**Ingegneria del Software e Progettazione Web**

**Progetto A.A. 2021/2022**

**MyPet**

**0278853**

**Matteo Basili**

**1. Software Requirement Specification**

1.1. Introduction

1.1.1. Aim of the document

The main objective of this document is to guide the reader in the development of the "MyPet" software application, starting from the analysis of the requirements up to the implementation and testing, showing the design steps that are undertaken for the realization of the system.

1.1.2. Overview of the defined system

MyPet wants to be the point of reference for the custody, maintenance and care of pets in Italy. The purpose of the app is to help people who find it difficult to manage their puppies, allowing them to post adoption announcements that can be seen by anyone who wants to make a new furry friend, thus supporting the no abandonment of animals. The system also allows you to find pet sitters (with their contacts), vets and pet shops nearby, for a complete overview of the world in question.

There are many different applications and sites on the Internet about pet adoption or pet sitting. Nowadays to find a vet or a pet shop just do a little search on Google and there in a few seconds we can find what we want. However, having a system at hand that encompasses all of this in a single app is not very common.

1.1.3. Operational settings

The application is designed to run on android phone systems targeting an API level ranging from 21 to the latest (currently 33).

1.1.4. Related systems, Pros and Cons

Two applications to be considered similar to MyPet are, for example, "ZampyLife" and "Appets".

Compared to MyPet, ZampyLife has additional features such as the "report abuse" section, the veterinary SOS and the possibility of creating a health card and sharing it with the vet. Appets, on the other hand, has a "Lost Pets" section that we don't find in either of the other two systems.

What distinguishes MyPet from these is the ability to also find pet shops, as well as having a more refined search system (for adoptions compared to ZampyLife and for pet sitters compared to Appets).

1.2. User Stories

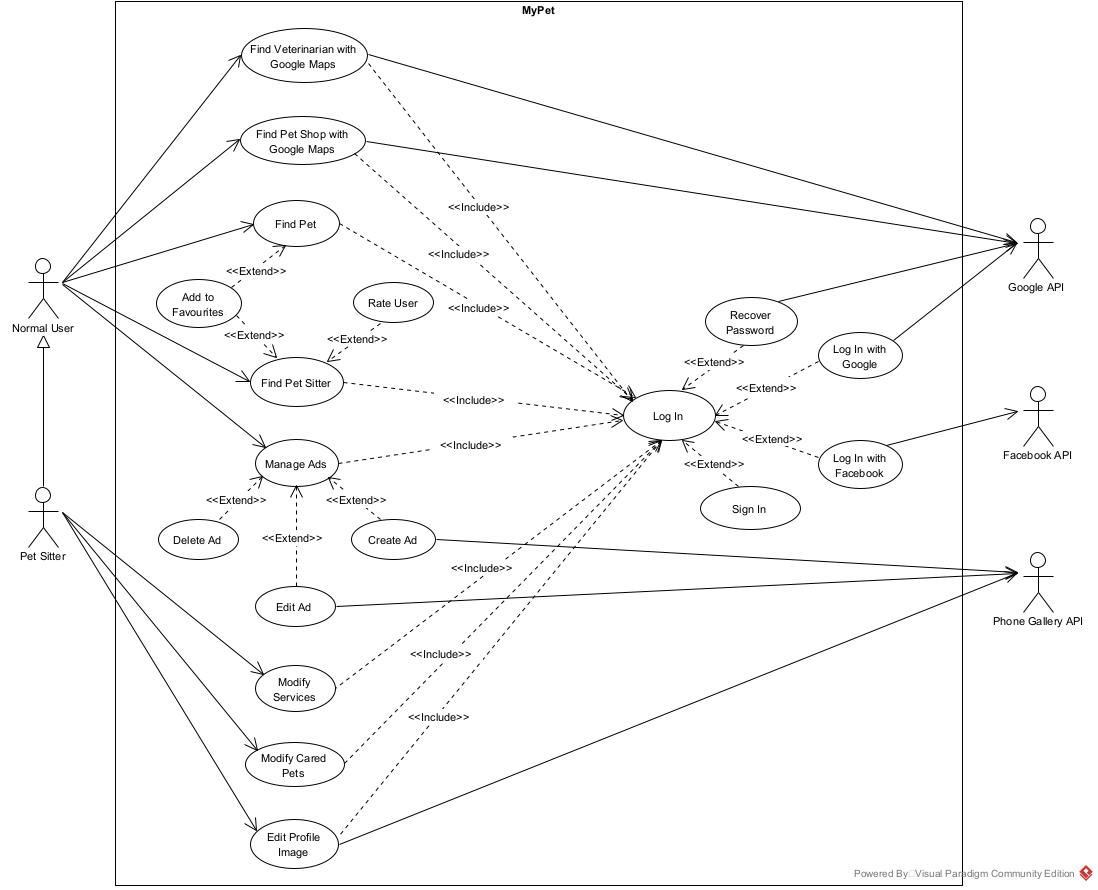
1. As a new user, I want a mini tutorial, so that I can easily learn how to use the application.
2. As a user looking for a pet sitter, I want to be able to add a pet sitter's profile to my favorites, so that I can consult it more quickly at another time.
3. As a pet sitter, I want to see my number of likes and dislikes, so that I can understand what people think about me.

1.3. Functional Requirements

1. The system shall provide the registration as normal user and the registration as pet sitter.
2. The system shall provide a map for searching veterinarians and pet shops.
3. The system shall provide the recovery of the account password by sending the password to the e-mail of the account in question.

1.4. Use Cases: Overview Diagram

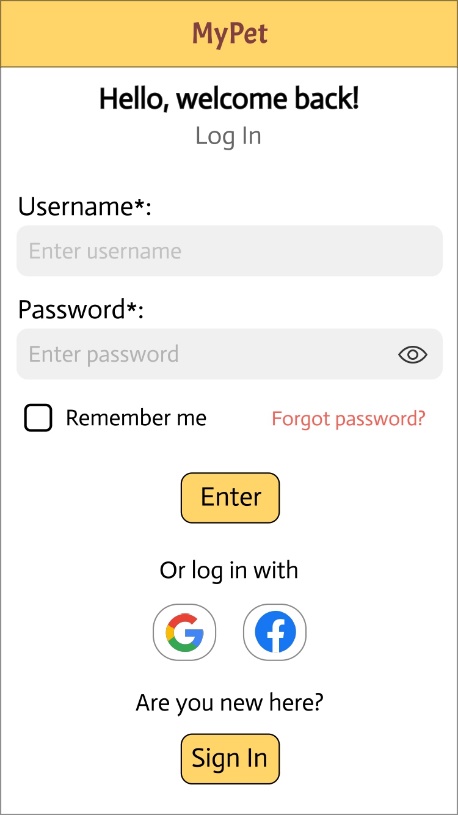
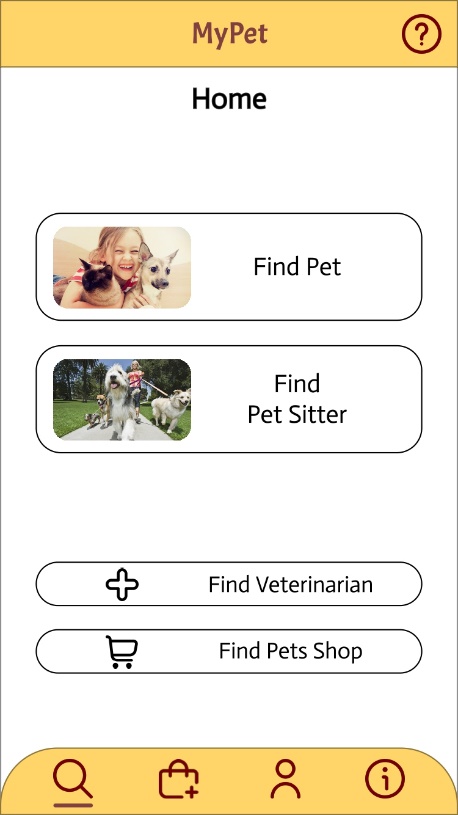
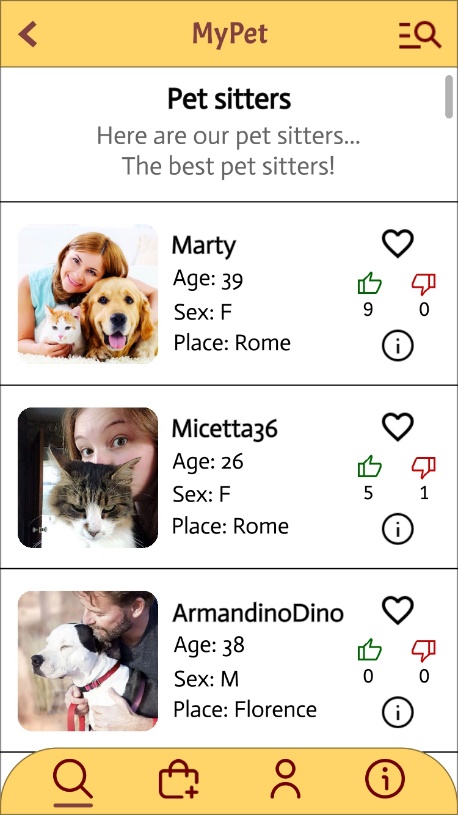
Figure 1.1 shows the diagram of the use cases for the application in question.

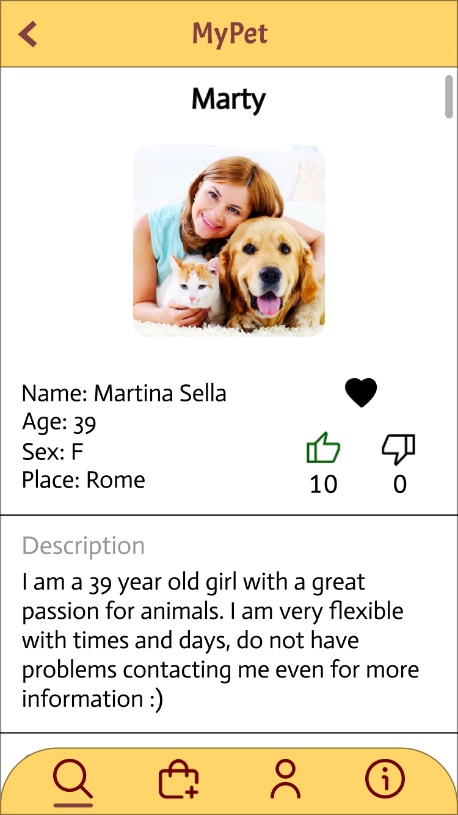
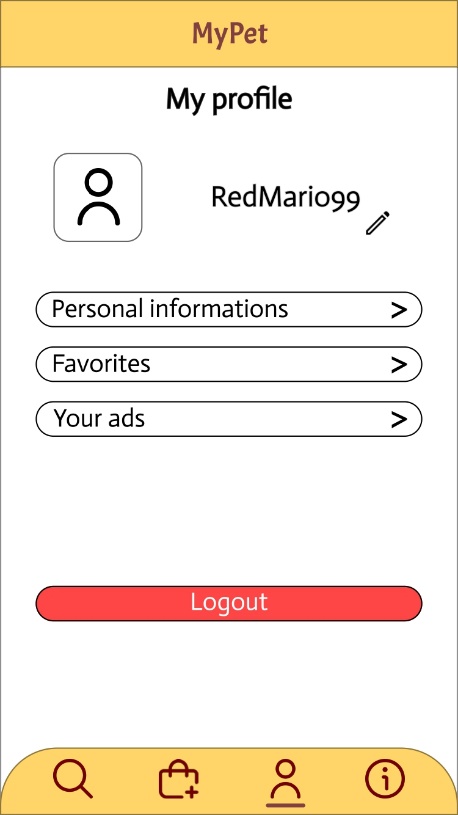
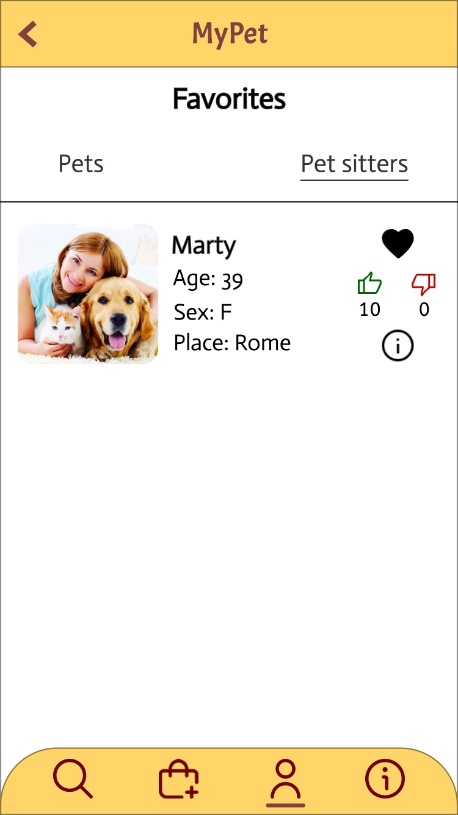


**Figure 1.1** - Use case diagram.

**2. User Interface Prototypes**

Figure 2.1 shows six screens of application interface prototypes.

**  **

**  **

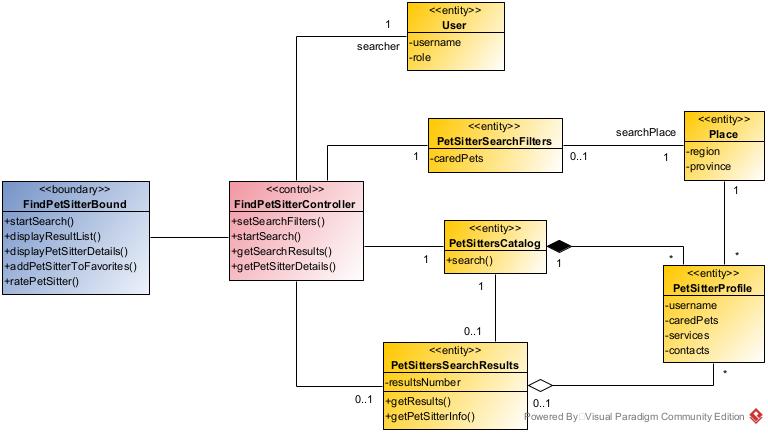
**Figure 2.1** - UI prototypes.

**3. Design**

3.1. Class diagram

3.1.1. VOPC

Figure 3.1 shows the view of the participating classes (VOPC), according to the ECB (Entity-Control-Boundary) analysis model, for the "Find Pet Sitter" use case.

****

**Figure 3.1** - VOPC.

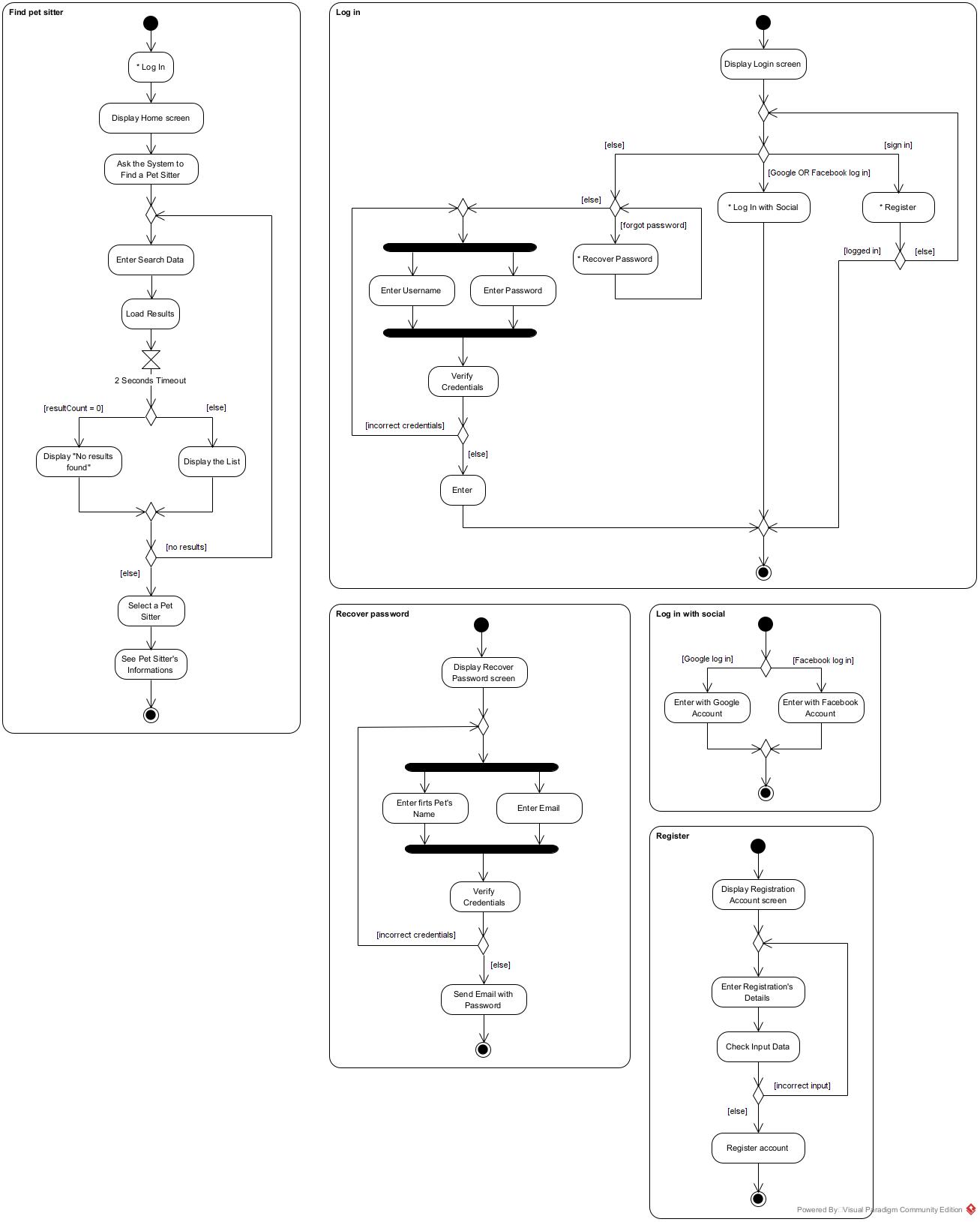
3.1.2. Design-level diagram

Figure 3.2 shows the DCD (Design Class Diagram) of the "Find a Pet Sitter" use case as a refinement, according to the MVP (Model-View-Presenter) architectural pattern, of the previous analysis diagram.

Factory Method 🡪 ProvincesSpinner

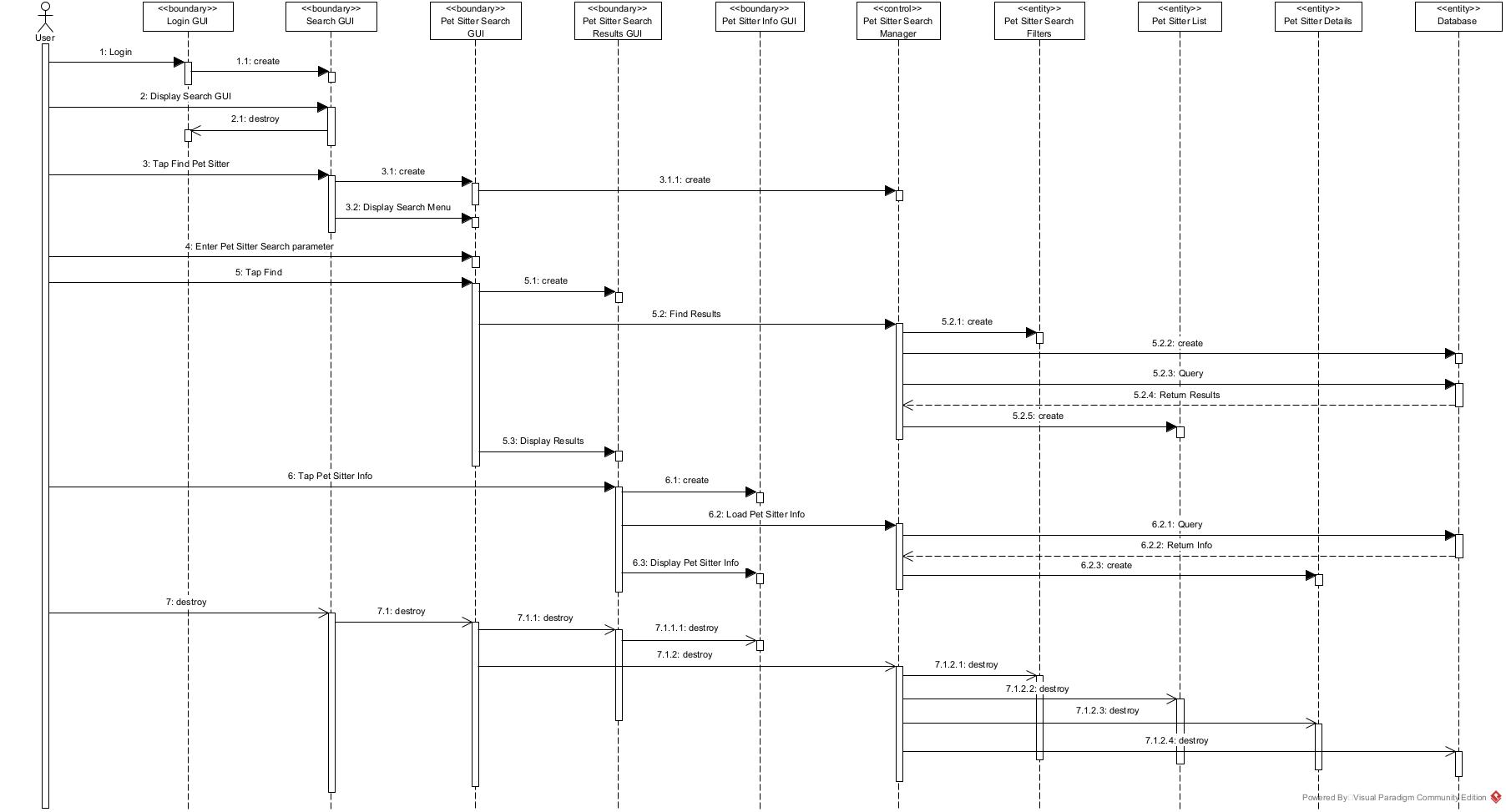
Singleton 🡪 DBConnection

3.3. Activity diagram (Use Case: “Find Pet Sitter”)

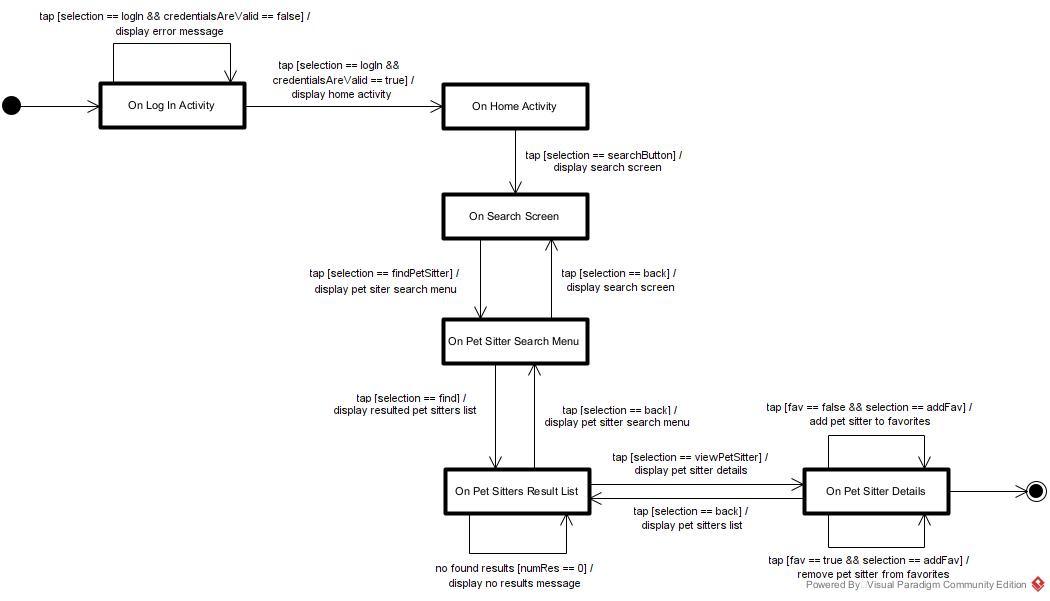
****

**Figure 3.3** -

3.4. Sequence diagram (Use Case: “Find Pet Sitter”)

****

3.5. State diagram (Use Case: “Find Pet Sitter”)

****