
Petopia
Fur-ever care, Forever friends!

Project title: Petopia

Slogan: fur-ever care, forever friends!

project Statement

Our mission is to empower pet owners with the tools and resources they need to provide optimal care for their beloved pets, ensuring their health, happiness, and longevity.

To be the go-to solution for pet owners, offering a seamless and intuitive platform for managing all aspects of their pets' well-being, from healthcare to daily routines.

Main project problem

Problem Statement: pet owners are facing increasing challenges in Finding reliable pet care services and pets social needs, It becomes time-consuming tasks and needs a lot of effort, Many pet owners may struggle to take care of their pets due to lack of knowledge and awareness, Difficulties in accessing affordable and reliable veterinary care, And Finding trustworthy and suitable pet services.

Sub-problems

Difficulty in Managing Medication Schedules and pet appointments.

Time-Consuming Search for Reliable Pet-Sitting Services and Matchmaking.

Financial Constraints in Accessing Premium Pet Care Services.

Limited Availability of Pet Care Resources in Remote Areas.

lack of finding guidance on pet behavior and training.

Main project objective

Our objective is to improve pet care services by connecting pet owners with trusted providers. Focusing on addressing pets' social needs. We aim to make pet care more accessible and affordable, ensuring trustworthy services. Overall, our goal is to empower pet owners to provide optimal care for their pets.

Sub-objectives

The fastest way to provide social Needs of Pets (matchmaking and pet-sitting).

Reminder for Medication Schedules.

Pet appointments and daily tasks.

Enhance Access to Affordable and Reliable Pet Care centers.

Ensure the nearest Trustworthy and Suitable Pet centers and shops.

Resources to learn about your pet behavior and how to train them.

Target Users

Pet Owners: These are the primary users who sign their pets on the app for various services such as pet-matching, petcare or veterinary care.

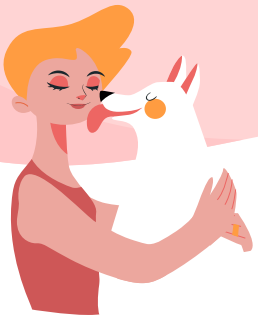
Veterinarians: Professionals who will interact with the app to manage appointments, prescriptions, and health records of pets.

Target Users

Pet service providers: Pet Sitters, Pet Walkers

Pet care centers & shops: Pet Stores, Veterinary Clinics & Hospitals, Pet Daycare Center, Adoption & Rescue Centers

Technologies and Frameworks:



Frontend: (mobile app)
Flutter

Backend: Spring Boot (Java)

Database: MySQL

Expected Design Patterns

- Strategy pattern
- Observer Pattern
- Singleton Pattern
- Factory Pattern

Initial High-Level System Overview

User Module: Authentication, profile management, pet profiles, and booking history.

Pet Service Module: Listings of veterinarians, pet sitters, trainers, and groomers with categories and details.

Service Provider Module: Profiles for vets, sitters, and trainers, including availability and pricing.

Booking Module: Real-time appointment scheduling, service availability, and secure payments.

Notification System: Push notifications, SMS/email reminders for appointments and medication schedules.

Admin Dashboard: Manage users, service providers, bookings, and reviews.

Functional Requirements

1) User Registration & Profile Management – The system shall allow pet owners and service providers to **create, update, and manage their profiles** with relevant details

2) Service Search & Booking – The platform shall enable pet owners to **search for pet care services based on filters like location, availability, and category**, and allow them to **book appointments** seamlessly.

3) Rating & Review System – Users shall be able to **rate and review service providers** after appointments, helping others make informed decisions.

4) Appointment Scheduling – The system shall provide a **calendar-based scheduling feature** for users to **set, modify, and track appointments** with service providers.

5) Messaging & Notifications – The application shall support **real-time**

Non-Functional Requirements

1) Performance – The application must remain **responsive and efficient**, even during peak usage periods.

2) Strong Security – User data, payment transactions, and communications must be **secure and protected** against unauthorized access.

3) Scalability – The system should be **capable of handling an increasing number of users and service requests** without performance degradation.

4) Usability – The interface should be **intuitive, easy to navigate, and provide clear instructions** for a seamless user experience.

5) Reliability – The application must maintain **consistent uptime and functionality**, minimizing technical issues and downtime.

6) Accessibility – The system should meet **accessibility standards**, ensuring usability for individuals with disabilities.

Constrains

1)**Platform Compatibility –** The app must support both **Android and iOS.**

2) **Secure Payments –** Must integrate with **Banks, Vodafone cash, fawry, or Google Pay/Apple Pay** for app transactions.

3) **User Data Privacy –** Must follow **data protection rules (GDPR)**, ensuring secure login and encrypted user data.

4)**Internet Requirement –** The app needs an **active internet connection** for bookings, messaging, and payments.

Assumptions

1) Users have access to the internet – The app requires an active connection for bookings, messaging, and payments.

2) Users own smartphones that support modern apps– The app assumes users have (Android 11+ or iOS 13+) for compatibility.

3) Pet service providers will register and update their availability – The system depends on vets, trainers, and pet sitters (actively using the app).

4)Users will provide accurate pet and appointment details– To avoid service disruptions, users must (input correct information).

Dependencies

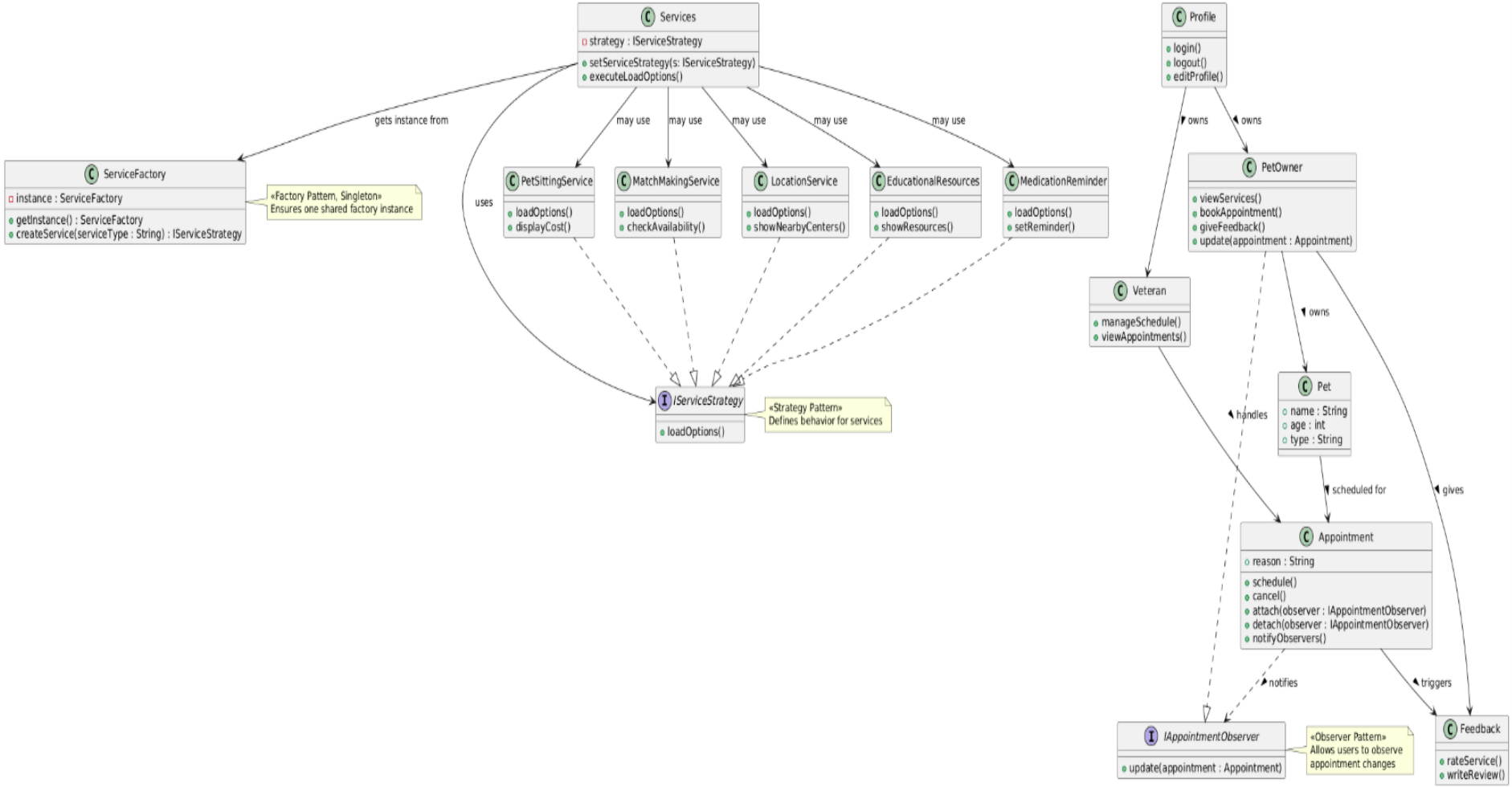
1) Google Play Store & Apple App Store – The app depends on these stores for distribution and updates.

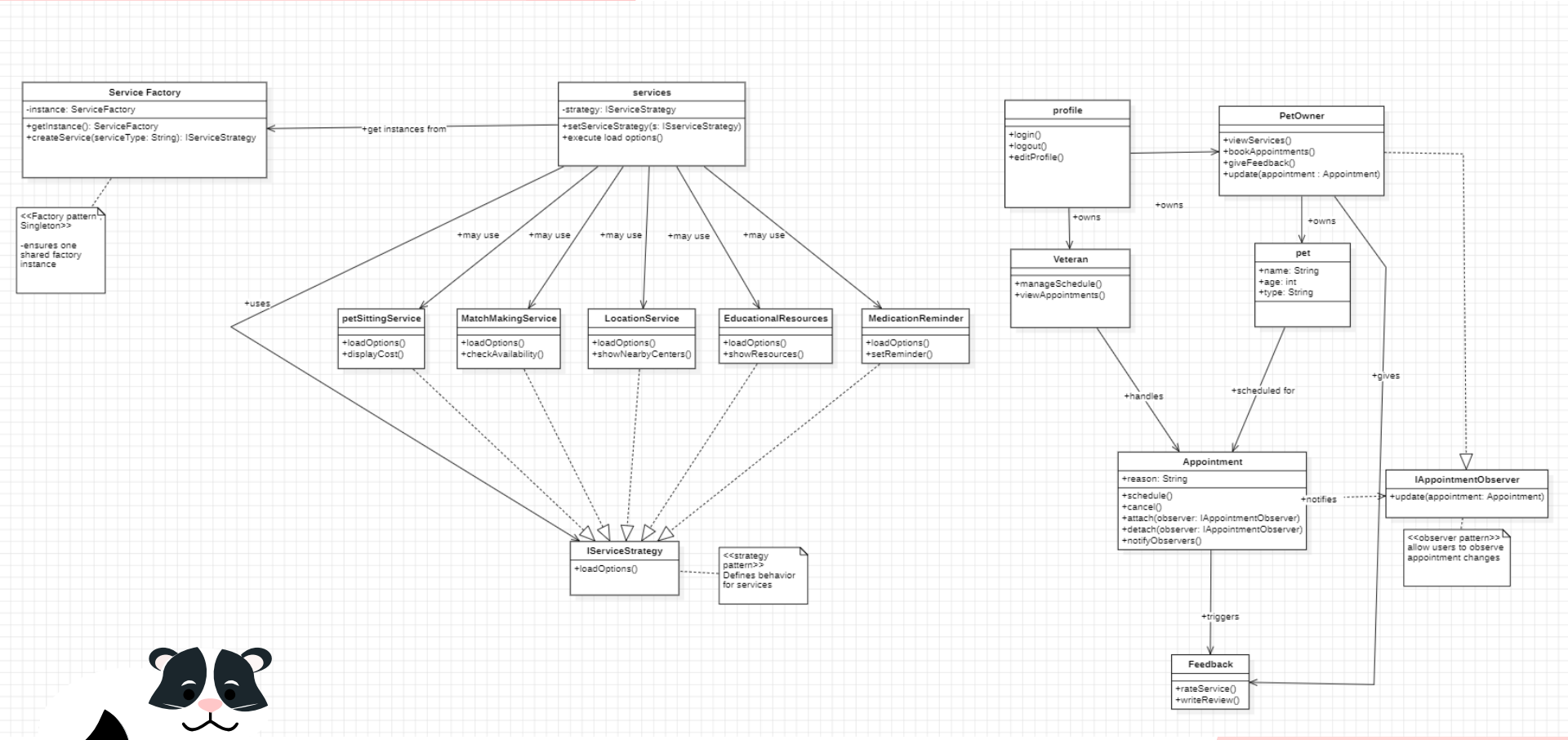
2) Payment Gateways (Banks, Vodafone cash, fawry, or Google Pay/Apple Pay) – For secure transactions when booking pet services.

3) Firebase Cloud Messaging (FCM) – Used for push notifications and reminders.

4) Google Maps API – Required for location-based search (finding nearby vets or pet shops).

Petopia - Class Diagram with Strategy, Factory, singleton and Observer Patterns




Service Factory
-instance: ServiceFactory
+getInstance(): ServiceFactory
+createService(serviceType: String): IServiceStrategy
<<Factory pattern Singleton>>
-ensures one shared factory instance
+get instances from
services
-strategy: IServiceStrategy
+setServiceStrategy(s: IServiceStrategy)
+execute load options()
+may use
+may use
+may use
+may use
+may use
+uses
petSittingService
+loadOptions()
+displayCost()
MatchMakingService
+loadOptions()
+checkAvailability()
LocationService
+loadOptions()
+showNearbyCenters()
EducationalResources
+loadOptions()
+showResources()
MedicationReminder
+loadOptions()
+setReminder()
IServiceStrategy
+loadOptions()
<<strategy pattern>>
Defines behavior for services
profile
+login()
+logout()
+editProfile()
+owns
+owns
PetOwner
+viewServices()
+bookAppointments()
+giveFeedback()
+update(appointment : Appointment)
+owns
Veteran
+manageSchedule()
+viewAppointments()
pet
+name: String
+age: int
+type: String
+handles
+scheduled for
+gives
Appointment
+reason: String
+schedule()
+cancel()
+attach(observer: IAppointmentObserver)
+detach(observer: IAppointmentObserver)
+notifyObservers()
+notifies
IAppointmentObserver
+update(appointment: Appointment)
<<observer pattern>>
allow users to observe appointment changes
+triggers
Feedback
+rateService()
+writeReview()

Explanation of Responsibilities per Class

IServiceStrategy

- (Interface) Defines the standard behavior (loadOptions()) that all service types must implement.
- Used to implement the Strategy Pattern for runtime behavior change.

Services

- Holds a reference to an IServiceStrategy implementation.
- Provides setServiceStrategy() to change behavior at runtime.
- Calls executeLoadOptions() which delegates to the strategy.

PetSittingService, MatchMakingService, LocationService, EducationalResources, MedicationReminder

- Implement IServiceStrategy interface.
- Contain service-specific logic (displayCost(), checkAvailability()).
- Are created via the factory.

Explanation of Responsibilities per Class

Profile

- Base class for all user types.
- Contains login-related methods.

PetOwner
(inherits from Profile)

- Can view services, book appointments, and give feedback.
- Owns Pet objects and creates Appointment.

Veteran (inherits from Profile)
Handles appointments, manages schedules.

Pet

- Represents a pet with name, age, and type.
- Linked to appointments.

Appointment

- Linked to both pet and veteran.
- Triggers Feedback.

Feedback

- Allows users to rate and review a service after appointments.

Explanation of Responsibilities per Class

IAppointmentObserver (Interface)

- Defines an update(appointment: Appointment method.
- Used in the Observer Pattern to notify observers of appointment changes.

ServiceFactory

- Contains createService(serviceType: String): IServiceStrategy.
- Implements the Factory Pattern by generating service strategy objects based on string input.
- Implements the Singleton Pattern by ensuring only one instance of ServiceFactory exists via getInstance() method.
- Promotes object creation management and consistency across the application.

Relationships Between Classes

Inheritance:

- PetOwner and Veteran inherit from Profile.
- PetOwner implements IAppointmentObserver to receive updates from Appointment.

Association:

- PetOwner owns one or more Pet objects.
- PetOwner gives Feedback after appointments.
- Each Pet is scheduled for an Appointment.
- Veteran handles the Appointment.
- Apointment triggers creation of Feedback.

Relationships Between Classes

Dependency:

- Services depends on ServiceFactory (via getInstance() and createService()) to create the appropriate IServiceStrategy.
- Services delegates behavior to an IServiceStrategy implementation (PetSittingService, etc)
- Appointment depends on IAppointmentObserver to notify observers (PetOwner, etc.) about changes, this is part of the Observer Pattern.

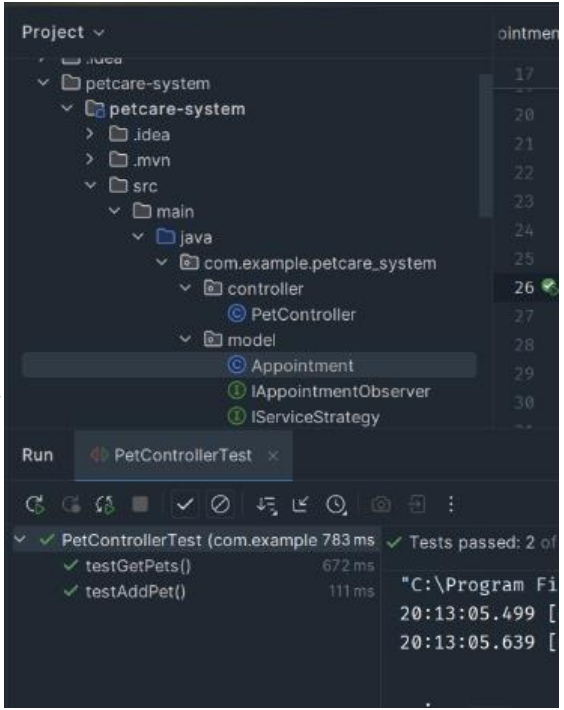
Petopia Unit Testing

1. PetControllerTest

Functionality: Add Pet, Get All Pets

Test Case Name	Tested Functionality	Expected Result	Actual Result	Status
testAddPet	POST /pets	Pet saved correctly	Passed	✅
testGetPets	GET /pets	List of Pets retrieved	Passed	✅

Petopia Unit Testing



Test Description:

POST request is sent to /pets with JSON body representing a Pet. GET request is sent to /pets to fetch all existing pets.

Key Code Sample:

```
mockMvc.perform(post("/pets")
        .contentType(MediaType.APPLICATION_JSON)
        .content(objectMapper.writeValueAsString(pet)))
        .andExpect(status().isOk());
```

2. AppointmentTest

Functionality: Schedule Appointment, Cancel Appointment, Observer Notification

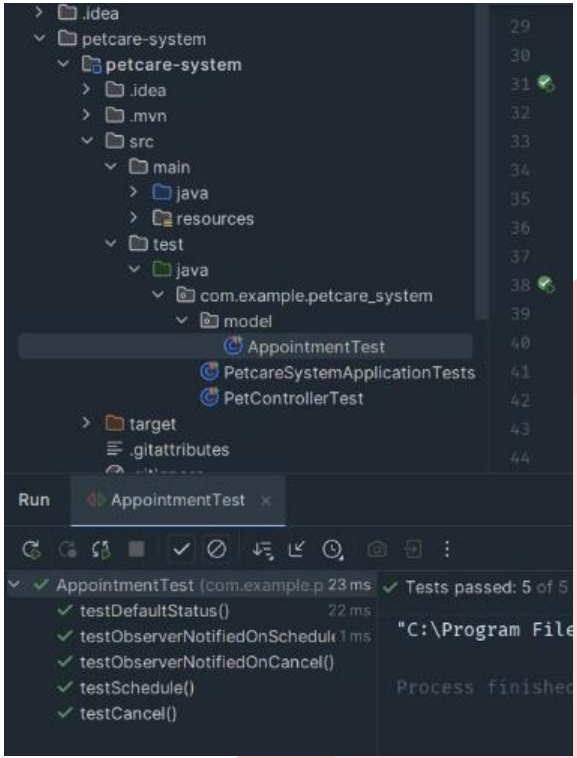
Test Case Name	Tested Functionality	Expected Result	Actual Result	Status
testDefaultStatus	New Appointment default status	Status = Pending	Passed	✅
testSchedule	Schedule Appointment	Status = Scheduled	Passed	✅
testCancel	Cancel Appointment	Status = Cancelled	Passed	✅
testObserverNotifiedOnSchedule	Observer update after schedule	Observer notified	Passed	✅
testObserverNotifiedOnCancel	Observer update after cancellation	Observer notified	Passed	✅

Test Description:

Appointment’s initial state is verified. Observer pattern behavior is tested when appointment state changes.

Key Code Sample:

```
appointment.schedule();
assertEquals("Scheduled", appointment.getStatus());
assertTrue(observer.updated);
```



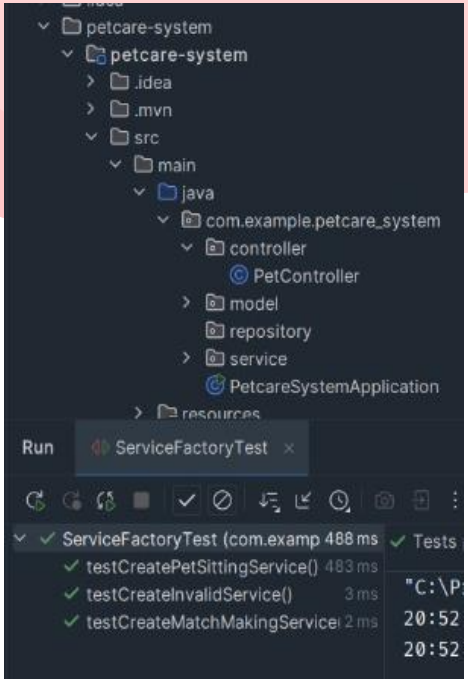
3. ServiceFactoryTest

Functionality: Create Different Service Types, Handle Invalid Input

Test Case Name	Tested Functionality	Expected Result	Actual Result	Status
testCreatePetSittingService	Create PetSittingService	Correct instance created	Passed	✅
testCreateMatchMakingService	Create MatchMakingService	Correct instance created	Passed	✅
testCreateInvalidService	Invalid service type	Returns null	Passed	✅

Test Description:

Valid services are instantiated correctly. Unknown service type returns null.



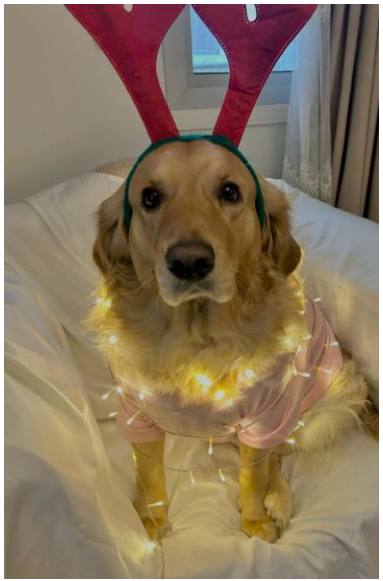
Key Code Sample:

```
IServiceStrategy service =
serviceFactory.createService("PetSitting");
assertNotNull(service);
```

our pets

Maylo

Falco



Honey

THANK YOU!

by
Sama Nabil
Mark Ehab
Batool wesam
Malak wael
Ali el azouny
Youssef Ahmad