Software Design – Pet R Us Website

[1. Overview - 4](#_Toc128738889)

[1.1 Purpose - 4](#_Toc128738890)

[1.2 Stakeholders – 4](#_Toc128738891)

[2. Document version - 5](#_Toc128738892)

[3. Design decision - 6](#_Toc128738893)

[3.1 Class diagram 6](#_Toc128738894)

[3.2 Database Diagram 7](#_Toc128738896)

[3.3 Event Flow diagram – Edit User Details 8](#_Toc128738897)

[3.4 Event Flow diagram – Sign Up & Login 9](#_Toc128738898)

[3.5 Event Flow diagram – Upload Image 10](#_Toc128738899)

[3.6 Event Flow diagram – Create Playdate 11](#_Toc128738900)

[3.7 Event Flow diagram – Accepting Playdate 12](#_Toc128738901)

[4. Mock Interface Design 13](#_Toc128738902)

[5. Pseudo Code 23](#_Toc128738903)

[5.1 Version Control 24](#_Toc128738904)

[6. Functions Explanation 24](#_Toc128738905)

[6.1 Register New User 24](#_Toc128738906)

[6.2 Uploading Image 25](#_Toc128738907)

[6.3 API 25](#_Toc128738908)

[6.4 Themes 25](#_Toc128738909)

[7. Test Cases - 26](#_Toc128738910)

# Overview -

## Purpose -

The purpose of this website is to provide Pets R Us a platform to display information about their business and the people that work there, it will also allow customers to create an account to upload photos of their pets to the website and select whether they prefer cats or dogs which will change the theme of the website to correspond with their choice.

## Stakeholders –

This document is intended for the Pet R Us team and shareholders, to give them an overview of the overall design of their new website and the thought process that went into designing and creating it.

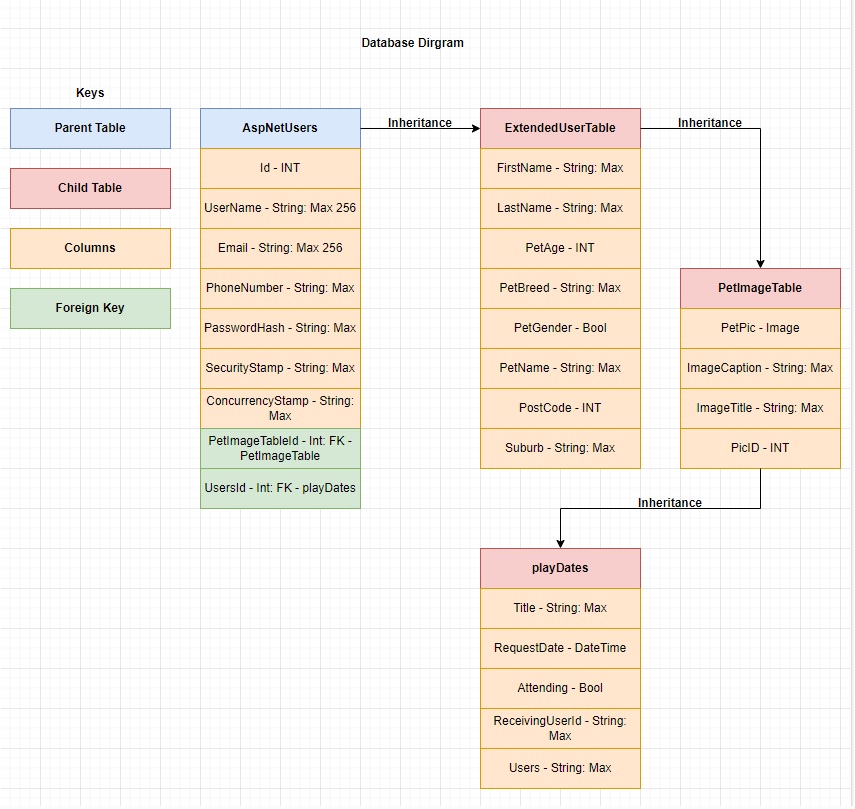
# Document version -

|  |  |  |
| --- | --- | --- |
| **Name** | **Section** | **Date** |
| Matthew Cockram | 1,2,3,4,5,6,7 | Mar, 03rd, 2023 |
| Jordan Wilson | 3,4,5,6,7 | Mar, 03rd, 2023 |

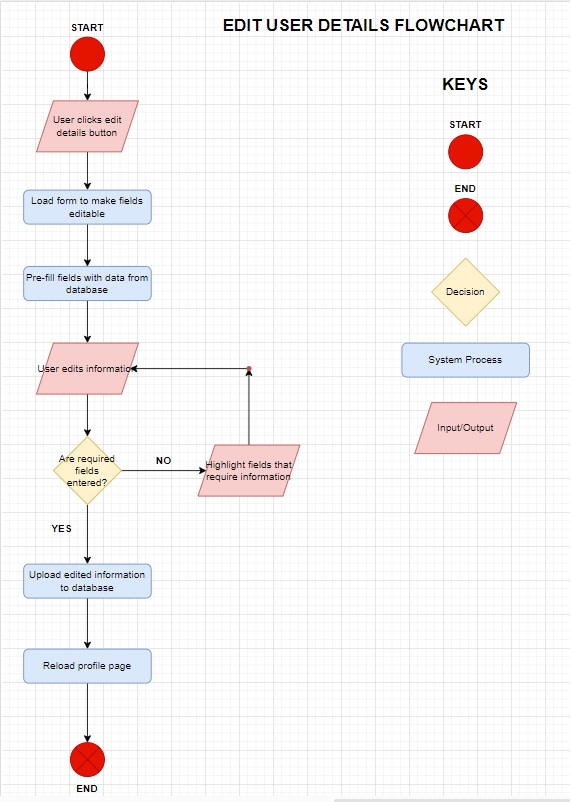
# Design decision -

## Class diagram

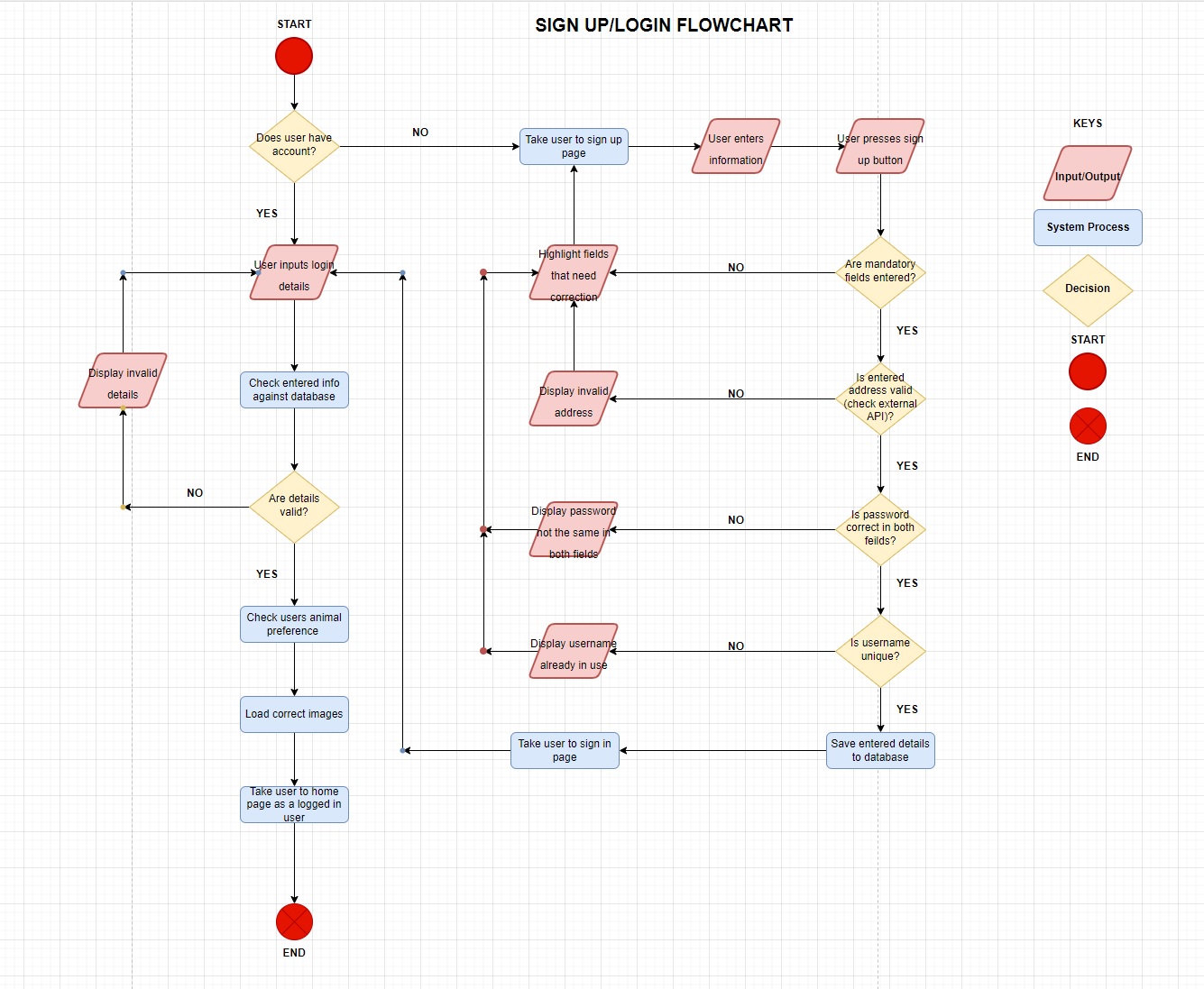
## Database Diagram



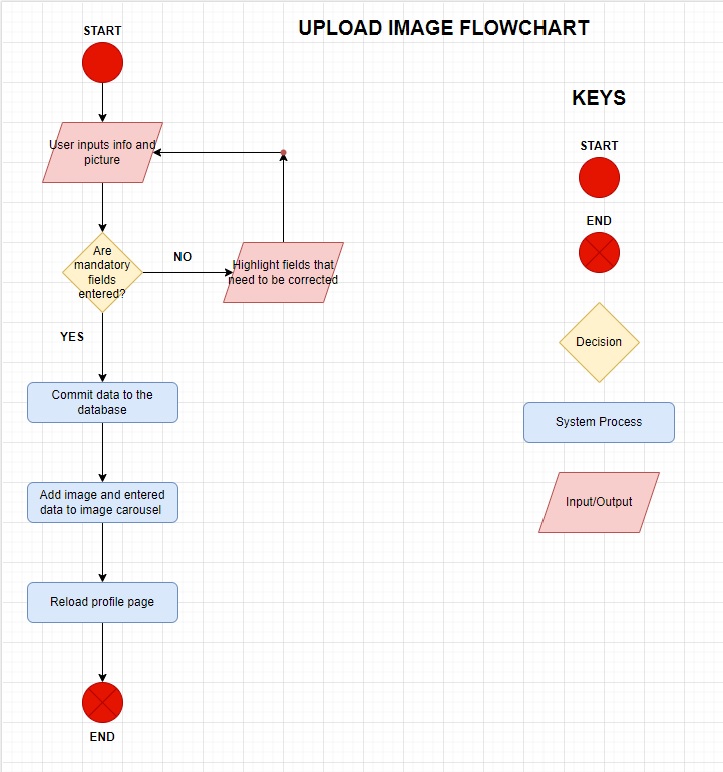
## Event Flow diagram – Edit User Details



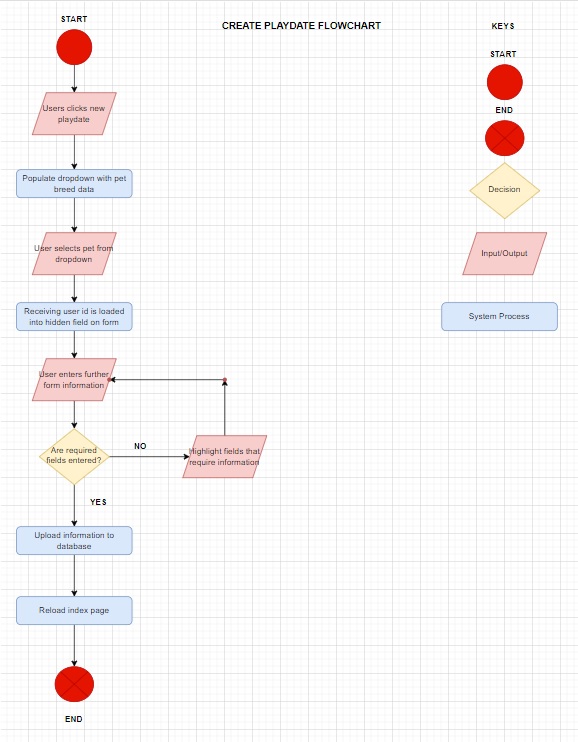
## Event Flow diagram – Sign Up & Login



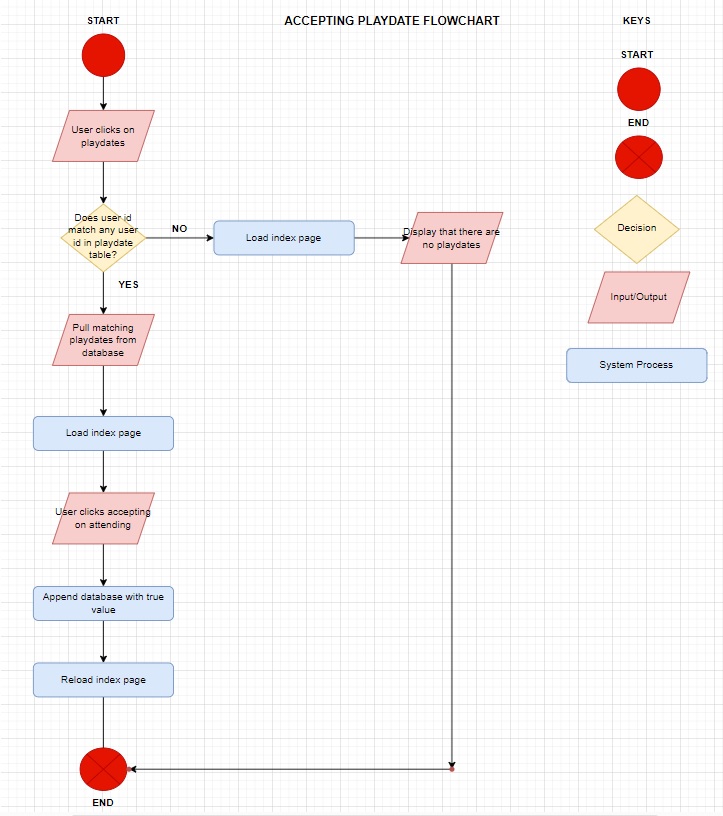
## Event Flow diagram – Upload Image



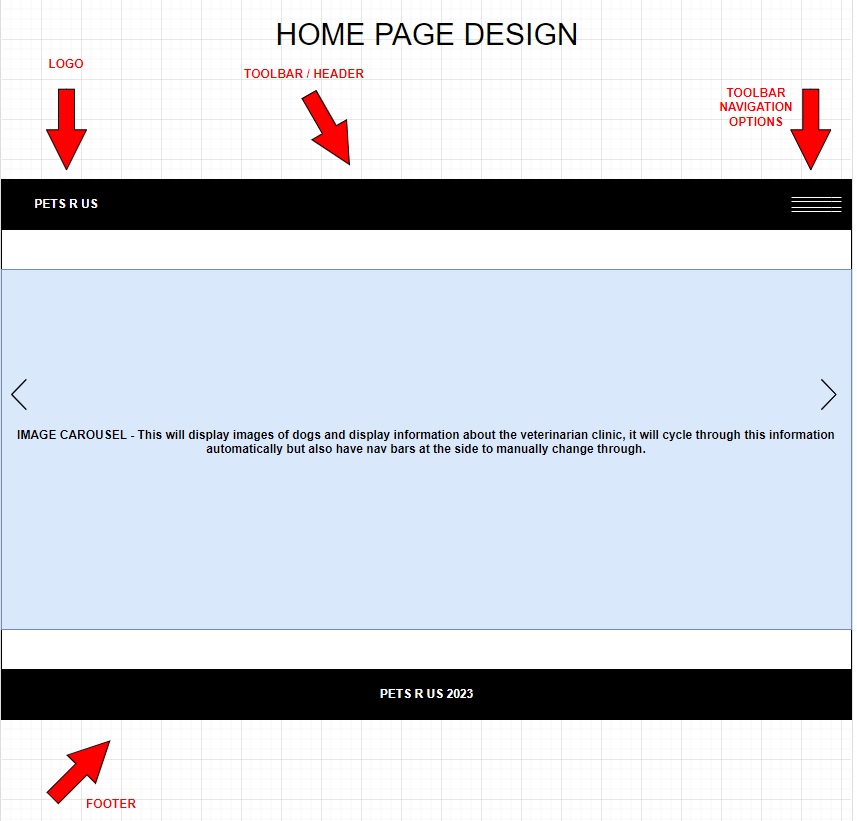
## Event Flow diagram – Create Playdate

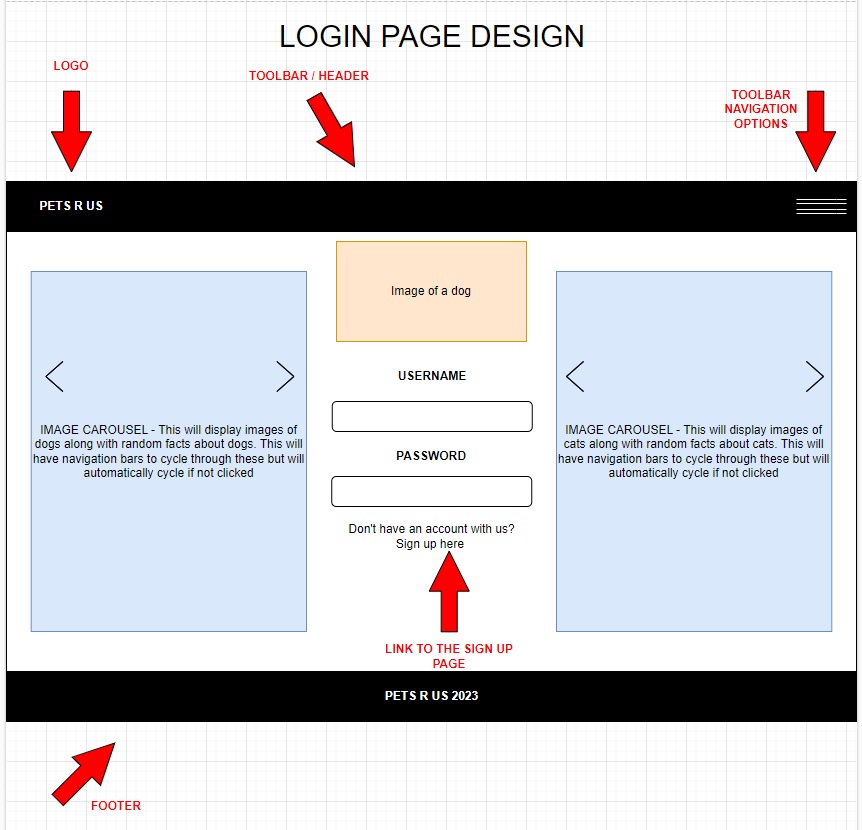


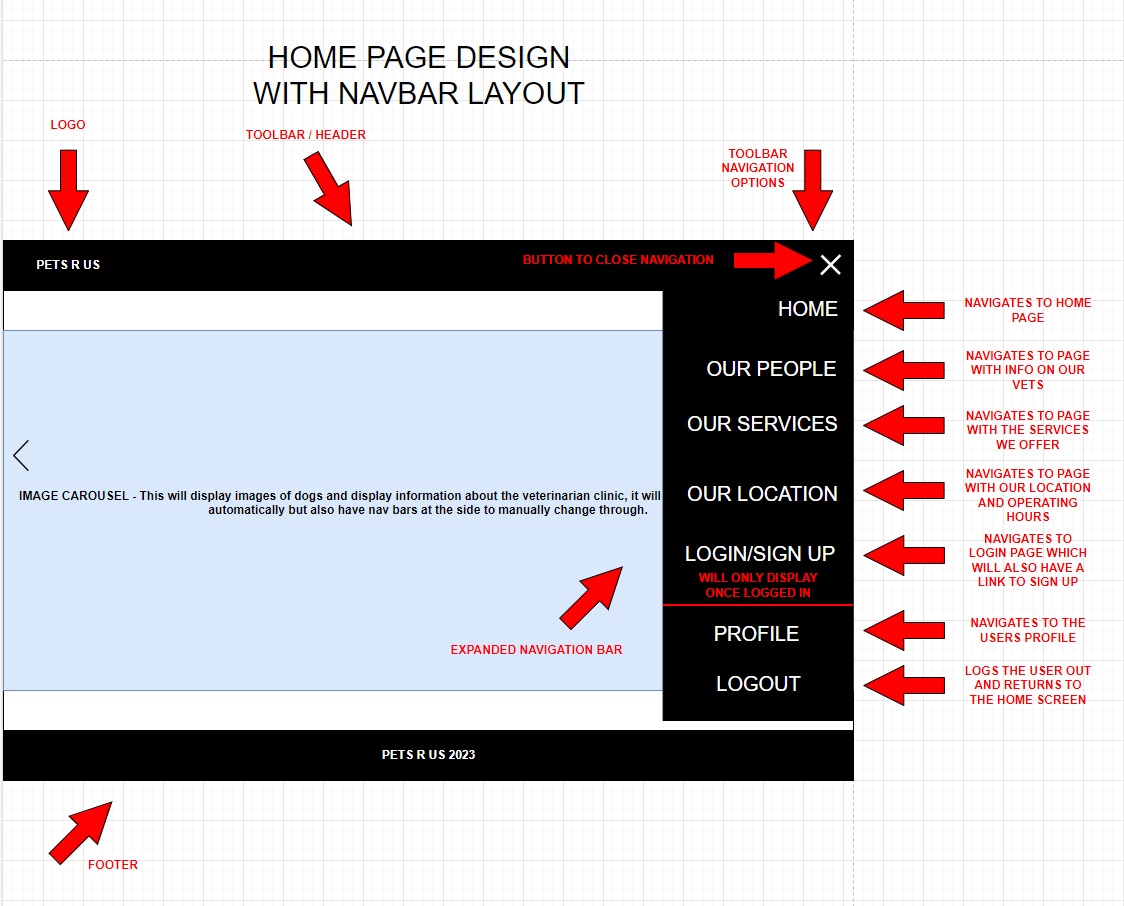
## Event Flow diagram – Accepting Playdate

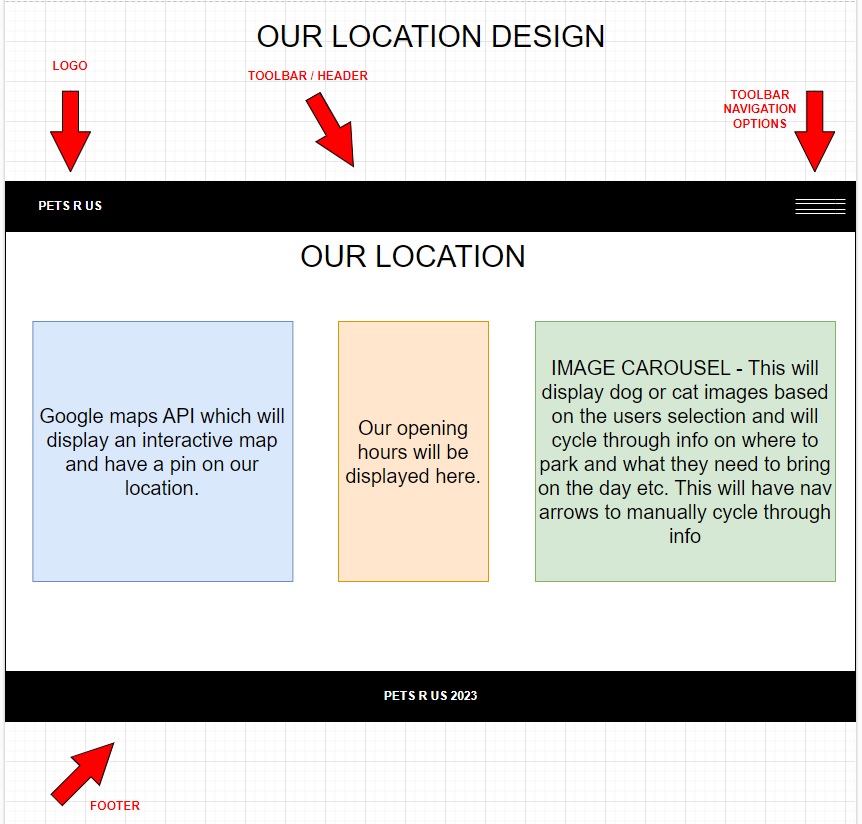


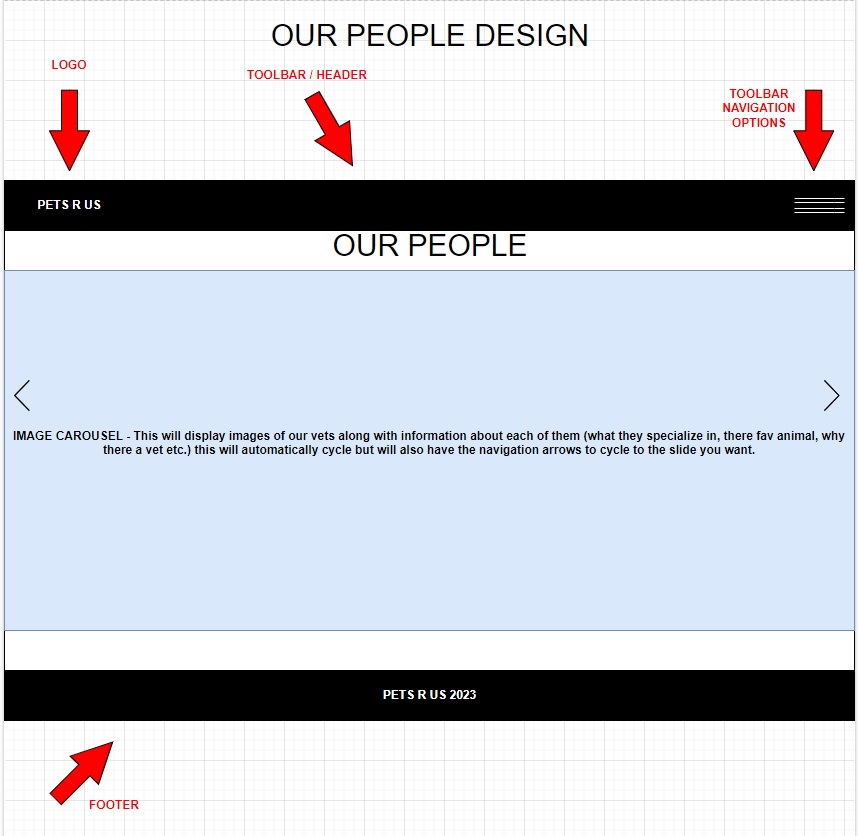
## Mock Interface Design

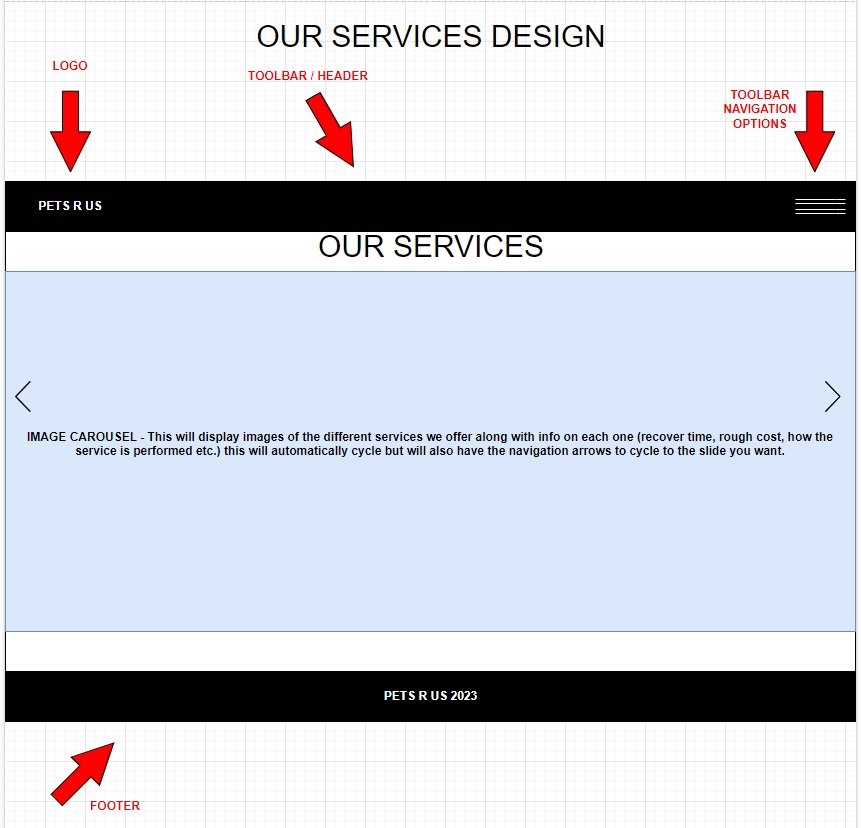


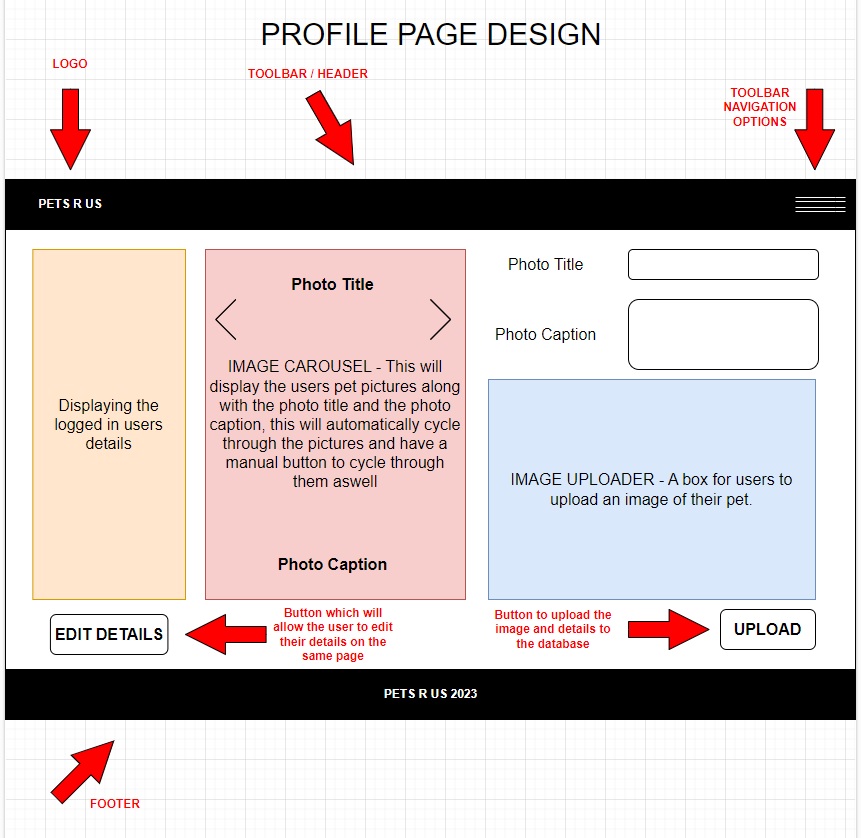


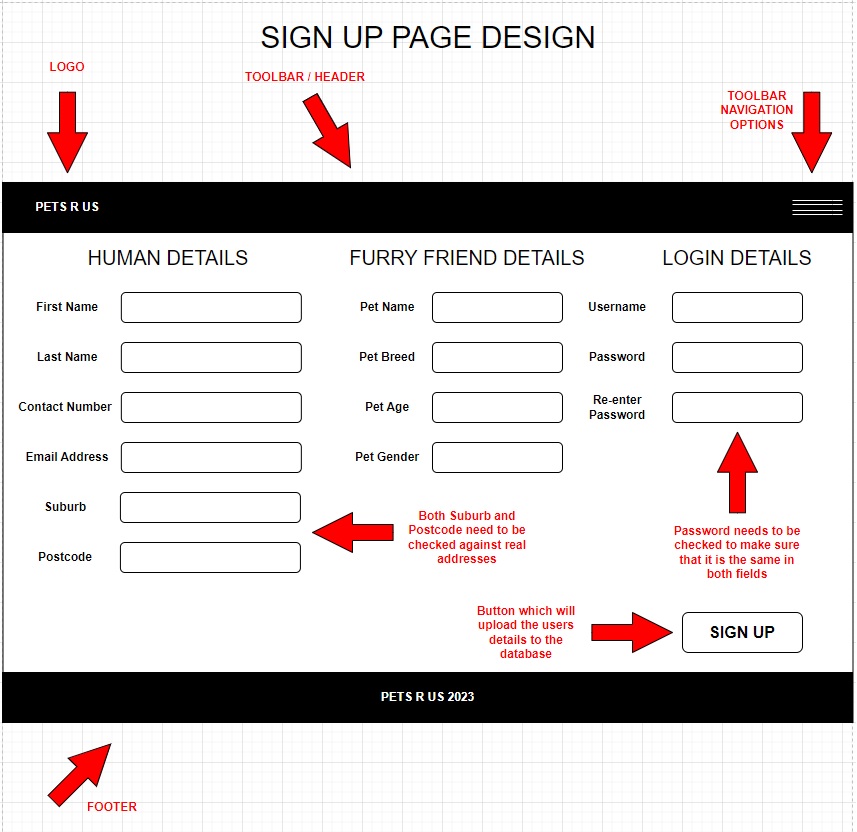


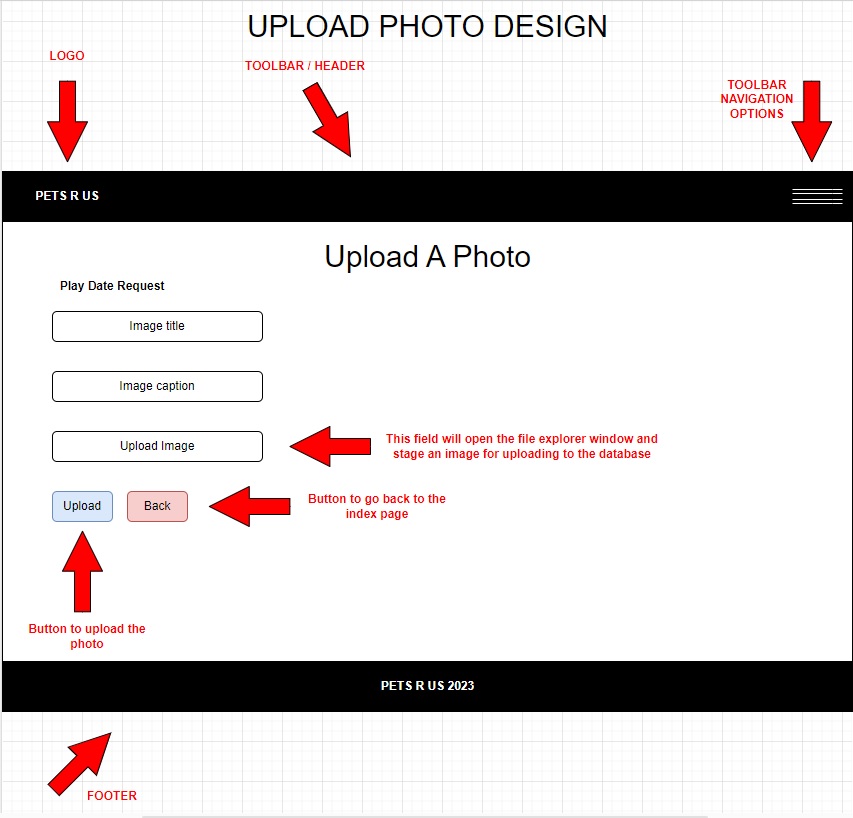
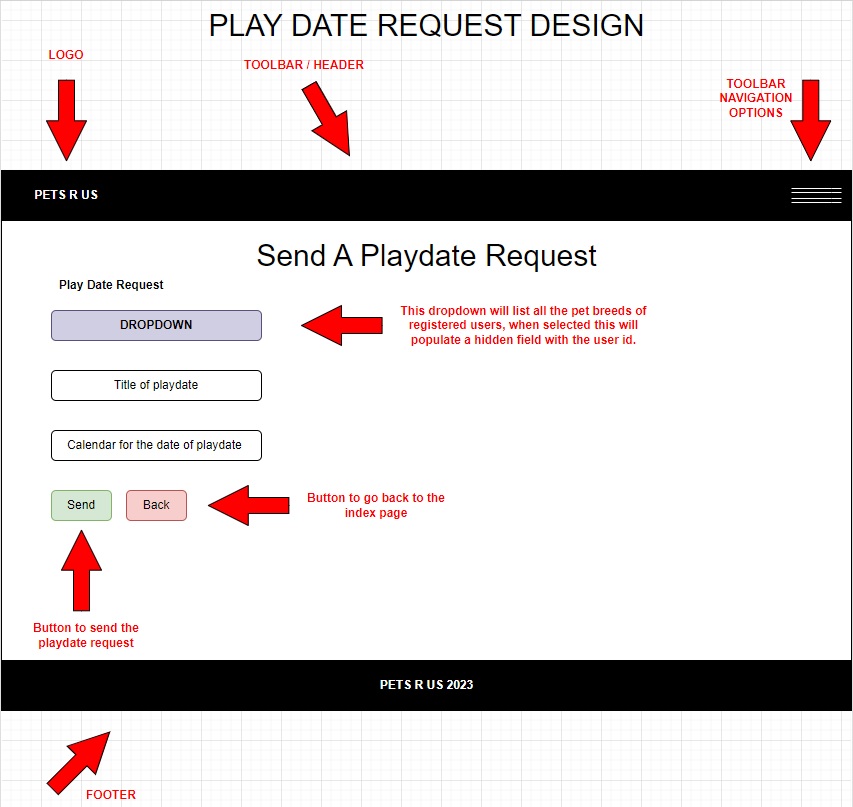












## Pseudo Code

**Login Page –**

* User enters login details
* System checks entered details against database
* System sets user preface of pet (cat, dog or both)
* System loads images from database
* System loads homepage

**Sign Up Page –**

* User enters their details
* User sets their preferred pet (cat, dog or both)
* System checks all require fields are filled out
* System checks that the address is valid
* System checks that the password is matching in both fields
* System checks that username is unique
* System commits the entered information to the database
* System loads the login page

**Upload Pet Image –**

* User enters photo details
* User drags photo into the uploader
* System checks the required fields are filled out
* System adds information to the data base
* System adds the information and image to the image carousel
* System reloads the profile page

**Edit Details Profile Page –**

* User presses edit details button
* System loads form to make fields editable
* System pre-fills fields with information from the database
* User edits information
* System checks all required fields are entered
* System commits changes to the database
* System reloads profile page

## Version Control

We have decided to work as a two-person team on this project, and we will be sharing all of our source code and design documentation on Github. To ensure that we are not working on the same piece of code, we have divided up the functions we need to do for the website to function as planned. This will help prevent conflicts when merging with main.

We haven’t put any protection on our main repository as we didn’t deem it necessary as there are only two of us working within the project and decide to just communicate directly with pushing and making changes.

# Functions Explanation

## Register New User

The form has basic level form validation in the form of Regex code for the password and email fields that look for specific characters to validate that it is a valid entry. This function takes the information from the form displayed in the UI and saves this information within the database in the correct tables.

We have all so included more advance form validation in the form of an external API which will validate the address entered against a known address database to validate that it is an actual address, we are all so using the MVC (Model View Controller) method to validate if an entry is correct according to the model we have set up which checks the model state before uploading anything to the database, if this model state doesn’t match the program will reject the entry.

No passwords will be saved in plain text within the database and we have used the in-built 32 bit encryption method that comes with ASP.net, this converts all characters of the password to a few different characters that now represent that character within the system, along with this it is linked with a Security Stamp and Concurrency Stamp that the system will validate the hashed password against which is to ensure that if someone obtains the hashed password they still won’t be able to login as they don’t have the Security Stamp and Concurrency Stamp that are linked with that password.

Once the form has passed all the validation checks it will add the information to the database and assign a random id to the user which will be used to identify the user from now on within the system, it will also assign a null value to the foreign key table of PetImageTableId as this doesn’t need a value until the user uploads an image of their pet, this will be handle by another function.

## Uploading Image

When a user clicks the upload image button in the user interface, the file explorer window opens, allowing them to browse through the files on their local machine. Once the user has found the image they want to upload and entered all other necessary data, the system will verify that the model state requirements have been met.

If the model state is compatible, the system will store the image to a folder on the server computer. It will then take the image's default name and assign it a randomly generated string that will be placed before the image name, preventing conflicts in the database where images have the same name.

It will then assign the string to the database table along with the other entered information, the system will link this assigned string with the image that is saved on the server machine, this is done to make the process more efficient as MySQL doesn’t allow images to be directly saved to it’s database which would have to then be converted in to a string of bytes and converted back when the image is required which will effect the system performance heavily, making for an unpleasant user experience.

## API

To gather the true Australian residential address of the user signing up we used an API from Loqate.com that will make the user use the API for the address field when signing up, making them enter their Australian address. It has been configured to take the street number, street name, suburb and postcode entered and puts this information in the address field of the User SQL table.

## Themes

The application has been configured to change between various themes, based on the role the user chooses when signing up to the application. To do these four layouts were made, three corresponding to the three themes (Cat, Dog and CatDog) and the fourth used for the Default theme when no user is signed in. As we only wanted signed in users to be able to use “Contact Us” feature we hid that menu item in the Default layout also.

6.5. Playdates

When the users clicks on the dropdown of pet breeds and selects a pet they would like to have a playdate with the system will load the user id of that pet into a hidden field in the form to identify the user this playdate is to be sent to, once the user fills out the rest of the form they can then click the send button.

The system will then check that the model state is valid, if it is the system will commit the information to the database with the attending Boolean value set to false. When the user that was sent the playdate next logs in and clicks on the playdates page the system will check that users id against all the playdates in the database and load any that match that id into a table which will then display on the index page.

If the user would like to attend the playdate they can click on the attending button on the table, this will get the system to update the attending value in the database to true, it will then reload the index page and show the updated information.

# Test Cases -

Test Case: Application does not display required output

| **Step** | **Description** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| --- | --- | --- | --- | --- |
| 1. | User clicks register | Application opens register page | Application opens register page | **PASS** |
| 2. | User doesn’t enter all required fields. | UI displays the fields to have to be filled out | UI displays the fields that need to be completed | **PASS** |
| 3. | User completes the register | Data saves to data base, UI asks user to confirm email | Data saves to data base, UI asks user to confirm email | **PASS** |
| 4. | User login with their new account | Application checks info against database and takes them to home screen | Home screen loaded, database check successful | **PASS** |
| 5. | User tries to login with wrong password | Application should reject login attempt and display the reason | Application rejects the attempt and displays “invalid login attempt” | **PASS** |
| 6. | User uploads a photo | Application saves image to folder and links to database, then displays image in UI | Application saves the string to the database and links to database, then displays image in the UI | **PASS** |
| 7. | User edits the caption of the image | Application should prefill from the database and commit any changes to database, then return to the image screen | Application doesn’t edit the database and loops back to edit screen | **FAIL** |
| 8. | User deletes uploaded image | Application will delete rows from data base and reload the page | Rows deleted and page reload | **PASS** |
| 9. | User logs out of account | Application will logout current user and return to home page with the default theme | Home page is loaded and default theme is loaded | **PASS** |
| 10. | User clicks on username to edit details | Application will load the manage account page and load data from database and commit any changes | Application loads page and updates changes in database | **PASS** |
| 11. | User clicks the nav bar | Nav bar will open on right hand side of screen | Nav bar opens on the right side of the screen | **PASS** |
| 12. | Image carousel should cycle through information and images automatically | Carousel cycles through loaded info | Carousel cycles through loaded info | **PASS** |
| 13. | User creates a new playdate | Information saves to database, Index page loads with the new playdate | Information is uploaded and index is reloaded | **PASS** |
| 14. | User tries to create a playdate without entering all the fields. | Form should reload and highlight fields that need to be entered | Website crashes with unhandled exception | **FAIL** |
| 15. | Receiving users loads the playdate page | Index page will load with the matching user id | Index page was loaded and only displayed the matching playdates | **PASS** |
| 16. | Receiving user accepts the playdate | Database will update with the true value and index page will reload | Database was updated and page was reloaded | **PASS** |