COMP 3059 – Capstone Project I

Software Requirements Analysis and Design Assignment

Motley Zoo Service Gateway

By
Maksim Sen 101225087
Ming Yang 101010466
Gustavo Beltran 101234979
Raiyan Rofiquzzaman 101287844
Vanja Vego 100535769

Table of Contents

1.0 Introduction

Motley Zoo Gateway Service is aiming to replace the costly third-party service from "Time to Pet" to a full in-house system/web app. Motley Zoo Gateway Service contains three sub-systems for customers, employees, and managers. The whole system will be based on a microservice architectural style. Each sub-system has features that fit Motley Zoo's business needs and build on separate services which are running in their own process and communicating with lightweight mechanisms, such as HTTP resource API.

The customer sub-system will include the following features: High-security encryption, Overview of available time slots, setting the desired dog-walker, setting subscription time slot for services, setting single time slot for services, setting profile info, payment gateway, an overview of reports from dogwalkers, feedback on the specific service instance, and feedback of overall business.

The staff sub-system will include the following features: High-security encryption, Overview of booked schedule, setting schedules (if enabled, member-by-member basis depending on contract), setting workload, an overview of hours worked, writing reports to the owner, writing reports to clients, and quick-format communications with clients.

The manager sub-system will include the following features: High-security encryption, sub-systems based on user type, scheduling services, user control/permissions, profile editing, business reporting and pdf export, scheduling request approval, overriding scheduling ability, access to all reports site-wide, and API webhook setup and toggling.

1.1 Purpose

The purpose of this Software Requirements Specification is to describe the Motley Zoo Replacement Gateway. This will include the scope which will describe the functionalities of the system. In addition, it will also include the details of the system, and diagrams for the specified system.

Motley Zoo is an organization that provides walks for pet owners. The purpose of our project is to make some well needed improvements to the admin and payment services and improve communications between the staff and the client. This project will also help make the stakeholder's outsourced project less expensive to the owner.

1.2 Scope

Motley Zoo Gateway Services is planned to be developed as a Single Page Application using the MERN stack with proprietary microservices in aims to supply functionality for an existing website and replace their current services portal.

Through ZGS, pet owners can provide information about their pets and request dog walker services and unlock features such as requesting and scheduling a dog walk, manage their payments, view account activity, retrieve invoice history, be able to print them and be able to create and edit their pet and owner profiles. Pet owners can also send communication to Motley Zoo to request for changes in their service or ask questions. Staff will also have access to enter their availability to work and request changes in their schedules as well as being able to send messages to their employer. Staff can setup their own profiles that can be shared with customers.

Administrators will have access to employee's timesheet and are able to make any type of changes to employee's profiles and communicate with clients and staff. Admins can also access customer's files and retrieve any information needed, for example, copies of past invoices, due invoices, services history, etc.

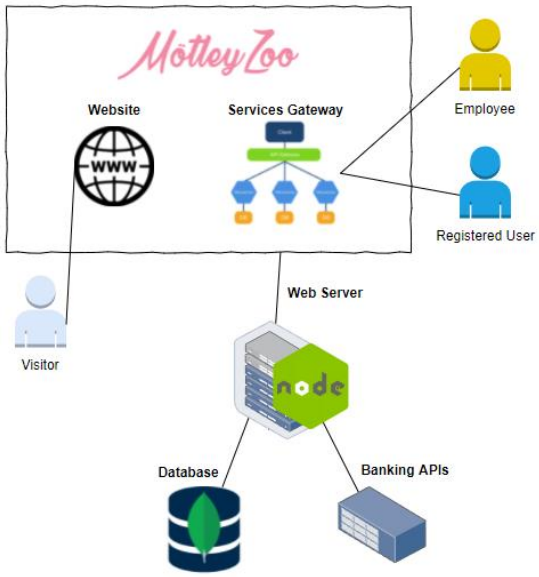
Our software must be implemented with an existing website system, and we are replacing a portal that currently provides the above-mentioned services to Motley Zoo. We will make sure all their features are replicated correctly and if possible, find ways to improve our delivery. This might pose a minor constrain in having to deal with people outside our development team to supervise the successful integration of our software with the existing website.

2 System Overview

This section will give an overview of the whole system. The system will be explained in its context to show how the system interacts with other systems and introduce the basic functionality of it. It will also describe what type of stakeholders that will use the system and what functionality is available for each type. At last, the constraints and assumptions for the system will be presented.

2.1 Project Perspective

Our system will be built using microservices architecture. We want to remain open to working with any type of framework that best suits our application, nonetheless, we have certain frameworks in consideration that we think could suit our needs for developing this software. The service gateway is an integral part of Motley Zoo's business model and is going to be running parallel with their main website. This means that we will focus solely on developing the service gateway. Their main site's purpose is to showcase their services and advertise their company and our gateway is what a registered customer uses to request services.



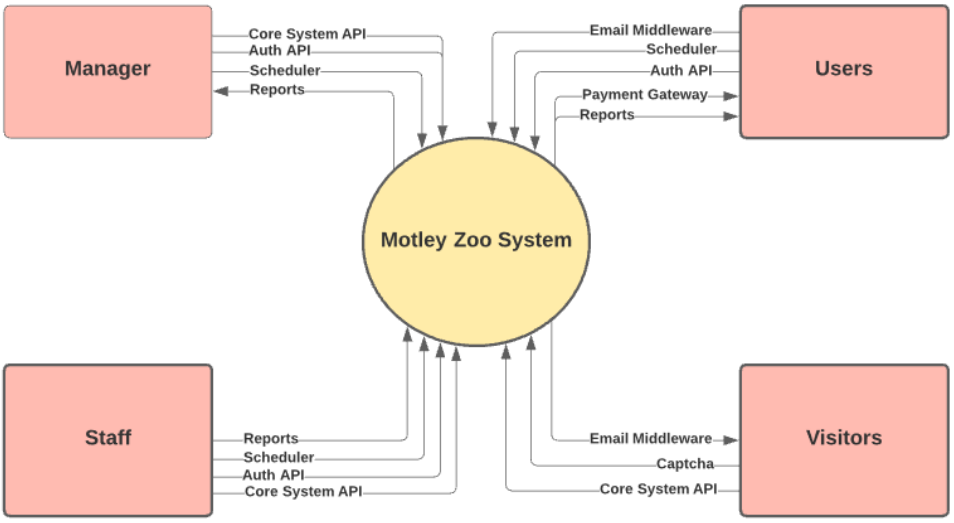
Visitors will browse Motley Zoo's website and once they are ready to request a service, they will register an account with our platform to unlock access to MZ's pet walks, pet adventures and cat visits. These services require the pet owner to create a profile for themselves and for their pet. Creating a profile for their pet allows Motley Zoo to develop a database that can be used to research for better pet care. Pet owners will also be able to send communications to the company about any concerns or questions. Staff members will have the ability to request changes to their schedule. Higher management will have access to internal operations such as changing employee schedules, have access to change details about services, produce a report based on various KPIs and have access to customer and employee account management tools.

Service Gateway will communicate to a scalable database to store all the information about customer's interactions with the company. This database will only be connected to our system and have no ties to their website. Our gateway will also communicate with 3rd party banking APIs to perform actions, for example, process credit cards and offer other methods of payment. This software works in parallel with Motley Zoo's existing website, to this effect, interactions with their webmaster is a possibility.

One of the constraints in our system is our dependency on 3rd party database hosting and web server hosting to deploy our APIs. We will overcome this challenge by choosing the appropriate database company that best suit our forecasted activity of users to our system. We want to have a bit of headroom from the start in terms of load capacity but not so much as to sacrifice the cost of paying for the server and database hosting. However, one of our mayor challenges is security. Because of this, we will design secure solutions and test for vulnerabilities exhaustively to ensure transactions are safe.

The Motley Zoo Service Gateway will abide by the highest security practices and follow all the regulations regarding PCI DSS compliance and the Consumer Privacy Act.

2.2 System Context



2.3 General Constraints

- Payment gateway may be switched out from Stripe to any other payment gateway if desired by principal stakeholder at a later stage
- Review and Promotional API may not be used, and may be omitted from this project at the time of development based on main site webmaster's co-operation, owners change of mind, or time constraints (this is considered a low-priority functionality by the owner)
- The scope of this project currently covers only the dog walking service as per the owners wishes. This may change later as business-related matters on those services get ironed out more.
- Software development dependencies (Node packages) may change based on development teams wishes as the development commences and throughout several sprints of development stage of project.
- With the architecture being microservice based, some of those microservices may be developed using any other language and packages than specified that the developer of said microservice is most comfortable with. All modern languages support REST through HTTP, so this will be decided upon in development stage.

- Testing and deployment in actual production system would be executed as described in section 2.4, however due to costs of setting up multiple environments with each containing multiple servers, it may be reduced to local environments and deployment for capstone project purposes. This constraint may be lifted if team manages to step around payment through AWS free-tier systems and/or Travis-CI test deployment.

- If the principal stakeholder chooses to use the system, the cost of cloud servers would be covered by Motley Zoo. This should still cut their monthly costs by at least half and remove the additional charge of $10 US per staff member, but naturally there is no guarantee of this happening.

2.4 Assumptions and Dependencies

Assumption	Reasoning
Specification	
Our site will integrate seamlessly with existing code from 3rd party company	Since the main site is largely a static front page, with the only connection to the system being a hyperlink, this will be pretty easy to implement. The system will have an option (low priority), to enable a couple of APIs for that site's webmaster to implement into their source. Outside of that and the link to the system, this project is largely independent from that one.
The new gateway will perfectly replace the current system used by the client	Single change on main site, the hyperlink will no longer lead to Time to Pet, but to a login gateway under the Motley Zoo domain
The gateway system will have documentation that will further provide details about its capabilities.	Basic training documentation. May also be replaced with one or more training videos and/or FAQ section if stakeholder prefers. The system is not too extensive and is fairly straight-forward in terms of functionality but would still require a concise training documentation or guide.

Design	
System will be designed with expandability in mind	The system will be largely based on microservices, and may use a monolith structure as it's backbone, particularly in terms of frontend. The microservice backend aspects will be largely based on RESTful APIs. This will enable a wide degree of loose coupling, all of which can be replaced and expanded across multiple frameworks and even languages.
System will be developed to be both computer and mobile friendly	The UI will adapt based on screen size and screen orientation. This is essentially a matter of setting up CSS styling to account for aforementioned factors. A menu may be fully shown on a full screen computer monitor browser, but would be moved into an expandable hamburger menu configuration on a vertically oriented mobile screen or a half-expanded browser.
Sub-systems based on user, staff member or administrator will be closely related, however each version will have its own features and UI aspects and views that are related to role type and permissions	Users should not have functionality of staff members. That same way, staff members and users should not have functionalities of admin account. This will be enforced with permissions and for security reasons, may also include a separate non-exposed login screen for staff members and admin.
Implementation	
The software source code will be easy to understand and easy to change	The compartmentalization and loose coupling nature of microservices will enable the team to work on features and aspects independently with diminished risk of breaking each others code.
Relevant development process will be communicated to the stakeholder on an agreed upon schedule or with major updates or uncovered concerns that have not been accounted for in prior stages of design	Stakeholder communication must be balanced. It is important to keep them in the loop, but it's also equally important to keep in mind that they have other responsibilities with their business and personal lives. As such communications will be limited to agreed upon schedule or exceptionally, when appropriate. Specifications and requirements are already agreed upon, so the team understands what needs to be done. Aforementioned exceptions include cases such as newly discovered factors which may need direction or clarification.

Constraints will be handled in a realistic and organized manner and well documented depending on the nature of constraint	The process of agile development and nature of CI/CD along with task tracking pretty much covers most of these aspects. Any exceptional issues can be resolved through reporting and meetings.
Testing	
System will be tested for bugs on several levels, and any found bugs will be corrected	Software testing will be done through manual testing and automated API testing. Infrastructure testing will be done through automated server load tests.
Testing will be extensive before overall production deployment, and the system will be replicated with staging and testing versions for CI/CD upon deployment	Standard assumption with any developed system. System will be thoroughly tested, and any bugs found will be stomped out. Any subsequent bugs will be handled through CI/CD on testing and staging replicas.

Dependencies

The Databases will be either based on a NoSQL solution such as MongoDB, or an SQL database such as MySQL or MariaDB. Upon further sprints and systems analysis and design stages, in particular database design stages, one or the other, or perhaps a combination of both will be chosen.

Development dependencies include various Node packages; however, this list cannot be finalized at this time, and will be narrowed down as sprints progress further. Some packages/libraries under consideration are:

Package/Library Name	Description	Use
Axios	HTTP Client	Sends web requests
Express	Server	Listen and serve web requests
bcrypt.js	Encryption and decryption package	Secure encryption and decryption of sensitive information like passwords, payment information
JWT	JSON Web Tokens	Secure authorization and permission storage client-side, and decryption and matching server-side
nodemailer	Mailing package	Emailing package for any emails the system sends out

flatpickr	Date and time picker package	Package for building out scheduler calendar
sequelize	SQL ORM	Sequel object relational mapper to convert db tables and db CRUD to OOP
Mongoose	MongoDB ODM	MongoDB object document mapper to convert db documents and db CRUD to OOP
PDFkit	PDF generation package	Used for templating and creation of PDF documents
Nodemon	Development dependency package	Used for testing/development server
Stripe API	Stripe services API	Payment gateway

3.0 Functional Requirements

3.1 Functional Requirements as Use Case Descriptions

3.1.1 Use Case Description Table of contents

Criterion	Description	ID
Register User Account	A user must be able to register an account to use Motley Zoo services.	1
Reset User Password	A user must be able to reset their password.	2
Login User Account	A user must be able to login with their credentials.	3
View User Account	A user must be able to view their own account.	4
Edit User Account	A user must be able to edit their own account	5
Delete User Account	Users must be able to delete their own accounts. Admins must have the ability to delete user accounts.	6
Schedule Single Time Slot	Users must be able to schedule time slots for dog walking.	7

Schedule Daily Time Slot	Users must be able to schedule daily time slots for dog walking.	8
Cancel Scheduled Appointments	Users must be able to cancel their scheduled dog walking appointments.	9
User Report on Dog Walk	Users must have the ability to provide reports based on their experience with a dog walking appointment.	10
Setting Desired Dog Walker	Users must be able to request a specific dog walker for their appointment	11
User Viewing Reports from Dog Walking	Users must be able to view reports left by the dog walker after a completed appointment. Administrators must be able to view user/staff reports.	12
User Write Business Review	Users must have the ability to write a review after a successful appointment.	13
Admin Quick Register User Accounts	Admins must be able to quickly register new user accounts.	14

Admin Register Staff Accounts	Admins must be able to register new staff accounts.	15
Admin Set Permissions	Admins must be able to set user permissions however they see fit.	16
Admin Modify Accounts	Admins must posses the ability to edit user accounts.	17
Admin View & Export Reports	Admins must be able to view and export PDF reports	18
Admin Schedule Request Approval	Admins must be able to approve or deny staff scheduling requests.	19
Admin Override Staff Scheduling	Admin must have the ability to override set schedules of staff members.	20
Admin Setup API Webhooks	Admin must be able to enable/disable API webhooks on the main website.	21
Staff Member View Booked Schedule	Staff members must be able to view what schedules they have booked.	22
Staff Set Schedule of Availability	Staff members must be able to set their own schedules based on their availability.	23
Staff Set Dog Handling Limit	Staff members must be able to limit how many dogs they're able to handle per appointment.	24
Staff See Hours Worked and Paid Off	Staff members must have the ability to view how many hours they have worked and how much they were paid.	25
Staff Reports to Admin	Staff members must have the ability to write a report about their client to the admin.	26

Staff Reports to Client	Staff members must have the ability to report directly to the client.	27
Staff Quick Chat	Staff members must have the ability to quickly chat with their client or owner.	28

3.1.2 Use Case Descriptions – Overview Level

Use Case Name: Register User Account	**ID: 1**	**Importance Level:** High
Primary Actor: Visitor, Future User/Client	**Use Case Type:** Overview, Essential	
Stakeholders and Interests: Client wants to register an account to be able to use Motley Zoo services System does a captcha check to confirm a human user is attempting to register (as opposed to a script) System sends out registration confirmation email to finalize registration		
Brief Description: This use case describes how a visitor registers an account with Motley Zoo Service Gateway to become a user/client		
Trigger: Visitor clicks on "Register Account" link in main login screen. **Type:** External		
Relationships: **Association:** Client, Emailing middleware, Captcha middleware **Include:** **Extend:** **Generalization:**		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: Reset User Password	**ID:** 2	**Importance Level:** High
Primary Actor: User/Client	**Use Case Type:** Overview, Essential	
Stakeholders and Interests:	Existing client wants to recover/reset their accounts password System does a captcha check to confirm a human user is attempting to request recovery of password System generates and sends out password recovery link System sends out a password changed notice to registered email account	
Brief Description:	This use case describes how a user resets their password in case it's forgotten	
Trigger:	Visitor clicks on "Forgot password" link in main login screen.	
Type:	External	
Relationships:		
Association:	Client, Emailing middleware, Captcha middleware	
Include:		
Extend:		
Generalization:		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: Login User Account	**ID:** 3	**Importance Level:** High
Primary Actor: User/Client	**Use Case Type:** Overview, Essential	
Stakeholders and Interests: Existing client wants to log into the system System checks for email existence and password matching System creates JWT and logs user into system		
Brief Description: This use case describes how a user logs into their account		
Trigger: Visitor inputs their email and password and clicks "Login" button. **Type:** External		
Relationships: **Association:** Client, Authentication API, Authorization API **Include:** **Extend:** **Generalization:**		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: User Account View	**ID:** 4	**Importance Level:** High
Primary Actor: User/Client	**Use Case Type:** Overview, Essential	
Stakeholders and Interests: Existing client wants to view their account information System checks Authorization API/JWT for role type		
Brief Description: This use case describes how a user views their account information		
Trigger: Visitor goes to the "Account Information" section. **Type:** External		
Relationships: **Association:** Client, Authorization API **Include:** **Extend:** **Generalization:**		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: User Account Edit	**ID:** 5	**Importance Level:** High
Primary Actor: User/Client, Administrator	**Use Case Type:** Overview, Essential	
Stakeholders and Interests: Existing client wants to edit their account Administrator wants to edit the users account System checks Authorization API/JWT for role type		
Brief Description: This use case describes how a user or administrator edits a user account		
Trigger: Visitor goes to the "Account Information" section and clicks edit button which loads view data into editable form. Administrator can access account profile from a list, and follow same procedure described above, or they can click "Edit" button in the admin panels user list which loads a form to edit fields of said account. **Type:** External		
Relationships: **Association:** Client, Administrator, Authorization API **Include:** **Extend:** **Generalization:**		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: User Account Delete	**ID: 6**	**Importance Level:** High
Primary Actor: User/Client, Administrator	**Use Case Type:** Overview, Essential	
Stakeholders and Interests: Existing client wants to delete their account Administrator wants to delete the users account System checks Authorization API/JWT for role type		
Brief Description: This use case describes how a user or administrator deletes a user account		
Trigger: Visitor goes to the "Account Information" section and clicks edit button, in subsequent form clicks "Delete account" and follows prompt to confirm account deletion Administrator can access account profile from a list, and follow same procedure described above, or they can click "Delete" button in the admin panels user list and follow prompt to delete account. **Type:** External		
Relationships: **Association:** Client, Administrator, Authorization API **Include:** **Extend:** **Generalization:**		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: Schedule Single Time Slot	**ID: 7**	**Importance Level:** High
Primary Actor: User/Client, Administrator	**Use Case Type:** Overview, Essential	
Stakeholders and Interests: Existing client wants to schedule an appointment Administrator wants to schedule appointment for client System checks Authorization API/JWT for role type		
Brief Description: This use case describes how a user or administrator schedules an appointment for dog walk		
Trigger: Visitor goes to the scheduler section, chooses a free time slot length or group adventure option, chooses free time slot, proceeds to payment for appointed time slot. Administrator can access account profile from a list, and follow same procedure described above. In this case however, the system sends out a temporary generated link to user for payment, which is the actual trigger in this flow. **Type:** External		
Relationships: **Association:** Client, Administrator, Authorization API, Payment Gate API, Email middleware **Include:** **Extend:** **Generalization:**		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: Schedule Daily Time Slot	**ID:** 8	**Importance Level:** High
Primary Actor: User/Client, Administrator	**Use Case Type:** Overview, Essential	
Stakeholders and Interests: Existing client wants to set up a daily time slot dog walking subscription Administrator wants to schedule a daily time slot dog walking subscription System checks Authorization API/JWT for role type		
Brief Description: This use case describes how a user or administrator schedules a daily appointment for dog walk		
Trigger: Visitor goes to the scheduler section, chooses a free time slot length or group adventure option, chooses free time slots for each day for the week, proceeds to payment for appointed time slots. Administrator can access account profile from a list, and follow same procedure described above. In this case however, the system sends out a temporary generated link to user for payment, which is the actual trigger in this flow. **Type:** External		
Relationships: **Association:** Client, Administrator, Authorization API, Payment Gate API, Email middleware **Include:** **Extend:** **Generalization:**		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: Cancel Scheduled Appointments	**ID: 9**	**Importance Level:** High
Primary Actor: User/Client, Administrator	**Use Case Type:** Overview, Essential	
Stakeholders and Interests: Existing client wants to cancel their scheduled appointment or appointments in case of daily. Administrator wants to cancel users scheduled appointments. System checks Authorization API/JWT for role type		
Brief Description: This use case describes how a user or administrator cancels a single or daily range of appointments.		
Trigger: Visitor goes to the scheduler section, clicks on scheduled appointment to highlight it, clicks "Cancel Appointment", confirms prompt. Administrator can access account profile from a list, and follow same procedure described above. In this case however, the system sends out a temporary generated link to users email for cancellation confirmation, which is the trigger. **Type:** External		
Relationships: **Association:** Client, Administrator, Authorization API, Payment Gate API, Email middleware **Include:** **Extend:** **Generalization:**		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: User Report on Dog Walk	**ID: 10**	**Importance Level:** High
Primary Actor: User/Client	**Use Case Type:** Overview, Essential	
Stakeholders and Interests:	Existing client wants to put in a short and concise optional experience report of dog walk System checks Authorization API/JWT for role type	
Brief Description:	This use case describes how a user would give feedback on a walker in terms of their pickup and drop off procedure, communications if needed, etc.	
Trigger:	Visitor goes to their main profile, chooses a completed job, answers set questions, and optionally adds in a short-written comment/concern if they wish	
Type:	External	
Relationships:		
Association:	Client, Authorization API	
Include:		
Extend:		
Generalization:		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: Setting Desired Dog Walker	**ID: 11**	**Importance Level:** High
Primary Actor: User/Client	**Use Case Type:** Overview, Essential	
Stakeholders and Interests: Client wants to request a specific dog walker for a job System checks Authorization API/JWT for role type		
Brief Description: This use case describes how a user can request a specific dog walker, with no guarantee of fulfillment in case of scheduling conflict, or other factors		
Trigger: Visitor goes to their main profile and chooses a desired dog walker from a generated list of walkers that did jobs for them already. **Type:** External		
Relationships: **Association:** Client, Authorization API **Include:** **Extend:** **Generalization:**		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: User Viewing Reports from Dog Walker	**ID: 12**	**Importance Level:** High
Primary Actor: User/Client	**Use Case Type:** Overview, Essential	
Stakeholders and Interests: Client wants to view reports left by dog walker on specific job. System checks Authorization API/JWT for role type		
Brief Description: This use case describes how a user, or the administrator view reports, and if applicable, pictures of dog walks made by the dog walker		
Trigger: Visitor goes to their main profile page, and clicks on a completed dog walk, which loads the report of the chosen completed walk Administrator goes to user list, clicks the chosen user and then continues the same procedure as described above. Administrator also has ability to view reports in their reports section by filtering report type and either user or staff member. **Type:** External		
Relationships: **Association:** Client, Staff, Administrator, Authorization API **Include:** **Extend:** **Generalization:**		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: User Write Business Review	**ID: 13**	**Importance Level:** Med
Primary Actor: User/Client	**Use Case Type:** Overview, Essential	
Stakeholders and Interests: Client wants to write general review of business once enabled System enables review ability upon a set amount of completed jobs System checks Authorization API/JWT for role type		
Brief Description: This use case describes how a user can write an overall business review once enabled to do so		
Trigger: Visitor wishing to do review if enabled to do so **Type:** External		
Relationships: **Association:** Client, Staff, Administrator, Authorization API **Include:** **Extend:** **Generalization:**		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: Admin Quick Register User Accounts	**ID: 14**	**Importance Level:** High
Primary Actor: Administrator	**Use Case Type:** Overview, Essential	
Stakeholders and Interests: Administrator registers account for users who may not be tech-savvy when requested System checks for existence of email/profile System checks Authorization API/JWT for role type		
Brief Description: This use case describes how the administrator can create profiles in the system, bypassing regular registration processes		
Trigger: Administrator is contacted by potential client who may need assistance setting up account **Type:** External		
Relationships: **Association:** Administrator, User **Include:** Auth API **Extend:** **Generalization:**		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: Admin Register Staff Accounts	**ID: 15**	**Importance Level:** High
Primary Actor: Administrator	**Use Case Type:** Overview, Essential	
Stakeholders and Interests: Administrator registers account for staff System checks for existence of email/profile System checks Authorization API/JWT for role type		
Brief Description: This use case describes how the administrator can create staff accounts		
Trigger: Administrator sets up a new staff members online account **Type:** External		
Relationships: **Association:** Administrator, Staff Members **Include:** **Extend:** **Generalization:**		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: Set Permissions	**ID: 16**	**Importance Level:** High
Primary Actor: Administrator	**Use Case Type:** Overview, Essential	
Stakeholders and Interests: Administrator sets permissions of staff members' accounts System checks Authorization API/JWT for role type		
Brief Description: This use case describes how the administrator can set either standard or special permissions on staff members' accounts		
Trigger: Administrator changes/adds permissions for staff member from default permissions **Type:** External		
Relationships: **Association:** Administrator, Staff Members **Include:** **Extend:** **Generalization:**		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: Admin Modify Accounts	**ID: 17**	**Importance Level:** High
Primary Actor: Administrator	**Use Case Type:** Overview, Essential	
Stakeholders and Interests: Administrator wishes to modify user or staff accounts System checks Authorization API/JWT for role type		
Brief Description: This use case describes how the administrator can modify user or staff accounts		
Trigger: Administrator makes changes to either user or staff accounts **Type:** External		
Relationships: **Association:** Administrator, Staff Members, Users/Clients **Include:** **Extend:** **Generalization:**		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: Admin View/Export Reports	**ID:** 18	**Importance Level:** High
Primary Actor: Administrator	**Use Case Type:** Overview, Essential	
Stakeholders and Interests: Administrator wishes to view, filter and/or export reports to pdf System checks Authorization API/JWT for role type		
Brief Description: This use case describes how the administrator can view all reports, filter them by type, user, or staff member, and export them to pdf format		
Trigger: Administrator enters the reports section of admin panel **Type:** External		
Relationships: **Association:** Administrator, Staff Members, Users/Clients **Include:** **Extend:** **Generalization:**		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: Admin Schedule Request Approval	**ID: 19**	**Importance Level:** High
Primary Actor: Administrator	**Use Case Type:** Overview, Essential	
Stakeholders and Interests: Administrator wishes to have the ability to approve or decline staff scheduling requests System checks Authorization API/JWT for role type		
Brief Description: This use case describes how the administrator can approve or decline the work scheduling requests from staff members		
Trigger: Administrator enters scheduling section and proceeds to staff scheduling sub-section **Type:** External		
Relationships: **Association:** Administrator, Staff Members **Include:** **Extend:** **Generalization:**		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: Admin Override Staff Scheduling	**ID:** 20	**Importance Level:** High
Primary Actor: Administrator	**Use Case Type:** Overview, Essential	
Stakeholders and Interests:	Administrator wishes to have the ability to override set schedules of staff who have permissions to make their own schedules System checks Authorization API/JWT for role type	
Brief Description:	This use case describes how the administrator can override staff schedules in cases such as emergencies, short staffing, etc.	
Trigger:	Administrator enters scheduling section and proceeds to staff scheduling sub-section, chooses a staff members schedule and clicks override button to set new schedule	
Type:	External	
Relationships:		
Association:	Administrator, Staff Members	
Include:		
Extend:		
Generalization:		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: Admin Set Up API Webhooks	ID: 21	**Importance Level:** Low
Primary Actor: Administrator	**Use Case Type:** Overview, Essential	
Stakeholders and Interests:	Administrator wishes to have the ability to toggle on and off APIs for future use on main site System checks Authorization API/JWT for role type	
Brief Description:	This use case describes how the administrator can toggle APIs for promotions and reviews to be used on main site	
Trigger:	Administrator enters settings section, and enables relevant APIs	
Type:	External	
Relationships:		
Association:	Administrator	
Include:		
Extend:		
Generalization:		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: Staff View Booked Schedule	**ID:** 22	**Importance Level:** High
Primary Actor: Staff Member	**Use Case Type:** Overview, Essential	
Stakeholders and Interests:	Staff member wishes to see the dog walks they are booked for. System checks Authorization API/JWT for role type	
Brief Description:	This use case describes the staff members' ability to see their booked schedule and make plans accordingly	
Trigger: Staff member enters the scheduling section of their staff user panel. **Type:**		
Relationships: **Association:** Staff Members **Include:** **Extend:** **Generalization:**		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: Staff Set Schedule of Availability	**ID: 23**	**Importance Level:** High
Primary Actor: Staff Member	**Use Case Type:** Overview, Essential	
Stakeholders and Interests: Staff member, if given permission wishes to set their own work schedule System checks Authorization API/JWT for role type		
Brief Description: This use case describes some of the staff members' ability to set their own schedules provided they have the permission to do so.		
Trigger: Staff member enters the scheduling section of their staff user panel, chooses their desired work hours for any given week. **Type:** External		
Relationships: **Association:** Staff Members **Include:** **Extend:** **Generalization:**		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: Staff Set Dog Handling Limit	**ID:** 24	**Importance Level:** High
Primary Actor: Staff Member	**Use Case Type:** Overview, Essential	
Stakeholders and Interests:	Staff member wishes to set their maximum dog/dog weight limit that they can handle at a time System checks Authorization API/JWT for role type	
Brief Description:	This use case describes the dog walker's ability to set their maximum handling ability in terms of both weight classes of dogs, and number of dogs they can take on at once	
Trigger:	Staff member enters their profile and sets both the dog amount and weight classes they can handle for each walk	
Type:	External	
Relationships:		
Association:	Staff Members	
Include:		
Extend:		
Generalization:		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: Staff See Hours Worked and Paid Out	**ID: 25**	**Importance Level:** High
Primary Actor: Staff Member	**Use Case Type:** Overview, Essential	
Stakeholders and Interests:	Staff member wishes to set the number of hours they've worked, and how much of it was paid out to them in a given time span System checks Authorization API/JWT for role type	
Brief Description:	This use case describes the staff members ability to track their hours worked and how much of it was paid	
Trigger:	Staff member enters scheduling section, chooses time span, and sees total of hours worked and total of hours paid out	
Type:	External	
Relationships:		
Association:	Staff Members	
Include:		
Extend:		
Generalization:		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: Staff Reports to Admin	**ID:26**	**Importance Level:** High
Primary Actor: Staff Member	**Use Case Type:** Overview, Essential	
Stakeholders and Interests: Staff member wishes to write report on dog walk or client to the owner System checks Authorization API/JWT for role type		
Brief Description: This use case describes the staff members optional ability to write a report to the owner regarding a dog walk or interaction with a client		
Trigger: Staff member enters main page of app, chooses a job from a list of completed jobs, and writes a private report to the owner. **Type:** External		
Relationships: **Association:** Staff Members **Include:** **Extend:** **Generalization:**		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: Staff Reports to Clients	**ID:**27	**Importance Level:** High
Primary Actor: Staff Member	**Use Case Type:** Overview, Essential	
Stakeholders and Interests: Staff member writes a report of a dog walk to client System checks Authorization API/JWT for role type		
Brief Description: This use case describes the staff members ability to write a report to the client regarding a dog walk		
Trigger: Staff member enters main page of app, chooses a job from a list of completed jobs, and writes a report to the owner with ability to upload a picture. **Type:** External		
Relationships: **Association:** Staff Members, Clients **Include:** **Extend:** **Generalization:**		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

Use Case Name: Quick Chat	**ID:28**	**Importance Level:** High
Primary Actor: Staff Member, User, Admin	**Use Case Type:** Overview, Essential	
Stakeholders and Interests: Staff member messages a client or the owner/administrator System checks Authorization API/JWT for role type		
Brief Description: This use case describes the staff members ability to contact either the client or administrator in a chat format		
Trigger: Staff member enters communications section of UI, and starts a chat with person they wish to speak with **Type:** External		
Relationships: **Association:** Staff Members, Clients, Administrator **Include:** **Extend:** **Generalization:**		
Normal Flow of Events:		
SubFlows:		
Alternate/Exceptional Flow:		

3.1.2 Use Case Descriptions – Overview Level

Use Case Name: Register User Account	**ID: 1-1**	**Importance Level:** High
Primary Actor: Visitor, Future User/Client	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Client wants to register an account to be able to use Motley Zoo services System does a captcha check to confirm a human user is attempting to register (as opposed to a script) System sends out registration confirmation email to finalize registration		
Brief Description: This use case describes how a visitor registers an account with Motley Zoo Service Gateway to become a user/client		
Trigger: Visitor clicks on "Register Account" link in main login screen. **Type:** External		
Relationships: **Association:** Client, Emailing middleware, Captcha middleware **Include:** Form, Emailing, Registration Key Generation **Extend:** Captcha **Generalization:**		
Normal Flow of Events: 1. Visitor clicks on portal link on Motley Zoo's main site 2. Visitor clicks on "Register Account" link in login screen 3. Visitor fills out registration form consisting of name, email address, and password fields 4. Visitor completes captcha challenge 5. Visitor clicks "Register" button 6. Visitor finalizes registration through sent confirmation email link		
SubFlows: S-1: System checks if email exists in database 1. If email does not exist, load generic message stating to check email 2. System does form validations on client side, checks password and confirms password fields match 3. System encrypts password for database storage		
Alternate/Exceptional Flow: S-1: 1. If email does exist, system loads same generic message, does nothing further		

Use Case Name: Reset User Password	**ID:** 2-1	**Importance Level:** High
Primary Actor: User/Client	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Existing client wants to recover/reset their accounts password System does a captcha check to confirm a human user is attempting to request recovery of password System generates and sends out password recovery link System sends out a password changed notice to registered email account		
Brief Description: This use case describes how a user resets their password in case it's forgotten		
Trigger: Visitor clicks on "Forgot password" link in main login screen. **Type:** External		
Relationships: **Association:** Client, Emailing middleware, Captcha middleware **Include:** Emailing, DB Check **Extend:** Form Generation **Generalization:**		
Normal Flow of Events: 1. User clicks on "Forgot Password" link 2. User inputs their email address 3. User clicks "Reset Password" button 4. User checks email and resets password through the sent link		
SubFlows: S-1: System checks if email exists in database: 1. If email exists, system executes password reset procedure and loads generic password recovery message 2. System sets a 15-minute expiry of password procedure 3. System sends out generated recovery form link 4. System encrypts new password and updates database entry		
Alternate/Exceptional Flow: S-1: 1. If email does not exist, system loads same password recovery message, does nothing further		

Use Case Name: Login User Account	**ID:** 3-1	**Importance Level:** High
Primary Actor: User/Client	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Existing client wants to log into the system System checks for email existence and password matching System creates JWT and logs user into system		
Brief Description: This use case describes how a user logs into their account		
Trigger: User inputs their email and password and clicks "Login" button. **Type:** External		
Relationships: **Association:** Client, Authentication API, Authorization API **Include:** Auth API, DB **Extend:** Auth API **Generalization:**		
Normal Flow of Events: 1. User inputs their email address and password 2. User clicks "Login" button		
SubFlows: S-1: System runs Authentication and Authorization APIs 1. If account exists, system authenticates user, and sets JWT with appropriate Authorization/Role field 2. System loads user's main site/panel		
Alternate/Exceptional Flow: S-1: 1. If Account does not exist, system loads message stating either the account does not exist or the password is incorrect.		

Use Case Name: User Account View	**ID:** 4-1	**Importance Level:** High
Primary Actor: User/Client	**Use Case Type:** Overview, Essential	
Stakeholders and Interests: Existing client wants to view their account information System checks Authorization API/JWT for role type		
Brief Description: This use case describes how a user views their account information		
Trigger: Visitor goes to the "Account Information" section. **Type:** External		
Relationships: **Association:** Client, Authorization API **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. User has successfully logged in 2. Authorization API has loaded and matched hashed Web Token (JWT), and checked UserType and permissions		
SubFlows: S-1: Authorization API Init 1. System sets timer on JWT, and initiates refresher token process to be triggered with activity		
Alternate/Exceptional Flow: S-1: 2. If no activity by the time JWT times out, system logs user out.		

Use Case Name: User Account Edit	**ID:** 5-1	**Importance Level:** High
Primary Actor: User/Client	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Existing client wants to edit their account System checks Authorization API/JWT for role type		
Brief Description: This use case describes how a user or administrator edits a user account		
Trigger: Visitor goes to the "Account Information" section and clicks edit button which loads profile data into editable form. **Type:** External		
Relationships: **Association:** Client, Authorization API **Include:** Auth API, DB **Extend:** **Generalization:**		
Normal Flow of Events: 1. User clicks on profile menu choice 2. User clicks "Edit Profile" option within loaded profile 3. User enters information 4. User clicks "Save" or "Cancel" button.		
SubFlows: S-1: Database Profile Information Update 1. System loads information from view only to pre-filled form 2. System does field validation, both client and server-side upon user clicking "Save" 3. System updates information into database		
Alternate/Exceptional Flow: S-1: 1. Upon loading of form, if the user clicks "Cancel" at any point, system loads back to view-only page		

Use Case Name: User Account Delete	**ID: 6-1**	**Importance Level:** High
Primary Actor: User/Client	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Existing client wants to delete their account System checks Authorization API/JWT for role type		
Brief Description: This use case describes how a user or administrator deletes a user account		
Trigger: User goes to the "Account Information" section and clicks edit button, in subsequent form clicks "Delete account" and follows prompt to confirm account deletion **Type:** External		
Relationships: **Association:** Client, Administrator, Authorization API **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. User clicks on profile menu choice 2. User clicks "Delete Profile" 3. User confirms deletion prompt		
SubFlows: S-1: Data Profile Deletion 1. System checks deletion prompt response 2. If confirmed, system sets account to inactive, deletes most profile information		
Alternate/Exceptional Flow: S-1: 1. If user cancels action upon deletion prompt, system loads back to profile site		

Use Case Name: Schedule Single Time Slot	**ID:** 7-1	**Importance Level:** High
Primary Actor: User/Client	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Existing client wants to schedule an appointment System checks Authorization API/JWT for role type		
Brief Description: This use case describes how a user or administrator schedules an appointment for dog walk		
Trigger: User goes to the scheduler section, chooses a free time slot length or group adventure option, chooses free time slot, proceeds to payment for appointed time slot. **Type:** External		
Relationships: **Association:** Client, Administrator, Authorization API, Payment Gate API, Email middleware **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. User clicks on Schedule Walk menu choice 2. User chooses a walk type and length 3. User chooses an empty time slot for service 4. User confirms time slot and proceeds to checkout 5. User pays through the payment gate		
SubFlows: S-1: Schedule loads 1. Upon user choosing walk type and length, system dynamically loads appropriate scheduler 2. System checks staff availability S-2: Schedule fills out availability 1. Upon checking staff availability, system fills out the scheduler appropriately, blocking unavailable time slots 2. System containerizes available time slots and fills them into scheduler as available slots for purchase S-3: Update System Upon Purchase 1. Upon purchase system updates staff member's schedule 2. System also generates invoice db entry for Users and Administrators record keeping and PDF conversion 3. System loads purchase confirmation message, and sends purchase confirmation email to User		
Alternate/Exceptional Flow: S-1,2,3 – Any errors or payment decline results in appropriate message output to User		

Use Case Name: Schedule Daily Time Slot	**ID:** 8-1	**Importance Level:** High
Primary Actor: User/Client	**Use Case Type:** Detail, Essential	
Stakeholders and Interests:	Existing client wants to set up a daily time slot dog walking subscription System checks Authorization API/JWT for role type	
Brief Description:	This use case describes how a user schedules a daily appointment for dog walk	
Trigger:	User goes to the scheduler section, chooses a free time slot length or group adventure option, chooses free time slots for each day for the week, proceeds to payment for appointed time slots.	
Type:	External	

Relationships:

Association: Client, Authorization API, Payment Gate API, Email middleware

Include:

Extend: 7-1

Generalization:

Normal Flow of Events:

1. User clicks on Schedule Walk menu choice
2. User chooses a walk type as Daily walk and walk length, and subscription amount of days
3. User chooses an empty time slot for each day in amount of days
4. User confirms choices and proceeds to checkout
5. User pays through payment gate

SubFlows:

S-1: Schedule loads
1. Upon user choosing walk length and subscription length, system dynamically loads appropriate scheduler
2. System checks staff availability for each day

S-2: Schedule fills out availability
1. Upon checking staff availability, system fills out the scheduler appropriately, blocking unavailable time slots
2. System containerizes available time slots and fills them into scheduler as available slots for purchase

S-3: Update System Upon Purchase
1. Upon purchase system updates staff member's schedule
2. System also generates invoice db entry for Users and Administrators record keeping and PDF conversion
3. System loads purchase confirmation message, and sends purchase confirmation email to User

Alternate/Exceptional Flow:

S-1,2,3 – Any errors or payment decline results in appropriate message output to User

Use Case Name: Cancel Scheduled Appointments	**ID:** 9-1	**Importance Level:** High
Primary Actor: User/Client	**Use Case Type:** Detail, Essential	
Stakeholders and Interests:	Existing client wants to cancel their scheduled appointment or appointments in case of daily. System checks Authorization API/JWT for role type	
Brief Description:	This use case describes how a user or administrator cancels a single or daily range of appointments.	
Trigger:	Visitor goes to the scheduler section, clicks on scheduled appointment to highlight it, clicks "Cancel Appointment", confirms prompt.	
Type:	External	

Relationships:

- **Association:** Client, Authorization API, Payment Gate API, Email middleware
- **Include:** Payment API
- **Extend:** 7-1, 8-1
- **Generalization:**

Normal Flow of Events:

1. User clicks on Schedule Walk menu choice
2. Schedule Walk view has "currently scheduled walks" sub-section, user clicks on the walk they wish to cancel.
3. User clicks on "Cancel" button of chosen walk/service
4. User confirms they wish to cancel upon reading prompt with the date and time of walk, and possible penalty payment depending on time to cancellation.

SubFlows:

S-1: Refund Process
1. Upon confirmation of cancellation, system initiates refund process, if with penalty, system calculates out the penalty percentage and initiates partial refund.

S-2: Schedule Update
1. Upon refund success, schedule of dog walker is updated, and the canceled time slot is added back to availability for scheduling
2. User and dog walker are both sent appropriate emails notifying them of the cancellation

Alternate/Exceptional Flow:

S-2: If refund should fail, user is notified and may attempt flow of events steps 3 and 4 again

Use Case Name: User Report On Dog Walk	**ID: 10-1**	**Importance Level:** High
Primary Actor: User/Client	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Existing client wants to put in a short and concise optional experience report of dog walk System checks Authorization API/JWT for role type		
Brief Description: This use case describes how a user would give feedback on a walker in terms of their pickup and drop off procedure, communications if needed, etc.		
Trigger: User goes to their main profile, chooses a completed job, answers set questions, and optionally adds in a short-written comment/concern if they wish **Type:** External		
Relationships: **Association:** Client, Authorization API **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. User goes to scheduled walks sections 2. User clicks on a completed job 3. User answers set questions and optionally adds a comment 4. User clicks “Submit” button on report form		
SubFlows: S-1: System updates finished job’s db entry 1. System updates the jobs db entry with the answers to questions and comment 2. System loads message thanking the User for their feedback and letting them know they may be contacted by Administrator if the feedback contains a concern S-2: System notifies Administrator 1. System notifies Administrator that a new User report has been added. 2. If set questions are negatively answered, notification is marked as urgent		
Alternate/Exceptional Flow: S-1: 1. If update fails User is notified of the error and asked to try again.		

Use Case Name: Setting Desired Dog Walker	**ID: 11-1**	**Importance Level:** High
Primary Actor: User/Client	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Client wants to request a specific dog walker for a job System checks Authorization API/JWT for role type		
Brief Description: This use case describes how a user can request a specific dog walker, with no guarantee of fulfillment in case of scheduling conflict, or other factors		
Trigger: Visitor goes to their main profile and chooses a desired dog walker from a generated list of walkers that did jobs for them already. **Type:** External		
Relationships: **Association:** Client, Authorization API **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. User clicks into their profile preferences 2. If user had previous walks ordered, a dropdown menu with list of previous walkers is generated, user chooses their preferred walker from list 3. User clicks "Save" form button 4. User receives prompt letting them know that their preference is not guaranteed 5. User agrees to the condition		
SubFlows: S-1: User's preference is updated into their profile db entry 1. System updates the user's profile into in the database to reflect users' preference		
Alternate/Exceptional Flow: S-1: 1. If user did not agree to prompt, process is canceled		

Use Case Name: User Viewing Reports from Dog Walker	**ID: 12-1**	**Importance Level:** High
Primary Actor: User/Client	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Client wants to view reports left by dog walker on specific job. System checks Authorization API/JWT for role type		
Brief Description: This use case describes how a user, or the administrator view reports, and if applicable, pictures of dog walks made by the dog walker		
Trigger: User goes to their main profile page, and clicks on a completed dog walk, which loads the staff members report of the chosen completed walk **Type:** External		
Relationships: **Association:** Client, Staff, Administrator, Authorization API **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. User navigates to their scheduled walks section 2. User clicks on a completed job 3. If a report was submitted user clicks on it for a dynamically loaded detailed view of it		
SubFlows: S-1: System fetches full report from database 1. When user loads completed job, the system fetches the report entry from the database 2. Part of the report entry is links of any pictures stored in a file system, so in this step the picture(s) will be fetched and loaded as well into the view.		
Alternate/Exceptional Flow: S-1: 1. If there is no database entry, system will not load anything 2. If there is a report on the database, but there is no link of picture(s) uploaded, the system will skip step 2 of S-1.		

Use Case Name: User Write Business Review	**ID:** 13-1	**Importance Level:** Med
Primary Actor: User/Client	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Client wants to write general review of business once enabled System enables review ability upon a set amount of completed jobs System checks Authorization API/JWT for role type		
Brief Description: This use case describes how a user can write an overall business review once enabled to do so		
Trigger: User wishing to do review if enabled to do so **Type:** External		
Relationships: **Association:** Client, Staff, Administrator, Authorization API **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. If feature is enabled for user, user clicks on "Review Motley Zoo" button on the main page 2. User gives a rating out of 5 stars, in half-star increments 3. User fills out textbox with their review 4. User clicks Submit button		
SubFlows: S-1: Validation 1. When user click submit button, the form fields are validated client-side and server-side 2. If validation passes, the review is added to reviews table of database		
Alternate/Exceptional Flow: S-1: 1. If validation fails, system outputs a message notifying the user of validation failure 2. If addition of the review in database fails, user is notified of error, and prompted to try again		

Use Case Name: Quick Register User Accounts	ID: 14-1	**Importance Level:** High
Primary Actor: Administrator	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Administrator registers account for users who may not be tech-savvy when requested System checks for existence of email/profile System checks Authorization API/JWT for role type		
Brief Description: This use case describes how the administrator can create profiles in the system, bypassing regular registration processes		
Trigger: Administrator is contacted by potential client who may need assistance setting up account **Type:** External		
Relationships: **Association:** Administrator, Users, Emailing API **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. Administrator navigates to the users list on their system 2. Admin clicks "Add User" button 3. Admin inputs given user information into registration form 4. Admin clicks submit button of the form		
SubFlows: S-1: Send email confirming registration 1. Upon user being added, system sends email to user confirming they are registered, and leaving a set password link for the user to set their password 2. System generates the set password view for user to input and save their password.		
Alternate/Exceptional Flow: S-1: 1. If no password is set within 24 hours, administrator is notified, and system re-sends email		

Use Case Name: Admin Register Staff Accounts	ID: 15-1	**Importance Level:** High
Primary Actor: Administrator	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Administrator registers account for staff System checks for existence of email/profile System checks Authorization API/JWT for role type		
Brief Description: This use case describes how the administrator can create staff accounts		
Trigger: Administrator sets up a new staff members online account **Type:** External		
Relationships: **Association:** Administrator, Staff Members, Emailing API **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. Admin navigates to their staff panel 2. Admin clicks "Add Staff" button 3. Admin inputs information 4. Admin clicks Submit button of the form		
SubFlows: S-1: Send email confirming registration 1. Upon staff member being added, system sends email to member confirming they are registered, and adds a set password link for the user to set their password, as well as a copy of their work contract and NDA if applicable 2. System generates the set password view for staff member to input and save their password.		
Alternate/Exceptional Flow: S-1: 1. If no password is set within 12 hours, administrator is notified, and system re-sends email		

Use Case Name: Admin Set Permissions	ID: 16-1	**Importance Level:** High
Primary Actor: Administrator	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Administrator sets permissions of staff members' accounts System checks Authorization API/JWT for role type		
Brief Description: This use case describes how the administrator can set either standard or special permissions on staff members' accounts		
Trigger: Administrator changes/adds permissions for staff member from default permissions **Type:** External		
Relationships: **Association:** Administrator, Staff Members, Authorization API **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. Admin navigates to the staff members section of their panel 2. Admin clicks on a staff members profile from list 3. Admin clicks "set permissions" button 4. Admin sets permissions 5. Admin clicks submit button		
SubFlows: S-1: Authorization API updates permissions 1. Upon admin submitting permissions change, the system updates the staff members permissions in the staff members database table 2. If the user is logged in at the time, they have to either log out and log in again, or wait until the next refresher token for permissions to reflect on their JWT token.		
Alternate/Exceptional Flow: **S-1:** 1. If permissions update fails, admin is notified with error message		

Use Case Name: Admin Modify Accounts	ID: 17-1	**Importance Level:** High
Primary Actor: Administrator	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Administrator wishes to modify user or staff accounts System checks Authorization API/JWT for role type		
Brief Description: This use case describes how the administrator can modify user or staff accounts		
Trigger: Administrator makes changes to either user or staff accounts **Type:** External		
Relationships: **Association:** Administrator, Staff Members, Users/Clients **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. Admin navigates either to the user's section or staff member's section 2. Admin clicks on either user or staff member respectively 3. Admin clicks "edit profile" button 4. Admin makes edit(s) in form 5. Admin clicks submit button.		
SubFlows: S-1: Profiles update 1. When admin makes the changes and submits, the respective database entry is updated 2. Upon successful update, admin's view changes from form to view-only fields		
Alternate/Exceptional Flow: S-1: 1. If profile does not update, the view remains in edit mode, and an error message is output below the form.		

Use Case Name: Admin View/Export Reports	ID: 18-1	**Importance Level:** High
Primary Actor: Administrator	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Administrator wishes to view, filter and/or export reports to pdf System checks Authorization API/JWT for role type		
Brief Description: This use case describes how the administrator can view all reports, filter them by type, user, or staff member, and export them to pdf format		
Trigger: Administrator enters the reports section of admin panel **Type:** External		
Relationships: **Association:** Administrator, Staff Members, Users/Clients, PDF Middleware **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. Admin enters the reports section of admin panel 2. Admin filters type of report they need, user or staff member report is related to 3. Admin clicks on report title from filtered list 4. Admin clicks “Export to PDF” button		
SubFlows: S-1: Fetch reports full list 1. When admin enters reports section the system fetches titles of all reports, report types and user/staff member names of each report 2. System places all data into a datatable with pagination if report number exceeds 25 reports S-2: Filter reports 1. As admin filters report type and user, the reports in the datatable decrease to only the filtered reports S-3: Convert report to PDF format 3. As admin clicks export to PDF button, the system generates the report based on appropriate template, and opens the save feature of admins computer file system		
Alternate/Exceptional Flow: S-1: 1. If data fetch fails, system notifies admin with error messages S-3: 1. If PDF conversion fails, system notifies admin with error message, does not open file system save		

Use Case Name: Admin Schedule Request Approval	**ID:** 19-1	**Importance Level:** High
Primary Actor: Administrator	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Administrator wishes to have the ability to approve or decline staff scheduling requests System checks Authorization API/JWT for role type		
Brief Description: This use case describes how the administrator can approve or decline the work scheduling requests from staff members		
Trigger: Administrator enters scheduling section and proceeds to staff scheduling sub-section **Type:** External		
Relationships: **Association:** Administrator, Staff Members, Emailing API **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. Admin navigates to staff section of their panel 2. If scheduling request exists, scheduling requests sub-section is populated 3. Admin clicks on scheduling request for review 4. Admin approves or denies request 5. Admin clicks submit button		
SubFlows: S-1: Fetch requests 1. When Admin loads staff section, system checks staff members database for scheduling requests 2. If requests exist without status (approved or denied), system fetches them and places them into the scheduling requests sub-section S-2: Approve Requests 1. If request is approved, staff member's schedule is updated to reflect changes in availability		
Alternate/Exceptional Flow: S-2: 1. If request is denied, admins view adds a text box where they can optionally add reasoning for request denial 2. Once admin clicks submit button, staff member is emailed with request response		

Use Case Name: Admin Override Staff Scheduling	**ID:** 20-1	**Importance Level:** High
Primary Actor: Administrator	**Use Case Type:** Detail, Essential	
Stakeholders and Interests:	Administrator wishes to have the ability to override set schedules of staff who have permissions to make their own schedules System checks Authorization API/JWT for role type	
Brief Description:	This use case describes how the administrator can override staff schedules in cases such as emergencies, short staffing, etc.	
Trigger:	Administrator enters scheduling section and proceeds to staff scheduling sub-section, chooses a staff members schedule and clicks override button to set new schedule	
Type:	External	

Relationships:

Association: Administrator, Staff Members, Emailing API

Include:

Extend:

Generalization:

Normal Flow of Events:

1. Admin enters staff scheduling section
2. Admin filters staff member
3. Admin clicks on staff members chosen schedule
4. Admin clicks “Override Schedule”
5. Admin makes necessary changes
6. Admin optionally fills out text area box explaining why the members schedule was overridden
7. Admin clicks submit button

SubFlows:

S-1: Update schedule in db

1. As the admin submits override, the staff members schedule is updated in db
2. Staff member is emailed notifying them of the override, along with the message if admin added one

Alternate/Exceptional Flow:

S-1: Update schedule fails

1. If the override update fails, a message notifies admin that the change did not go through

Use Case Name: Admin Set Up API Webhooks	ID: 21-1	**Importance Level:** Low
Primary Actor: Administrator	**Use Case Type:** Detail, Essential	
Stakeholders and Interests:	Administrator wishes to have the ability to toggle on and off APIs for future use on main site System checks Authorization API/JWT for role type	
Brief Description:	This use case describes how the administrator can toggle APIs for promotions and reviews to be used on main site	
Trigger:	Administrator enters settings section, and enables relevant APIs	
Type:	External	

Relationships:

Association: Administrator

Include:

Extend:

Generalization:

Normal Flow of Events:

1. Admin clicks on “APIs” section of their panel
2. Admin enables either the promotions, or reviews APIs, or both
3. Admin clicks on promotions api, adds promotional text and or photos and submits form
4. Admin clicks on reviews API and enables either all or selected reviews to be added to API response and submits form.

SubFlows:

S-1: Upload photo and/or promotional text and save to db
   1. If admin enabled and set promotional API, system uploads photo if applicable and saves link to db, and saves promotional text to db

S-2: Fetch reviews from database
   1. If admin enabled reviews API, system fetches all reviews from users db and outputs them into a datatable
   2. If admin chooses all reviews or chosen review, the system sets the API to broadcast appropriate reviews in its API response

Alternate/Exceptional Flow:

Use Case Name: Staff View Booked Schedule	**ID: 22-1**	**Importance Level:** High
Primary Actor: Staff Member	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Staff member wishes to see the dog walks they are booked for. System checks Authorization API/JWT for role type		
Brief Description: This use case describes the staff members' ability to see their booked schedule and make plans accordingly		
Trigger: Staff member enters the scheduling section of their staff user panel. **Type:**		
Relationships: **Association:** Staff Members **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. Staff member logs on to their account 2. Staff member clicks on the schedule section 3. Schedule view of staff member is shown in main view 4. Staff member can narrow down view to a specific week or date range and the view dynamically reloads appropriately selected week or time span		
SubFlows: S-1: Load schedule calendar 1. Once the member has clicked into their accounts schedule section, the system fetches and populates the schedule calendar on a month calendar, each day being clickable. 2. Once member clicks on specific day, the schedule view changes to that day's schedule broken down into hourly basis S-2: Filtering 1. If a member filters a specific date range or week, the schedule view focuses on that range		
Alternate/Exceptional Flow: S-1: 1. If schedule calendar cannot get data required to populate fields, the member is notified with an error message.		

Use Case Name: Staff Set Schedule of Availability	**ID: 23-1**	**Importance Level:** High
Primary Actor: Staff Member	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Staff member, if given permission wishes to set their own work schedule System checks Authorization API/JWT for role type		
Brief Description: This use case describes some of the staff members' ability to set their own schedules provided they have the permission to do so.		
Trigger: Staff member enters the scheduling section of their staff user panel, chooses their desired work hours for any given week. **Type:** External		
Relationships: **Association:** Staff Members **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. Staff member navigates to schedule section of their panel 2. Staff member clicks "Set Schedule" button 3. Staff member chooses date range to edit 4. Staff member sets work hours for each selected day 5. Staff member clicks submit button		
SubFlows: S-1: Update schedule in db 1. Once Staff member clicks the submit button, system updates the members' schedule in database		
Alternate/Exceptional Flow: S-1: 1. If any conflicts exist, the view remains the same, nothing is modified, and user is notified about the conflict.		

Use Case Name: Staff Set Dog Handling Limit	**ID: 24-1**	**Importance Level:** High
Primary Actor: Staff Member	**Use Case Type:** Detail, Essential	
Stakeholders and Interests:	Staff member wishes to set their maximum dog/dog weight limt that they can handle at a time System checks Authorization API/JWT for role type	
Brief Description:	This use case describes the dog walker's ability to set their maximum handling ability in terms of both weight classes of dogs, and amount of dogs they can take on at once	
Trigger:	Staff member enters their profile and sets both the dog amount and weight classes they can handle for each walk	
Type:	External	
Relationships: **Association:** Staff Members **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. Staff member navigates to their account preferences 2. Staff member clicks "edit workload limits" button 3. Staff member sets maximum number of dogs they can take on and the maximum weight limits 4. Staff member clicks submit button		
SubFlows: S-1: Update user profile in database 1. Once member submits their preferences, the system updates the members database table		
Alternate/Exceptional Flow: S-1: 1. If update fails, member is notified with error message		

Use Case Name: Staff See Hours Worked and Paid Out	**ID: 25-1**	**Importance Level:** High
Primary Actor: Staff Member	**Use Case Type:** Detail, Essential	
Stakeholders and Interests:	Staff member wishes to set the number of hours they've worked, and how much of it was paid out to them in a given time span System checks Authorization API/JWT for role type	
Brief Description:	This use case describes the staff members ability to track their hours worked and how much of it was paid	
Trigger:	Staff member enters scheduling section, chooses time span, and sees total of hours worked and total of hours paid out	
Type:	External	

Relationships:

Association: Staff Members

Include:

Extend:

Generalization:

Normal Flow of Events:

1. Staff member navigates to scheduling section
2. Staff member clicks "check hours worked/paid" button
3. Staff member chooses date range from day of checking to any point in past
4. Staff member views data table of days and hours worked and whether it was paid out or not

SubFlows:

S-1: Get hours and pay status from database

1. Once the member has chosen their date range, the system fetches hours worked and their pay status

Alternate/Exceptional Flow:

S-1:

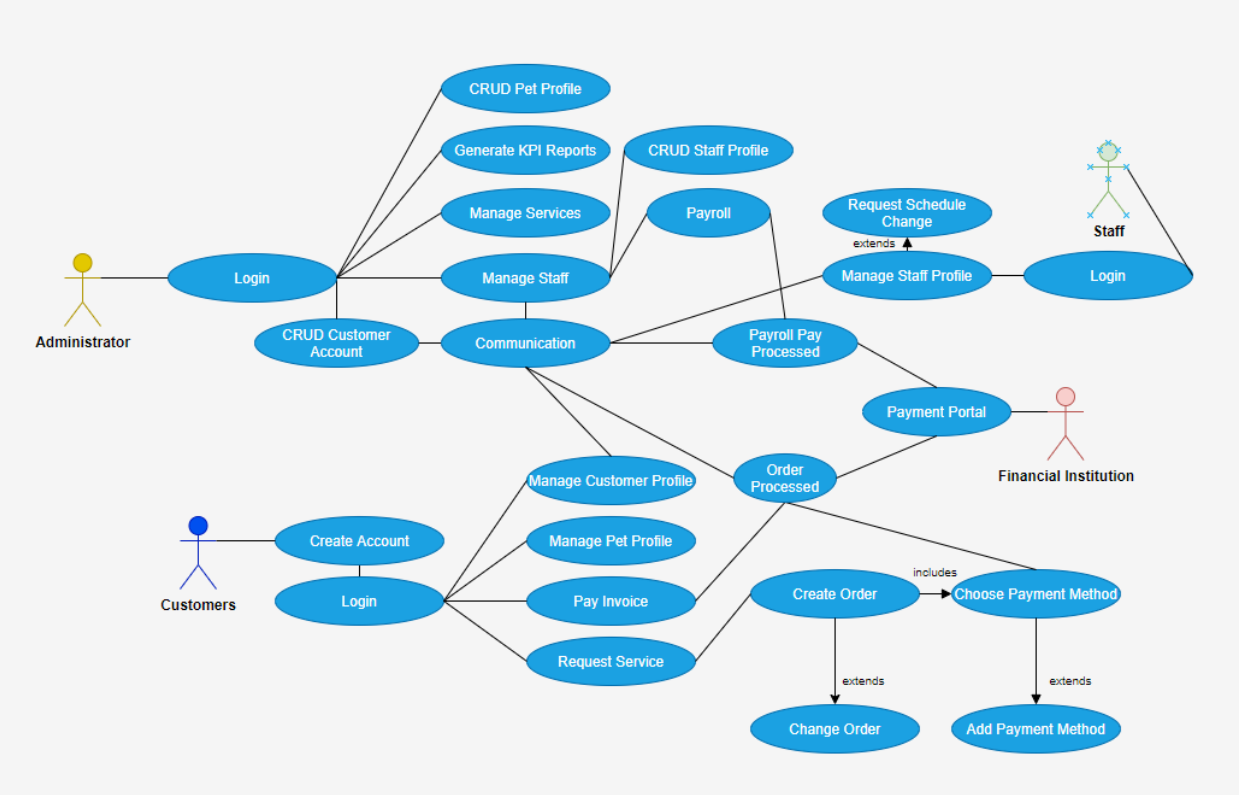
1. If there are no hours in date range, the data table does not load, instead the view shows a message notifying the user that there are no hours on record for selected time period

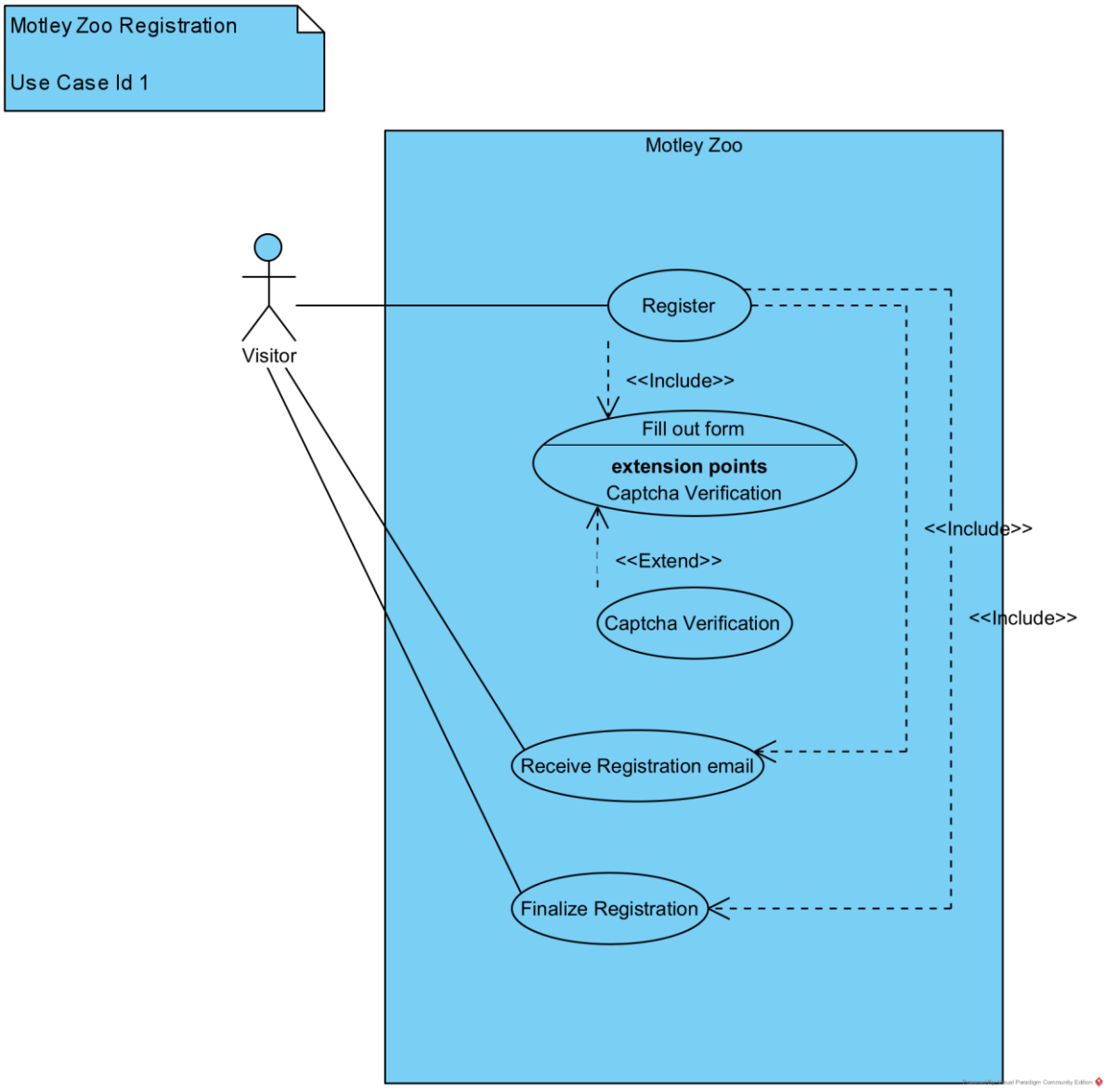
Use Case Name: Staff Reports to Admin	**ID:**26-1	**Importance Level:** High
Primary Actor: Staff Member	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Staff member wishes to write report on dog walk or client to the owner System checks Authorization API/JWT for role type		
Brief Description: This use case describes the staff members optional ability to write a report to the owner regarding a dog walk or interaction with a client		
Trigger: Staff member enters main page of app, chooses a job from a list of completed jobs, and writes a private report to the owner. **Type:** External		
Relationships: **Association:** Staff Members **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. Staff member enters main page of their user panel 2. Staff member chooses a job from a list of recently completed jobs 3. Staff member fills out form and optionally adds pictures for uploading 4. Staff member submits form 5. Administrator is notified		
SubFlows: S-1: Load jobs into data table 1. System loads a range of recently finished jobs into a data table 2. System dynamically loads report form for each clicked job 3. If report for job is already done, data table row/entry reflects this		
Alternate/Exceptional Flow: S-1 1. Once member submits form, the report is added to database 2. If any pictures were added, they get uploaded to file system (S3 bucket) and the link to it gets also added to database		

Use Case Name: Staff Reports to Clients	**ID:**27-1	**Importance Level:** High
Primary Actor: Staff Member	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Staff member writes a report of a dog walk to client System checks Authorization API/JWT for role type		
Brief Description: This use case describes the staff members ability to write a report to the client regarding a dog walk		
Trigger: Staff member enters main page of app, chooses a job from a list of completed jobs, and writes a report to the owner with ability to upload a picture. **Type:** External		
Relationships: **Association:** Staff Members, Clients **Include:** **Extend:** 26-1 **Generalization:**		
Normal Flow of Events: **Normal Flow of Events:** 1. Staff member enters main page of their user panel 2. Staff member chooses a job from a list of recently completed jobs 3. Staff member fills out form and optionally adds pictures for uploading 4. Staff member has option to also check that admin is also included 4. Staff member submits form 5. User and, if applicable, Admin is/are notified		
SubFlows: S-1: Load jobs into data table 1. System loads a range of recently finished jobs into a data table 2. System dynamically loads report form for each clicked job 3. If report for job is already done, data table row/entry reflects this		
Alternate/Exceptional Flow: S-1: 1. Once member submits form, the report is added to database 2. If any pictures were added, they get uploaded to file system (S3 bucket) and the link to it gets also added to database		

Use Case Name: Quick Chat	**ID:28-1**	**Importance Level:** High
Primary Actor: Staff Member, User, Admin	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: Staff member messages a client or the owner/administrator System checks Authorization API/JWT for role type		
Brief Description: This use case describes the staff members ability to contact either the client or administrator in a chat format		
Trigger: Staff member enters communications section of UI, and starts a chat with person they wish to speak with **Type:** External		
Relationships: **Association:** Staff Members, Clients, Administrator **Include:** **Extend:** **Generalization:**		
Normal Flow of Events: 1. User/Staff Member/Admin open chat window 2. They choose relevant person to message – for users and staff members, only admin, staff member on the job, or user who ordered the job will be given choices 3. Chat is initiated		
SubFlows: S-1: Reflect chat to recipient's account 1. As one person messages another, the system should immediately send that message to the other persons profile and notify them		
Alternate/Exceptional Flow: S-1: 1. All messaging through chat should be saved in a database table as history and in case the person is not logged on at the moment the message is sent.		

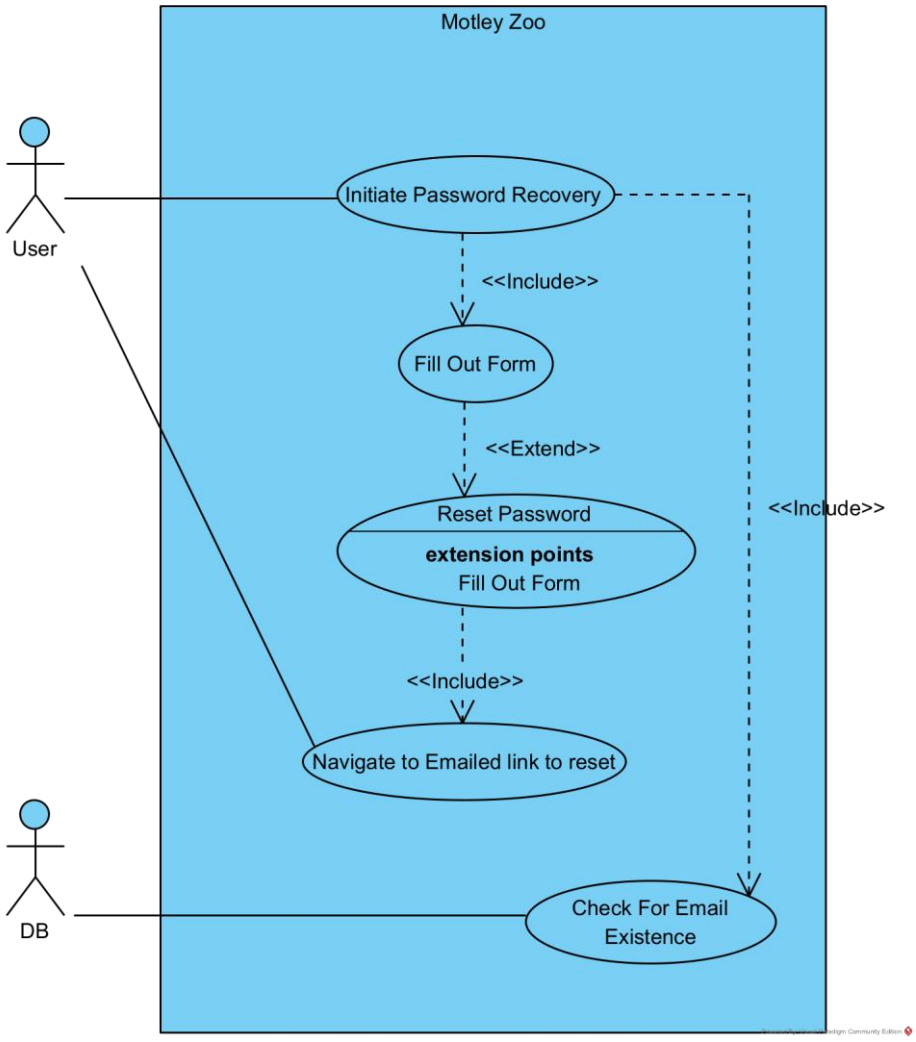
3.2 Use Case Diagrams

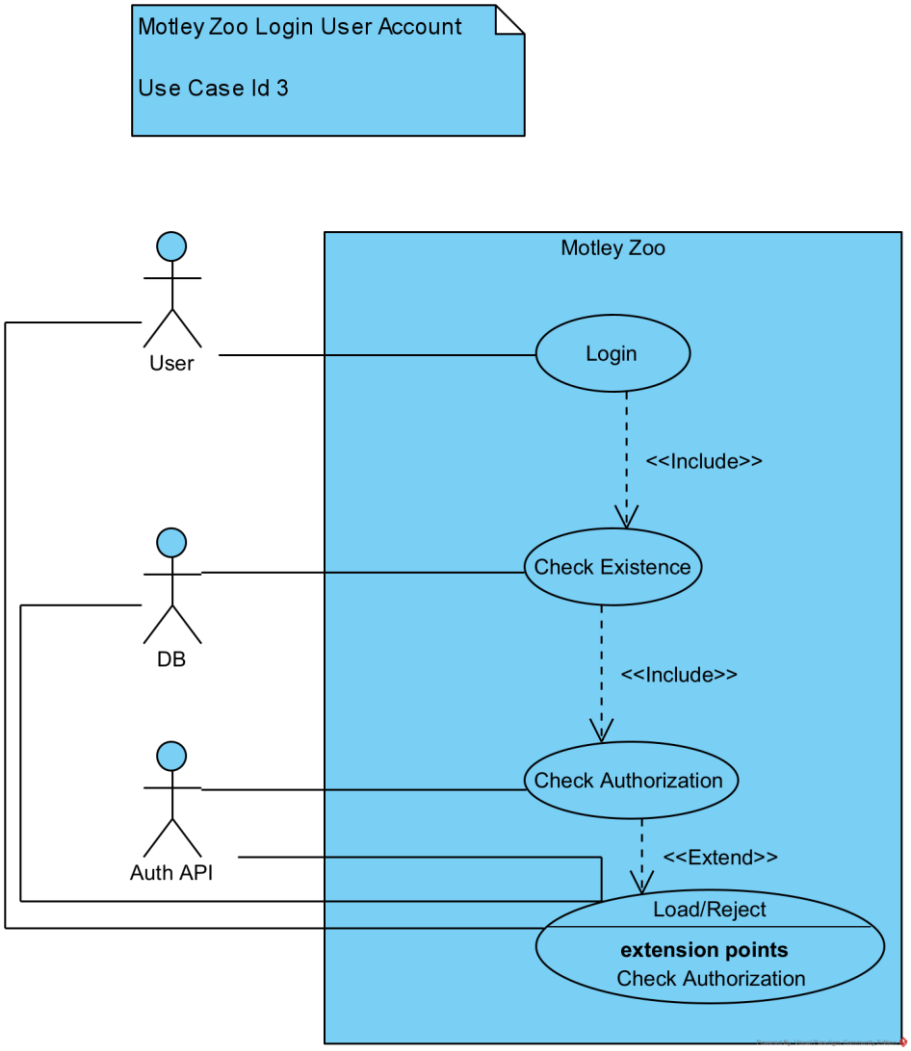


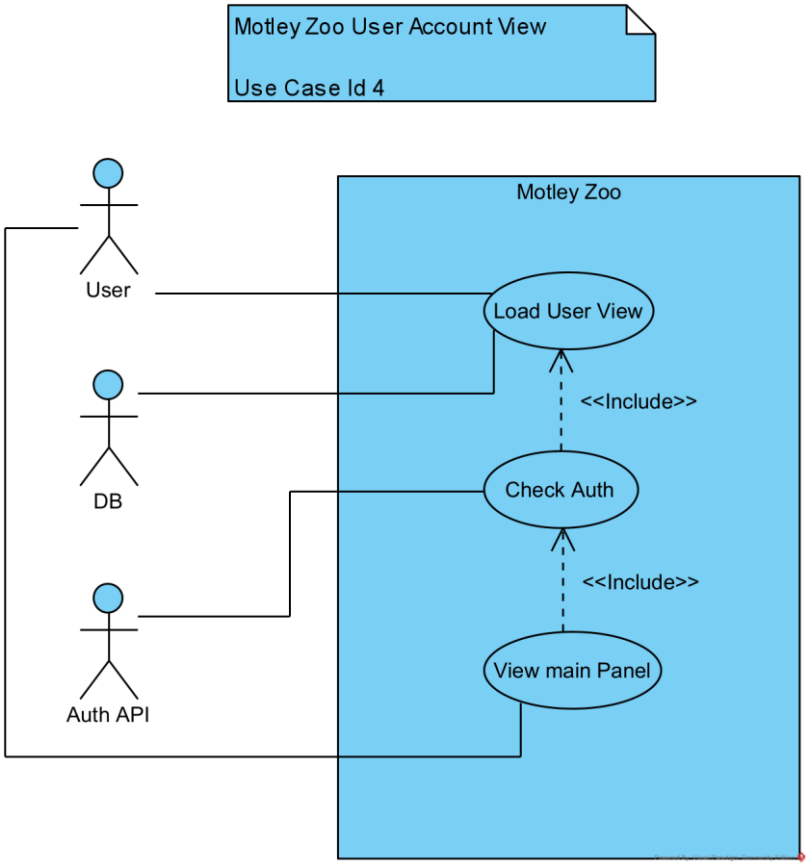

Motley Zoo Registration
Use Case Id 1
Motley Zoo
Visitor
Register
<<Include>>
Fill out form
extension points
Captcha Verification
<<Extend>>
Captcha Verification
<<Include>>
<<Include>>
Receive Registration email
Finalize Registration

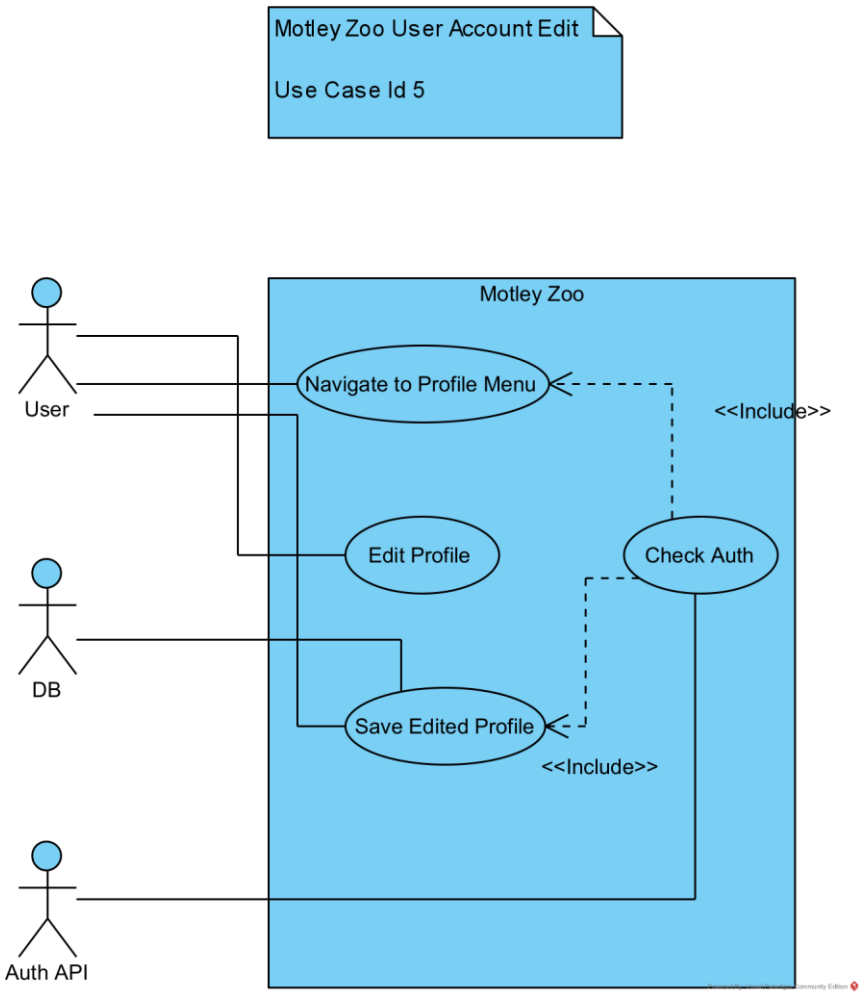
Motley Zoo Reset User Password

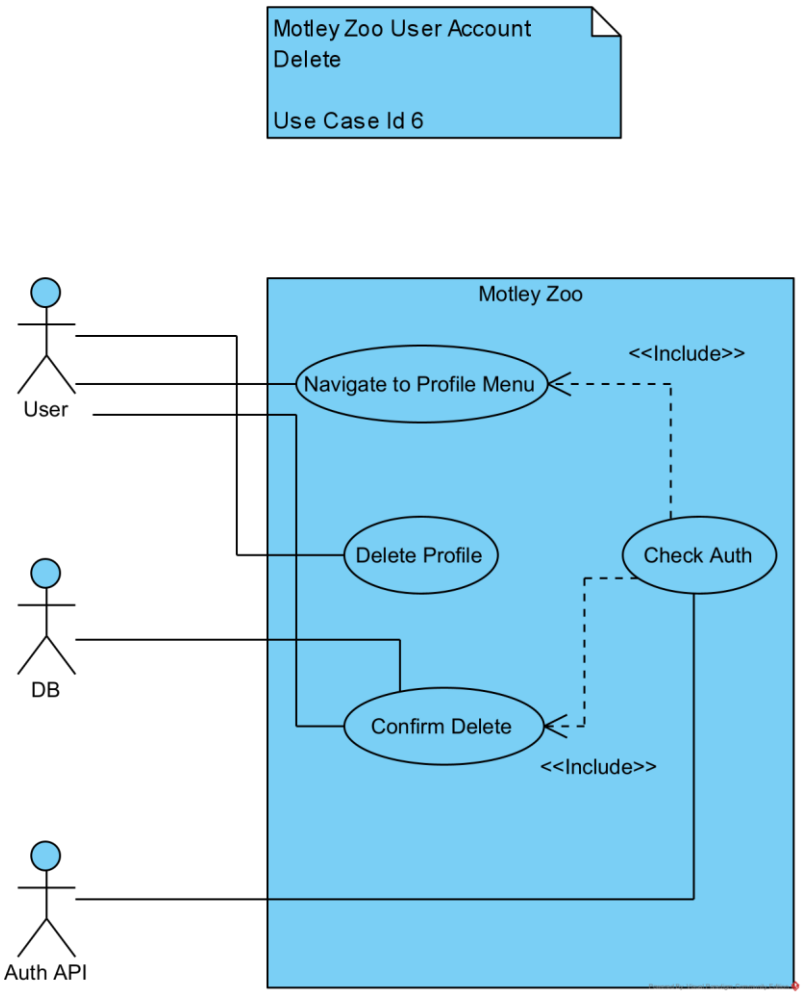
Use Case Id 2

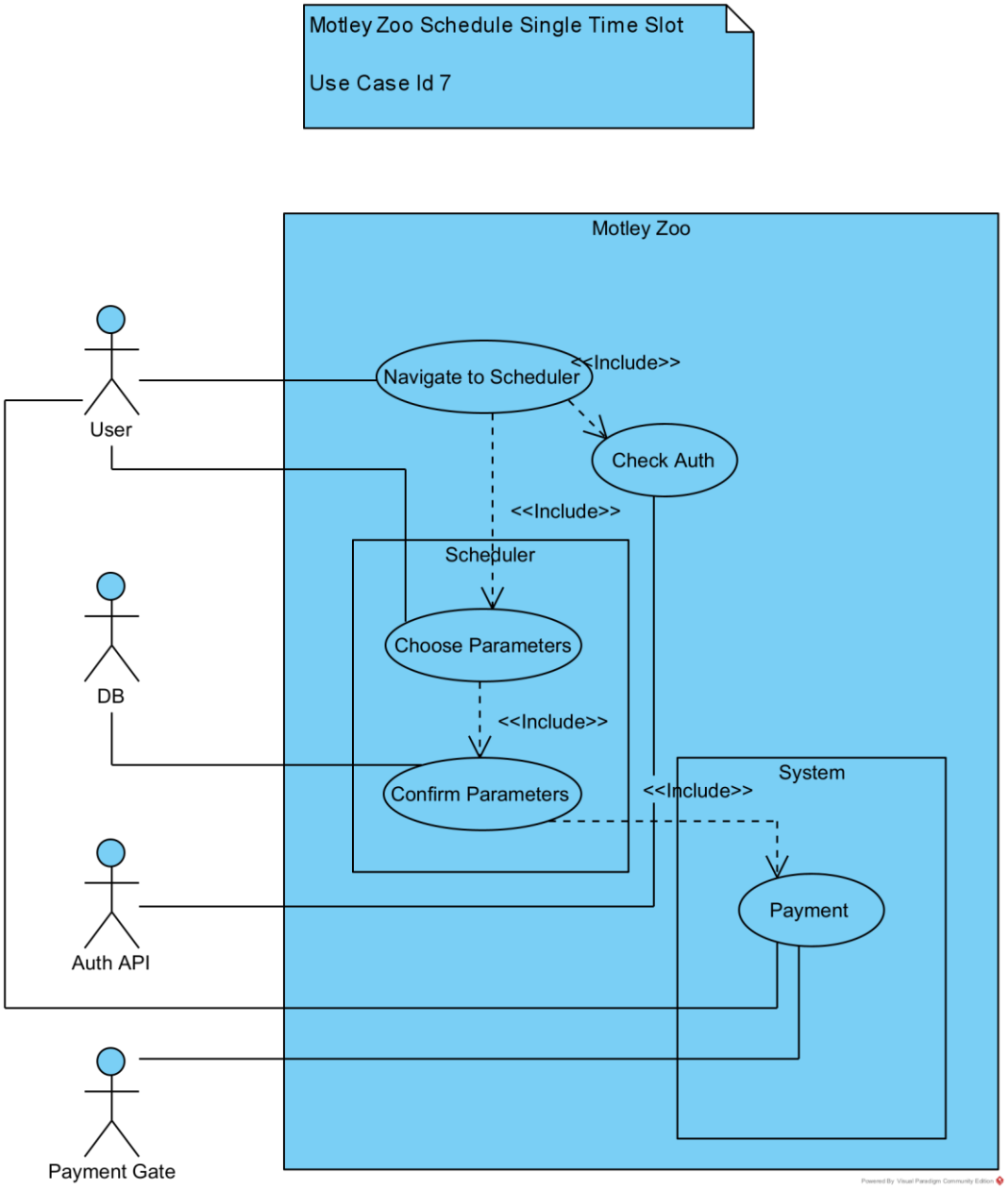


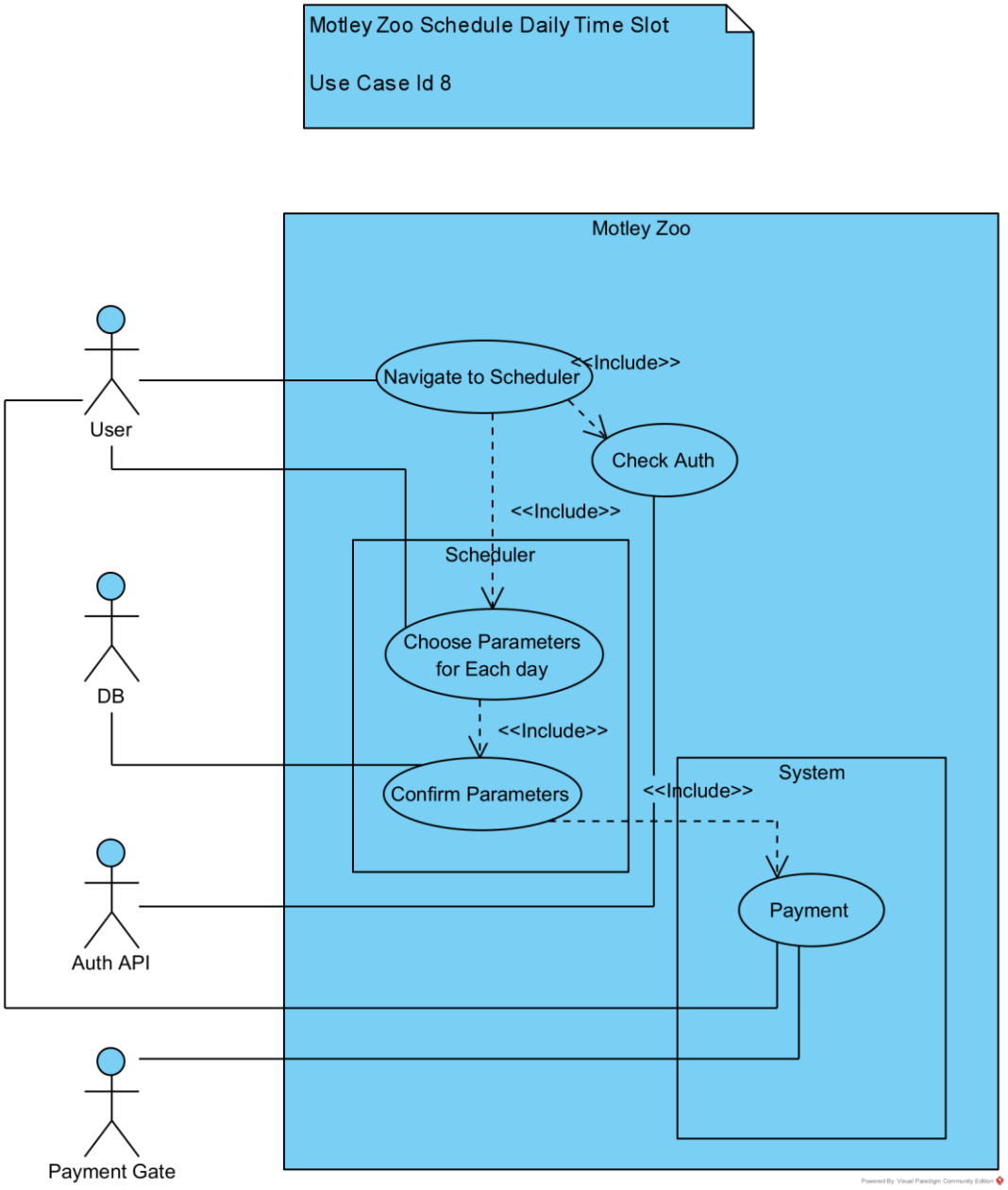

Motley Zoo Login User Account
Use Case Id 3
Motley Zoo
User
Login
<<Include>>
Check Existence
DB
<<Include>>
Check Authorization
Auth API
<<Extend>>
Load/Reject
extension points
Check Authorization

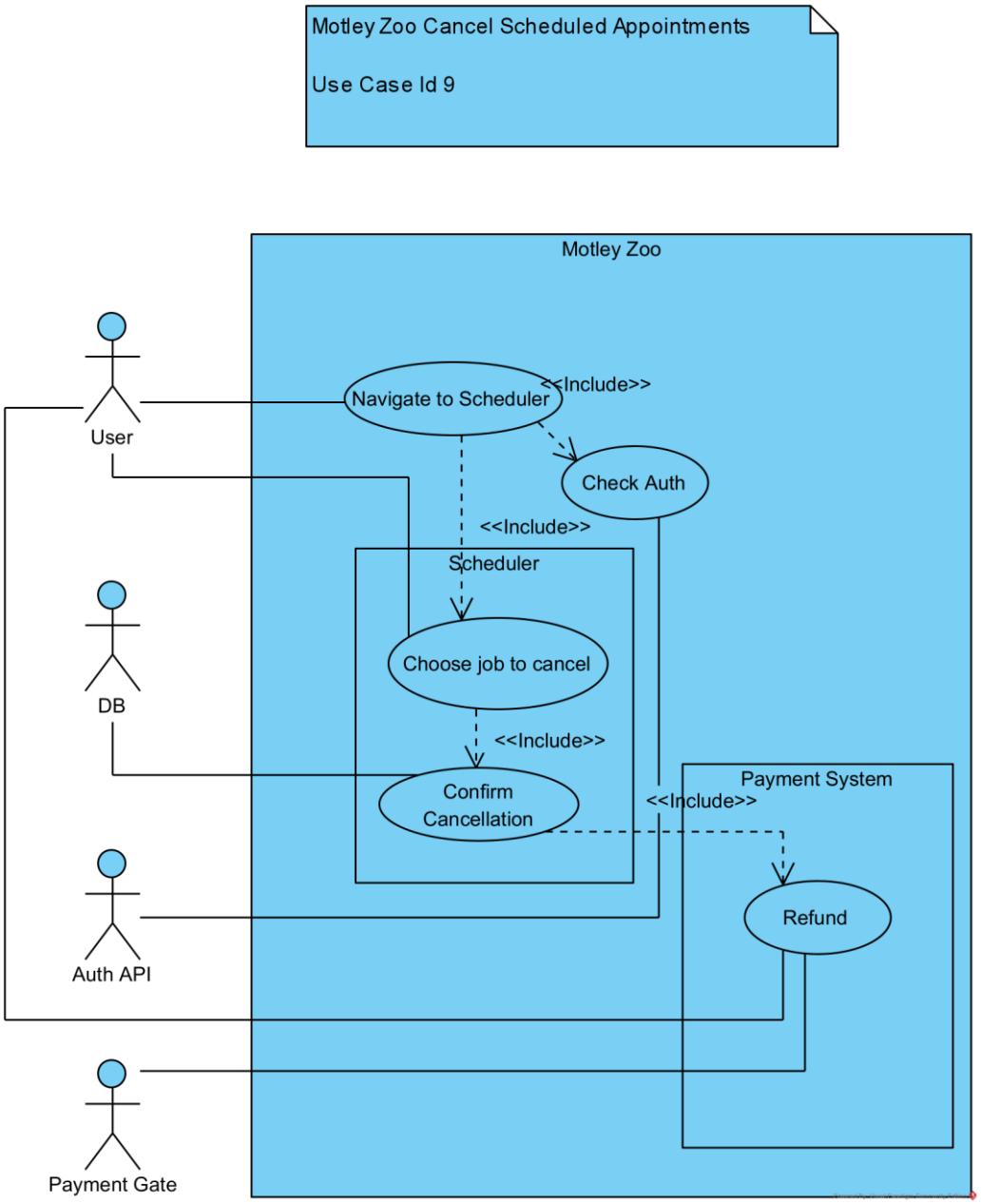

Motley Zoo User Account View
Use Case Id 4
Motley Zoo
User
DB
Auth API
Load User View
<<Include>>
Check Auth
<<Include>>
View main Panel

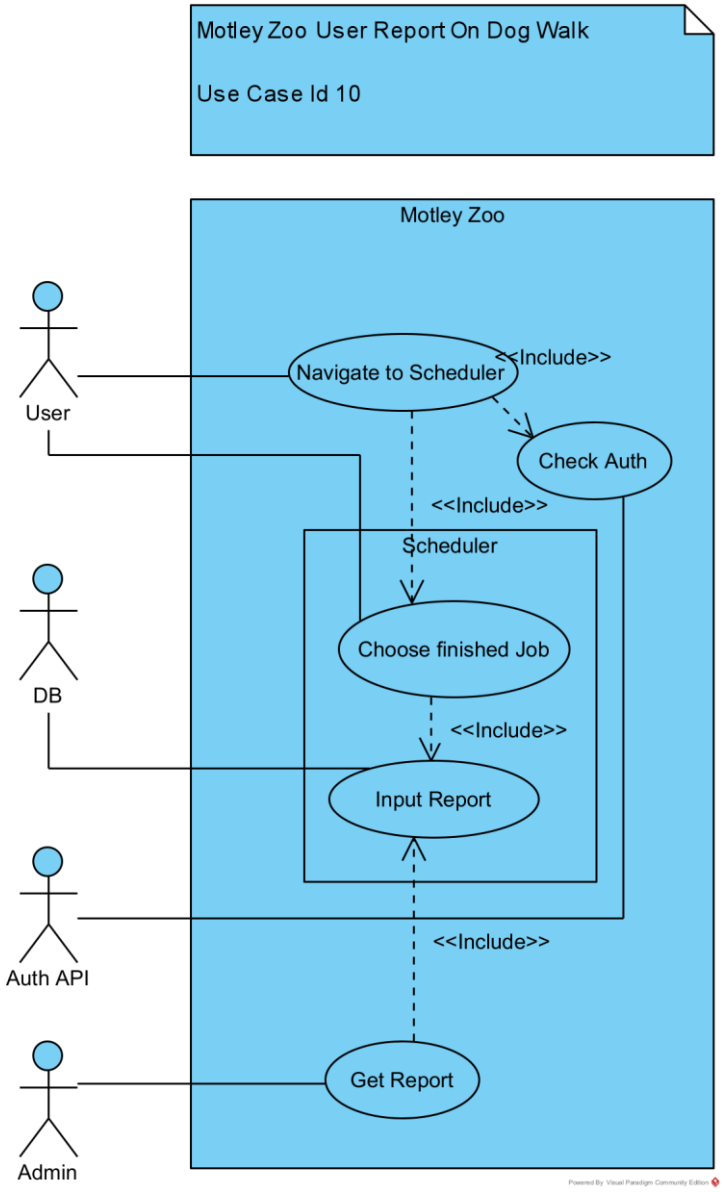

Motley Zoo User Account Edit
Use Case Id 5
Motley Zoo
User
Navigate to Profile Menu
<<Include>>
Edit Profile
Check Auth
DB
Save Edited Profile
<<Include>>
Auth API

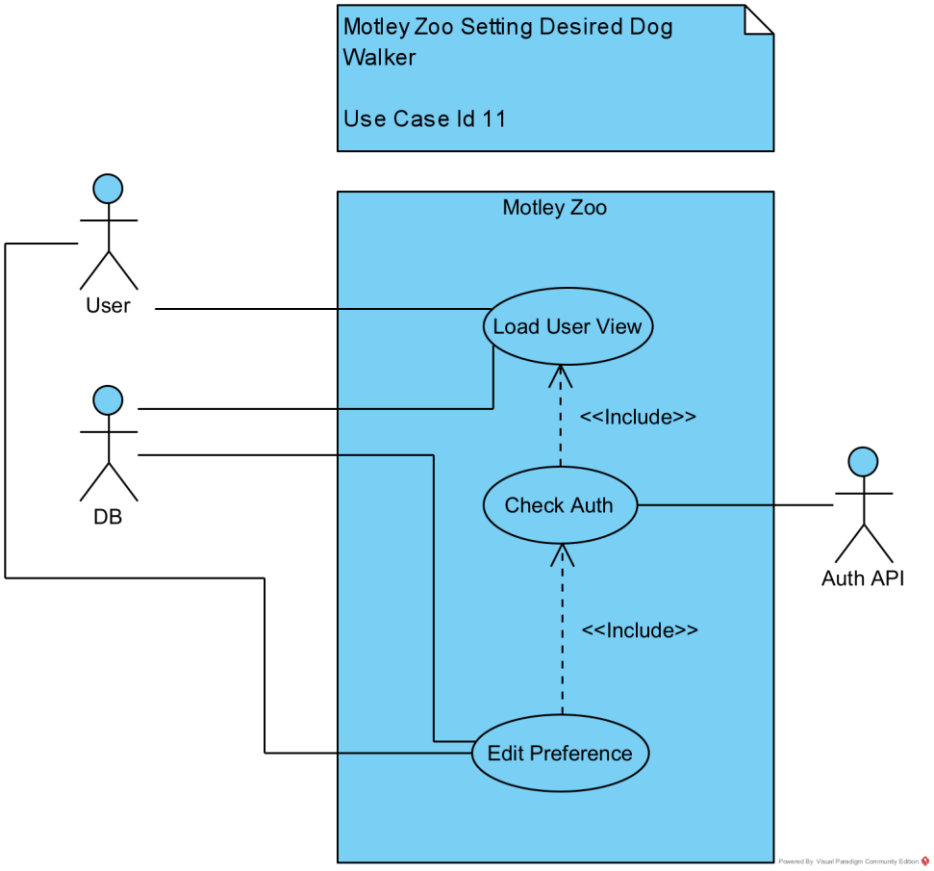

Motley Zoo User Account Delete
Use Case Id 6
Motley Zoo
<<Include>>
Navigate to Profile Menu
User
Delete Profile
Check Auth
DB
Confirm Delete
<<Include>>
Auth API

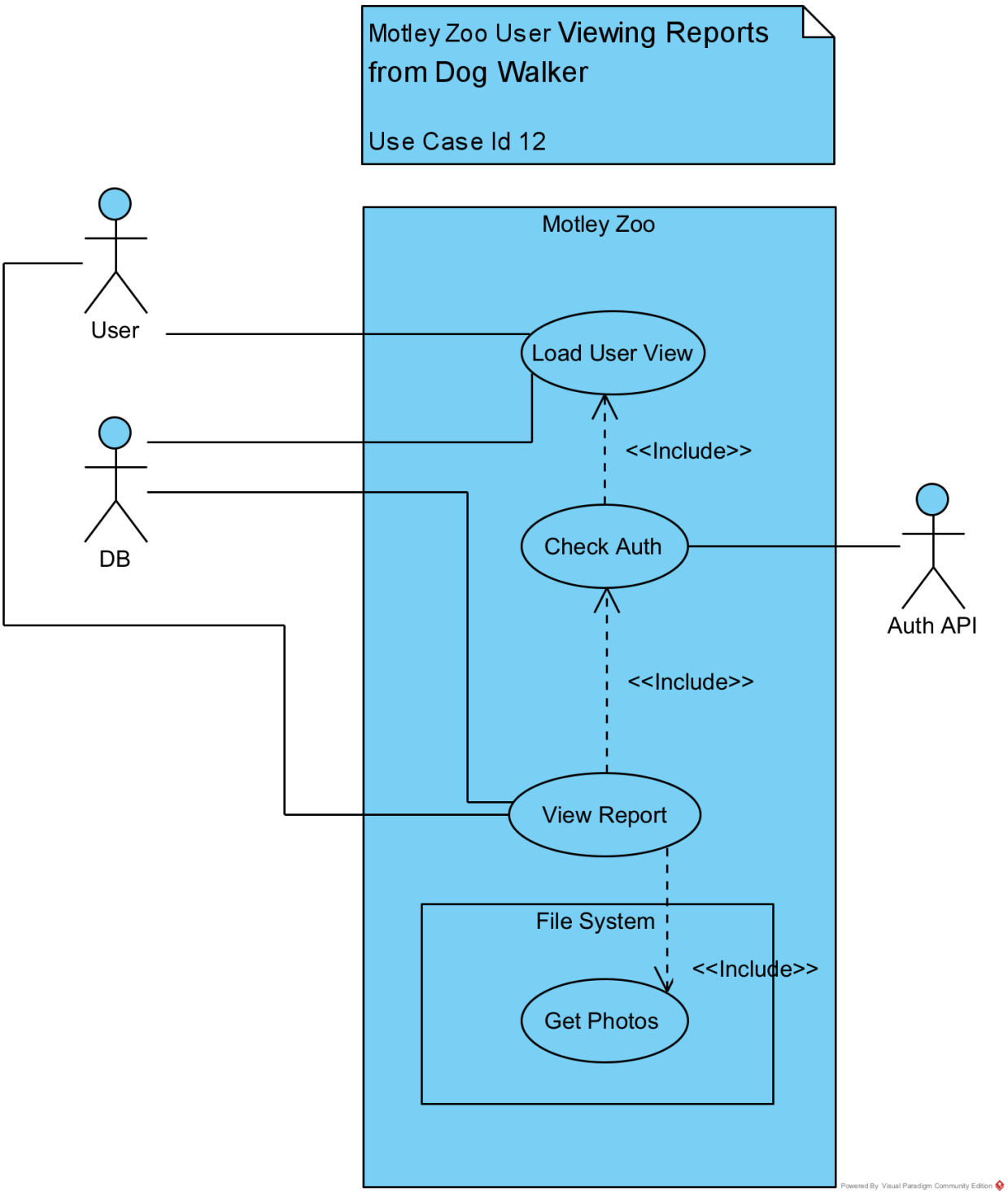

Motley Zoo Schedule Single Time Slot
Use Case Id 7
Motley Zoo
User
Navigate to Scheduler
<<Include>>
Check Auth
<<Include>>
Scheduler
DB
Choose Parameters
<<Include>>
Confirm Parameters
<<Include>>
System
Payment
Auth API
Payment Gate

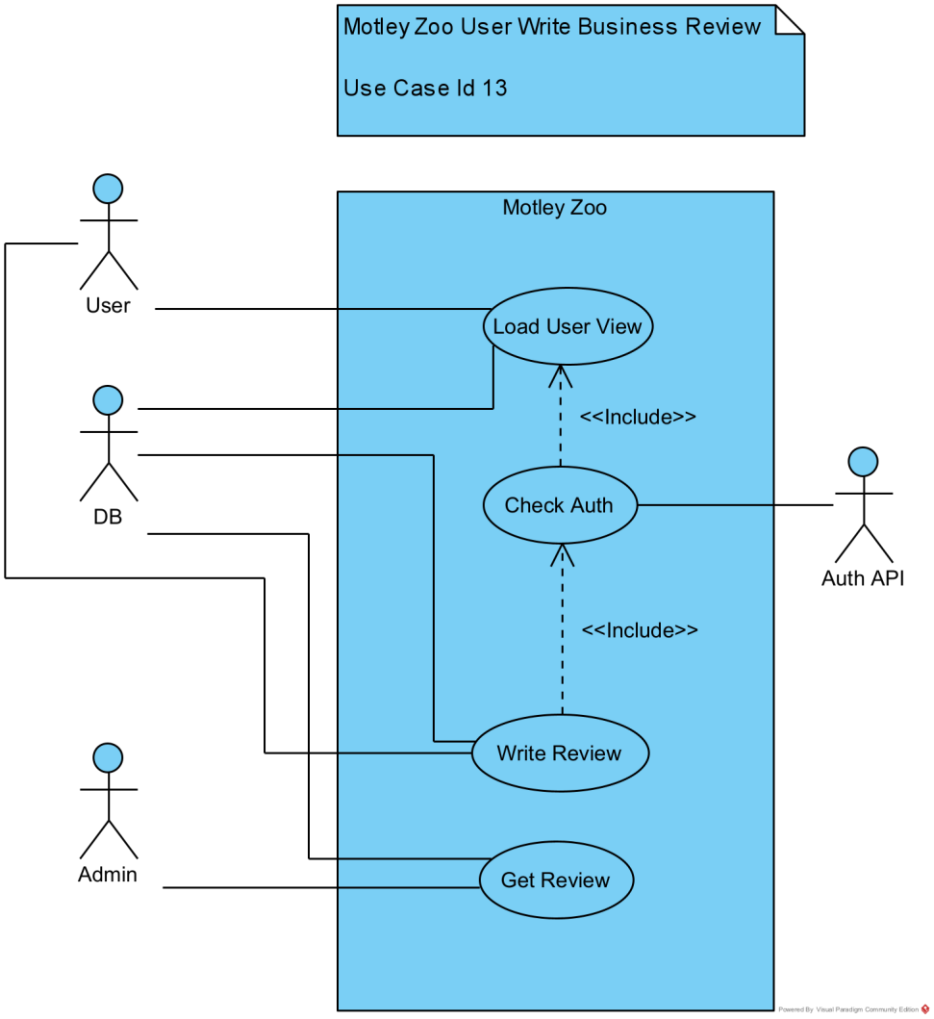

Motley Zoo Schedule Daily Time Slot
Use Case Id 8
Motley Zoo
User
Navigate to Scheduler
<<Include>>
Check Auth
<<Include>>
Scheduler
DB
Choose Parameters for Each day
<<Include>>
Confirm Parameters
<<Include>>
System
Payment
Auth API
Payment Gate

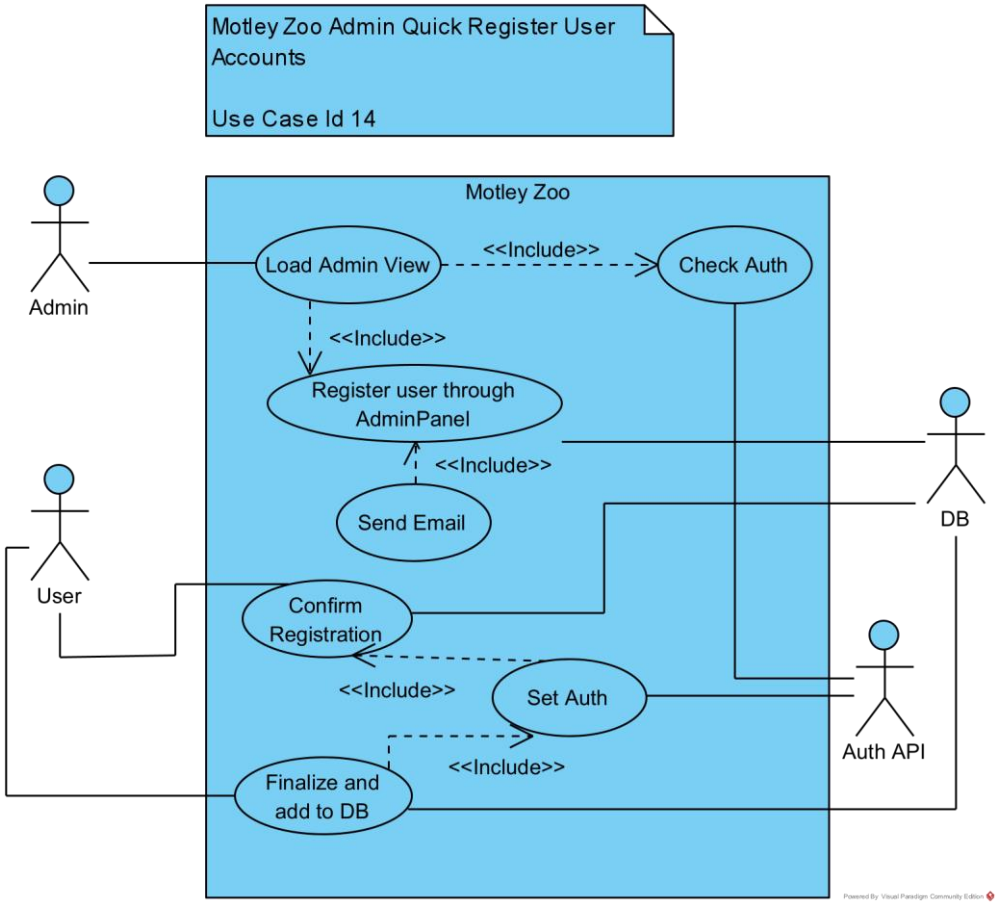

Motley Zoo Cancel Scheduled Appointments
Use Case Id 9
Motley Zoo
User
Navigate to Scheduler
<<Include>>
Check Auth
<<Include>>
Scheduler
DB
Choose job to cancel
<<Include>>
Confirm Cancellation
Payment System
<<Include>>
Refund
Auth API
Payment Gate

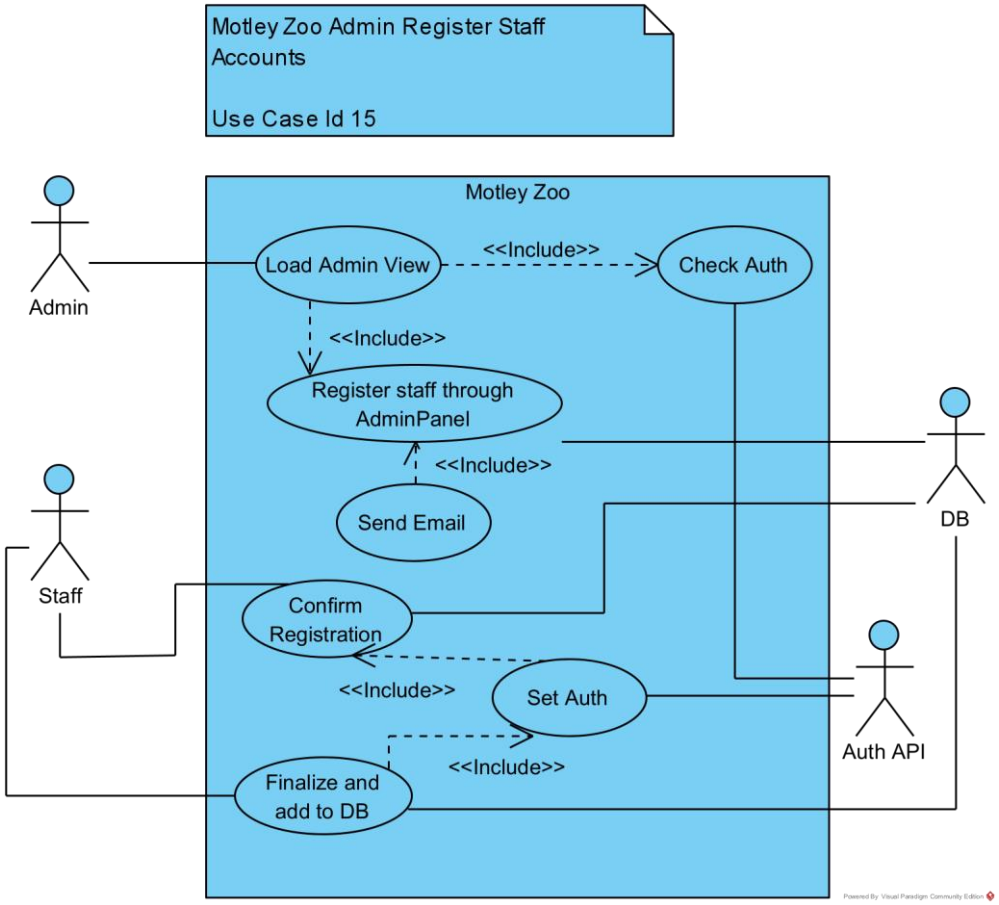

Motley Zoo User Report On Dog Walk
Use Case Id 10
Motley Zoo
User
Navigate to Scheduler
<<Include>>
Check Auth
<<Include>>
Scheduler
DB
Choose finished Job
<<Include>>
Input Report
Auth API
<<Include>>
Get Report
Admin

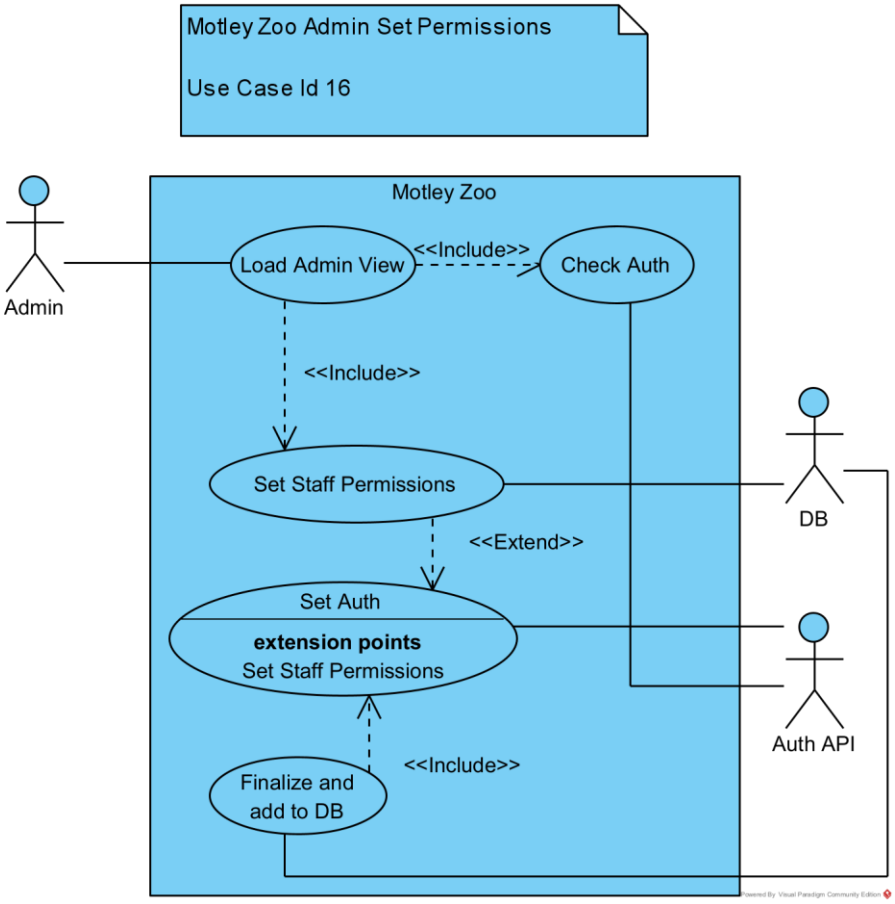

Motley Zoo Setting Desired Dog Walker
Use Case Id 11
Motley Zoo
User
DB
Load User View
<<Include>>
Check Auth
Auth API
<<Include>>
Edit Preference

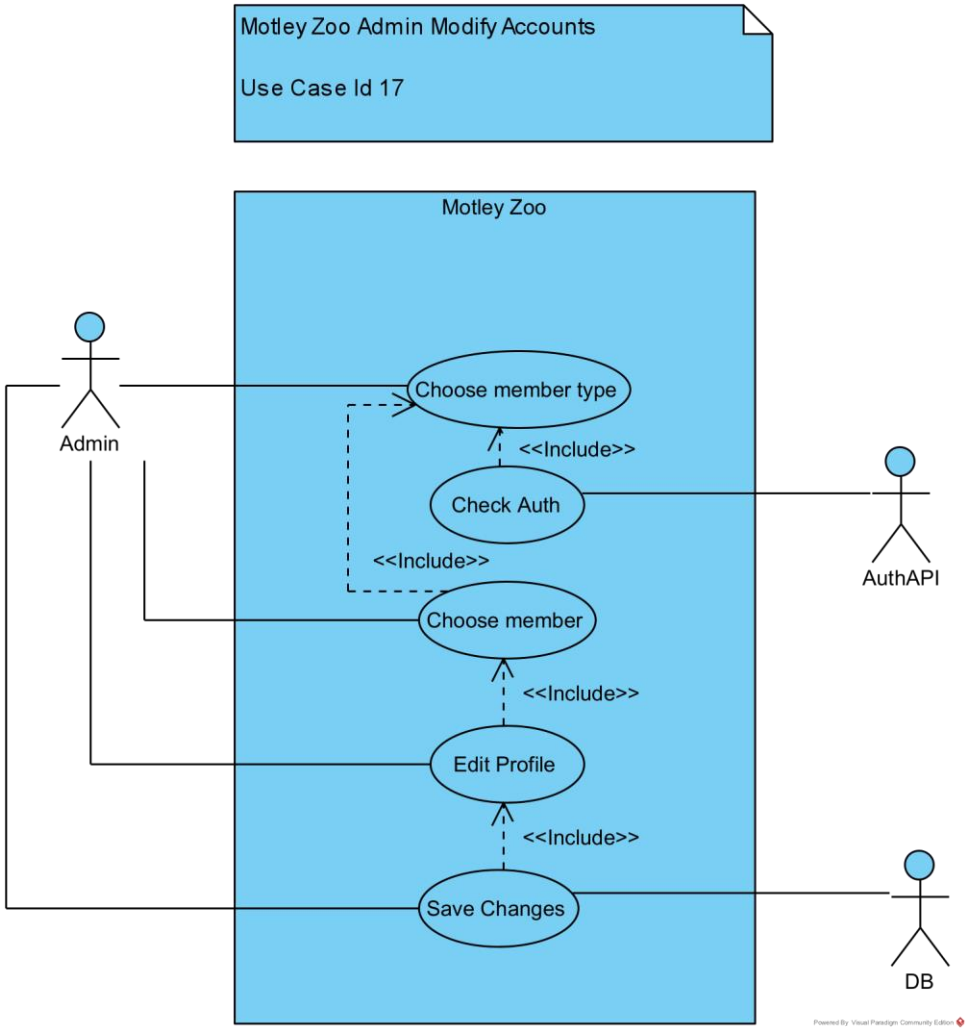

Motley Zoo User Viewing Reports from Dog Walker
Use Case Id 12
Motley Zoo
User
DB
Load User View
<<Include>>
Check Auth
Auth API
<<Include>>
View Report
File System
<<Include>>
Get Photos
Powered By Visual Paradigm Community Edition

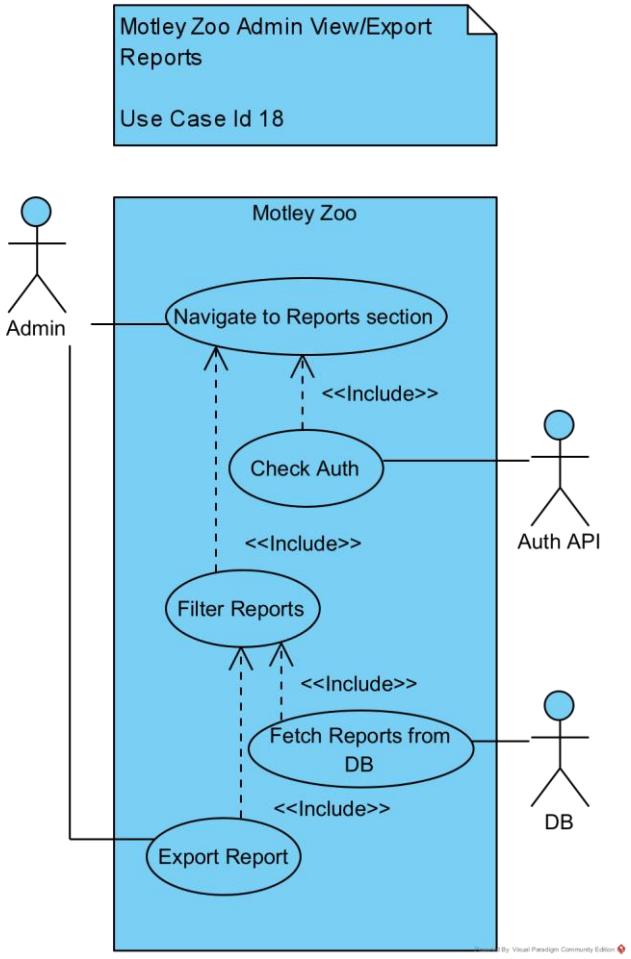

Motley Zoo User Write Business Review
Use Case Id 13
Motley Zoo
User
DB
Admin
Load User View
<<Include>>
Check Auth
<<Include>>
Write Review
Get Review
Auth API

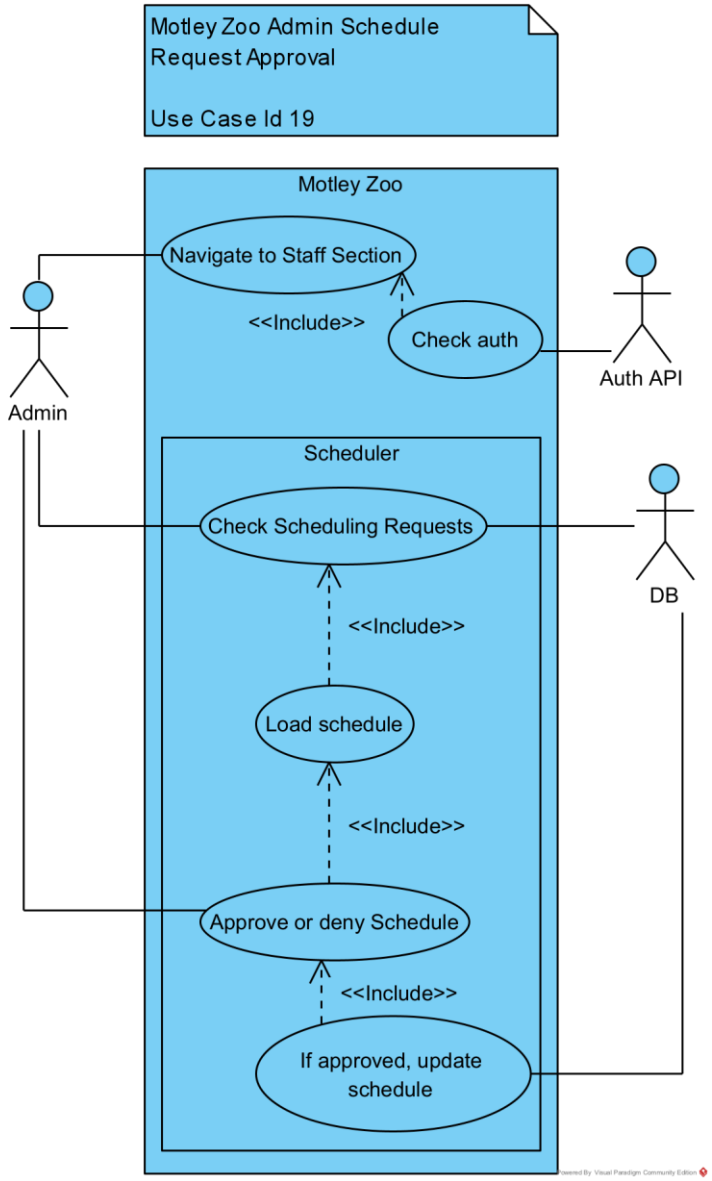

Motley Zoo Admin Quick Register User Accounts
Use Case Id 14
Motley Zoo
Admin
Load Admin View
<<Include>>
Check Auth
<<Include>>
Register user through AdminPanel
<<Include>>
Send Email
DB
User
Confirm Registration
<<Include>>
Set Auth
Auth API
<<Include>>
Finalize and add to DB

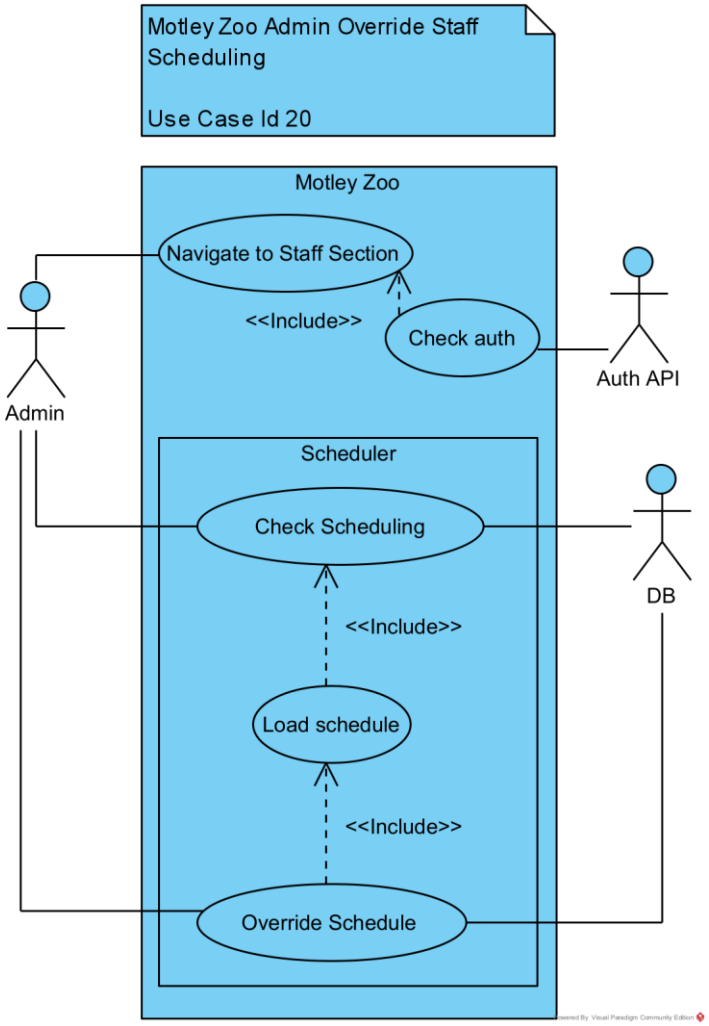

Motley Zoo Admin Register Staff Accounts
Use Case Id 15
Motley Zoo
Admin
Load Admin View
<<Include>>
Check Auth
<<Include>>
Register staff through AdminPanel
<<Include>>
Send Email
DB
Staff
Confirm Registration
<<Include>>
Set Auth
Auth API
<<Include>>
Finalize and add to DB

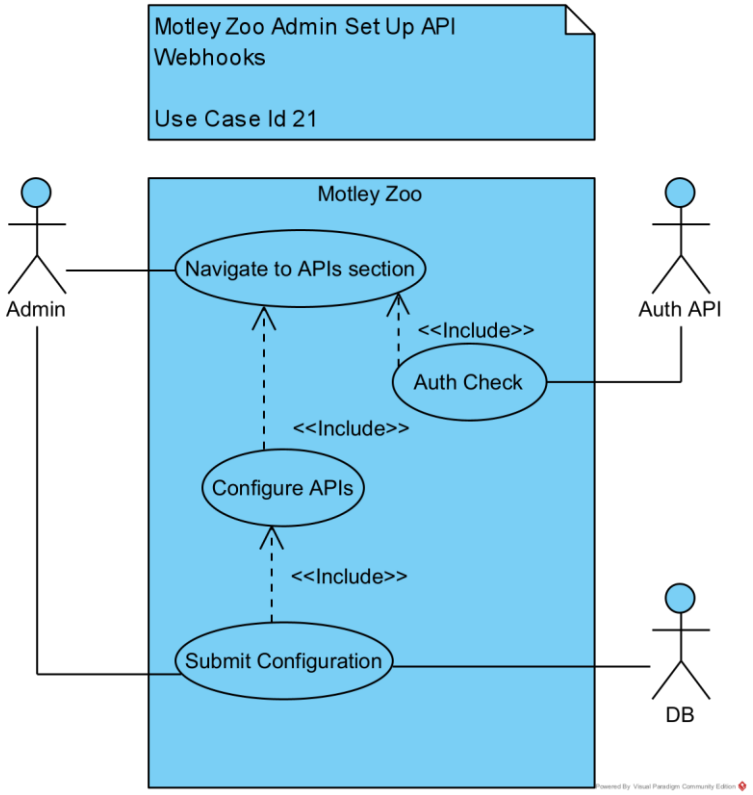

Motley Zoo Admin Set Permissions
Use Case Id 16
Motley Zoo
Admin
Load Admin View
<<Include>>
Check Auth
<<Include>>
Set Staff Permissions
DB
<<Extend>>
Set Auth
extension points
Set Staff Permissions
Auth API
<<Include>>
Finalize and
add to DB

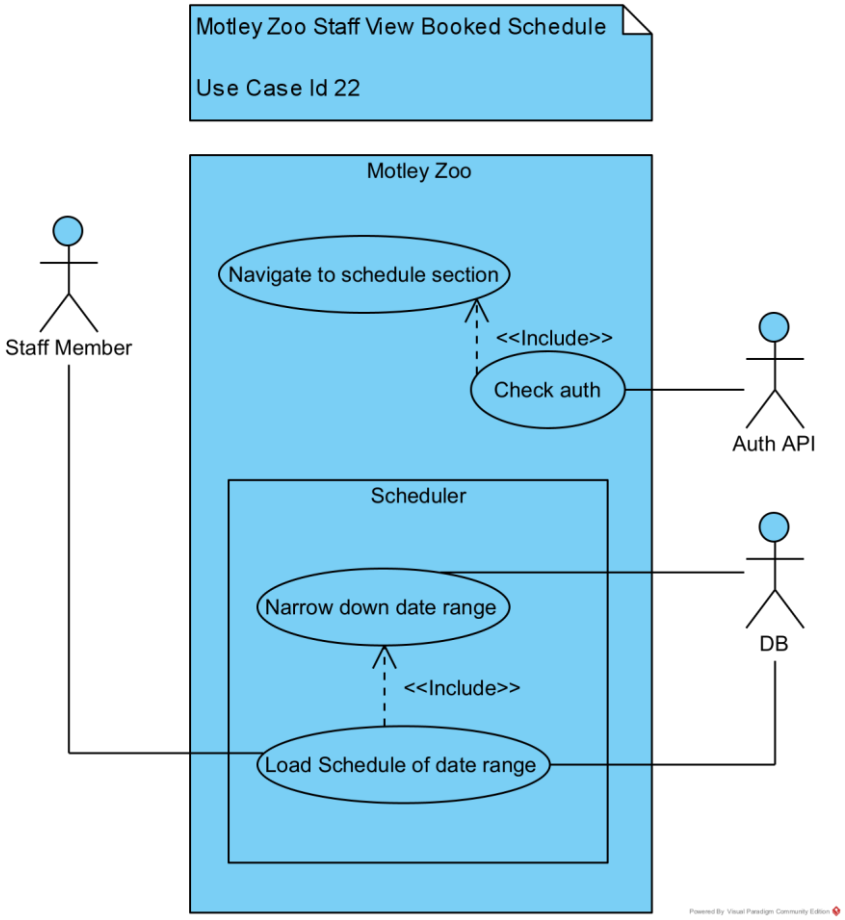

Motley Zoo Admin Modify Accounts
Use Case Id 17
Motley Zoo
Admin
Choose member type
<<Include>>
Check Auth
AuthAPI
<<Include>>
Choose member
<<Include>>
Edit Profile
<<Include>>
Save Changes
DB

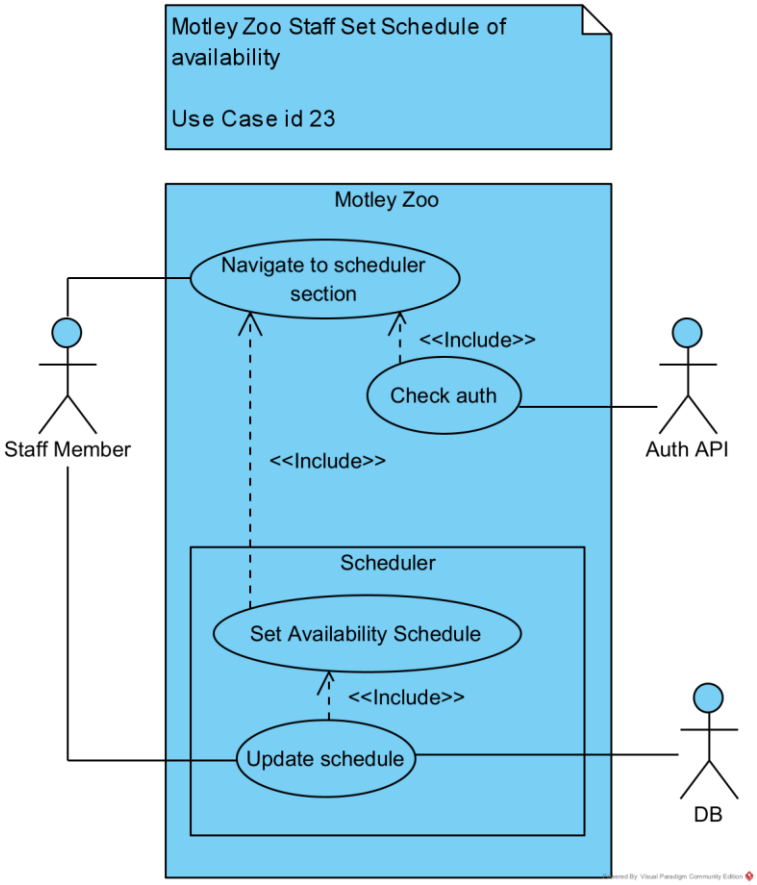

Motley Zoo Admin View/Export Reports
Use Case Id 18
Motley Zoo
Admin
Navigate to Reports section
<<Include>>
Check Auth
Auth API
<<Include>>
Filter Reports
<<Include>>
Fetch Reports from DB
DB
<<Include>>
Export Report

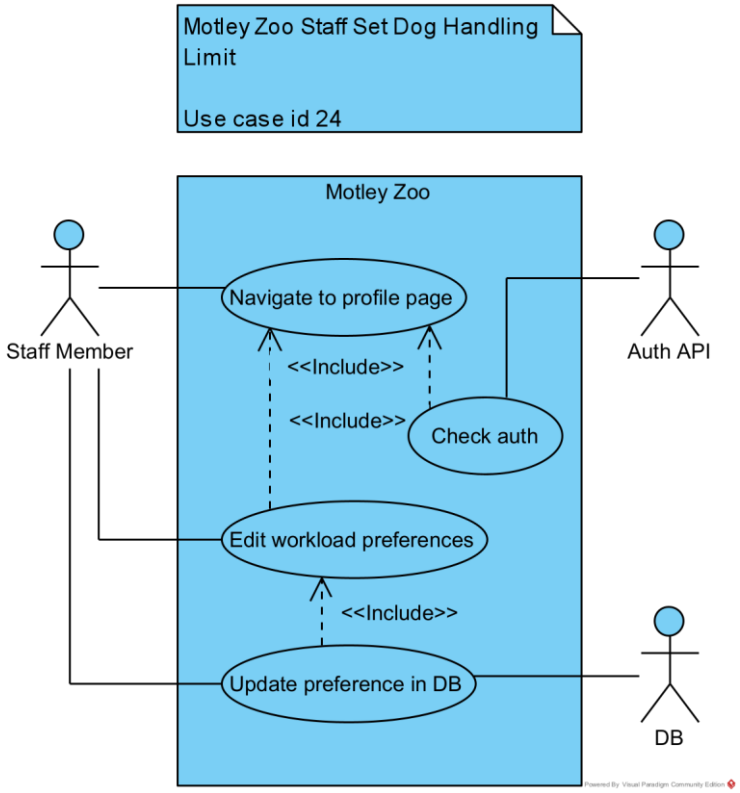

Motley Zoo Admin Schedule Request Approval
Use Case Id 19
Motley Zoo
Navigate to Staff Section
<<Include>>
Check auth
Admin
Auth API
Scheduler
Check Scheduling Requests
DB
<<Include>>
Load schedule
<<Include>>
Approve or deny Schedule
<<Include>>
If approved, update schedule

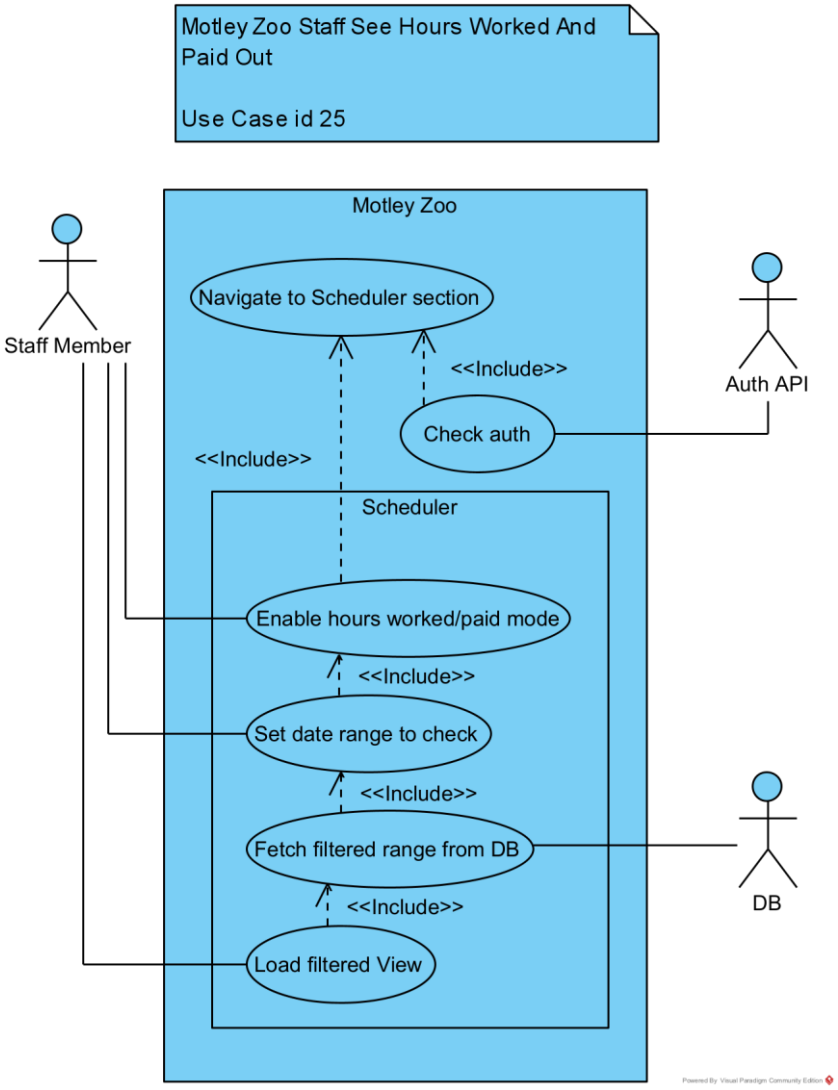

Motley Zoo Admin Override Staff Scheduling
Use Case Id 20
Motley Zoo
Navigate to Staff Section
<<Include>>
Check auth
Admin
Auth API
Scheduler
Check Scheduling
DB
<<Include>>
Load schedule
<<Include>>
Override Schedule

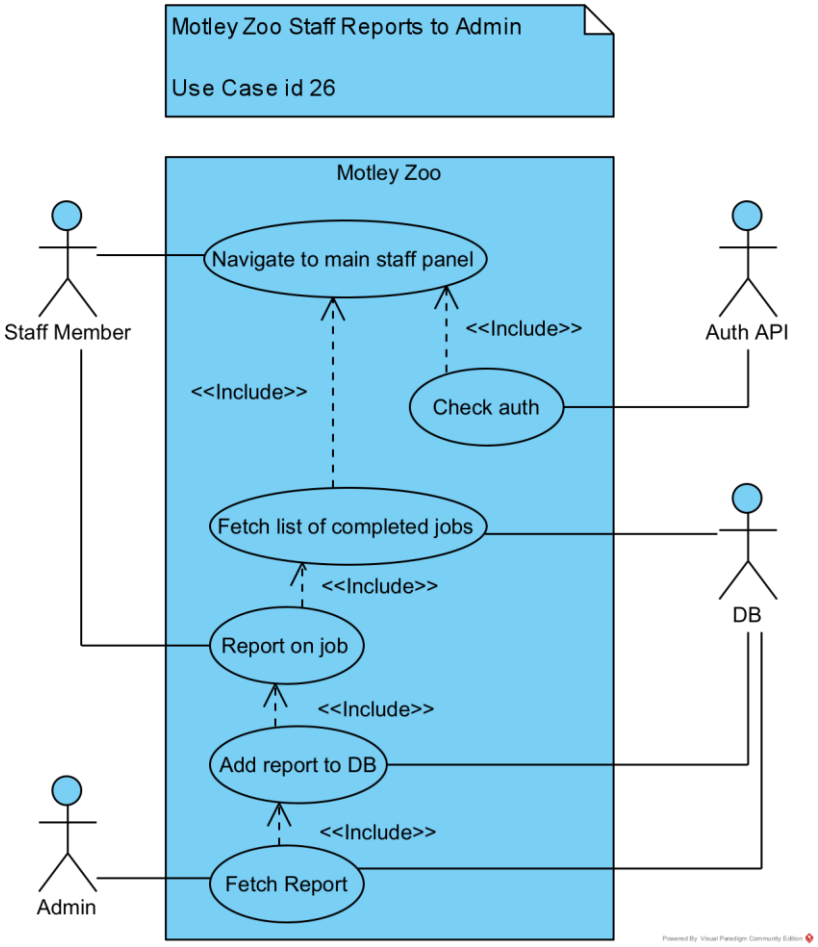

Motley Zoo Admin Set Up API Webhooks
Use Case Id 21
Motley Zoo
Admin
Navigate to APIs section
Auth API
<<Include>>
Auth Check
<<Include>>
Configure APIs
<<Include>>
Submit Configuration
DB

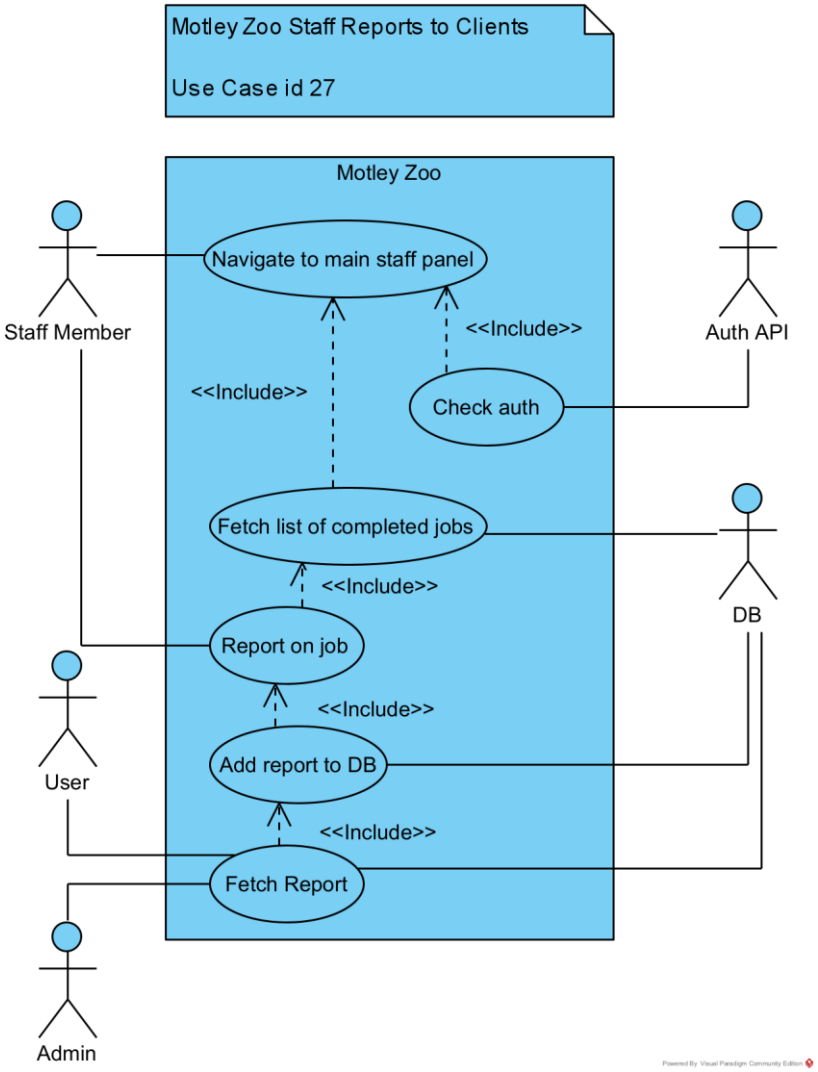

Motley Zoo Staff View Booked Schedule
Use Case Id 22
Motley Zoo
Navigate to schedule section
<<Include>>
Check auth
Staff Member
Auth API
Scheduler
Narrow down date range
DB
<<Include>>
Load Schedule of date range

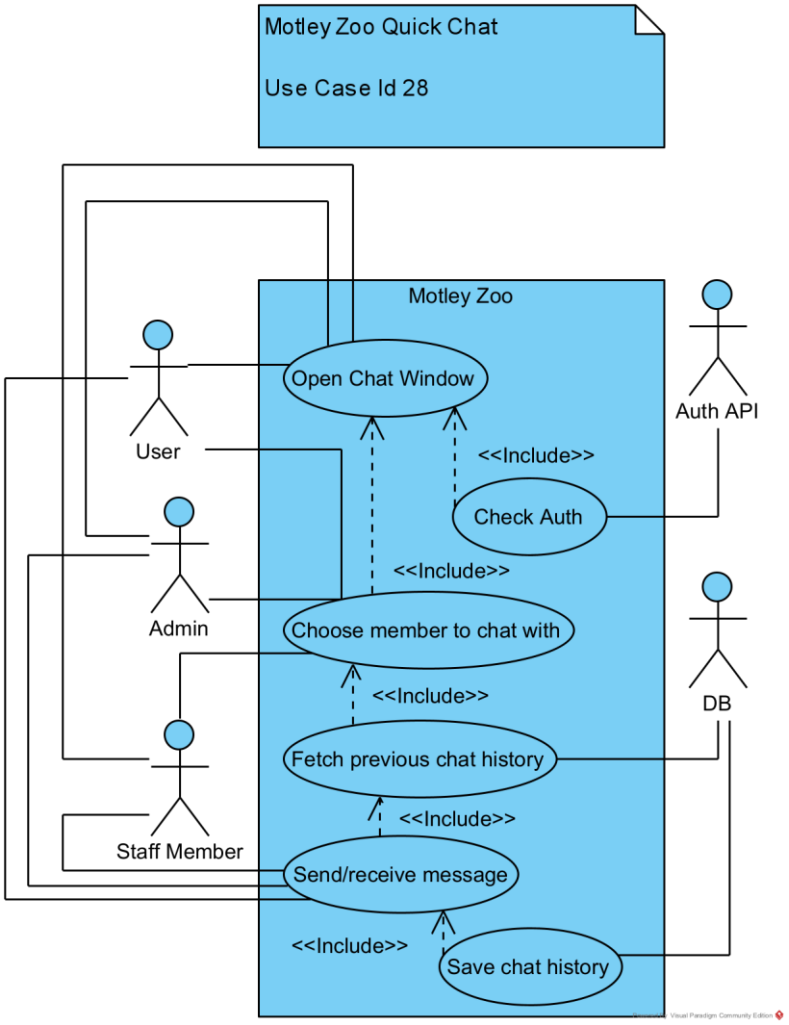

Motley Zoo Staff Set Schedule of availability
Use Case id 23
Motley Zoo
Navigate to scheduler section
<<Include>>
Check auth
Staff Member
Auth API
<<Include>>
Scheduler
Set Availability Schedule
<<Include>>
Update schedule
DB


Motley Zoo Staff Set Dog Handling Limit
Use case id 24
Motley Zoo
Staff Member
Navigate to profile page
<<Include>>
<<Include>>
Check auth
Auth API
Edit workload preferences
<<Include>>
Update preference in DB
DB


Motley Zoo Staff See Hours Worked And Paid Out
Use Case id 25
Motley Zoo
Staff Member
Navigate to Scheduler section
<<Include>>
Check auth
Auth API
<<Include>>
Scheduler
Enable hours worked/paid mode
<<Include>>
Set date range to check
<<Include>>
Fetch filtered range from DB
DB
<<Include>>
Load filtered View

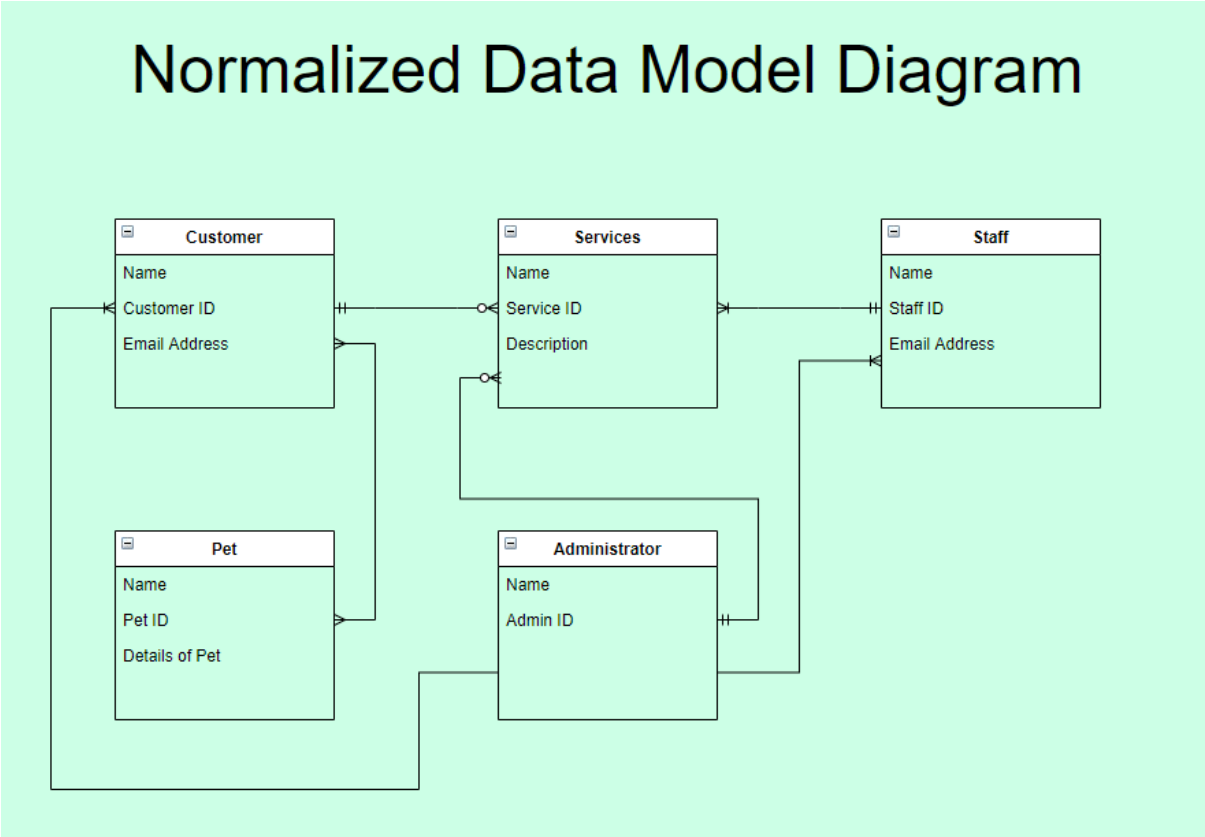

Motley Zoo Staff Reports to Admin
Use Case id 26
Motley Zoo
Staff Member
Navigate to main staff panel
<<Include>>
Auth API
<<Include>>
Check auth
Fetch list of completed jobs
DB
<<Include>>
Report on job
<<Include>>
Add report to DB
<<Include>>
Admin
Fetch Report


Motley Zoo Staff Reports to Clients
Use Case id 27
Motley Zoo
Staff Member
Navigate to main staff panel
<<Include>>
Auth API
<<Include>>
Check auth
Fetch list of completed jobs
DB
<<Include>>
Report on job
User
<<Include>>
Add report to DB
<<Include>>
Fetch Report
Admin

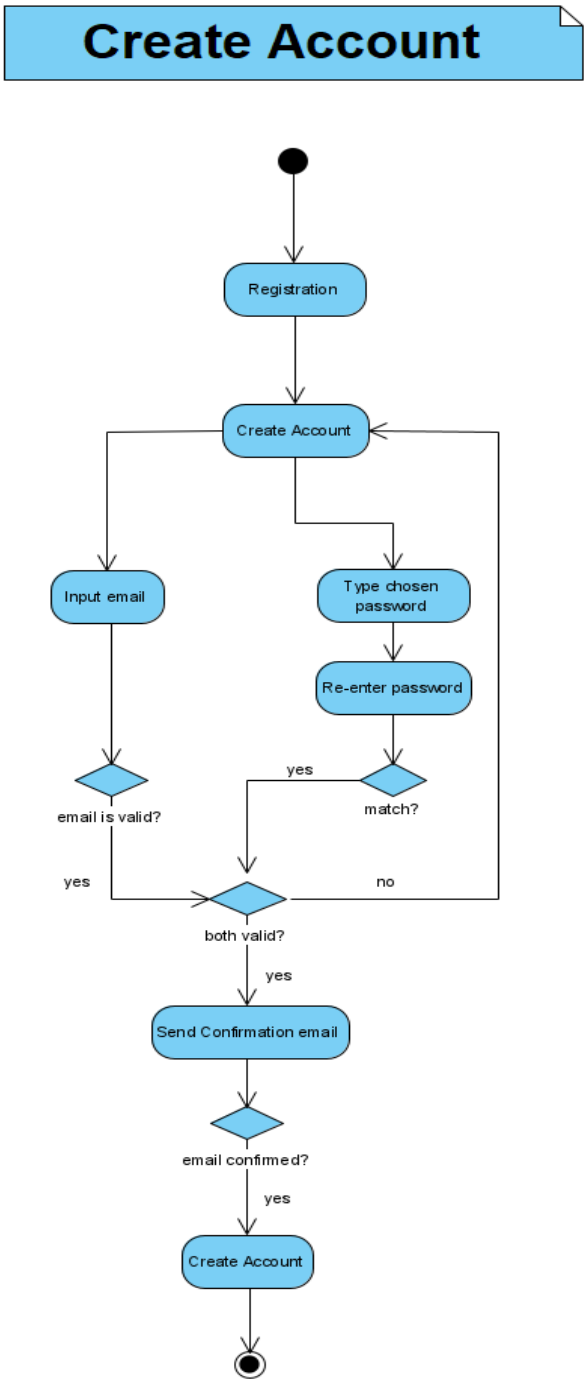

Motley Zoo Quick Chat
Use Case Id 28
Motley Zoo
User
Admin
Staff Member
Open Chat Window
Check Auth
Choose member to chat with
Fetch previous chat history
Send/receive message
Save chat history
<<Include>>
<<Include>>
<<Include>>
<<Include>>
<<Include>>
Auth API
DB

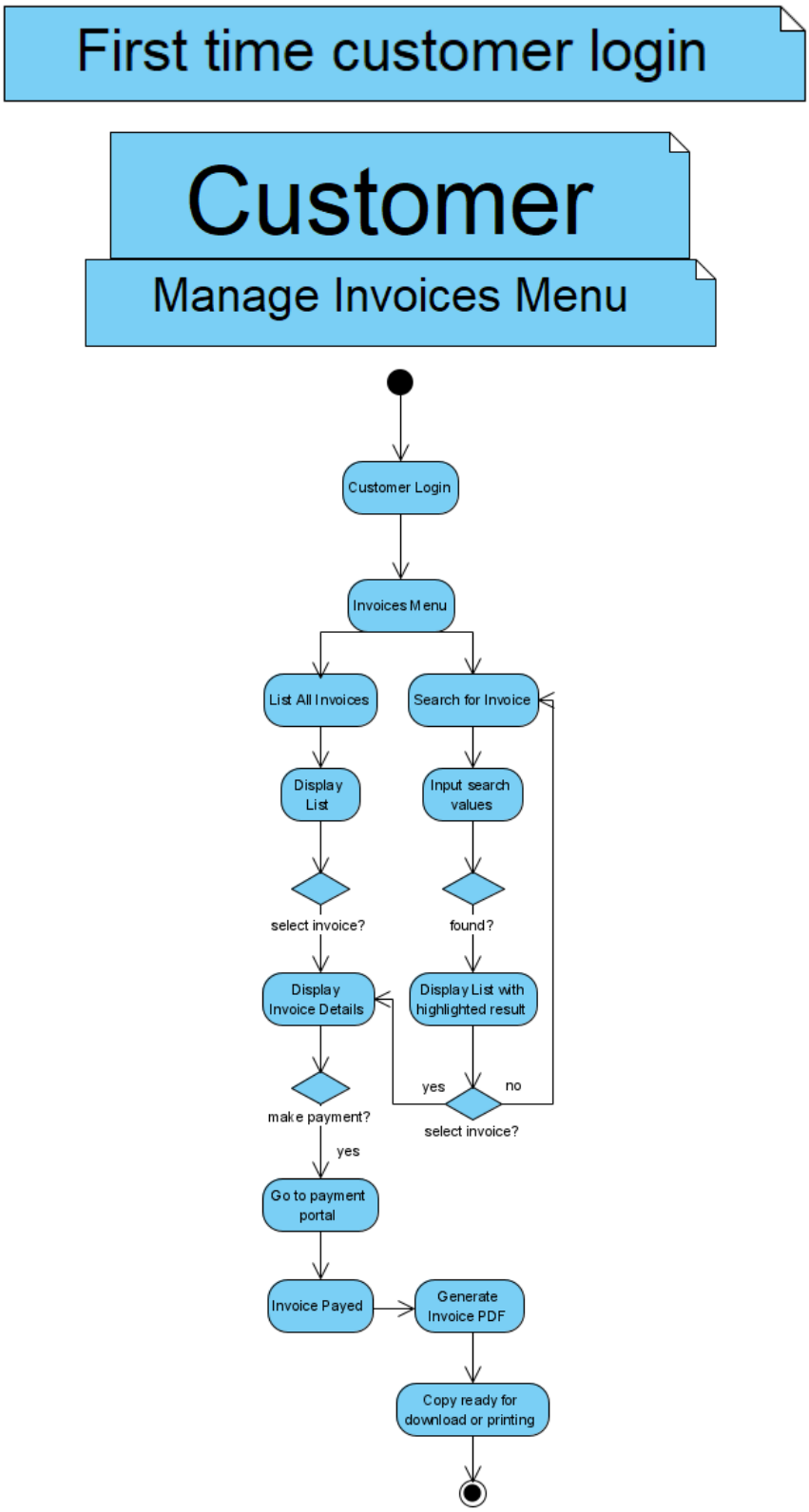
3.3 Data Modelling and Analysis

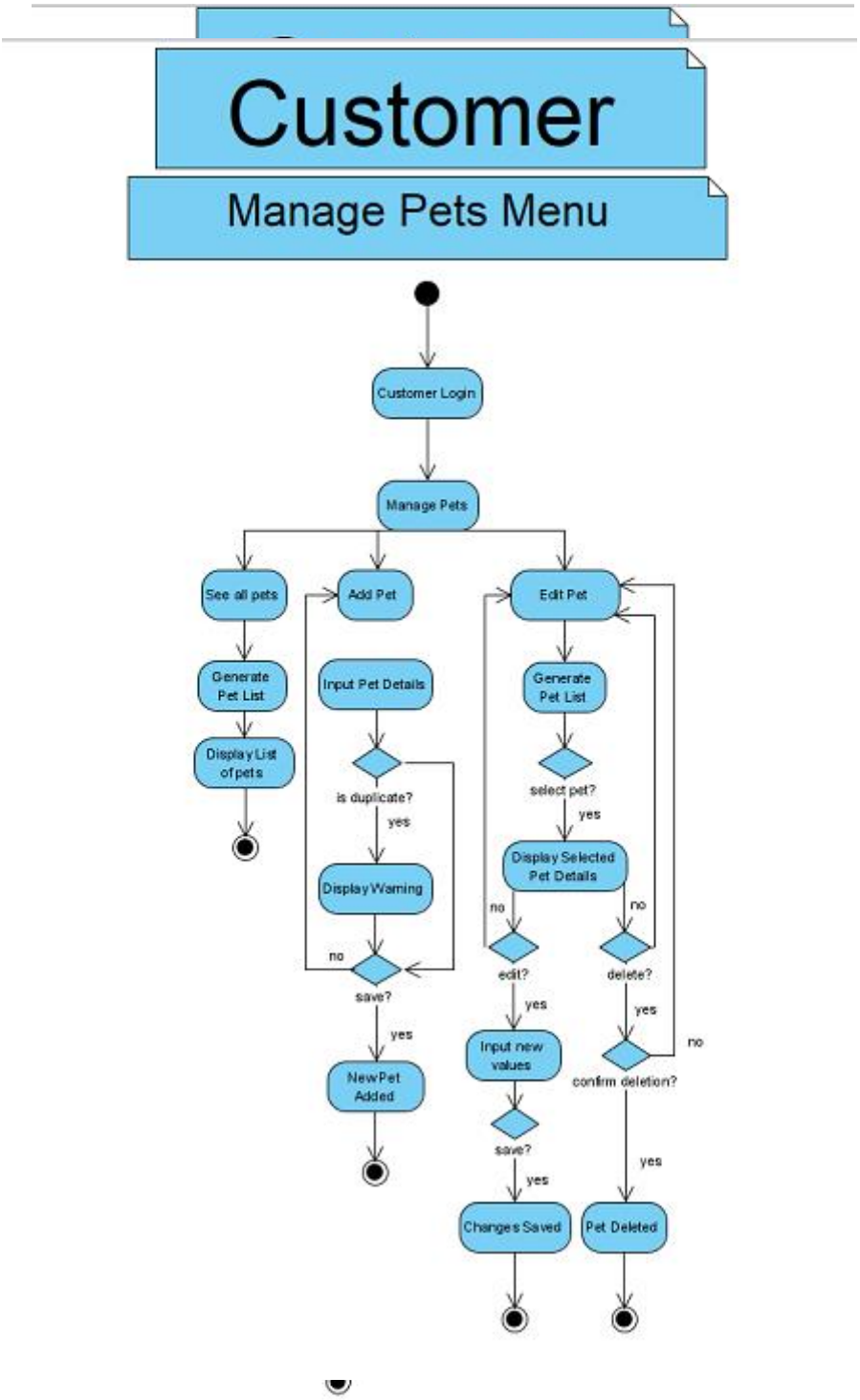
3.3.1 Normalized Data Model Diagram

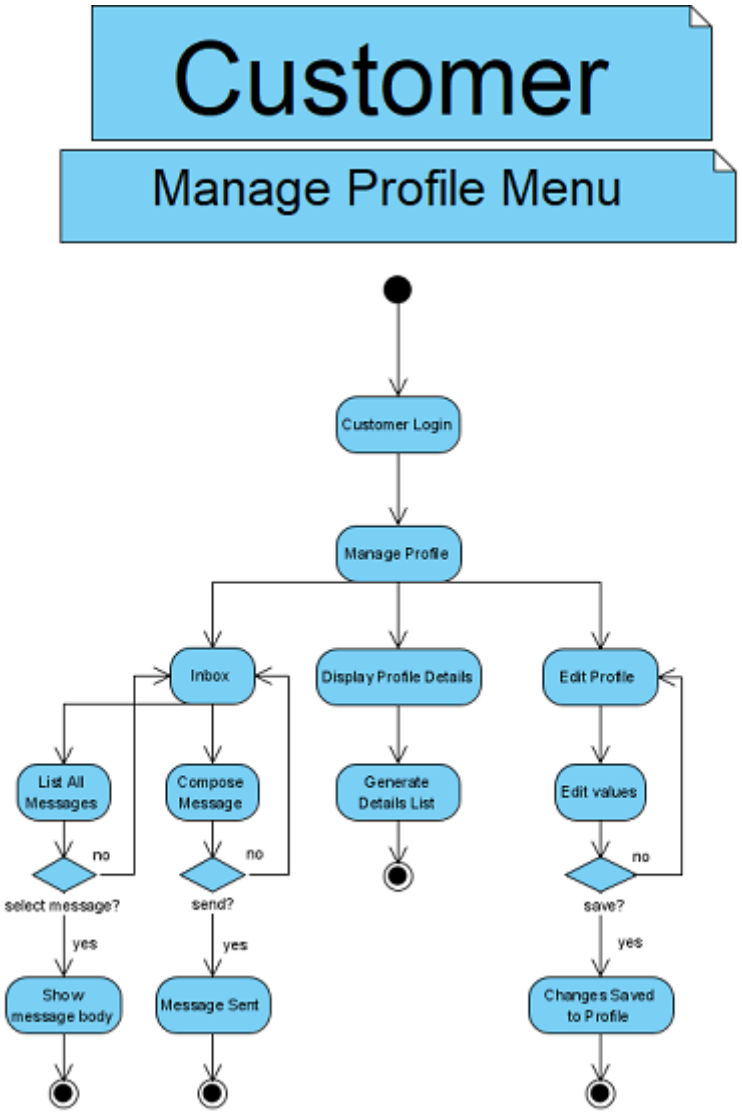


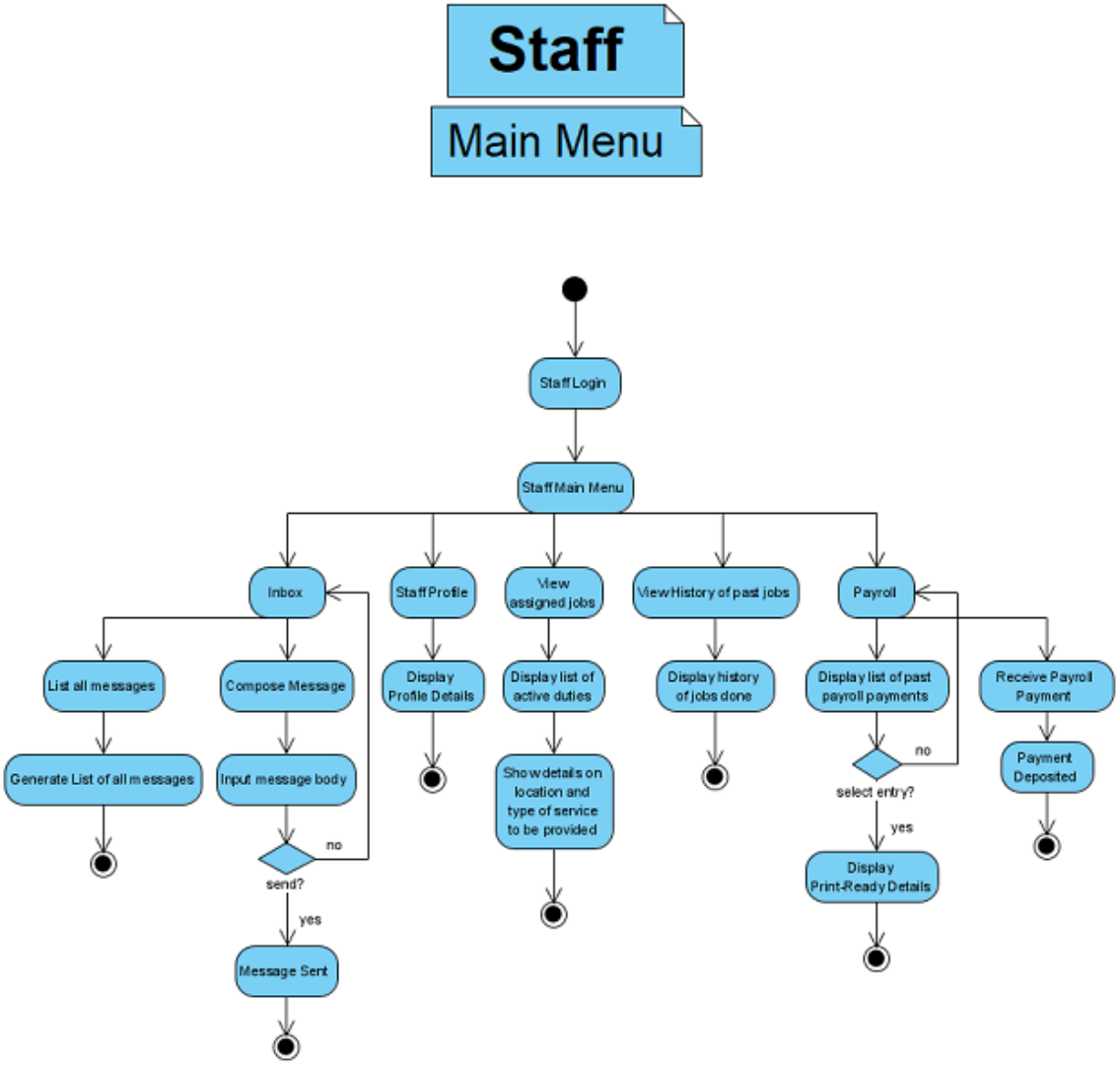
3.3.2 Activity Diagrams

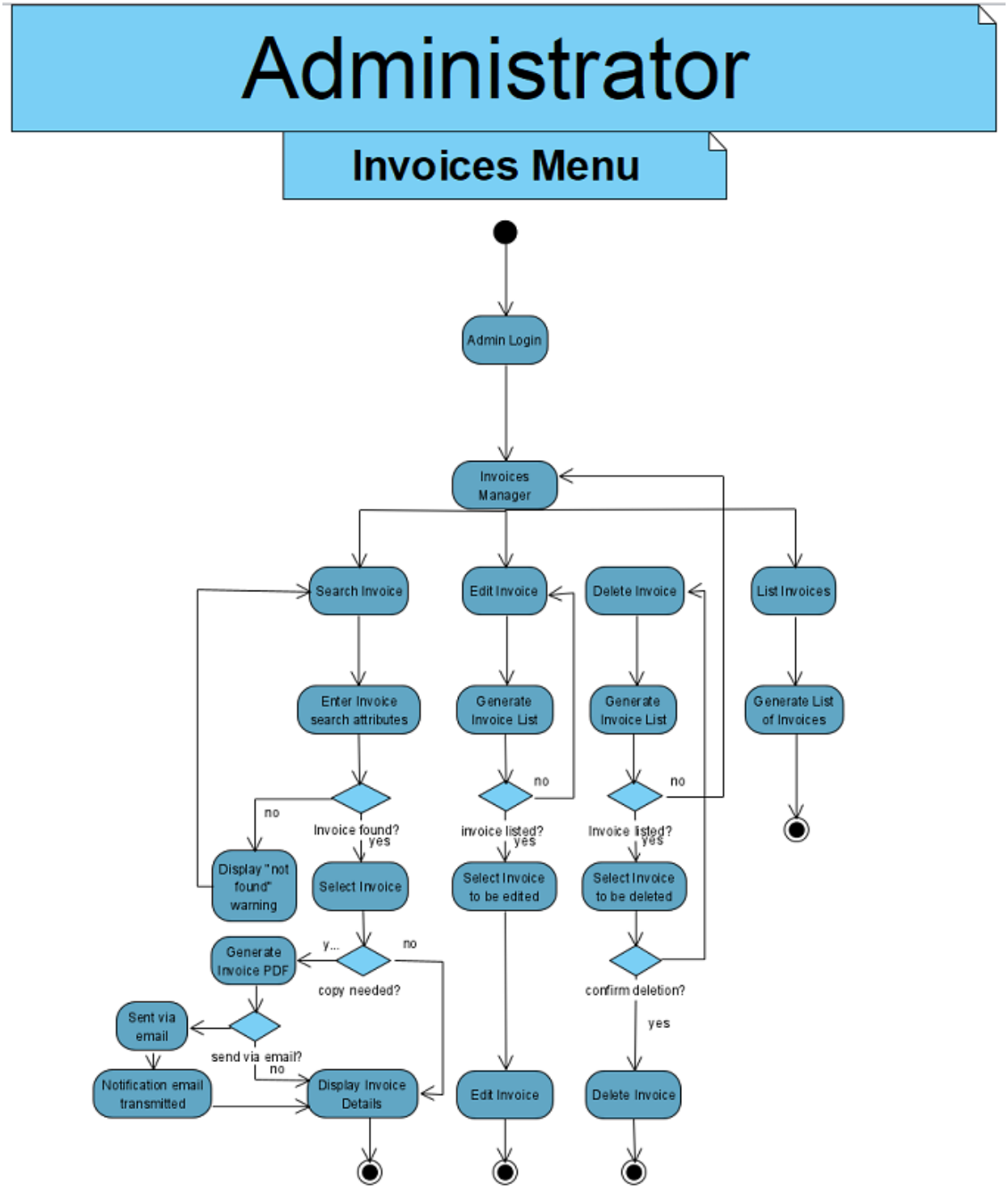


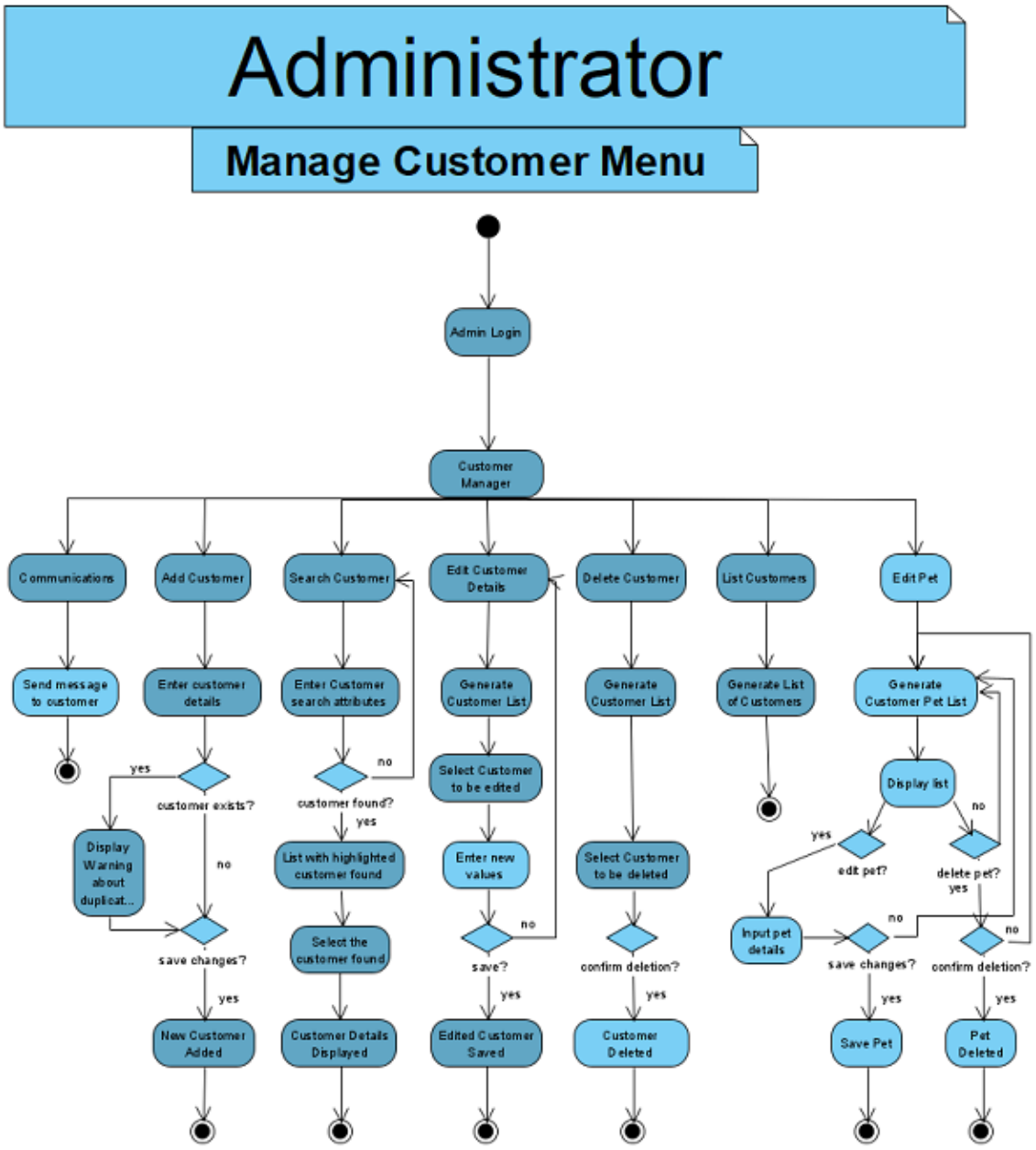

First time customer login
Customer
Manage Invoices Menu
Customer Login
Invoices Menu
List All Invoices
Search for Invoice
Display List
Input search values
select invoice?
found?
Display Invoice Details
Display List with highlighted result
make payment?
yes
no
select invoice?
yes
Go to payment portal
Invoice Payed
Generate Invoice PDF
Copy ready for download or printing

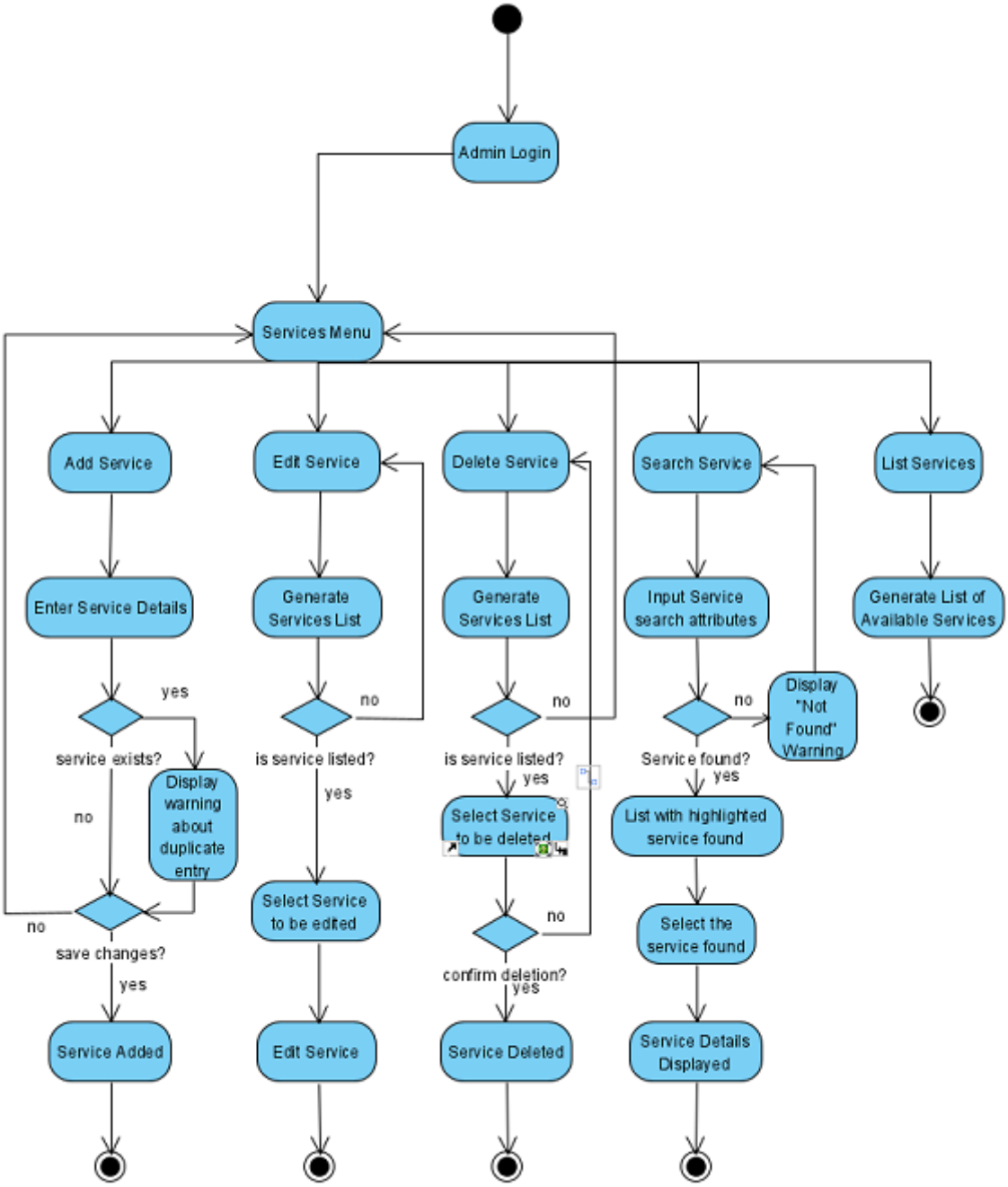

Customer
Manage Pets Menu
Customer Login
Manage Pets
See all pets
Add Pet
Edit Pet
Generate Pet List
Display List of pets
Input Pet Details
is duplicate?
yes
Display Warning
no
save?
yes
New Pet Added
Generate Pet List
select pet?
yes
Display Selected Pet Details
no
edit?
yes
Input new values
save?
yes
Changes Saved
no
delete?
yes
confirm deletion?
yes
Pet Deleted
no

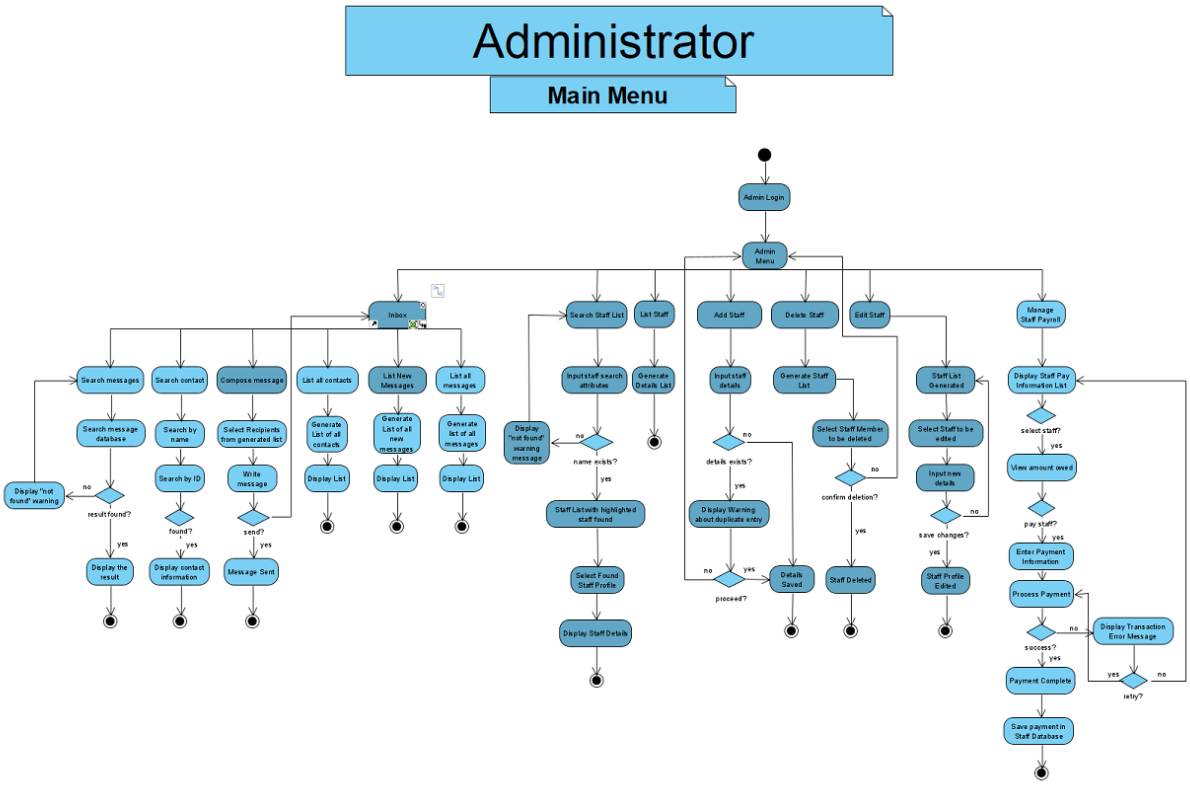

Customer
Manage Profile Menu
Customer Login
Manage Profile
Inbox
Display Profile Details
Edit Profile
List All Messages
Compose Message
Generate Details List
Edit values
no
no
no
select message?
send?
save?
yes
yes
yes
Show message body
Message Sent
Changes Saved to Profile


Staff
Main Menu
Staff Login
Staff Main Menu
Inbox
Staff Profile
View assigned jobs
View History of past jobs
Payroll
List all messages
Compose Message
Display Profile Details
Display list of active duties
Display history of jobs done
Display list of past payroll payments
Receive Payroll Payment
Generate List of all messages
Input message body
Show details on location and type of service to be provided
no
send?
yes
Message Sent
no
select entry?
yes
Display Print-Ready Details
Payment Deposited

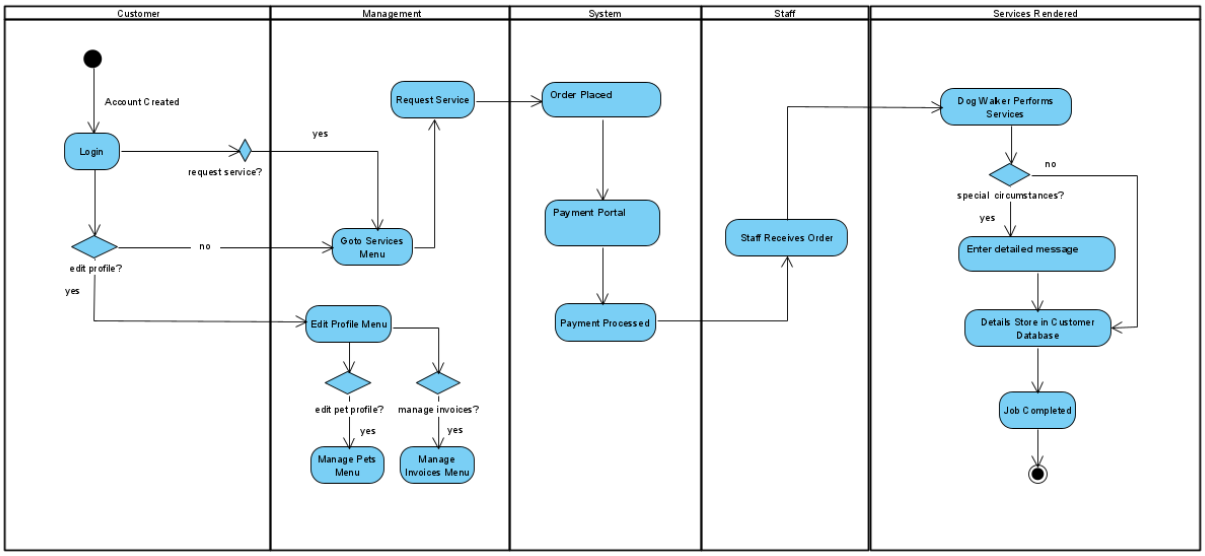

Administrator
Invoices Menu
Admin Login
Invoices Manager
Search Invoice
Edit Invoice
Delete Invoice
List Invoices
Enter Invoice search attributes
Generate Invoice List
Generate Invoice List
Generate List of Invoices
no
Invoice found?
yes
no
invoice listed?
yes
no
Invoice listed?
yes
Display "not found" warning
Select Invoice
Select Invoice to be edited
Select Invoice to be deleted
Generate Invoice PDF
y...
no
copy needed?
confirm deletion?
yes
Sent via email
send via email?
no
Notification email transmitted
Display Invoice Details
Edit Invoice
Delete Invoice


Administrator
Manage Customer Menu
Admin Login
Customer Manager
Communications
Add Customer
Search Customer
Edit Customer Details
Delete Customer
List Customers
Edit Pet
Send message to customer
Enter customer details
Enter Customer search attributes
Generate Customer List
Generate Customer List
Generate List of Customers
Generate Customer Pet List
yes
customer exists?
no
Display Warning about duplicat...
save changes?
yes
New Customer Added
no
customer found?
yes
List with highlighted customer found
Select the customer found
Customer Details Displayed
Select Customer to be edited
Enter new values
no
save?
yes
Edited Customer Saved
Select Customer to be deleted
confirm deletion?
yes
Customer Deleted
Display list
yes
edit pet?
no
delete pet?
yes
Input pet details
no
save changes?
yes
Save Pet
no
confirm deletion?
yes
Pet Deleted

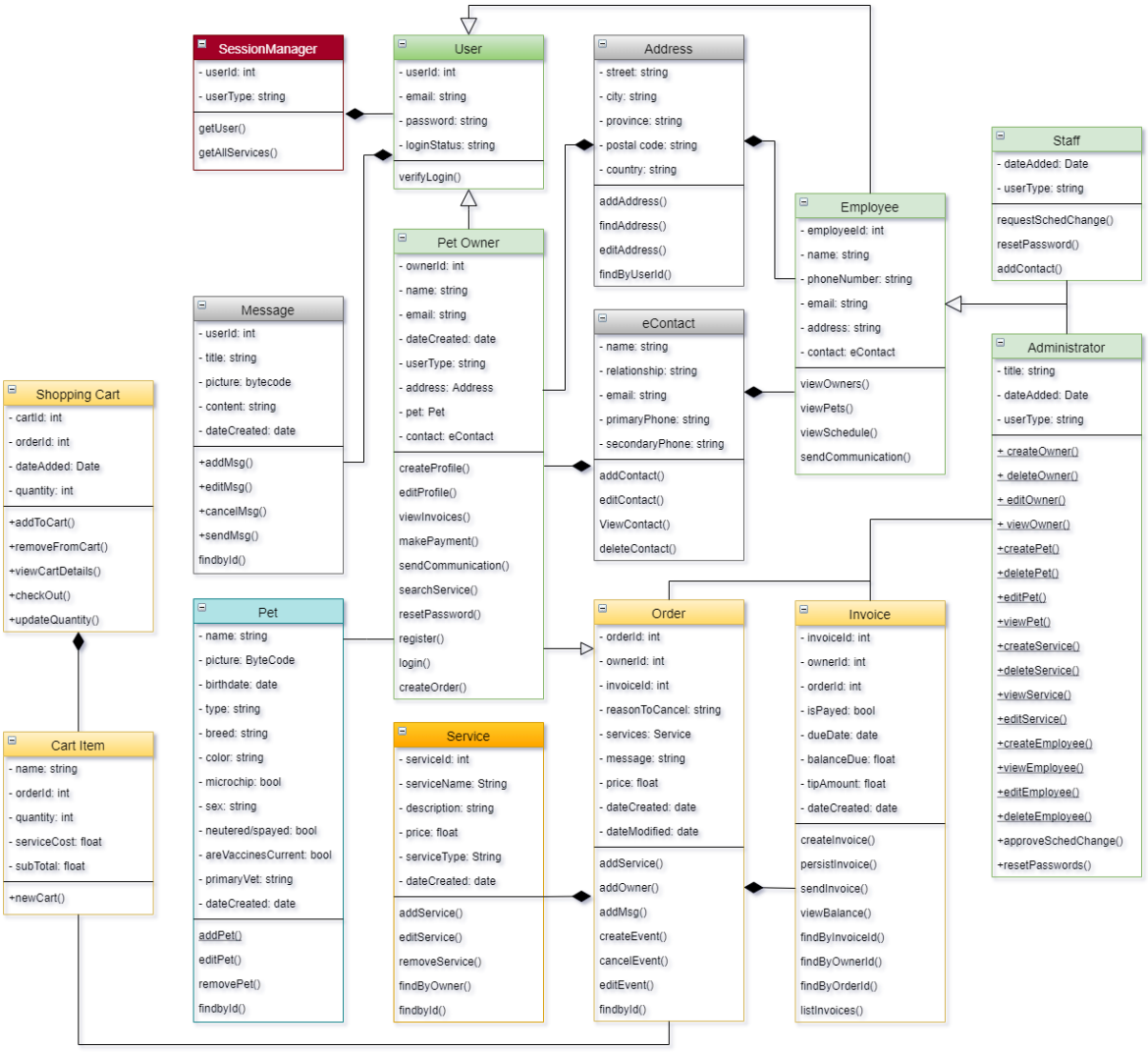

Admin Login
Services Menu
Add Service
Edit Service
Delete Service
Search Service
List Services
Enter Service Details
Generate Services List
Generate Services List
Input Service search attributes
Generate List of Available Services
yes
service exists?
no
Display warning about duplicate entry
no
save changes?
yes
Service Added
no
is service listed?
yes
Select Service to be edited
Edit Service
no
is service listed?
yes
Select Service to be deleted
no
confirm deletion?
yes
Service Deleted
no
Service found?
yes
Display "Not Found" Warning
List with highlighted service found
Select the service found
Service Details Displayed


Administrator
Main Menu
Admin Login
Admin Menu
Inbox
Search messages
Search contact
Compose message
List all contacts
List New Messages
List all messages
Search Staff List
List Staff
Add Staff
Delete Staff
Edit Staff
Manage Staff Payroll
Search message database
Search by name
Select Recipients from generated list
Generate List of all contacts
Generate List of all new messages
Generate list of all messages
Input staff search attributes
Generate Details List
Input staff details
Generate Staff List
Staff List Generated
Display Staff Pay Information List
Display "not found" warning
result found?
Search by ID
Write message
Display List
Display List
Display List
Display "not found" warning message
name exists?
details exists?
Select Staff Member to be deleted
Select Staff to be edited
select staff?
Display the result
found?
Display contact information
send?
Message Sent
Staff List with highlighted staff found
Display Warning about duplicate entry
confirm deletion?
Input new details
View amount owed
save changes?
pay staff?
Select Found Staff Profile
Display Staff Details
proceed?
Details Saved
Staff Deleted
Staff Profile Edited
Enter Payment Information
Process Payment
Display Transaction Error Message
success?
retry?
Payment Complete
Save payment in Staff Database

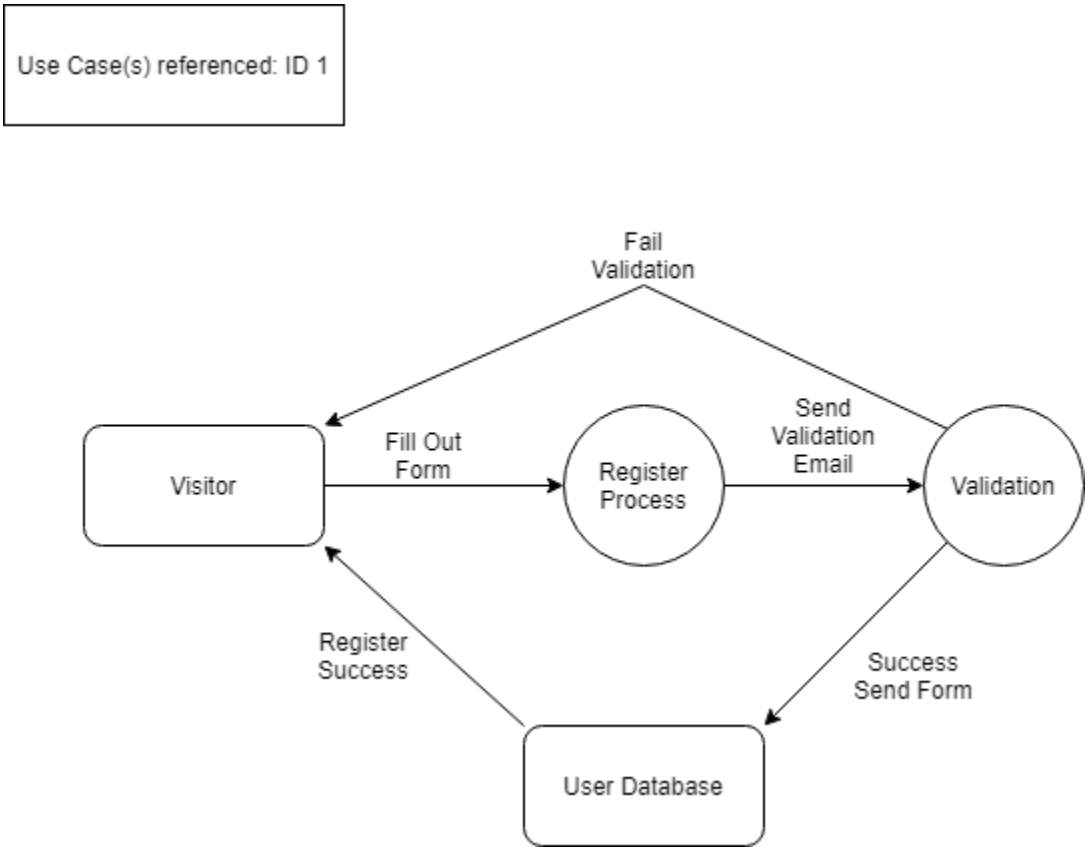
3.3.3 Sequence Diagrams



3.3.4 UML Class Diagram

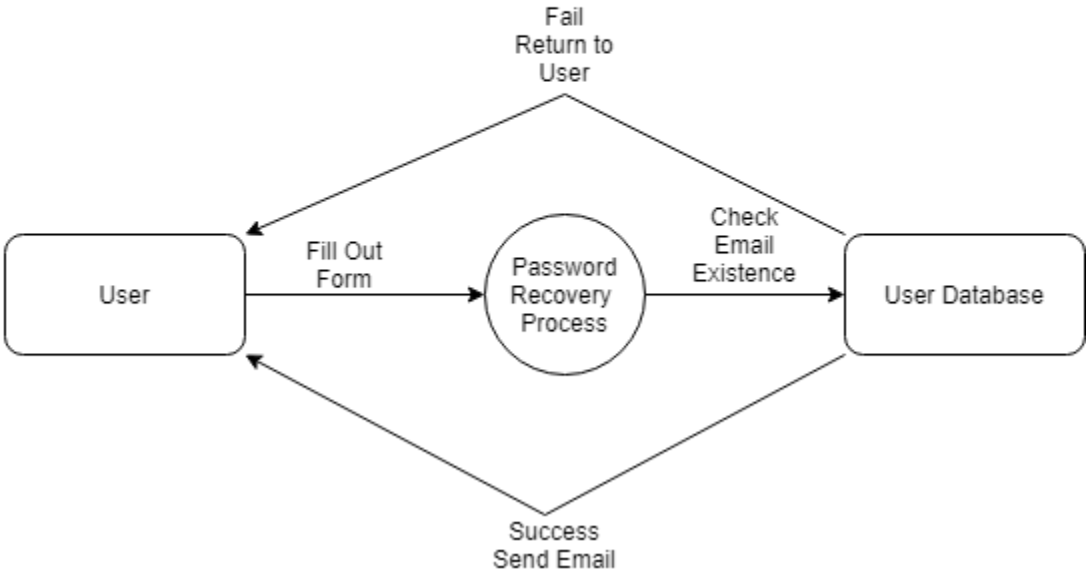


3.4 Process Modelling - Data Flow Diagrams



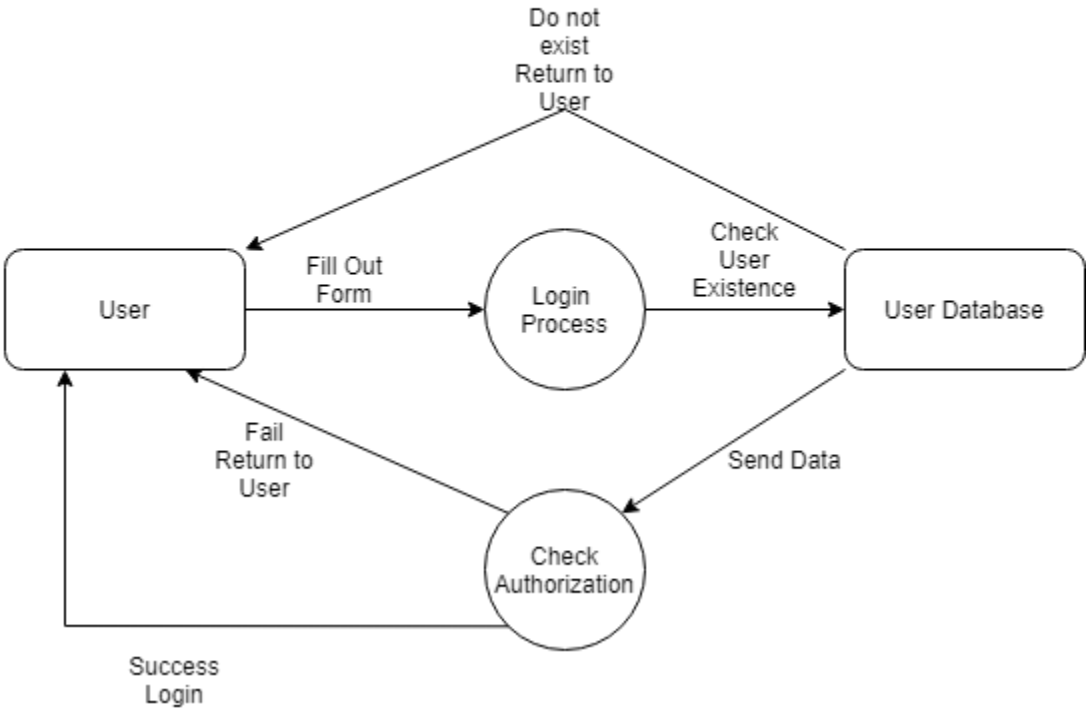
1 Register User Account

Use Case(s) referenced: ID 2



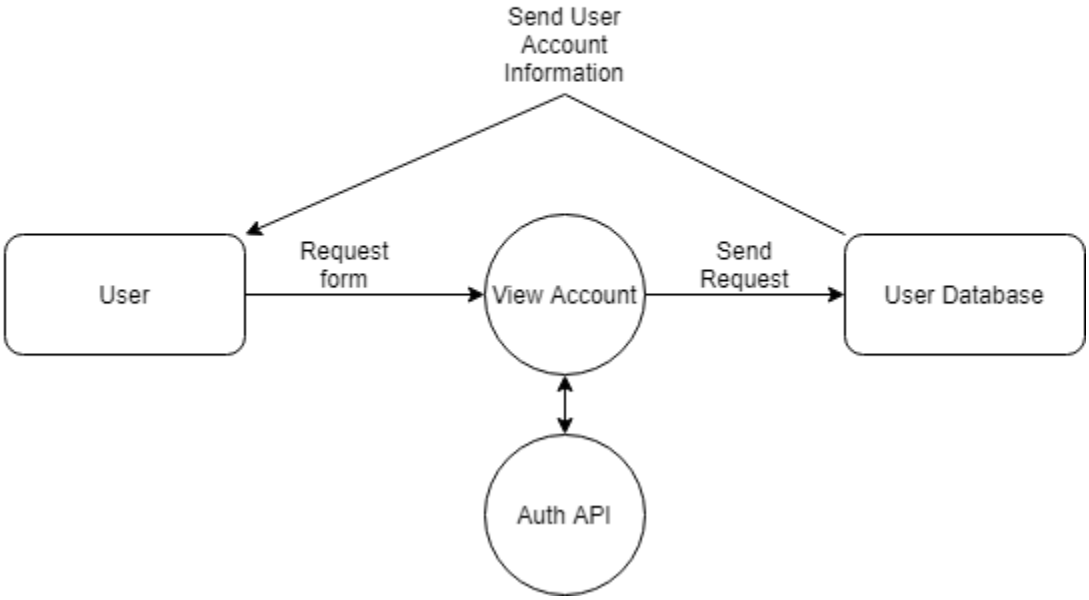
2 Reset User Password

Use Case(s) referenced: ID 3



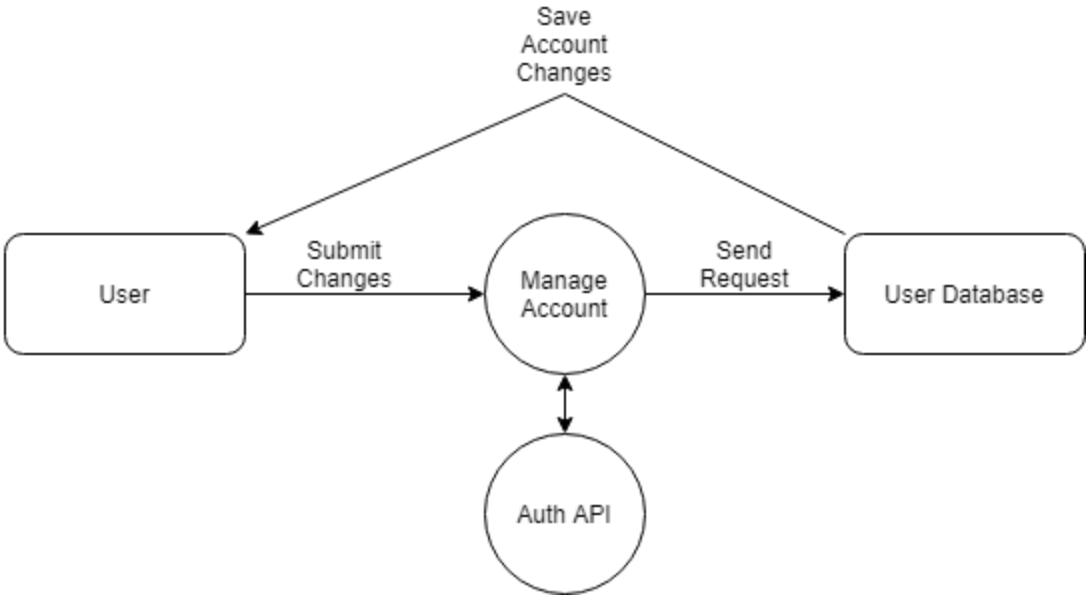
3 Login User Account

Use Case(s) referenced: ID 4



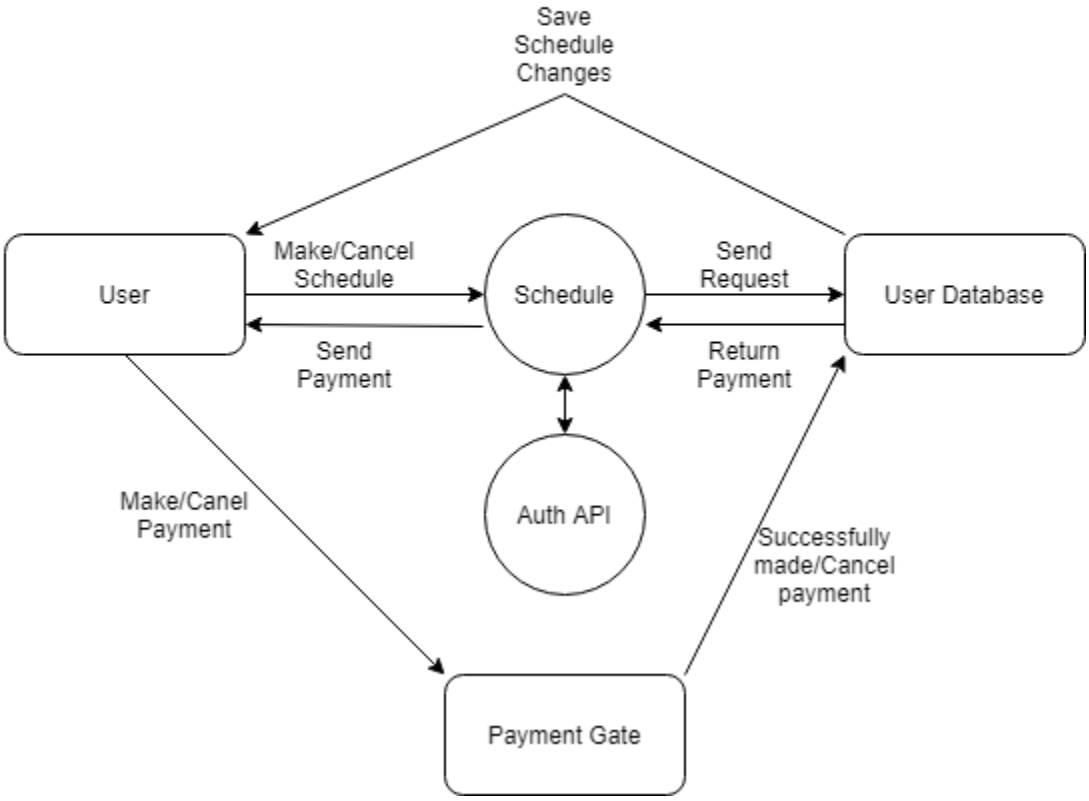
4 View User Account

Use Case(s) referenced: ID 5,6



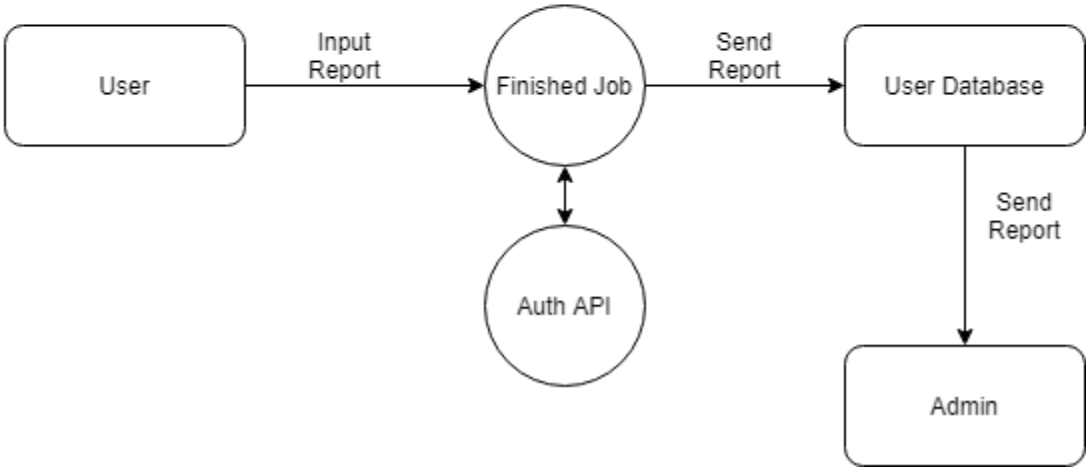
5, 6 Edit User Account, Delete User Account

Use Case(s) referenced: ID 7,8,9



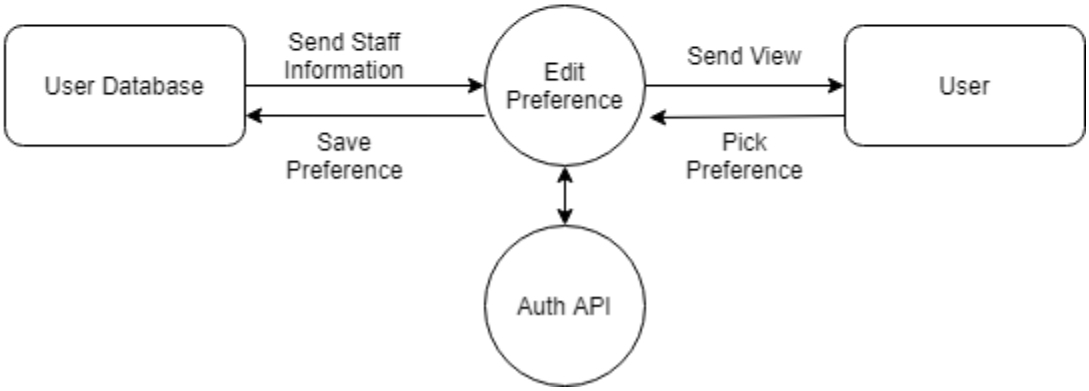
7,8,9 Schedule Single Time Slot, Schedule Daily Time Slot, Cancel Scheduled Appointments

Use Case(s) referenced: ID 10

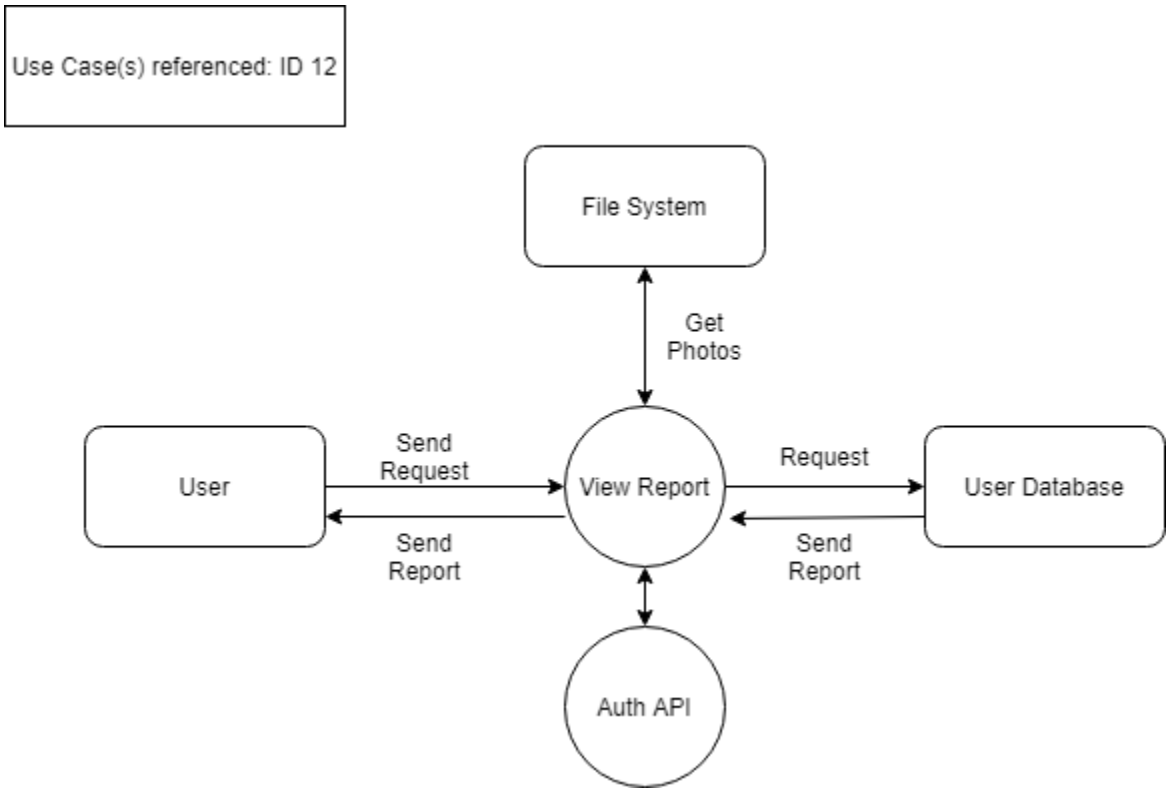


10 User Report on Dog Walk

Use Case(s) referenced: ID 11

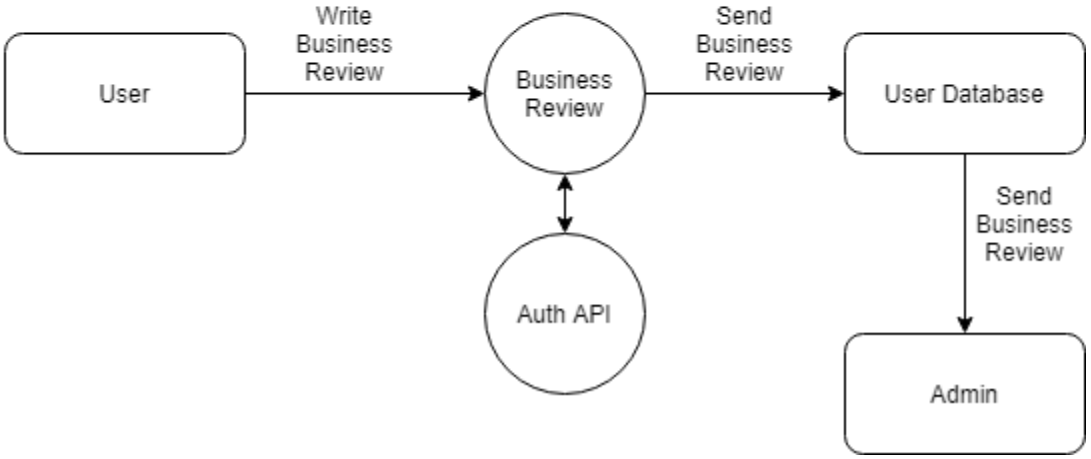


11 Setting Desired Dog Walker



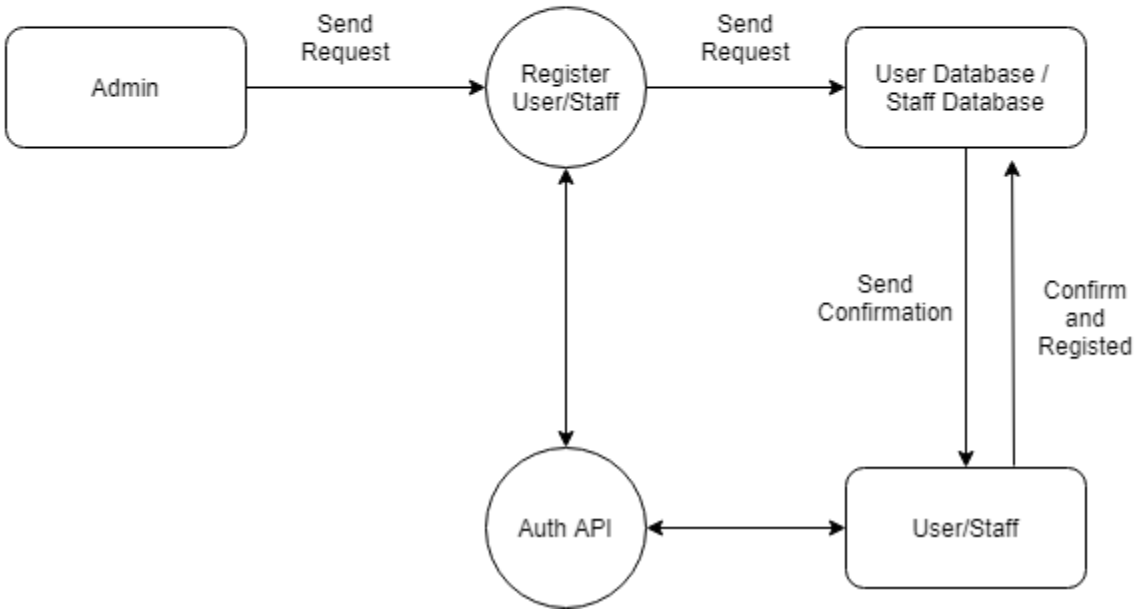
12 User Viewing Reports from Dog Walking

Use Case(s) referenced: ID 13



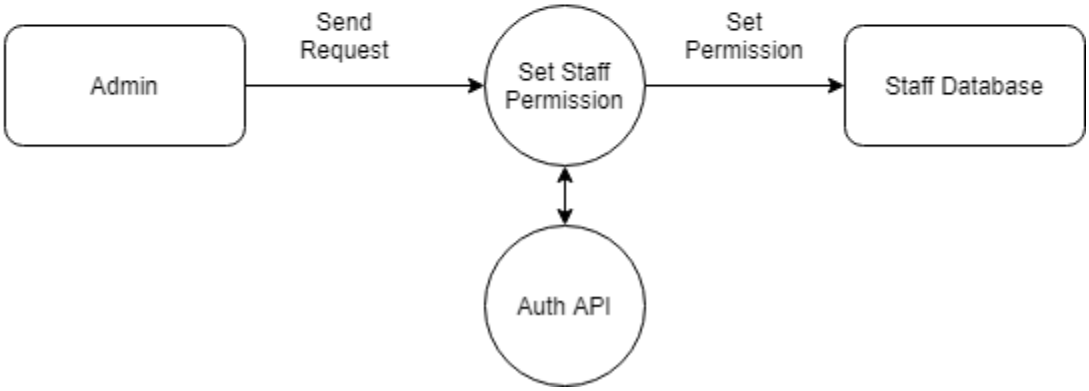
13 User Write Business Review

Use Case(s) referenced: ID 14,15



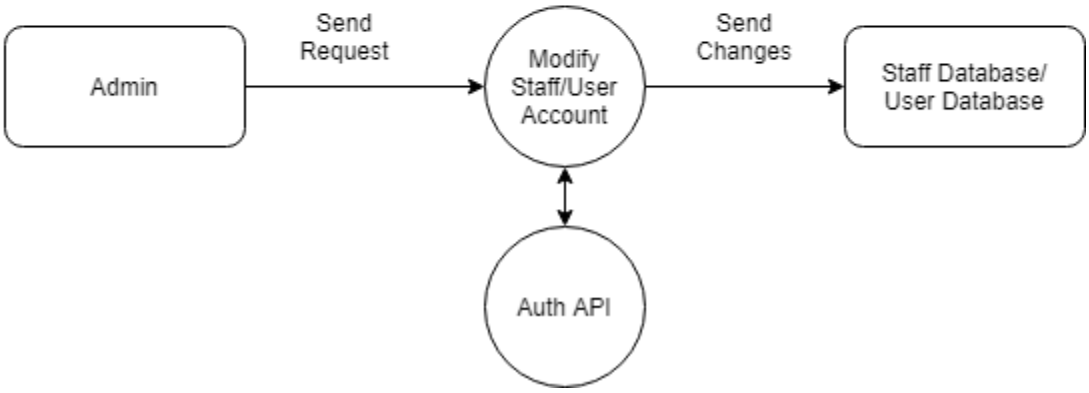
14, 15 Admin Quick Register User Accounts, Admin Register Staff Accounts

Use Case(s) referenced: ID 16



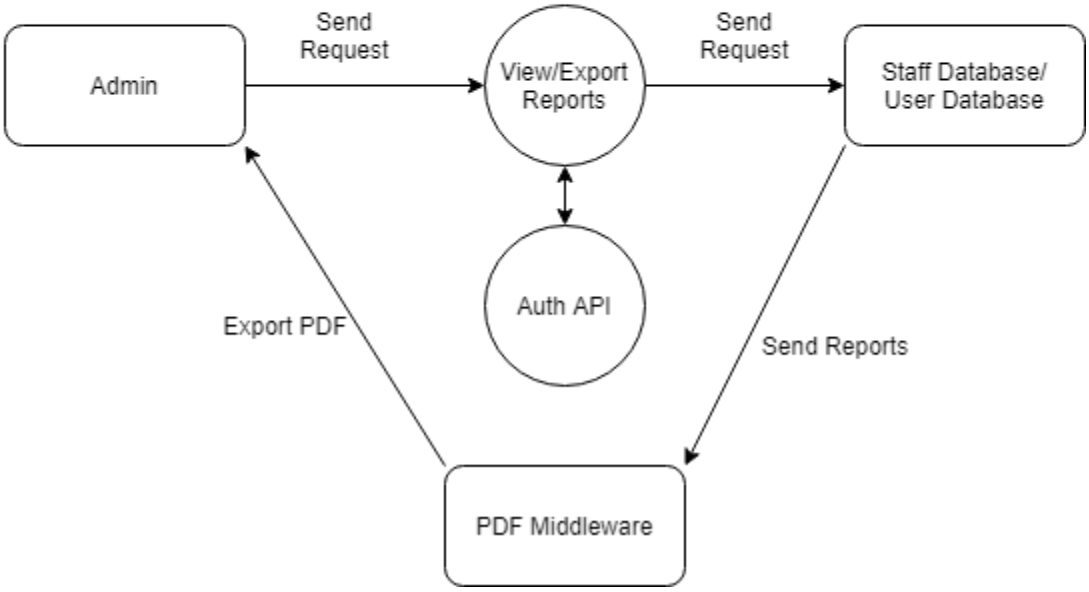
16 Admin Set Permissions

Use Case(s) referenced: ID 17



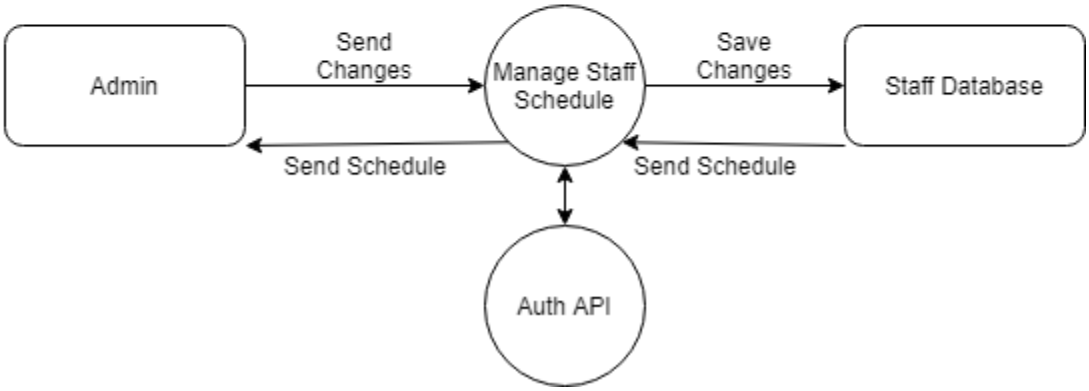
17 Admin Modify Accounts

Use Case(s) referenced: ID 18



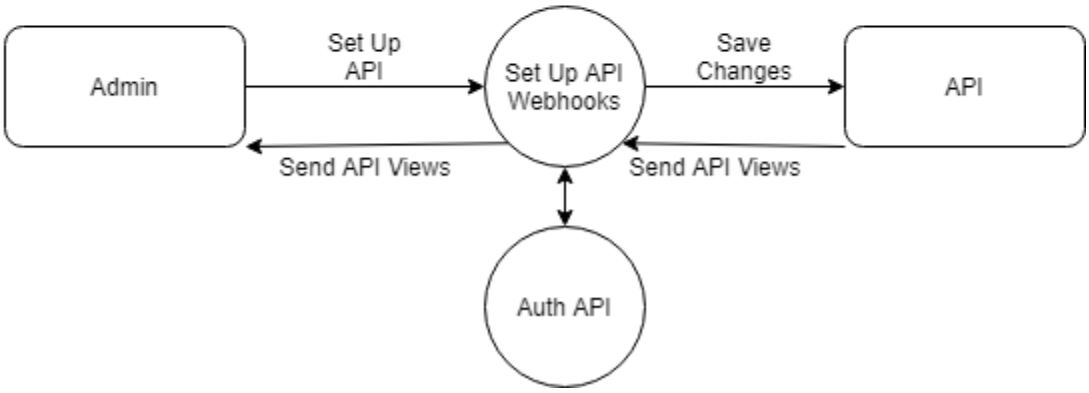
18 Admin View & Export Reports

Use Case(s) referenced: ID 19,20

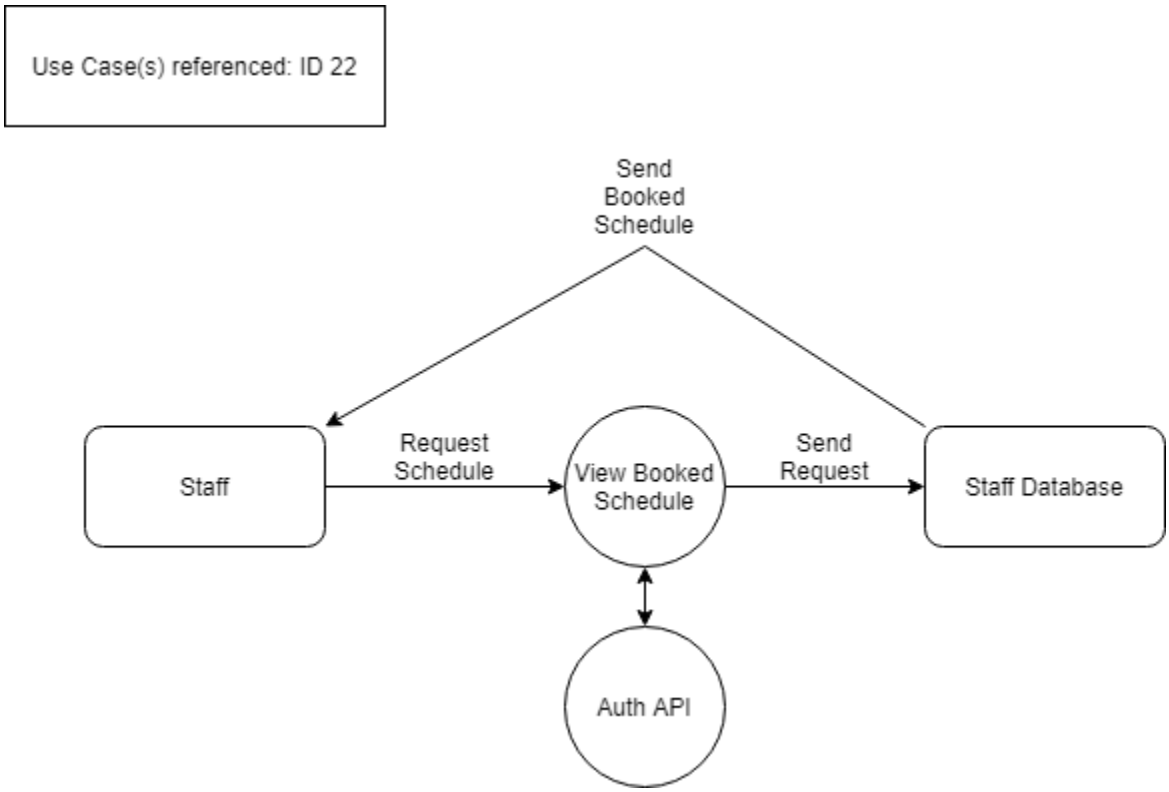


19, 20 Admin Schedule Request Approval, Admin Override Staff Scheduling

Use Case(s) referenced: ID 21

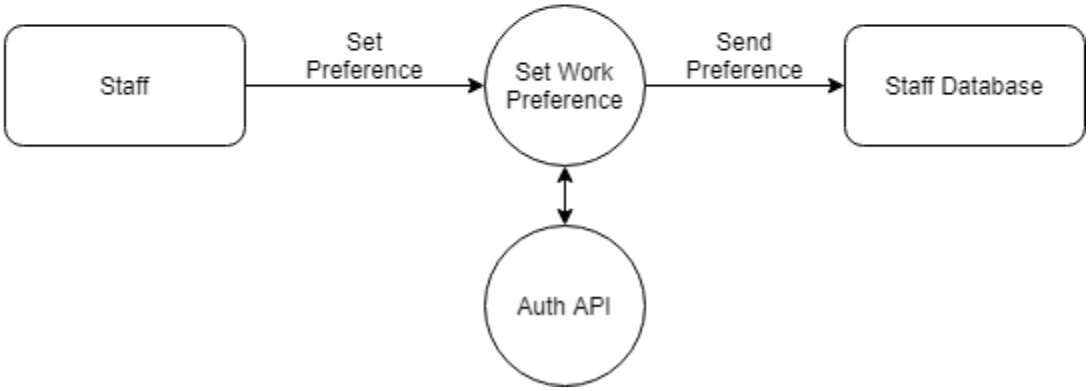


21 Admin Setup API Webhooks

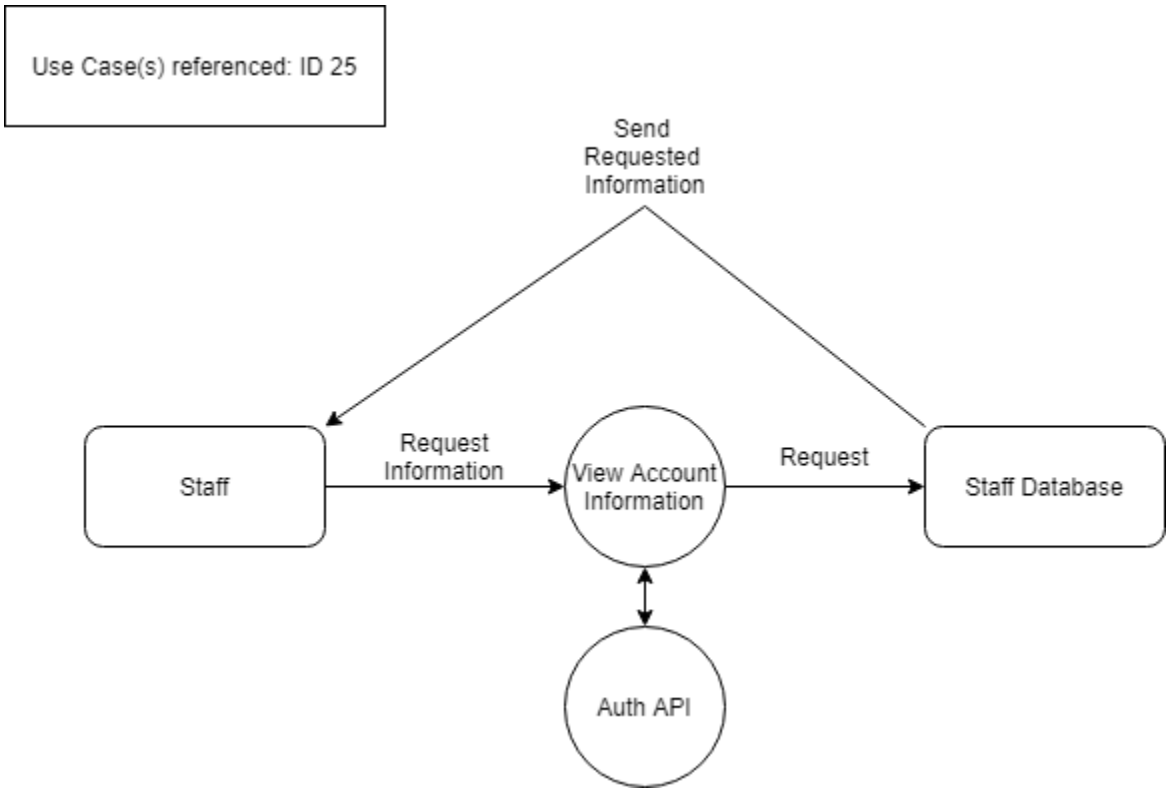


22 Staff Member View Booked Schedule

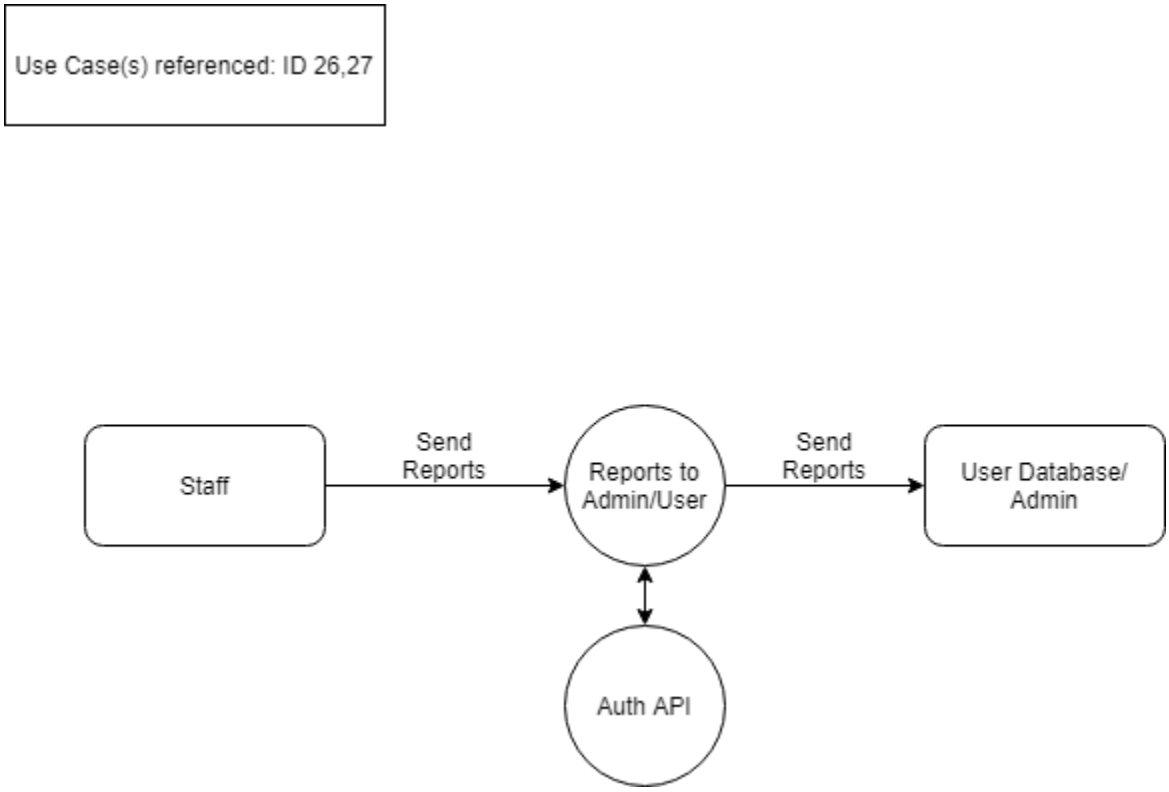
Use Case(s) referenced: ID 23,24



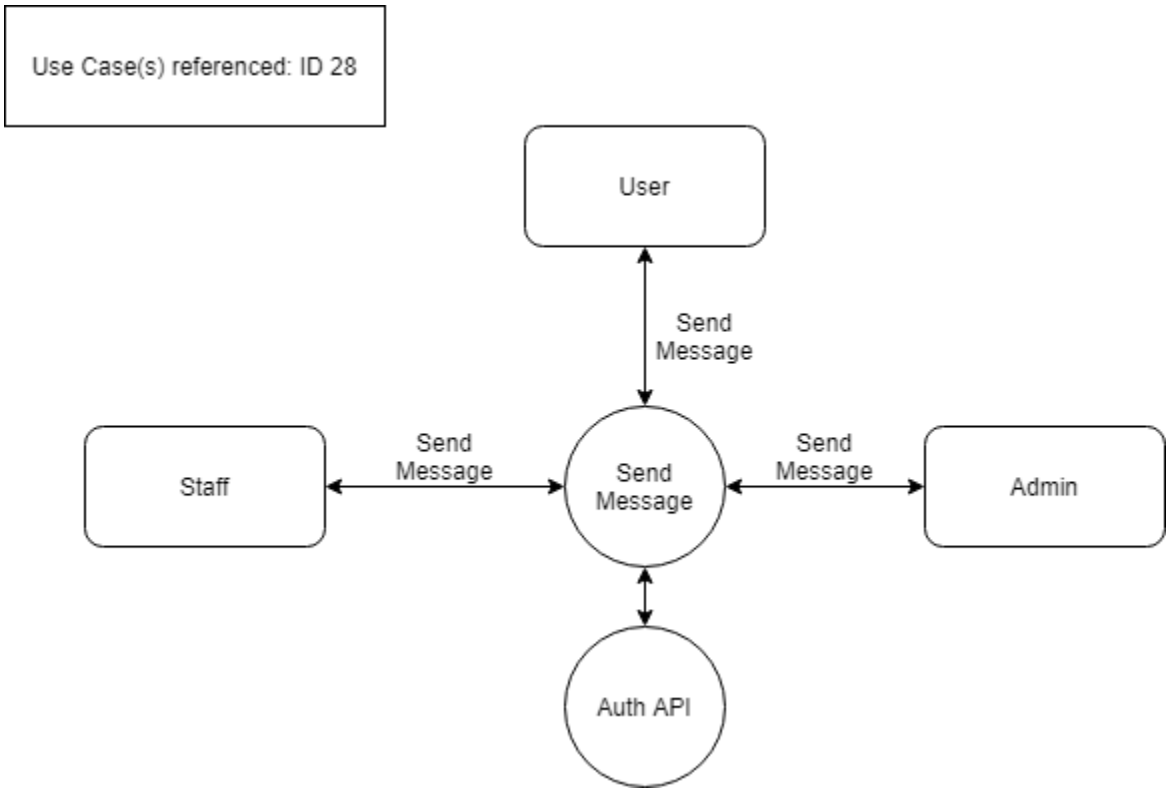
23, 24 Staff Set Schedule of Availability, Staff Set Dog Handling Limit



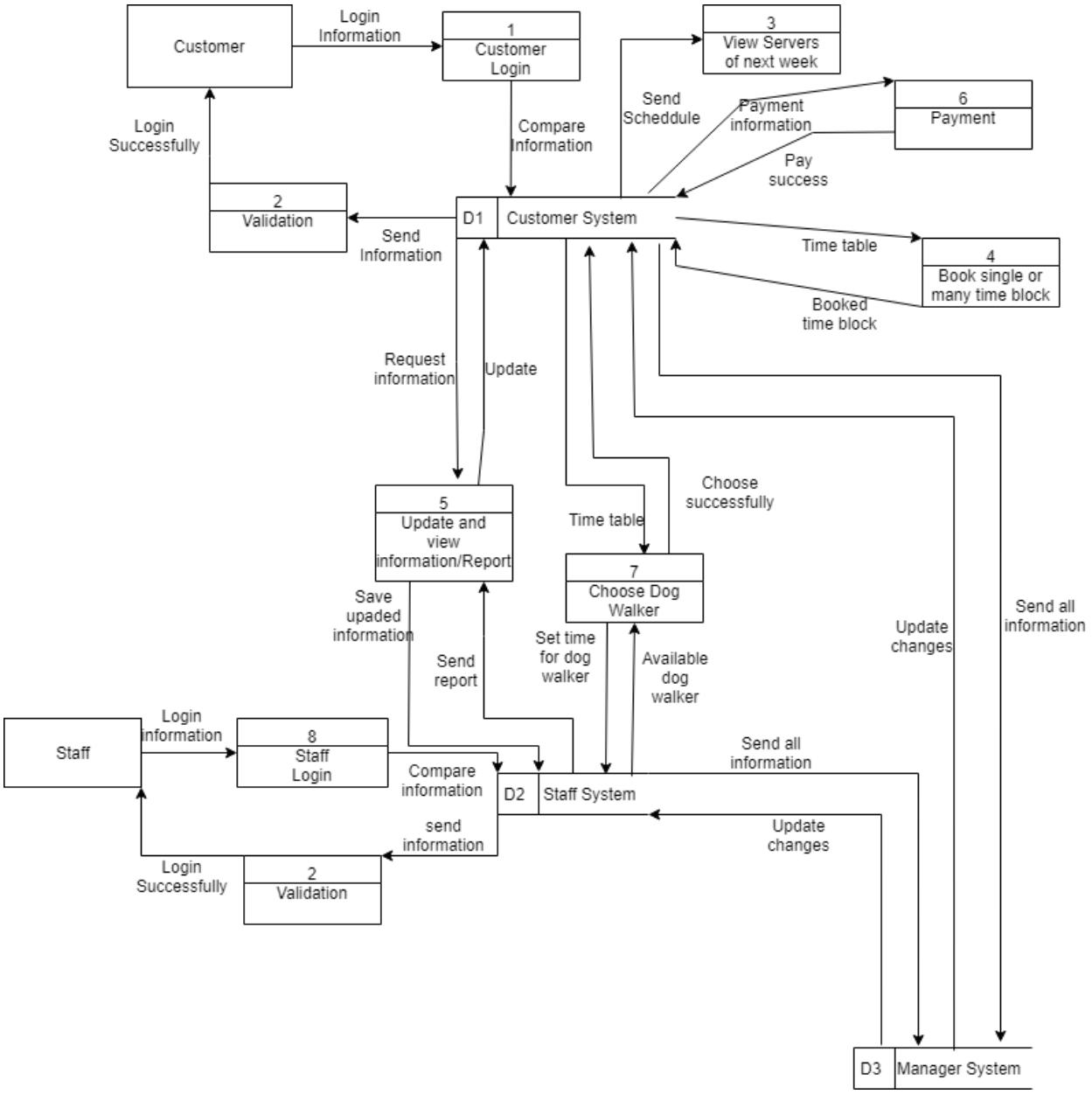
25 Staff See Hours Worked and Paid Off



26, 27 Staff Reports to Admin, Staff Reports to Client



28 Staff Quick Chat


Customer
Login Information
1
Customer Login
3
View Servers of next week
Send Scheddule
Payment information
6
Payment
Pay success
Login Successfully
Compare Information
2
Validation
D1
Customer System
Send Information
Time table
4
Book single or many time block
Booked time block
Request information
Update
5
Update and view information/Report
Time table
Choose successfully
7
Choose Dog Walker
Save upaded information
Send report
Set time for dog walker
Available dog walker
Update changes
Send all information
Staff
Login information
8
Staff Login
Compare information
send information
D2
Staff System
Send all information
Update changes
Login Successfully
2
Validation
D3
Manager System

4.0 Non-Functional Requirements

1. System downtime may not exceed 1 hour per week for regular maintenance.
2. Customers need to stay at Motley Zoo's official website before and after logging in.
3. The registration process needs to be done within 12 hours after the validation email is sent.
4. Database security must meet HIPAA requirements
5. The system is easy to use by customers, staff, and managers
6. The system is broken into multiple modules for easy maintenance and debug
7. The system will be available on all platforms that can access the website
8. Each page must be loaded within 2 seconds
9. The system must meet the requirements of Web Content Accessibility Guidelines WCAG2.1

5.0 Logical Database Requirements

Further details are available in section 3.3.4 of this document. The following list of requirements will be used below.

Data Formats

- Fields will be mostly stored as strings, except for the following exceptions being encrypted strings:
- Passwords (encrypted)
- Credit Card information (encrypted)
- Images are stored in a filesystem, a string is stored as a link to the image file

Storage Capabilities

- Serverless architecture on AWS
- Scaling storage size (starting at 8gb)
- Scaling CPU/RAM based on workload

Data Retention

- All data is to be retained indefinitely

Data Integrity

Entity Integrity

- Auto incrementing and auto managed primary keys (mongoDB objectId field)

Referential Integrity

- MongoDB is a non-relational database, it doesn't use the concept of foreign keys - Relationships are embedded

6.0 Other Requirements

Requirement	Details
Developers Skills Upgrades	Although the team is getting a firm grasp of full stack development which falls directly in line with the development plans, and the team includes two members with existing professional development experience, there are some aspects that will need to be further studied. The developers will need to have a firm grasp of both the fundamentals of Node.js development as well as React.js development. Aside from that the implementation of JWT authorization/permission system will need to be further studied and mastered by all members of the team by the time development commences. To that extent, the team lead, Vanja, has discussed this with the fullstack class professor Pritesh Patel, who's agreed to help the team out to learn the mechanism before development commences. Aside from this, the team will strive to study all the chosen libraries so that by the time the development starts, there will be no time wasted trying to figure out how a package/library works in context with the entire system.
Github integration with Jira	Once development commences, the team's GitHub will be added to Jira, forming a connection directly between the code and the issues.
Github integration with Travis CI	The GitHub code repos will be continuously integrated, and builds will be tested for success or failure.

Jenkins	The team lead will build out a Jenkins server on AWS, and it will be integrated with the GitHub repos as well as with Jira, enabling another layer of CI/CD.
Training Documentation	Training documentation of the system will be written, shot as development is happening. For admin, this will include a detailed PDF guide. For staff, it will include a PDF guide and perhaps some guide videos. For users this will include an FAQ Modal window, with possible addition of guide videos in the answers sections.

7.0 Approval

The signatures below indicate their approval of the contents of this document.

Project Role	Name	Signature	Date
Principal Stakeholder	Angelica B	Angelica Bailey	2021-11-10
Project Manager	Vanja Vego	Vanja Vego	2021-11-10
Instructor	Anjana Shah		--