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)
    - Manage their pet(s)
        - Add, edit or disable
    - Manage their account
        - Change their password
        - Reset their password
        - Reset their account activation ode
        - View their account activity
- An employee administrator can:
    - Manage their account
    - Managed an employee with a registered account
        - Reset their password
        - Reset their account activation ode
        - View their account activity
        - Disable a registered user’s account or pet

    - Manage their pet(s)
        - Add, edit or disable their pet
    - Manage the company's pet policy
- A super administrator can:
    - View the website's code errors
    - Manage administrators:
        - Promote registered users to administrators
        - Demote administrators to registered users

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

- Free to use
- Mobile friendly website
- Fast registration process
- Easy login process to sign in pet
- Allows the company to have a audit trail of all pets
- Using latest security to protect data in transit and at rest
- Database is manually backed up on a weekly basis

2.3 Scope

This website is a pet sign in application installed in a pet friendly company's website. 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. More to be added later.

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 7 Professional
Version	6.1.7601 Service Pack 1 Build 7601
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

User:

- Email
  - Extension must match company email address
- Password
  - Password between 8-45 characters long
- Active – 1 use
  - Application functions are disabled if its 0
  - Email needs to be verified via ActivationCode method

- Attempts – 2 uses
    - Each failed login or registration to a registered account attempt adds an increments of 1
    - >4 locks the account
- AdminCode – 3 uses
    - Code 0 is a registered user
    - Code 1 is an administrator
    - Code 2 a super administrator
- ActivationCode – 1 use
    - A code sent to the email to verify the email address
- Disabled – 1 use
    - Can only be disabled by administrators

Pet:

- Name
- Breed
- DOB
- Sex
- Picture
- Disabled

4.2 Outputs – Information

Account Activity

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 contain at least six characters.
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 66	
Error 67	
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 Processes – Manual/Automatic

Manual:

- Administrator uploads company pet policy PDF
- User approves, updates, notifies and close accounts

Automatic:

- Account activity logging
- Error logging

4.4 Storage – Database

- All data will be stored in a MySQL Database.

4.5 Control – Interfaces

The website will consist of three roles:

- Registered user.
- Administrator.
- Super administrator.

Each will have distinct rights and ability throughout the application.

4.6 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

4.7 Training

A help how to will be available on the website. I will be using Joyride (see 2.6) as the tour.

4.9 Use Cases

4.9.1 Register

Use Case Name	Register 1.0
Description	Create an account.
Actors	Visitor.
Pre-Conditions	Valid Gmail email account.
Basic Flow	1. Enter email address 2. Enter password (Needs 1 lower case, upper case, number. 3. Click on "Terms and Conditions" and read it. 4. Click on "I agree" radio button. 5. 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.

4.9.2 Sign in pet

Use Case Name	
Description	Sign in pet.

Actors	Registered user.
Pre-Conditions	Already signed in and enabled pet that wasn't signed in on the same day.
Basic Flow	1. Click on the pet's name.
Post Conditions	None.
Alternate Flows	- Any other problem you experience will give you Error 1 or Error 2.
Notes	None.

4.9.3 Add pet

Use Case Name	**Add Pet 3.0**
Description	Add a pet.
Actors	Registered user.
Pre-Conditions	Able to successfully sign in.
Basic Flow	1. Enter pet's name. 2. Select pet's breed. 3. Select pet's gender. 4. Click on "I agree" radio button. 5. 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. - Any other problem you experience will give you Error 1 or Error 2.
Notes	None.

4.9.4 Upload pet document

Use Case Name	**Upload pet document 4.0**
Description	Upload pet vaccination and rabies documentations.
Actors	Registered user.
Pre-Conditions	Registered user email address, existing pet.
Basic Flow	1. Enter email address. 2. Enter pet's name. 3. Click on the "Choose File" button. 4. Select pet's PDF required document. 5. 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	

4.9.5

Use Case Name	Reset password
Description	Reset
Actors	Registered user or administrator.
Pre-Conditions	
Basic Flow	
Post Conditions	
Alternate Flows	
Notes	

4.9.6 Change password

Use Case Name	Change password
Description	
Actors	Registered user or administrator.
Pre-Conditions	Signed in.
Basic Flow	
Post Conditions	- - 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 problem you experience will give you Error 1 or Error 2.
Alternate Flows	
Notes	

4.9.7 Sign in

Use Case Name	Sign In
Description	Sign into the website.
Actors	User.
Pre-Conditions	Account is enable.
Basic Flow	1. Enter email address. 2. Enter password. 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 problem you experience will give you Error 1, Error 2 or Error 3.
Notes	None.

4.9.8

Use Case Name	
Description	
Actors	
Pre-Conditions	
Basic Flow	
Post Conditions	
Alternate Flows	
Notes	

4.9.9

Use Case Name	
Description	
Actors	
Pre-Conditions	
Basic Flow	
Post Conditions	
Alternate Flows	
Notes	

4.9.10

Use Case Name	
Description	
Actors	
Pre-Conditions	
Basic Flow	
Post Conditions	
Alternate Flows	
Notes	

4.9.11

Use Case Name	
Description	
Actors	
Pre-Conditions	
Basic Flow	
Post Conditions	
Alternate Flows	
Notes	

4.9.12

Use Case Name	
Description	
Actors	
Pre-Conditions	
Basic Flow	
Post Conditions	
Alternate Flows	
Notes	

4.9.13

Use Case Name	
Description	
Actors	
Pre-Conditions	
Basic Flow	
Post Conditions	
Alternate Flows	
Notes	

4.9.14

Use Case Name	
Description	
Actors	
Pre-Conditions	
Basic Flow	
Post Conditions	
Alternate Flows	
Notes	

4.9.15

Use Case Name	
Description	
Actors	
Pre-Conditions	
Basic Flow	
Post Conditions	
Alternate Flows	
Notes	

4.9.16

Use Case Name	
Description	

Actors	
Pre-Conditions	
Basic Flow	
Post Conditions	
Alternate Flows	
Notes	

4.9.17

Use Case Name	
Description	
Actors	
Pre-Conditions	
Basic Flow	
Post Conditions	
Alternate Flows	
Notes	

4.9.18 Change pet’s gender

Use Case Name	**Change pet’s gender 18.0**
Description	
Actors	
Pre-Conditions	
Basic Flow	
Post Conditions	
Alternate Flows	
Notes	

4.9.19 View errors

Use Case Name	**View errors 19.0**
Description	
Actors	
Pre-Conditions	
Basic Flow	
Post Conditions	
Alternate Flows	
Notes	

4.9.20 Add administrator

Use Case Name	**Add administrator 20.0**
Description	
Actors	
Pre-Conditions	
Basic Flow	
Post Conditions	
Alternate Flows	

Notes	

4.9.21 Upload pet policy

Use Case Name	**Upload pet policy 21.0**
Description	
Actors	
Pre-Conditions	
Basic Flow	
Post Conditions	
Alternate Flows	
Notes	

4.10 Assess Project Worth in terms of Cost vs. Value

4.10.1 Estimated Costs

Total Development Cost Estimate: $0.00

Operation Cost		
Name	Description	Cost
Computer Hardware		$0.00
Computer Software		$0.00
Hosting		$0.00
Books		$0.00
Others		$0.00
	Total	

Personal Cost		
Name	Description	Cost
Programmer		$0.00
Database Administrator		$0.00
System Analyst		$0.00
	Total	

4.10.2 Value

4.13 Preliminary Project Plan

4.13.2 Resource Assignment

Personnel:

- 1 Programmer/DBA/System Analyst

4.14 Browser Compatibility

Browser	**Version**
Chrome	
Firefox	
Internet Explorer	

Opera	
Safari	

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

- Negligible
- Loss of personal information

4.15.1 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

4.15.2 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.

4.15.3 Business Logic, Authentication and Authorization
4.15.4 Session Management
4.15.5 Data Validation
4.15.6 Vulnerabilities
My system is vulnerable to Denial of Service.

4.15.8 Auditing
4.15.9 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.

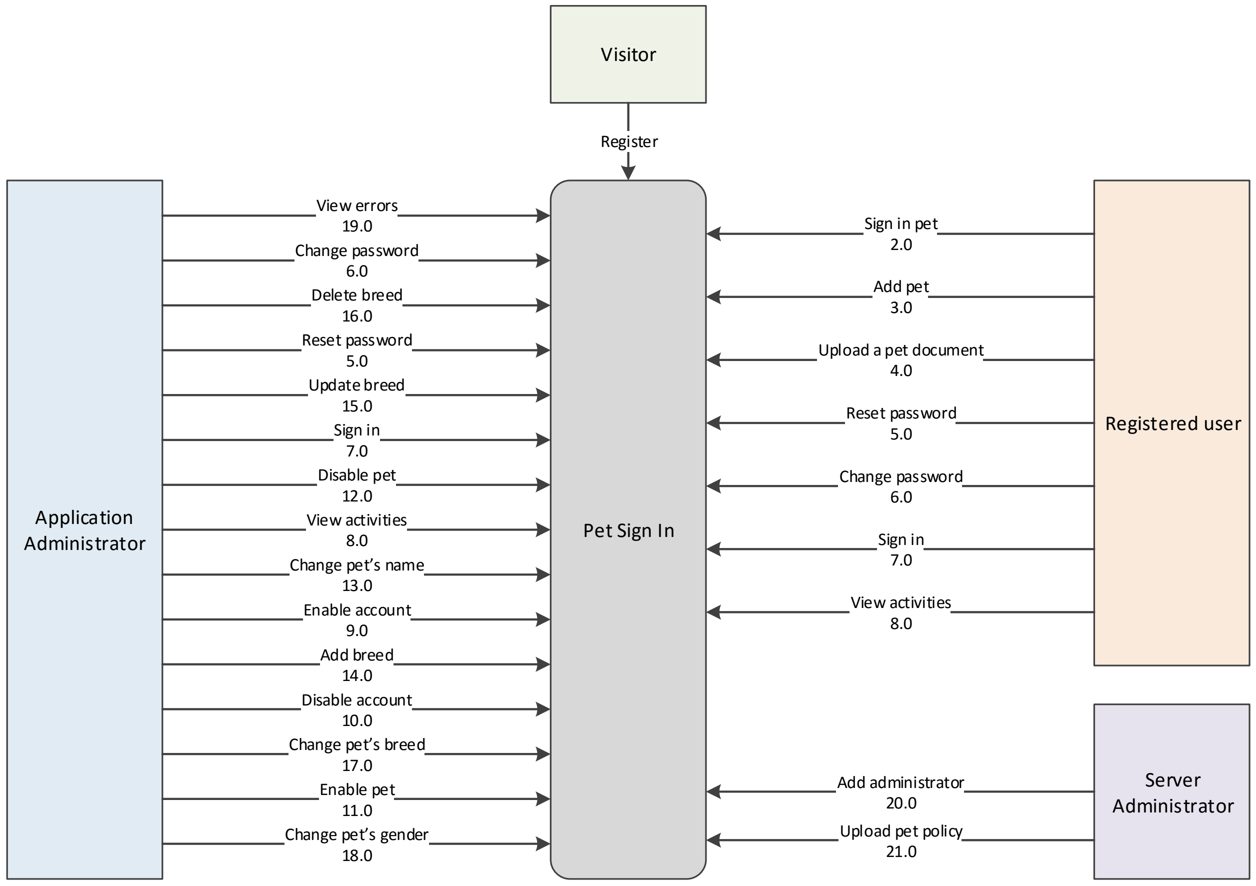
The dashboard will show a brief history of the user's account.

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

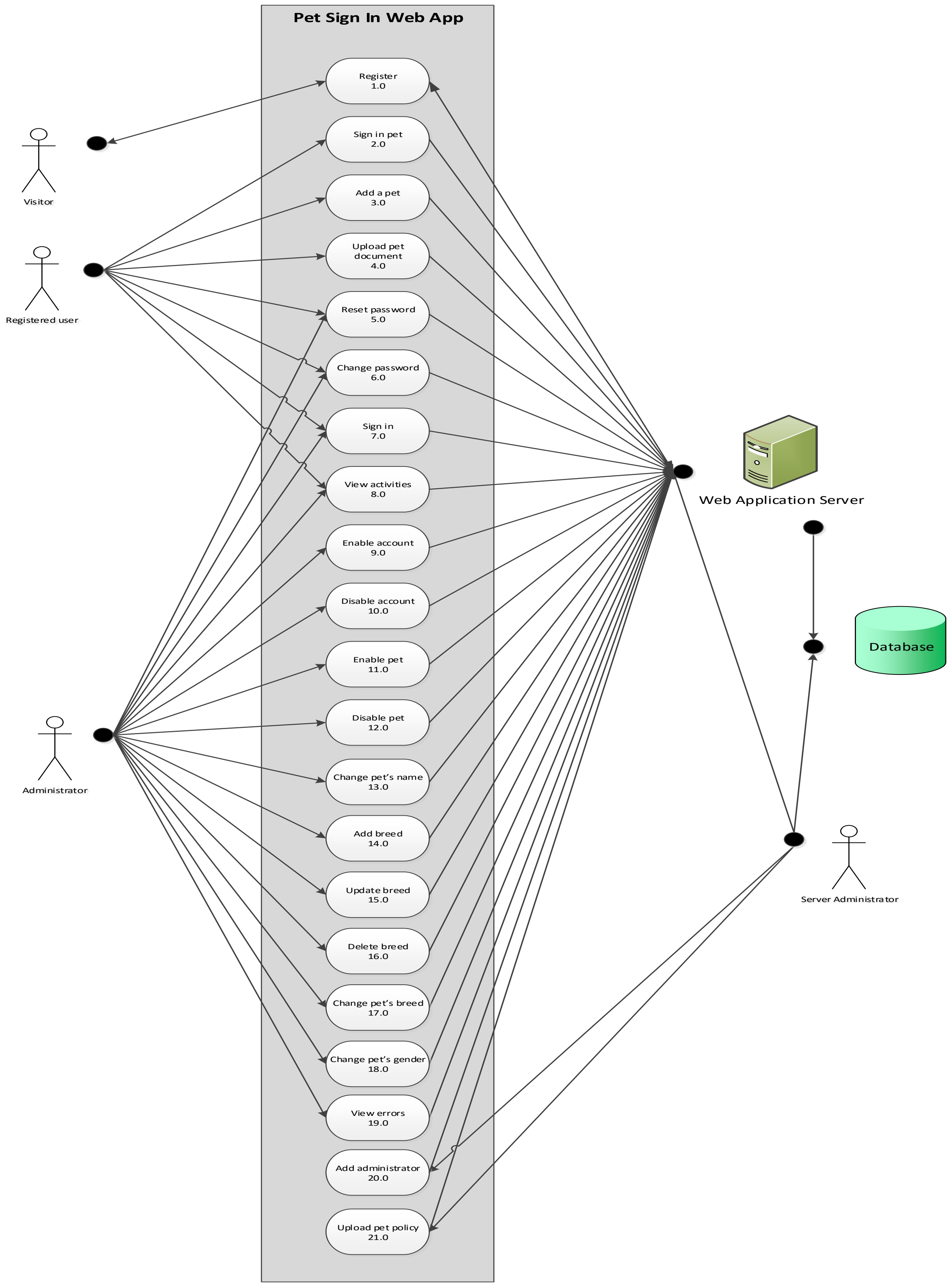
A two factor login will be optional (via email or possibly phone multimedia message).

5 Design

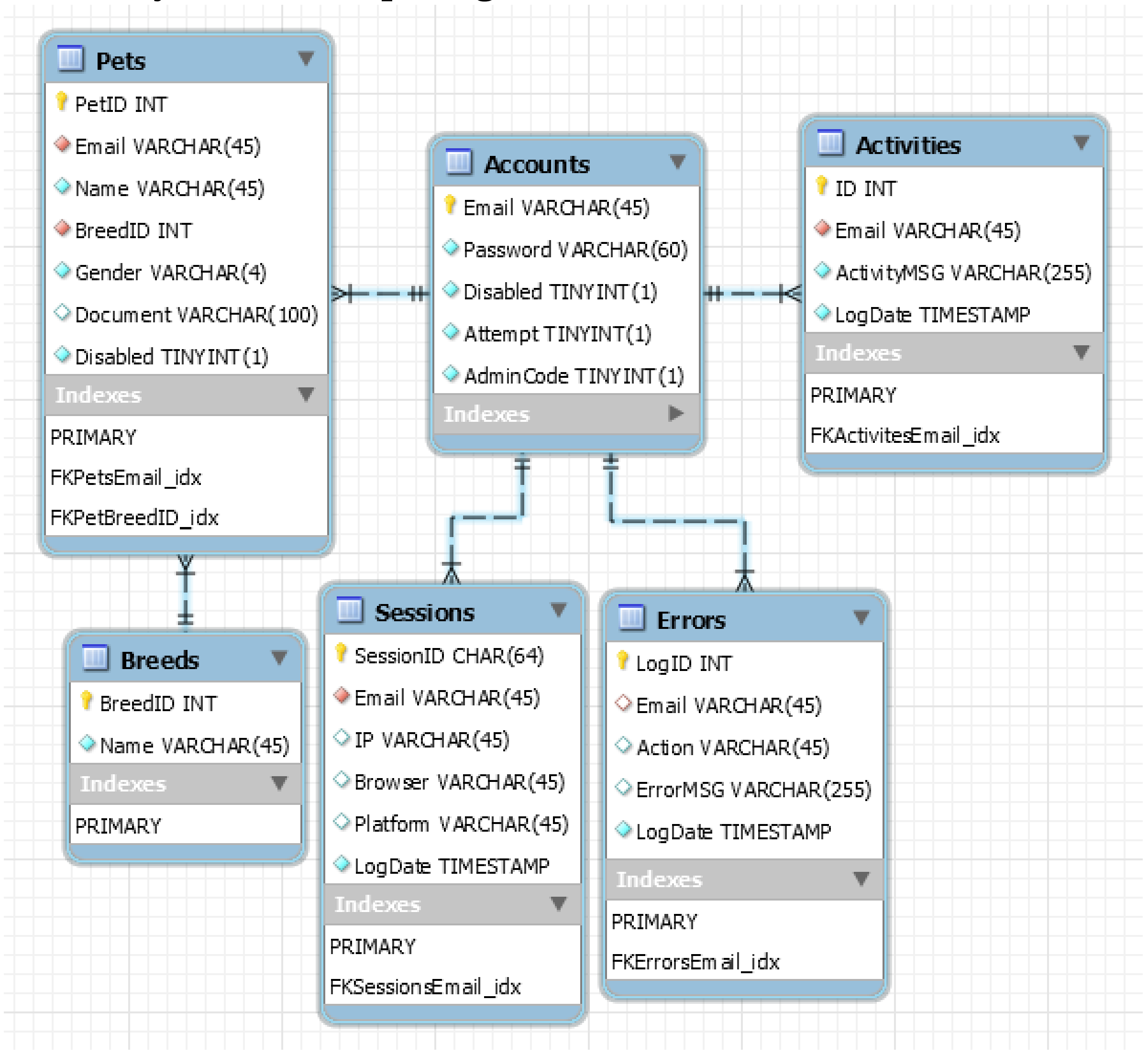
5.1 Context Dataflow Diagram



5.2 Use Case Diagram


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

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`$$
```

```
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;
```

```
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