Software Engineering and Testing. BSC Year 2, 2024/2025
(Assignment 3 - 20%)

Assessment 3: Design and Draft Implementation

Submitted by: Likhitha Atmakuri B00165167, Isabella Higgins B00162371, Sean Kelly B00147339

Submission date: 24/03/25

Declaration

I herby certify that this material, which I now submit for assessment on the programme of study leading to the award of Ordinary Degree in Computing in the Institute of Technology Blanchardstown, is entirely my own work except where otherwise stated.

Author: Likhitha Atmakuri Dated: 23/03/25

Author: Isabella Higgins Dated :23/03/25

Author: Sean Kelly Dated :23/03/25

Table of Contents

Title: Pawsible

- Pawsible is an interactive pet adoption website aimed at connecting adopters with pets in need of homes. Users can create profiles, browse available pets, and submit adoption applications. The system includes features such as user authentication, pet search filters, and admin-controlled adoption approval. The goal is to simplify the adoption process and reduce pet abandonment rates.

1. Project Definitions

- The purpose of this document outlines the design and implementation of how Pawsible works, including functional specifications, methodologies, system architecture, and models that all support the core development of this project.

- Pawsible is a web-based platform that enables adopters to find pets based on various filters such as breed, age and location. Admins manage adoption requests and maintain pet listings.

Functional specifications for Pawsible:

- User Registration: Users must create accounts to adopt pets.
- Pet Search and Filtering: Users can search for pets of their choice
- Adoption Request Management: All pet and user data are stored securely
- Database Management: All data about pets and users are stored securely
- Donation System: Users can donate to Pawsible.

Main components of the software system

- Frontend (User Interface): Designed using HTML, CSS, and JavaScript to provide a seamless user experience.
- Backend (Server & Business Logic): Developed using PHP to handle data processing and server-side logic.
- Database (MySQL): Stores user profiles, pet listings, adoption applications, and donation records.

2. Methodology

- This project adopts Methodology, by promoting continual progress and ongoing feedback to enhance system adaptability. Additionally, Object-Orientated Analysis and Design (OOAD) is used to efficiently structure the system's architecture.

- OOAD is used to design in a structured manner by defining objects, their attributes, and behaviours.

- Classes serve as blueprints for objects in an object-oriented system, some of their purposes are:

- **Encapsulation**: Keeps data and functions together, making the system more secure and manageable.

- **Reusability**: Allows code to be reused, reducing duplication and saving development effort.

- **Modularity**: Breaks the system into smaller parts that can be developed and updated independently.

- **Scalability**: Makes it easier to add new features and expand the system.

- **Flexibility**: Adapts easily to changes and new requirements.

- A class diagram is a type of UML diagram, that represents the structure of a system by showing its classes, attributes, operations and relationships. It helps visualize how different components interact and ensures proper system design.

- **Static Case Diagrams –** represents the structural aspects of a system, focuses on system components and their relationships, showing how different arts interact without considering any changes over time.
- **Dynamic Case Diagrams-** represents the behaviour of a system, showing how objects interact and change over time.

- An ERD is a visual representation of a system's data and how different entities within that system interact with each other. It helps the database design by illustrating relationships, attributes and cardinalities between entities.

- **One-to-One (1:1)**: One entity is linked to exactly one other entity.
- **One-to-Many (1:M)**: One entity is linked to multiple entities.

- **Many-to-One (M:1)**: Many entities are linked to one entity.

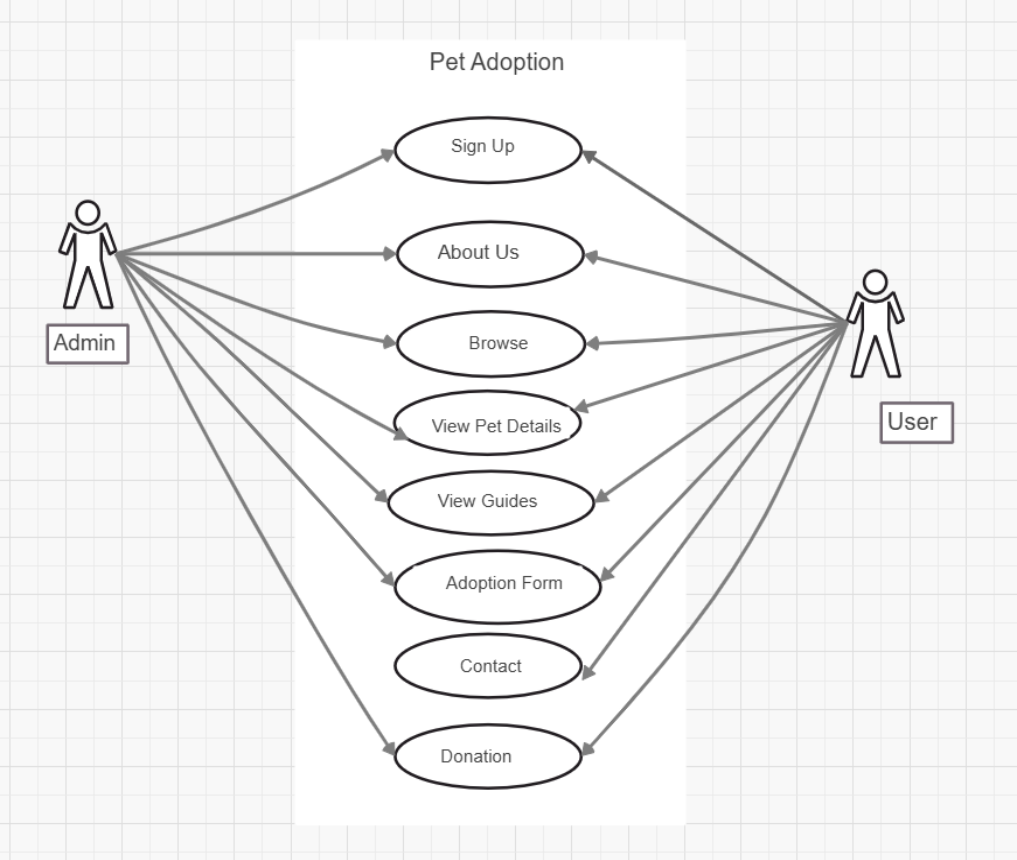
- **Many-to-Many (M: N)**: Many entities are linked to many other entities.

- **Volatile Storage -** temporary storage and loses data when system is powered off. Object instances in RAM store temporary user sessions, search

results, and cached data for faster processing. This type of storage is useful for enhancing performance but does not retain data permanently.

- **Persistent Storage –** retains data even when system is turned off, unlike volatile storage, persistent storage ensures long-term data availability and consistency.
  User Interface template chosen and how it can aid in executing the functional specification of the project.

3. Requirements 4.1 Use Cases



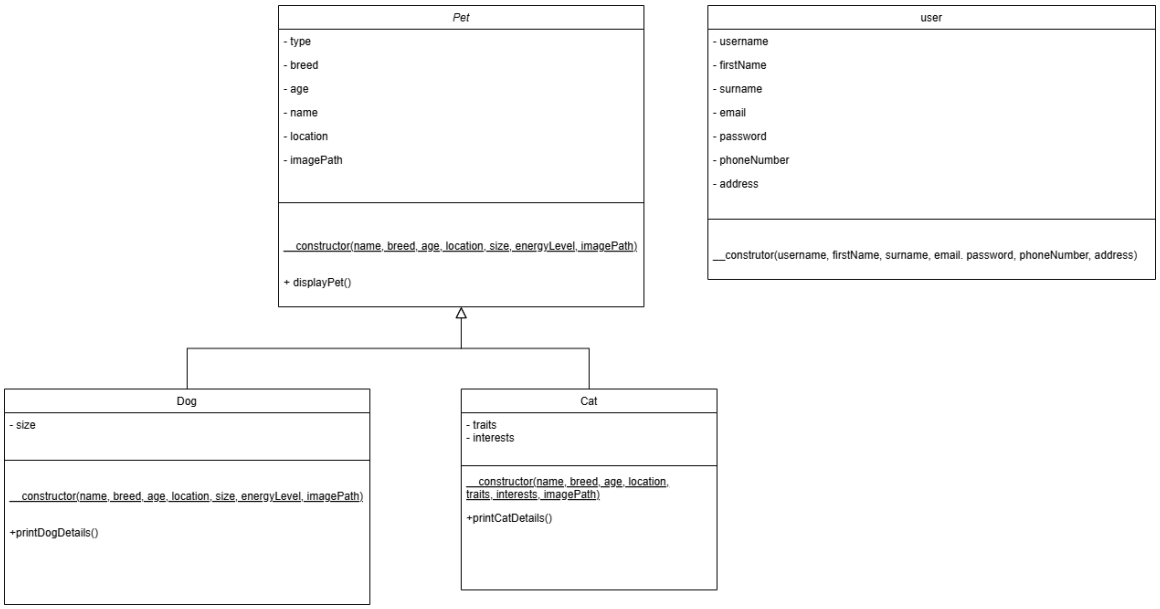
4.2 Use Case Specifications

User Registration	• Users enters personal details (name, email, password) • Systems validates input (checks email format, password strength) • If valid, creates an account and stores data. • If invalid or already registered, shows error or recovery options

Browse	• Users enters search criteria (breed, age, location) • System finds and filters matching pets • Users selects a pet profile to view details •
View Pet Care Guides	• User selects a pet care guide • System displays the selected guide
View Pet Details	• Users clicks on a pet profile • System displays pet details
Submit Request	• User fills out personal details and message. • System validates input and submits the request • Confirmation message displayed
Submit Adoption Application	• User selects a pet and clicks “Adopt” • Users fills out the adoption form • System validates input and stores the application • Admin reviews the application
Donation	• User selects donation amount • System displays payment form • User enters payment details and confirms • System processes payment and confirmed

- Use case specifications define the system's functionality by describing user actions and system responses. These specifications guide the development of classes, their attributes, and methods by identifying the data and operations needed to fulfil user requirements. They also inform the design of database tables by outlining the entities that need to be stored and the relationships between them. Essentially, use cases provide a blueprint for translating user interactions into system components, ensuring the system is both efficient and aligned with user needs.

4. Case Diagrams



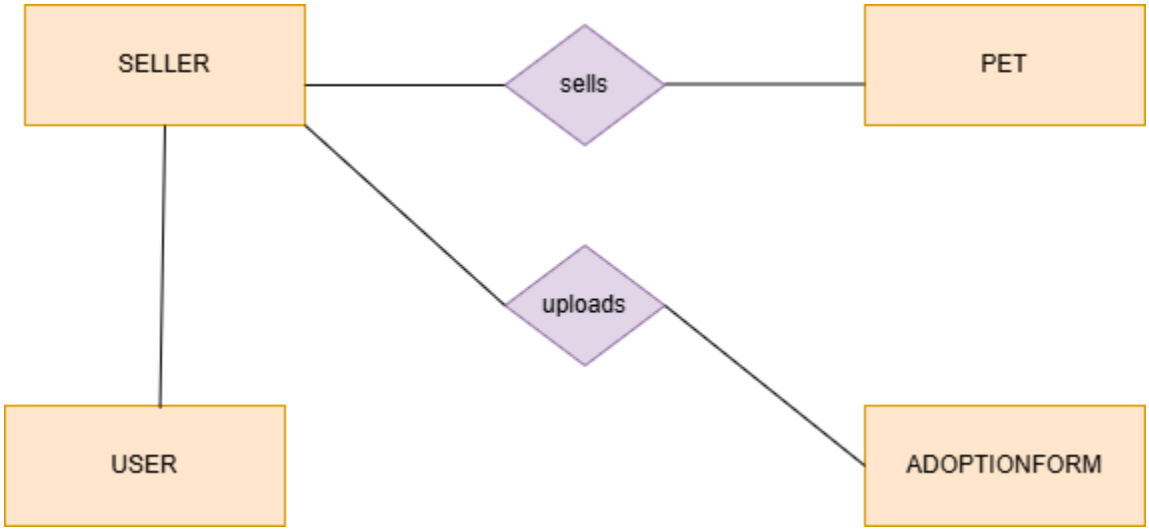
The class diagram shows the object orientated design of the platform. It has a class for the user, pets and the types of pets.

- Inheritence
  The pet class is the parent class. It has the common attributes such as type, breed, age, name, location and imagePath. It also has a method which displays the pet's details. The Dog and Cat class inherit Pet aswell as having their own attributes. The Dog class has two more attributes, size and energyLevel. It also contains a function to display the dog's details. The Cat class does the same, it has two more attributes which are traits and interests. It has a function to display the Cat's details.

- User

The user class has attributes specific to account creation such as their username, firstName, surname, email, password, phoneNumber and address.

CONCEPTUAL DIAGRAM



The conceptual diagram gives us an overview of the entities in the system and their relationships without focusing on the attributes yet.

User -> Represents the users on Pawsbile who are both adopters and sellers. The user table is linked to the seller table because some users will be sellers.

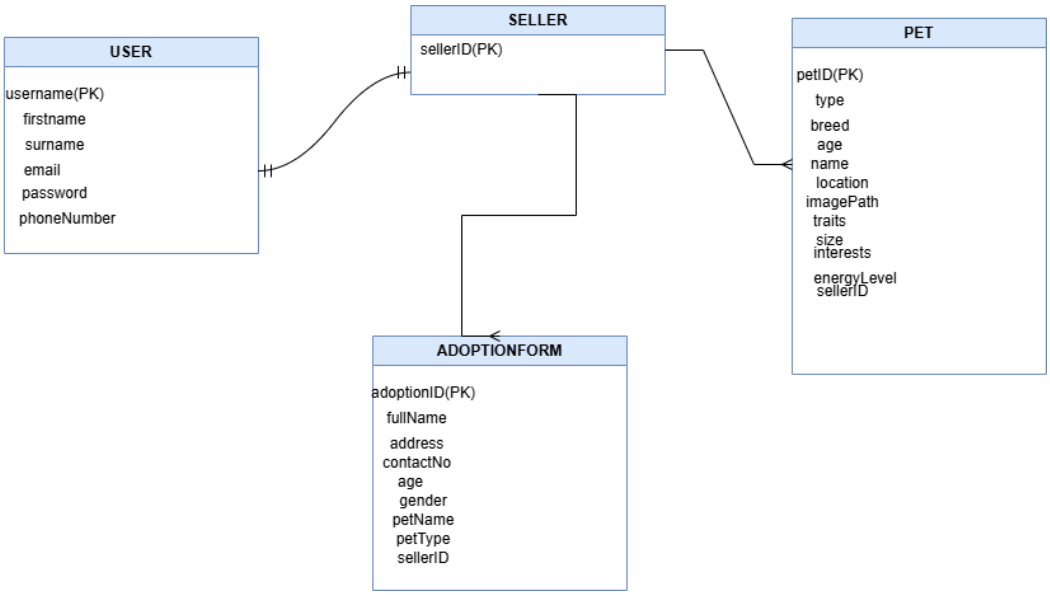
Seller -> A specific type of user who lists their pets for adoption on Pawsible. The seller table is linked to the user table because all sellers are a user.

Pet -> The pet table represents the pets that will be listed for adoption on pawsible. The pet table is linked to the seller because they sell the pets.

AdoptionForm -> Represents the adoption forms that are submitted by the users. It will store the details of the user and pet they hope to adopt.

LOGICAL DIAGRAM

The logical diagram expands the conceptual diagram by adding the attributes to the tables and addressing their primary keys.



- **USER**

This table represents all of the users on Pawsible, both adopters and sellers. The primary key is username. The attributes in this table are firstName, surname, email, password and phoneNumber. The user table has a one to one relationship with the seller table because a user can be a seller.

- **SELLER**

The seller table is a specific type of user who lists their pets for adoption to be adopted by users. The primary key is the sellerID and the foreign key is username. The seller table has a one to many relationship with the Pet table because a seller can list many of their pets for adoption.

- **PET**

The pet table represents the pets who are available for adoption. The primary key is the petID and foreign key is sellerID. The attributes of this table are type, breed, age, name, location, imagePath, traits, size, interests and energyLevel. The pet table has a many to one relationship with the seller table because each pet belongs to one seller.

- **ADOPTIONFORM**

This table represents the adoption forms from users. The primary key is adoptionID and foreign key is sellerID. The attributes are fullName, address, contactNo, age, gender, petName and petType. The adoptionForm table has a many to one relationship with the seller table because each adoption form is for a specific seller.

5. Conclusions

In conclusion, the use case specifications for Pawsible are key in shaping the system's design. They guide the creation of use case diagrams, which show how users interact with the system, like adopting pets or making donations. Class diagrams are created based on these use cases to define the system's structure, showing what data is stored and how it works. ERDs help organize how different parts of the system, like pets and users, are connected in the database. Together, these diagrams ensure that the system is well-organized and easy to use for both pet seekers and administrators.