**High level Architecture**

**P01:PetsWala**

**<team member names & ids>**

| **Student ID** | **Name** |
| --- | --- |
| **22100300** | **Muhammad Aaish Javed** |
| **22100098** | **Muhammad Ibrahim Bhalli** |
| **20100049** | **Syed Raza Abbas** |
| **22100200** | **Muhammad Tayyab** |

**Table of Contents**

[1.](#_heading=h.gjdgxs) Introduction 3

[2.](#_heading=h.30j0zll) System Architecture 4

[2.1 Architecture Diagram 4](#_heading=h.1fob9te)

[2.2 Architecture Description 5](#_heading=h.3znysh7)

[2.3 Justification of the Architecture 6](#_heading=h.2et92p0)

[3.](#_heading=h.tyjcwt) Risk Management 7

[3.1 Potential Risks and Mitigation Strategies 7](#_heading=h.3dy6vkm)

[4.](#_heading=h.1t3h5sf) Tools and Technologies 8

[5.](#_heading=h.4d34og8) Hardware Requirements 9

[6.](#_heading=h.2s8eyo1) Who Did What? 10

[7.](#_heading=h.17dp8vu) Review checklist 10

# Introduction

## 1.1 Document Purpose

This document is the system requirements specification for an online web-based platform for pet owners and pet enthusiasts. Our product aims to provide pet owners and others in the veterinary or pet service industry with an online platform where both service providers and customers can interact and engage in commerce. The document will provide an overview of what our web application would offer, including but not limited to its functionalities, constraints, assumptions and dependencies.

## 1.2 Product Scope

We will be developing a web application which registers pet owners, vets, other pet service and accessories providers and possibly pet experts and allows them to interact based on their needs. For instance, pet service providers and vets would be able to advertise their services and products and pet owners would be able to contact these service providers as well as connect with other pet owners for any purpose such as breeding, purchasing or selling.

## 1.3 Potential Users

As mentioned earlier, potential users of the web-application will be key characters in the pets’ communities:

· Pet owners

· Service providers

· Rescue Services

· Vets

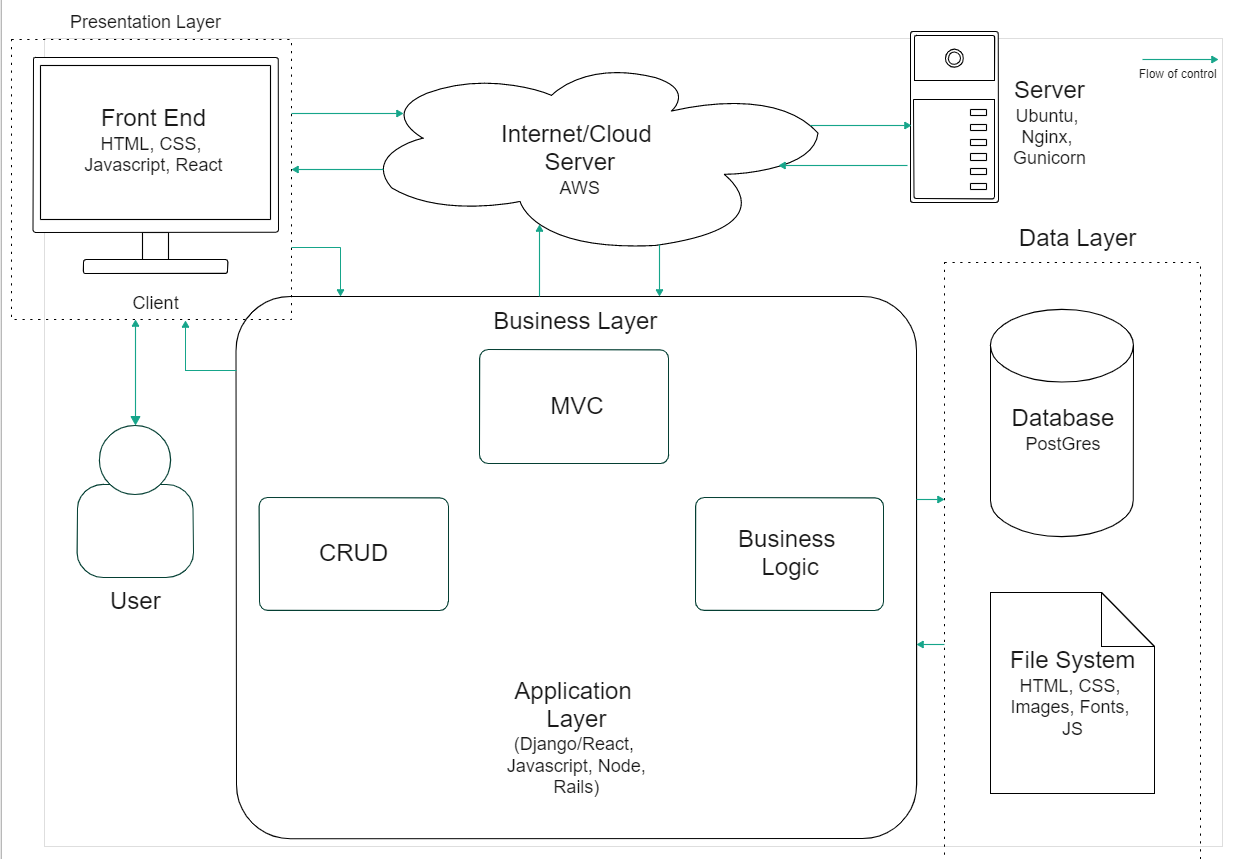
· Pet Buyers

· Pet Sellers

· Business administrators

# System Architecture

## Architecture Diagram

****

## Architecture Description

<Give description of each subsystem in the architecture diagram above. Moreover, give description of how subsystems interact with each other>

User:

The user uses a client machine (any device with modern web browser support) with a decent functioning internet connection to interact with our system. Examples of client machines include but are not limited to laptops and desktops (running Windows, Mac OSX, Linux), modern iOS and android devices etc.

Client (Presentation Layer):

Client is the machine responsible for communicating with the server over the internet. It must have an active internet connection. This is where all the user interaction and user experience happens (UI/UX). The client fetches data from the server and displays it to the client. This layer is called the presentation layer because the data is rendered from the server and presented to the client in this layer. The client requests the server for files necessary to display the website and data according to the user’s needs. The stacks that we use to accomplish this are HTML, CSS, Javascript, Django. The client is where the front end of the system exists.

Server:

Server is the machine responsible for handling all incoming client requests. This is the physical location where all the application layer and business layer is implemented. We use AWS to rent a cloud based server responsible for hosting the service (Petswala). There are a variety of AWS packages available, the free ones being the lowest tier obviously. Our server must have a large enough bandwidth to handle incoming traffic without a bottleneck. Preferably there should be flash storage (SSD) to make processing faster and avoid fail stop failures. The server must have at least 8-16GB of high speed RAM. The server will be running Ubuntu, and we’ll be using services like *Nginx* and *Gunicorn* for the deployment of our website. The server is where the business layer is also implemented. Server is where the backend of the system exists.

Business Layer:

The business layer is the part within the server where the business logic is implemented. Since the business layer is intricately connected with the application layer, we have represented them together in the diagram. The application layer uses Python on Django. This layer is made up of many subsystems, like a model view controller, a CRUD, and business logic etc. For the sake of simplicity we have not represented every subsystem in the diagram.

1. MVC:

The model view controller is a popular schema/pattern for implementing web applications. The model manages system data and primarily interacts with the database. The view is responsible for managing the data presentation to the user. It renders the pages for the users. The controller manages user interactions and passes them to the model and view components. It interacts with the file system in order to fetch and save data.

1. CRUD:

The crud will primarily be used by the administrator and will be part of the admin panel. It has administrator privileges and can edit the database and the file systems. This will be used for registering services, rescue services, and vets. It can also be used to edit or remove blog posts or edit user credentials.

1. Business Logic:

Business logic is the main subsystem and application logic. This will be composed of various modules that are programmed for a specific purpose. These modules will be mostly independent which will ensure efficient maintenance and upgrades. The code will be modular and object oriented.

It will have a sign up and login module, which will be called the authentication module. It will be responsible for storing the user credentials in the database and verifying them upon login.

There will be an admin module which will be responsible for the admin panel. It will have administrative functions and privileges.

There will be a fetch/search module responsible for fetching data from the database. This is necessary for implementing search functionality throughout the website.

Blog management will be a module that implements the functionality of creating blog posts, editing blog posts etc. It also implements the functionality of commenting and rating/reviewing.

A comms module will be responsible for implementing the chat functionality in the application.

Similarly, there will be a services module responsible for implementing the functionality of services in the website.

A module of marketplace will deal with the specifics of how the marketplace is implemented, how the prices, images, and additional information is fetched from the database and file system.

For the sake of simplicity these modules were not represented in the diagram.

Data Layer:

Data layer is made up of the database and file system. The system will be using the PostGres database. The database will store the encrypted information of the clients. Since this is the last layer, by using the layered architecture, we are mitigating the danger of byzantine failures.

The file system also exists within the data layer, this holds all the files used by the backend and the server. From HTML, CSS, JS, Python files to images and other resources that the system may require.

The data layer is where everything the system needs to be functional is placed. It’s important that the data layer is secure and redundant to failures.

Interaction of subsystems:

The front end interacts with the server (backend) in order to render the pages and provide optimal user interaction and experience. The model view controller interacts with the file system, database, and front end. The CRUD and business layer also interacts directly with the data layer. As for the details of how every submodule interacts with every other submodule are beyond the scope of this document.

## Justification of the Architecture

<List down pros and cons of the architecture you have defined in the context of your system. Moreover, give a justification of why this architecture is appropriate for your system. Make sure that you also discuss how this architecture helps in implementation of non-functional requirements. >

We’re using a combination of three system architecture:

* Client-Server
* Layered
* MVC (Model View Controller)

## Django MVC Justification

We’re developing a complex system with several components, which will require a lot of programming. The DJango MVC architectural framework will allow us to use pluggable modules for basic functionality such as user input validation, secure user profiles and information, etc. Since we are already familiar and affluent with Python, the learning curve for the framework will not pose any challenges. Moreover, the framework provides protection against common attacks such as SQL Injection.

| **Pros** | **Cons** |
| --- | --- |
| Instead of writing the code, developers can use readymade packages for adding functionalities. This saves valuable time. | It proves challenging for developers who have worked on frameworks with Convention over Configuration. |
| Since it is written in Python, a versatile programming language, Django for web development provides more flexibility and dynamism to the developers. | Django calls for a lot of coding, which takes server processing time and bandwidth while development. It is generally used for projects which need scaling or are going to be launched on a large scale. |
| Faster development. Startups and enterprises can utilize it to develop rapid MVP and get more time to market the product. | Steep learning curve of this web framework is challenging for developers who are switching to Python. |
| The Django REST framework has a modular and configurable architecture which makes API development easier. | Unlike most web development frameworks, Django can’t handle multiple requests simultaneously as it encourages developers to explore individual processes and make decisions. |
| Django is more advanced and is compatible with some of the powerful machine learning libraries like PyTorch, NumPy, etc. | Although it is easy to learn, its mandatory nature means that developers cannot use their own file structure. One needs to follow those rules to deploy anything using Django. |
| It also includes prevention of common attacks like SQL injections and cross-site request forgery. |  |

## Layered Architecture Justification

Most web applications require a layered architecture. The front end is just static web pages without the backend to link them and implement the functionality. Similarly, the backend depends on the data layer to fetch the data that it requires to implement the business logic and application code.

| **Pros** | **Cons** |
| --- | --- |
| Redundant features (authentication) in each layer can enhance security. | Management cost increases if there are too many layers. |
| Allows replacement of entire layers as long as the interface of the layers does not change. | The performance is affected as more and more layers are added. |
| When changes occur, only the adjacent layer is affected. |  |

## Client Server Architecture Justification

No web application can exist without a client server architecture. We want a web application for people with pets. This non-functional requirement can only be satisfied with a client server architecture. Furthermore, there are going to be a number of clients connecting to the server and interacting through it. The only practical way of meeting the non-functional requirements of our intended system is that we use the client server architecture.

# Risk Management

## Potential Risks and Mitigation Strategies

| **Sr.** | **Risk Description** | **Mitigation Strategy** |
| --- | --- | --- |
| 1. | Risks can take the form of a new cyber security threat, a supplier or service provider that’s no longer able to service your company, or an equipment failure | Identification of vulnerabilities, implementation of firewalls, strict input validations. Investigating for new service providers to be replaced. |
| 2. | Breach in user credential security | Make use of secure frameworks that ensure strong encryption and multiple verification techniques and steps. |
| 3. | Risk of losing to competitors | Designing and executing a proper marketing plan, innovating and improving services to customer satisfaction. |
| 4. | Risk of not finishing project on time | Keeping track of team members’ contribution to the project on a daily basis and providing them feedback on how to improve. |
| 5. | Risk of client not liking the end product and requesting for change in functional requirements | Informing the client beforehand that such requests will take additional time and resources. |
| 6. | Team members become ill during critical times in the projects. | Reorganize the team so that there is more overlap of work and team members should have knowledge of each other's work. |
| 7. | Security gaps or flawed processes that might leave room for vulnerabilities | Identifying potential risks and replacing flawed components with more reliable bought-in components. |
| 8. | Database performance does not keep with the demand on the system. | Investigate the possibility of investing in a higher performing database. |
| 9. | Drawbacks of an initiative before investing resources, time, or money | Cost-benefit analysis in light of budget and scope of initiative |
| 10. | Risk of getting project’s funding cut off | Preparing a proper cost-benefit analysis report of the project and presenting to the relevant authority to prove that the project is making contributions to the end goal and cutting the budget would not be beneficial in the long run. |

# Tools and Technologies

<List down tools and technologies that you plan to use for development and deployment. Make sure that you mention name and version of the tools.>

We will be using following Tools and Technologies for development and deployment:

* Django(v3.0 or above)
* Python(v3.7 or above)
* PostgreSQL (v12.3)
* AWS CLI (v2)
* HTML5
* CSS (v2.1)
* Trello
* Javascript (ES2015)

# Hardware Requirements

The system will use a typical hardware interface. A screen (either of a phone or laptop/PC) will be used. In case of a laptop/PC a keyboard will be needed for inputs. All inputs will be given through a touch screen or keyboard. Outputs will also be shown on this screen. Our system will support both Windows and MacOS. The machine used should have a functioning browser which supports CSS, HTML & Javascript. Since the system will be run over the internet, all the hardware shall be required to connect to the internet through Modem, WAN – LAN, Ethernet Cross-Cable, etc.

## 5.1 Development Machine Hardware Specifications

Development machines must have

* at least 8 GigaBytes of RAM
* quad-core processors
* installed SSDs preferably (for faster load time and execution)
* supported by a 64-bit OS(Windows 10 or above).
* CPU speeds > 2.0 GHz

## 5.2 Deployment Server

We’ll be deploying our web application on a Ubuntu 20.04 LTS (or above) because of its reliability and compatibility. Moreover, it must offer at least

* 25 GB of Storage
* 25 provisioned Write Capacity Units(WCU)
* 25 provisioned Read Capacity Units(RCU)
* Enough to handle up to 200M requests per month

(These specific requirements are based on the minimum Free Tier package offered by AWS)

# Who Did What?

| **Name of the Team Member** | **Tasks done** |
| --- | --- |
| Muhammad Ibrahim Bhalli | Architecture Diagram, System Architecture (Section 2), Hardware Requirements |
| Muhammad Aaish Javed | Architecture Diagram, Hardware Requirements, Introduction, System Architecture |
| Muhammad Tayyab | Tools and Technologies, Risk Management |
| Syed Raza Abbas | Risk Management |

# Review checklist

Before submission of this deliverable, the team must perform an internal review. Each team member will review one or more sections of the deliverable.

| **Section** **Title** | **Reviewer Name(s)** |
| --- | --- |
| Risk management, Hardware Requirements, Introduction. | Muhammad Ibrahim Bhalli |
| Tools and Technologies, Risk Management. | Muhammad Aaish Javed |
| System Architecture, Tools and Technologies. | Syed Raza Abbas |
| System Architecture, Hardware Requirements. Introduction. | Muhammad Tayyab |