

## National College of Ireland

### Project Submission Sheet

**Student Name:** Nipun Bakshi  
**Student ID:** 23202513  
**Programme:** MSc. In Cloud Computing **Year:** 2024  
**Module:** Cloud Platform Programming  
**Lecturer:** Shivani Jaswal  
**Submission Due Date:** 23th April 2024  
**Project Title:** .....  
**Word Count:** .....

**I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.**

**ALL internet material must be referenced in the references section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.**

**Signature:** Nipun Bakshi

**Date:** 23<sup>th</sup> April 2024

#### PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. Projects should be submitted to your Programme Coordinator.
3. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
4. You must ensure that all projects are submitted to your Programme Coordinator on or before the required submission date. **Late submissions will incur penalties.**
5. All projects must be submitted and passed in order to successfully complete the year. **Any project/assignment not submitted will be marked as a fail.**

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

## AI Acknowledgement Supplement

[Cloud Platform Programming]

[Purrific Pet Adoption WebApp]

Your Name/Student Number	Course	Date
Nipun Bakshi/23202513	Masters in cloud Computing	23 <sup>th</sup> April 2024

This section is a supplement to the main assignment, to be used if AI was used in any capacity in the creation of your assignment; if you have queries about how to do this, please contact your lecturer. For an example of how to fill these sections out, please click [here](#).

### AI Acknowledgment

This section acknowledges the AI tools that were utilized in the process of completing this assignment.

Tool Name	Brief Description	Link to tool

### Description of AI Usage

This section provides a more detailed description of how the AI tools were used in the assignment. It includes information about the prompts given to the AI tool, the responses received, and how these responses were utilized or modified in the assignment. **One table should be used for each tool used.**

[Insert Tool Name]	
[Insert Description of use]	
[Insert Sample prompt]	[Insert Sample response]

### Evidence of AI Usage

This section includes evidence of significant prompts and responses used or generated through the AI tool. It should provide a clear understanding of the extent to which the AI tool was used in the assignment. Evidence may be attached via screenshots or text.

#### Additional Evidence:

[Place evidence here]

#### Additional Evidence:

[Place evidence here]

# Purrific Pet Adoption WebApp

Nipun Bakshi

X23202513

[x23202513@student.ncirl.ie](mailto:x23202513@student.ncirl.ie)

National College of Ireland

Deployed application URL:

[PURRIFIC \(eba-hpsw3qd6.central-1.elasticbeanstalk.com\)](http://PURRIFIC(eba-hpsw3qd6.central-1.elasticbeanstalk.com))

**Abstract--** Bringing a new pet into your home is often a thrill that offers warmth and company. Millions of animals housed in shelters may also benefit from it. By the last year, adoption of dogs and cats has been gradually growing. One of the largest welfare organizations called Dogs Trust, showed its statistics that past decades, their numbers indicating a 5% rise in the adoptions of dogs have remained the same. Adoptions of long-lived members of the cat kingdom increase by 2%. the growth of the same adoption can be largely attributed to the spreading of information on damping of to animals. The purpose of app is to promote adoptions in general to close the gap between the adopters that are looking for a pet and those that already looking for a family at different community shelters and rescue centres. The report has details on the services used from AWS and how stable and project functioning web application is. It enlists the advantages of employing cloud-native architecture in terms of unlimited scalability, redundancy, and low cost. Like that, practicing usage of the cloud interface, AWS CLI, by means of which you can create deployment scripts and control AWS resources will be another problem will be focused on. The challenges created by the mission to define and run the program without having these troubles. Aiding in the development and actualization of contemporary online features and systems. It helps me get hands-on experience in setting up AWS cloud infrastructure that can be scaled up, relied upon, and cost-effective for adoption or related areas.

AWS services, which are, e.g., used for application deployment and cloud scaling with the help of AWS Elastic Beanstalk, database management with RDS, continuous integration and deployment with CodeBuild and CodePipeline, image storing with Amazon S3, sending notifications with AWS SNS and integrated development environments with AWS Cloud9.

**Keywords—** Django web application, S3 bucket, Cloud9, Amazon RDS, Elastic Beanstalk, Code Pipeline, AWS Cli, CodeBuild, python library

## I. INTRODUCTION

A user-friendly web application that makes adopting a pet easier for both adopters and shelters. The initiative aims

to streamline the process of searching for adoptable dogs by gathering data into a single, easily navigable platform. This would facilitate the process for individuals or families in finding the perfect companion.

Home checks are done to make sure the adoption can provide the animal a suitable home and to reduce the challenges faced by both future pet adopters and shelters/pet rescue organizations during the adoption process. Adoption of pets is gaining pace but the diversity and magnitude of shelters and animals available for petting creates a wall of confusion due to fragmented details and limited information access. Also pet shelters are most of the times when it comes to outreach and managing adoption events effectively always find time to come up with a better approach.

Main objective of Purrific is to adopt pets and put your pet up for adoption. The admin has access to host events for the pets and show case them. Website can be used to view the pets and images is store on S3 bucket. If any user wants to contact regarding any pet adoption service, they would be asked to fill out a contact form and will receive an email notification from SNS service and if a user would want to donate, a confirmation slip is sent to them specifying the amount and other necessities. Created a library in python which sorts according to age and latest postings, filters according to categories and search for desired results by the user.

Hosting this on cloud9 as it as cloud-based IDE where it made work seamless, resource scaling is done automatically and can skip the hassle of managing the infrastructure. Allows to integrate with other AWS services conveniently and has an efficient environment for developing my web application.

## II. SPECIFICATIONS AND REQUIREMENTS

**Django web framework** – I chose Django for this project because Django<sup>[1]</sup> offers a platform for building applications quickly and its feature an (ORM) Object-

relational mapping which interacts with the database, and its built-in feature manages content, URL routing, authentications. Its MVT architecture stands out, as each layer is responsible for their own. This framework fit the best for my applications a its scalable, secure, and easy to maintain.

**Cloud9**<sup>[2]</sup> - Hosting my web application on cloud9 eliminated most of my manual work and ensured consistency, where could work on my application from anywhere as it has automatic resource scaling. Integrates easily with other AWS services for development workflow.

**Amazon RDS**- A fully managed database provided by AWS, it has high-availability features like automatic backups and multi-deployments, it ensures fault-tolerance and is reliable. Scalable and allows workloads to vary seamlessly while providing security features, such as encryption and safeguarding your data. It also automates database engines allowing it to efficiently deploy, monitor and automate. My application does not need NoSQL database, so while choosing a database Amazon RDS MySQL was more fit for my application as it needed a structured database and provides the security of data. Setup was easy as I followed a tutorial<sup>[3]</sup>

**CodeBuild**<sup>[4]</sup> – It is a continuous integration service provided by Amazon to build, test and process application code. CodeBuild allows to easily integrate with Code Pipeline (CI/CD) to enable end-to-end automation. Using CodeBuild I can configure my build environment, source code repository, output artifacts, and build commands. Basically, it simplified and accelerated the build and test stage of my web application by delivering high-quality.

**Code Pipeline**- A fully managed continuous integration and continuous delivery service. Automates the build, test, and deploy from application source code to production. Connected my source code repository from Github<sup>[5]</sup> to code pipeline and added build stage using CodeBuild and in my deployed it on Elastic Beanstalk deploy stage. With the help of this I could easily visualize and track the progress my software releases, monitor pipeline execution stage and action.

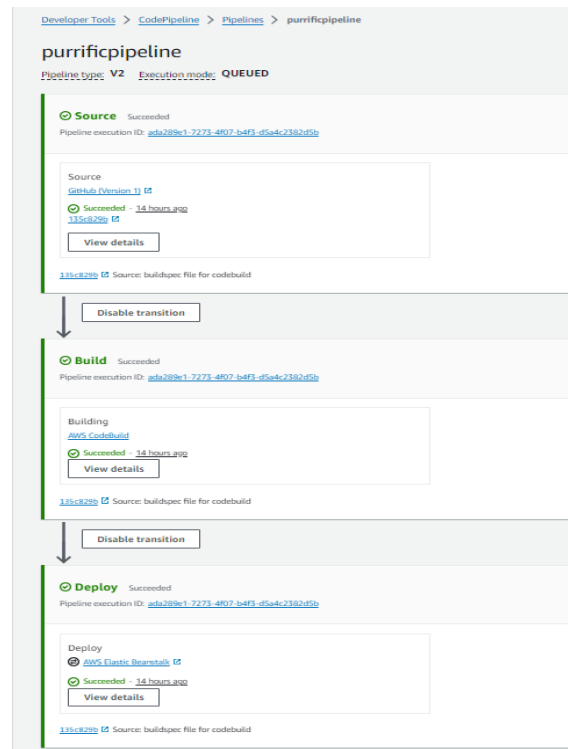


Figure 1. CI/CD Codepipeline flow<sup>[6]</sup>

**AWS Elastic Beanstalk**<sup>[7]</sup> - Amazon Web Services (AWS) offers a fully managed service, I used Elastic Beanstalk, because it supports easy deployment as well as management of the web applications and services. It robotises the tasks of adding and removing servers and providing monitoring and control over the infrastructure, taking these tasks off the shoulders of developers who thus can focus on writing code. With Elastic Beanstalk, you can, though, use applications developed in various programming languages (Java, DOTNET, PHP, Node.js, Python, Ruby, and Docker images) with the ease offered by this platform. Elastic Beanstalk is not meant to expose you to infrastructure complexities. It gives a platform whereby you comfortably deploy, monitor, and scale up your applications by either clicking few times or giving commands. It is concerned with self-balancing of load, auto-scaling, health check-ups, and rolling updates, which guarantee application availability and reliability. In a nutshell, Elastic Beanstalk speeds up developers' tasks and minimizes operational processes, empowering them to have faster iterations and innovation, so this becomes a right option to serve and manage web applications and services in the cloud. For setup and

deployment I followed the steps from the guide provided to us by our college.

**AWS S3** – S3 is an exceptionally reliable and scalable object storage service provided by the reCognition of AWS (Amazon Web Services). It was very easy to set up and it stands out as the indispensable basis in the cloud systems. It serves the users with the opportunity to manage infinite but highly reliable datasets via the internet. S3 will support the durability of 99.999999999%, and data can be managed from 1 byte to 100TB. In turn, it was highly convenient for me, as S3 has an easy-to-use API, which provides a possibility of customising the access and integrating with the entire platforms of AWS services and third-party apps. Through AWS S3, empowers us to take the whole of AWS infrastructure and use it for our storage needs.

**AWS SNS<sup>[8]</sup>** - SNS (Simple Notification Service) is a fully managed and supervised messaging service offered by Amazon Web Services (AWS) and allows for sending of notifications to different endpoints/clients such as device mobile, email address, HTTP endpoints and many more. In the project that I worked on, I used AWS SNS to send out notifications by email to those who filled "Contact US" form on the website and as well as donors who contributed to my project. I coded my app in such a way that it will automatically send the notifications on precise events triggered by certain actions, like form submissions and donations. Thus, this keeps users updated, similar to when we answer questions in real-time or appreciate their contributions. The AWS SNS software offers a scalable and reliable alternative to workflow in notifications, since it is made with features including message delivery retries, message filtering and communication protocols. Through the help of AWS SNS, I will be in a position to collaborate better with users to interact with them with ease as well as provide them with a good user experience.

**AWS CLI** - AWS CLI<sup>[9]</sup> is the advanced tool provided by Amazon Web Services (AWS) grants users an authority to explore different AWS services and resources from the command line interface. Through AWS CLI, users can control their AWS resources, execute scripts to automate tasks and simplify workflows, as well as access a finance management

console, using simple commands. In this project, I used it to manage permissions and accessibility for your resources through the console command line interface, AWS CLI. In particular, I have employed AWS CLI to provide public accessibility for my S3 (Simple Storage Service) buckets through HTTP queries and response which will help users to retrieve and upload objects. Besides I likewise utilized the AWS CLI for configuring permissions for the SNS (Simple Notification Service) topics that allows anyone to be able to subscribe to topics and get notifications. Through the tool Amazon Web Services Command Line Interface. I have been able to manage permissions and access controls for your Amazon Web Services resources easily, which allowed me to interact smoothly with my web application through the Amazon Web Services Command Line Interface tool and at the same time while ensuring security and compliance with best practices.

### III. ARCHITECTURE DESIGN

**Cloud9 IDE:** AWS Cloud9 is a cloud-based environment that has been designed to serve as an IDE directly from the web. It's also very useful for team projects.

With it, I could code, run and debug projects directly from the browser.

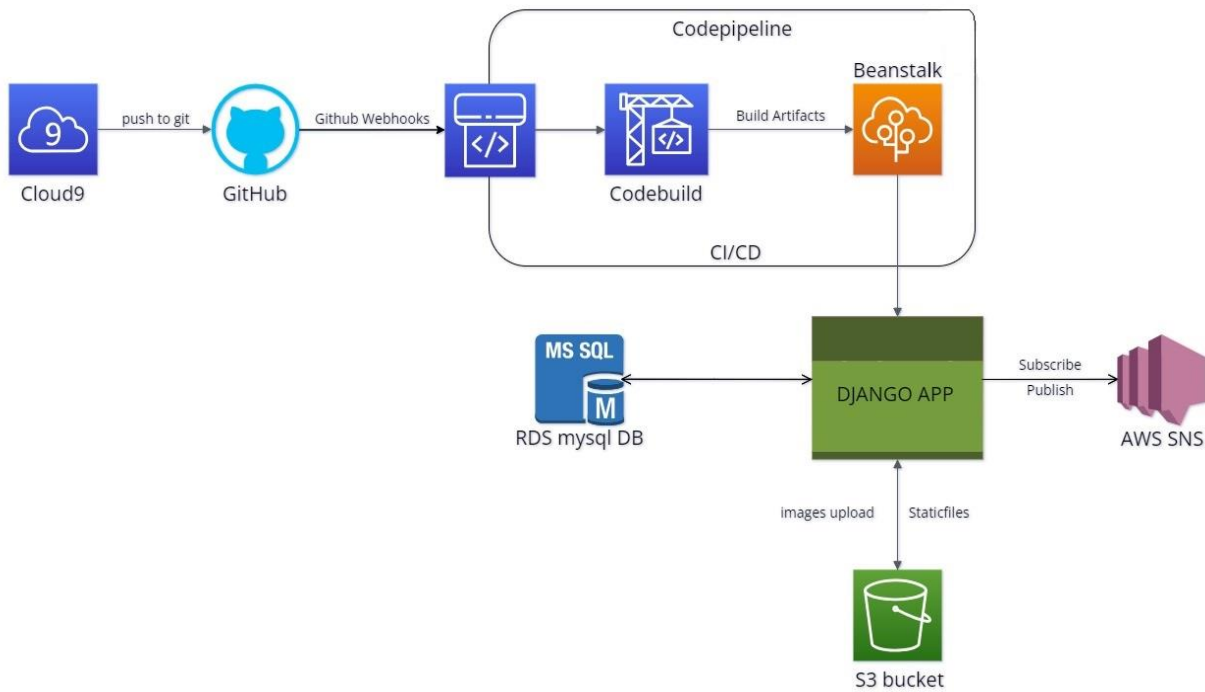


Figure 2. Cloud Architectural Diagram of the App

I used Cloud9 to write and control my application code and pushed it to GitHub.

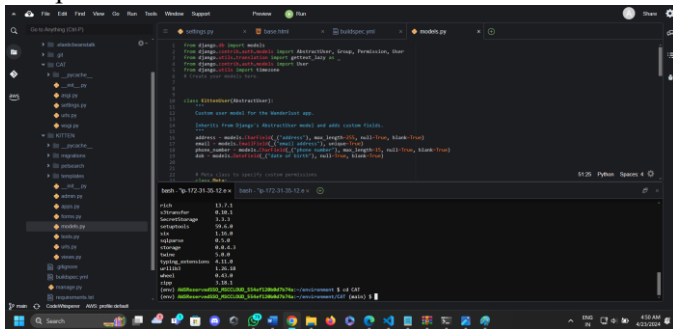


Figure 3. my cloud9 environment

**GitHub:** Github is an internet based platform as well as a Version Control System that helps developers to collaborate on various projects, tracking changes in the code and implementing the framework for the software development. It hosts code, and consists of features like issue tracking, pull requests, and project management which makes the central hub where software development teams can work together to communicate and release software efficiently. Through GitHub webhooks, the change in the repository is noted and it sends a signal to AWS codepipeline.

**CodePipeline:** CodePipeline is a CI/CD service that is automated, whereby it wraps around the releases processes, executing the build, tests and release phases for you.

As soon as a change is made in the github files, the codepipeline starts again going through all the stages

which allows us for continuous integration and continuous Deployment.

CodePipeline combines with several other AWS services, including CodeBuild and ElasticBeanstalk to automate the deployment process for the apps.

**CodeBuild:** CodeBuild is a fully managed build service that participates in compiling source code, running tests, creating artifacts, and doing them for deployment purposes.

For the use of CodeBuild in the framework of my Django application, I build the application, run tests using a buildspect.yml file, and wrap the application code up for deployment purposes.

It incorporates with CodePipeline for the automation of the build process and works in conjunction with my CI/CD pipeline.

**AWS Elastic Beanstalk:** AWS Elastic Beanstalk is an instance of PaaS (platform as a service), which was developed for the purpose of enabling deploying applications and services to be less difficult to manage than usual.

It actually provisions and manages the underlying infrastructure needed to run my application. (example – EC2 instances, load balancer) .

I am also capable of auto-scaling, load balancing, and general health monitoring of the application in your Django deployment by using Amazon Elastic Beanstalk, which takes care of capacity provisioning for me.



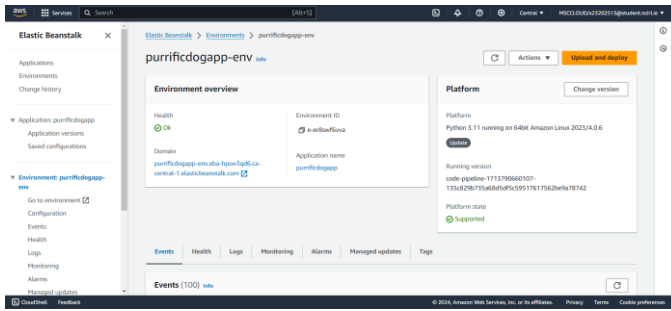


Figure 4. my beanstalk environment

**Django App:** On the user side, the Django web app will serve as a node to process client requests, database connection and dynamic content generation. Four major components which consist of Django views, Django models, templates and static files that work harmonious to provide my application functionality. The Django app service created on Elastic Beanstalk makes use of the RDS database together with the S3 storage for storing and managing data and assets respectively. It also sends notifications to users using AWS SNS service and allow them to subscribe to a topic.

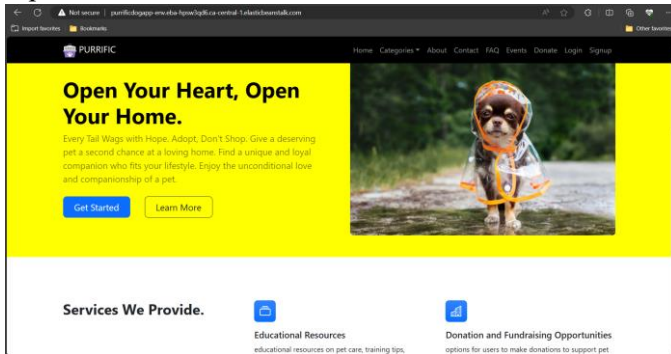


Figure 5. app homepage

## RDS (MySQL):

Amazon RDS (Relational Database Service) is a managed relational database service which focuses on showing an easy way to create, maintain, and scale relational databases in the cloud.

I hosted a MySQL database instance for your Django application using RDS and thus application can store and manage its data, including user accounts, pet data and so on.

The RDS performs all the base functions of database like backups, application of patches, monitoring and scaling of databases, on the other hand, I can use all the energy on building your application.

## Amazon S3:

Amazon S3 is a platform object storage service that provides scalable, resilient, and highly accessible storage options for different data types.

In terms of my Django application, I saved files that do not need to be edited (like CSS, JavaScript files) as well as my static images and images uploaded by users to S3. S3 has a storage capability with versioning, encryption and lifecycle management as its essential features.

## SNS:

Amazon SNS (Simple Notification Service) is an easy-to-use pub/sub message service which helped me to deliver electronic messages to a number of endpoints for example email, SMS, HTTP or even AWS Lambda functions. I used email in my case.

In the Django application version of SNS, it can send emails as a notification when visitors submit forms like contact form or make a donation. SNS is a great tool that assists in sending reminders, using channels of communication that are favorable for my users as well as helpful in reaching maximum number of targets.

## IV. LIBRARY DESCRIPTION

In my Django Project, I have created an in house library<sup>[10]</sup> named "petsearch" which helps in creation of easy to use operations that help in pet searching and segmenting base on different attributes. This library comprises two modules: "filters.py" and "search.py",. I was also able to successfully publish this library on PYPI<sup>[11]</sup> and I was able to install it via "pip install petsearch"

```

AWSReservedSSO_MSCLOUD_554ef120b0d7b74a:~/environment $ source env/bin/activate
(env) AWSReservedSSO_MSCLOUD_554ef120b0d7b74a:~/environment $ pip list
Package            Version
-----
asgiref            3.8.1
backports.tarfile  1.1.0
bcrypt            4.1.2
boto3              1.34.88
botocore           1.34.88
certifi            2024.2.2
cffi               1.16.0
charset-normalizer 3.3.2
cryptography       42.0.5
Django             4.2.11
django-storages    1.14.2
docutils           0.21.1
idna               3.7
importlib_metadata 7.1.0
jaraco.classes     3.4.0
jaraco.context     5.3.0
jaraco.funtools    4.0.1
jeepney            0.8.0
jmespath           1.0.1
keyring            25.1.0
markdown-it-py     3.0.0
mdurl              0.1.2
more-itertools     10.2.0
mysqlclient        2.2.4
nh3                0.2.17
paramiko           3.4.0
petsearch          1.1
pillow             10.3.0

```




Figure 6. Installed 'petsearch' library using pip install petsearch

Created in the "filters.py" module, we have specified functions that enable you to filter potential pets depending on the personal profile like type age. Say for instance, the function filter\_pets\_by\_type() becomes handy in refining pets by types (like dogs, cats, birds) while the function filter\_pets\_by\_age() gives a user the opportunity to filter the pets by age. These resources - functions - are taking the input parameters represented by a pet attribute and return a QuerySet containing pets that satisfy the given criteria.

The file "search.py" has functions that carry out pet search depending on the name, breed and type of the the pet under consideration. By way of the illustration, the search\_pets\_by\_name() method produces a search for pet names that contain a given query string while the search\_pets\_by\_breed() pinpoint the search by pet breed. The next function in the code is also quite interesting, search\_pets\_by\_type() as it allows you to search your pets based on their type. To achieve this, the functions make use of the icontains lookup, to enable case-insensitive partial match on the relevant attribute.

Filters and search capabilities which also use Django functions have been integrated into our views.py file by importing this library to provide our users with search and filtering possibilities. Z. For instance, we use search fields to allow the user to look for pets by name, breed or

type. More so, our filter functions help users to retrieve information about pets by type or age range. Such attributes are well combined in us so as to be more precise at pet finding for users. Effectively, this is done through the use of keywords or pre-defined filters when identifying pets.

In summary, "petsearch" library being inserted into my project is a valuable printing that provides the user with convenient tools to access and handle pet information, categorize it by different attributes.

## V. CONCLUSIONS AND FINDINGS

Meanwhile I was gathering information and statistics about the evolution of our Django-based pet adoption app on AWS services, these new realizations and insights helped me to approach the task of building scalable and reliable web applications in different way.

Moreover, Django as the web framework was a robust basis that gave me all I needed to continue with the development of my Pet app. Django's inbuilt features helped the development process a lot such as ORM for database interaction, a built-in authentication system, and template engine helps us to render dynamic content and speed up the development process and which helped to focus more on developing the application core features and user interface improvements.

I made a highly customized AWS architecture (elastic beanstalk, RDS, S3, code pipeline, codebuild) to streamline the deployment and CI/CD pipelines. Elastic Beanstalk transformed the deployment and management of my application into an effortless process by automatically dealing with infrastructure provisioning, load balancing, as well as auto-scaling. This not only reduced the operational burden of managing server instances manually but also provided all the potentially useful features.

RDS<sup>[12]</sup> was a key component in establishing the database's performance, scalability and security so that I could save and manage the pet listings, user data and application settings and make them available to the system correctly. S3 was a scalable and replacement of files such as user uploaded photos, JS and CSS among



others, whereas CodePipeline and CodeBuild were used for implementing our CI/CD pipe and availing a seamless integration, testings and deployment of the coded files.

Understanding the challenges I would face in future for a pet adoption platform, scalability and reliability of the platform comes out to be an important factor, since these sites usually show this amount of traffic and user activity. The AWS cloud services can offer the necessary scaling and robustness to accommodate these variations in demand. Consequently, the users will have the best performance and seamless experience.

Moreover, CI/CD pipelines implementation dramatically increased my development efficiency by eradicating repetitive tedious work such as code compilation, testing, and deployment that were formerly done manually. Thus I was able to polish any technical deficiencies speedily as well as bring new functionalities to my app in very less amount of time. Furthermore, the quality of the code remained quite high.

At last, I would say that I have received teaching on creating the Django program and applying AWS services which have granted me with hands-on knowledge of how to create a large-scale, efficient and intelligible web applications. Bringing the power of Django web framework and the flexibility of AWS cloud services, I have made a Pet adoption app that is for all the users and the developers too. With these insights at hand in mind as we embark on future projects, we'll be better equipped to develop forward-thinking and robust solutions that not only meet the expectations of our users, but also surpass them.

## VI. REFERENCES

- [1]. Why use Django-  
<https://djangostars.com/blog/why-we-use-django-framework/>
- [2]. How to use cloud9-  
<https://www.heptabit.com/blog/aws-cloud9-what-it-is-how-it-works>
- [3]. Setup AWS RDS mysqlldb-  
[https://www.youtube.com/watch?v=by0EJ4qL8ek&ab\\_channel=AWSMadeEasy](https://www.youtube.com/watch?v=by0EJ4qL8ek&ab_channel=AWSMadeEasy)
- [4]. What is codebuild?-  
<https://docs.aws.amazon.com/codebuild/latest/userguide/welcome.html>
- [5]. Introducing github-  
<https://docs.aws.amazon.com/codepipeline/latest/userguide/concepts.html>
- [6]. Concepts of Codepipeline. How it works?-  
<https://docs.aws.amazon.com/codepipeline/latest/userguide/concepts.html>
- [7]. AWS elastic beanstalk features  
<https://aws.amazon.com/elasticbeanstalk/details/>
- [8]. Setting up a SNS topic and create subscription  
<https://www.javatpoint.com/aws-sns>
- [9]. Grant permissions using aws CLI-  
<https://awscli.amazonaws.com/v2/documentation/api/latest/reference/lakeformation/grant-permissions.html>
- [10]. How to create Python Library-  
<https://medium.com/analytics-vidhya/how-to-create-a-python-library-7d5aea80cc3f>
- [11]. Publishing to PYPI  
<https://www.serviceobjects.com/blog/step-by-step-guide-to-publishing-python-libraries-to-pypi/>
- [12]. Why RDS should be preferred-  
<https://cloudacademy.com/blog/amazon-rds-vs-dynamodb-12-differences/#:~:text=The%20main%20difference%20between%20the,offers%20more%20features%20and%20flexibility.>