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

THIS PAGE INTENTIONALLY LEFT BLANK

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 server 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 (https://petsignin.alibkaba.com/petsignin/) 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

Individuals that have an understanding of web applications and how to secure them.

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 (hosting website, www.JustHost.com)

- www.Dropbox.com
- www.GitHub.com
- JSFiddle
- www.Runnables.com

2.7 Languages

- HTML 5
- JavaScript
- PHP
- MySQL

2.8 Libraries

- Bootstrap
- jQuery
- Joyride

3 Analysis Overview

3.1 Assumptions, Dependencies and Constraints

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

The confidentiality of the website will depend on how well the intranet infrastructure of the company.

The integrity of the website will depend on how secure the database and the host sever is as well as the server and database administrators.

The below is the password for two directories.

URL	User name	password
https://petsignin.alibkaba.com/petsignin/uploads/	admin	2,c2sIoUI,=D
https://petsignin.alibkaba.com/petsignin/sa/admin.php	sadmin	qt4UqE5a+60q

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

- Negligible
- Loss of personal information

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	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)

4.2 Outputs – Information

4.2.1 Error Messages

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 – Roles

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
Phase	Start Date	End Data
	Iteration 1	
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
	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

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	User
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	Visitor, 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 10.0**
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 11.0**
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	49
Firefox	44
Internet Explorer/Microsoft Edge	11/25
Opera	35
Safari	9

4.8 Security

4.8.1 Overview

This section show the various security measures I took in making sure the web application is secured.

I followed a variety of Open Web Application Security Project’s (OWASP) suggestions.

Since there are various type of vulnerabilities for websites, I will list a few in 4.8.2. Section 4.8.3 will have a list of OWASP test I use against my website.

4.8.2 Attack Preventions

HTML injection is a type of injection issue that occurs when a user is able to control an input point and is able to inject arbitrary HTML code into a vulnerable web page. This vulnerability can have many consequences, like disclosure of a user's session cookies that could be used to impersonate the victim, or, more generally, it can allow the attacker to modify the page content seen by the victims. (Testing for HTML Injection (OTG-CLIENT-003), n.d.)

Measures taken to prevent HTML injection:

- User inputs do not directly affect HTML.
- Data retrieved from the Database affect the HTML

In PHP Code Injection testing, a tester submits input that is processed by the web server as dynamic code or as an included file. These tests can target various server-side scripting engines, e.g.., ASP or PHP. Proper input validation and secure coding practices need to be employed to protect against these attacks. (Testing for Code Injection (OTG-INPVAL-012), n.d.)

Measures taken to prevent PHP code injection:

- POST HTTP request method
    - Never cache
    - Doesn’t remain in the browser history or display data in the URL
    - Parameters are not saved in browser history or web server logs
    - Cannot be bookmarked

A SQL injection attack consists of insertion or "injection" of either a partial or complete SQL query via the data input or transmitted from the client (browser) to the web application A successful SQL injection attack can read sensitive data from the database, modify database data (insert/update/delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file existing on the DBMS file system or write files into the file system, and, in some cases, issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to affect the execution of predefined SQL commands. (Testing for SQL Injection (OTG-INPVAL-005), n.d.)

Measures taken to prevent SQL Injection:

- Prepared statements with parameterized queries
- Stored procedures
- Sanitized the user supplied input using PHP’s stripslashes to un-quote a quoted string

Asynchronous Javascript and XML (AJAX) is one of the latest techniques used by web application developers to provide a user experience similar to that of a traditional (i.e., "pre-web") application Since AJAX is still a new technology, there are many security issues that have not yet been fully researched. Some of the security issues in AJAX include: (Testing for AJAX Vulnerabilities (OWASP-AJ-001), n.d.)

- Increased attack surface with many more inputs to secure
- Exposed internal functions of the application
- Client access to third-party resources with no built-in security and encoding mechanisms
- Failure to protect authentication information and sessions
- Blurred line between client-side and server-side code, possibly resulting in security mistakes

Client side security preventions:

- The important business and security logic of the application is in the PHP code
- All-important data manipulation resides in the PHP side
- No encryption is done on client side

Server side security preventions:

- Non-exploitable JSON strings by having the outside primitive be an object

Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which he/she is currently authenticated. (Testing for CSRF (OTG-SESS-005), n.d.)

Measures taken to prevent CSRF:

- Only accept POST requests
- Disabling autocomplete
- Session expirations (Session Management Cheat Sheet, n.d.)

The Session Hijacking attack consists of the exploitation of the web session control mechanism, which is normally managed for a session token. (Session hijacking attack, n.d.)

Measures taken to prevent Session Hijacking attacks:

- Forcing all traffic through HTTPS/SSL (Transport Layer Protection Cheat Sheet, n.d.)
    - This also solves a few other vunerabilities
- Recording additional browser attribute with session (Session Management Cheat Sheet, n.d.)

4.8.2 Testing Checklist

I used the following OWASP's tests to assess my website. Many of them automatically pass due to other test.

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

Ref. No.	Category	Test Name
4.2		**Information Gathering**
4.2.1	OTG-INFO-001	Conduct Search Engine Discovery and Reconnaissance for Information Leakage
4.2.6	OTG-INFO-006	Identify application entry points
4.4		**Identity Management Testing**
4.4.1	OTG-IDENT-001	Test Role Definitions
4.4.2	OTG-IDENT-002	Test User Registration Process
4.4.3	OTG-IDENT-003	Test Account Provisioning Process
4.4.4	OTG-IDENT-004	Testing for Account Enumeration and Guessable User Account

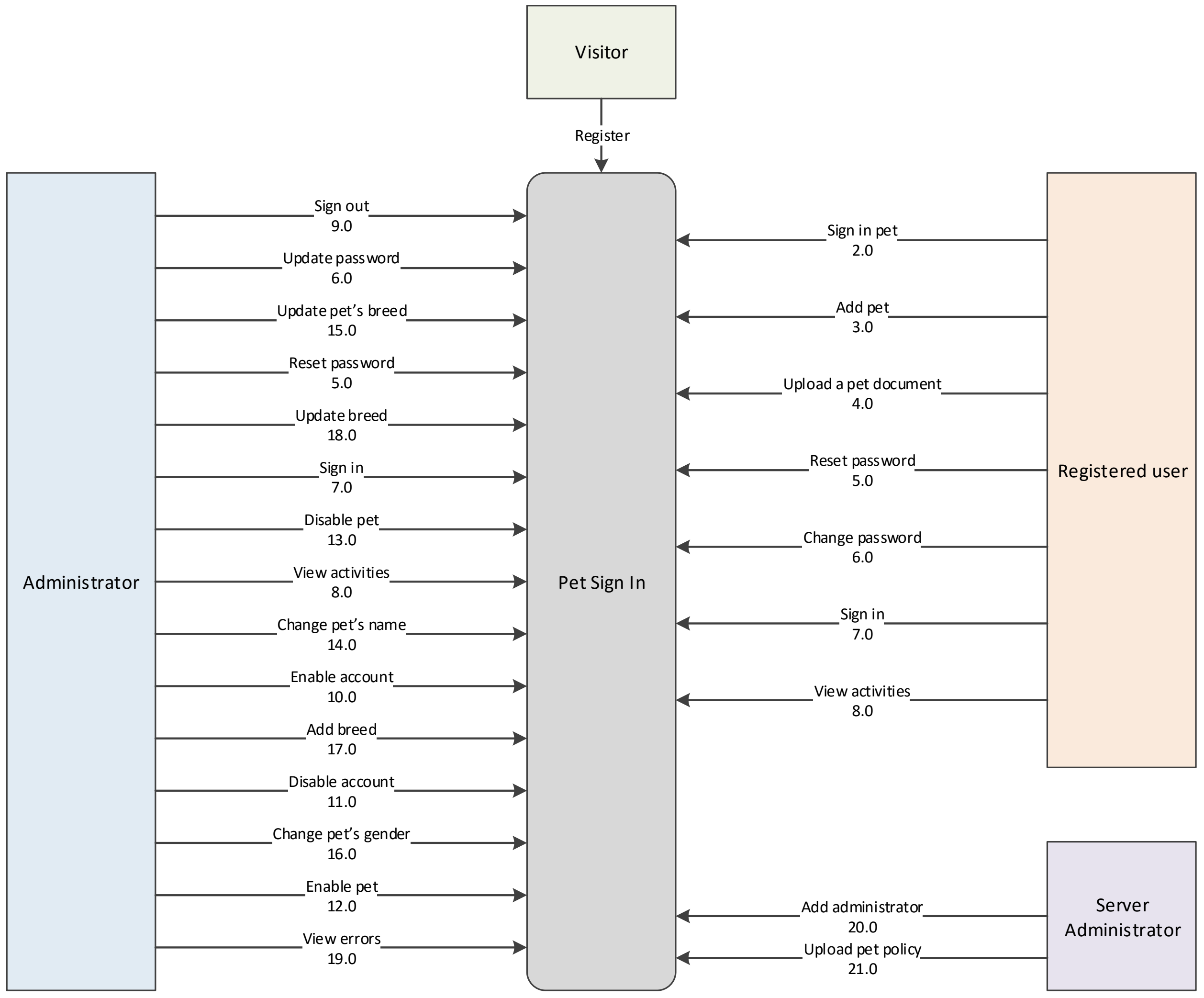
4.4.5	OTG-IDENT-005	Testing for Weak or unenforced username policy
4.4.7	OTG-IDENT-007	Test Account Suspension/Resumption Process
4.5		**Authentication Testing**
4.5.1	OTG-AUTHN-001	Testing for Credentials Transported over an Encrypted Channel
4.5.2	OTG-AUTHN-002	Testing for default credentials
4.5.3	OTG-AUTHN-003	Testing for Weak lock out mechanism
4.5.4	OTG-AUTHN-004	Testing for bypassing authentication schema
4.5.5	OTG-AUTHN-005	Test remember password functionality
4.5.6	OTG-AUTHN-006	Testing for Browser cache weakness
4.5.7	OTG-AUTHN-007	Testing for Weak password policy
4.5.8	OTG-AUTHN-008	Testing for Weak security question/answer
4.5.9	OTG-AUTHN-009	Testing for weak password change or reset functionalities
4.5.10	OTG-AUTHN-010	Testing for Weaker authentication in alternative channel
4.6		**Authorization Testing**
4.6.1	OTG-AUTHZ-001	Testing Directory traversal/file include
4.6.2	OTG-AUTHZ-002	Testing for bypassing authorization schema
4.6.3	OTG-AUTHZ-003	Testing for Privilege Escalation
4.7		**Session Management Testing**
4.7.2	OTG-SESS-002	Testing for Cookies attributes
4.7.3	OTG-SESS-003	Testing for Session Fixation
4.7.4	OTG-SESS-004	Testing for Exposed Session Variables
4.7.5	OTG-SESS-005	Testing for Cross Site Request Forgery
4.7.6	OTG-SESS-006	Testing for logout functionality
4.7.7	OTG-SESS-007	Test Session Timeout
4.8		**Data Validation Testing**
4.8.1	OTG-INPVAL-001	Testing for Reflected Cross Site Scripting
4.8.2	OTG-INPVAL-002	Testing for Stored Cross Site Scripting
4.8.15	OTG-INPVAL-015	Testing for incubated vulnerabilities
4.8.16	OTG-INPVAL-016	Testing for HTTP Splitting/Smuggling
4.9		**Error Handling**
4.9.1	OTG-ERR-001	Analysis of Error Codes
4.9.2	OTG-ERR-002	Analysis of Stack Traces
4.1		**Cryptography**
4.10.1	OTG-CRYPST-001	Testing for Weak SSL/TSL Ciphers, Insufficient Transport Layer Protection
4.11		**Business Logic Testing**
4.11.1	OTG-BUSLOGIC-001	Test Business Logic Data Validation
4.11.2	OTG-BUSLOGIC-002	Test Ability to Forge Requests
4.11.3	OTG-BUSLOGIC-003	Test Integrity Checks
4.11.7	OTG-BUSLOGIC-007	Test Defenses Against Application Mis-use
4.11.8	OTG-BUSLOGIC-008	Test Upload of Unexpected File Types
4.11.9	OTG-BUSLOGIC-009	Test Upload of Malicious Files
4.12		**Client Side Testing**
4.12.2	OTG-CLIENT-002	Testing for JavaScript Execution
4.12.3	OTG-CLIENT-003	Testing for HTML Injection
4.12.4	OTG-CLIENT-004	Testing for Client Side URL Redirect

My web application failed to the follow important test:

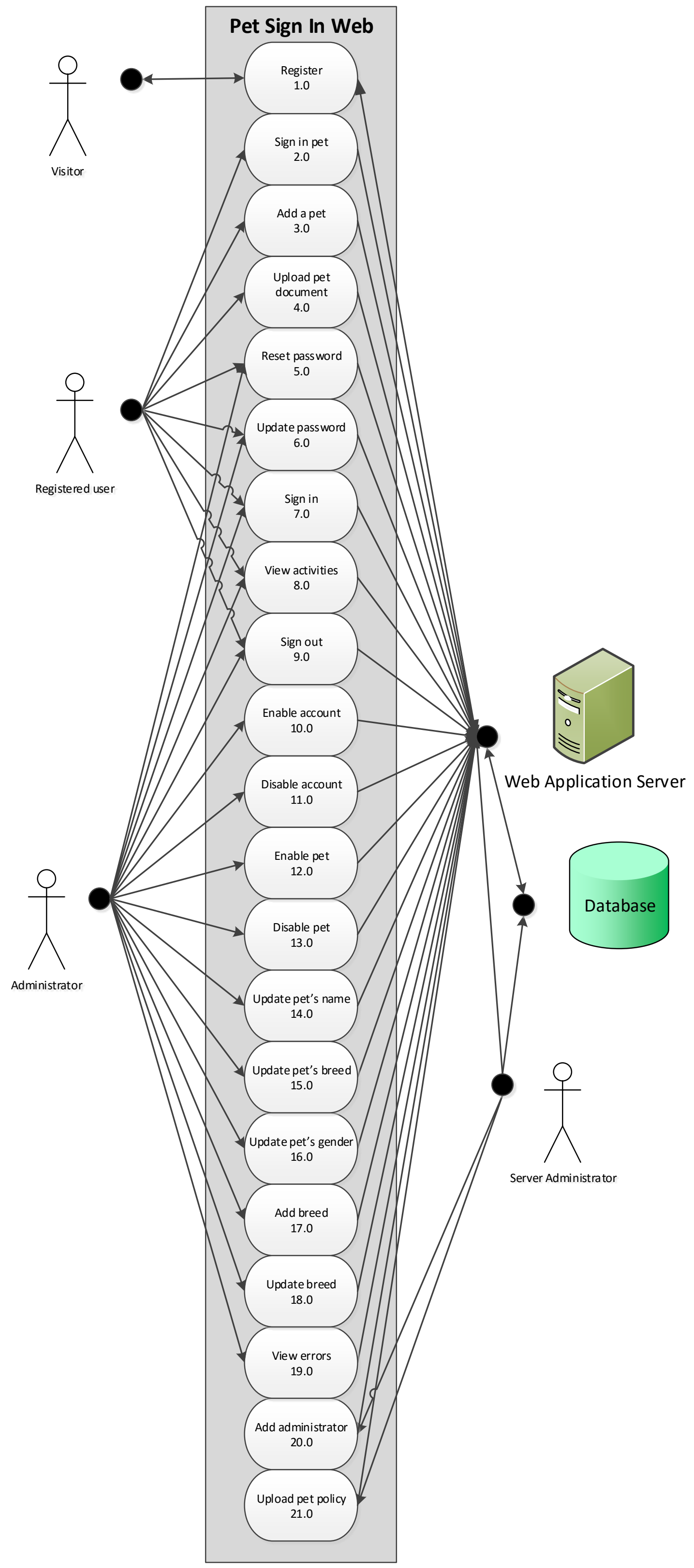
- Testing for Incubated Vulnerability (OTG-INPVAL-015)
- Test for Process Timing (OTG-BUSLOGIC-004)
- Test number of times a function can be used limits (OTG-BUSLOGIC-005)
- Testing for the Circumvention of Work Flows (OTG-BUSLOGIC-006)

5 Design

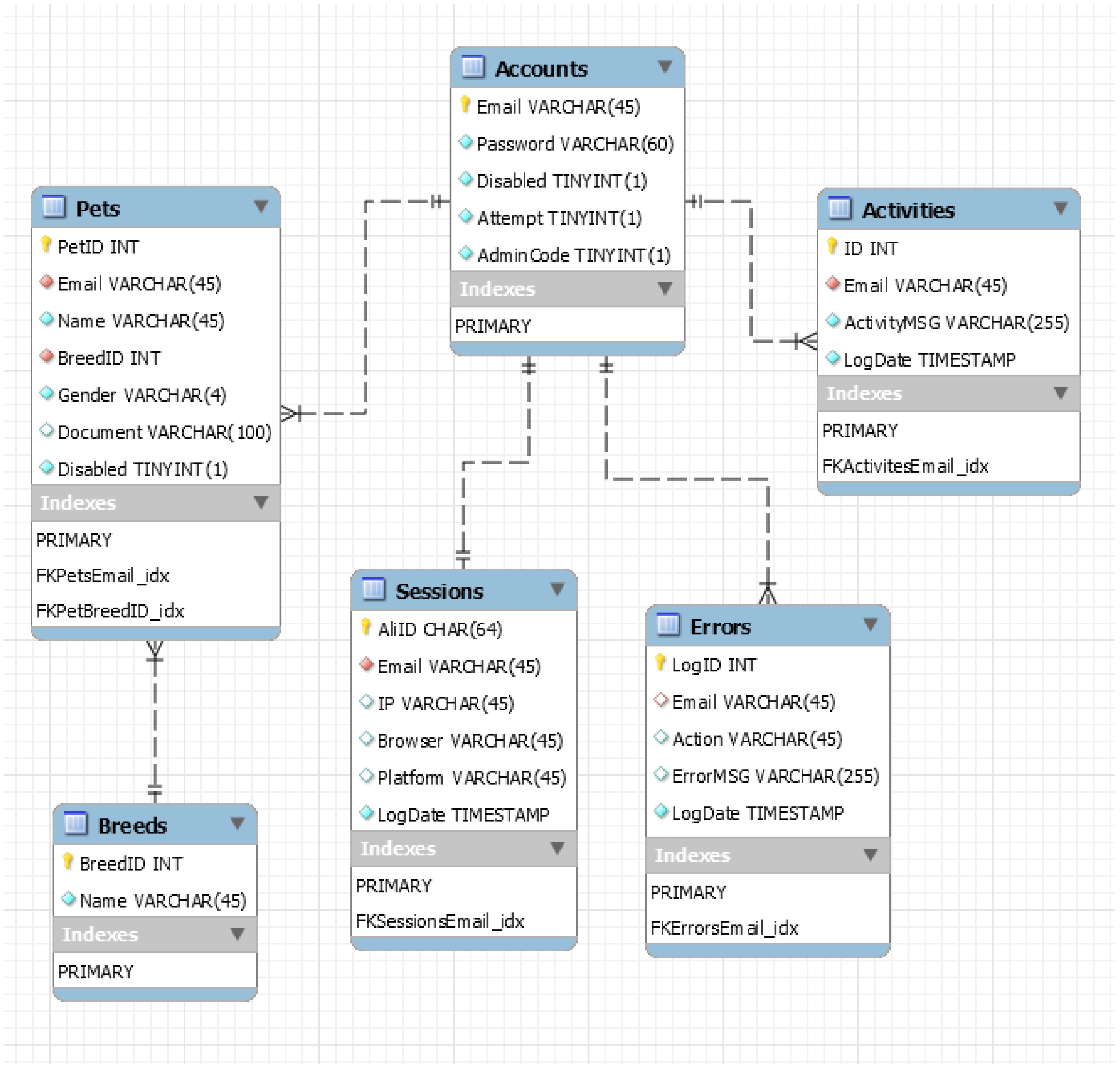
5.1 Context Dataflow Diagram



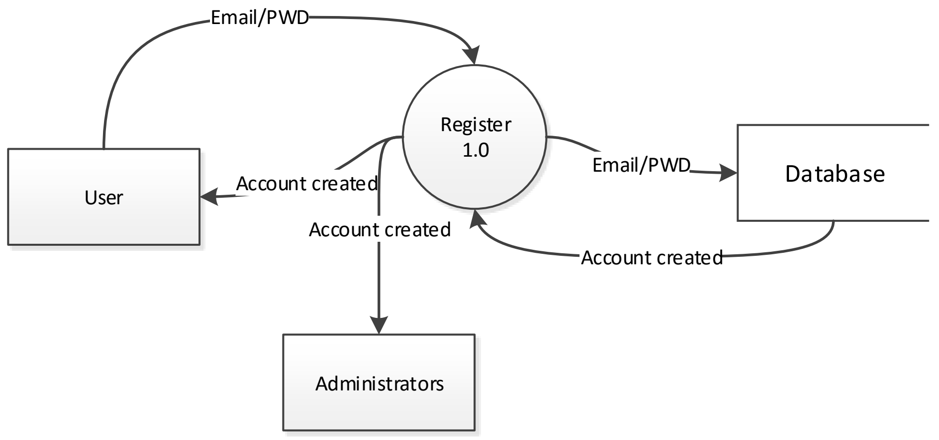
5.2 Use Case Diagram

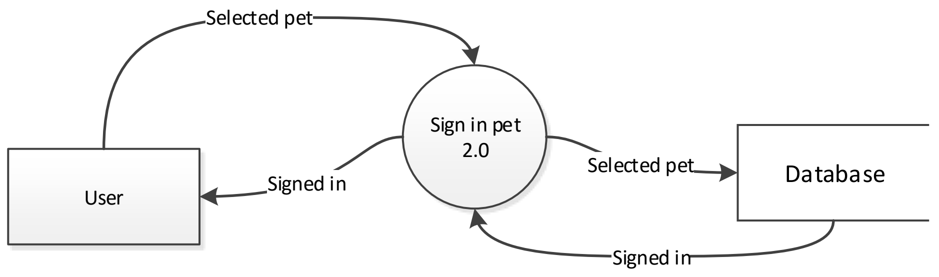

Pet Sign In Web
Register 1.0
Sign in pet 2.0
Add a pet 3.0
Upload pet document 4.0
Reset password 5.0
Update password 6.0
Sign in 7.0
View activities 8.0
Sign out 9.0
Enable account 10.0
Disable account 11.0
Enable pet 12.0
Disable pet 13.0
Update pet's name 14.0
Update pet's breed 15.0
Update pet's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

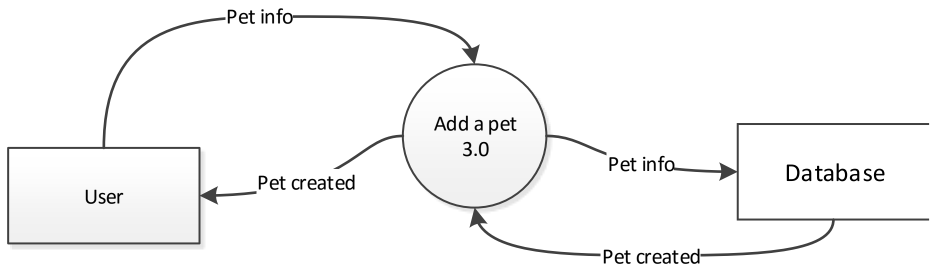
5.3 Entity Relationship Diagram

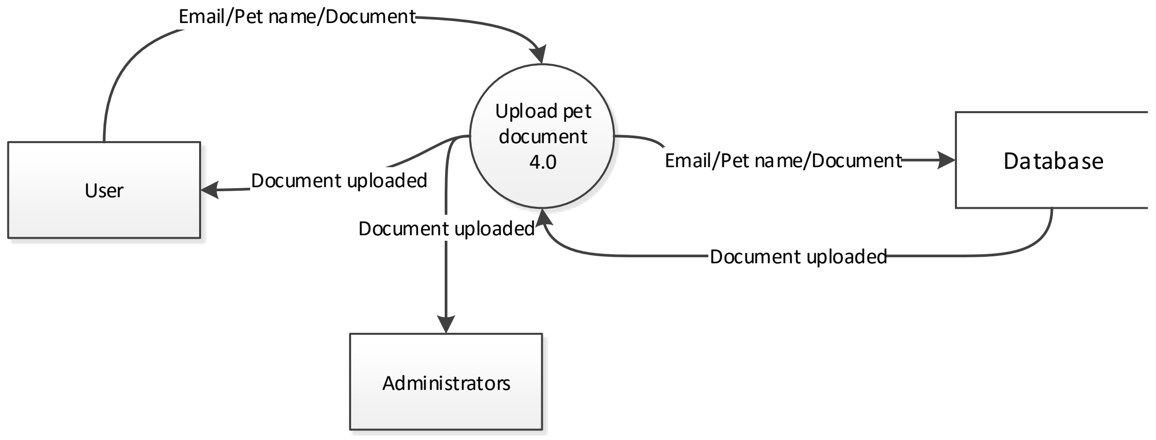


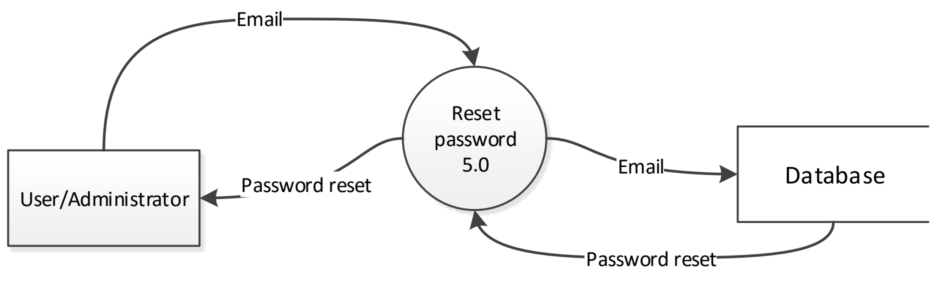
5.5 Dataflow Diagram

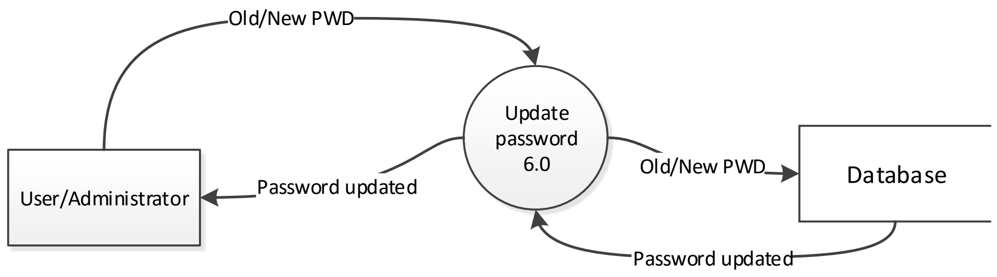


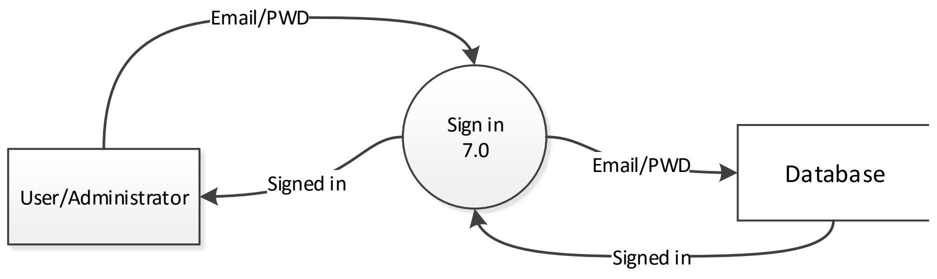


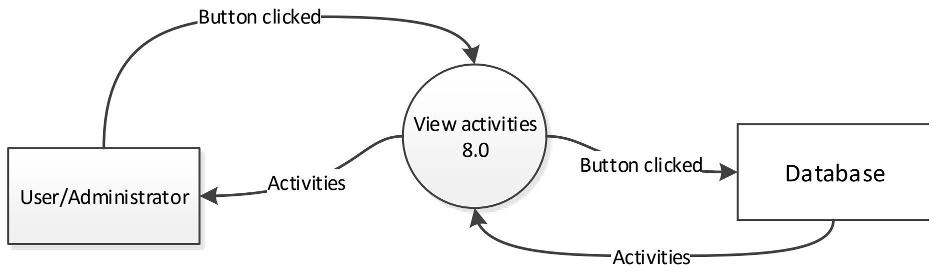


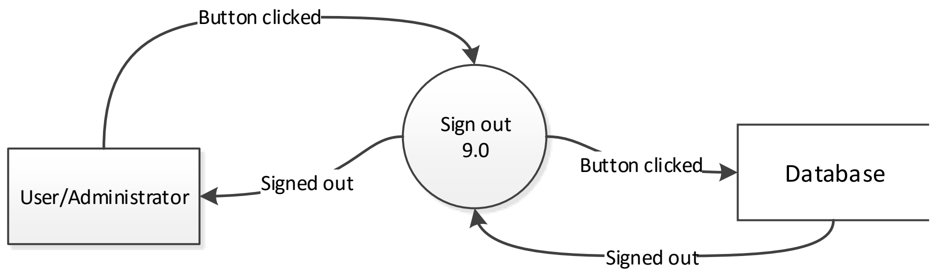


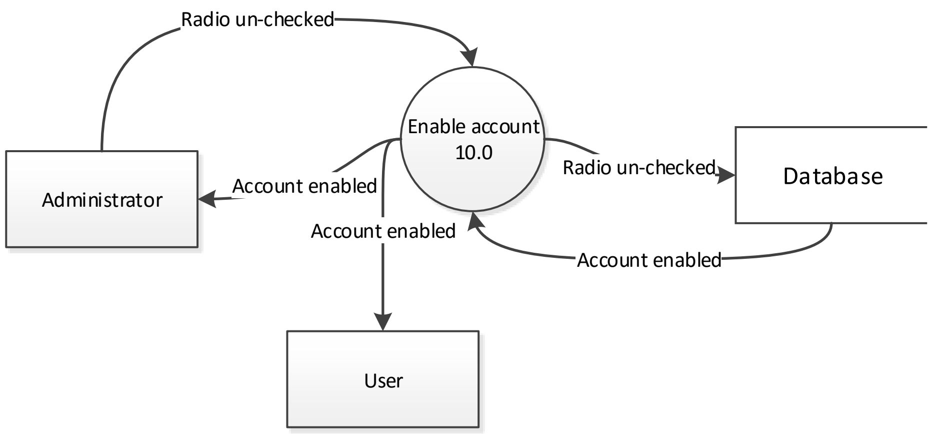

Email
Reset password 5.0
User/Administrator
Password reset
Email
Database
Password reset

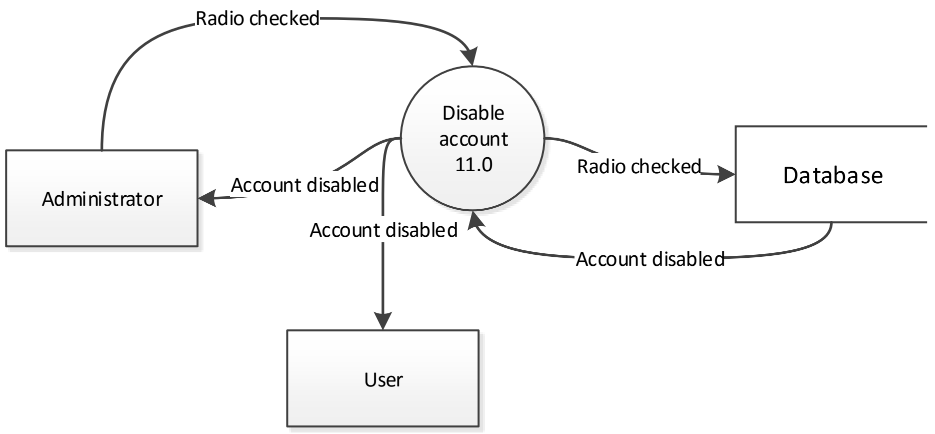

Old/New PWD
Update password 6.0
User/Administrator
Password updated
Old/New PWD
Database
Password updated

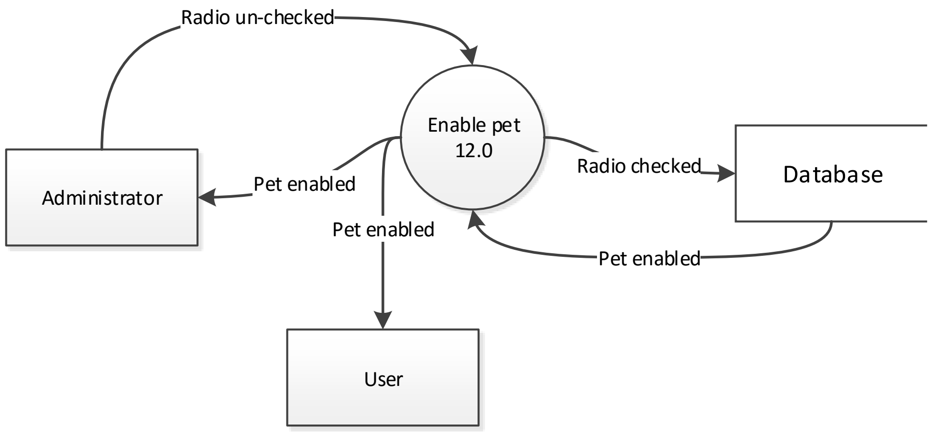

Email/PWD
Sign in 7.0
User/Administrator
Signed in
Email/PWD
Database
Signed in

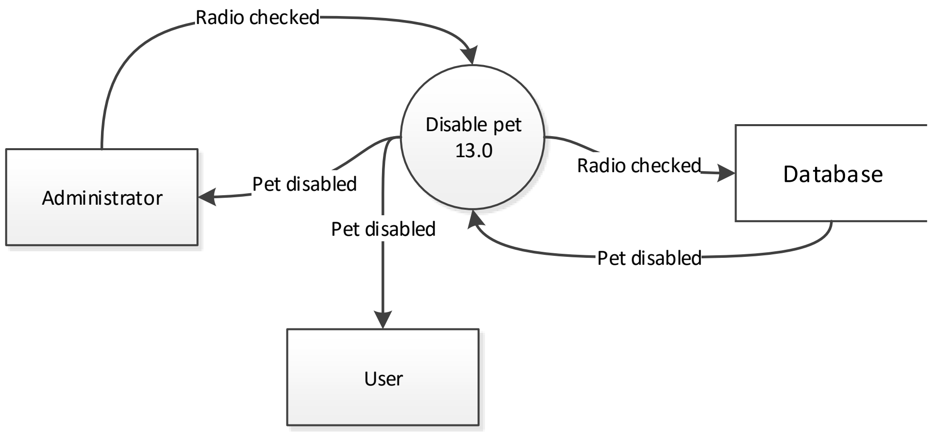

Button clicked
View activities 8.0
User/Administrator
Activities
Button clicked
Database
Activities

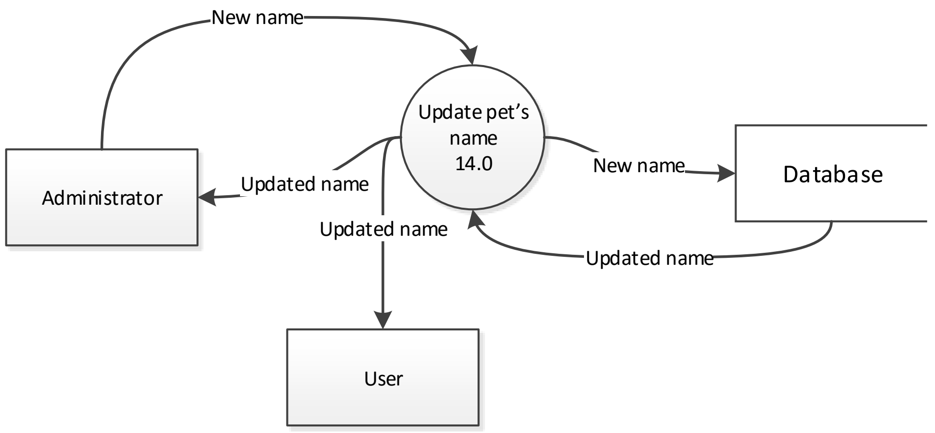

Button clicked
Sign out 9.0
User/Administrator
Signed out
Button clicked
Database
Signed out

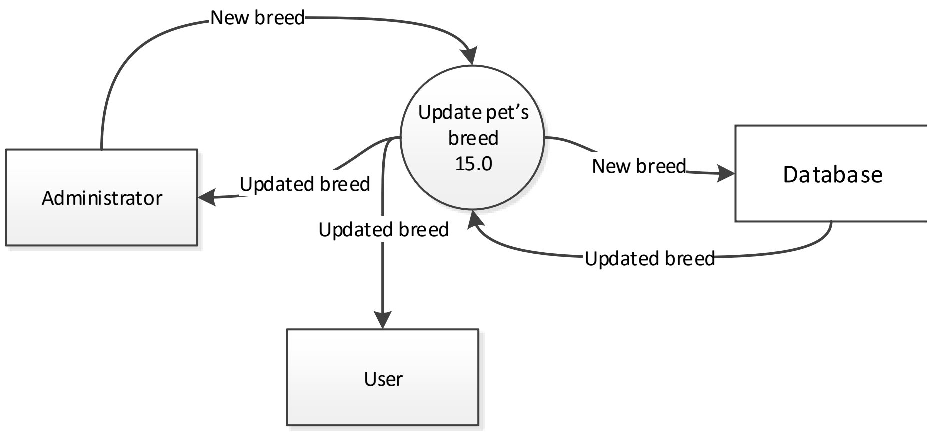

Radio un-checked
Enable account 10.0
Administrator
Account enabled
Account enabled
Radio un-checked
Database
Account enabled
User

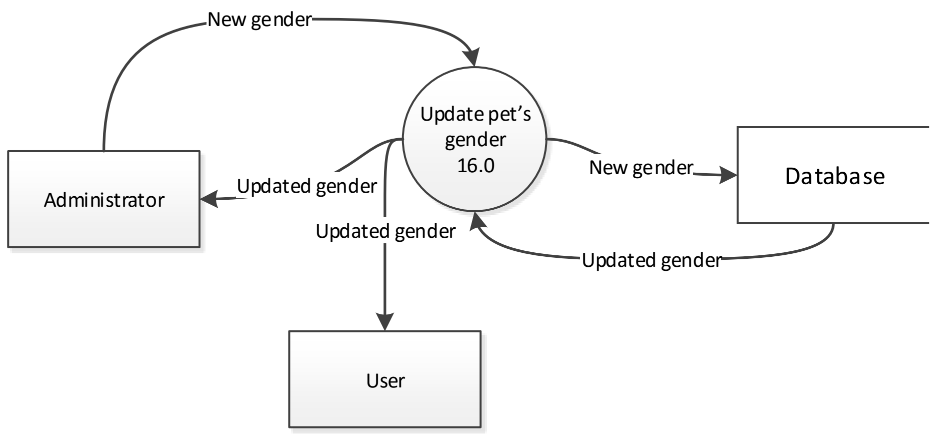

Radio checked
Disable account 11.0
Administrator
Account disabled
Account disabled
Radio checked
Database
Account disabled
User

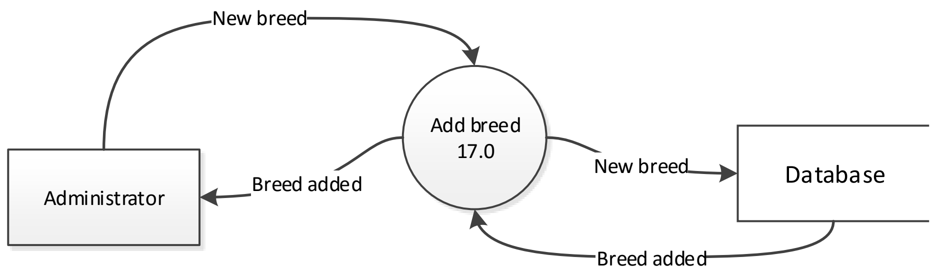

Radio un-checked
Enable pet 12.0
Administrator
Pet enabled
Pet enabled
Radio checked
Database
Pet enabled
User

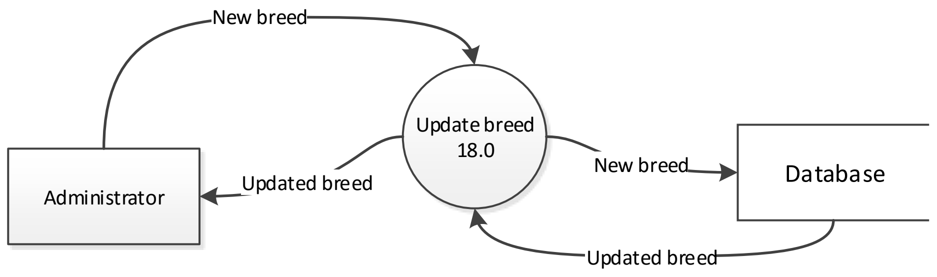

Radio checked
Disable pet
13.0
Administrator
Pet disabled
Pet disabled
Radio checked
Database
Pet disabled
User

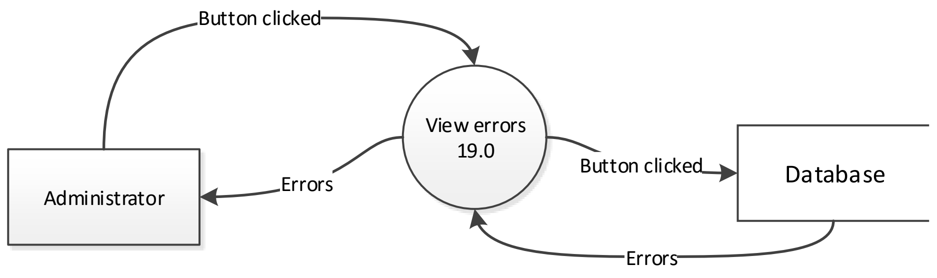

New name
Update pet's
name
14.0
Administrator
Updated name
Updated name
New name
Database
Updated name
User

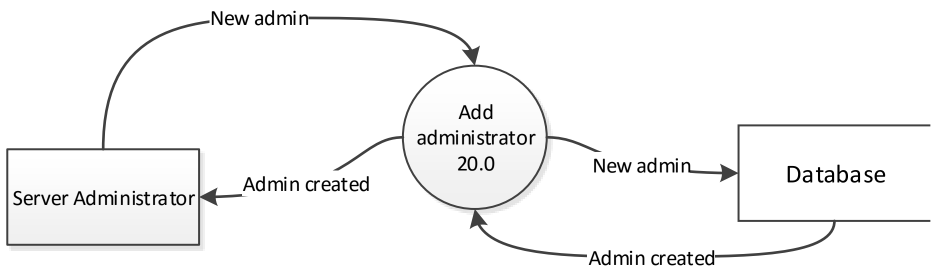

New breed
Update pet's
breed
15.0
Administrator
Updated breed
Updated breed
New breed
Database
Updated breed
User

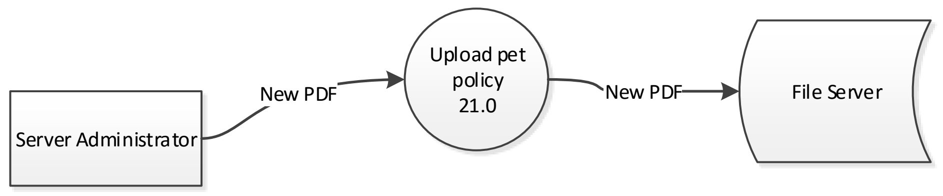

New gender
Update pet's gender 16.0
Administrator
Updated gender
Updated gender
New gender
Database
Updated gender
User


New breed
Add breed 17.0
Administrator
Breed added
New breed
Database
Breed added

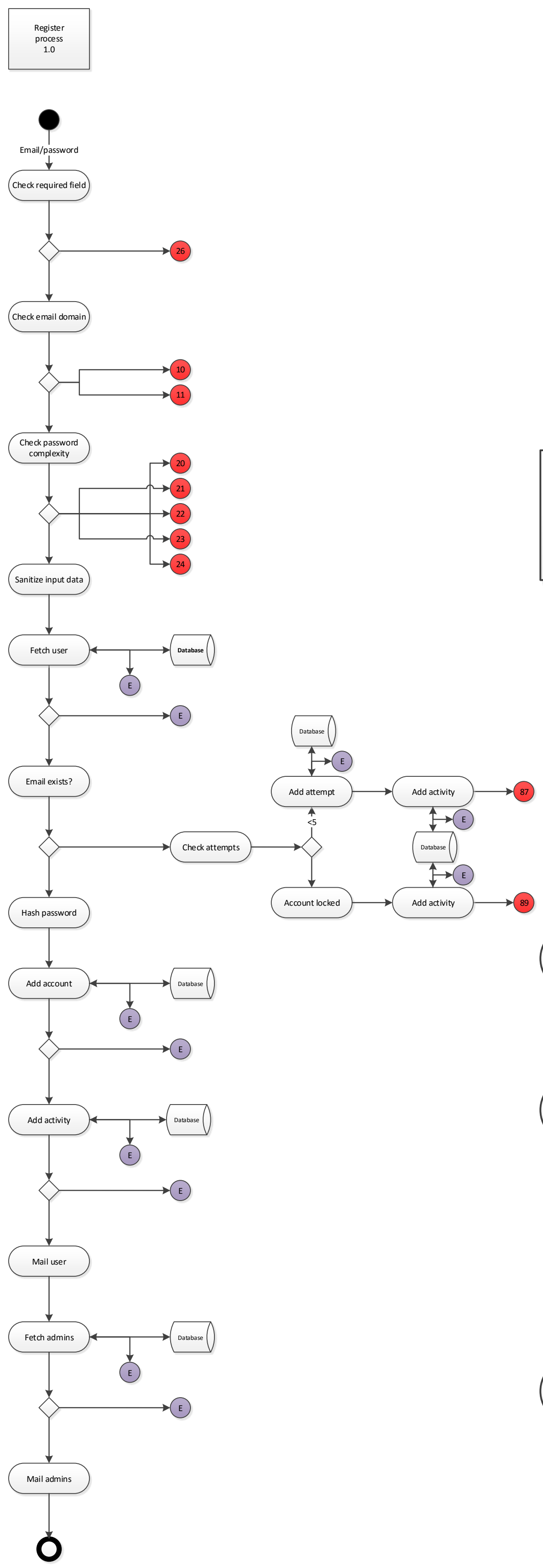

New breed
Update breed 18.0
Administrator
Updated breed
New breed
Database
Updated breed

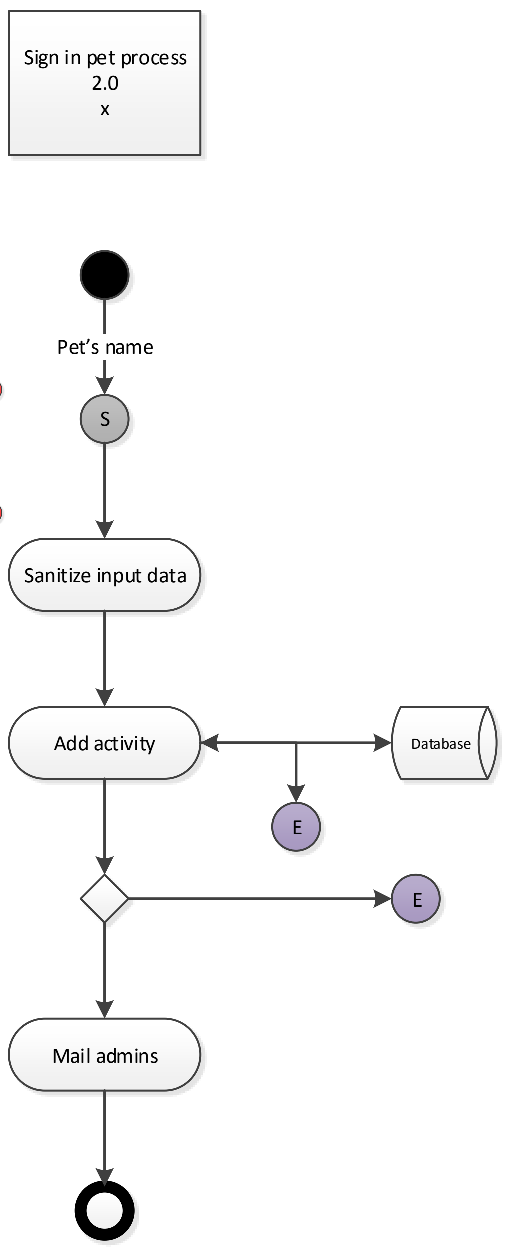

Button clicked
View errors 19.0
Administrator
Errors
Button clicked
Database
Errors

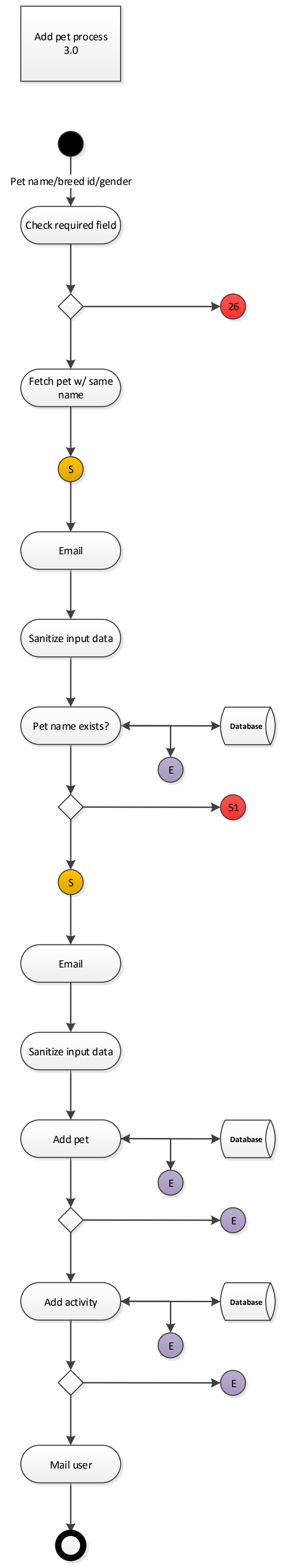

New admin
Add administrator 20.0
Server Administrator
Admin created
New admin
Database
Admin created

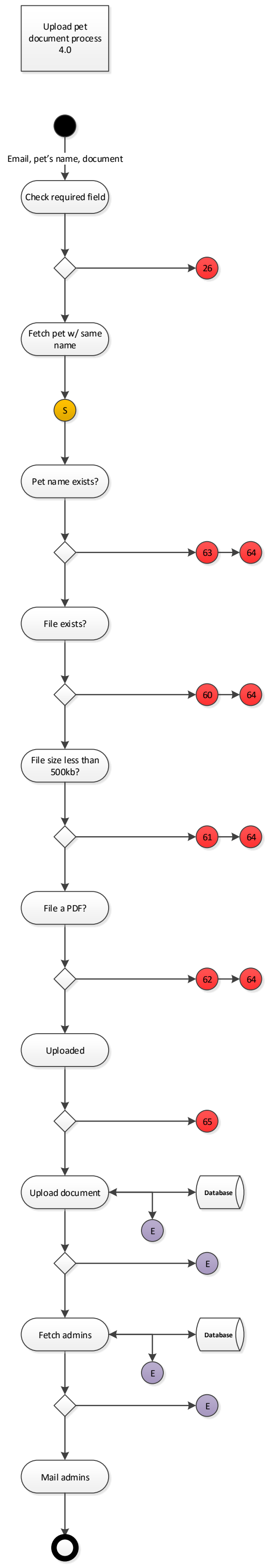

Server Administrator
New PDF
Upload pet policy 21.0
New PDF
File Server

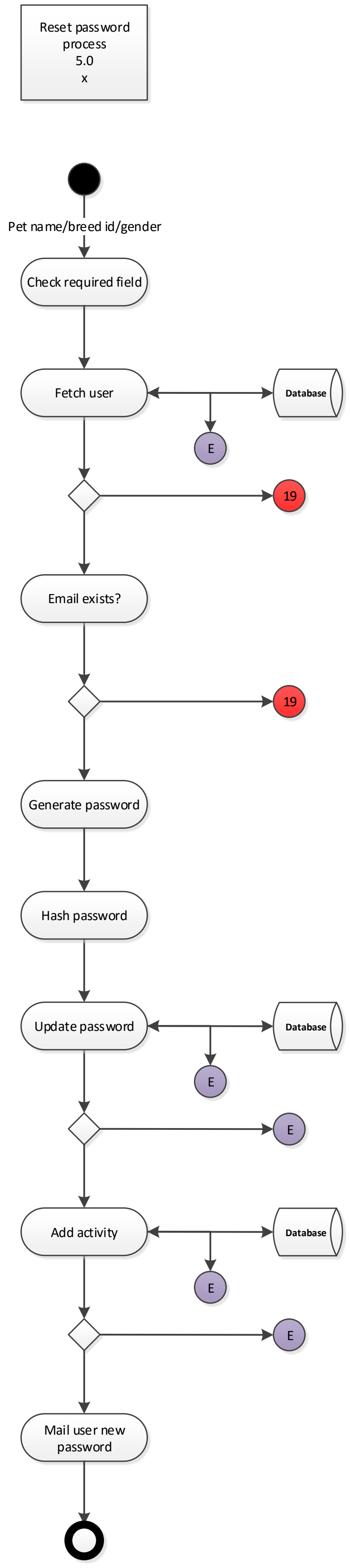
5.6 Activity Diagram

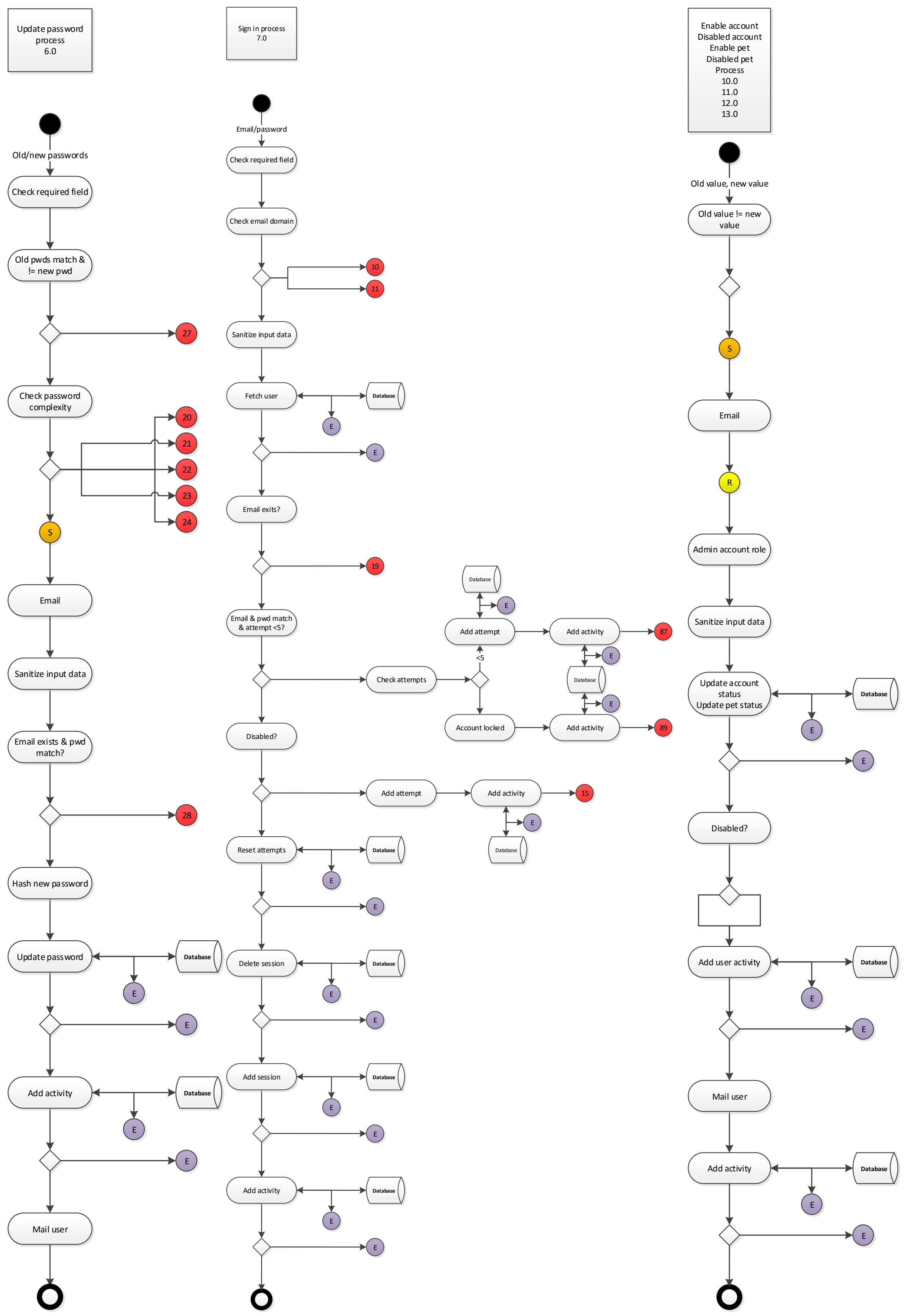

Register process 1.0
Email/password
Check required field
26
Check email domain
10
11
Check password complexity
20
21
22
23
24
Sanitize input data
Fetch user
Database
E
E
Email exists?
Check attempts
<5
Database
E
Add attempt
Add activity
87
E
Database
E
Account locked
Add activity
89
Hash password
Add account
Database
E
E
Add activity
Database
E
E
Mail user
Fetch admins
Database
E
E
Mail admins


Sign in pet process 2.0 x
Pet's name
S
Sanitize input data
Add activity
Database
E
E
Mail admins

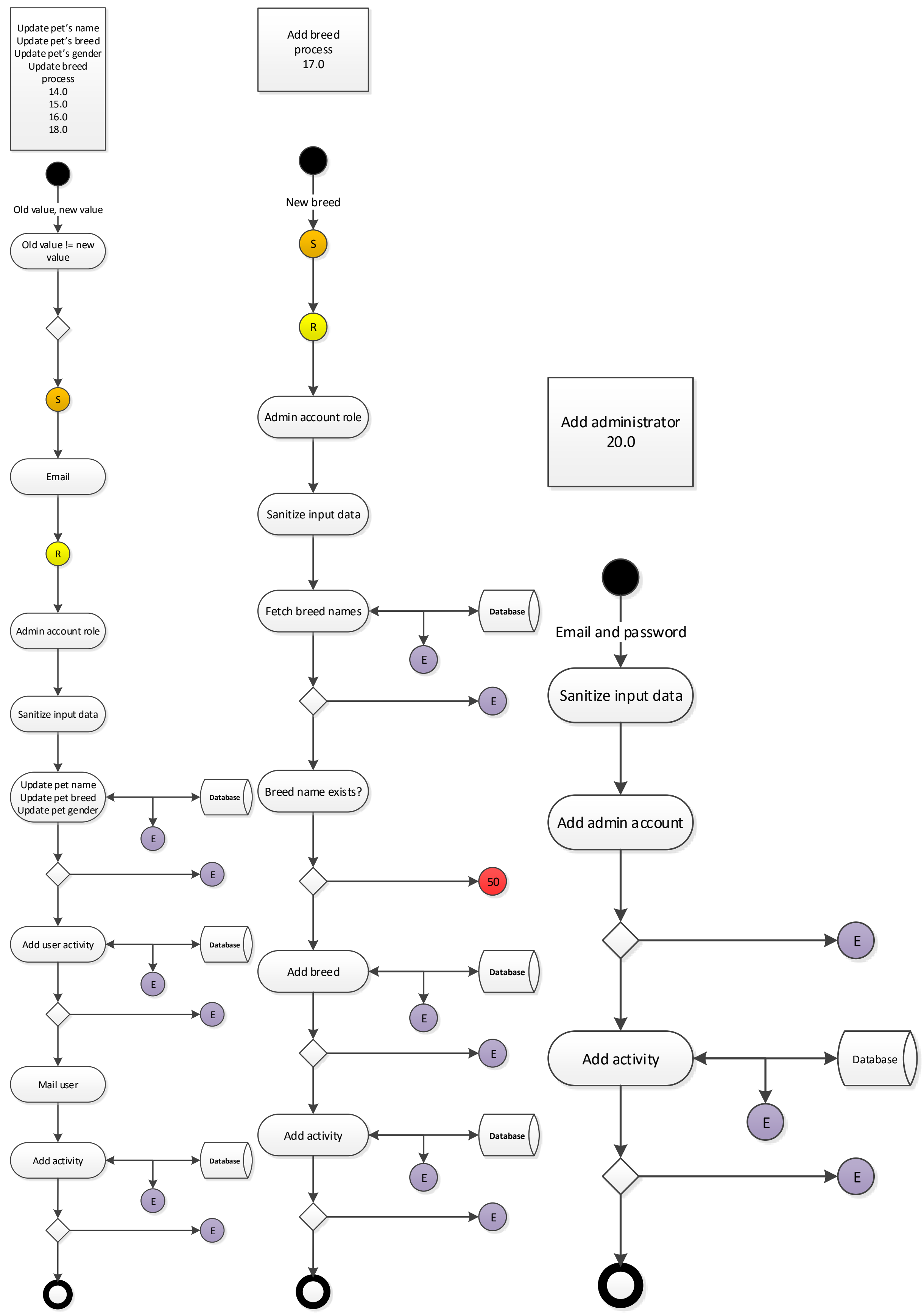

Add pet process
3.0
Pet name/breed id/gender
Check required field
26
Fetch pet w/ same name
S
Email
Sanitize input data
Pet name exists?
Database
E
51
S
Email
Sanitize input data
Add pet
Database
E
E
Add activity
Database
E
E
Mail user

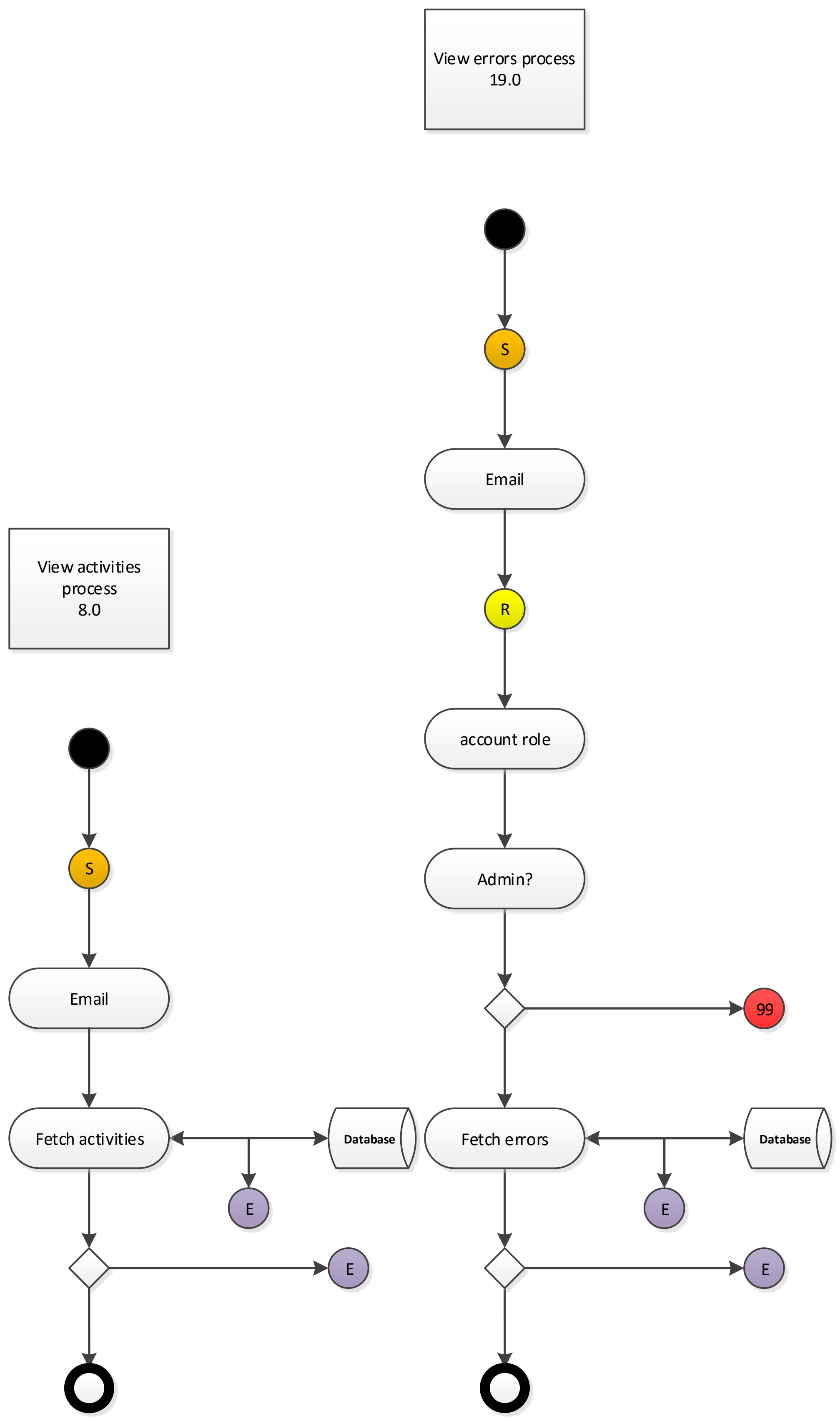

Upload pet document process
4.0
Email, pet's name, document
Check required field
26
Fetch pet w/ same name
S
Pet name exists?
63
64
File exists?
60
64
File size less than 500kb?
61
64
File a PDF?
62
64
Uploaded
65
Upload document
Database
E
E
Fetch admins
Database
E
E
Mail admins


Reset password process
5.0
x
Pet name/breed id/gender
Check required field
Fetch user
Database
E
19
Email exists?
19
Generate password
Hash password
Update password
Database
E
E
Add activity
Database
E
E
Mail user new password

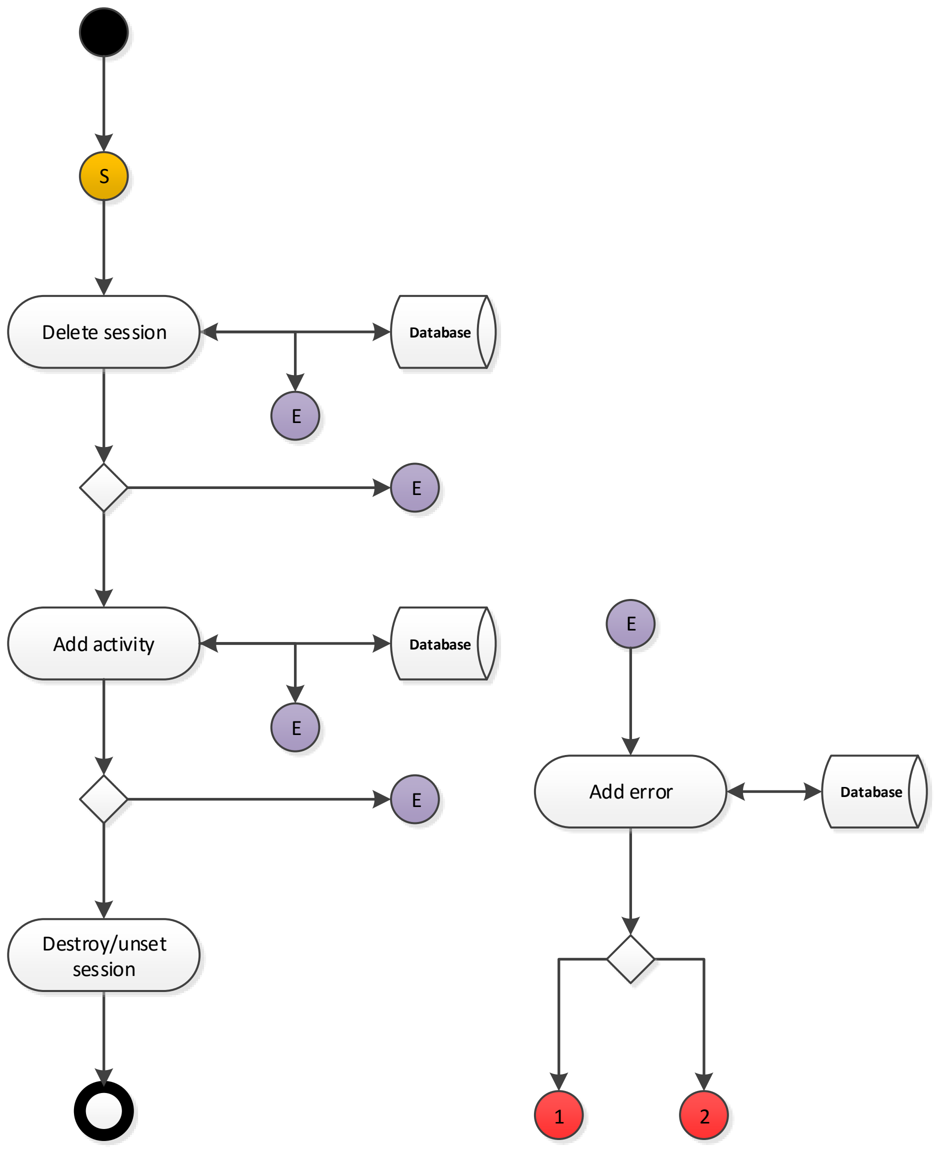


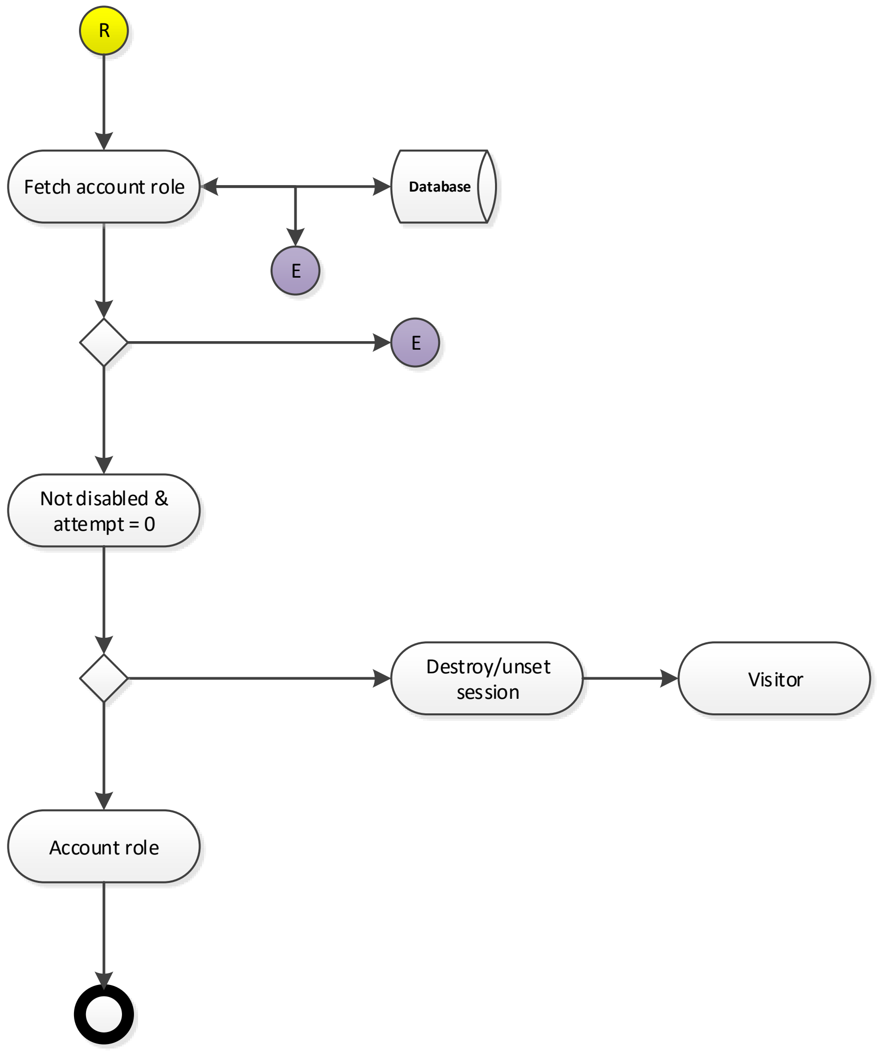
Update password process 6.0
Old/new passwords
Check required field
Old pwds match & != new pwd
27
Check password complexity
20
21
22
23
24
S
Email
Sanitize input data
Email exists & pwd match?
28
Hash new password
Update password
Database
E
Add activity
Database
E
Mail user
Sign in process 7.0
Email/password
Check required field
Check email domain
10
11
Sanitize input data
Fetch user
Database
E
Email exits?
19
Email & pwd match & attempt <5?
Check attempts
<5
Add attempt
Database
Add activity
87
Account locked
Add activity
89
Disabled?
Add attempt
Add activity
15
Reset attempts
Delete session
Add session
Add activity
Enable account Disabled account Enable pet Disabled pet Process 10.0 11.0 12.0 13.0
Old value, new value
Old value != new value
S
Email
R
Admin account role
Sanitize input data
Update account status Update pet status
Database
E
Disabled?
Add user activity
Mail user
Add activity

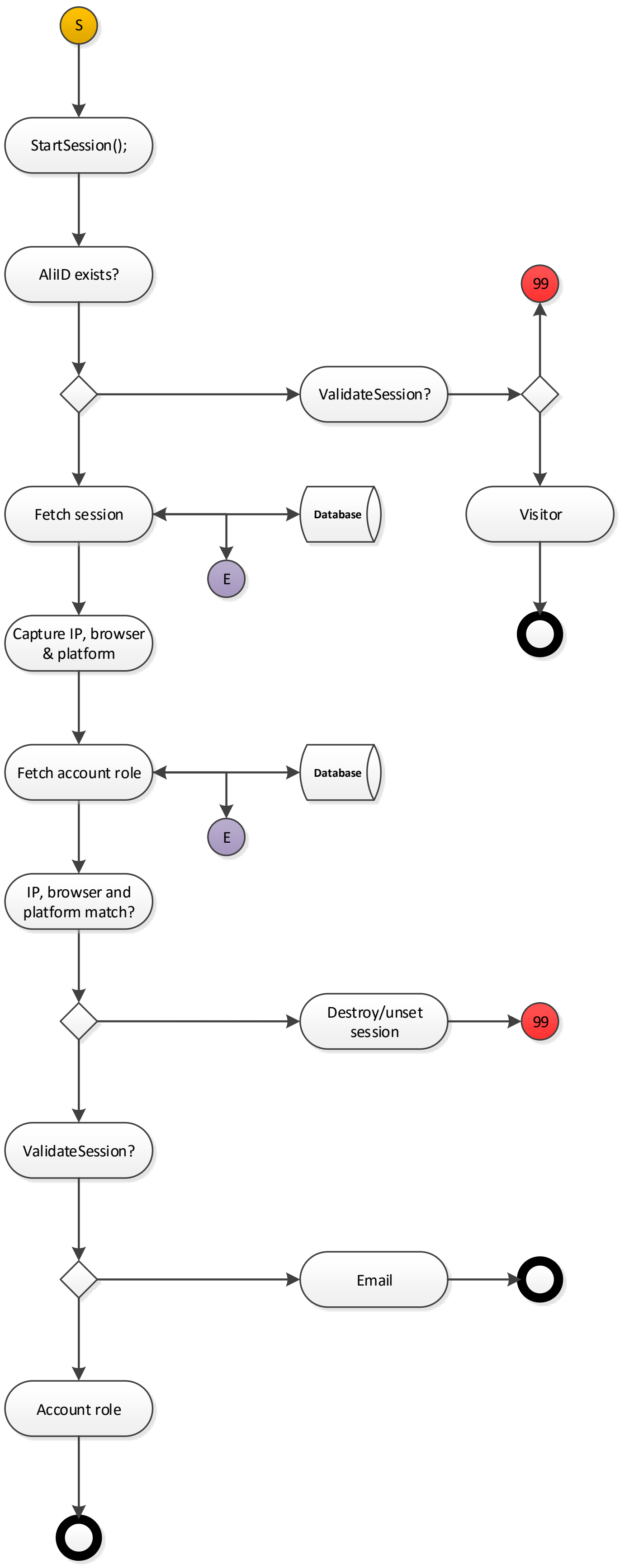

Update pet's name
Update pet's breed
Update pet's gender
Update breed process
14.0
15.0
16.0
18.0
Old value, new value
Old value != new value
S
Email
R
Admin account role
Sanitize input data
Update pet name
Update pet breed
Update pet gender
Database
E
Add user activity
Mail user
Add activity
Add breed process
17.0
New breed
Fetch breed names
Breed name exists?
50
Add breed
Add administrator
20.0
Email and password
Add admin account

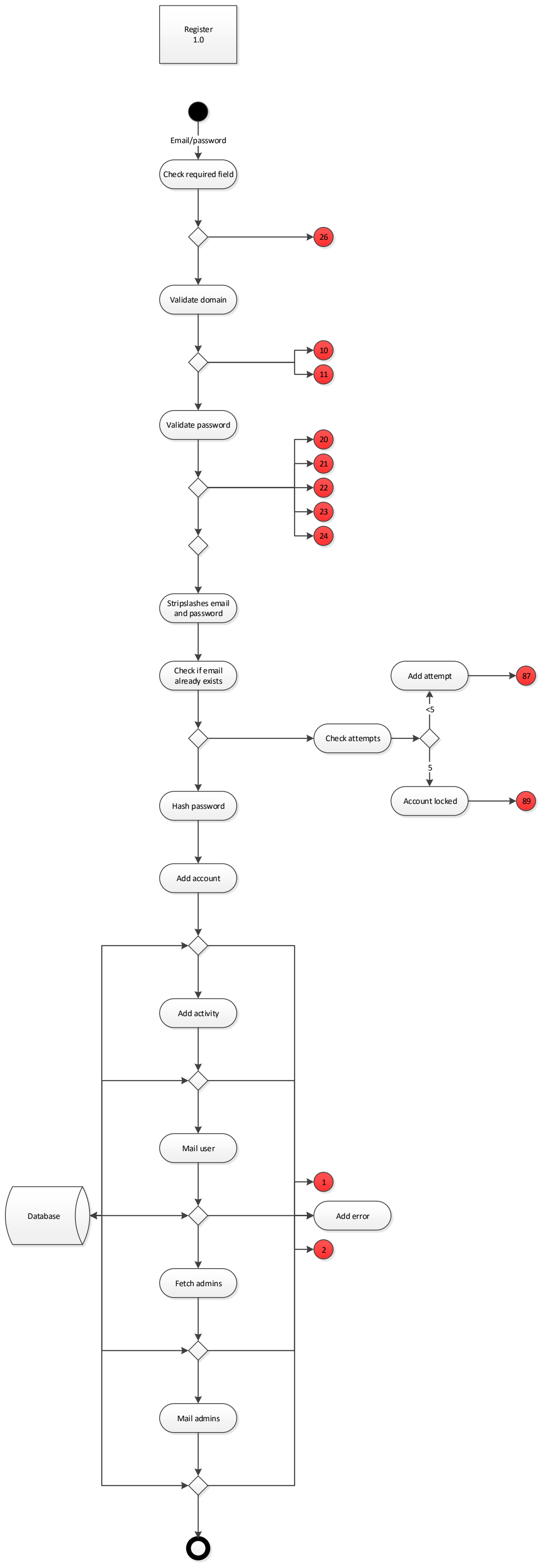

View errors process
19.0
S
Email
R
account role
Admin?
99
Fetch errors
Database
E
E
View activities process
8.0
S
Email
Fetch activities
Database
E
E

Sign out process
9.0

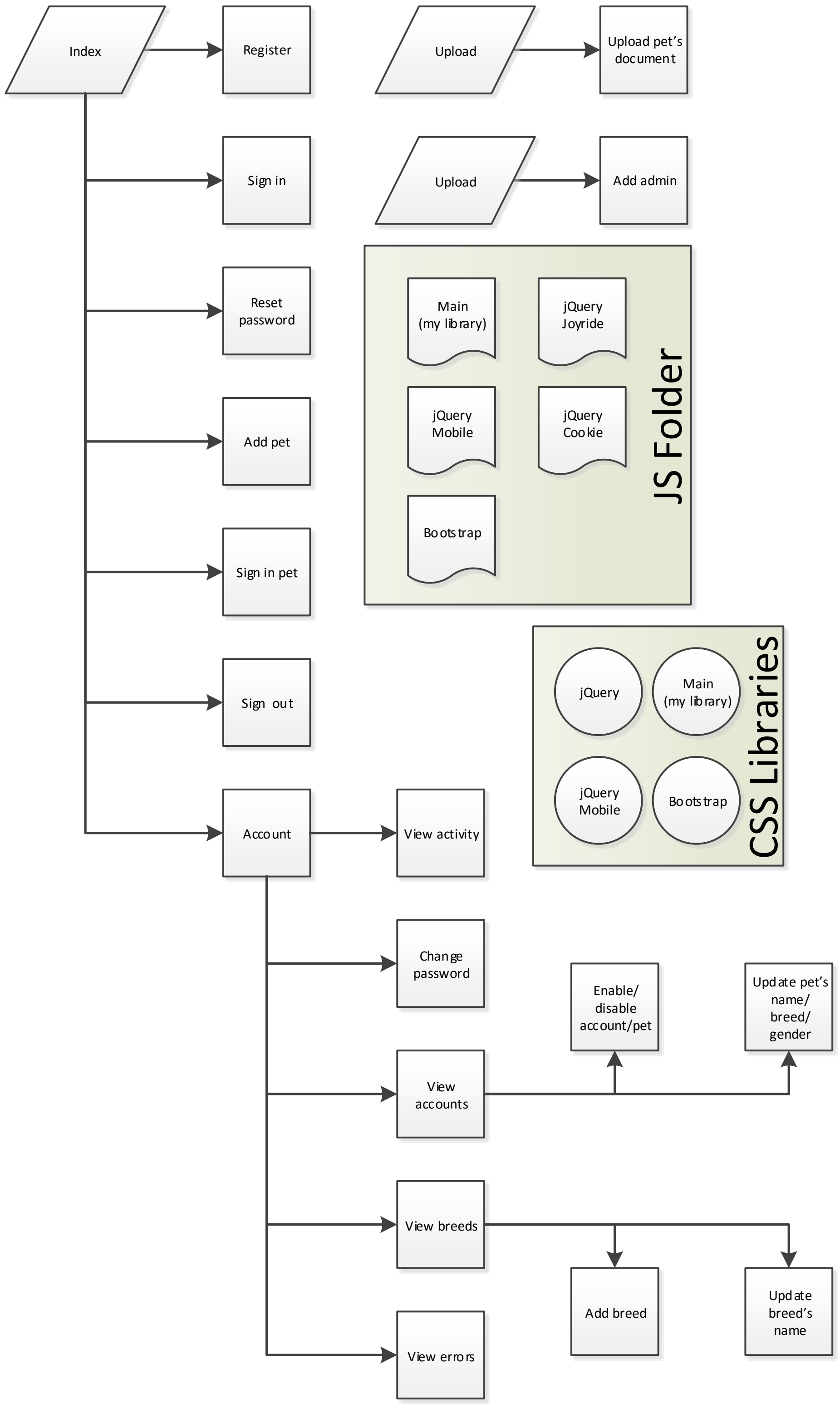



R
Fetch account role
Database
E
E
Not disabled & attempt = 0
Destroy/unset session
Visitor
Account role


S
StartSession();
AlIID exists?
ValidateSession?
99
Fetch session
Database
E
Visitor
Capture IP, browser & platform
Fetch account role
Database
E
IP, browser and platform match?
Destroy/unset session
99
ValidateSession?
Email
Account role


Register
1.0
Email/password
Check required field
26
Validate domain
10
11
Validate password
20
21
22
23
24
Stripslashes email and password
Check if email already exists
Add attempt
87
<5
Check attempts
5
Account locked
89
Hash password
Add account
Add activity
Mail user
1
Database
Add error
2
Fetch admins
Mail admins

5.7 Conceptual Website Diagram



5.8 Database Script

```
-- MySQL Workbench Forward Engineering


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` (
  `AliID` 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 (`AliID`)  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 xAliID VARCHAR(64), IN xEmail VARCHAR(45), IN xSessionIP
VARCHAR(45), IN xSessionBrowser VARCHAR(45), IN xSessionPlatform VARCHAR(45))

BEGIN

INSERT INTO djkabau1_petsignin.Sessions (AliID, Email, IP, Browser, Platform) VALUES (xAliID,
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 xAliID VARCHAR(64))

BEGIN

SELECT * FROM djkabau1_petsignin.Sessions WHERE AliID = (xAliID);

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 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;
```

6 Web Pages Code

6.1 index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" type="text/css" href="css/main.css">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" integrity="sha384-
1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGmkzs7" crossorigin="anonymous">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"
integrity="sha384-0mSbJDEHialfmuBBQP6A4Qrprq5OVfW37PRR3j5ELqxss1yVqOtnepnHVP9aJ7xS"
crossorigin="anonymous"></script>
    <script src="js/main.js" type="text/javascript"></script>
    <title>Pet Sign In</title>
</head>
<body>
<nav class="navbar navbar-default">
    <div class="container-fluid">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-
target="#bs-example-navbar-collapse-1" aria-expanded="false">
                <span class="sr-only">Toggle navigation</span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand" href="#">Pet Sign In</a>
        </div>
        <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
            <ul class="nav navbar-nav navbar-right">
                <li><a data-target="#AccountModal" data-toggle="modal" href="#"
id="Account"><span class="glyphicon glyphicon-user"></span>Account</a></li>
                <li><a href="#" id="SignOut"><span class="glyphicon glyphicon-log-
out"></span>Sign Out</a></li>
            </ul>
        </div>
    </div>
</nav>
<div class="btn-group-vertical" data-toggle="buttons" id="ViewSignInPet"></div>
<br>
<button type="button" class="btn btn-primary" data-dismiss="modal" data-target="#AddNewPetModal"
data-toggle="modal" id="AddNewPetButton">Add a new Pet</button>

<div id="Visitor">
    <p>Sign into your account</p>
    <div class="form-group">
        <label for="Email1">Email address</label><br>
        <input type="text" id="Email1" placeholder="Email address" autocomplete="off">
    </div>
    <div class="form-group">
        <label for="Password1">Password</label><br>
        <input type="password" id="Password1" placeholder="Password" autocomplete="off"><br>
    </div>
    <button type="button" class="btn btn-primary btn-sm" id="SignInButton">Sign In</button>
    <br>
    <a href="#" data-toggle="modal" data-target="#ResetAccountModal" id="ResetAccount" >Can't
access your account?</a>
    <br>
    <div class="modal fade" id="ResetAccountModal" role="dialog">
        <div class="modal-dialog modal-sm">
            <div class="modal-content">
                <div class="modal-header">
                    <button type="button" class="close" data-dismiss="modal">&times;</button>
                    <h4 class="modal-title">Reset your password</h4>
                </div>
                <div class="modal-body">
                    <div class="form-group">
                        <label for="Email3">Email address</label><br>
                        <input type="text" id="Email3" placeholder="Email address"
autocomplete="off">
                    </div>
                </div>
                <div class="modal-footer">
                    <button type="button" class="btn btn-primary btn-sm" id="ResetPassword"
disabled>Reset</button>
                    <button type="button" class="btn btn-default" data-
dismiss="modal">Close</button>
                </div>
            </div>
        </div>
    </div>
    <p>New to Pet Sign In?</p>
    <a href="#" data-toggle="modal" data-target="#RegisterModal" id="Register" >Register for a
new account</a>
    <br>
</div>

<div class="modal fade" id="RegisterModal" role="dialog">
    <div class="modal-dialog modal-sm">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal">&times;</button>
                <h4 class="modal-title">Register</h4>
            </div>
            <div class="modal-body">
                <div class="form-group">
                    <label for="Email2">Email address</label><br>
                    <input type="text" id="Email2" placeholder="Email address"
autocomplete="off">
                </div>
                <div class="form-group">
                    <label for="Password2">Password (req. At least 1 upper, lower case and a
number.</label><br>
                    <input type="password" id="Password2" placeholder="Password"
autocomplete="off"><br>
                </div>
                <div class="form-group">
                    <br><a href="https://petsignin.alibkaba.com/petsignin/petpolicy.pdf"
target="_blank">Terms and Conditions</a><br><br>
                    <input type="radio" name="accounttnc" value="no" id="accounttncno" checked>I
do not agree<br>
                    <input type="radio" name="accounttnc" value="yes">I agree<br>
                </div>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-success btn-sm" id="RegisterButton"
disabled>Register</button>
                <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
            </div>
        </div>
    </div>
</div>
```

```
<div class="modal fade" id="AddNewPetModal" role="dialog">
    <div class="modal-dialog modal-sm">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal">&times;</button>
                <h4 class="modal-title">Add a Pet</h4>
            </div>
            <div class="modal-body">
                <div class="form-group">
                    <label for="PetName">Pet's name</label><br>
                    <input type="text" id="PetName" placeholder="Pet's name" autocomplete="off">
                </div>
                <div class="form-group">
                    <label for="ViewBreeds">Pet's breed</label><br>
                    <select class="form-control" id="ViewBreeds">
                        <option value="0">Select your pet's breed</option>
                    </select>
                </div>
                <div class="btn-group">
                    <label for="ViewGenders">Pet's gender</label><br>
                    <select class="form-control" id="ViewGenders">
                        <option value="0">Select your pet's gender</option>
                        <option value="Boy">Boy</option>
                        <option value="Girl">Girl</option>
                    </select>
                </div>
                <div class="form-group">
                    <br><a href="https://petsignin.alibkaba.com/petsignin/petpolicy.pdf"
target="_blank">Terms and Conditions</a><br><br>
                    <input type="radio" name="pettnc" value="no" id="pettncno" checked>I do not
agree<br>
                    <input type="radio" name="pettnc" value="yes">I agree<br>
                </div>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-success" id="AddPetButton" disabled>Add
Pet</button>
                <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
            </div>
        </div>
    </div>
</div>

<div class="modal fade" id="AccountModal" role="dialog">
    <div class="modal-dialog modal-sm">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal">&times;</button>
                <h4 class="modal-title">Account</h4>
            </div>
            <div class="modal-body">
                <button type="button" class="btn btn-primary" data-dismiss="modal" data-
target="#ViewActivitiesModal" data-toggle="modal" id="ViewActivitiesButton">View
activities</button>
                <br>
                <button type="button" class="btn btn-primary" data-dismiss="modal" data-
target="#ChangePasswordModal" data-toggle="modal" id="ChangePasswordButton">Change
password</button>
                <br>
                <button type="button" class="btn btn-primary" data-dismiss="modal" data-
target="#ViewAccountsModal" data-toggle="modal" id="ViewAccountsButton">View accounts</button>
                <br>
                <button type="button" class="btn btn-primary" data-dismiss="modal" data-
target="#ViewBreedsModal" data-toggle="modal" id="ViewBreedsButton">View breeds</button>
                <br>
                <button type="button" class="btn btn-primary" data-dismiss="modal" data-
target="#ViewErrorModal" data-toggle="modal" id="ViewErrorsButton">View errors</button>
                <br>
            </div>
            <div class="modal-footer">
```

```
                <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
            </div>
        </div>
    </div>
</div>

<div class="modal fade" id="ViewBreedsModal" role="dialog">
    <div class="modal-dialog modal-lg">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal">&times;</button>
                <h4 class="modal-title">Manage Breeds</h4>
            </div>
            <div class="modal-body">
                <div class="form-group">
                    <label for="ViewBreeds1" id="ViewBreedsLabel">Breeds</label><br>
                    <select class="form-control" id="ViewBreeds1">
                        <option value="0">Select a breed</option>
                    </select>
                </div>
                <div class="form-group">
                    <label for="ViewBreedName" id="ViewBreedNameLabel">Breed's name</label><br>
                    <input type="text" id="ViewBreedName" placeholder="Name" disabled
autocomplete="off">
                </div>
                <div class="form-group">
                    <label for="AddNewBreed" id="AddNewBreedLabel">Add a new breed</label><br>
                    <input type="text" id="AddNewBreed" placeholder="New breed name"
autocomplete="off">
                </div>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-default" data-dismiss="modal" data-
target="#AccountModal" data-toggle="modal">Back</button>
                <button type="button" class="btn btn-warning" data-dismiss="modal"
id="UpdateBreedButton" disabled>Update</button>
                <button type="button" class="btn btn-success" data-dismiss="modal"
id="AddBreedButton" disabled>Add</button>
                <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
            </div>
        </div>
    </div>
</div>

<div class="modal fade" id="ChangePasswordModal" role="dialog">
    <div class="modal-dialog modal-sm">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal">&times;</button>
                <h4 class="modal-title">Change password</h4>
            </div>
            <div class="modal-body">
                <div class="form-group">
                    <label for="OldPassword">Enter old password</label><br>
                    <input type="password" id="OldPassword" placeholder="Old password">
                </div>
                <div class="form-group">
                    <label for="OldPassword1">Enter old password again</label><br>
                    <input type="password" id="OldPassword1" placeholder="Old password again"
autocomplete="off">
                </div>
                <div class="form-group">
                    <label for="NewPassword">Enter new password</label><br>
                    <input type="password" id="NewPassword" placeholder="New password"
autocomplete="off">
                </div>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-default" data-dismiss="modal" data-
target="#AccountModal" data-toggle="modal">Back</button>
                <button type="button" class="btn btn-warning btn-sm" id="UpdatePasswordButton"
```

```
disabled>Change password</button>
                <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
            </div>
        </div>
    </div>
</div>

<div class="modal fade" id="ViewActivitiesModal" role="dialog">
    <div class="modal-dialog modal-lg">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal">&times;</button>
                <h4 class="modal-title">Activities</h4>
            </div>
            <div class="modal-body">
                <table class="table table-bordered" id="ViewActivities">
                    <thead>
                    <tr>
                        <th>Activity</th>
                        <th>Date</th>
                    </tr>
                    </thead>
                </table>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-default" data-dismiss="modal" data-
target="#AccountModal" data-toggle="modal">Back</button>
                <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
            </div>
        </div>
    </div>
</div>

<div class="modal fade" id="ViewAccountsModal" role="dialog">
    <div class="modal-dialog modal-lg">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal">&times;</button>
                <h4 class="modal-title">Manage Accounts</h4>
            </div>
            <div class="modal-body">
                <div class="form-group">
                    <label for="ViewAllAccounts">Select an account</label><br>
                    <select class="form-control" id="ViewAllAccounts">
                        <option value="0">Select an account</option>
                    </select>
                </div>
                <div class="checkbox" id="AccountStatusLabel">
                    <label><input type="checkbox" id="AccountStatus" disabled>Disable
account</label><br>
                </div>
                <div class="form-group">
                    <label for="ViewAllAccountsPets">Account's Pets</label><br>
                    <select class="form-control" id="ViewAllAccountsPets" disabled>
                        <option value="0">Select a pet</option>
                    </select>
                </div>
                <div class="checkbox" id="PetStatusLabel">
                    <label><input type="checkbox" id="PetStatus" disabled>Disable pet</label><br>
                </div>
                <div class="checkbox" id="DocumentLabel">
                    <p id="Document">Pet document</p>
                </div>
                <div class="form-group">
                    <label for="ViewPetName" id="ViewNameLabel">Pet's name</label><br>
                    <input type="text" id="ViewPetName" placeholder="Pet's name" disabled
autocomplete="off">
                </div>
                <div class="form-group">
                    <label for="ViewBreed" id="ViewBreedLabel">Pet's breed</label><br>
                    <select class="form-control" id="ViewBreed" disabled>
                    </select>
                </div>
                <div class="btn-group">
                    <label for="ViewGender" id="ViewGenderLabel">Pet's gender</label><br>
                    <select class="form-control" id="ViewGender" disabled>
                    </select>
                </div>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-default" data-dismiss="modal" data-
target="#AccountModal" data-toggle="modal">Back</button>
                <button type="button" class="btn btn-warning" data-dismiss="modal"
id="UpdateAccountButton" disabled>Update</button>
                <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
            </div>
        </div>
    </div>
</div>

<div class="modal fade" id="ViewErrorModal" role="dialog">
    <div class="modal-dialog modal-lg">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal">&times;</button>
                <h4 class="modal-title">Errors</h4>
            </div>
            <div class="modal-body">
                <table class="table table-bordered" id="ViewErrors">
                    <thead>
                    <tr>
                        <th>Account</th>
                        <th>Action</th>
                        <th>Error Message</th>
                        <th>Date</th>
                    </tr>
                    </thead>
                </table>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-default" data-dismiss="modal" data-
target="#AccountModal" data-toggle="modal">Back</button>
                <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
            </div>
        </div>
    </div>
</div>
<div class="btn-group-vertical" data-toggle="buttons" id="HiddenValues">
    <input type="hidden" id="HiddenValue1">
    <input type="hidden" id="HiddenValue2">
    <input type="hidden" id="HiddenValue3">
    <input type="hidden" id="HiddenValue4">
    <input type="hidden" id="HiddenValue5">
    <input type="hidden" id="HiddenValue6">
    <input type="hidden" id="HiddenValue7">
    <input type="hidden" id="HiddenValue8">
    <input type="hidden" id="HiddenValue9">
</div>
</body>
</html>
```

6.2 upload.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" type="text/css" href="css/main.css">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" integrity="sha384-
1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGmkzs7" crossorigin="anonymous">
```

```
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"
integrity="sha384-0mSbJDEHialfmuBBQP6A4Qrprq5OVfW37PRR3j5ELqxss1yVqOtnepnHVP9aJ7xS"
crossorigin="anonymous"></script>
    <title>Pet Sign In</title>
</head>
<body>
<form action="upload.php" method="post" enctype="multipart/form-data">
    <div class="form-group">
        <label for="Email">Email address</label><br>
        <input type="text" id="Email" name="Email" placeholder="Email address" required>
    </div>
    <div class="form-group">
        <label for="Name">Pet's name</label><br>
        <input type="text" id="Name" name="Name"placeholder="Pet's name" required>
    </div>
    Select your pet's documentation to upload (PDF only) 500kb file size limit:
    <input type="file" name="fileToUpload" id="fileToUpload">
    <br>
    <input type="submit" value="Upload Document" name="submit">
</form>
</body>
</html>
```

6.3 main.css

```
body {
    background-color: #f2f2f3;
}

.modal-dialog, .modal-content {
    /* 80% of window height */
    height: 80%;
}

.modal-body {
    /* 100% = dialog height, 120px = header + footer */
    max-height: calc(100% - 120px);
    overflow-y: auto;
    -webkit-overflow-scrolling: touch ;
}

#Visitor{
    width: 50%;
    margin: 0 auto;
}

#Account, #SignOut, #ViewActivitiesButton, #ViewErrorsButton, #Visitor, #AddNewPetButton,
#ViewAccountsButton, #HiddenValues, #ChangePasswordButton, #ViewBreedsButton{
    display: none;
    visibility: hidden;
}

.kv-avatar .file-preview-frame,.kv-avatar .file-preview-frame:hover {
    margin: 0;
    padding: 0;
    border: none;
    box-shadow: none;
    text-align: center;
}
.kv-avatar .file-input {
    display: table-cell;
    max-width: 220px;
}
```

6.4 main.js

```
$(document).ready(function() {
    //console.log("ready!");
    $.ajaxSetup({
        url: 'operations.php',
        type: 'post',
        cache: 'false',
        async: false,
        success: function(AjaxData) {
            //console.log('Ajax passed');
        },
        error: function(xhr, status, error) {
            //console.log("Error: " + xhr.status);
        }
    });

    Start();

    //SignInButton
    $("#SignInButton").click(function() {
        var Email = document.getElementById("Email1").value;
        var Password = document.getElementById("Password1").value;
        CheckRequiredField(Email);
        CheckRequiredField(Password);
        ValidateEmailDomain(Email);
        var Action = "SignIn";
        PrepareAjax(Action,Email,Password);
    });

    //ViewActivitiesModal
    $("#ViewActivitiesButton").click(function() {
        var Action = "FetchActivities";
        PrepareAjax(Action);
    });

    //ViewErrorModal
    $("#ViewErrorsButton").click(function() {
        var Action = "FetchErrors";
        PrepareAjax(Action);
    });

    //RegisterModal
    $("#Register").click(function() {
        document.getElementById("Email2").value = "";
        document.getElementById("Password2").value = "";
        document.getElementById("accounttncno").checked = true;
        document.getElementById("RegisterButton").disabled = true;
    });

    //UpdatePasswordButton
    $('#ChangePasswordButton').click(function() {
        document.getElementById("OldPassword").value = "";
        document.getElementById("OldPassword1").value = "";
        document.getElementById("NewPassword").value = "";
        document.getElementById("UpdatePasswordButton").disabled = true;
        var Action = "FetchUserEmail";
        PrepareAjax(Action);
    });

    //UpdatePasswordButton
    $('#OldPassword').change(function() {
        var Old = document.getElementById("OldPassword").value;
        var Old1 = document.getElementById("OldPassword1").value;
        var New = document.getElementById("NewPassword").value;
        if (Old != "" && Old1 != "" && New != "") {
            document.getElementById("UpdatePasswordButton").disabled = false;
        }else{
            document.getElementById("UpdatePasswordButton").disabled = true;
        }
    });

    //UpdatePasswordButton
    $('#OldPassword1').change(function() {
        var Old = document.getElementById("OldPassword").value;
        var Old1 = document.getElementById("OldPassword1").value;
        var New = document.getElementById("NewPassword").value;
        if (Old != "" && Old1 != "" && New != "") {
            document.getElementById("UpdatePasswordButton").disabled = false;
        }else{
            document.getElementById("UpdatePasswordButton").disabled = true;
        }
    });

    //UpdatePasswordButton
    $('#NewPassword').change(function() {
        var Old = document.getElementById("OldPassword").value;
        var Old1 = document.getElementById("OldPassword1").value;
        var New = document.getElementById("NewPassword").value;
        if (Old != "" && Old1 != "" && New != "") {
            document.getElementById("UpdatePasswordButton").disabled = false;
        }else{
            document.getElementById("UpdatePasswordButton").disabled = true;
        }
    });

    //UpdatePasswordButton
    $("#UpdatePasswordButton").click(function() {
        var OldPassword = document.getElementById("OldPassword").value;
        var OldPassword1 = document.getElementById("OldPassword1").value;
        var NewPassword = document.getElementById("NewPassword").value;
        var Email = document.getElementById("HiddenValue1").value;
        CheckRequiredField(OldPassword);
        CheckRequiredField(OldPassword1);
        CheckRequiredField(NewPassword);
        if(OldPassword == OldPassword1 && OldPassword1 != NewPassword){
            ValidatePassword(Email,NewPassword);
            var Action = "UpdatePassword";
            PrepareAjax(Action,OldPassword,NewPassword);
        }else{
            alert("Error 27: Your old passwords don't match and/or they match with your new
password.")
        }
    });

    //ResetPassword
    $('#ResetAccount').click(function() {
        document.getElementById("Email3").value = "";
        document.getElementById("ResetPassword").disabled = true;
    });

    //ResetPassword
    $('#Email3').change(function() {
        var Email = document.getElementById("Email3").value;
        if (Email != "") {
            document.getElementById("ResetPassword").disabled = false;
        }else{
            document.getElementById("ResetPassword").disabled = true;
        }
    });

    //RegisterButton
    $('input[type=radio][name=accounttnc]').change(function() {
        if (this.value == "yes") {
            document.getElementById("RegisterButton").disabled = false;
        }else if (this.value == "no") {
            document.getElementById("RegisterButton").disabled = true;
        }
    });

    //ViewSignInPet
    $("#ViewSignInPet :input").change(function() {
        var Action = "SignInPet";
        var PetName = document.getElementById("SignInPet").value;
        PrepareAjax(Action,PetName);
    });

    //AddNewPetModal
    $('input[type=radio][name=pettnc]').change(function() {
```

```
    if (this.value == "yes") {
        document.getElementById("AddPetButton").disabled = false;
    }else if (this.value == "no") {
        document.getElementById("AddPetButton").disabled = true;
    }
});

//ResetAccountModal
$("#ResetPassword").click(function() {
    var Email = document.getElementById("Email3").value;
    CheckRequiredField(Email);
    var Action = "ResetPassword";
    PrepareAjax(Action,Email);
});

//ViewBreedsModal
$("#ViewBreeds1").change(function() {
    if (this.value == 0) {
        document.getElementById("ViewBreedName").disabled = true;
        document.getElementById("UpdateBreedButton").disabled = true;
        document.getElementById("ViewBreedName").value = "";
        document.getElementById("ViewBreedName").style.backgroundColor = "transparent";
    }else{
        document.getElementById("ViewBreedName").disabled = false;
        document.getElementById("UpdateBreedButton").disabled = true;
        document.getElementById("ViewBreedName").value = "";
        var SelectedBreed = document.getElementById("ViewBreeds1");
        var SelectedBreedName = SelectedBreed.options[SelectedBreed.selectedIndex].text;
        document.getElementById("ViewBreedName").style.backgroundColor = "transparent";
        document.getElementById("HiddenValue1").value =  SelectedBreed.value;
        document.getElementById("HiddenValue2").value = SelectedBreedName;
        document.getElementById("ViewBreedName").value = SelectedBreedName;
    }
});

//ViewAccountsModal
$("#ViewBreed").change(function() {
    var DefaultValue = document.getElementById("HiddenValue3").value;
    var NewValue = document.getElementById("ViewBreed").value;
    if(DefaultValue == NewValue){
        document.getElementById("ViewBreed").style.backgroundColor = "transparent";
        document.getElementById("UpdateAccountButton").disabled = true;
    }else{
        document.getElementById("ViewBreed").style.backgroundColor = "lightgreen";
        document.getElementById("UpdateAccountButton").disabled = false;
        var BreedID = document.getElementById("ViewBreed").value;
        document.getElementById("HiddenValue6").value = BreedID;
    }
});

//ViewAccountsModal
$("#ViewGender").change(function() {
    var DefaultValue = document.getElementById("HiddenValue4").value;
    var NewValue = document.getElementById("ViewGender").value;
    if (DefaultValue == NewValue) {
        document.getElementById("ViewGender").style.backgroundColor = "transparent";
        document.getElementById("UpdateAccountButton").disabled = true;
    }else{
        document.getElementById("ViewGender").style.backgroundColor = "lightgreen";
        document.getElementById("UpdateAccountButton").disabled = false;
    }
});

//ViewAccountsModal
$("#ViewPetName").change(function() {
    var DefaultValue = document.getElementById("HiddenValue2").value;
    var NewValue = document.getElementById("ViewPetName").value;
    if(DefaultValue == NewValue){
        document.getElementById("ViewPetName").style.backgroundColor = "transparent";
        document.getElementById("UpdateAccountButton").disabled = true;
    }else{
        document.getElementById("ViewPetName").style.backgroundColor = "lightgreen";
        document.getElementById("UpdateAccountButton").disabled = false;
    }
});

//ViewBreedsModal
$("#ViewBreedName").change(function() {
    var DefaultValue = document.getElementById("HiddenValue2").value;
    var NewValue = document.getElementById("ViewBreedName").value;
    if(DefaultValue == NewValue){
        document.getElementById("ViewBreedName").style.backgroundColor = "transparent";
        document.getElementById("UpdateBreedButton").disabled = true;
    }else{
        document.getElementById("ViewBreedName").style.backgroundColor = "lightgreen";
        document.getElementById("UpdateBreedButton").disabled = false;
    }
});

//ViewBreedsModal
$("#AddNewBreed").change(function() {
    var NewValue = document.getElementById("AddNewBreed").value;
    if(NewValue == ""){
        document.getElementById("AddNewBreed").style.backgroundColor = "transparent";
        document.getElementById("AddBreedButton").disabled = true;
    }else{
        document.getElementById("AddNewBreed").style.backgroundColor = "lightgreen";
        document.getElementById("AddBreedButton").disabled = false;
    }
});

//SignOut
$("#SignOut").click(function() {
    var Action = "SignOut";
    PrepareAjax(Action);
});

//RegisterModal
$("#RegisterButton").click(function() {
    var Email = document.getElementById("Email2").value;
    var Password = document.getElementById("Password2").value;
    CheckRequiredField(Email);
    CheckRequiredField(Password);
    ValidateEmailDomain(Email);
    ValidatePassword(Email,Password);
    var Action = "AddAccount";
    PrepareAjax(Action,Email,Password);
});

//AddNewPetModal
$("#AddNewPetButton").click(function() {
    document.getElementById("PetName").value = "";
    document.getElementById("ViewBreeds").options.length = 1;
    document.getElementById("ViewGenders").selectedIndex  = 0;
    document.getElementById("pettncno").checked = true;
    document.getElementById("AddPetButton").disabled = true;
    var Action = "FetchBreeds";
    var ResponseData = Fetch(Action);
    ViewBreeds(ResponseData);
});

////AddNewPetModal
$("#AddPetButton").click(function() {
    var PetName = document.getElementById("PetName").value;
    var BreedID = document.getElementById("ViewBreeds").value;
    var Gender = document.getElementById("ViewGenders").value;
    CheckRequiredField(PetName);
    CheckRequiredField(BreedID);
    CheckRequiredField(Gender);
    var Action = "FetchPetNameCount";
    var ResponseData = Fetch(Action,PetName);
    if(ResponseData['Count'] == 0){
            var Action = "AddPet";
            PrepareAjax(Action,PetName,BreedID,Gender);
        }else{
            alert("Error 51: You already have a pet name " + PetName + ".  Please pick a 
different name.")
        }
    });

    //ViewBreedsModal
    $("#ViewBreedsButton").click(function() {
        document.getElementById("ViewBreedName").disabled = true;
        document.getElementById("ViewBreedName").disabled = true;
        document.getElementById("UpdateBreedButton").disabled = true;
        document.getElementById("AddBreedButton").disabled = true;
        document.getElementById("ViewBreeds1").options.length = 1;
        document.getElementById("ViewBreedName").value = "";
        document.getElementById("AddNewBreed").value = "";
        document.getElementById("ViewBreedName").style.backgroundColor = "transparent";
        document.getElementById("AddNewBreed").style.backgroundColor = "transparent";
        var Action = "FetchBreeds";
        var ResponseData = Fetch(Action);
        ViewBreeds1(ResponseData);
    });

    //ViewAccountsModal
    $("#ViewAccountsButton").click(function() {
        document.getElementById("AccountStatus").disabled = true;
        document.getElementById("ViewAllAccountsPets").disabled = true;
        document.getElementById("PetStatus").disabled = true;
        document.getElementById("ViewPetName").disabled = true;
        document.getElementById("ViewBreed").disabled = true;
        document.getElementById("ViewGender").disabled = true;
        document.getElementById("UpdateAccountButton").disabled = true;
        document.getElementById("AccountStatus").checked = false;
        document.getElementById("PetStatus").checked = false;
        document.getElementById("AccountStatusLabel").style.backgroundColor = "transparent";
        document.getElementById("PetStatusLabel").style.backgroundColor = "transparent";
        document.getElementById("ViewPetName").style.backgroundColor = "transparent";
        document.getElementById("ViewBreed").style.backgroundColor = "transparent";
        document.getElementById("ViewGender").style.backgroundColor = "transparent";
        document.getElementById("ViewAllAccounts").options.length = 1;
        document.getElementById("ViewAllAccountsPets").options.length = 1;
        document.getElementById("Document").value = "Pet Document";
        document.getElementById("ViewBreed").innerHTML = "";
        document.getElementById("ViewGender").innerHTML = "";
        document.getElementById("ViewPetName").value = "";
        var Action = "FetchUsers";
        PrepareAjax(Action);
    });

    //ViewAccountsModal
    $("#ViewAllAccounts").change(function() {
        if (this.value == 0) {
            document.getElementById("AccountStatus").disabled = true;
            document.getElementById("ViewAllAccountsPets").disabled = true;
            document.getElementById("PetStatus").disabled = true;
            document.getElementById("ViewPetName").disabled = true;
            document.getElementById("ViewBreed").disabled = true;
            document.getElementById("ViewGender").disabled = true;
            document.getElementById("UpdateAccountButton").disabled = true;
            document.getElementById("AccountStatus").checked = false;
            document.getElementById("PetStatus").checked = false;
            document.getElementById("AccountStatusLabel").style.backgroundColor = "transparent";
            document.getElementById("PetStatusLabel").style.backgroundColor = "transparent";
            document.getElementById("ViewPetName").style.backgroundColor = "transparent";
            document.getElementById("ViewBreed").style.backgroundColor = "transparent";
            document.getElementById("ViewGender").style.backgroundColor = "transparent";
            document.getElementById("ViewAllAccountsPets").options.length = 1;
            document.getElementById("Document").innerHTML = "Pet Document";
            document.getElementById("ViewBreed").innerHTML = "";
            document.getElementById("ViewGender").innerHTML = "";
            document.getElementById("ViewPetName").value = "";
        }else{
            document.getElementById("AccountStatus").checked = false;
            document.getElementById("PetStatus").checked = false;
            document.getElementById("AccountStatusLabel").style.backgroundColor = "transparent";
            document.getElementById("PetStatusLabel").style.backgroundColor = "transparent";
            document.getElementById("ViewPetName").style.backgroundColor = "transparent";
            document.getElementById("ViewBreed").style.backgroundColor = "transparent";
            document.getElementById("ViewGender").style.backgroundColor = "transparent";
            document.getElementById("ViewAllAccountsPets").options.length = 1;
            document.getElementById("Document").innerHTML = "Pet Document";
            document.getElementById("ViewBreed").innerHTML = "";
            document.getElementById("ViewGender").innerHTML = "";
            document.getElementById("ViewPetName").value = "";
            document.getElementById("HiddenValue1").value = this.value;
            var Action = "FetchUserStatus";
            var Email = document.getElementById("ViewAllAccounts").value;
            PrepareAjax(Action,Email);
            Action = "FetchUserPets";
            PrepareAjax(Action, Email);
        }
    });

    //ViewAccountsModal
    $("#ViewAllAccountsPets").change(function() {
        if (this.value == 0) {
            document.getElementById("PetStatus").disabled = true;
            document.getElementById("ViewPetName").disabled = true;
            document.getElementById("ViewBreed").disabled = true;
            document.getElementById("ViewGender").disabled = true;
            document.getElementById("UpdateAccountButton").disabled = true;
            document.getElementById("AccountStatus").checked = false;
            document.getElementById("PetStatus").checked = false;
            document.getElementById("PetStatusLabel").style.backgroundColor = "transparent";
            document.getElementById("ViewPetName").style.backgroundColor = "transparent";
            document.getElementById("ViewBreed").style.backgroundColor = "transparent";
            document.getElementById("ViewGender").style.backgroundColor = "transparent";
            document.getElementById("Document").innerHTML = "Pet Document";
            document.getElementById("ViewBreed").innerHTML = "";
            document.getElementById("ViewGender").innerHTML = "";
            document.getElementById("ViewPetName").value = "";
        }else{
            document.getElementById("PetStatus").checked = false;
            document.getElementById("PetStatusLabel").style.backgroundColor = "transparent";
            document.getElementById("ViewPetName").style.backgroundColor = "transparent";
            document.getElementById("ViewBreed").style.backgroundColor = "transparent";
            document.getElementById("ViewGender").style.backgroundColor = "transparent";
            document.getElementById("Document").innerHTML = "Pet Document";
            document.getElementById("ViewBreed").innerHTML = "";
            document.getElementById("ViewGender").innerHTML = "";
            document.getElementById("ViewPetName").value = "";
            var PetID = this.value;
            document.getElementById("HiddenValue7").value = PetID;
            var Action = "FetchPetStatus";
            PrepareAjax(Action,PetID);
            var Action = "FetchPet";
            var PetData = Fetch(Action,PetID);
            var Action = "FetchBreeds";
            var Breeds = Fetch(Action);
            ViewPetData(PetData,Breeds);
        }
    });

    //ViewAccountsModal
    $("#AccountStatus").change(function() {
        var DefaultValue = document.getElementById("AccountStatus").value;
        if(this.checked){
            document.getElementById("HiddenValue9").value = 1;
        }else{
            document.getElementById("HiddenValue9").value = 0;
        }
        if(DefaultValue == 1 && this.checked || DefaultValue == 0 && !this.checked){
            document.getElementById("AccountStatusLabel").style.backgroundColor = "transparent";
            document.getElementById("UpdateAccountButton").disabled = true;
        }else{
            document.getElementById("AccountStatusLabel").style.backgroundColor = "lightgreen";
            document.getElementById("UpdateAccountButton").disabled = false;
        }
    });

    //ViewAccountsModal
    $("#PetStatus").change(function() {
        var DefaultValue = document.getElementById("PetStatus").value;
        if(this.checked){
            document.getElementById("HiddenValue8").value = 1;
        }else{
            document.getElementById("HiddenValue8").value = 0;
        }
        if(DefaultValue == 1 && this.checked || DefaultValue == 0 && !this.checked){
            document.getElementById("PetStatusLabel").style.backgroundColor = "transparent";
            document.getElementById("UpdateAccountButton").disabled = true;
        }else{
            document.getElementById("PetStatusLabel").style.backgroundColor = "lightgreen";
            document.getElementById("UpdateAccountButton").disabled = false;
        }
    });

    //AddNewPetModal
    $("#ViewPetName").change(function() {
        var DefaultValue = document.getElementById("HiddenValue2").value;
        var NewValue = document.getElementById("ViewPetName").value;
        if(DefaultValue == NewValue){
            document.getElementById("ViewPetName").style.backgroundColor = "transparent";
            document.getElementById("UpdateAccountButton").disabled = true;
        }else{
            document.getElementById("ViewPetName").style.backgroundColor = "lightgreen";
            document.getElementById("UpdateAccountButton").disabled = false;
        }
    });

    //AddNewPetModal
    $("#ViewGender").change(function() {
        var DefaultValue = document.getElementById("HiddenValue4").value;
        var NewValue = document.getElementById("ViewPetName").value;
        if(DefaultValue == NewValue){
            document.getElementById("ViewGender").style.backgroundColor = "transparent";
            document.getElementById("UpdateAccountButton").disabled = true;
        }else{
            document.getElementById("ViewGender").style.backgroundColor = "lightgreen";
            document.getElementById("UpdateAccountButton").disabled = false;
        }
    });

    //ViewAccountsModal
    $("#UpdateAccountButton").click(function() {
        var Email = document.getElementById("HiddenValue1").value;
        var OldAccountStatus = document.getElementById("AccountStatus").value;
        var OldPetStatus = document.getElementById("PetStatus").value;
        var OldName = document.getElementById("HiddenValue2").value;
        var OldBreedID = document.getElementById("HiddenValue3").value;
        var OldGender = document.getElementById("HiddenValue4").value;
        var PetID = document.getElementById("HiddenValue7").value;
        var AccountStatus = document.getElementById("HiddenValue9").value;
        var PetStatus = document.getElementById("HiddenValue8").value;
        var PetName = document.getElementById("ViewPetName").value;
        var BreedID = document.getElementById("HiddenValue6").value;
        var Gender = document.getElementById("ViewGender").value;
        if(OldAccountStatus != AccountStatus){
            var Action = "UpdateAccountStatus";
            PrepareAjax(Action,AccountStatus,Email);
        }
        if(OldPetStatus != PetStatus && PetStatus != ""){
            var Action = "UpdatePetStatus";
            PrepareAjax(Action,PetStatus,PetName,PetID,Email);
        }
        if(OldName != PetName && PetName != ""){
            var Action = "UpdatePetName";
            PrepareAjax(Action,PetName,OldName,Email);
        }
        if(OldBreedID != BreedID && BreedID != "" && (typeof document.getElementById("ViewBreed"
=== "undefined"))){
            var Action = "UpdatePetBreed";
            PrepareAjax(Action,BreedID,PetName,Email);
        }
        if(OldGender != Gender && Gender != ""){
            var Action = "UpdatePetGender";
            PrepareAjax(Action,Gender,PetName,Email);
        }
    });

    //ViewBreedsModal
    $("#UpdateBreedButton").click(function() {
        var OldBreedName = document.getElementById("HiddenValue2").value;
        var NewBreedName = document.getElementById("ViewBreedName").value;
        var BreedID = document.getElementById("ViewBreeds1").value;
        if(OldBreedName != NewBreedName){
            var Action = "UpdateBreed";
            PrepareAjax(Action,NewBreedName,OldBreedName,BreedID);
        }
    });

    //ViewBreedsModal
    $("#AddBreedButton").click(function() {
        var NewBreedName = document.getElementById("AddNewBreed").value;
        var Action = "AddBreed";
        PrepareAjax(Action,NewBreedName);
    });
});

function Fetch(Action,D1){
    var D1;
    try{
        var AjaxData = {
            Action: Action,
            D1: D1
        };
        var ResponseData = JSON.parse(OutgoingAjax(AjaxData));
        //console.log(ResponseData);
        return ResponseData;
    }catch(e){
        var ErrorMSG = e;
        var FailedAction = Action;
        AddError(FailedAction,ErrorMSG);
        alert('Error 3: Oops, something went wrong.  Contact an administrator with this error
message.');
    }
}

//ViewAccountsModal
function ViewAccountStatus (ResponseData){
    document.getElementById("AccountStatus").disabled = false;
    if(ResponseData.Disabled == 0){
        document.getElementById("AccountStatus").checked = false;
        document.getElementById("AccountStatus").value = 0;
        document.getElementById("HiddenValue9").value = 0;
    }else{
        document.getElementById("AccountStatus").checked = true;
        document.getElementById("AccountStatus").value = 1;
        document.getElementById("HiddenValue9").value = 1;
    }
}

//UpdateAccountButton
```

```
function ViewPetStatus (ResponseData){
    document.getElementById("PetStatus").disabled = false;
    if(ResponseData.Disabled == 0){
        document.getElementById("PetStatus").checked = false;
        document.getElementById("PetStatus").value = 0;
        document.getElementById("HiddenValue8").value = 0;
    }else{
        document.getElementById("PetStatus").checked = true;
        document.getElementById("PetStatus").value = 1;
        document.getElementById("HiddenValue8").value = 1;
    }
}

//UpdatePasswordButton
function ViewSessionEmail (ResponseData){
    document.getElementById("HiddenValue1").value = ResponseData;
}

//ViewAccountsModal
function ViewPetData(PetData,Breeds){
    document.getElementById("ViewPetName").disabled = false;
    document.getElementById("ViewBreed").disabled = false;
    document.getElementById("ViewGender").disabled = false;
    document.getElementById("HiddenValue2").value = PetData.Name;
    document.getElementById("HiddenValue3").value = PetData.BreedID;
    document.getElementById("HiddenValue4").value = PetData.Gender;
    var Document = PetData.Document;
    document.getElementById("ViewPetName").value  = PetData.Name;

    if(Document != "" && !Document){
        document.getElementById("Document").innerHTML = "Pet Document";
    }else{
        document.getElementById("Document").innerHTML = '<a
href="https://alibkaba.com/petsignin/uploads/' + Document + '">Pet document</a>';
    }

    var select = document.getElementById("ViewBreed");
    var i;
    var PetIndex;
    for (i = 0; i < Breeds.length; i++) {
        select.options[select.options.length] = new Option(Breeds[i].Name, Breeds[i].BreedID);
        if(Breeds[i].BreedID == PetData.BreedID){
            PetIndex = i;
        }
    }
    document.getElementById("ViewBreed").selectedIndex = PetIndex;

    var select = document.getElementById("ViewGender");
    if(PetData.Gender == "Boy"){
        select.options[select.options.length] = new Option("Boy", "Boy");
        select.options[select.options.length] = new Option("Girl", "Girl");
        document.getElementById("ViewGender").selectedIndex  = 0;
    }else{
        select.options[select.options.length] = new Option("Boy", "Boy");
        select.options[select.options.length] = new Option("Girl", "Girl");
        document.getElementById("ViewGender").selectedIndex  = 1;
    }
}

//AddNewPetModal
function ViewBreeds(ResponseData){
    var select = document.getElementById("ViewBreeds");
    var i;
    for (i = 0; i < ResponseData.length; i++) {
        select.options[select.options.length] = new Option(ResponseData[i].Name,
ResponseData[i].BreedID);
    }
}

//ViewBreedsModal
function ViewBreeds1(ResponseData){
    var select = document.getElementById("ViewBreeds1");
    var i;
    for (i = 0; i < ResponseData.length; i++) {
        select.options[select.options.length] = new Option(ResponseData[i].Name,
ResponseData[i].BreedID);
    }
}

//ViewAccountsModal
function ViewUserPets(ResponseData){
    document.getElementById("ViewAllAccountsPets").disabled = false;
    var select = document.getElementById("ViewAllAccountsPets");
    var i;
    for (i = 0; i < ResponseData.length; i++) {
        select.options[select.options.length] = new Option(ResponseData[i].Name,
ResponseData[i].PetID);
    }
}

//ViewAccountsModal
function ViewAllAccounts(ResponseData){
    var select = document.getElementById("ViewAllAccounts");
    var i;
    for (i = 0; i < ResponseData.length; i++) {
        select.options[select.options.length] = new Option(ResponseData[i].Email,
ResponseData[i].Email);
    }
}

//ViewSignInPet
function ViewSignInPet(ResponseData){
    var ViewSignInPet = "";
    for (var i = 0; i < ResponseData.length; i++) {
        if (ResponseData[i].Disabled == 0){
            if (ResponseData[i].DiffDate == 0){
                ViewSignInPet += '<label class="btn btn-primary" disabled>';
            }else{
                ViewSignInPet += '<label class="btn btn-primary">';
            }
            ViewSignInPet += '<input type="radio" name="options" id="SignInPet"
autocomplete="off" value="' + ResponseData[i].Name + '">' + ResponseData[i].Name + '</button>';

            ViewSignInPet += '</label>';
        }else{
            ViewSignInPet += '<label class="btn btn-danger" disabled><input type="radio"
name="options" id="SignInPet" autocomplete="off" value="' + ResponseData[i].Name + '">' +
ResponseData[i].Name + '</button></label>';
        }
    }
    document.getElementById("ViewSignInPet").innerHTML = ViewSignInPet;
}

//Multiple use
function CheckRequiredField(Field){
    if(Field == null || Field == ""){
        alert('Error 26: Please fill all of the fields.');
        throw e = "Error 26: Please fill all of the fields.";
    }
}

//Single use
function Start(){
    UnitTest();
    ValidateSession();
}

function Visitor(){
    document.getElementById("Visitor").style.display="block";
    document.getElementById("Visitor").style.visibility="visible";
}
```

```
function ViewUser(){
    FetchSignInPet();
    document.getElementById("SignOut").style.display="block";
    document.getElementById("SignOut").style.visibility="visible";
    document.getElementById("Account").style.display="block";
    document.getElementById("Account").style.visibility="visible";
    document.getElementById("ViewActivitiesButton").style.display="block";
    document.getElementById("ViewActivitiesButton").style.visibility="visible";
    document.getElementById("AddNewPetButton").style.display="block";
    document.getElementById("AddNewPetButton").style.visibility="visible";
    document.getElementById("ChangePasswordButton").style.display="block";
    document.getElementById("ChangePasswordButton").style.visibility="visible";
}

function ViewAdmin(){
    document.getElementById("SignOut").style.display="block";
    document.getElementById("SignOut").style.visibility="visible";
    document.getElementById("Account").style.display="block";
    document.getElementById("Account").style.visibility="visible";
    document.getElementById("ViewActivitiesButton").style.display="block";
    document.getElementById("ViewActivitiesButton").style.visibility="visible";
    document.getElementById("ViewErrorsButton").style.display="block";
    document.getElementById("ViewErrorsButton").style.visibility="visible";
    document.getElementById("AddPetButton").style.display="block";
    document.getElementById("AddPetButton").style.visibility="visible";
    document.getElementById("ViewAccountsButton").style.display="block";
    document.getElementById("ViewAccountsButton").style.visibility="visible";
    document.getElementById("ViewBreedsButton").style.display="block";
    document.getElementById("ViewBreedsButton").style.visibility="visible";
    document.getElementById("ChangePasswordButton").style.display="block";
    document.getElementById("ChangePasswordButton").style.visibility="visible";
}

function ValidateSession(){
    var Action = "ValidateSession";
    PrepareAjax(Action);
}

function FetchSignInPet(){
    var Action = "FetchSignInPet";
    PrepareAjax(Action);
}

function ViewActivities(ResponseData){
    var ViewActivities = '<thead><tr><th>Activities</th><th>Date</th></tr></thead><tbody>';
    for (var i = 0; i < ResponseData.length; i++) {
        ViewActivities += '<tr><td>' + ResponseData[i].ActivityMSG + '</td><td>' +
ResponseData[i].LogDate + '</td></tr>';
    }
    ViewActivities += '</tbody>';
    document.getElementById("ViewActivities").innerHTML = ViewActivities;
}

function ViewErrors(ResponseData){
    var ViewErrors = '<thead><tr><th>Account</th><th>Action</th><th>Error
Message</th><th>Date</th></tr></thead><tbody>';
    for (var i = 0; i < ResponseData.length; i++) {
        ViewErrors += '<tr><td>' + ResponseData[i].Email +  '<td>' + ResponseData[i].Action +
'<td>' + ResponseData[i].ErrorMSG + '</td><td>' + ResponseData[i].LogDate + '</td></tr>';
    }
    ViewErrors += '</tbody>';
    document.getElementById("ViewErrors").innerHTML = ViewErrors;
}

function UnitTest() {
    var Action = "UnitTest";
    var AjaxData = {
        Action: Action
    };
    var AjaxData = OutgoingAjax(AjaxData);
}
```

```
function ValidateEmailDomain(Email) {
    var re = /^\s*[\w\-\+_]+(\.[\w\-\+_]+)*\@[\w\-\+_]+\.[\w\-\+_]+(\.[\w\-\+_]+)*\s*$/;
    if (re.test(Email)) {
        if (Email.indexOf('@gmail.com', Email.length - '@gmail.com'.length) == -1) {
            alert('Error 10: Please enter a valid GMAIL e-mail address (your.name@gmail.com).');
            throw e = "Error 10: Please enter a valid GMAIL e-mail address
(your.name@gmail.com).";
        }
    } else {
        alert('Error 11: Not a valid e-mail address.');
        throw e = "Error 11: Not a valid e-mail address.";
    }
}

function ValidatePassword(Email,Password){
    if(Password.length < 8 || Password.length > 15) {
        alert("Error 20: Password must be between 8 and 15 characters long.");
        throw e = "Error 20: Password must be between 8 and 15 characters long.";
    }
    if(Password == Email) {
        alert("Error 21: Password must be different from your email.");
        throw e = "Error 21: Password must be different from your email.";
    }
    re = /[0-9]/;
    if(!re.test(Password)) {
        alert("Error 22: Password must contain at least one number (0-9).");
        throw e = "Error 22: Password must contain at least one number (0-9).";
    }
    re = /[a-z]/;
    if(!re.test(Password)) {
        alert("Error 23: Password must contain at least one lowercase letter (a-z).");
        throw e = "Error 23: Password must contain at least one lowercase letter (a-z).";
    }
    re = /[A-Z]/;
    if(!re.test(Password)) {
        alert("Error 24: Password must contain at least one uppercase letter (A-Z).");
        throw e = "Error 24: Password must contain at least one uppercase letter (A-Z).";
    }
}

function PrepareAjax(Action,D1,D2,D3,D4,D5,D6,D7,D8,D9){
    var D1;
    var D2;
    var D3;
    var D4;
    var D5;
    var D6;
    var D7;
    var D8;
    var D9;
    try{
        var AjaxData = {
            Action: Action,
            D1: D1,
            D2: D2,
            D3: D3,
            D4: D4,
            D5: D5,
            D6: D6,
            D7: D7,
            D8: D8,
            D9: D9
        };
        var ResponseData = JSON.parse(OutgoingAjax(AjaxData));
        JSOperation(Action, ResponseData);
    }catch(e){
        var ErrorMSG = e;
        var FailedAction = Action;
        AddError(FailedAction,ErrorMSG);
        alert('Error 1: Oops, something went wrong.  Contact an administrator with this error
message.');

    }
}

function AddError(FailedAction, ErrorMSG) {
    var Action = "AddError";
    var AjaxData = {
        Action: Action,
        FailedAction: FailedAction,
        ErrorMSG: ErrorMSG
    }
    OutgoingAjax(AjaxData);
}

function JSOperation(Action, ResponseData){
    //console.log(ResponseData);
    switch(ResponseData) {
        case "locked":
            alert("Error 89: This account has been locked.  Reset your account or contact the
administrator.");
            break;
        case "notlocked":
            alert("Error 87: Your account will be locked out soon.");
            break;
        case "invalid":
            alert("Error 25: Invalid email and/or password.  If you forgot your password, reset
it.");
            break;
        case "notactive":
            alert("Error 15: Your account is not activate.  Wait or contact an Admin.");
            break;
        case "none":
            alert("Error 19: This account doesn't exist.  Please click on \"Register for a new
account\".");
            break;
        case "pupdated":
            alert("Password was updated.");
            window.location = "/petsignin/";
            break;
        case "xupdated":
            alert("Error 28: Your old passwords don't match and/or they match with your new
password.");
            window.location = "/petsignin/";
            break;
        case "refresh":
            window.location = "/petsignin/";
            break;
        case 2:
            ViewAdmin();
            break;
        case 1:
            ViewUser();
            break;
        case 0:
            Visitor();
            break;
        case "expired":
            alert("Error 99: Your session expired, please sign in again.");
            window.location = "/petsignin/";
            break;
        case "breedexist":
            alert("Error 50: This breed already exist.");
            break;
        default:
            JSOperation2(Action,ResponseData);
    }
}

function JSOperation2(Action,ResponseData){
    //console.log(ResponseData);
    switch(Action) {
        case "FetchActivities":
            ViewActivities(ResponseData);
            break;
        case "FetchErrors":
            ViewErrors(ResponseData);
            break;
        case "FetchSignInPet":
            ViewSignInPet(ResponseData);
            break;
        case "FetchUserPets":
            ViewUserPets(ResponseData);
            break;
        case "FetchUsers":
            ViewAllAccounts(ResponseData);
            break;
        case "FetchUserStatus":
            ViewAccountStatus(ResponseData);
            break;
        case "FetchPetStatus":
            ViewPetStatus(ResponseData);
            break;
        case "FetchUserEmail":
            ViewSessionEmail(ResponseData);
            break;
        default:
            alert("Error 2: Oops, something went wrong.  Contact an administrator with this error
message.!");
    }
}

function OutgoingAjax(AjaxData) {
    var IncomingAjaxData = $.ajax({
        data: AjaxData
    }).responseText;
    return IncomingAjaxData;
}
```

6.5 operation.php

```
<?php
require_once('db.php');
error_reporting(E_ALL);
ini_set('display_errors', 1);

ValidateAjaxRequest();

function ValidateAjaxRequest() {
    if (isset($_SERVER['HTTP_X_REQUESTED_WITH']) && strtolower($_SERVER['HTTP_X_REQUESTED_WITH'])
== 'xmlhttprequest'){
        ValidateAction();
    }
}

function ValidateAction(){
    if (isset($_POST["Action"]) && !empty($_POST["Action"])) {
        $Action = $_POST["Action"];
        PHPOperation($Action);
    }
}

function PHPOperation($Action){
    switch($Action) {
        case "UnitTest": UnitTest($Action);
            break;
        case "AddAccount": AddAccount($Action);
            break;
        case "SignIn": SignIn($Action);
            break;
        case "SignInPet": SignInPet($Action);
            break;
        case "SignOut": SignOut($Action);
```

```
            break;
        case "FetchErrors": FetchErrors($Action);
            break;
        case "AddError": AddError($Action);
            break;
        case "FetchActivities": FetchActivities($Action);
            break;
        case "FetchSignInPet": FetchSignInPet($Action);
            break;
        case "FetchBreeds": FetchBreeds($Action);
            break;
        case "ValidateSession": ValidateSession($Action);
            break;
        case "ResetPassword": ResetPassword($Action);
            break;
        case "UpdatePassword": UpdatePassword($Action);
            break;
        case "AddPet": AddPet($Action);
            break;
        case "AddBreed": AddBreed($Action);
            break;
        case "FetchUsers": FetchUsers($Action);
            break;
        case "FetchUserPets": FetchUserPets($Action);
            break;
        case "FetchUserStatus": FetchUserStatus($Action);
            break;
        case "FetchPetStatus": FetchPetStatus($Action);
            break;
        case "FetchPet": FetchPet($Action);
            break;
        case "UpdateAccountStatus": UpdateAccountStatus($Action);
            break;
        case "UpdatePetStatus": UpdatePetStatus($Action);
            break;
        case "UpdatePetName": UpdatePetName($Action);
            break;
        case "UpdatePetBreed": UpdatePetBreed($Action);
            break;
        case "UpdatePetGender": UpdatePetGender($Action);
            break;
        case "UpdateBreed": UpdateBreed($Action);
            break;
        case "FetchPetNameCount": FetchPetNameCount($Action);
            break;
        case "FetchUserEmail": FetchUserEmail($Action);
            break;
    }
}

function Execute($Action,$Statement){
    try {
        $Statement->execute();
    } catch (PDOException $e) {
        $ErrorMSG = 'Error 0: ' . $e->getMessage() . "\n";
        $PHP = 1;
        AddError($Action,$ErrorMSG,$PHP);
    }
}

function AdminRole($Action,$Email){
    $Role = FetchAccountRole($Action,$Email);
    if($Role != 2){
        echo json_encode("expired");
        exit;
    }
}

function UserRole($Action,$Email){
    $Role = FetchAccountRole($Action,$Email);
    if($Role != 1){
        echo json_encode("expired");
        exit;
    }
}

function AddActivity($Action,$Email,$ActivityMSG){
    global $PDOconn;
    $Query = 'CALL AddActivity (?, ?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $Email, PDO::PARAM_STR, 45);
    $Statement->bindParam(2, $ActivityMSG, PDO::PARAM_STR, 255);
    Execute($Action,$Statement);
}

function HashIt($Password){
    $HashedPassword = password_hash($Password, PASSWORD_DEFAULT);
    return $HashedPassword;
}

function FetchUser($Action,$Email){
    global $PDOconn;
    $Query = 'CALL FetchUser (?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $Email, PDO::PARAM_STR, 45);
    Execute($Action,$Statement);
    $Response = $Statement->fetch(PDO::FETCH_ASSOC);
    return $Response;
}

function FetchAdmins($Action){
    global $PDOconn;
    $Query = 'CALL FetchAdmins';
    $Statement = $PDOconn->prepare($Query);
    Execute($Action,$Statement);
    $Response = $Statement->fetchAll();
    return $Response;
}

function ValidatePassword($Password,$HashedPassword){
    if (password_verify($Password, $HashedPassword)) {
        return 1;
    } else {
        return 0;
    }
}

function FetchAccountRole($Action,$Email){
    global $PDOconn;
    $Query = 'CALL FetchAccountRole (?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $Email, PDO::PARAM_STR, 45);
    Execute($Action,$Statement);
    $Response = $Statement->fetch(PDO::FETCH_ASSOC);
    if($Response['Disabled'] == 0 && $Response['Attempt'] == 0){
        if($Response['AdminCode'] == 2){
            $AccountRole = 2;//admin
        }else{
            $AccountRole = 1;//registered user
        }
    }else{
        session_unset();
        session_destroy();
        $AccountRole = 0;//user
    }
    return $AccountRole;
}

function FetchSession($Action,$AliID){
    global $PDOconn;
    $Query = 'CALL FetchSession (?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $AliID, PDO::PARAM_STR, 64);
    Execute($Action,$Statement);
    $Response = $Statement->fetch(PDO::FETCH_ASSOC);
    return $Response;
}

function PasswordGenerator(){
    //https://www.catchstudio.com/labs/password-generator/
    //$Password = hash('sha256', uniqid(rand(), true));
    // Characters to use for the password
    $Strings = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789.-
+=_,!@$#*%<>[]{}";

    // Desired length of the password
    $PasswordLength = 12;

    // Length of the string to take characters from
    $StringLength = strlen($Strings);

    // RANDOM.ORG - We are pulling our list of random numbers as a
    // single request, instead of iterating over each character individually
    $uri = "http://www.random.org/integers/?";
    $Random = file_get_contents(
        $uri ."num=$PasswordLength&min=0&max=".($StringLength-
1)."&col=1&base=10&format=plain&rnd=new"
    );
    $Indexes = explode("\n", $Random);
    array_pop($Indexes);

    // We now have an array of random indexes which we will use to build our password
    $Password = '';
    foreach ($Indexes as $int){
        $Password .= substr($Strings, $int, 1);
    }

    return $Password;
}

function GetBrowserData(){
    $SessionIP = $_SERVER['REMOTE_ADDR'];
    $u_agent = $_SERVER['HTTP_USER_AGENT'];
    $BrowserName = 'Unknown';
    $Platform = 'Unknown';

    if (preg_match('/linux/i', $u_agent)) {
        $Platform = 'Linux';
    }
    elseif (preg_match('/macintosh|mac os x/i', $u_agent)) {
        $Platform = 'Mac';
    }
    elseif (preg_match('/windows|win32/i', $u_agent)) {
        $Platform = 'Windows';
    }

    // Next get the name of the useragent yes seperately and for refresh reason
    if(preg_match('/MSIE/i',$u_agent) && !preg_match('/Opera/i',$u_agent))
    {
        $BrowserName = 'Internet Explorer';
    }
    elseif(preg_match('/Firefox/i',$u_agent))
    {
        $BrowserName = 'Mozilla Firefox';
    }
    elseif(preg_match('/Chrome/i',$u_agent))
    {
        $BrowserName = 'Google Chrome';
    }
    elseif(preg_match('/Safari/i',$u_agent))
    {
        $BrowserName = 'Apple Safari';
    }
    elseif(preg_match('/Opera/i',$u_agent))
    {
        $BrowserName = 'Opera';
    }
    elseif(preg_match('/Netscape/i',$u_agent))
    {
        $BrowserName = 'Netscape';
    }

    return array(
        'IP' => $SessionIP,
        'Browser' => $BrowserName,
        'Platform' => $Platform
    );
}

//Single use
function UnitTest($Action){
    global $PDOconn;
    $Query = 'CALL UTCreate';
    $Statement = $PDOconn->prepare($Query);
    Execute($Action,$Statement);

    $UTValue = "1";
    $Query = 'CALL UTInsert (?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $UTValue, PDO::PARAM_INT);
    Execute($Action,$Statement);

    $UpdatedUTValue = "2";
    $Query = 'CALL UTUpdate (?, ?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $UpdatedUTValue, PDO::PARAM_INT);
    $Statement->bindParam(2, $UTValue, PDO::PARAM_INT);
    Execute($Action,$Statement);

    $Query = 'CALL UTDelete (?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $UpdatedValue, PDO::PARAM_INT);
    Execute($Action,$Statement);

    $Query = 'CALL UTDrop';
    $Statement = $PDOconn->prepare($Query);
    Execute($Action,$Statement);
    echo json_encode("Unit Test successful");//do I need try and catch for unit test? do I need
to fetch?
    $PDOconn = null;
}

function AddError($Action,$ErrorMSG,$PHP){
    $Email = ValidateSession($Action);
    if (!isset($Email)) {
        $Email = NULL;
    }
    if (!isset($PHP)) {
        $FailedAction = stripslashes($_POST["FailedAction"]);
        $ErrorMSG = stripslashes($_POST["ErrorMSG"]);
    }else{
        $FailedAction = $Action;
    }
    global $PDOconn;
    $Query = 'CALL AddError (?, ?, ?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $Email, PDO::PARAM_STR, 45);
    $Statement->bindParam(2, $FailedAction, PDO::PARAM_STR, 45);
    $Statement->bindParam(3, $ErrorMSG, PDO::PARAM_STR, 255);
    $Statement->execute();
    $PDOconn = null;
}

function UpdateAccountStatus($Action){
    $AdminEmail = ValidateSession($Action);
    AdminRole($Action,$AdminEmail);
    $Disabled = stripslashes($_POST["D1"]);
    $Email = stripslashes($_POST["D2"]);
    global $PDOconn;
    $Query = 'CALL UpdateAccountStatus (?, ?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $Disabled, PDO::PARAM_INT, 1);
    $Statement->bindParam(2, $Email, PDO::PARAM_STR, 45);
    Execute($Action,$Statement);
    if($Disabled == 0){
        $ActivityMSG = "Your account was activated by an Admin.";
        AddActivity($Action,$Email,$ActivityMSG);
        mail($Email,"Account activated","Your account was activated by an Admin.");
        $ActivityMSG = "You activated " . $Email . "'s account.";
        AddActivity($Action,$AdminEmail,$ActivityMSG);
    }else{
        $ActivityMSG = "Your account was dis-activated by an Admin.";
        AddActivity($Action,$Email,$ActivityMSG);
        mail($Email,"Account dis-activated","Your account was dis-activated by an Admin.");
        $ActivityMSG = "You dis-activated " . $Email . "'s account";
        AddActivity($Action,$AdminEmail,$ActivityMSG);
    }
    echo json_encode("refresh");
    $PDOconn = null;
}

function UpdatePetStatus($Action){
    $AdminEmail = ValidateSession($Action);
    AdminRole($Action,$AdminEmail);
    $Disabled = stripslashes($_POST["D1"]);
    $PetName = stripslashes($_POST["D2"]);
    $PetID = stripslashes($_POST["D3"]);
    $Email = stripslashes($_POST["D4"]);
    global $PDOconn;
    $Query = 'CALL UpdatePetStatus (?, ?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $Disabled, PDO::PARAM_INT, 1);
    $Statement->bindParam(2, $PetID, PDO::PARAM_INT);
    Execute($Action,$Statement);
    if($Disabled == 0){
        $ActivityMSG = $PetName . " has been activated by an Admin.";
        AddActivity($Action,$Email,$ActivityMSG);
        mail($Email,"Pet activated","Your pet " . $PetName . " has been activated by an Admin.");
        $ActivityMSG = "You activated " . $Email . "'s pet " . $PetName . ".";
        AddActivity($Action,$AdminEmail,$ActivityMSG);
    }else{
        $ActivityMSG = $PetName . " has been dis-activated by an Admin.";
        AddActivity($Action,$Email,$ActivityMSG);
        mail($Email,"Pet dis-activated","Your pet " . $PetName . " has been dis-activated by an
Admin.");
        $ActivityMSG = "You dis-activated " . $Email . "'s pet " . $PetName . ".";
        AddActivity($Action,$AdminEmail,$ActivityMSG);
    }
    echo json_encode("refresh");
    $PDOconn = null;
}

function UpdatePetName($Action){
    $AdminEmail = ValidateSession($Action);
    AdminRole($Action,$AdminEmail);
    $PetName = stripslashes($_POST["D1"]);
    $OldPetName = stripslashes($_POST["D2"]);
    $Email = stripslashes($_POST["D3"]);
    global $PDOconn;
    $Query = 'CALL UpdatePetName (?, ?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $PetName, PDO::PARAM_STR, 45);
    $Statement->bindParam(2, $Email, PDO::PARAM_STR, 45);
    Execute($Action,$Statement);
    $ActivityMSG = $OldPetName . "'s name was changed to " . $PetName . " by an Admin.";
    AddActivity($Action,$Email,$ActivityMSG);
    mail($Email,"Pet name change","Your pet " . $OldPetName . "'s name was changed to " .
$PetName . " by an Admin.");
    $ActivityMSG = "You changed " . $Email . "'s pet name to " . $PetName . " from " .
$OldPetName . ".";
    AddActivity($Action,$AdminEmail,$ActivityMSG);
    echo json_encode("refresh");
    $PDOconn = null;
}

function UpdatePetBreed($Action){
    $AdminEmail = ValidateSession($Action);
    AdminRole($Action,$AdminEmail);
    $BreedID = stripslashes($_POST["D1"]);
    $PetName = stripslashes($_POST["D2"]);
    $Email = stripslashes($_POST["D3"]);
    global $PDOconn;
    $Query = 'CALL UpdatePetBreed (?, ?, ?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $BreedID, PDO::PARAM_INT);
    $Statement->bindParam(2, $PetName, PDO::PARAM_STR, 45);
    $Statement->bindParam(3, $Email, PDO::PARAM_STR, 45);
    Execute($Action,$Statement);
    $ActivityMSG = $PetName . "'s breed was changed by an Admin.";
    AddActivity($Action,$Email,$ActivityMSG);
    mail($Email,"Pet breed change", $PetName . "'s breed was changed by an Admin.");
    $ActivityMSG = "You've changed the breed of " . $Email . "'s pet name " . $PetName . ".";
    AddActivity($Action,$AdminEmail,$ActivityMSG);
    echo json_encode("refresh");
    $PDOconn = null;
}

function UpdatePetGender($Action){
    $AdminEmail = ValidateSession($Action);
    AdminRole($Action,$AdminEmail);
    $Gender = stripslashes($_POST["D1"]);
    $PetName = stripslashes($_POST["D2"]);
    $Email = stripslashes($_POST["D3"]);
    global $PDOconn;
    $Query = 'CALL UpdatePetGender (?, ?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $Gender, PDO::PARAM_STR, 4);
    $Statement->bindParam(2, $Email, PDO::PARAM_STR, 45);
    Execute($Action,$Statement);
    $ActivityMSG = $PetName . "'s gender was changed to a " . $Gender . " by an Admin.";
    AddActivity($Action,$Email,$ActivityMSG);
    mail($Email,"Pet gender change","Your pet " . $PetName . "'s gender was changed to " .
$Gender . " by an Admin.");
    $ActivityMSG = "You changed the gender of " . $Email . "'s pet name " . $PetName . ".";
    AddActivity($Action,$AdminEmail,$ActivityMSG);
    echo json_encode("refresh");
    $PDOconn = null;
}

function UpdateBreed($Action){
    $Email = ValidateSession($Action);
    AdminRole($Action,$Email);
    $BreedName = stripslashes($_POST["D1"]);
    $OldBreedName = stripslashes($_POST["D2"]);
    $BreedID = stripslashes($_POST["D3"]);
    global $PDOconn;
    $Query = 'CALL UpdateBreed (?, ?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $BreedName, PDO::PARAM_INT);
    $Statement->bindParam(2, $BreedID, PDO::PARAM_STR, 45);
    Execute($Action,$Statement);
    $ActivityMSG = "Breed changed to " . $BreedName . " from " . $OldBreedName . " by an Admin.";
    AddActivity($Action,$Email,$ActivityMSG);
    echo json_encode("refresh");
    $PDOconn = null;
}
```

```
function AddBreed($Action){
    $Email = ValidateSession($Action);
    AdminRole($Action,$Email);
    $Name = stripslashes($_POST["D1"]);
    global $PDOconn;
    $Query = 'CALL FetchBreedNameCount (?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $Name, PDO::PARAM_STR, 45);
    Execute($Action,$Statement);
    $Response = $Statement->fetch(PDO::FETCH_ASSOC);
    $Statement->closeCursor();
    if($Response['Count'] == 0){
        $Query = 'CALL AddBreed (?)';
        $Statement = $PDOconn->prepare($Query);
        $Statement->bindParam(1, $Name, PDO::PARAM_STR, 45);
        Execute($Action,$Statement);
        $ActivityMSG = "You added " . $Name . " as a new breed.";
        AddActivity($Action,$Email,$ActivityMSG);
        echo json_encode("refresh");
        exit;
    }else{
        echo json_encode("breedexist");
        exit;
    }
}

function AddAttempt($Action,$UserData,$Email){
    $NewAttempt = $UserData['Attempt'];
    $NewAttempt++;
    global $PDOconn;
    $Query = 'CALL AddAttempt (?, ?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $NewAttempt, PDO::PARAM_INT, 1);
    $Statement->bindParam(2, $Email, PDO::PARAM_STR, 45);
    Execute($Action,$Statement);
}

function SignIn($Action){
    $Email = stripslashes($_POST["D1"]);
    $Password = stripslashes($_POST["D2"]);
    $UserData = FetchUser($Action,$Email);
    if(!empty($UserData['Email'])){
        $HashedPassword = $UserData['Password'];
        $PasswordResponse = ValidatePassword($Password,$HashedPassword);
        if($Email == $UserData['Email'] && $PasswordResponse == 1 && $UserData['Attempt'] < 5){
            if($UserData['Disabled'] == 0) {
                ResetAttempt($Action,$Email);
                DeleteSession($Action,$Email);
                AddSession($Action,$Email);
                $ActivityMSG = "You signed in.";
                AddActivity($Action,$Email,$ActivityMSG);
                echo json_encode("refresh");
                $PDOconn = null;
            }else{
                $ActivityMSG = "You attempted to sign in but your account wasn't activated by an
admin yet.";
                AddActivity($Action,$Email,$ActivityMSG);
                echo json_encode("notactive");
                $PDOconn = null;
            }
        }else{
            if($UserData['Attempt'] < 5){
                AddAttempt($Action,$UserData,$Email);
                $ActivityMSG = "Account to be locked due to multipel sign in attempts.";
                AddActivity($Action,$Email,$ActivityMSG);
                echo json_encode("notlocked");
                $PDOconn = null;
            }else{
                $ActivityMSG = "Account was locked out due to multiple sign in attempts.";
                AddActivity($Action,$Email,$ActivityMSG);
                echo json_encode("locked");
                $PDOconn = null;
            }
        }
    }else{
        echo json_encode("none");
        $PDOconn = null;
    }
}

function UpdatePassword($Action){
    $Email = ValidateSession($Action);
    $OldPassword = stripslashes($_POST["D1"]);
    $NewPassword = stripslashes($_POST["D2"]);
    $UserData = FetchUser($Action,$Email);
    $HashedPassword = $UserData['Password'];
    $PasswordResponse = ValidatePassword($OldPassword,$HashedPassword);
    if($Email == $UserData['Email'] && $PasswordResponse == 1){
        $NewHashedPassword = HashIt($NewPassword);
        global $PDOconn;
        $Query = 'CALL UpdatePassword (?, ?)';
        $Statement = $PDOconn->prepare($Query);
        $Statement->bindParam(1, $NewHashedPassword, PDO::PARAM_STR, 64);
        $Statement->bindParam(2, $Email, PDO::PARAM_STR, 45);
        Execute($Action, $Statement);
        $ActivityMSG = "Your password was changed.";
        AddActivity($Action, $Email, $ActivityMSG);
        mail($Email, "Password was changed", "Your password was changed.");
        echo json_encode("pupdated");
        $PDOconn = null;
    }else{
        echo json_encode("xupdated");
    }
}

function ResetPassword($Action){
    $Email = stripslashes($_POST["D1"]);
    $UserData = FetchUser($Action,$Email);
    if(!empty($UserData['Email'])){
        $Password = PasswordGenerator();
        $HashedPassword = HashIt($Password);
        global $PDOconn;
        $Query = 'CALL UpdatePassword (?, ?)';
        $Statement = $PDOconn->prepare($Query);
        $Statement->bindParam(1, $HashedPassword, PDO::PARAM_STR, 64);
        $Statement->bindParam(2, $Email, PDO::PARAM_STR, 45);
        Execute($Action, $Statement);
        $ActivityMSG = "Your password was changed.";
        AddActivity($Action, $Email, $ActivityMSG);
        mail($Email, "Password was reset", "Your password was reset to " . $Password . ".");
        echo json_encode("refresh");
        $PDOconn = null;
    }else{
        echo json_encode("none");
    }
}

function SignInPet($Action){
    $Email = ValidateSession($Action);
    $Name = stripslashes($_POST["D1"]);
    $ActivityMSG = "Your pet " . $Name . " has been signed in.";
    AddActivity($Action,$Email,$ActivityMSG);
    echo json_encode("refresh");
}

function DeleteSession($Action,$Email){
    global $PDOconn;
    $Query = 'CALL DeleteSession (?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $Email, PDO::PARAM_STR, 45);
    Execute($Action,$Statement);
```

```
}

function ResetAttempt($Action,$Email){
    global $PDOconn;
    $Query = 'CALL ResetAttempt (?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $Email, PDO::PARAM_STR, 45);
    Execute($Action,$Statement);
}

function AddAccount($Action){
    $Email = stripslashes($_POST["D1"]);
    $UserData = FetchUser($Action,$Email);
    if($Email == $UserData['Email']){
        if($UserData['Attempt'] < 5){
            AddAttempt($Action,$UserData,$Email);
            $ActivityMSG = "Account to be locked due to multiple registration attempts.";
            AddActivity($Action,$Email,$ActivityMSG);
            echo json_encode("notlocked");
            exit;
        }else{
            $ActivityMSG = "Account was locked out due to multiple registration attempts.";
            AddActivity($Action,$Email,$ActivityMSG);
            echo json_encode("locked");
            exit;
        }
    }
    $Password = stripslashes($_POST["D2"]);
    $HashedPassword = HashIt($Password);
    $Disabled = 1;
    $Attempt = 0;
    $AdminCode = 1;
    global $PDOconn;
    $Query = 'CALL AddAccount (?, ?, ?, ?, ?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $Email, PDO::PARAM_STR, 45);
    $Statement->bindParam(2, $HashedPassword, PDO::PARAM_STR, 64);
    $Statement->bindParam(3, $Disabled, PDO::PARAM_INT, 1);
    $Statement->bindParam(4, $Attempt, PDO::PARAM_INT, 1);
    $Statement->bindParam(5, $AdminCode, PDO::PARAM_INT, 1);
    Execute($Action,$Statement);
    $ActivityMSG = "Your account was created.";
    AddActivity($Action,$Email,$ActivityMSG);
    mail($Email,"Your account was created","The following email: " . $Email . " has been created.
The account will be activated by an Admin.  In the meantime, familiarize yourself with the pet
policy. https://petsignin.alibkaba.com/petsignin/petpolicy.pdf");
    $AdminAccounts = FetchAdmins($Action);
    foreach ($AdminAccounts as $AdminEmail) {
        mail($AdminEmail['Email'],"New account created","The following email: " . $Email . " has
been created.  Account is awaiting your approval.");
    }
    echo json_encode("refresh");
    $PDOconn = null;
}

function AddPet($Action){
    $Email = ValidateSession($Action);
    $Name = stripslashes($_POST["D1"]);
    $BreedID = stripslashes($_POST["D2"]);
    $Gender = stripslashes($_POST["D3"]);
    $Disabled = 1;
    global $PDOconn;
    $Query = 'CALL AddPet (?, ?, ?, ?, ?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $Email, PDO::PARAM_STR, 45);
    $Statement->bindParam(2, $Name, PDO::PARAM_STR, 45);
    $Statement->bindParam(3, $BreedID, PDO::PARAM_INT);
    $Statement->bindParam(4, $Gender, PDO::PARAM_STR, 4);
    $Statement->bindParam(5, $Disabled, PDO::PARAM_INT, 1);
    Execute($Action,$Statement);
    $ActivityMSG = "Your new pet " . $Name . " has been added.";
```

```php
    AddActivity($Action,$Email,$ActivityMSG);
    mail($Email,"Your pet was added","The following pet: " . $Name . " has been added.  Please go to this link (https://petsignin.alibkaba.com/petsignin/upload.html) URL to upload the necessary documentation of your pet requested from the pet policy: Pet Policy https://petsignin.alibkaba.com/petsignin/petpolicy.pdf");
    echo json_encode("refresh");
    $PDOconn = null;
}

function FetchActivities($Action){
    $Email = ValidateSession($Action);
    global $PDOconn;
    $Query = 'CALL FetchActivities (?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $Email, PDO::PARAM_STR, 45);
    Execute($Action,$Statement);
    $Response = $Statement->fetchAll();
    echo json_encode($Response);
    $PDOconn = null;
}

function FetchErrors($Action){
    $Email = ValidateSession($Action);
    AdminRole($Action,$Email);
    global $PDOconn;
    $Query = 'CALL FetchErrors';
    $Statement = $PDOconn->prepare($Query);
    Execute($Action,$Statement);
    $Response = $Statement->fetchAll();
    echo json_encode($Response);
    $PDOconn = null;
}

function FetchSignInPet($Action){
    $Email = ValidateSession($Action);
    UserRole($Action,$Email);
    global $PDOconn;
    $Query = 'CALL FetchSignInPet (?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $Email, PDO::PARAM_STR, 45);
    Execute($Action,$Statement);
    $Response = $Statement->fetchAll();
    echo json_encode($Response);
    $PDOconn = null;
}

function FetchUserStatus($Action){
    $Email = ValidateSession($Action);
    AdminRole($Action,$Email);
    $Email = stripslashes($_POST["D1"]);
    global $PDOconn;
    $Query = 'CALL FetchUserStatus (?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $Email, PDO::PARAM_STR, 45);
    Execute($Action,$Statement);
    $Response = $Statement->fetch(PDO::FETCH_ASSOC);
    echo json_encode($Response);
    $PDOconn = null;
}

function FetchPetStatus($Action){
    $Email = ValidateSession($Action);
    AdminRole($Action,$Email);
    $PetID = stripslashes($_POST["D1"]);
    global $PDOconn;
    $Query = 'CALL FetchPetStatus (?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $PetID, PDO::PARAM_STR, 45);
    Execute($Action,$Statement);
    $Response = $Statement->fetch(PDO::FETCH_ASSOC);
    echo json_encode($Response);
    $PDOconn = null;
}

function FetchPet($Action){
    $Email = ValidateSession($Action);
    AdminRole($Action,$Email);
    $PetID = stripslashes($_POST["D1"]);
    global $PDOconn;
    $Query = 'CALL FetchPet (?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $PetID, PDO::PARAM_STR, 45);
    Execute($Action,$Statement);
    $Response = $Statement->fetch(PDO::FETCH_ASSOC);
    echo json_encode($Response);
    $PDOconn = null;
}

function FetchUserPets($Action){
    $Email = ValidateSession($Action);
    AdminRole($Action,$Email);
    $AccountEmail = stripslashes($_POST["D1"]);
    global $PDOconn;
    $Query = 'CALL FetchUserPets (?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $AccountEmail, PDO::PARAM_STR, 45);
    Execute($Action,$Statement);
    $Response = $Statement->fetchAll();
    echo json_encode($Response);
    $PDOconn = null;
}

function FetchPetNameCount($Action){
    $Email = ValidateSession($Action);
    $Name = stripslashes($_POST["D1"]);
    global $PDOconn;
    $Query = 'CALL FetchPetNameCount (?, ?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $Email, PDO::PARAM_STR, 45);
    $Statement->bindParam(2, $Name, PDO::PARAM_STR, 45);
    Execute($Action,$Statement);
    $Response = $Statement->fetch(PDO::FETCH_ASSOC);
    $Statement->closeCursor();
    echo json_encode($Response);
    $PDOconn = null;
}

function FetchUsers($Action){
    $Email = ValidateSession($Action);
    AdminRole($Action,$Email);
    global $PDOconn;
    $Query = 'CALL FetchUsers';
    $Statement = $PDOconn->prepare($Query);
    Execute($Action,$Statement);
    $Response = $Statement->fetchAll();
    echo json_encode($Response);
    $PDOconn = null;
}

function FetchBreeds($Action){
    global $PDOconn;
    $Query = 'CALL FetchBreeds';
    $Statement = $PDOconn->prepare($Query);
    Execute($Action,$Statement);
    $Response = $Statement->fetchAll();
    echo json_encode($Response);
    $PDOconn = null;
}

function SignOut($Action){
    $Email = ValidateSession($Action);
    DeleteSession($Action,$Email);
    $ActivityMSG = "You signed out.";
    AddActivity($Action,$Email,$ActivityMSG);
    session_unset();
    session_destroy();
    echo json_encode("refresh");
}

function ValidateSession($Action){
    StartSession();
    if(isset($_SESSION['AliID'])){
        $AliID = $_SESSION["AliID"];
        $SessionData = FetchSession($Action,$AliID);
        $Email = $SessionData['Email'];
        $BrowserData = GetBrowserData();
        $AccountRole = FetchAccountRole($Action,$Email);
        if($SessionData['IP'] == $BrowserData['IP'] && $SessionData['Browser'] ==
$BrowserData['Browser'] && $SessionData['Platform'] == $BrowserData['Platform']){
            if($Action == "ValidateSession"){
                echo json_encode($AccountRole);
            }else{
                return $Email;
            }
        }else{
            session_unset();
            session_destroy();
            echo json_encode("expired");
            $PDOconn = null;
        }
    }else{
        if($Action == "ValidateSession"){
            echo json_encode(0);
        }else{
            echo json_encode("expired");
            exit;
        }
    }
}

function FetchUserEmail($Action){
    $Email = ValidateSession($Action);
    echo json_encode($Email);
}

function AddSession($Action,$Email){
    StartSession();
    $BrowserData = GetBrowserData();
    $AliID = hash('sha256', uniqid(rand(), true));
    $_SESSION["AliID"] = $AliID;
    $SessionIP = $BrowserData['IP'];
    $SessionBrowser = $BrowserData['Browser'];
    $SessionPlatform = $BrowserData['Platform'];

    global $PDOconn;
    $Query = 'CALL AddSession (?, ?, ?, ?, ?)';
    $Statement = $PDOconn->prepare($Query);
    $Statement->bindParam(1, $AliID, PDO::PARAM_STR, 64);
    $Statement->bindParam(2, $Email, PDO::PARAM_STR, 45);
    $Statement->bindParam(3, $SessionIP, PDO::PARAM_STR, 45);
    $Statement->bindParam(4, $SessionBrowser, PDO::PARAM_STR, 45);
    $Statement->bindParam(5, $SessionPlatform, PDO::PARAM_STR, 45);
    Execute($Action,$Statement);
}

function StartSession(){
    ini_set('session.cookie_lifetime', 1800);
    ini_set('session.gc_maxlifetime', 1800);
    session_start();
}
```

6.6 upload.php

```php
<?php
error_reporting(E_ALL);
ini_set('display_errors', 1);

if(isset($_POST['submit'])) {
    require_once('db.php');
    require_once('operations.php');

    $Email = stripslashes($_POST["Email"]);
    $Name = stripslashes($_POST["Name"]);
    if (!empty($Email) && !empty($Name)) {
        global $PDOconn;
        $Query = 'CALL FetchPetNameCount (?,?)';
        $Statement = $PDOconn->prepare($Query);
        $Statement->bindParam(1, $Email, PDO::PARAM_STR, 45);
        $Statement->bindParam(2, $Name, PDO::PARAM_STR, 45);
        $Statement->execute();
        $Response = $Statement->fetch(PDO::FETCH_ASSOC);
        $Statement->closeCursor();
        if($Response['Count'] == 1){
            $target_dir = "uploads/";
            $filename = $Email . "_" . $Name . "_" . basename($_FILES["fileToUpload"]["name"]);
            $target_file = $target_dir . $filename;
            $uploadOk = 1;
            $FileType = pathinfo($target_file,PATHINFO_EXTENSION);
            // Check if file already exists
            if (file_exists($target_file)) {
                echo nl2br("\r\n Error 60: Sorry, file already exists.  Ask an admin to remove it
before uploading a new file.");
                $uploadOk = 0;
            }
            // Check file size
            if ($_FILES["fileToUpload"]["size"] > 500000) {
                echo nl2br("\r\n Error 61: Sorry, your file is too large.");
                $uploadOk = 0;
            }
            // Allow certain file formats
            if($FileType != "pdf") {
                echo nl2br("\r\n Error 62: Sorry, only PDF files are allowed.");
                $uploadOk = 0;
            }
            // Check if $uploadOk is set to 0 by an error
            if ($uploadOk == 0) {
                echo nl2br("\r\n Error 64: Sorry, your file was not uploaded.");
                echo nl2br("\r\nYou will be redirected to the upload page in 5 seconds.");
                header('refresh: 5; url=upload.html');

                // if everything is ok, try to upload file
            } else {
                if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file)) {
                    echo nl2br("\r\nThe file ". basename( $_FILES["fileToUpload"]["name"]). " has
been uploaded.");
                    $Query = 'CALL UpdateDocument (?, ?, ?)';
                    $Statement = $PDOconn->prepare($Query);
                    $Statement->bindParam(1, $filename, PDO::PARAM_STR, 255);
                    $Statement->bindParam(2, $Email, PDO::PARAM_STR, 45);
                    $Statement->bindParam(3, $Name, PDO::PARAM_STR, 45);
                    $Statement->execute();
                    $ActivityMSG = "You uploaded " . $Name . "'s document.";
                    AddActivity($Email,$ActivityMSG);
                    $Action = "Upload";
                    $AdminAccounts = FetchAdmins($Action);
                    foreach ($AdminAccounts as $AdminEmail) {
                        mail($AdminEmail['Email'],"Pet document uploaded","The following account:
" . $Email . " has uploaded the pet documentation for " . $Name . ".  Pet is awaiting your
approval.");
                    }
                    echo nl2br("\r\nYou will be redirected to the homepage in 5 seconds.");
                    header('refresh: 5; url=index.html');
                } else {
                    echo nl2br("\r\n Error 65: Sorry, there was an error uploading your file.");
                    echo nl2br("\r\nYou will be redirected to the upload page in 5 seconds.");
                    header('refresh: 5; url=upload.html');
                }
            }

        }else{
            echo nl2br("\r\n Error 63: Sorry, you don't have a pet named " . $Name . ".");
            echo nl2br("\r\nYou will be redirected to the upload page in 5 seconds.");
            header('refresh: 5; url=upload.html');
        }
        $PDOconn = null;
    }

}
```

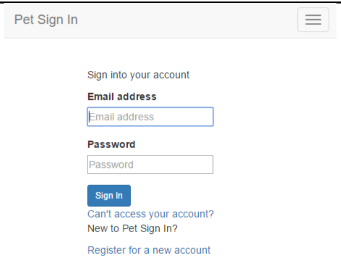
6.7. db.php

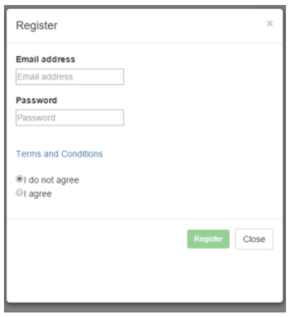
```
<?php
$dsn = "mysql:host=localhost;dbname=djkabau1_petsignin";
$u = "djkabau1_admin";
$p = "v,w_v;cpxzag";
try {
    $PDOconn = new PDO($dsn, $u, $p);
    $PDOconn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    echo 'Connection failed: ' . $e->getMessage() . "\n";
}
```

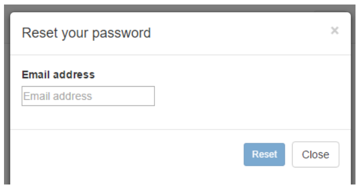
6.8 admin.php

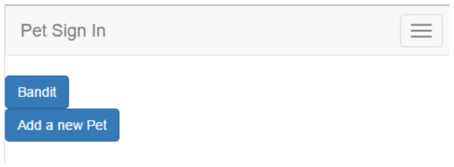
```
<html>
<body>
<p>Create Admin</p>
<form action="#" method='post'>
    Email: <input type="text" name="Email"><br>
    Password: <input type="password" name="Password"><br>
    <input type="submit" name="add" value="Submit" data-theme="b"/>
    <?php
    require_once('../db.php');
    require_once('../operations.php');
    error_reporting(E_ALL);
    ini_set('display_errors', 1);
    if(isset($_POST['add'])){
        $Action = "admin.php";
        $Email = $_POST['Email'];
        $Password = $_POST['Password'];
        $HashedPassword = HashIt($Password);
        $Disabled = 0;
        $Attempt = 0;
        $AdminCode = 2;
        global $PDOconn;
        $Query = 'CALL AddAdminAccount (?,?,?,?,?)';
        $Statement = $PDOconn->prepare($Query);
        $Statement->bindParam(1, $Email, PDO::PARAM_STR, 45);
        $Statement->bindParam(2, $HashedPassword, PDO::PARAM_STR, 255);
        $Statement->bindParam(3, $Disabled, PDO::PARAM_INT, 1);
        $Statement->bindParam(4, $Attempt, PDO::PARAM_INT, 1);
        $Statement->bindParam(5, $AdminCode, PDO::PARAM_INT, 1);
        $Statement->execute();
        $ActivityMSG = "Your account was created by an admin.";
        AddActivity($Action,$Email,$ActivityMSG);
        $MSG = "Super Admin created.";
        $PDOconn = null;
        echo "<script type='text/javascript'>alert('$MSG'); window.location =
\"/petsignin/sa/admin.php\";</script>";
    }
    ?>
</form>
</body>
</html>
```

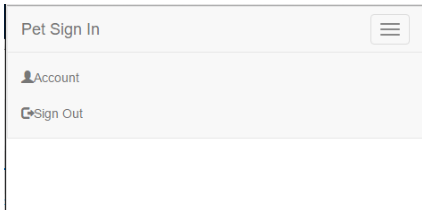
7 Screenshots

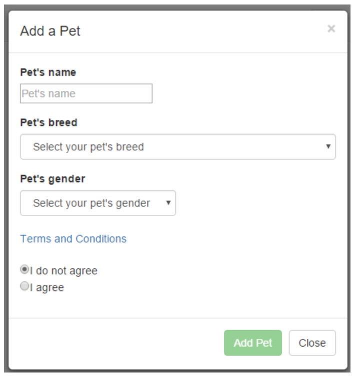











Add a Pet

Pet's name

Pet's name

Pet's breed

Select your pet's breed

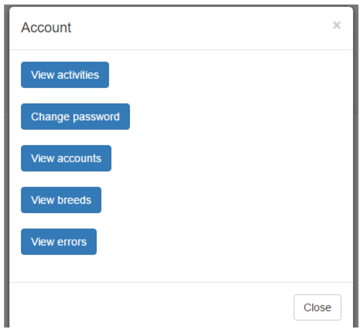
Pet's gender

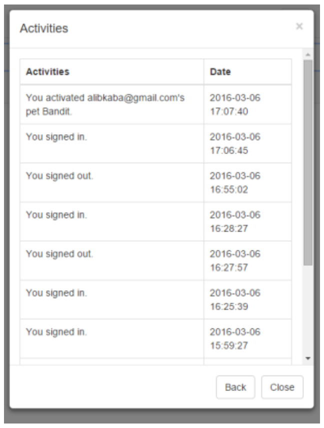
Select your pet's gender

Terms and Conditions

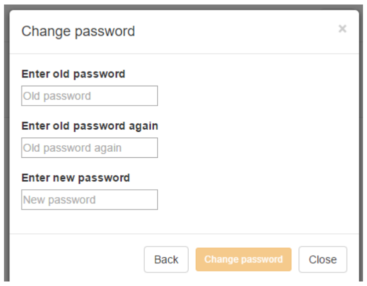
I do not agree [✓]

I agree




Activities

Activities	Date
You activated alibkaba@gmail.com's pet Bandit.	2016-03-06 17:07:40
You signed in.	2016-03-06 17:06:45
You signed out.	2016-03-06 16:55:02
You signed in.	2016-03-06 16:28:27
You signed out.	2016-03-06 16:27:57
You signed in.	2016-03-06 16:25:39
You signed in.	2016-03-06 15:59:27


Change password

Enter old password

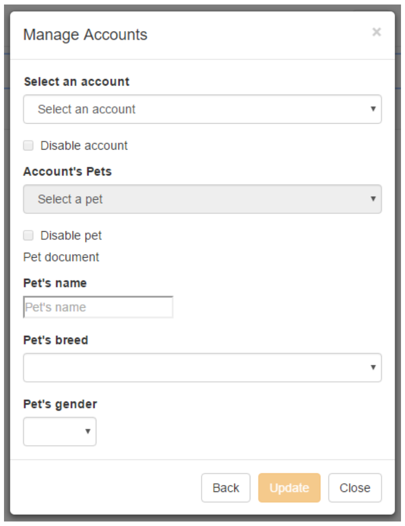
Old password

Enter old password again

Old password again

Enter new password

New password


Manage Accounts ×

Select an account

Select an account ▾

☐ Disable account

Account's Pets

Select a pet ▾

☐ Disable pet

Pet document

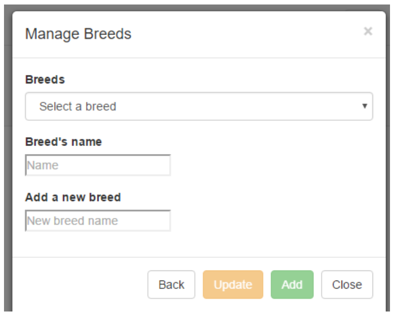
Pet's name

Pet's name

Pet's breed

Pet's gender

Back | Update | Close


Manage Breeds ×

Breeds

Select a breed ▾

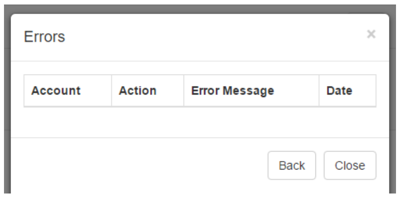
Breed's name

Name

Add a new breed

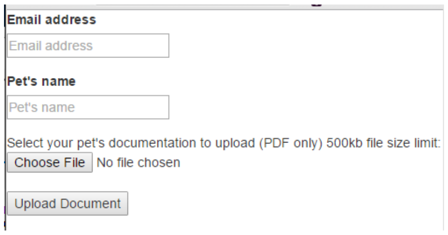
New breed name

Back | Update | Add | Close


Errors ×

Account	Action	Error Message	Date

Back | Close


Email address

Email address

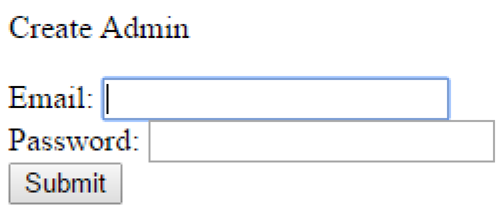
Pet's name

Pet's name

Select your pet's documentation to upload (PDF only) 500kb file size limit:

Choose File No file chosen

Upload Document


Create Admin

Email:

Password:

Submit

References

Cryptographic Storage Cheat Sheet. (n.d.). Retrieved from OWASP: https://www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet

Session hijacking attack. (n.d.). Retrieved from OWASP: https://www.owasp.org/index.php/Session_hijacking_attack

Session Management Cheat Sheet. (n.d.). Retrieved from OWASP: https://www.owasp.org/index.php/Session_Management_Cheat_Sheet

Testing for AJAX Vulnerabilities (OWASP-AJ-001). (n.d.). Retrieved from OWASP: https://www.owasp.org/index.php/Testing_for_AJAX_Vulnerabilities_(OWASP-AJ-001)

Testing for Code Injection (OTG-INPVAL-012). (n.d.). Retrieved from OWASP: https://www.owasp.org/index.php/Testing_for_Code_Injection_(OTG-INPVAL-012)

Testing for CSRF (OTG-SESS-005). (n.d.). Retrieved from OWASP: https://www.owasp.org/index.php/Testing_for_CSRF_(OTG-SESS-005)#Description_of_CSRF_Vulnerabilities

Testing for HTML Injection (OTG-CLIENT-003). (n.d.). Retrieved from OWASP: https://www.owasp.org/index.php/Testing_for_HTML_Injection_(OTG-CLIENT-003)

Testing for SQL Injection (OTG-INPVAL-005). (n.d.). Retrieved from OWASP: https://www.owasp.org/index.php/Testing_for_SQL_Injection_%28OTG-INPVAL-005%29

Transport Layer Protection Cheat Sheet. (n.d.). Retrieved from OWASP: https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet