Brenden Tran, Cashton Bieker, Christopher Manuel-Smith, Riley Hiltz, Nicholas Panos

Infinite PETS SchedULING SYSTEM

Client: Jenna HallGraphical user interface, logo

Description automatically generated

Instructor: John O’Loughlin

January 31, 2021

Table of Contents

[Contributions 17](#_Toc63015966)

[Preface 19](#_Toc63015967)

[Documentation Standards 19](#_Toc63015968)

[Diagrams 19](#_Toc63015969)

[General Documentation Standards 19](#_Toc63015970)

[User Requirements. 20](#_Toc63015971)

[Problem Statement 20](#_Toc63015972)

[Current System 20](#_Toc63015973)

[Project Scope 20](#_Toc63015974)

[System Environment 21](#_Toc63015975)

[Current System 21](#_Toc63015976)

[System Requirements 22](#_Toc63015977)

[Use Case Diagrams 22](#_Toc63015978)

[Use Case Descriptions and Scenarios 24](#_Toc63015979)

[Use Case 1: Login 24](#_Toc63015980)

[Use Case 2: Create an Account 25](#_Toc63015981)

[Use Case 3: Reset Password 26](#_Toc63015982)

[Use Case 4: Update Account Information 27](#_Toc63015983)

[Use Case 5: Change Password 28](#_Toc63015984)

[Use Case 6: Client Delete Account 29](#_Toc63015985)

[Use Case 7: Client Adds Pet Profile 30](#_Toc63015986)

[Use Case 8: Client Views Pet Profile List 31](#_Toc63015987)

[Use Case 9: Client views a pet’s profile 32](#_Toc63015988)

[Use Case 10: Client Updates a pet’s information 33](#_Toc63015989)

[Use Case 11: Client view’s the available schedule 34](#_Toc63015990)

[Use Case 12: Client books an appointment 35](#_Toc63015991)

[Use Case 13: Client views their upcoming appointments 36](#_Toc63015992)

[Use Case 14: Client updates an appointment 37](#_Toc63015993)

[User Case 15: Client Cancels Appointment 38](#_Toc63015994)

[User Case 16: Client Views Client Contract 39](#_Toc63015995)

[User Case 17: Client Prints Client Contract 40](#_Toc63015996)

[User Case 18: Staff Sets Work Preferences 41](#_Toc63015997)

[User Case 19: Staff Views Available Appointments 42](#_Toc63015998)

[User Case 20: Staff Views Client and Pet Information 43](#_Toc63015999)

[UserCase 21: Staff Accepts Available Appointment 44](#_Toc63016000)

[User Case 22: Send Confirmation Notice and Client Contract 45](#_Toc63016001)

[User Case 23: Staff Views Appointment Updates 46](#_Toc63016002)

[User Case 24: Staff Add Client Notes 47](#_Toc63016003)

[User Case 25: Staff View Scheduled Appointments 48](#_Toc63016004)

[Use Case 26: Admin adds a new staff member 49](#_Toc63016005)

[Use Case 27: Admin removes a staff member 50](#_Toc63016006)

[Use Case 28/29: Admin updates a staff member’s qualifications 51](#_Toc63016007)

[Use Case 30: Admin updates the current promotions 52](#_Toc63016008)

[Use Case 31: Admin emails the current promotions to the email list. 53](#_Toc63016009)

[Use Case 32: Admin adds a service to the system. 54](#_Toc63016010)

[Use Case 33: Admin removes a service from the system. 55](#_Toc63016011)

[Use Case 34: Admin updates or changes a service in the system. 56](#_Toc63016012)

[Use Case 35: Admin changes the hours of operation or schedule time 57](#_Toc63016013)

[Use Case 36: Admin is notified of an unaccepted appointment after 24 hours of the appointment being submitted. 58](#_Toc63016014)

[Use Case 37: Generate a report. 59](#_Toc63016015)

[Use Case 38: Export generated report. 60](#_Toc63016016)

[Use Case 39: Print generated report. 61](#_Toc63016017)

[Use Case 40: Pay for Appointments. 62](#_Toc63016018)

[Functional Requirements 63](#_Toc63016019)

[Client 63](#_Toc63016020)

[Staff 63](#_Toc63016021)

[Non-Functional Requirements 63](#_Toc63016022)

[Client 63](#_Toc63016023)

[System Interface Requirements 65](#_Toc63016024)

[Maintainability and Administration Requirements 65](#_Toc63016025)

[Maintainability 65](#_Toc63016026)

[Administration 65](#_Toc63016027)

[Usability Requirements 67](#_Toc63016028)

[Client 67](#_Toc63016029)

[Staff 67](#_Toc63016030)

[User Interface 67](#_Toc63016031)

[Administrator/Owner 67](#_Toc63016032)

[Problem Domain Class Diagram 68](#_Toc63016033)

[Sequence Diagrams 69](#_Toc63016034)

[Use Case 1: Login 69](#_Toc63016035)

[Use Case 2: Create an Account 70](#_Toc63016037)

[Use Case 3: Reset Password 71](#_Toc63016038)

[Use Case 4: Update Account Information 72](#_Toc63016039)

[Use Case 5: Change Password 73](#_Toc63016040)

[Use Case 6: Client Delete Account 74](#_Toc63016041)

[Use Case 8: Client Views Pet Profile List 76](#_Toc63016042)

[Use Case 10: Client Updates a pet’s information 78](#_Toc63016043)

[Use Case 11: Client view’s the available schedule 79](#_Toc63016044)

[Use Case 12: Client books an appointment 80](#_Toc63016045)

[Use Case 13: Client views their upcoming appointments 81](#_Toc63016046)

[Use Case 14: Client updates an appointment 82](#_Toc63016047)

[Use Case 15: Client Cancels Appointment 83](#_Toc63016048)

[Use Case 16: Client Views Client Contract 84](#_Toc63016049)

[Use Case 17: Client Prints Client Contract 85](#_Toc63016050)

[Use Case 18: Staff Sets Work Preferences 86](#_Toc63016051)

[Use Case 19: Staff Views Available Appointments 87](#_Toc63016052)

[Use Case 20: Staff Views Client and Pet Information 88](#_Toc63016053)

[UseCase 21: Staff Accepts Available Appointment 89](#_Toc63016054)

[Use Case 22: Send Confirmation Notice and Client Contract 90](#_Toc63016055)

[Use Case 23: Staff Views Appointment Updates 91](#_Toc63016056)

[Use Case 24: Staff Add Client Notes 92](#_Toc63016057)

[Use Case 25: Staff View Scheduled Appointments 93](#_Toc63016058)

[Use Case 26: Admin Add New Staff 94](#_Toc63016060)

[Use Case 27: Admin Removes Staff 95](#_Toc63016061)

[Use Case 28/29: Admin Set/Update Staff Qualifications 96](#_Toc63016062)

[Use Case 30: Admin Update Promotions 97](#_Toc63016063)

[Use Case 31: Admin emails the current promotions to the email list 97](#_Toc63016064)

[Use Case 32: Admin adds a service to the system 99](#_Toc63016065)

[Use Case 33: Admin removes a service from the system 100](#_Toc63016066)

[Use Case 34: Admin updates or changes a service in the system 101](#_Toc63016067)

[101](#_Toc63016068)

[Use Case 35: Admin changes the hours of operation or schedule time 102](#_Toc63016069)

[Use Case 36: Admin is notified of an unaccepted appointment after 24 hours. 103](#_Toc63016070)

[Use Case 37: Generate a report. 104](#_Toc63016071)

[Use Case 38: Export generated report. 105](#_Toc63016072)

[Use Case 39: Print generated report. 106](#_Toc63016073)

[Use Case 40: Pay for Appointment. 107](#_Toc63016074)

[Activity Diagram 108](#_Toc63016075)

[State Machine Diagram 109](#_Toc63016076)

[System Design 110](#_Toc63016077)

[Layered Architecture 110](#_Toc63016078)

[Hardware Architecture 110](#_Toc63016079)

[Hardware Platform 110](#_Toc63016080)

[Software Platform 111](#_Toc63016081)

[Login 112](#_Toc63016082)

[Services 113](#_Toc63016084)

[My Pets 114](#_Toc63016086)

[Client Account 115](#_Toc63016088)

[Client Appointment 116](#_Toc63016090)

[Schedule 117](#_Toc63016092)

[Persistence Model 118](#_Toc63016094)

[Project Management 119](#_Toc63016095)

[Schedule 119](#_Toc63016096)

[Team Configuration 120](#_Toc63016097)

[Project Standards and Procedures 120](#_Toc63016098)

[Communication 120](#_Toc63016099)

[Software 120](#_Toc63016100)

[Glossary 122](#_Toc63016101)

[Index 123](#_Toc63016102)

[Appendix A: Data Dictionary 124](#_Toc63016103)

[Appointment 124](#_Toc63016104)

[Service 124](#_Toc63016105)

[Discount 124](#_Toc63016106)

[Promotion 125](#_Toc63016107)

[ClientAccount 126](#_Toc63016108)

[StaffAccount 126](#_Toc63016109)

[StaffQualification 126](#_Toc63016110)

[Location 127](#_Toc63016111)

[Schedule 127](#_Toc63016112)

[ScheduleBlock 128](#_Toc63016113)

[Pet 128](#_Toc63016114)

[GalleryImage 129](#_Toc63016115)

[References 129](#_Toc63016116)

# Git Repository

<https://github.com/Dreiwm/Infinite-Pets.git>

# Contributions

#### Brendan Tran

* Not in OOSAD but in PROJ group
* Created Use Cases 31-40 and Respective Sequence Diagrams
* Assisted in Final Editing
* UI Mock Ups
* Created Use Case Diagrams
* Created Gannt Chart
* General Formatting
  + Spelling
  + Table of Contents (Updated With Proper Headings)
  + Rotated images

#### Christopher Manuel-Smith

* Created State Machine Diagram
* Created Use Cases 15-23 and Respective Sequence Diagrams
* Conducted Client Interviews and Fact Finding
* Assisted in Creation and Final Editing of Design Document
* UI Descriptions

#### Cashton Bieker

* Document Formatting
* Design of the team logo
* Usability Requirements
* Glossary
* Index
* Created Use Cases 9-14 and Respective Sequence Diagrams
* UI Mockups

#### Riley Hiltz

* General formatting changes (page numbers, title page…)
  + There was many weird formatting (ie. Bullets out of place, italic formatted for no reason, etc.).
* Add font standards in “Preface” section
* Added more details in some sections
* Added Data Dictionary section
* Created Use Cases 1-8 and Respective Sequence Diagrams
* (for revised Design Doc)
  + General grammar changes
  + Fixed some errors.
  + Completed 90% of UML class diagram for the announcement and gallery
  + Added some things in some sections (ie. Missing headers, add a little more information, etc.)
* Fully updated data dictionary

#### Nicholas Panos

* Not in OOSAD but in PROJ group
* Created Use Cases 24-30 and Respective Sequence Diagrams
* Assisted in Final Editing

# Preface

This document is the requirements document for the creation of a scheduling program that Jenna Hall has requested for her business, which is Infinite Pets. The goals of this project is to create a streamlined system that will allow ease of registration and management of customers, while providing tools to the staff to excel with their pet care contracts.

This project will require a creation of a database that will contain the profiles of clients and login credentials of staff and client. The data requirements will be those provided by the client and when accessed by a staff member they will be alerted if there is any key information.

# Documentation Standards

## Diagrams

This document will be using UML documentation.

## General Documentation Standards

This documentation will use Calibri font. Heading will be used to clearly define the hierarchy for better reading experience.

Font:

* Calibri

Headings:

* Title – Calibri 48pt
* Heading 1 – Calibri 22pt
* Heading 2 – Calibri 16pt
* Heading 3 – Calibri 12pt
* Body – Calibri 11pt

# User Requirements.

Infinite Pets

Website: <https://www.hallidaypets.com/>

Owner(s): Jenna Hall

Contact: Blair Slind

Infinite Pets is a self-owned, full-service pet care company that aims to “TO PROVIDE PROFESSIONAL, QUALITY CARE WITH THE LEAST AMOUNT OF STRESS TO PETS AS POSSIBLE.” [1] They operate throughout all of Calgary and provide services such as animal beauty care, animal sitting and dog walks.

## Problem Statement

### Current System

The current system is very manual, therefore, is not an efficient system for the owner and the staff to accept/approve the appointment(s) for a service that the client(s) requested. Currently, the clients would need to make a call to the Infinite Pets for any information on his/her appointments or to request a service (appointment). The appointment request will be booked with staff determined by Infinite. Pets. If the appointment information had been updated, the information would be relayed to the staff that had been booked with.

Infinite Pets requires a new scheduler and database that will allow for a higher level of coordination of staff while providing the staff with information to prepare for their upcoming contracts. For the scheduler, the clients need to be able to make an appointment, view the appointment(s), or cancel the appointment. The staff need to be able to view all the clients’ appointment requests, approve, modify information on pets (for example, behavior notes). The owner needs to be able to view the monthly reports and be able to clearly see the remaining appointment requests (usually when there are conflicts in schedule, or no staff is willing to accept the request).

## Project Scope

The project scope will be outlined below. The project scope is based on the requirement gathered from the analysis phase.

* The database will contain the client’s Information and pet(s) Information. The database will also contain information on the appointments, login credentials, gallery, and announcements.
* The clients will be able to create an account.
* The clients will be able to book specific pet care services.
* The staff will be able to view and claim available appointments.
* The staff will have the ability to view their own scheduled appointments.
* Handling Payment (Debit/Credit)
* Report Generation
  + For the owner to view the overall picture of the business operations such as most popular service requests, canceled appointments, etc.
* Backup and Restore
  + If the system crashes, the administration should be able to restore from previous backup to keep the system running.
  + The system should perform an incremental backup automatically per 6 hours.
  + The system should perform a full backup on Sunday at 11:30pm.

## System Environment

The system will exist in two environments. The first environment will be the client, consumer environment. The client environment will be accessible by clients using either a mobile device or a computer. The clients will be able to access their account and request services from Infinite Pets. The second environment will be the staff/business environment. This environment will be the Infinite Pets side of the system. This is where staff can see requests made by clients.

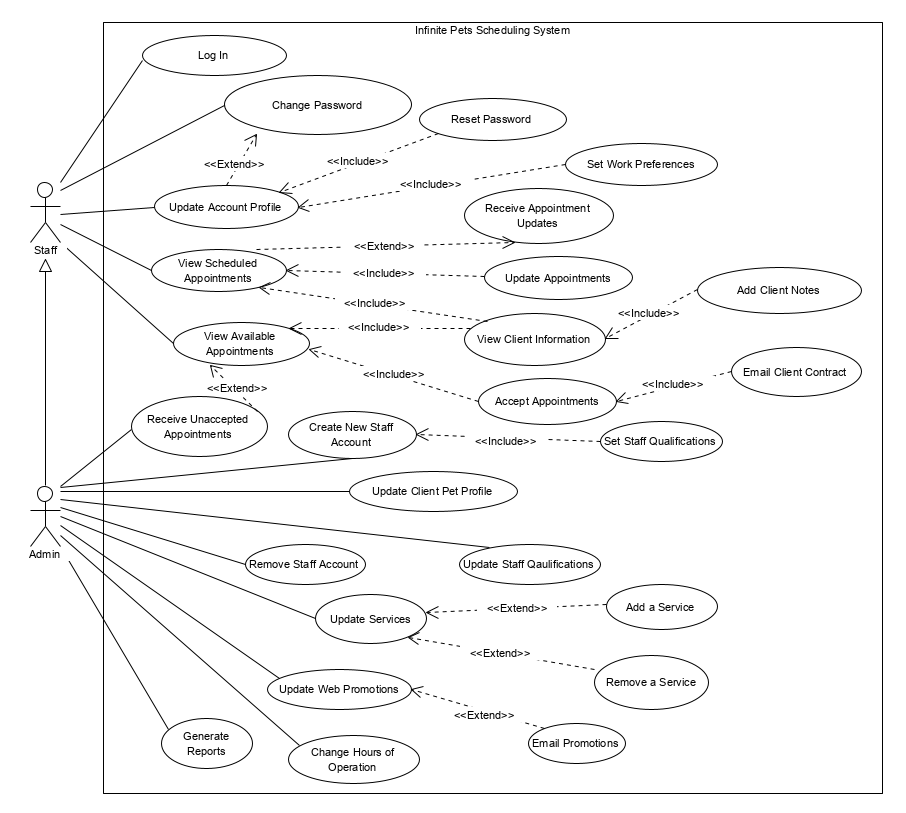
## Current System

The current system is very manual. A customer will need to make a call to Infinite Pets if the client want to request a service that Infinite Pets provides. Infinite Pets must then decide who is going to handle this service, manually check their schedule, then book this service. We will be integrating our system with a pre-existing calendar.

# System Requirements

## Use Case Diagrams





# Use Case Descriptions and Scenarios

## Use Case 1: Login

**Pre-condition:** The user must already have the account in the database

**Post-condition**: The user now is logged in. Will be allowed to have access to functionality relevant to the actor (ie. Customer, staff).

**Limitation(s)**: The user will be denied access for 24 hours if the actor entered the wrong credentials 5 times within 10 minutes.

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1. The user clicks “login” link anywhere on the website | 1. Displays the Login UI for the actor |
| 1. The actor clicks “login” button after entering the email and password credentials | 1. If the email and password matches what is in the database, the user is redirected to the dashboard UI.   -or-  If the credentials are not valid (alternative Flow 4.1) |

|  |  |
| --- | --- |
| **Actor Action (Alternative Flow 4.1)** | **System Response** |
|  | 4.1.1 Display an error message, refreshing the page and retain the user’s email but don’t retain password field. |

## Use Case 2: Create an Account

**Pre-condition:** The actor must not have already created an account in the system

**Post-condition**: The actor now have access to its account.

**Limitation(s)**: The email must not be already in the database.

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1. The user clicks “create an account” link anywhere on the website | 1. Displays the registration UI |
| 1. The user fills in all required information, clicks “register” | 1. If all the inputs are valid, proceeds to the dashboard UI. |

|  |  |
| --- | --- |
| **Actor Action (Alternative Flow 4.1)** | **System Response** |
|  | 4.1 If there were any errors in the input fields (ie. Email already exists, formatting errors), refresh the page and display an error message. Highlight the area of where inputs were invalid. All inputs will be retained except password fields. |
|  |  |

## Use Case 3: Reset Password

**Pre-condition:** The actor must be already registered in the system

**Post-condition**: The password is now set to new password set by the actor.

**Limitation(s)**: N/A

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1. The actor clicks on “forget password” in the login UI | 1. The reset password form will be displayed – email |
| 1. The actor enters the email address | 1. If the email is correct, the system will send an email containing a link to reset password. |
| 1. A message is displayed to the user saying along lines of “If the email you entered is valid, you will receive an email containing a link to reset your password” |  |
| 1. The actor clicks the link on email sent from the system | 1. The change password interface will be presented. |
| 1. The user enters new password, confirm new password field, and clicks change password button. | 1. Display a success message and redirect the user to Login UI. |

|  |  |
| --- | --- |
| **Actor Action (Alternative Flow 4.1)** | **System Response** |
|  | 4.1 if email is not email is not already in the database, do nothing. |

## Use Case 4: Update Account Information

**Pre-condition:** The user must be already registered in the database/system and is logged in.

**Post-condition**: The user now have the updated information

**Limitation(s)**: The actor cannot change an email address. All other fields must be not blank.

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1. The user clicks “profile” from the dashboard UI | 1. The actor is presented with the profile UI |
| 1. The actor clicks “update” button | 1. If all of the inputs were valid, a message will be displayed on top saying “Profile information was successfully updated.” (this is system message) 2. The system sends an email to the actor informing the changes. |

|  |  |
| --- | --- |
| **Actor Action (Alternative Flow 4.1)** | **System Response** |
|  | 4.1 if one or more of input fields were invalid (ie. Blank), the system message will be displayed on top, displaying which part needs to be looked into. |
|  |  |

## Use Case 5: Change Password

**Pre-condition:** The actor must be already registered into the database/system and is logged in

**Post-condition**: The actor now have new password set by the actor.

**Limitation(s)**:

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1. The user clicks “profile” in the dashboard UI | 1. The actor is presented with the profile UI |
| 1. The actor clicks “change password” link | 1. The change password form is displayed to the actor. |
| 1. The user enters old password, new password, confirm new password and clicks “change password” button. | 1. If all the inputs were valid, the system will display a success password change message and redirect back to profile UI. 2. The system will send an email to the actor informing the change. |

|  |  |
| --- | --- |
| **Actor Action (Alternative Flow 5.1)** | **System Response** |
| 5.1 the actor clicks “cancel” | 6.1 redirect the page to profile UI. |

## Use Case 6: Client Delete Account

**Pre-condition:** The actor’s account must be already in the database/system and is logged in.

**Post-condition**: The actor is now no longer in the system/database

**Limitation(s)**:

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1. The actor clicks “profile” link from the dashboard UI | 1. The actor is presented with a profile interface |
| 1. The actor clicks “delete account” | 1. The system prompts the actor for confirmation and inform that the actor needs to confirm it via email (link). |
| 1. The actor clicks “yes” | 1. The system will log the actor out and sends an email. |
| 1. The actor clicks the link to confirm account deletion | 1. The success message will be displayed. |

|  |  |
| --- | --- |
| **Actor Action (Alternative Flow 5.1)** | **System Response** |
| 5.1 the actor clicks “no” | 6.1 redirects the page to profile UI |

|  |  |
| --- | --- |
| **Actor Action (Alternative Flow 7.1)** | **System Response** |
| 7.1 opens the email but not click the link | 8.1 Do nothing. |

## Use Case 7: Client Adds Pet Profile

**Pre-condition:** The actor must be already in the database/system and is logged in.

**Post-condition**: The actor will now have new pet profile added to their account.

**Limitation(s)**: Pet profile can be only added to actor’s account, not others.

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1. The actor clicks “Pets” | 1. Displays the Pets page |
| 1. The actor clicks “add a pet” | 1. The pet profile creation UI is displayed |
| 1. The actor fills in all required input fields and click “add pet” | 1. If the inputs were valid, redirect the page to Pets UI. |

|  |  |
| --- | --- |
| **Actor Action (Alternative Flow 5.1)** | **System Response** |
| 5.1 the actor clicks “cancel” | 6.1 the page will be redirected to profile UI |

|  |  |
| --- | --- |
| **Actor Action (6.2)** | **System Response** |
| 7.1 opens the email but not click the link | 8.1 Do nothing. |

## Use Case 8: Client Views Pet Profile List

**Pre-condition:** The actor must have already registered in the system/database and is logged in.

**Post-condition**: The Pets UI is presented to the actor.

**Limitation(s)**: The actor can only view all pets that belongs to this actor.

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1. Actor clicks “Pets” link in the Dashboard UI | 1. The Pets UI is displayed. |
|  |  |

## Use Case 9: Client views a pet’s profile

**Precondition:** the client must be logged into the system and have previously added one or more pets to their account. The client must have already clicked the “my pets” button in order to view the list of their pets.

**Postcondition:** All the information for the requested pet is displayed to the user.

|  |  |
| --- | --- |
| **Actor action** | **System response** |
| 1. The user clicks the “my pets” button. | 2. return and display a list of all the pets associated to that user’s email. |
| 3. The user clicks on the “view pet information” button. | 4. return and display all the information associated with the pet that was clicked on. |

## Use Case 10: Client Updates a pet’s information

**Precondition:** The client must be logged into the system and have previously added one or more pets to their account. The client must have already clicked the “my pets” button and viewed the profile for the pet they would like to perform the update on.

**Postcondition:** The requested pet’s information is updated and displayed to the user.

|  |  |
| --- | --- |
| **Actor action** | **System response** |
| 1. The user clicks the “my pets” button. | 2. return and display a list of all the pets associated to that user’s email. |
| 3. The user clicks on the “view pet information” button. | 4. return and display all the information associated with the pet that was clicked on. |
| 5. The user clicks on the “update pet” button. | 6. the current information for the requested pet is displayed in editable fields allowing the user to change any of the pet’s information. |
| 7. The user changes any of the desired information for the displayed pet and clicks the “save” button. | 8. the new information for the pet is saved to the database overwriting any of the information that was changed by the user. |

## Use Case 11: Client view’s the available schedule

**Precondition:** the client must be logged into the system.

**Postcondition:** A schedule displaying the available dates is displayed to the user.

|  |  |
| --- | --- |
| **Actor action** | **System response** |
| 1. The user clicks the “available appointments” button. | 2. return and display a weekly calendar showing all the available dates that an appointment could be booked. |
| 3. The user selects a service in order to view the available dates for that service. | 4. return and display a weekly calendar updated to show all the available times for the service the user selected. |

## Use Case 12: Client books an appointment

**Precondition:** the client must be logged into the system. The user must have one or more pets added to their account.

**Postcondition:** A new appointment is added to the user’s account. The user is notified via email that a new appointment has been requested. The requested appointment is sent to the employee side of the application.

|  |  |
| --- | --- |
| **Actor action** | **System response** |
| 1. The user clicks the “available appointments” button. | 2. return and display a weekly calendar showing all the available dates that an appointment could be booked. |
| 3. The user selects a service to view the available dates for that service. | 4. return and display a weekly calendar updated to show all the available times for the service the user selected. |
| 5. The user selects the available date on which they would like to schedule their requested service. | 6. return and display the available times for the user requested date. |
| 7. The user selects their preferred time for their selected service. | 8. return and display a menu showing an “add pet button.” |
| 9. The user selects which pet or pets will be receiving the selected service. The user has the option to add any special notes for each pet and finally clicks the “book service button.” | 10. all the information including date, time, and a list of pets and their corresponding notes, entered by the user is sent to the database. A new appointment is sent to the user’s account. The user is notified via email that a new appointment was requested. All the information for that appointment is sent to the employee side of the application via the database. |

## Use Case 13: Client views their upcoming appointments

**Precondition:** the user must be logged in and have already scheduled one or more appointments.

**Postcondition:** a list of all the users scheduled appointments is displayed to the user.

**Limitation:** the appointments displayed to the user must have not been completed already.

|  |  |
| --- | --- |
| **Actor action** | **System response** |
| 1. The user clicks the “my appointments” button. | 2. return and display a list of all the appointments for that user that have not yet been completed. |

## Use Case 14: Client updates an appointment

**Precondition:** the user must be logged in and have already scheduled one or more appointments. The user must have already clicked the “my appointments” button and must be on the page displaying all of their current appointments.

**Postcondition:** the information is updated for the user specified appointment. The user is notified of this update via email. The employee assigned to the appointment is notified of the changes made to the appointment via email.

**Limitation:** the appointment to be updated must not have already occurred. The date and time for appointment to be updated must be more than 24 hours later than the date and time that the update occurs.

|  |  |
| --- | --- |
| **Actor action** | **System response** |
| 1. The user clicks the “edit appointment” button for an appointment in the displayed appointments list. | 2. return and display all the information for the user selected appointment in editable fields. |
| 3. The user makes their desired changes to the requested appointment and clicks the “save” button. | 4. the information for that appointment is updated in the database, a notification is sent to the user and the assigned employee displaying the changes made to that appointment. |

## User Case 15: Client Cancels Appointment

**Precondition:** Client has an existing appointment.

**Postcondition:** Client’s appointment is removed from the system and the staff assigned is informed of the cancelation.

|  |  |
| --- | --- |
| Actor Action | System Response |
| 1. The Client clicks on the Schedule Tab | 2. System displays the Client Schedule UI |
| 3. Select the appointment that will be cancelled | 4. Display the information of the appointment |
| 5. The Client will click “Cancel Appointment” | 6. System will display a confirmation message. |
| 7. Client clicks confirm to cancel the selected appointment. | 8. Confirmation message that the appointment has been cancelled. |
|  | 9. Staff assigned to the appointment and Admin are emailed about the appointment cancellation. |
|  | 10. System displays the Client Schedule UI |

|  |  |
| --- | --- |
| Alternate Flow 8.1 | 8.1.1 If the cancellation is within 24hr of the appointment time, Payment options are displayed. |
| 8.1.2 Client fills out cancellation payment  form | 8.1.3 Payment is processed |
|  | 8.1.4 Confirmation message that the appointment has been cancelled. |

## User Case 16: Client Views Client Contract

**Precondition:** Client has a valid appointment

**Postcondition:** Client Contract is Displayed Client is logged in.

|  |  |
| --- | --- |
| Actor Action | System Response |
| 1.The Client clicks on the Schedule Tab | 2. System displays the Client Schedule UI |
| 3. Client clicks on the desired appointment to view | 4. Display the information of the appointment |
| 5. Client clicks “View Contract” | 6. Display the Client Contract. |

## User Case 17: Client Prints Client Contract

**Precondition**: Client has a valid appointment, Client is logged in.

**Postcondition**: Client Contract is printed

|  |  |
| --- | --- |
| Actor Action | System Response |
| 1.The Client clicks on the Schedule Tab | 2. System displays the Client Schedule UI |
| 3. Client clicks on the desired appointment to view | 4. Display the information of the appointment |
| 5. Client clicks “View Contract” | 6. Display the Client Contract. |
| 7. Client selects “Print” from browser UI | 8. Browser opens print preview |
| 9. Client confirms print settings | 10. Client Contract is sent to printer |

## User Case 18: Staff Sets Work Preferences

**Precondition**: Staff account is valid, Staff is logged in.

**Postcondition**: Staff work settings are updated

**Limitations**: Staff preferences must be valid days of the week and work areas.

|  |  |
| --- | --- |
| Actor Action | System Response |
| 1. Staff member clicks on the Profile Tab | 2. System displays the Staff Profile UI |
| 3. Staff clicks “Update Preferences” | 4. System displays the preferences information. |
| 5. Staff changes the preferences. | 6. Confirmation message is displayed containing the changed preferences. |
| 7. Staff confirms the changes by clicking “Confirm” | 8. System is updated with Staff’s new preferences |
|  | 9. Staff Profile UI is displayed |

|  |  |
| --- | --- |
| Alternate Flow 7.1 |  |
| 7.1.1 Staff rejects the changes by clicking “Reject” | 7.1.2 Changes are discarded. |
|  | 7.1.3 System displays the preferences information. |

## User Case 19: Staff Views Available Appointments

**Precondition**: Valid and available Appointments exist, Staff account is valid, Staff is logged in.

**Postcondition**: Available appointments are displayed.

**Limitations**: Appointment displayed must meet Staff’s preferences and qualifications.

|  |  |
| --- | --- |
| Actor Action | System Response |
| 1. Staff clicks on the Schedule tab | 2. Staff Schedule UI is displayed |
| 3. Staff clicks “Show Available Appointments” | 4. Available appointments are populated using Staff member’s preferences and displayed. |

## User Case 20: Staff Views Client and Pet Information

**Precondition**: Staff account is valid, Staff is logged in, Client is valid and has an available Appointment.

**Postcondition**: Client and Pet information is displayed.

|  |  |
| --- | --- |
| Actor Action | System Response |
| 1. Staff clicks on the Schedule tab | 2. Staff Schedule UI is displayed |
| 3. Staff clicks “Show Available Appointments” | 4. Available appointments are populated using Staff member’s preferences and displayed. |
| 5. Staff clicks “Display Detailed Information” on desired appointment. | 6. System displays Client and Pet Information. |

## UserCase 21: Staff Accepts Available Appointment

Precondition: Appointment is valid and available, Staff account is valid, Staff is logged in.

**Postcondition**: Appointment is no longer available.

**Limitations**: Appointment meets Staff’s work preferences.

|  |  |
| --- | --- |
| Actor Action | System Response |
| 1. Staff clicks on the Schedule tab | 2. Staff Schedule UI is displayed |
| 3. Staff clicks “Show Available Appointments” | 4. Available appointments are populated using Staff member’s preferences and displayed. |
| 5. Staff clicks “Book Appointment” on desired appointment. | 6. Confirmation Message with any pet warnings is displayed. |
| 7. Staff confirms booking by clicks “Confirm” | 7. System Updates Appointment. |

|  |  |
| --- | --- |
| Alternate Flow 5.1 |  |
| 5.1.1 Staff clicks “Display Detailed Information” on appointment. | 5.1.2. System displays Client and Pet Information. |
| 5.1.3. Staff clicks “Book Appointment” on desired appointment. |  |

## User Case 22: Send Confirmation Notice and Client Contract

**Precondition**: Client and Staff accounts are valid, Appointment is valid, Staff accepted Appointment.

**Postcondition**: Client and Staff are sent emails confirming Appointment Booking.

|  |  |
| --- | --- |
| Actor Action | System Response |
| 1. Staff confirms booking by clicks “Confirm” | 2. System updates Appointment. |
|  | 3. System emails Client and Staff of the Appointment Booking. |

## User Case 23: Staff Views Appointment Updates

**Precondition**: Staff account is valid, Appointment is valid.

**Postcondition**: Appointment information is displayed.

**Limitations**: Staff has accepted an appointment that has been updated by Client.

|  |  |
| --- | --- |
| Actor Action | System Response |
|  | 1. System emails Staff that Appointment is updated. |
| 2. Staff logins into account | 3. Profile UI is Displayed. |
| 4. Staff clicks Schedule Tab | 5. System displays “Schedule UI” |
| 6. Staff clicks “Detailed Information” on updated Appointment. | 7. Appointment Information is displayed. |

## User Case 24: Staff Add Client Notes

**Precondition**: User is logged in to their staff account, Client is valid.

**Postcondition**: Client information is displayed.

|  |  |
| --- | --- |
| Actor Action | System Response |
| 1. Staff goes to client information page for a client | 2. System retrieves client information from database and displays it |
| 3. Staff Writes a note in the provided field and clicks submit | 4. System updates client information to include the note. |
|  | 5. System retrieves the changed client information from database and displays it |

## User Case 25: Staff View Scheduled Appointments

**Precondition**: User is logged in to their staff account

**Postcondition**: Scheduled appointments are displayed

|  |  |
| --- | --- |
| Actor Action | System Response |
| 1. Staff goes to their appointments | 2. System retrieves appointments that are assigned to the staff member from the database and displays it. |

## Use Case 26: Admin adds a new staff member

**Pre-condition:** The user must be logged into an admin account.

**Post-condition**: The updated staff list and information will be displayed and the new staff member will be in the database.

**Limitation(s)**: N/A

|  |  |
| --- | --- |
| Actor Action | System Response |
| 1. Admin goes to staff management page | 2. System retrieves staff list and information from database and displays it |
| 3. Admin enters the details for a new staff member in the provided fields and clicks add | 4. System adds new staff member to the database |
|  | 5. System retrieves the changed staff list and information and displays it. |

## Use Case 27: Admin removes a staff member

**Pre-condition:** The user must be logged into an admin account, and there must be at least one staff member to remove

**Post-condition**: The updated staff list and information will be displayed and the staff member will be removed from the database.

**Limitation(s)**: N/A

|  |  |
| --- | --- |
| Actor Action | System Response |
| 1. Admin goes to staff management page | 2. System retrieves staff list and information from database and displays it |
| 3. Admin selects a staff member and clicks remove | 4. System removes staff member from the database |
|  | 5. System retrieves the changed staff list and information and displays it. |

## Use Case 28/29: Admin updates a staff member’s qualifications

**Pre-condition:** The user must be logged into an admin account, and the user being updated must be valid.

**Post-condition**: The updated staff details will be displayed and the staff member’s details will be updated in the database.

**Limitation(s)**: N/A

|  |  |
| --- | --- |
| Actor Action | System Response |
| 1. Admin goes to staff details page for a staff member | 2. System retrieves staff user information from database and displays it |
| 3. Admin changes the services that staff member is qualified to preform and clicks save. | 4. System persists those changes to the database |
|  | 5. System retrieves the changed staff user information from database and displays it |

## Use Case 30: Admin updates the current promotions

**Pre-condition:** User is logged into their admin account.

**Post-condition**: The updated promotions list will be displayed and the changes persisted to the database.

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1. Admin goes to promotions management page. | 2. System retrieves current promotions from database and displays them |
| 3. Admin changes the promotions, adds, removes, or changes them in the UI and clicks save. | 4. System retrieves current promotions from database and compares them to edits |
|  | 5. System removes, updates and inserts to the database as needed. |
|  | 6. System retrieves updated promotions from database and displays them |

## Use Case 31: Admin emails the current promotions to the email list.

**Pre-condition:** User is logged into their admin account and currently viewing the promotions page. The admin must have also edited/updated the promotions page.

**Post-condition**: The user will send an email of the new updated promotions.

**Limitation(s)**: Promotion needs to be updated in order for updated to be sent out to active subscribed clients.

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1. The user clicks the “update” button. | 1. Send a response window asking if the user would like to send an email. |
| 1. The user will click “yes.” | 1. System would get list of active subscribed clients. |
|  | 1. Generates an email of the promotions and sends it out to each client. |
|  | 1. Send a response that promotions have been emailed. |
| 1. The user will click “okay” to confirm and automatically close the response window. |  |

## Use Case 32: Admin adds a service to the system.

**Pre-condition:** The user must be logged into an admin account and viewing the services page.

**Post-condition**: The service will be added to the list of available services in the database.

**Limitation(s)**: N/A

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1. The user clicks “add new service” button on the services page. | 1. Display the new service form on the service UI. |
| 1. The user will fill out the proper information for the service form |  |
| 1. The user clicks the “add” button to submit the service form | 1. Checks the data and save persist and merges it to the database. |
|  | 1. Send a confirmation that the database successfully added the data. |
| 1. The user clicks “okay” to confirm and automatically close the response window. |  |

## Use Case 33: Admin removes a service from the system.

**Pre-condition:** The user must be logged into an admin account and viewing the services page.

**Post-condition**: The selected service will be deleted from the services database.

**Limitation(s)**: N/A

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1. The user clicks the delete link corresponding to the desired removed service. | 1. Sends a response confirming the removal of selected service. |
| 1. User clicks “confirm” button to confirm removal of selected service. | 1. System removes the service from the database |
|  | 1. Displays updated service list o |

|  |  |
| --- | --- |
| **Alternate Flow from 4** |  |
| 4.1 User will click “cancel” | 5.1 Returns the user back to the service UI. |

## Use Case 34: Admin updates or changes a service in the system.

**Pre-condition:** The user must be logged into an admin account and viewing the services page.

**Post-condition**: The selected service will be updated from the services database.

**Limitation(s)**: N/A

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1. The user selects the service they would like to update/edit | 1. Populates an edit UI with selected service information |
| 1. The user alters service information. |  |
| 1. The user clicks “update” button to update the information | 1. System persist and merges the updated service. |

|  |  |
| --- | --- |
| **Alternate Flow from 4** |  |
| 4.1 User will click “cancel” | 5.1 Returns the user back to the service UI. |

## Use Case 35: Admin changes the hours of operation or schedule time

**Pre-condition:** The user must be logged into an admin account and viewing the user scheduling page.

**Post-condition**: The schedule will be updated with the available hours to book an appointment.

**Limitation(s)**: N/A

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1. The admin selects dates | 1. System returns the schedule block |
| 1. Admin edits schedule block information | 1. Update schedule with schedule block information |
|  | 1. Returns new schedule |
| 1. View new schedule |  |

## Use Case 36: Admin is notified of an unaccepted appointment after 24 hours of the appointment being submitted.

**Pre-condition:** The appointment has not been accepted by other staff within 24 hours of it being booked.

**Post-condition**: The admin will receive an email notification that an appointment has been booked but not accepted.

**Limitation(s)**: N/A

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
|  | 1. System will check the list of unaccepted appointments booking time. |
|  | 1. Will generate an email with unaccepted appointments information that has been booked for 24 hours. |
|  | 1. Send generated email to the admin for confirmation. |

## Use Case 37: Generate a report.

**Pre-condition:** The user must be logged into an admin account and viewing the reports page.

**Post-condition**: A report will be generated depending on what the admin’s criteria for the report.

**Limitation(s)**: N/A

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1. User will select the criteria for the report. |  |
| 1. User will click “generate report” after filling out the criteria. | 1. Retrieve the desired data |
|  | 1. Display the report with desired data on the report UI |

|  |  |
| --- | --- |
| **Alternate Flow from 2** | **System Response** |
|  | 3. System will display an error message saying report data cannot be found. |
| 4.1 User will click “confirm” | 5.1 Reset report criteria |
| **Extensions** |  |
| After step 4, the administrator can print or export the report. |  |

## Use Case 38: Export generated report.

**Pre-condition:** The user must be logged into an admin account and viewing the current generated report page.

**Post-condition**: A report will be export or printed depending on the user’s choice.

**Limitation(s)**: N/A

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1. The admin will select the “print” button on the current generated report. | 1. Will generate a document spreadsheet of the report. |
|  | 1. Sends a response displaying the report in a save UI. |
| 1. User will select the directory to save the export file and click “save.” | 1. Will save the exported report in selected directory. |
|  | 1. Return user to report UI. |

|  |  |
| --- | --- |
| **Actor Action (Alternate Flow 4.1)** | **System Response** |
| 4.1 User will click “cancel”. | 5.1 Close save UI window and return the user to the report UI. |

## Use Case 39: Print generated report.

**Pre-condition:** The user must be logged into an admin account and viewing the current generated report page.

**Post-condition**: A report will be export or printed depending on the user’s choice.

**Limitation(s)**: N/A

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1. The admin will select the “print” button on the current generated report. | 1. Will generate a document spreadsheet of the report. |
|  | 1. Sends a response displaying the report print preview in a print UI. |
| 1. User will enter the number of copies and click “print” button. | 1. Will send information to provided printer for printing. |
|  | 1. Return user to report UI. |

|  |  |
| --- | --- |
| **Actor Action (Alternate Flow 4.1)** | **System Response** |
| 4.1 User will click “cancel”. | 5.1 Close save UI window and return the user to the report UI. |

## Use Case 40: Pay for Appointments.

**Pre-condition:** The user must be logged as a client and is currently booking the appointment on the scheduling page.

**Post-condition**: The client will book an appointment and paid for it.

**Limitation(s)**: N/A

|  |  |
| --- | --- |
| **Actor Action** | **System Response** |
| 1. The user will click on “confirm appointment” button. | 1. System will check booking information. |
|  | 1. Send a response to display billing information and confirmation. |
| 1. User will then fill out or confirm saved billing information and click on “pay now” button. | 1. System will check billing and payment information. |
|  | 1. Save a printable copy to the users account with appointment details. |
|  |  |
|  | 1. Display printable receipt |
| 1. User will close the receipt. | 1. Returns and displays the updated scheduling page with new appointment. |

|  |  |
| --- | --- |
| **Actor Action (Alternate Flow 4.1)** | **System Response** |
| 4.1 User will click “cancel”. | 5.1 Close save UI window and return the user to the report UI. |

## Functional Requirements

### Client

* A client visits the company’s website, clicks register button, and the registration form is displayed. The user fills the form and clicks submit button.
* A client logs in the system, the user’s dashboard appears with actions. The user clicks “add a new pet”, and the form for adding a new pet is displayed. The user fills the form and clicks the submit button.
* A client log in the system, the user’s dashboard appears with actions. The user clicks add an appointment and the form is displayed. The user fills the form including a time block and a service. The user then clicks “add an appointment” button.
* A client selects a particular appointment and options for said appointment is displayed. The user clicks request cancellation. The staff at the company will confirm with the user.
* A client logs in the system and clicks “Account”. The account page is displayed and the user clicks “edit account”. The user makes some changes in her/his account and clicks submit.

### Staff

* A staff logs in the system and the dashboard is displayed. The staff clicks on the “view reservations” button and the reservations is displayed.
* A staff logs in the system and the dashboard is displayed. The staff clicks on the “view calendar” button and the calendar is displayed.
* A staff gets notified that a reservation is cancelled. The staff opens the cancelled reservation and views the information.
* Alerts for staff of important information regarding pet or client (medical or other) when file is accessed.
* The owner logs in the system, the dashboard is displayed. The owner clicks on “announcements” button. The owner clicks “new announcement” and enter all necessary input fields. The owner clicks on “publish”.
* The owner logs in the system, the dashboard is displayed. The owner clicks “gallery” and the gallery of images is displayed.
  + The owner clicks “add” and enter all necessary fields, including uploading an image.
  + The owner clicks “delete” on one of those image.

## Non-Functional Requirements

### Client

* The clients should be able to add a specific instruction with limit of 100 characters.
* The clients must provide a reason upon cancelling the appointment – as per to the policy provided.
* The clients can only book in a specific time block. They are:
* Early Morning (06:00-09:00)
* Morning (09:00-12:00)
* Afternoon (12:00-16:00)
* Evening (16:00-18:00)

Staff

* For the staff, the system must have access roles for staff and administrators.
* Staff members can have the option to have their Client Contracts pre-signed with a digital signature.
* Administrators can view an individual or all staff members schedules.
* An Administrator can send emails to both staff members and clients regarding upcoming promotions.

General

* The login process should be performed within 5 seconds in normal conditions.
* The process for schedule submission should be completed within 15 seconds.

## System Interface Requirements

* Interacts with Clients and Staff’s Emails
* Interaction with 3rd party payment methods

## Maintainability and Administration Requirements

### Maintainability

* Adjustments may have to be made to the system in order to keep the integrated calendar running smoothly.
  + Client may be finding themselves frustrated with user interface. We may want to adjust based on feedback from client and staff alike.
  + We’ll be gathering the bug reports within few weeks and fix them accordingly.
  + We’ll be monitoring, if allowed, the database to ensure that data is clean.

### Administration

* Admin can add or remove the staff’s qualifications.
  + For example, a staff is skilled in nail clipping and bathing. Do note that an owner is also considered as an administrator.
* Password and Account Reclamation
  + Client accounts will be able to recover their password through a uniquely generated link from a “account recovery” page
  + Staff accounts are only able to be recovered by contacting an admin who will then set a recovery password for the user.
  + Once an account is locked only an administrator can unlock it.
* Backup and Restore
  + The backup should be done every 12 hours. We believe this is sufficient as the company is a small business, currently.
* An admin account must always exist

## Usability Requirements

### Client

The client is any customer of Infinite Pets. All client interaction will be done through the Infinite Pets website. For this reason, the website will have a very easy to use interface. They will be able to easily understand how to add their profile to the Infinite Pets system. Their profile will include their name, contact info, and a description of their pet. Once a profile is created the client will be able to book services that Infinite Pets offers.

### Staff

The staff of Infinite Pets will use the system to access their schedule. The schedule will display their upcoming workdays. The staff will also receive various notifications based on new events added to their schedule, or event removed from their schedule. While the system will be straight forward, the staff will still be trained on how to use the system.

### User Interface

#### Forms

The user interface must be clear and easy to read. The “cancel” button should be in places where the client needs to submit a form (for example, change password). Upon clicking on submit or update button, the client should be directed to where they can see newly added information. For example, if a client added a pet profile, he/she should be directed to pets list view so that the client can see new information.

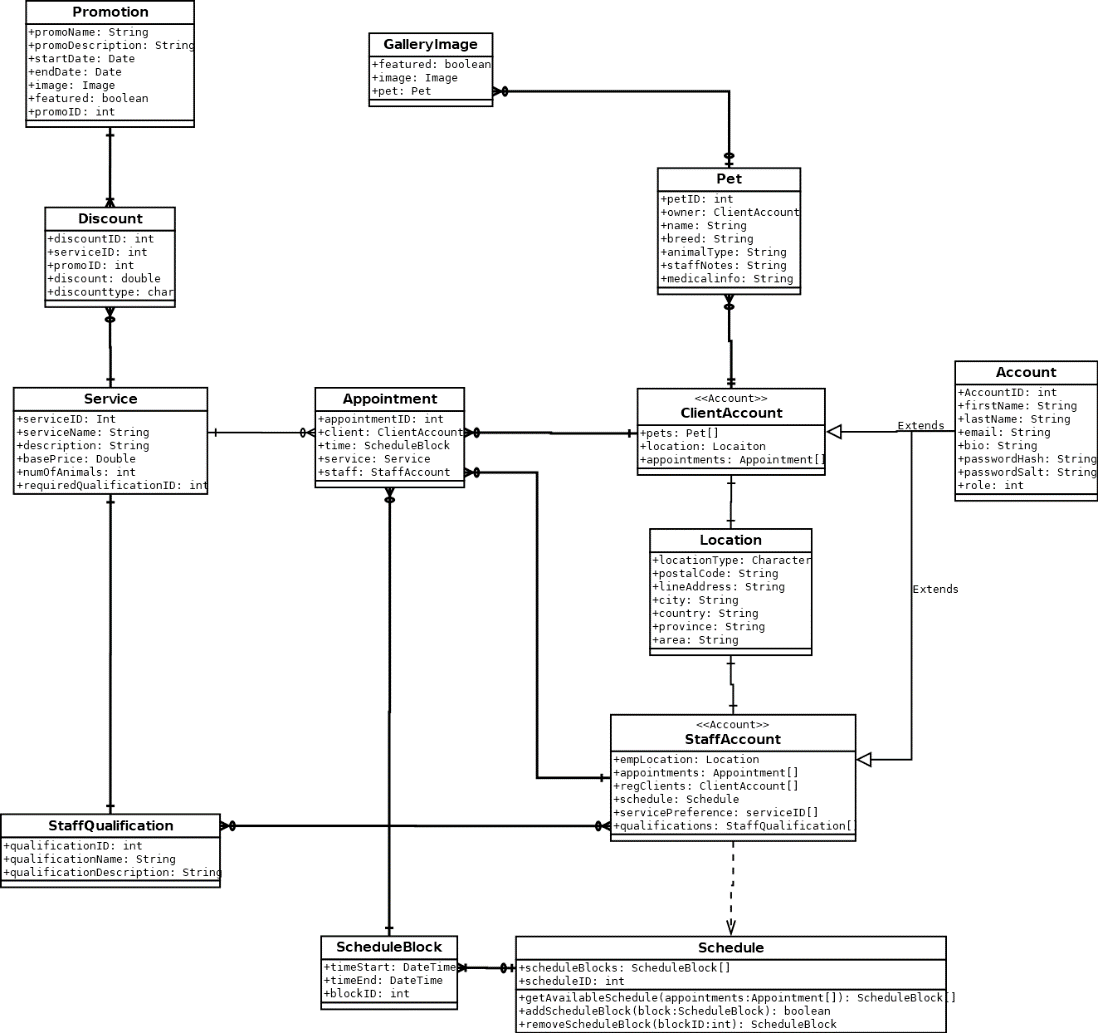
For any of the errors such as invalid login credentials, a clear and lightly shaded red box should appear on top area of form. It should list the label name of input (login credentials, pet name, pet birthday, etc) and a short description of what input is expected.

If the client changed any information regarding account and clicked submit, a system message on top of the page will say “Profile had been successfully updated”.

### Administrator/Owner

The administrator will have complete control over the system. They will be able to see which clients have requested their service. They can then assign this service to a staff member. The administrator will be able to create and remove a staff member from the system. The administrator will be able to view all of the clients in the system as well. Because the administrator side of the system will have a large range of functionality, the administrator will have to be trained on all aspects of the system.

# Problem Domain Class Diagram

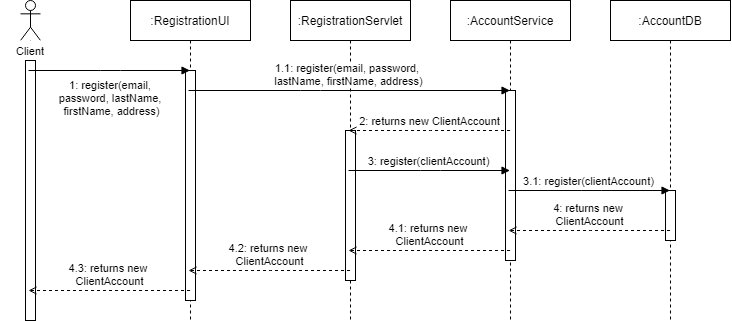


# Sequence Diagrams

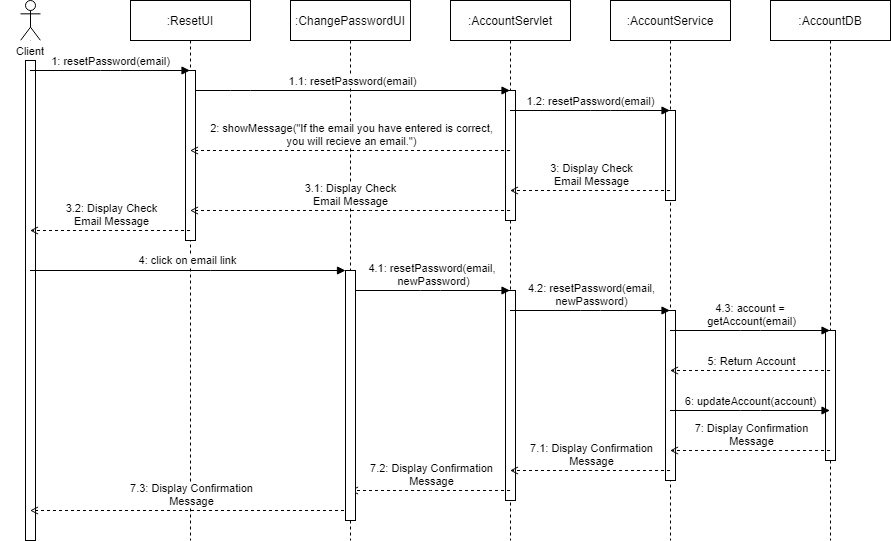
## Use Case 1: Login

## 

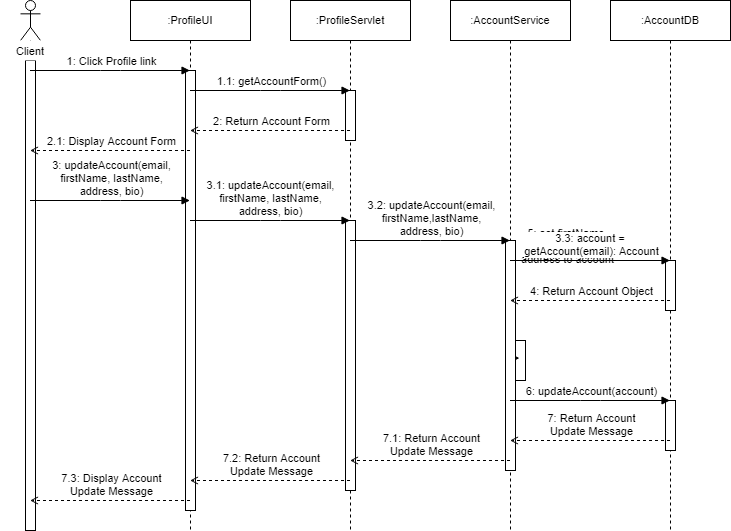
## Use Case 2: Create an Account



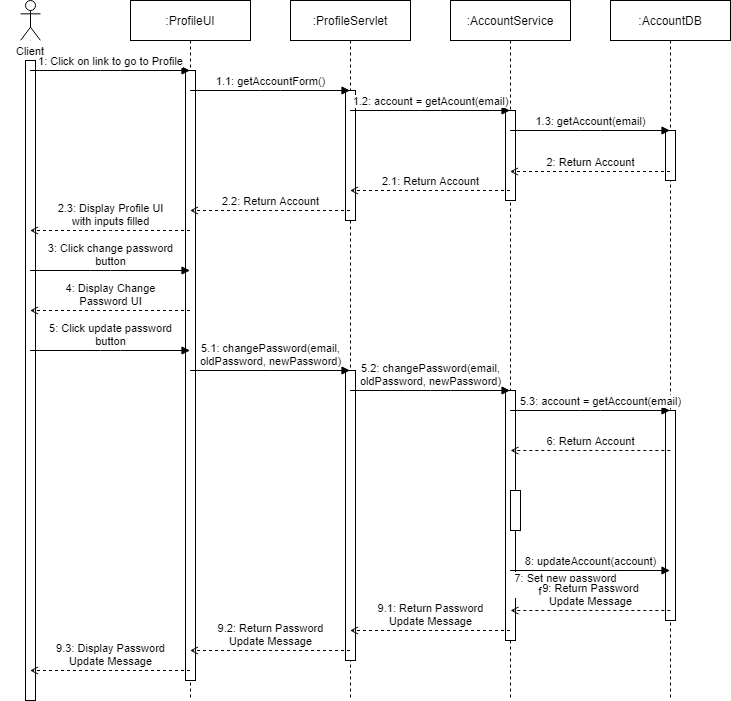
## Use Case 3: Reset Password



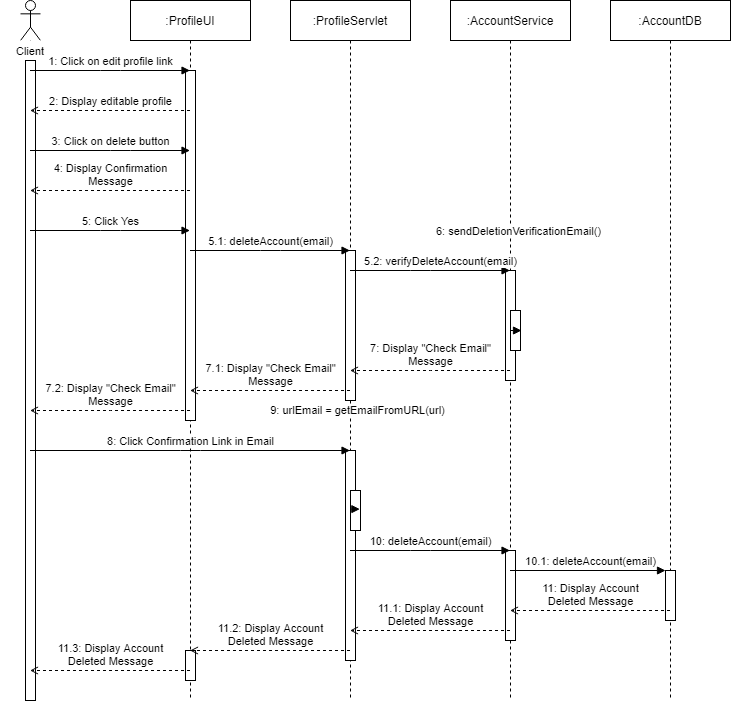
## Use Case 4: Update Account Information



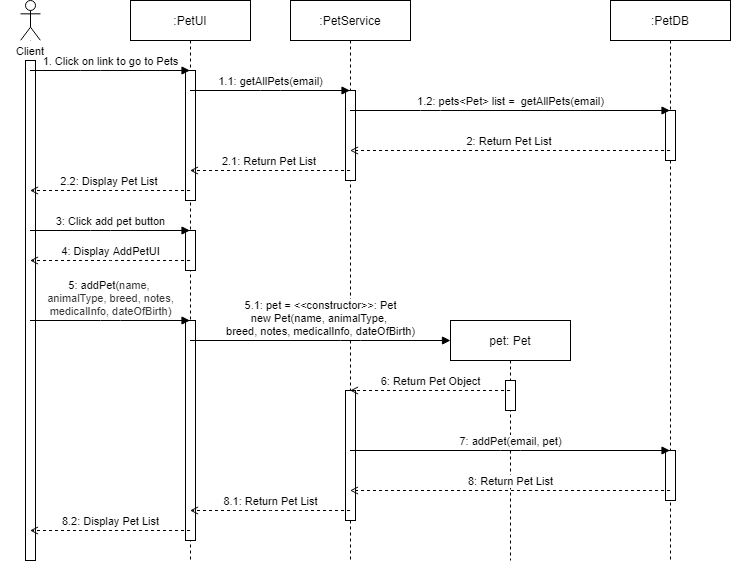
## Use Case 5: Change Password



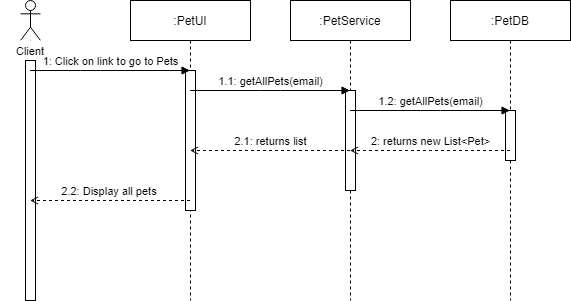
## Use Case 6: Client Delete Account



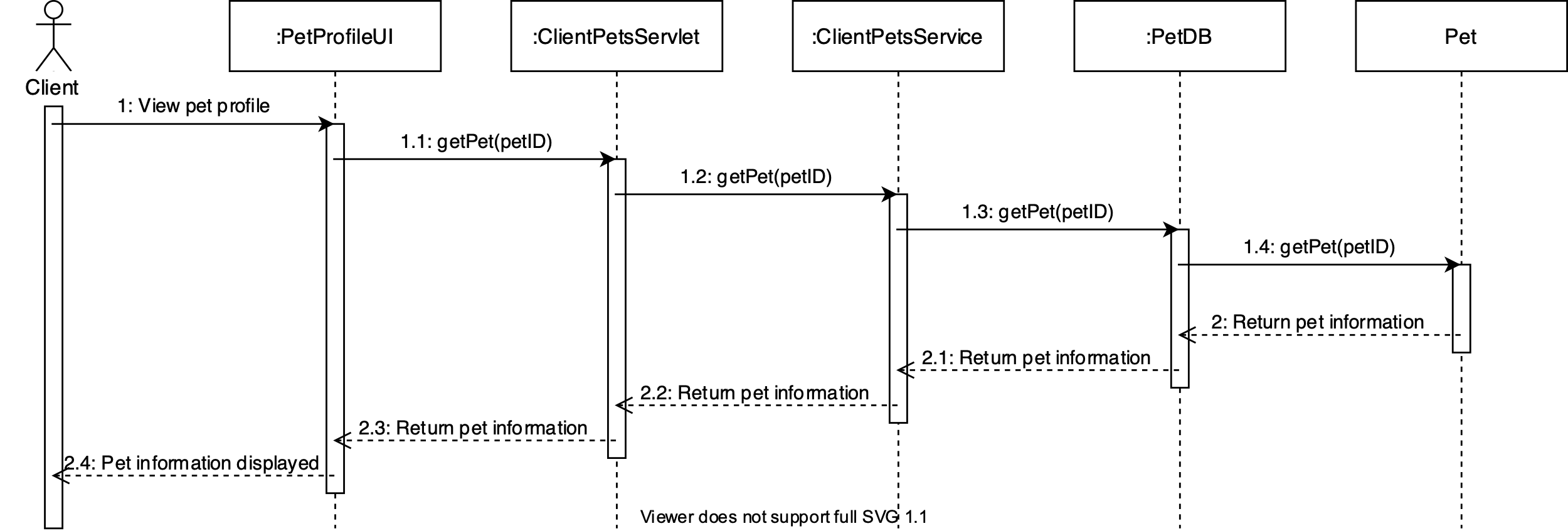
Use Case 7: Client Adds Pet Profile



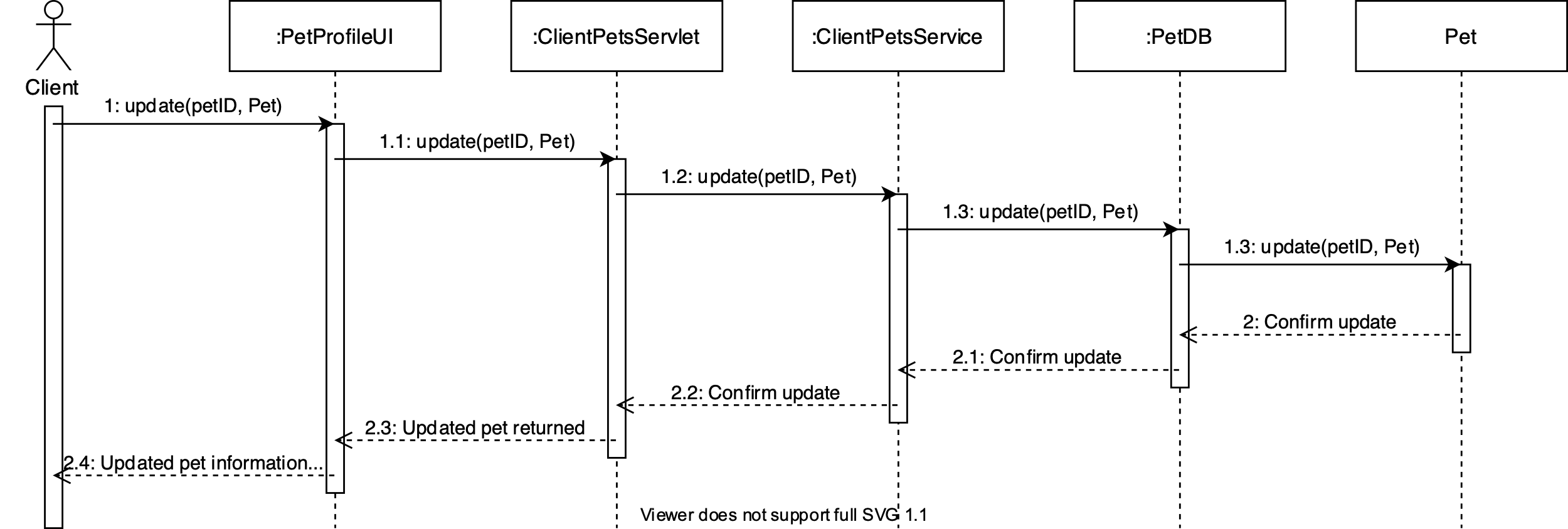
## Use Case 8: Client Views Pet Profile List



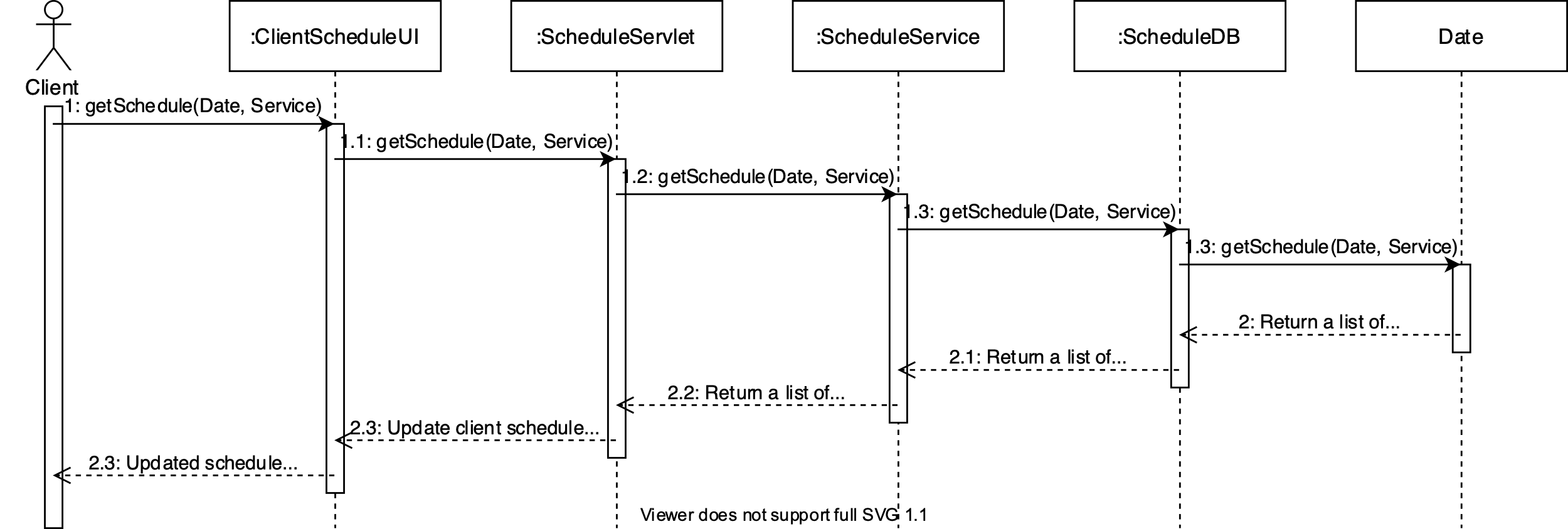
Use Case 9: Client views a pet’s profile



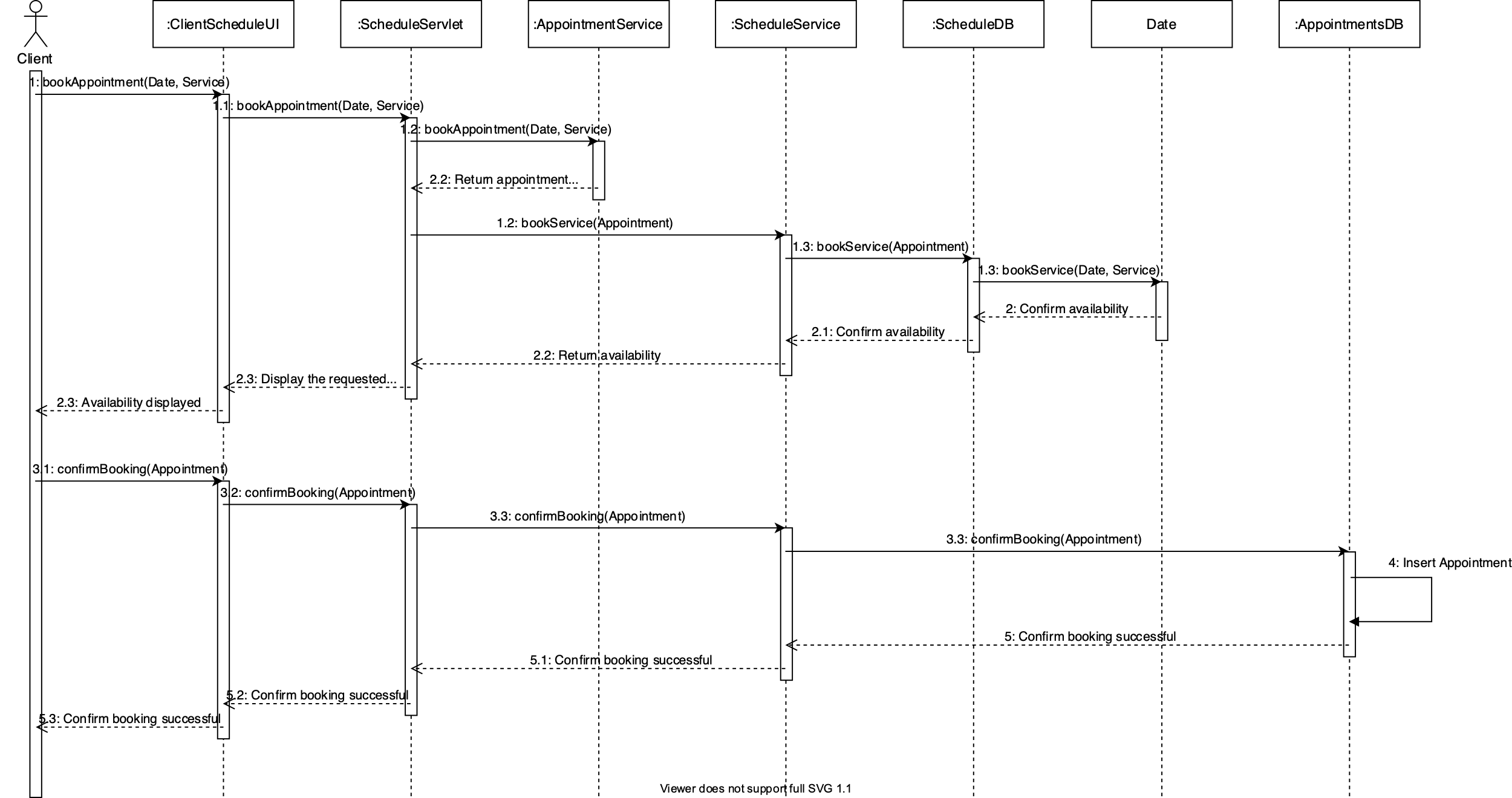
## Use Case 10: Client Updates a pet’s information



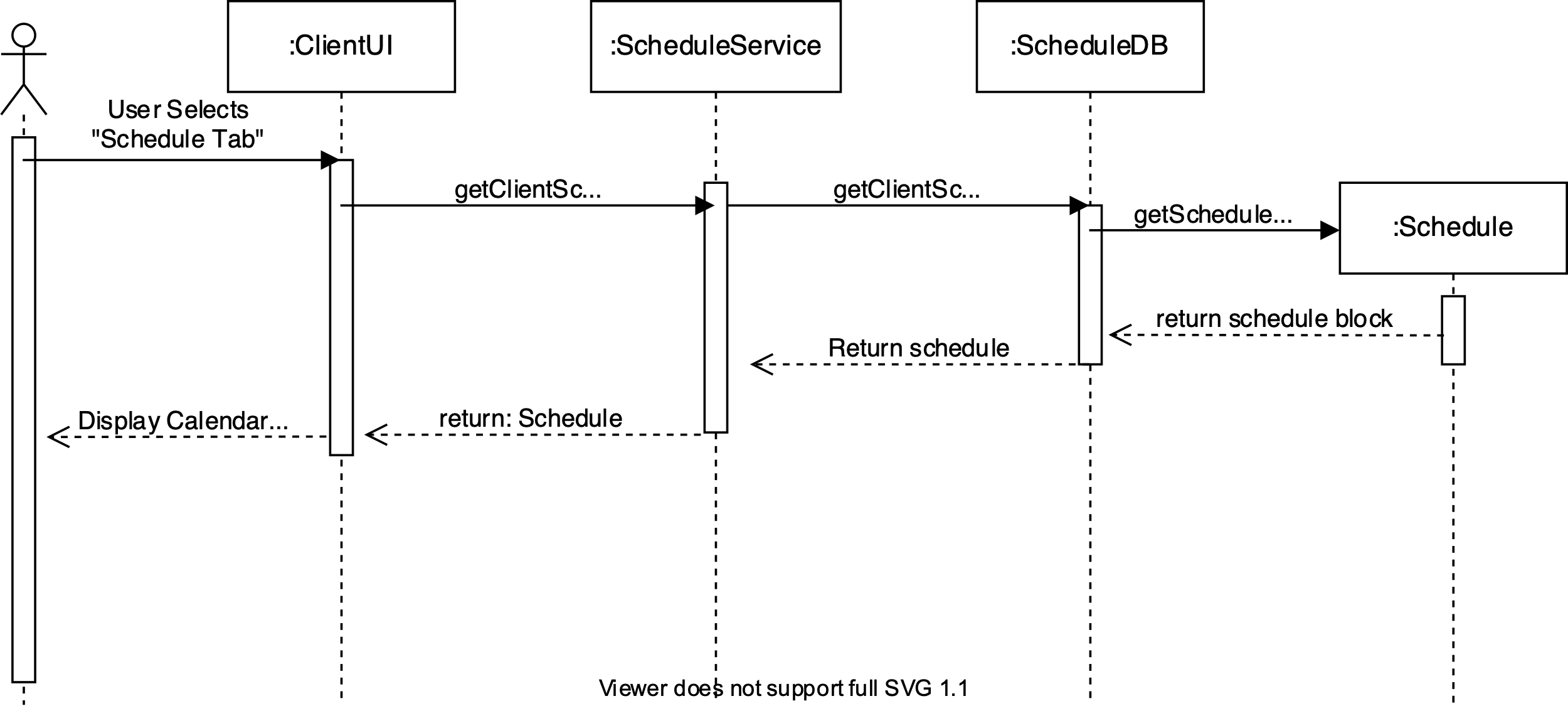
## Use Case 11: Client view’s the available schedule



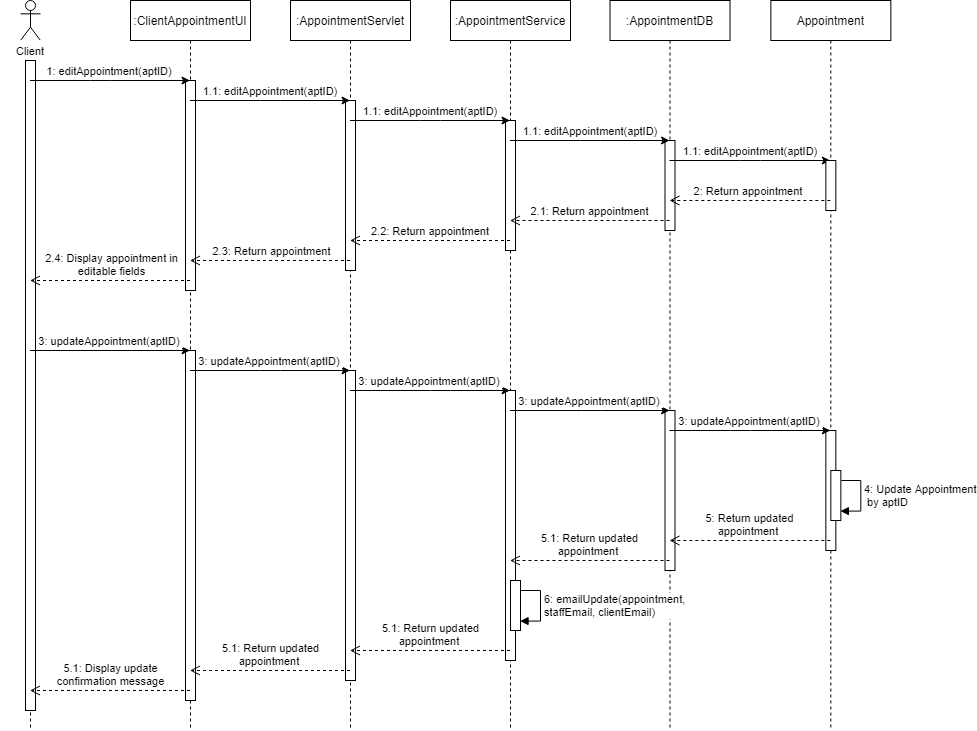
## Use Case 12: Client books an appointment



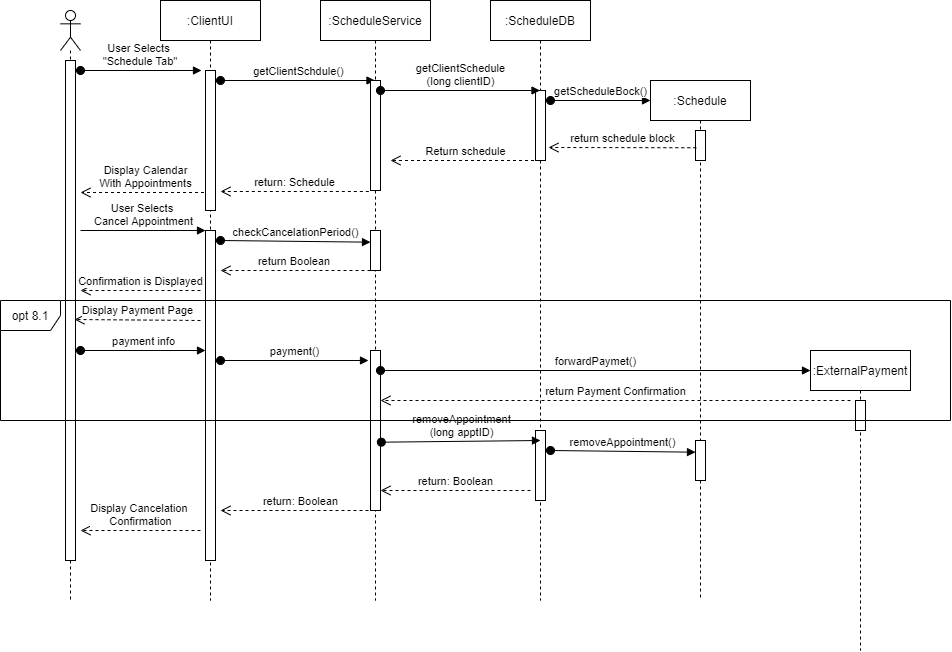
## Use Case 13: Client views their upcoming appointments



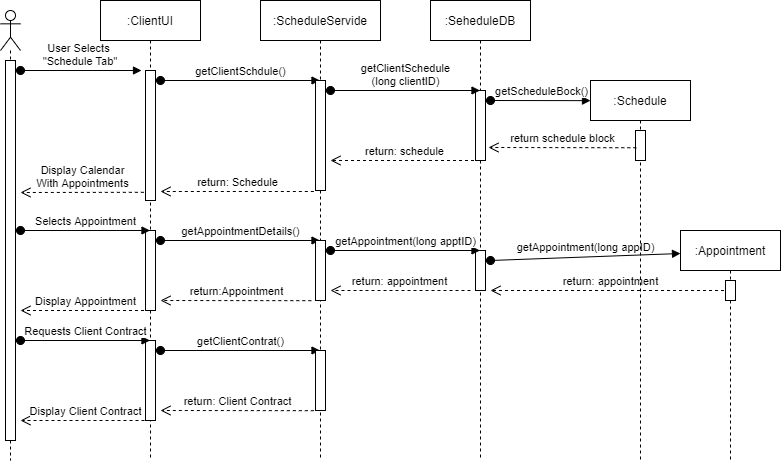
## Use Case 14: Client updates an appointment



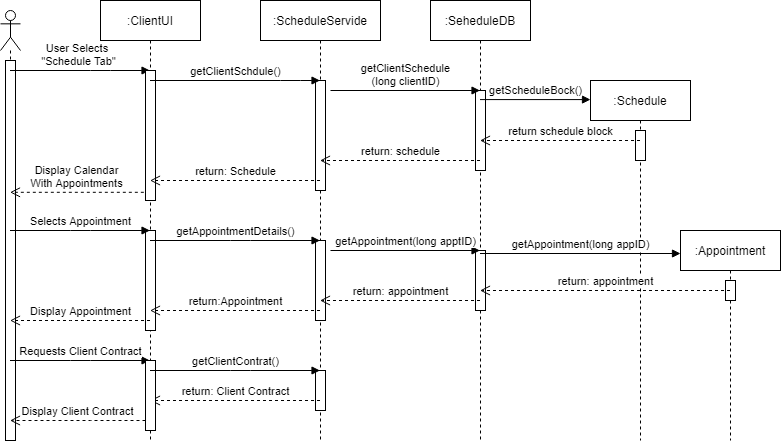
## Use Case 15: Client Cancels Appointment



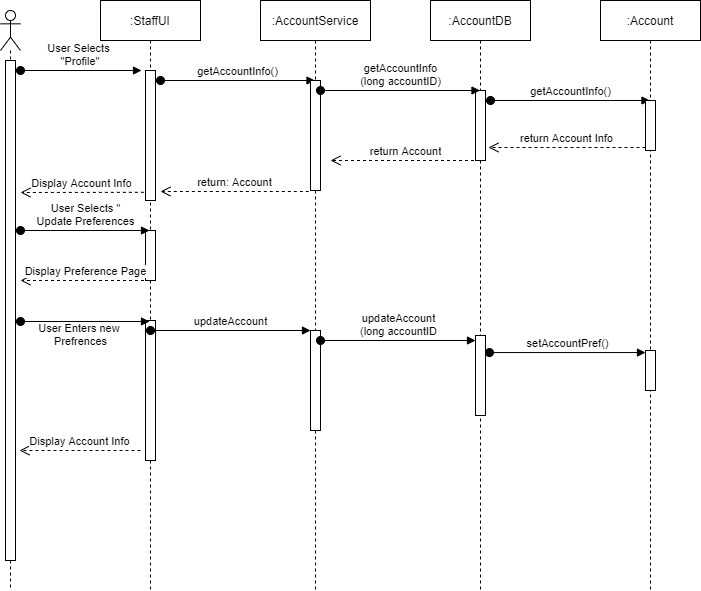
## Use Case 16: Client Views Client Contract



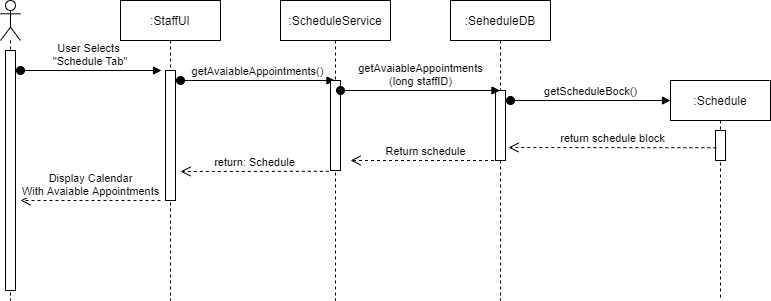
## Use Case 17: Client Prints Client Contract



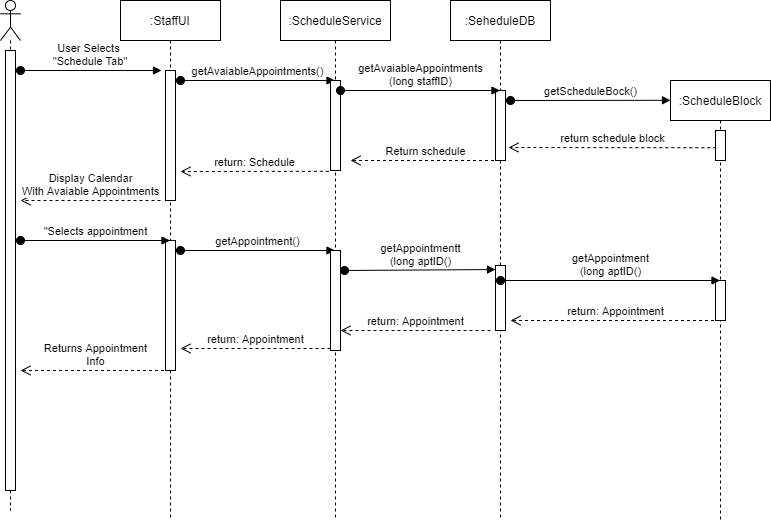
## Use Case 18: Staff Sets Work Preferences



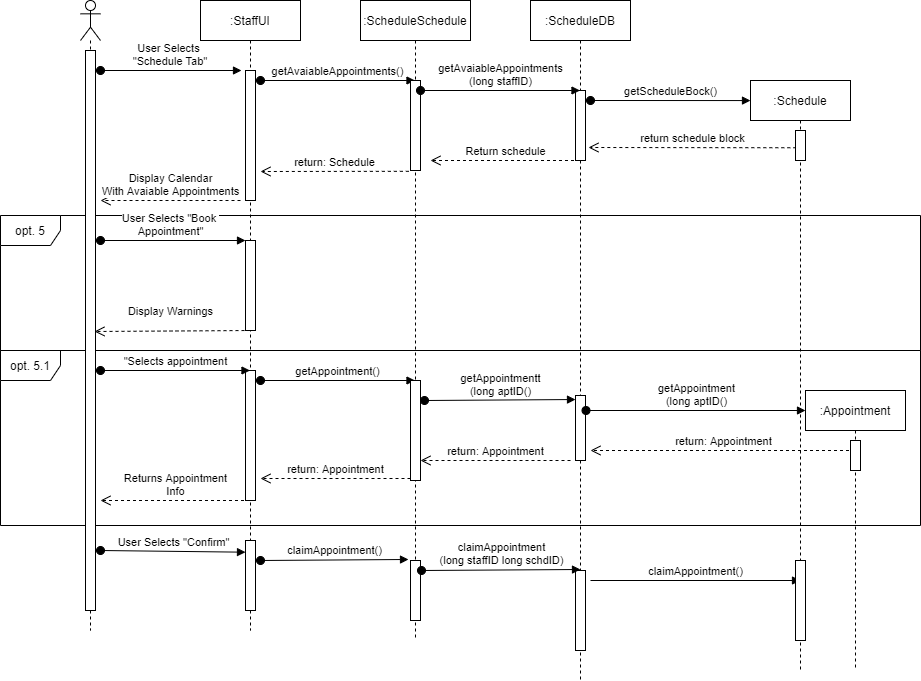
## Use Case 19: Staff Views Available Appointments



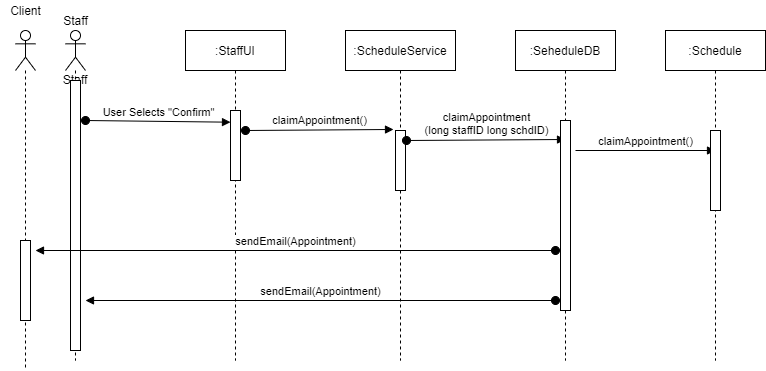
## Use Case 20: Staff Views Client and Pet Information



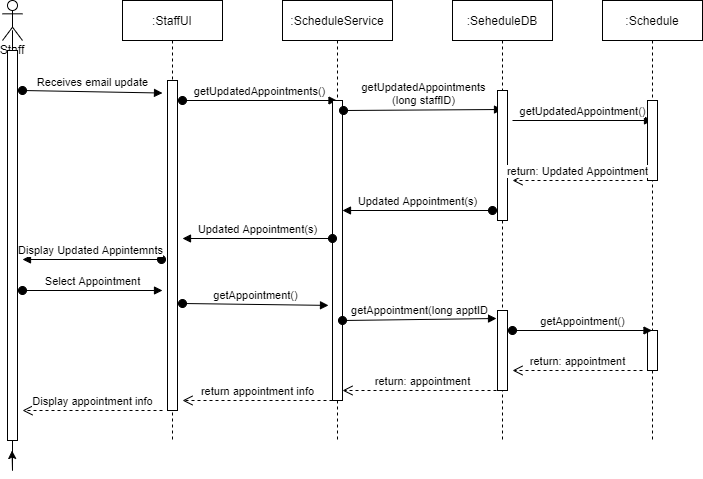
## UseCase 21: Staff Accepts Available Appointment



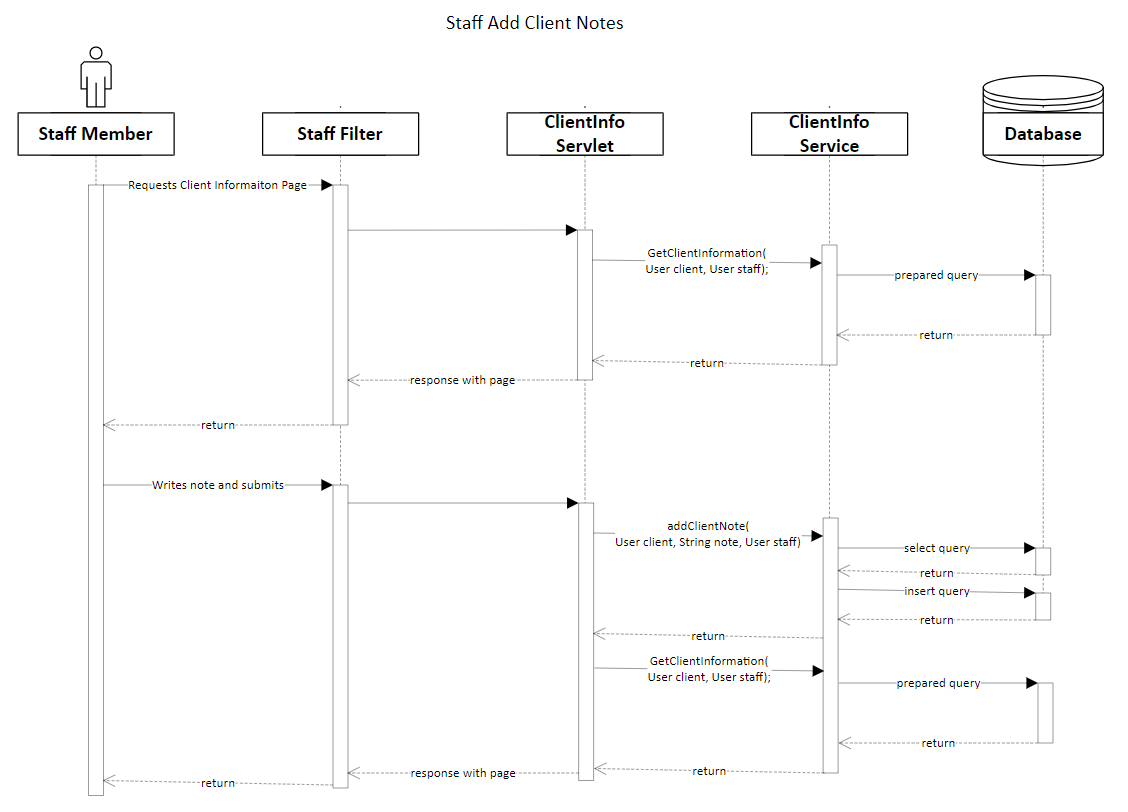
## Use Case 22: Send Confirmation Notice and Client Contract



## Use Case 23: Staff Views Appointment Updates



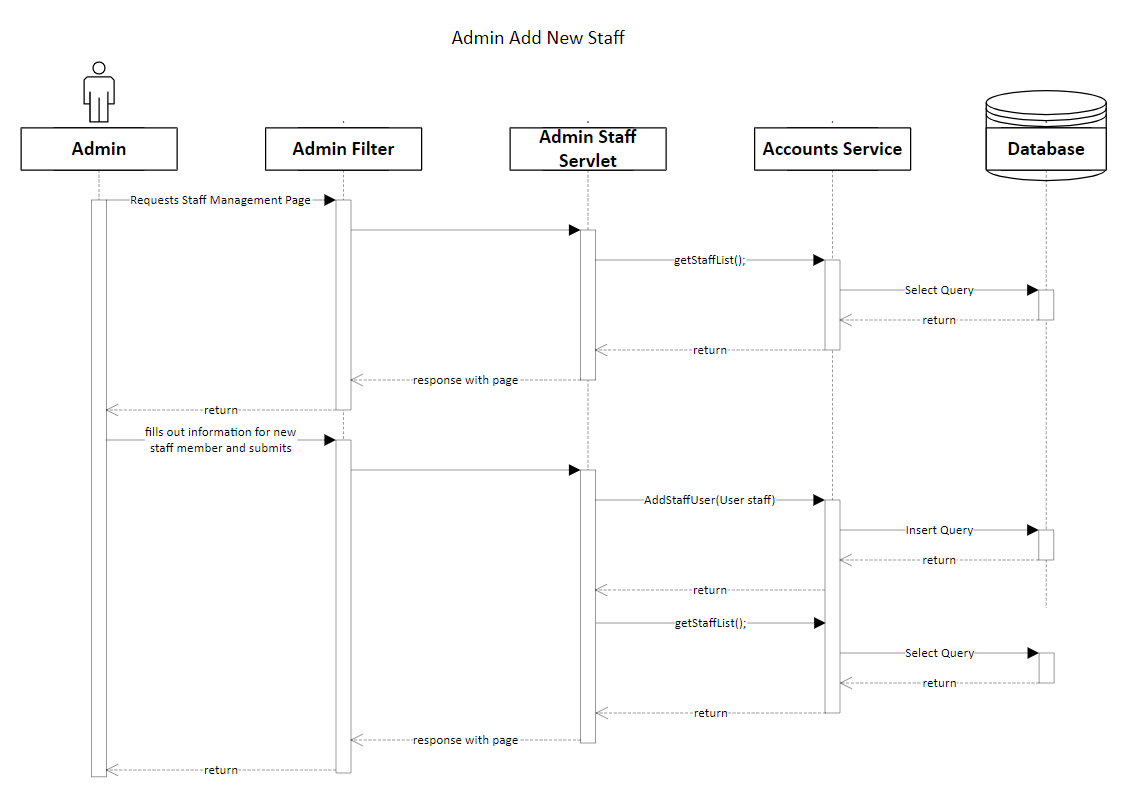
## Use Case 24: Staff Add Client Notes



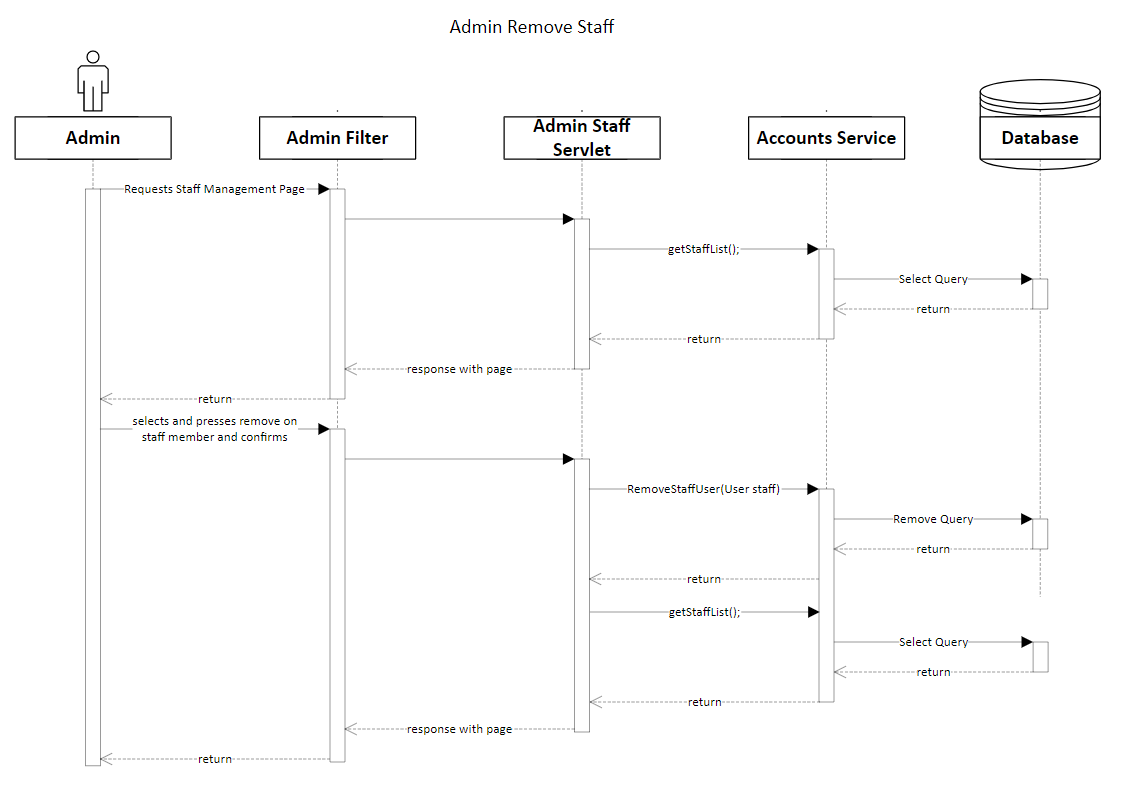
## Use Case 25: Staff View Scheduled Appointments

## 

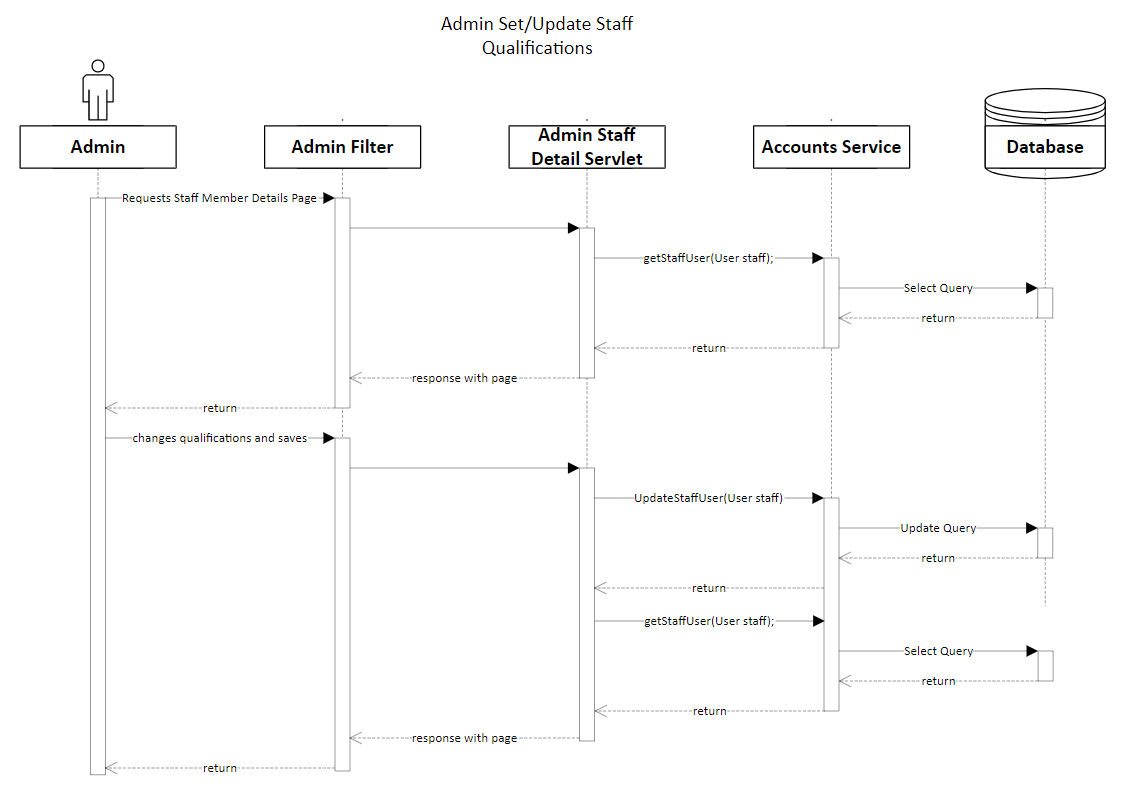
## Use Case 26: Admin Add New Staff



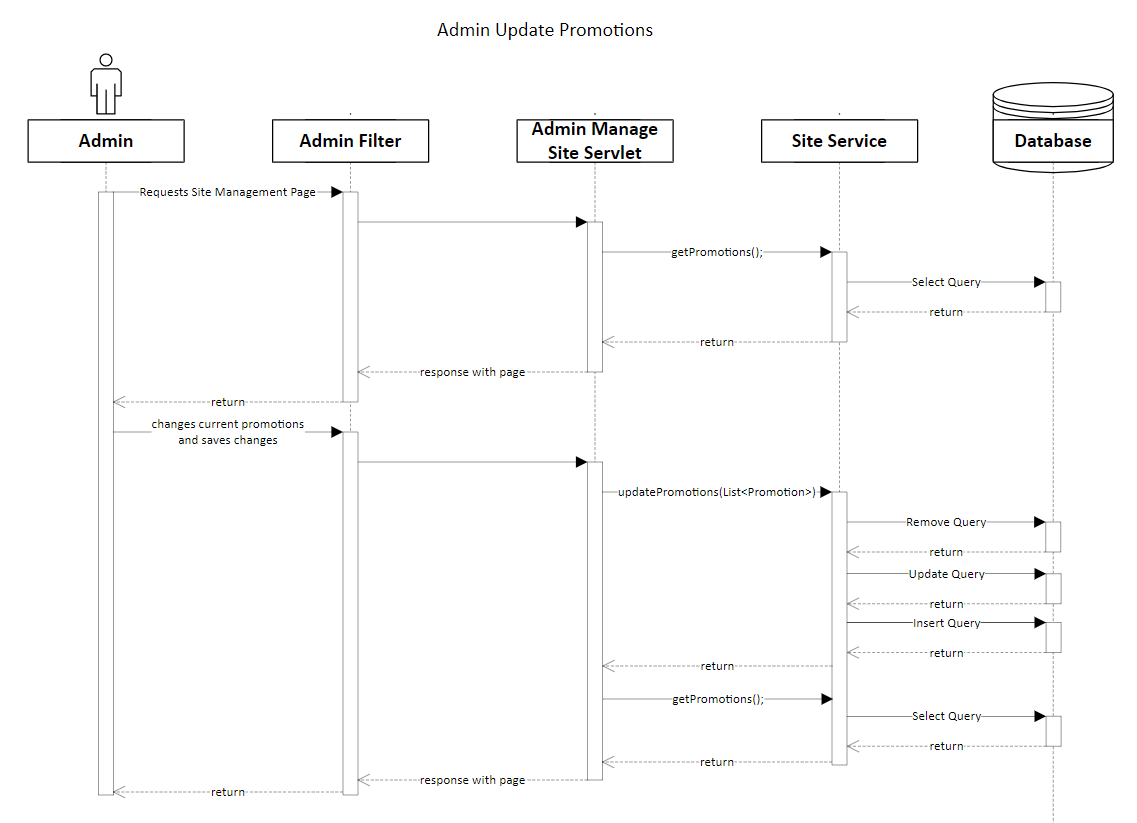
## Use Case 27: Admin Removes Staff



## Use Case 28/29: Admin Set/Update Staff Qualifications

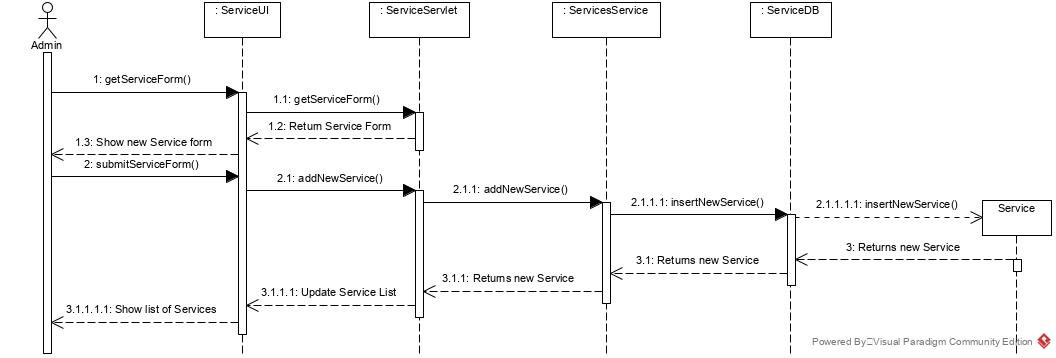


## Use Case 30: Admin Update Promotions

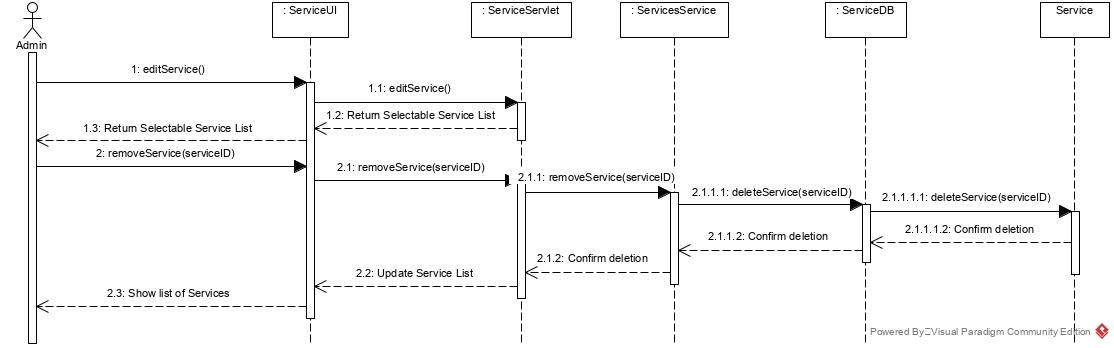


## Use Case 31: Admin emails the current promotions to the email list

## Use Case 32: Admin adds a service to the system



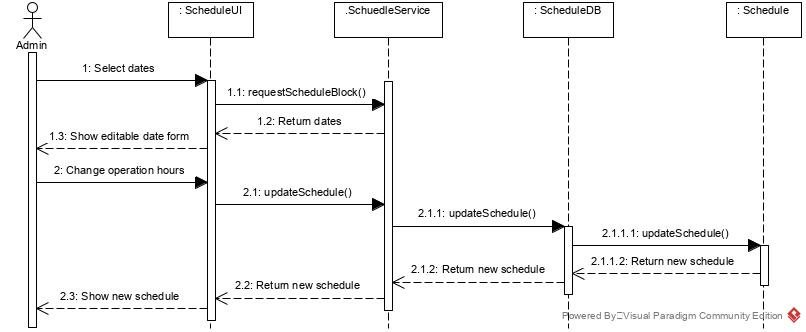
## Use Case 33: Admin removes a service from the system



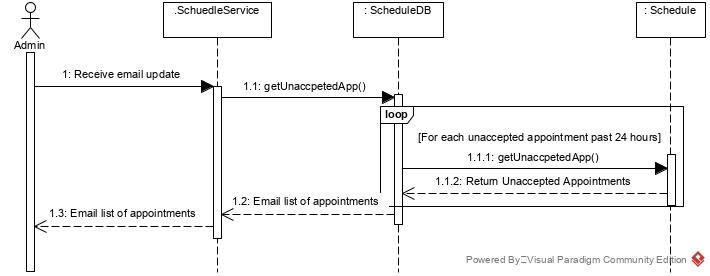
## Use Case 34: Admin updates or changes a service in the system

## 

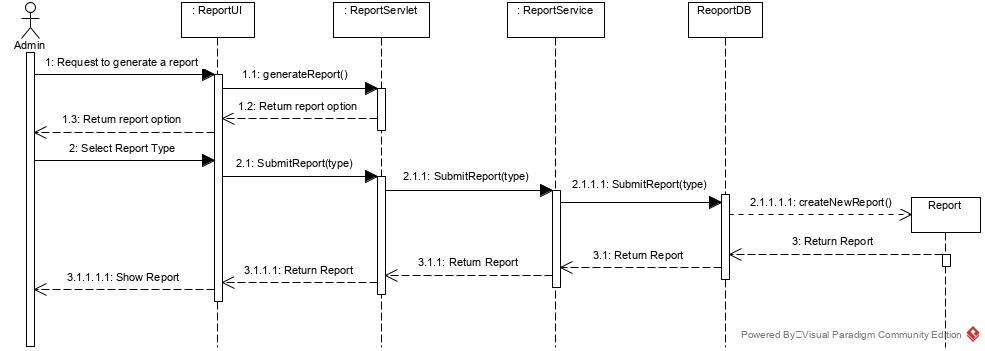
## Use Case 35: Admin changes the hours of operation or schedule time



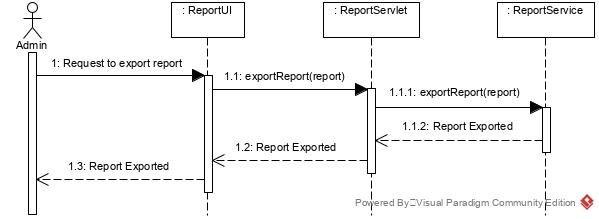
## Use Case 36: Admin is notified of an unaccepted appointment after 24 hours.



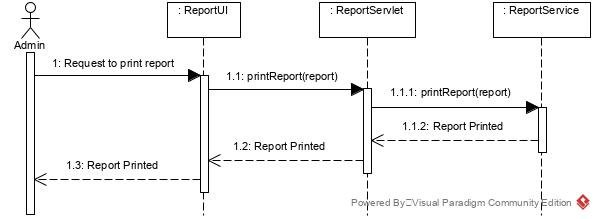
## Use Case 37: Generate a report.



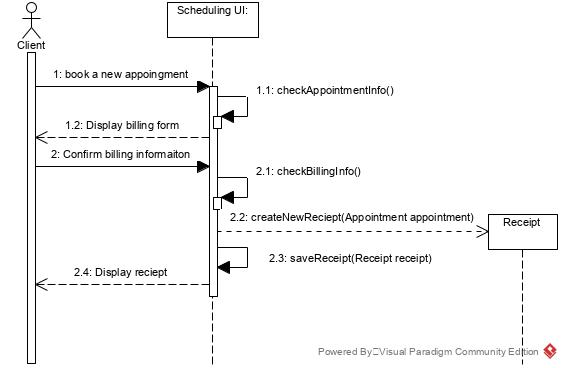
## Use Case 38: Export generated report.



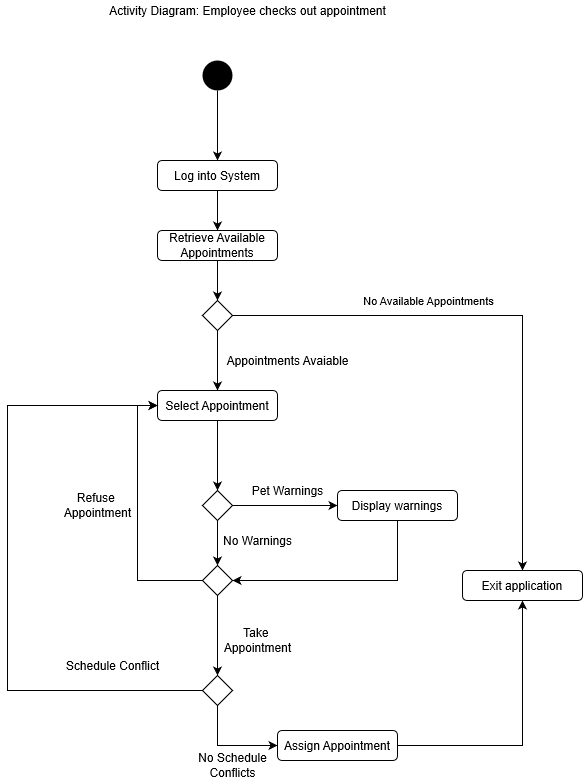
## Use Case 39: Print generated report.



## Use Case 40: Pay for Appointment.

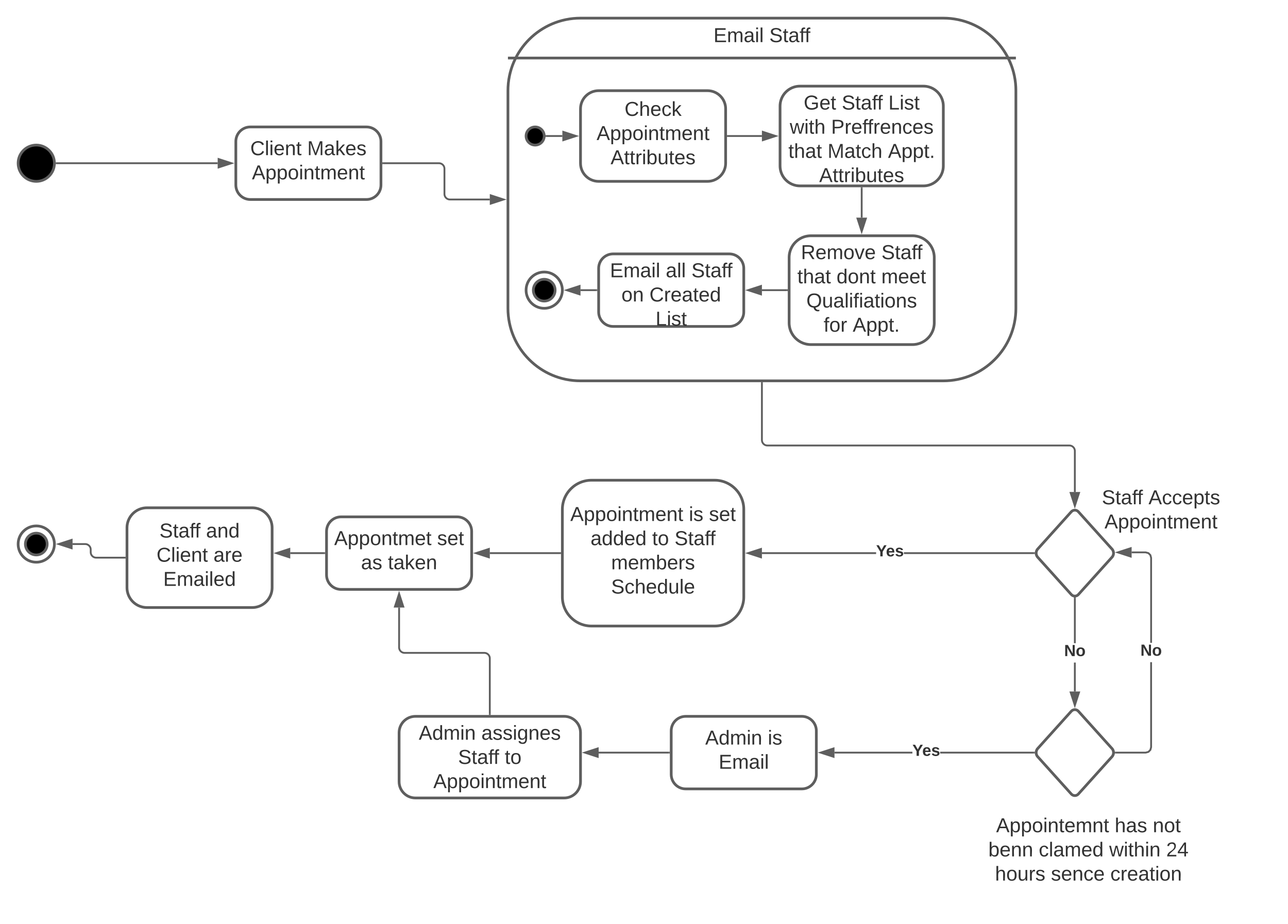


# Activity Diagram



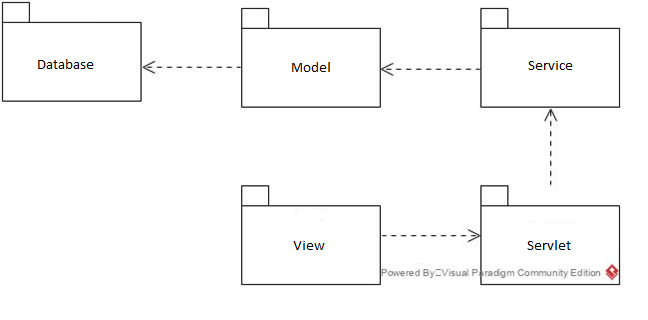
# State Machine Diagram

Client Appointment is Accepted by a Staff Member.



# System Design

## Layered Architecture



## Hardware Architecture

* Textual description
* Give an overview of the different computers, etc. that your system will need to function and how they will be related (in UML we use a deployment diagram to show this)
* How will the different parts of your system will communicate with each other? e.g. TCP/IP, HTTP, XML, JSON, JDBC etc.

## Hardware Platform

The hardware specifications of the computers used to build and maintain the system are as follows:

* **CPU:** Intel i7-7500U
* **Memory:** 16GB RAM
* **Storage:** 512GB Solid State Drive
* **Graphics:** Intel Integrated Graphics
* **Input Devices:** Keyboard, Touchpad
* **CPU:** 2.4GHz Intel Core i9 processor
* **Memory:** 32GB RAM
* **Storage:** 4TB SSD
* **Graphics:** AMD Radeon Pro Vega 20
* **Input Devices:** Keyboard, Touchpad

The hardware specifications required for the webserver running our system are:

* **CPU:** 8-core processor
* **Memory:** 16GB RAM
* **Storage:** 500GB HDD

## Software Platform

The software specifications for the programs being used to develop this system are as follows:

* **Operating System**: Windows 10 Enterprise version 10.0.17134
* **Integrated Development Environment:** NetBeans 12.1 or Visual Studio Code 1.51.1
* **Programming Language:** Java 8
* **Server Software:** Apache Tomcat 9
* **Database Management Software:** SQL Server 2019

The software specifications for the programs used to deploy the system are:

* **Operating System:** Ubuntu 19.10
* **Server Software:** Apache Tomcat 9
* **Database Management Software:** SQL Server 2019
* **Programming Language:** Java 8

Interaction Model

* Style: direct manipulation.
* Desired User Support: Error messages and error codes.
* System Feedback: Pop-up confirmation and error messages.
* Standards are in accordance to Infinite Pets current colour and design.

## Login

## 

* The Sign In page where Users can enter their username or email and password to access their personal account.
* They can also access their account though third-party services such as Google, Facebook, Instagram, and Twitter.
* The logos located in the top left and to the right of the sign in form are assets that are in current use by the client.

## Services

## 

* The Services page is where the current list of offered pet services are displayed.

## My Pets

## 

This page will display all current registered pets in the system.

* Each displayed pet will include:
* Name
* Breed
* Current Vet
* Date added to the system.
* Owner description of the pet
* The form on the left allows the user to add a new pet to their account.
* Information entered in this form:
  + Name of the pet
  + Breed of the pet
  + Current Vet
  + Current Date
  + Owner’s description of the pet (optional)
  + A photo of the pet can be uploaded(optional)

## Client Account

## 

* This page will display the information of the user's account with an option to edit the information.
* Information displayed is:
* Account owner’s name
* Birthday
* Email address
* Phone address
* Home address

## Client Appointment

## 

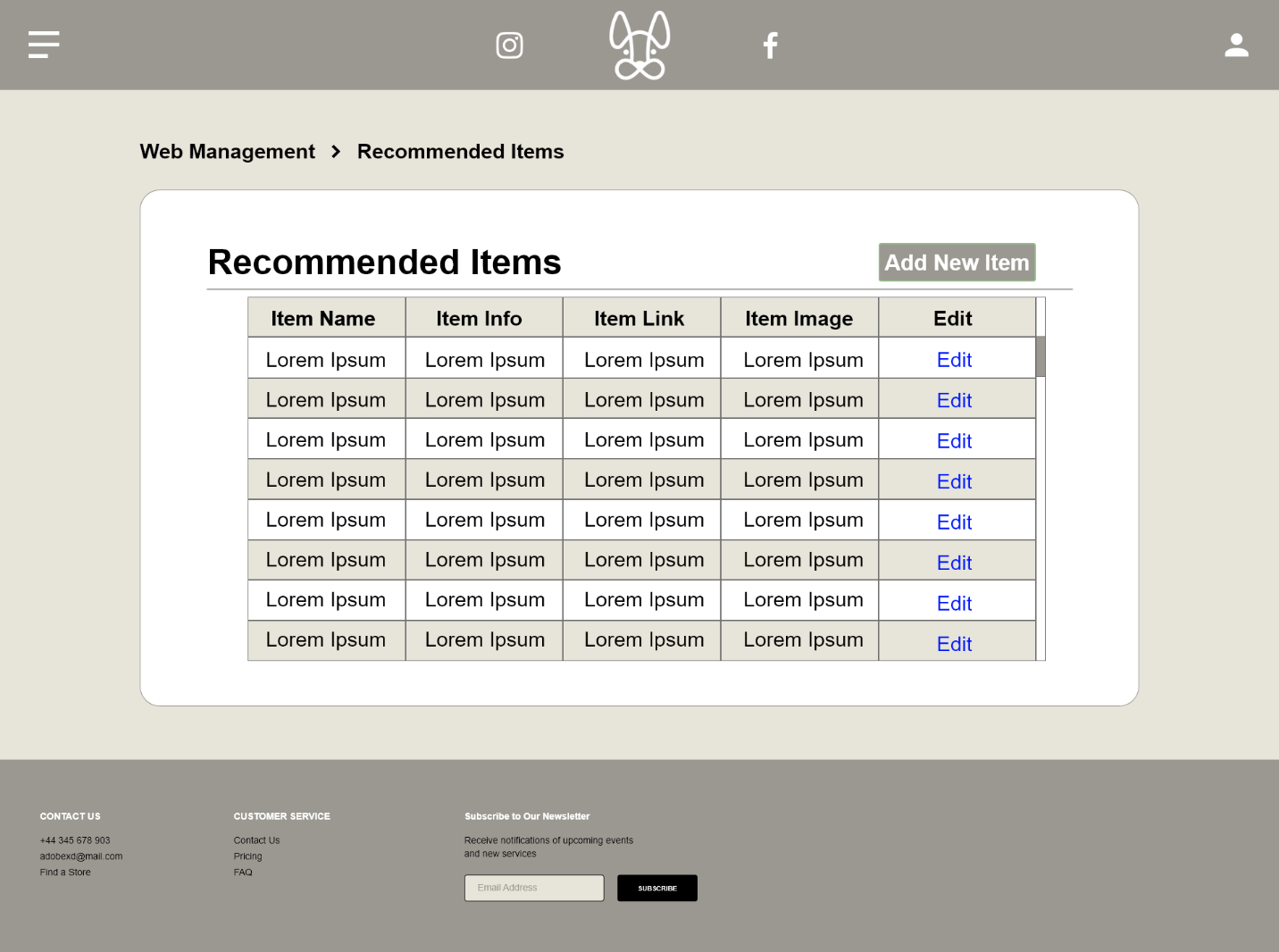
* This page will allow a Client to create an appointment.
* The Client will fill out the form of
  + Pet(s)
  + Date and Time
  + Service required
  + And the cost and option to confirm the booking are displayed on the right side of the window.

## Schedule

## 

* This page will display the Schedule of the User.
* Date to view is displayed on the left with a more detailed information of that day displayed on the right

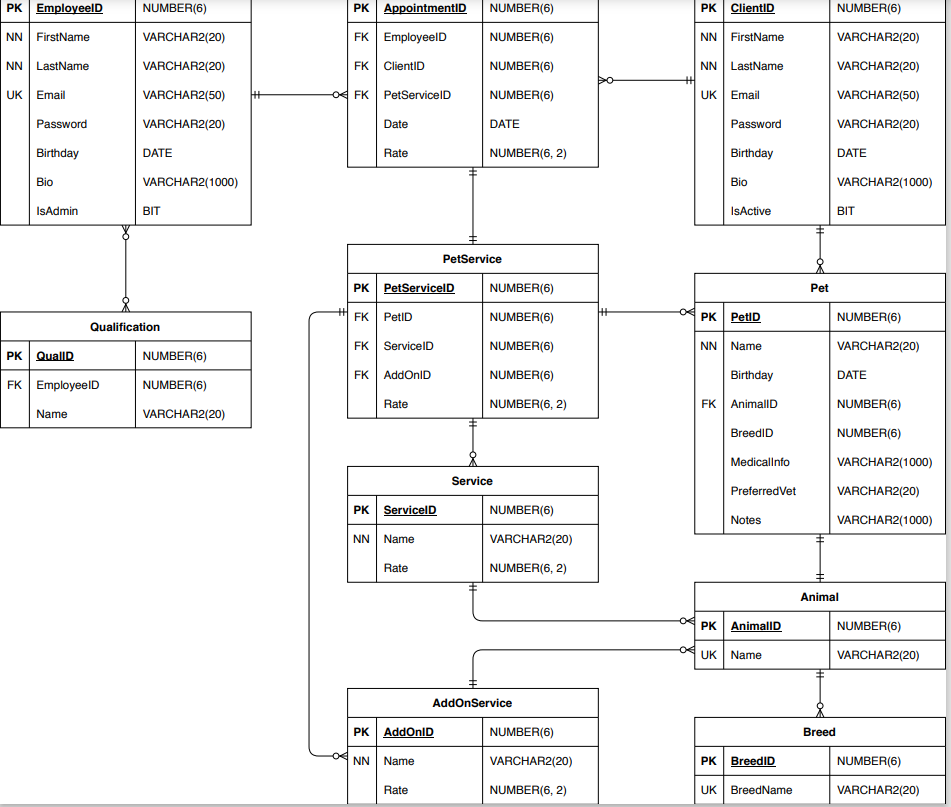
## User Interface Management



* This page display the Recommended Items page used only by the admin
* Scrollable table of all the recommended items displayed on the web application
* Admin can edit or add new items that will update and upload to the web application

# Persistence Model

* JPA will be used
* ERD



# Project Management

## Schedule

#### Milestones

* + Project Start Date: Fall Semester 2020
  + Requirements Analysis Document
    - Start Date: Sept. 8, 2020
    - End Date: Nov. 8, 2020
  + Team Constitution
    - Start Date: Nov. 6, 2020
    - End Date: Nov. 28, 2020
  + Design Document
    - Start Date: Nov. 9, 2020
    - End Date: Dec. 18, 2020
  + Project Schedule
    - Start Date: Nov. 27, 2020
    - End Date: Dec. 3, 2020
  + Client Review of Design
    - Start Date: Jan. 25, 2021
    - End Date: Jan, 25, 2021
  + Use Case Development
    - Start Date: Jan. 26, 2021
    - End Date: Mar. 12, 2021
  + Acceptance Testing
    - Start Date: Feb. 1, 2021
    - End Date: Apr. 7, 2021
  + Beta Installation
    - Start Date: Mar. 30, 2021
    - End Date: Apr. 12, 2021
  + Project Completion Date: Apr 12, 2021
  + Team Meetings will happen Tuesday or Wednesday weekly where progress reports will be due.

## Team Configuration

##### Infinite Pets

* Contact: Blair Slind
* Owner: Jenna Hall

##### Brendan Tran

* Role: Developer
* Contact: [brendan.tran@edu.sait.ca](mailto:brendan.tran@edu.sait.ca)

##### Christopher Manuel-Smith

* Role: Project Design, Main Client Contact
* Contact: [christopher.manuel-smith@edu.sait.ca](mailto:christopher.manuel-smith@edu.sait.ca)

##### Cashton Bieker

* Role: Database programmer, Front end design
* Contact: [cashton.bieker@edu.sait.ca](mailto:cashton.bieker@edu.sait.ca)

##### Riley Hiltz

* Role: analyst (Making sense of raw data of interviews, observations, etc. into organized documents)
* Contact: [riley.hiltz@edu.sait.ca](mailto:riley.hiltz@edu.sait.ca)

#### Nicholas Panos

* Role: Logic programmer (tentative)
* Contact: [nicholas.panos@edu.sait.ca](mailto:nicholas.panos@edu.sait.ca)

## Project Standards and Procedures

### Communication

The team members will communicate via emails and Microsoft Teams. The regular meeting will occur every Friday morning at 10:00, provided there are no other scheduling conflicts. The longer meeting will commence in the afternoon after PROJ-304 course.

### Software

#### Tools

* Java will be the language used to build the system.
* We will use PL/SQL for the database of the system.
* We will use the embedding code methods by Google in order to integrate the calendar into the website.
* Lucidchart and draw.io will be used for diagrams.

# Glossary

Cases

Calibri: The font used throughout the document 4

Client: A customer of Infinite Pets 7

draw.io: A software that allows for the making of charts and diagrams. 16

Infinite Pets: A full service pet care company that serves Calgary, Springbank, and Cochrane. 5

Java: A programming language commonly used for application development 16

Lucidchart: A software that allows for the making of charts and diagrams. 16

PL/SQL: A database programming language developed by Oracle 16

System: The software that is being developed 5

# Index

Activity Diagram 2, 14

admin 9

Administration 2, 9

Administrator 10

Client 2, 3, 6, 8, 10, 16, 20, 21

Database 6, 16

Diagrams 2, 4, 5

Infinite Pets 5, 6, 10, 16, 23

Maintainability 2, 9

Project Management 2, 16

Requirements 2, 4, 6, 7, 8, 9, 10, 16

Sequence Diagrams 2, 12

Staff 2, 8, 9, 10

State Machine Diagram 2, 15

system 5, 6, 8, 9, 10, 17

Use Case 2, 7, 8, 16

website 8, 10, 17

# Appendix A: Data Dictionary

## Appointment

|  |  |  |
| --- | --- | --- |
| **Field/Method** | **Parameters (if apply)** | **Description** |
| appointmentID: int |  | The appointment ID. Cannot be null. |
| Client: ClientAccount |  | The client associated with the appointment. Cannot be null. |
| time: scheduleBlock |  | Cannot be null. A time represented by ScheduleBlock. |
| Service: Service |  | The Service which is associated with appointment. |
| Staff: StaffAccount |  | The staff who is responsible for handling the appointment. |

## Service

|  |  |  |
| --- | --- | --- |
| **Field/Method** | **Parameters (if apply)** | **Description** |
| serviced: int |  | The ID for the service. Cannot be null. |
| serviceName: String |  | The string representing the service name. Cannot be null. |
| Description: String |  | The description of the service. Cannot be null. |
| basePrice: double |  | The base price of the service. Cannot be null and > 0.00. |
| numOfAnimals: int |  | The number of animals. Cannot be null and must be > 0. |
| requiredQualificationID: int |  | The required qualifications represented by ID. Cannot be null. |

## Discount

|  |  |  |
| --- | --- | --- |
| **Field/Method** | **Parameters (if apply)** | **Description** |
| discountID: int |  | The discount ID. Cannot be null. |
| serviceId: int |  | The service ID. Cannot be null. |
| promoID: int |  | Cannot be null. ID for Promotion. |
| Discount: double |  | The amount of a service that would be discounted. Cannot be < 0. |
| discountType: char |  | The discount type represented by char. |

## Promotion

|  |  |  |
| --- | --- | --- |
| **Field/Method** | **Parameters (if apply)** | **Description** |
| promoName: String |  | The name of promotion. Cannot be null. |
| promoDescription: String |  | Cannot be null. The promotion description. |
| startDate: Date |  | The date of which a promo starts. Cannot be null. |
| endDate: Date |  | The end date of which ha promo ends. Cannot be a date before start date. Cannot be null. |
| Image: Image |  | The image of the promotion. |
| Featured: Boolean |  | A Boolean representing whether this promo is being featured. |
| promoID: int |  | The ID of the promotion. Cannot be null. |

Account

|  |  |  |
| --- | --- | --- |
| **Field/Method** | **Parameters (if apply)** | **Description** |
| Id: int |  | The ID from the database. Used to make all account unique. |
| firstName: String |  | The first name. Cannot be null. |
| lastName: String |  | The last name. Cannot be null. |
| email: String |  | An email address of an account. |
| PasswordHash: String |  | Password hash for the account. Cannot be null. |
| passwordSalt: String |  | Password salt to use with password hash. Cannot be null. |
| Bio: String |  | Bio of the client/staff. Can be null. |
| Role: int |  | Role to identify the privilege access. Default is 0, which is client level. |

## ClientAccount

|  |  |  |
| --- | --- | --- |
| **Field/Method** | **Parameters (if apply)** | **Description** |
| Location: Location |  | The location of client’s address. |
| Pets: Pet[] |  | The array of Pet. Can be null. System will initialize the array if necessary. |
| Appointments: Appointment[] |  | The array of Appointment. Can be null (meaning empty). |

## StaffAccount

|  |  |  |
| --- | --- | --- |
| **Field/Method** | **Parameters (if apply)** | **Description** |
| empLocation: Location |  | The location of where employee will be working. |
| Appointments: Appointment[] |  | The array of Appointment. Holds all appointments that the staff have accepted. |
| regClients: ClientAccount |  | The array of regular clients. Can be null (meaning empty). The system will initialize when necessary. |
| Schedule: Schedule |  | The schedule of a staff. This tells what time/block the staff is able to work on. Uses the appointments[] for determining the availability.. |
| servicePreference: int[] |  | Staff’s preferred service. Contains ID for service. |
| staffQualifications: StaffQualification[] |  | Cannot be null. Contains all staff qualifications. |

## StaffQualification

|  |  |  |
| --- | --- | --- |
| **Field/Method** | **Parameters (if apply)** | **Description** |
| qualificationID: int |  | The ID for the qualification. Cannot be null. |
| QualificationName: String |  | Name of the qualification. Cannot be null. |
| QualificationDescription: String |  | The description of the qualification. Cannot be null. |

## Location

|  |  |  |
| --- | --- | --- |
| **Field/Method** | **Parameters (if apply)** | **Description** |
| locationType: char |  | Cannot be null. Character ‘E’ represents the location where employee is working at. ‘R’ represents the residence (address) of a client. |
| postalCode: String |  | Cannot be null. Represents the postal code (area). The sequence of characters would be as follows: ‘A1A1A1’ |
| lineAddress: String |  | Cannot be null. Represents the address line. Format as follows: “123 Street\_name road\_type quadrant\_section |
| City: String |  | The city name. Cannot be null. |
| Province: String |  | The province name. Cannot be null. |
| Country: String |  | The country name. Cannot be null. |

## Schedule

|  |  |  |
| --- | --- | --- |
| **Field/Method** | **Parameters (if apply)** | **Description** |
| scheduleBlock: ScheduleBlock[] |  | The schedule block. Can be null (represents empty, not available for work) |
| scheduleID |  | The ID for schedule. Cannot be null. Made to make it unique. |
| getAvailableSvhedule(appointments: Appointment[]: ScheduleBlock[] | Appointments: Appointment[] is the array of Appointment which is held in StaffAccount. | Returns the available schedule block. The logic is unknown at this time,however, the appointments which is held in StaffAccount will be used to determine if the staff is available.. |
| addScheduleBlock(block: ScheduleBlock): boolean | Block: ScheduleBlock | Adds the schedule block. The staff will be able to add a time range, which tells the system what time the staff is available for work. |
| removeScheduleBlock(blockID: int): ScheduleBlock | blockID: int | Removes the schedule block, which removes the availability time for the employee to work. Returns the ScheduleBlock that had been removed. |

## ScheduleBlock

|  |  |  |
| --- | --- | --- |
| **Field/Method** | **Parameters (if apply)** | **Description** |
| timeStart: Date |  | The start time represented by Date. Cannot be null. |
| timeEnd: Date |  | End time represented by Date. Cannot be null. Can’t have a date before start date and must be > 5 minutes difference. |
| blockID: int |  | The ID of the schedule block. Cannot be null. |

## Pet

|  |  |  |
| --- | --- | --- |
| **Field/Method** | **Parameters (if apply)** | **Description** |
| Breed: String |  | The breed of a pet. Cannot be null |
| Birthday: Date | **YYYY/MM/DD** | Birth date of the pet. Cannot be null. |
| animalType: String |  | Cannot be null. Indicates which animal, ie. Cat, dog, turtle… |
| Sex: String | **Male, Female, Altered** | The sex of the pet. Cannot be null. |
| Picture: jpg |  | An image or photo of the pet. Cannot be null. |
| StaffNotes: String |  | Notes about this animal, can be warning or notes about its personality. |
| Name: String |  | The name of a pet. Must be non-null. |
| medicalInfo: String |  | Can be null, information about the medication. |
| Owner: ClientAccount |  | The owner of the pet. Cannot be null. |

## GalleryImage

|  |  |  |
| --- | --- | --- |
| **Field/Method** | **Parameters (if apply)** | **Description** |
| Featured: Boolean |  | Whether the image is featured. False by default. Cannot be null. |
| pet: Pet |  | The pet. |
| Image: Image |  | The image of the pet. |

# References

[1] Infinite Pets, N/A, [Online]. Available: <https://www.hallidaypets.com/>. [Accessed: Sept. 27, 2020]