreate` ()
BEGIN
```

```
DROP TABLE IF EXISTS djkabau1_petsignin.UnitTest ;
        CREATE TABLE IF NOT EXISTS djkabau1_petsignin.UnitTest (
        TestColumn INT NOT NULL,
        PRIMARY KEY (TestColumn))
        ENGINE = InnoDB;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure UTInsert
-- -----------------------------------------------------

USE `djkabau1_petsignin`;
DROP procedure IF EXISTS `djkabau1_petsignin`.`UTInsert`;

DELIMITER $$
USE `djkabau1_petsignin`$$
CREATE PROCEDURE `UTInsert` (IN UTValue INT)
BEGIN
INSERT INTO djkabau1_petsignin.UnitTest (TestColumn) VALUES (UTValue);
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure UTUpdate
-- -----------------------------------------------------

USE `djkabau1_petsignin`;
DROP procedure IF EXISTS `djkabau1_petsignin`.`UTUpdate`;

DELIMITER $$
USE `djkabau1_petsignin`$$
CREATE PROCEDURE `UTUpdate` (IN UpdatedUTValue INT, IN UTValue INT)
BEGIN
```

```
UPDATE djkabau1_petsignin.UnitTest SET TestColumn = (UpdatedUTValue) WHERE TestColumn = (UTValue);
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure UTDelete
-- -----------------------------------------------------

USE `djkabau1_petsignin`;
DROP procedure IF EXISTS `djkabau1_petsignin`.`UTDelete`;

DELIMITER $$
USE `djkabau1_petsignin`$$
CREATE PROCEDURE `UTDelete` (IN UpdatedUTValue INT)
BEGIN
DELETE FROM djkabau1_petsignin.UnitTest WHERE TestColumn = (UpdatedUTValue);
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure UTDrop
-- -----------------------------------------------------

USE `djkabau1_petsignin`;
DROP procedure IF EXISTS `djkabau1_petsignin`.`UTDrop`;

DELIMITER $$
USE `djkabau1_petsignin`$$
CREATE PROCEDURE `UTDrop` ()
BEGIN
DROP TABLE IF EXISTS djkabau1_petsignin.UnitTest;
END$$

DELIMITER ;
```

```
-- -----------------------------------------------------
-- procedure AddAttempt
-- -----------------------------------------------------

USE `djkabau1_petsignin`;
DROP procedure IF EXISTS `djkabau1_petsignin`.`AddAttempt`;

DELIMITER $$
USE `djkabau1_petsignin`$$
CREATE PROCEDURE `AddAttempt` (IN NewAttempt INT, IN Email VARCHAR(45))
BEGIN
UPDATE djkabau1_petsignin.Accounts SET Attempt = (NewAttempt) WHERE Email = (Email);
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure ResetAttempt
-- -----------------------------------------------------

USE `djkabau1_petsignin`;
DROP procedure IF EXISTS `djkabau1_petsignin`.`ResetAttempt`;

DELIMITER $$
USE `djkabau1_petsignin`$$
CREATE PROCEDURE `ResetAttempt` (IN Email VARCHAR(45))
BEGIN
UPDATE djkabau1_petsignin.Accounts SET Attempt = ("0") WHERE Email = (Email);
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure AddSession
-- -----------------------------------------------------
```

```
USE `djkabau1_petsignin`;

DROP procedure IF EXISTS `djkabau1_petsignin`.`AddSession`;


DELIMITER $$

USE `djkabau1_petsignin`$$

CREATE PROCEDURE `AddSession` (IN xSessionID VARCHAR(64), IN xEmail VARCHAR(45), IN xSessionIP VARCHAR(45), IN
xSessionBrowser VARCHAR(45), IN xSessionPlatform VARCHAR(45))

BEGIN

INSERT INTO djkabau1_petsignin.Sessions (SessionID, Email, IP, Browser, Platform) VALUES (xSessionID, xEmail, xSessionIP,
xSessionBrowser, xSessionPlatform);

END$$


DELIMITER ;



-- -----------------------------------------------------

-- procedure FetchSession

-- -----------------------------------------------------


USE `djkabau1_petsignin`;

DROP procedure IF EXISTS `djkabau1_petsignin`.`FetchSession`;


DELIMITER $$

USE `djkabau1_petsignin`$$

CREATE PROCEDURE `FetchSession` (IN xSessionID VARCHAR(64))

BEGIN

SELECT * FROM djkabau1_petsignin.Sessions WHERE SessionID = (xSessionID);

END$$


DELIMITER ;



-- -----------------------------------------------------

-- procedure FetchAccountRole

-- -----------------------------------------------------


USE `djkabau1_petsignin`;

DROP procedure IF EXISTS `djkabau1_petsignin`.`FetchAccountRole`;
```

```
DELIMITER $$

USE `djkabau1_petsignin`$$

CREATE PROCEDURE `FetchAccountRole` (IN xEmail VARCHAR(45))

BEGIN

SELECT Disabled, Attempt, AdminCode FROM djkabau1_petsignin.Accounts WHERE Email = (xEmail);

END$$


DELIMITER ;


-- -----------------------------------------------------

-- procedure FetchSession

-- -----------------------------------------------------


USE `djkabau1_petsignin`;

DROP procedure IF EXISTS `djkabau1_petsignin`.`FetchSession`;


DELIMITER $$

USE `djkabau1_petsignin`$$

CREATE PROCEDURE `FetchSession` (IN xSessionID VARCHAR(64))

BEGIN

SELECT * FROM djkabau1_petsignin.Sessions WHERE SessionID = (xSessionID);

END$$


DELIMITER ;


-- -----------------------------------------------------

-- procedure DeleteSession

-- -----------------------------------------------------


USE `djkabau1_petsignin`;

DROP procedure IF EXISTS `djkabau1_petsignin`.`DeleteSession`;


DELIMITER $$

USE `djkabau1_petsignin`$$

CREATE PROCEDURE `DeleteSession` (IN xEmail VARCHAR(45))
```

```
BEGIN

DELETE FROM djkabau1_petsignin.Sessions WHERE Email = (xEmail);

END$$


DELIMITER ;


-- -----------------------------------------------------

-- procedure AddAccount

-- -----------------------------------------------------


USE `djkabau1_petsignin`;

DROP procedure IF EXISTS `djkabau1_petsignin`.`AddAccount`;


DELIMITER $$

USE `djkabau1_petsignin`$$

CREATE PROCEDURE `AddAccount` (IN xEmail VARCHAR(45), IN xPassword VARCHAR(60), IN xDisabled INT, IN xAttempt INT, IN
xAdminCode INT)

BEGIN

INSERT INTO djkabau1_petsignin.Accounts (Email, Password, Disabled, Attempt, AdminCode) VALUES (xEmail, xPassword, xDisabled,
xAttempt, xAdminCode);

END$$


DELIMITER ;


-- -----------------------------------------------------

-- procedure FetchUser

-- -----------------------------------------------------


USE `djkabau1_petsignin`;

DROP procedure IF EXISTS `djkabau1_petsignin`.`FetchUser`;


DELIMITER $$

USE `djkabau1_petsignin`$$

CREATE PROCEDURE `FetchUser` (IN xEmail VARCHAR(45))

BEGIN

SELECT * FROM djkabau1_petsignin.Accounts WHERE Email = (xEmail);

END$$
```

```
DELIMITER ;

-- -----------------------------------------------------
-- procedure FetchBreeds
-- -----------------------------------------------------

USE `djkabau1_petsignin`;
DROP procedure IF EXISTS `djkabau1_petsignin`.`FetchBreeds`;

DELIMITER $$
USE `djkabau1_petsignin`$$
CREATE PROCEDURE `FetchBreeds` ()
BEGIN
SELECT * FROM djkabau1_petsignin.Breeds ORDER BY Name ASC;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure FetchActivities
-- -----------------------------------------------------

USE `djkabau1_petsignin`;
DROP procedure IF EXISTS `djkabau1_petsignin`.`FetchActivities`;

DELIMITER $$
USE `djkabau1_petsignin`$$
CREATE PROCEDURE `FetchActivities` (IN xEmail VARCHAR(45))
BEGIN
SELECT ActivityMSG, LogDate FROM djkabau1_petsignin.Activities WHERE Email = (xEmail) ORDER BY LogDate DESC;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure AddActivity
-- -----------------------------------------------------

USE `djkabau1_petsignin`;
DROP procedure IF EXISTS `djkabau1_petsignin`.`AddActivity`;

DELIMITER $$
USE `djkabau1_petsignin`$$
CREATE PROCEDURE `AddActivity` (IN xEmail VARCHAR(45), IN xActivityMSG VARCHAR(255))
BEGIN
INSERT INTO djkabau1_petsignin.Activities (Email, ActivityMSG) VALUES (xEmail, xActivityMSG);
END
$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure AddError
-- -----------------------------------------------------

USE `djkabau1_petsignin`;
DROP procedure IF EXISTS `djkabau1_petsignin`.`AddError`;

DELIMITER $$
USE `djkabau1_petsignin`$$
CREATE PROCEDURE `AddError` (IN xEmail VARCHAR(45), IN xAction VARCHAR(45), IN xErrorMSG VARCHAR(255))
BEGIN
INSERT INTO djkabau1_petsignin.Errors (Email, Action, ErrorMSG) VALUES (xEmail, xAction, xErrorMSG);
END
$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure AddPet
-- -----------------------------------------------------
```

```
USE `djkabau1_petsignin`;

DROP procedure IF EXISTS `djkabau1_petsignin`.`AddPet`;


DELIMITER $$

USE `djkabau1_petsignin`$$

CREATE PROCEDURE `AddPet` (IN xEmail VARCHAR(45), IN xName VARCHAR(45), IN xBreedID INT, IN xGender VARCHAR(4), IN
xDisabled INT)

BEGIN

INSERT INTO djkabau1_petsignin.Pets (Email, Name, BreedID, Gender, Disabled) VALUES (xEmail, xName, xBreedID, xGender, xDisabled);

END$$


DELIMITER ;


-- -----------------------------------------------------

-- procedure AddBreed

-- -----------------------------------------------------


USE `djkabau1_petsignin`;

DROP procedure IF EXISTS `djkabau1_petsignin`.`AddBreed`;


DELIMITER $$

USE `djkabau1_petsignin`$$

CREATE PROCEDURE `AddBreed` (IN xName VARCHAR(45))

BEGIN

INSERT INTO djkabau1_petsignin.Breeds (Name) VALUES (xName);

END$$


DELIMITER ;


-- -----------------------------------------------------

-- procedure FetchBreedNameCount

-- -----------------------------------------------------


USE `djkabau1_petsignin`;

DROP procedure IF EXISTS `djkabau1_petsignin`.`FetchBreedNameCount`;
```

```
DELIMITER $$

USE `djkabau1_petsignin`$$

CREATE PROCEDURE `FetchBreedNameCount` (IN xName VARCHAR(45))

BEGIN

SELECT count(*) as Count FROM djkabau1_petsignin.Breeds WHERE Name = (xName);

END$$


DELIMITER ;



-- -----------------------------------------------------

-- procedure FetchSignInPet

-- -----------------------------------------------------



USE `djkabau1_petsignin`;

DROP procedure IF EXISTS `djkabau1_petsignin`.`FetchSignInPet`;


DELIMITER $$

USE `djkabau1_petsignin`$$

CREATE PROCEDURE `FetchSignInPet` (IN xEmail VARCHAR(45))

BEGIN

SELECT PetID, Name, Disabled, (SELECT DATEDIFF(now(), (SELECT LogDate FROM djkabau1_petsignin.Activities WHERE Email =
(xEmail) AND ActivityMSG = CONCAT("Your pet ", Pets.Name, " has been signed in.") ORDER BY LogDate DESC LIMIT 1))) AS DiffDate
FROM djkabau1_petsignin.Pets WHERE Email = (xEmail) ORDER BY Name ASC;

END$$


DELIMITER ;



-- -----------------------------------------------------

-- procedure AddAdminAccount

-- -----------------------------------------------------



USE `djkabau1_petsignin`;

DROP procedure IF EXISTS `djkabau1_petsignin`.`AddAdminAccount`;


DELIMITER $$

USE `djkabau1_petsignin`$$
```

```sql
CREATE PROCEDURE `AddAdminAccount` (IN xEmail VARCHAR(45), IN xPassword VARCHAR(60), IN xDisabled INT, IN xAttempt
INT, IN xAdminCode INT)

BEGIN

INSERT INTO djkabau1_petsignin.Accounts (Email, Password, Disabled, Attempt, AdminCode) VALUES (xEmail, xPassword, xDisabled,
xAttempt, xAdminCode);

END$$


DELIMITER ;


-- -----------------------------------------------------

-- procedure UpdateAccountStatus

-- -----------------------------------------------------


USE `djkabau1_petsignin`;

DROP procedure IF EXISTS `djkabau1_petsignin`.`UpdateAccountStatus`;


DELIMITER $$

USE `djkabau1_petsignin`$$

CREATE PROCEDURE `UpdateAccountStatus` (IN xDisabled INT, IN xEmail VARCHAR(45))

BEGIN

UPDATE djkabau1_petsignin.Accounts SET Disabled = (xDisabled) WHERE Email = (xEmail);

END$$


DELIMITER ;


-- -----------------------------------------------------

-- procedure FetchErrors

-- -----------------------------------------------------


USE `djkabau1_petsignin`;

DROP procedure IF EXISTS `djkabau1_petsignin`.`FetchErrors`;


DELIMITER $$

USE `djkabau1_petsignin`$$

CREATE PROCEDURE `FetchErrors` ()

BEGIN

SELECT Email, Action, ErrorMSG, LogDate FROM djkabau1_petsignin.Errors;
```

```sql
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure FetchAdmins
-- -----------------------------------------------------

USE `djkabau1_petsignin`;
DROP procedure IF EXISTS `djkabau1_petsignin`.`FetchAdmins`;

DELIMITER $$
USE `djkabau1_petsignin`$$
CREATE PROCEDURE `FetchAdmins` ()
BEGIN
SELECT * FROM djkabau1_petsignin.Accounts WHERE AdminCode = 2;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure FetchUsers
-- -----------------------------------------------------

USE `djkabau1_petsignin`;
DROP procedure IF EXISTS `djkabau1_petsignin`.`FetchUsers`;

DELIMITER $$
USE `djkabau1_petsignin`$$
CREATE PROCEDURE `FetchUsers` ()
BEGIN
SELECT Email FROM djkabau1_petsignin.Accounts WHERE AdminCode = 1;
END$$

DELIMITER ;
```

```
-- -----------------------------------------------------
-- procedure FetchPet
-- -----------------------------------------------------

USE `djkabau1_petsignin`;
DROP procedure IF EXISTS `djkabau1_petsignin`.`FetchPet`;

DELIMITER $$
USE `djkabau1_petsignin`$$
CREATE PROCEDURE `FetchPet` (IN xPetID INT)
BEGIN
SELECT Name, BreedID, Gender, Document FROM djkabau1_petsignin.Pets WHERE PetID = (xPetID);
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure FetchUserPets
-- -----------------------------------------------------

USE `djkabau1_petsignin`;
DROP procedure IF EXISTS `djkabau1_petsignin`.`FetchUserPets`;

DELIMITER $$
USE `djkabau1_petsignin`$$
CREATE PROCEDURE `FetchUserPets` (IN xEmail VARCHAR(45))
BEGIN
SELECT PetID, Name FROM djkabau1_petsignin.Pets WHERE Email = (xEmail) ORDER BY Name;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure FetchUserStatus
-- -----------------------------------------------------
```

```
USE `djkabau1_petsignin`;

DROP procedure IF EXISTS `djkabau1_petsignin`.`FetchUserStatus`;


DELIMITER $$

USE `djkabau1_petsignin`$$

CREATE PROCEDURE `FetchUserStatus` (IN xEmail VARCHAR(45))

BEGIN

SELECT Disabled FROM djkabau1_petsignin.Accounts WHERE Email = (xEmail);

END$$


DELIMITER ;


-- -----------------------------------------------------

-- procedure FetchPetStatus

-- -----------------------------------------------------


USE `djkabau1_petsignin`;

DROP procedure IF EXISTS `djkabau1_petsignin`.`FetchPetStatus`;


DELIMITER $$

USE `djkabau1_petsignin`$$

CREATE PROCEDURE `FetchPetStatus` (IN xPetID INT)

BEGIN

SELECT Disabled FROM djkabau1_petsignin.Pets WHERE PetID = (xPetID);

END$$


DELIMITER ;


-- -----------------------------------------------------

-- procedure UpdatePetStatus

-- -----------------------------------------------------


USE `djkabau1_petsignin`;

DROP procedure IF EXISTS `djkabau1_petsignin`.`UpdatePetStatus`;


DELIMITER $$
```

```
USE `djkabau1_petsignin`$$

CREATE PROCEDURE `UpdatePetStatus` (IN xDisabled INT, IN xPetID INT)

BEGIN

UPDATE djkabau1_petsignin.Pets SET Disabled = (xDisabled) WHERE PetID = (xPetID);

END$$


DELIMITER ;


-- -----------------------------------------------------

-- procedure UpdatePetName

-- -----------------------------------------------------


USE `djkabau1_petsignin`;

DROP procedure IF EXISTS `djkabau1_petsignin`.`UpdatePetName`;


DELIMITER $$

USE `djkabau1_petsignin`$$

CREATE PROCEDURE `UpdatePetName` (IN xName VARCHAR(45), IN xEmail VARCHAR(45))

BEGIN

UPDATE djkabau1_petsignin.Pets SET Name = (xName) WHERE Email = (xEmail);

END$$


DELIMITER ;


-- -----------------------------------------------------

-- procedure UpdatePetBreed

-- -----------------------------------------------------


USE `djkabau1_petsignin`;

DROP procedure IF EXISTS `djkabau1_petsignin`.`UpdatePetBreed`;


DELIMITER $$

USE `djkabau1_petsignin`$$

CREATE PROCEDURE `UpdatePetBreed` (IN xBreedID INT, IN xName VARCHAR(45), IN xEmail VARCHAR(45))

BEGIN

UPDATE djkabau1_petsignin.Pets SET BreedID = (xBreedID) WHERE Name = (xName) AND Email = (xEmail);
```

```
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure UpdatePetGender
-- -----------------------------------------------------

USE `djkabau1_petsignin`;
DROP procedure IF EXISTS `djkabau1_petsignin`.`UpdatePetGender`;

DELIMITER $$
USE `djkabau1_petsignin`$$
CREATE PROCEDURE `UpdatePetGender` (IN xGender VARCHAR(4), IN xEmail VARCHAR(45))
BEGIN
UPDATE djkabau1_petsignin.Pets SET Gender = (xGender) WHERE Email = (xEmail);
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure UpdateBreed
-- -----------------------------------------------------

USE `djkabau1_petsignin`;
DROP procedure IF EXISTS `djkabau1_petsignin`.`UpdateBreed`;

DELIMITER $$
USE `djkabau1_petsignin`$$
CREATE PROCEDURE `UpdateBreed` (IN xName VARCHAR(45), IN xBreedID INT)
BEGIN
UPDATE djkabau1_petsignin.Breeds SET Name = (xName) WHERE BreedID = (xBreedID);
END$$

DELIMITER ;
```

```
-- -----------------------------------------------------
-- procedure FetchPetNameCount
-- -----------------------------------------------------

USE `djkabau1_petsignin`;
DROP procedure IF EXISTS `djkabau1_petsignin`.`FetchPetNameCount`;

DELIMITER $$
USE `djkabau1_petsignin`$$
CREATE PROCEDURE `FetchPetNameCount` (IN xEmail VARCHAR(45), IN xName VARCHAR(45))
BEGIN
SELECT count(*) as Count FROM djkabau1_petsignin.Pets, djkabau1_petsignin.Accounts WHERE Accounts.Email = Pets.Email and
Pets.Email = (xEmail) AND Name = (xName);
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure UpdateDocument
-- -----------------------------------------------------

USE `djkabau1_petsignin`;
DROP procedure IF EXISTS `djkabau1_petsignin`.`UpdateDocument`;

DELIMITER $$
USE `djkabau1_petsignin`$$
CREATE PROCEDURE `UpdateDocument` (IN xDocument VARCHAR(255), IN xEmail VARCHAR(45), IN xName VARCHAR(45))
BEGIN
UPDATE djkabau1_petsignin.Pets SET Document = (xDocument) WHERE Email = (xEmail) AND Name = (xName);
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure UpdatePassword
-- -----------------------------------------------------
```

```
USE `djkabau1_petsignin`;

DROP procedure IF EXISTS `djkabau1_petsignin`.`UpdatePassword`;


DELIMITER $$

USE `djkabau1_petsignin`$$

CREATE PROCEDURE `UpdatePassword` (IN xPassword VARCHAR(64), IN xEmail VARCHAR(45))

BEGIN

UPDATE djkabau1_petsignin.Accounts set Password = (xPassword) where Email = (xEmail);

END$$


DELIMITER ;


SET SQL_MODE=@OLD_SQL_MODE;

SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;

SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

7 Screenshots