Programming III – Fall 2022

Course Project: Topic

Team Information

|  |  |  |
| --- | --- | --- |
|  | Name | Student Id |
| Member A | Hana Louiza Moussaoui | 2275881 |
| Member B | Taryn Beaupre | 2173710 |

Project Description

Our application allows the user to browse a virtual pet adoption center and to adopt one or multiple pets. The user can refresh the pet display page to see randomly selected pets from the pet database and click on a pet to see details such as age, type, adoption status, and description of the pet. There is the option to view only adopted or only available pets as well. They can then proceed and adopt the pet if they wish, by filling out an adoption form.

Development Approach

1. **Understanding the problem.**

Develop a pet adoption/rescue center that will be able to show pets available in the center and let the user adopt one or many pets.

1. **Formulating the problem.**

A pet database will keep track of which pet has been adopted or not. In the main window, the pets need to be randomly displayed to the user. The user should be able to refresh the page and there should not be duplicate pets on the window. The user should also be able to select a pet to view the pet’s details.

The user can decide to adopt a pet and a new window will appear where they can fill out information. The user can only adopt an available pet and their form details must be valid (details to be decided aka which home type is required, minimum age needed, etc) for the adoption to be completed. Their information will be saved to a file in a folder.

1. **Developing the application \ algorithm.**

Coding in C# for the backend via Visual Studio 2022 as well as WPF for the front end.

1. Create a pet class which holds information about a pet such as name, type, age, adoption status etc.
2. Create a pet database which will hold a list of the pets in the adoption center.
3. Create a main window that will allow the user to see pets from the database 4 at a time.
4. Create a randomizer so 4 random pets from the database are chosen and shown on the main window.
5. Create a pet detail page that will be shown when the user clicks on a pet to show extra information about the pet.
6. Create a filter to view the adopted and available pets.
7. Create an adopt button on the pet detail page that will allow the user to navigate to the adoption form.
8. Create an adoption form that will ask for information such as the user’s name, address, date of birth, etc.
9. Update the pet’s status to “adopted”.
10. **Implementing the application \algorithm.**

All code was implemented locally and runs locally.

1. **Testing.**

More rigorous testing to follow, testing by examining cases and trying different actions the user can do. Adding validation to form submission

OOP Design

Talk about the classes you need to create for the application and what is the purpose of each class. Include the UML class diagram in this section. The UML class diagram should include the relations between the created classes. Do not mention the WPF classes (Window, etc.)

**Pet class:** This class contains the backing fields and methods to create Pet objects. The user is never able to access the pet’s backing fields.

**Pet Database class**: Stores the Pet objects in a file and retrieves them from the file. When retrieved, the pets are stored in an array. Acts as a storage for the pets and their information (which is then passed to other classes to be utilized).

**Adopter class:** This class contains the backing fields and methods to create Adopter objects. Via the Adoption form submission, the user can add to the Adopter class. Links

**Adopter database class:** Stores the Adopter objects in a file and retrieves them from the file. Links the Adopters with their adopted pets in the file. Acts as a storage for the Adopters and their information (which is then passed to other classes to be utilized).

Contributions

**What did each team member do?**

|  |  |
| --- | --- |
| **Member** | **Work** |
| **Taryn** | * WPF design for all pages * adoption form and class * saving adoptee info to a file * updating pet status * submission popup * code improvements across multiple classes * summaries + comments * code organization/making it more modular to fit OOP * team document * report of all adopted pets in the database * 3 filters (show all, show adopted, show available) * Validation on the adopter form * Creating starting files for the project (all files in ./ ) |
| **Hana** | * WPF base code design for the pet windows * Pet class * refreshing and saving status of pets * pet details window * pet database class * adopter class * reading from file and showing adopter’s and their adopted pets |

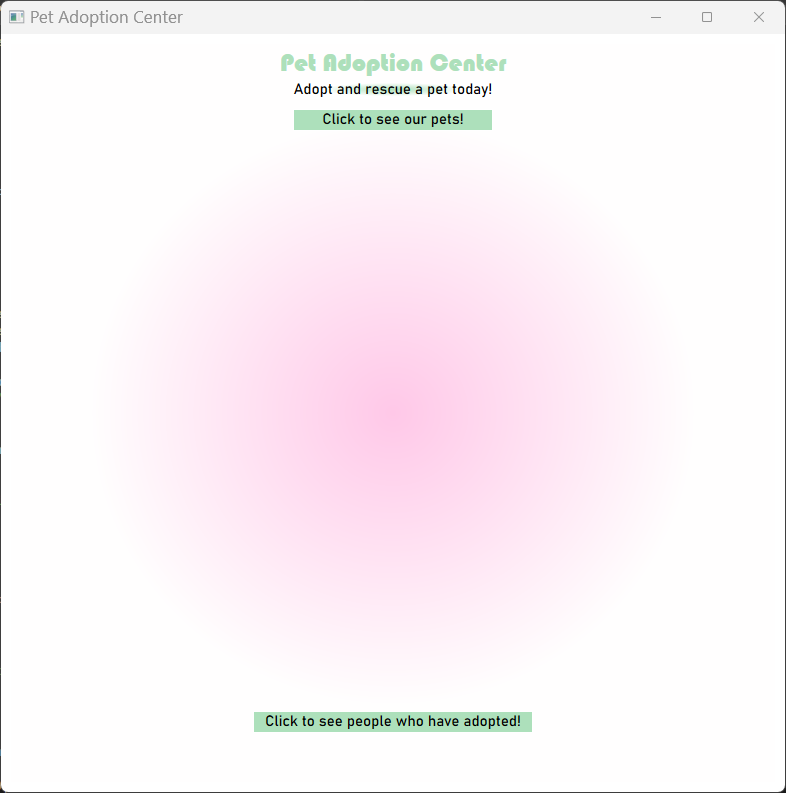
**How was the work in the project divided?**

It was decided early on that the work would be split in half (which was decided as the pet portion and the adoption portion) and the WPF was also split evenly. All pet icons are hand drawnas well.

App Snapshots

**The Main Window**

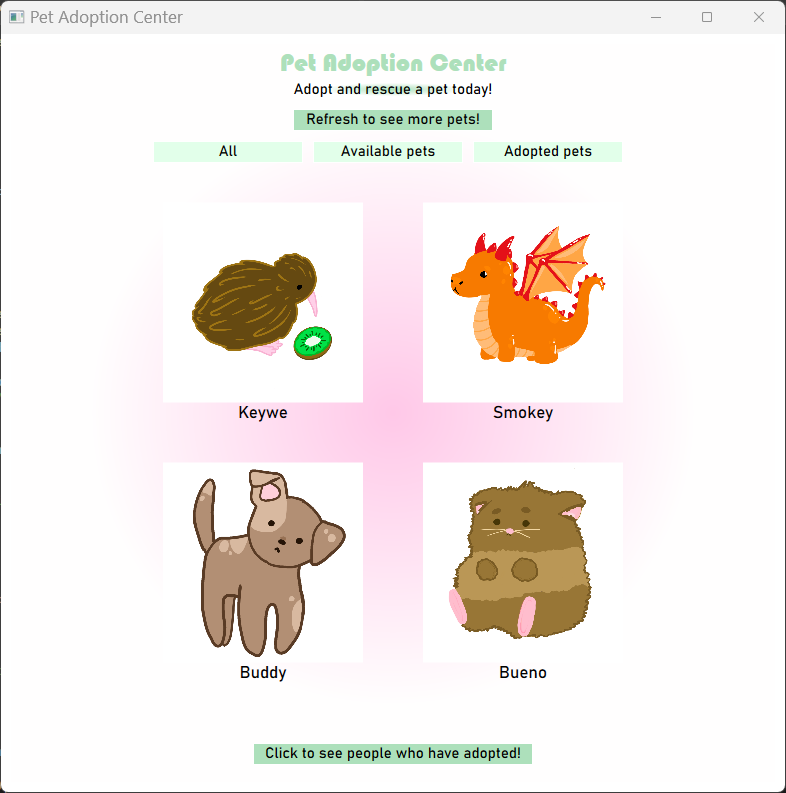
Appears on initial launch and is the first window displayed to the user. User can choose to access the pet center when they click on the “Click to see available pets” button



**Pet page**

This page appears after the user clicks the first button. It shows the user 4 randomized pets. The user can then select a pet and it will open the Pet Details window. They can also choose to refresh the page to get another 4 random pets.

**3 further options available to filter the pets being displayed.**

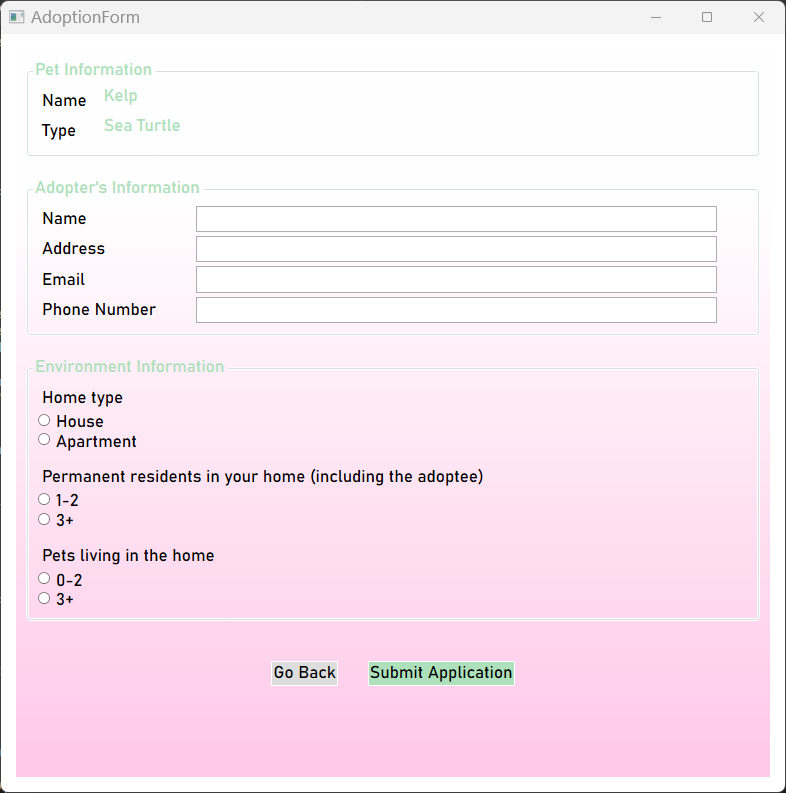


**Pet details page**

This page appears after the user selects a pet from the pets page. 

**Adoption form**

Appears when a user decides they would like to adopt the pet and selects the button “Adopt” on the pet details

****

**Submission popups:**

A screenshot of a computer

Description automatically generated



Future Work

* Allow the user to put up a rescued pet and register the pet to be available for adoption
* Bonded pets/dual adoption (where pairs of pets have bonded and would prefer to be adopted together)
* Mini pet shop where the user can select some items to buy alongside their adoption/make a donation
* Adoption fees + receipts. Having a total of all the money raised by the adoption center.

Appendix A: Team Contract

Team Contract

Appending A: Team Contract Template

# Team Members (Name & ID)

|  |  |  |
| --- | --- | --- |
|  | Name | Student Id |
| Member A | Hana Louiza Moussaoui | 2275881 |
| Member B | Taryn Beaupre | 2173710 |

# Strength & Weaknesses

* Member A:

*Strengths*: persistence, debugging skills, attention to detail

*Weaknesses*: Sometimes reluctant to trying new methods/ways of programming things.

* Member B:

*Strengths*: Debugging, creativity, organized and efficient code

*Weaknesses*: Can get ahead of myself and start multiple sections without looking at the bigger picture.

# Definition of “good enough”

An application that achieves its basic intended purpose with an aesthetic user interface and that handles errors in an elegant manner.

# Picked Topic

We want to create a virtual pet adoption center with one window displaying available pets where they can see detailed information about a selected pet, including their name, age, and adoption status. Another page will contain the adoption papers that the user can fill out, and then the pet will no longer be available to adopt on the main page.

Or (if first option not accepted)

Quiz trivia game

# Division of work

How will each member contribute to the project?

Member A: Will contribute to approximately half of each component

Member B: Will contribute to approximately half of each component

*\*\* We both intend to actively contribute to each part of the project*

# Frequency of communication

*How often will the team be in touch and what tools will be used to communicate?*

Every few days, but closer to the due date, it will be more like every day. We have multiple ways of communicating and contacting one another (Discord, IMessage, Instagram).

# Receiving feedback

*Each member must provide a sample sentence for how they would like to receive constructive feedback from their peers.*

Member A: “Hi, I would like to discuss \*insert subject\*, I think it could be improved in this way… etc”

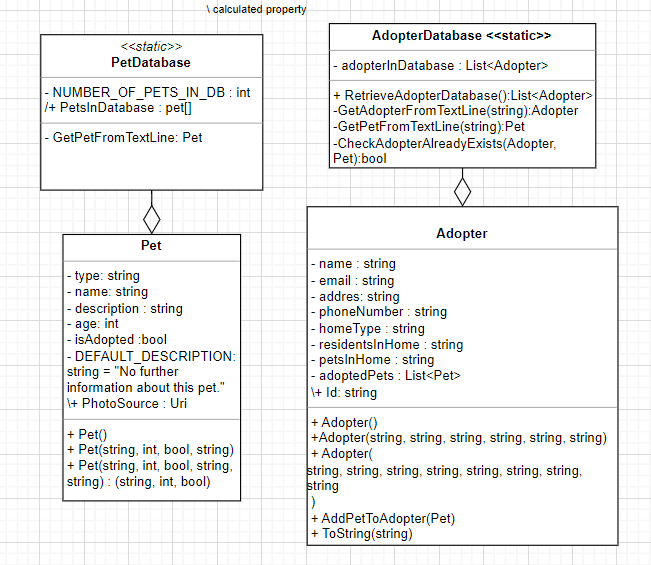
Member B: “Hey I would like to propose an alternative method to do \*insert subject\*, what do you think of doing it this way instead? \*insert new method\* I think this part just needs to be improved to do this, etc”

# In case of conflict

*If a team member fails to communicate as described in this contract or does not respond to constructive feedback, what measures should the other teammate take?*

They can try reassigning the current task at hand and remind the other of the expectations described in this contract. They can organize a little meeting to communicate openly about any concerns they have. Worse comes to worse, they can involve a mediator or escalate the issue to Aref.

Appendix B: UML Class Diagram UPDATE ME



* DO NOT PLACE A  LINK TO THE DIAGRAM.
* Do not include WPF created classes in the class diagram.
* The diagram should be placed in the document.