1. INTRODUCTION

1.1 Project Overview

A naturalist is someone who studies the patterns of nature, identifies a different kind of flora and fauna in nature. Being able to identify the flora and fauna around us often leads to an interest in protecting wild spaces and collecting and sharing information about the species we see on our travels is very useful for conservation groups like NCC.

When venturing into the woods, field naturalists usually rely on common approaches like always carrying a guidebook around everywhere or seeking help from experienced ornithologists. There should be a handy tool for them to capture, identify and share the beauty to the outside world.

Field naturalists can only use this web app from anywhere to identify the birds, flowers, mammals, and other species they see on their hikes, canoe trips and other excursions.

In this project, we are creating a web application which uses a deep learning model, trained on different species of birds, flowers and mammals (2 subclasses in each for a quick understanding) and get the prediction of the bird when an image is been given.

1.2 Purpose

This Project aims to identify a species in a forest or in any other place, we need to carry a heavy book or seek a professional like botanist or zoologist or an ornithologist, but there should be a handy tool for them to capture, identify and share the beauty to the outside world.

2. LITERATURE SURVEY

2.1 Existing problem

There is a need to carry a guidebook around every time we leave for a biodiversity reserve to identify its organisms. Internet databases does not have a easier method to search through them and we need to manually go through them but using ai and advanced ml algorithms we can improve this problem

2.2 References

PAPER-1

Plant Species Recognition Using Morphological Features and Adaptive Boosting Methodology (2019).

Authors: MUNISH KUMAR, SURBHI GUPTA, XIAO-ZHI GAO AND AMITOJ SINGH

The paper uses a novel plant species classifier that recognizes the plant species in the image. Out of many features, leaf shape is a conspicuous element that most algorithms rely on to perceive and describe a plant. The system extracts the morphological features of the plant leaf and classifies using Multilayer Perceptron and other classification algorithm along with AdaBoost methodology. Different classifiers, i.e., KNN, Decision Tree and Multilayer perceptron are employed to test the accuracy of the algorithm. The authors have observed that the maximum precision rate of 95.42% has been achieved for 32 kinds of plant leaves and the proposed system has performed better than the existing techniques for plant leaf recognition.

PAPER-2

Ungulate Detection and Species Classification from Camera Trap Images Using RetinaNet and Faster R-CNN (2022)

Authors: Gholamreza Anbarjafari, Ilja Pavlovs, Kadir Aktas ,Egils Avots, Jevgenijs Filipovs, Agris Brauns, Gundega Done, Dainis Jakovels, Gholamreza Anbarjafari

This paper presents a new dataset of wild ungulates which was collected in Latvia. It demonstrate two methods, which use RetinaNet and Faster R-CNN as backbones respectively, to detect the animals in the images. Faster R-CNN–ResNet50 network and RetinaNet were trained for 34,850 iterations (10 epochs) on the training dataset with a batch size of 4, learning rate of 0.0001 and Adam optimizer for the weight update. The general structure of the detector involves image embedding, object localization and classification. DNN consisting of convolutional layers which are used for the feature extraction from the input image. Usually, backbone networks which are pretrained on a natural image dataset such as ImageNet are used. Common networks used as the backbone are ResNet50, VGG160, Inception-ResNetV2 and DarkNet-19. The neck network takes and processes inputs from the different layers of the backbone, harnessing advantages of data pattern distribution over different feature map scalesby using FPN (Feature Pyramid Network). A feed-forward neural network which performs the classification or regression task.

PAPER-3

Species determination using AI machine-learning algorithms: Hebeloma as a case study.

Authors: Peter Bartlett1et

In this paper, Peter Bartlett1et al (2022) proposed the work related to the species determination using AI and machine learning algorithms. They used the Hebeloma species as a case study and found that any species with sufficient datasets will find the best algorithm for processing it. The species identifier was able to identify 77% correctly with its highest probabilistic match, 96% within its three most likely determinations and over 99% of collections within its five most likely determinations. Each hidden layer has an activation function of either Rectified Linear Unit (ReLU) or Mish. The dimensionality of each hidden layer was set equal to the maximum of

the dimensionality of the feature set and the dimensionality of the class.A total of five optimizers were used to minimise the loss function. For both the Adam and AdamW optimizers, the “AMSGrad” variation proposed was also evaluated.

PAPER-4

Animal classification system: a block-based approach. Procedia Computer Science.

Authors: Kumar YS, Manohar N, Chethan HK

This paper provides insight into a computer-assisted animal classification system from wildlife and field images. An experiment was conducted using 4000 sample images which consisted of 25 different classes of animals, for which each class varies from 40 to 300 images for the classification task. The images were taken to study the effect of the proposed method with large intra-class variations and different viewpoints. Classification of animals mainly had 3 stages viz, segmentation, feature extraction and classification. Segmentation is carried out to discard the background and obtain the region of interest with the animal in it. Iterated Graph Cuts algorithm is used for this task. Then feature extraction is performed using facial features, body shape and colour texture moments making use of local Fourier transform. Then classification is done using K- Nearest Neighbour (KNN) and Probabilistic Neural Networks (PNN). The performance of different classifiers for the classification of animal images is documented and the results are favourable for KNN classifiers to achieve relatively higher accuracy in all cases.

PAPER-5

"Deep Learning for Plant Identification in Natural Environment", Computational Intelligence and Neuroscience, vol. 2017, Article ID 7361042, 6 pages, 2017. https://doi.org/10.1155/2017/7361042

Authors: Yu Sun, Yuan Liu, Guan Wang, Haiyan Zhang

Yu Sun et al. have proposed a deep learning model containing 26 layers and 8 residual building blocks for uncontrolled plant identification, designed for large-scale plant classification in the natural environment. They have used the BJFU100 dataset containing 10,000 images, collected by mobile phones, of various plant species from the Beijing Forestry University. The 26-layer residual network i.e., the ResNet26 model is mainly designed using bottleneck building blocks. The input image is fed into a 7 × 7 convolution layer and a 3 × 3 max pooling layer followed by 8 bottleneck building blocks. Using SGD optimization, the proposed ResNet26 model results in 91.78% accuracy. This is also seen to be significantly higher than the ResNet18, ResNet34, and ResNet50 models which yield a test accuracy of 89.27%, 88.28%, and 86.15%, respectively. They have also tested the ResNet26.

PAPER-6

Convolutional Network based Animal Recognition using YOLO and Darknet (2021)

Authors: B.Karthikeya Reddy, Shahana Bano, g.Greeshmanth Reddy,Rakesh Kommineni, p.Yaswanth Reddy

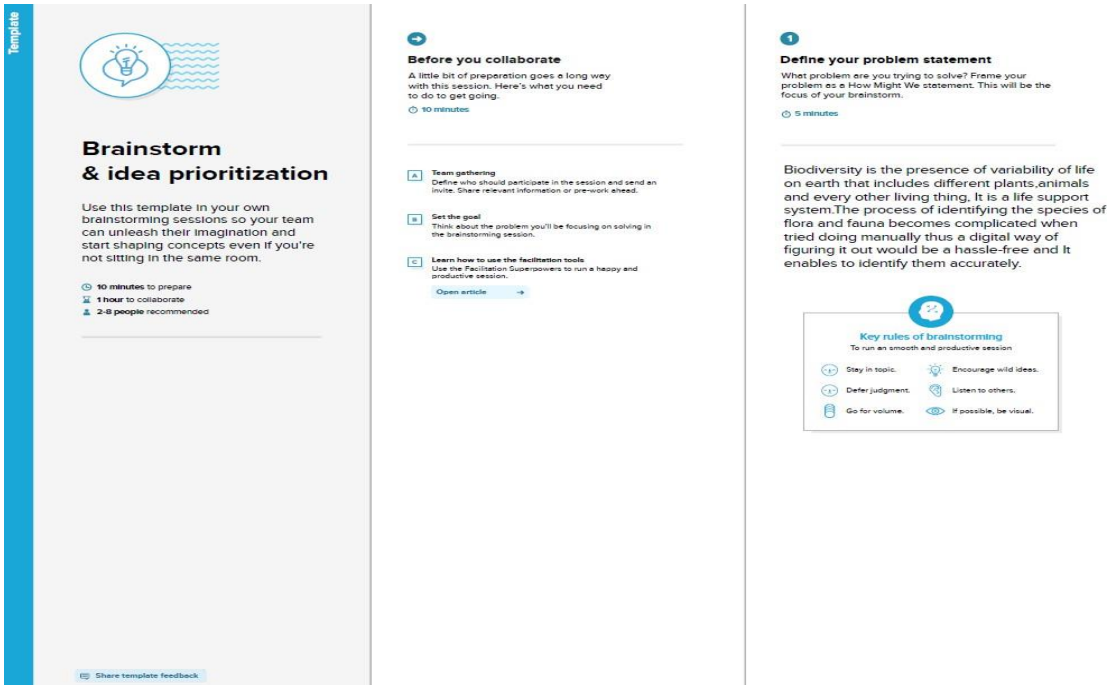
This research work has developed a YOLOV3 model to identify the animal present in the image given by user. The algorithm used in YOLOV3 model is darknet, which has a pretrained dataset. Machine learning has been applied to image processing. The image of animal will be given as input, then it will display the name of the animal as output by using YOLOV3 model. The detection is done by using a pre-trained coco dataset from darknet. The image is broken into variouslengths and widths based on the given input image. Here for the recognition of image, YOLOV3 model is using recognizer deep learning package. The overall performance of the model is based on the different training images and testing images of the dataset. The detection is done by using a pre-trained coco dataset from darknet.

2.3 Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

3.2 Ideation & Brainstorming



2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Ganesh K

- Closely Related Species
- Edibleness of the species
- Identify poisonous species of plants
- Rarity of species

Vijaya raghavan V

- Biological name of the species
- Interactive UI with camera interface
- Data Augmentation
- Describing the threat level of the species

Sudhakaran S

- Uses of the species
- Appropriate CNN based classification model
- Origin of the species
- Accurate Results

Kishore P

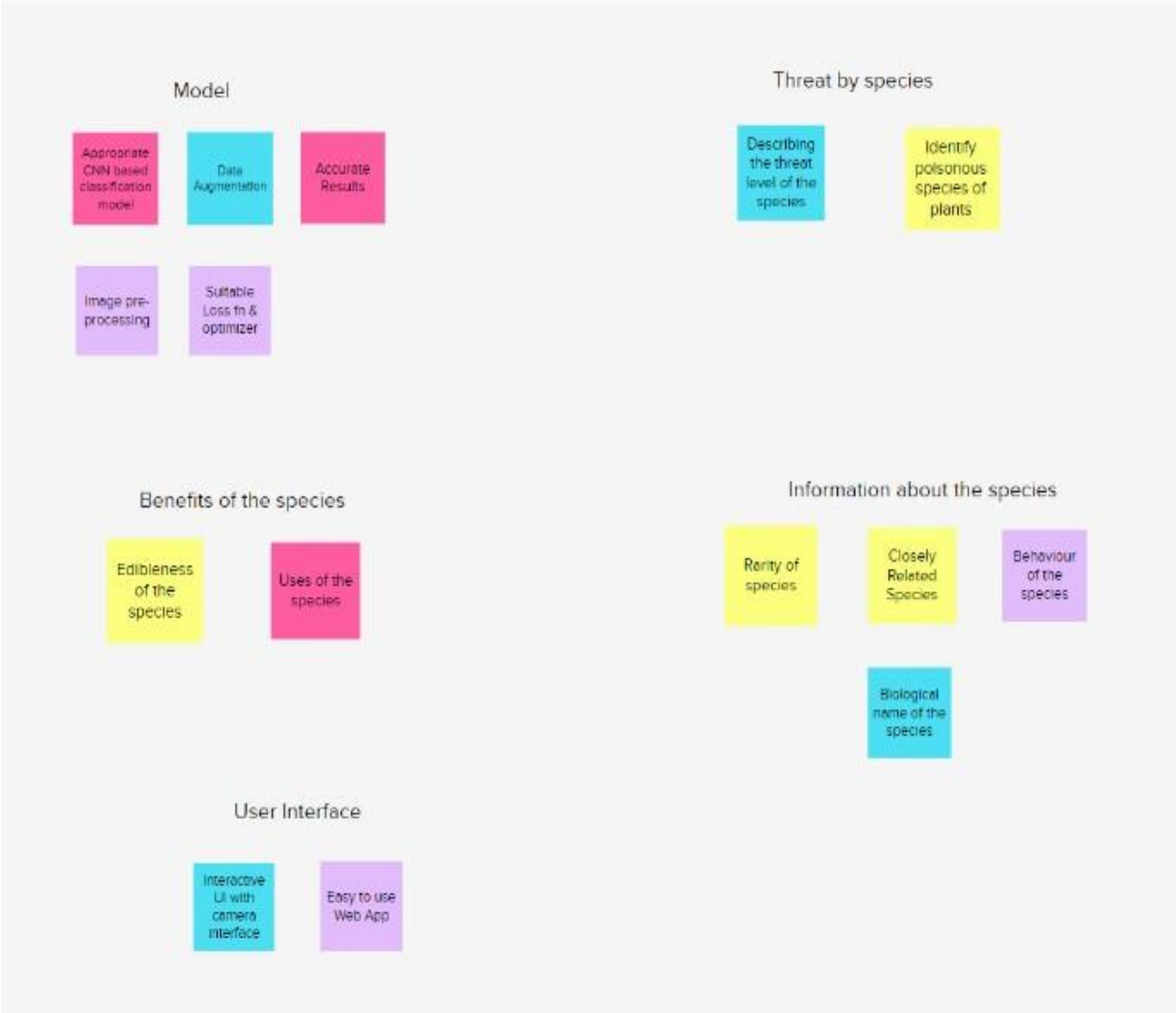
- Behaviour of the species
- Image pre-processing
- Suitable Loss fn & optimizer
- Easy to use Web App

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes



3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To identify a species in a forest or in any other place, we need to carry a heavy book or seek a professional like botanist or zoologist or an ornithologist, but there should be a handy tool for them to capture, identify and share the beauty to the outside world.
2.	Idea / Solution description	A system is built by using the Image/object recognition and classification using (CNN) Convolutional neural network which while using this system, we can capture the image of any animals and plants and can obtain the information about the flora and fauna at any time.
3.	Novelty / Uniqueness	Use of transfer learning in pre trained models to increase accuracy and training time along with data augmentation to increase the dataset size which will in turn yield more accuracy.
4.	Social Impact / Customer Satisfaction	The user can identify the type of species faster and easier without searching in books page by page. It is a useful product for all the research analyst, Ornithologist, Biologist and Marine drivers who can instantly capture images of different species and are able to get all the relevant information about those breeds.
5.	Business Model (Revenue Model)	The model could be open sourced, but we can get some revenue via ads. we will also add a few extra applications like storage and bookmarks permanently for a specific amount of payment.
6.	Scalability of the Solution	The system apart from researchers can also be used by students, hikers or other people who are very much interested in the wildlife. This can also help children learn about different species and their sub species of different flora and fauna which can only be found in the other part of the world

3.4 Problem Solution fit

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

- Ornithologist
- Botanist
- Zoologist
- Students
- Hiker
- Marine biologist
- Research people
- Tourist

6. CUSTOMER CONSTRAINTS CC

- Network issues
- Insufficient knowledge about the biodiversity.
- Cannot remember all the basic life saving tips
- Making observations among species.

5. AVAILABLE SOLUTIONS AS

- Need to always carry a guidebook around everywhere
- Internet databases where we must search for certain species from the mountain of images from the web using modern algorithms.
- Usage of ai to tackle different complex difficulties in the wildlife.

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

- Unable to identify sub species of certain amphibians or birds.
- Cannot find a suitable place to work in the workplace
- Cannot find the exact habitat of certain species.

9. PROBLEM ROOT CAUSE RC

- complexities in identification
- Information gathering
- Need to depend on external resources
- Large dataset

7. BEHAVIOUR BE

- Volunteering for jobs where we can actively work with wildlife
- Finding rare and endangered species of flora and fauna and help them navigate in current

Focus on J&P, tap into BE, understand RC

3. TRIGGERS

- Save nature
- Save Endangered Species
- Expanding the lifespan of certain species through medicine
- Helps to gather aerial species away from places where they are prone to tower kill or other dangers

4.EMOTIONS: BEFORE/AFTER

- Co2 to o2
- Imbalanced world to sustainable world
- Accumulation of waste to renewable energy

10. YOUR SOLUTION SL

- It can be in offline mode
- all information about the Species should be displayed.
- Medical Benefits of different plants can be displayed.
- Display alert messages for plants/animals.
- display alert messages for plants and animals.

8. CHANNELS of BEHAVIOUR CH

ONLINE

- capture image and search it
- Browse using the internet

OFFLINE

- Hand notes
- Getting the information from experienced user

Identify strong TR & EM

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	• Registration through Google API
FR-2	User Confirmation	• Confirmation via Email • Confirmation via OTP
FR-3	Transactions	• Through UPI, Credit/Debit cards and NetBanking.
FR-4	Authentication	• Through OTP sent to mobile. • User created secured passwords.
FR-5	Authorization	• Basic Authorization
FR-6	Administrative functions	• Adding, Updating and Maintaining descriptiondata about various species.
FR-7	External interfaces	• Easy to access UI • Community for discussions

4.2 Non-Functional requirements

5. Following are the non-functional requirements of the proposed solution.

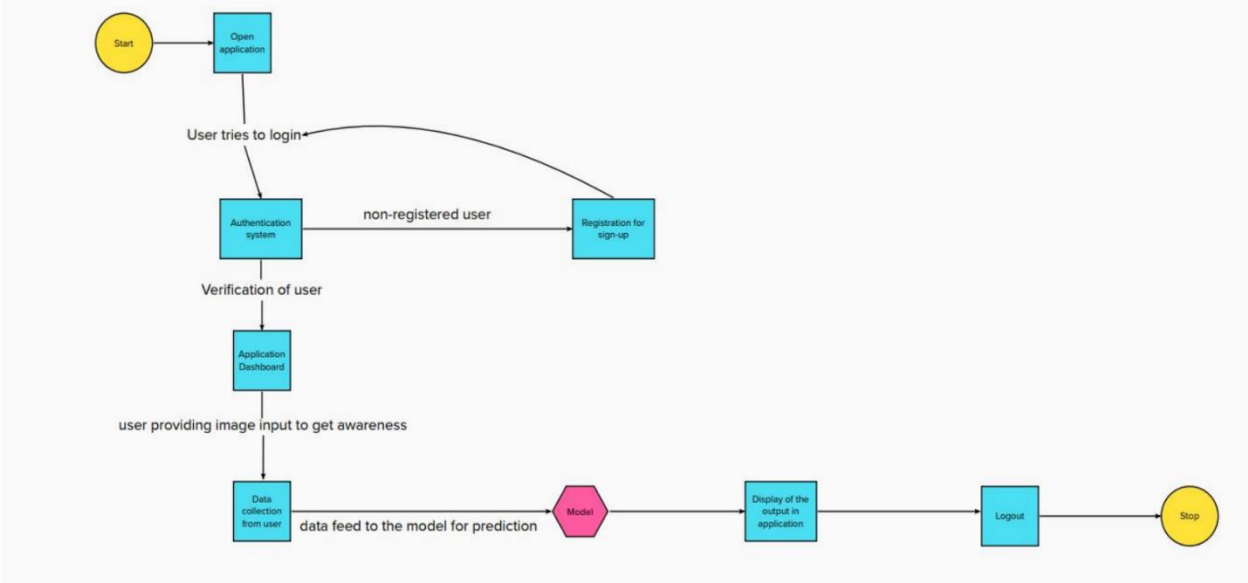
6.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Our solution is demanded for scientific researchers Such as Ornithologists , Zoologists in order to predict and analyse about flora and fauna.
NFR-2	**Security**	Authentication process involves multilayer security to make user data and collected data more secured, also to avoid unknown authorization and data integrity issues. Most security methods include Encryption and Authorization.
NFR-3	**Reliability**	Our framework should be reliable to cover wide range of species spanning across various habitats.
NFR-4	**Performance**	Data Augmentation to increase dataset size along with transfer learning to increase accuracy and performance for better working of application.
NFR-5	**Availability**	Our application possess full-time service (either offline or online) and dataset is constantly updated.
NFR-6	**Scalability**	Our application supports large number of concurrent users without any hurdles or errors through scaled cloud resources.

7. PROJECT DESIGN

7.1 Data Flow Diagrams

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

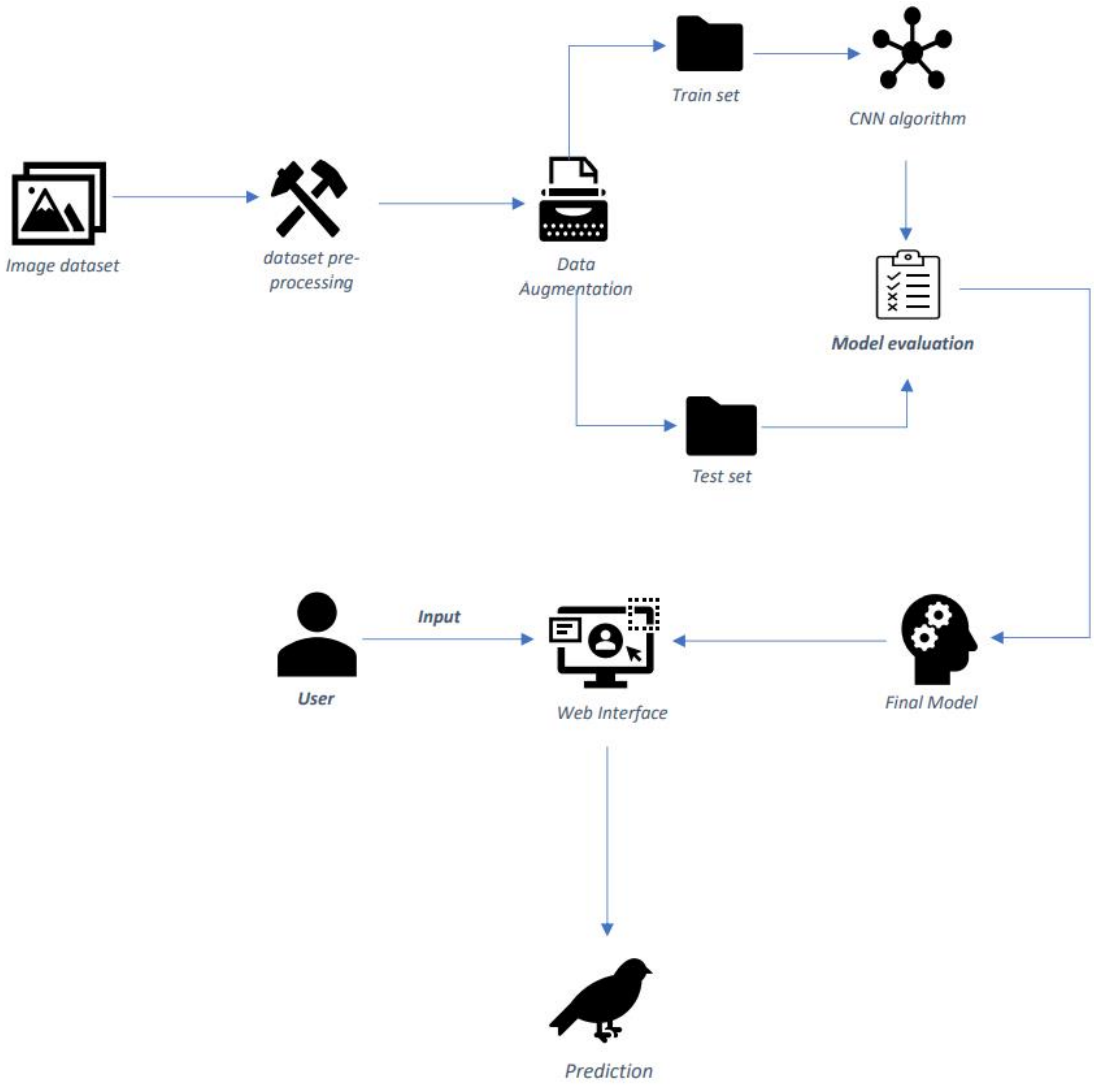


It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one.

7.2 Solution & Technical Architecture

Solution architecture provides the ground for software development projects by tailoring IT solutions to specific business needs and defining their functional requirements and stages of implementation. It is comprised of many subprocesses that draw guidance from various enterprise architecture viewpoints.

In solution architecture, the client needs are expanded to business needs that in one way or another are related to technology. These needs usually crystallize through re-assessing existing systems and finding out how they benefit or harm the organization in the long run. Solution architecture can be seen as a support system that provides structure and reduces the scope of complexity when developing and rolling out new systems and applications.



Technical Architecture (TA) is a form of IT architecture that is used to design computer systems. It involves the development of a technical blueprint with regard to the arrangement, interaction, and interdependence of all elements so that system-relevant requirements are met.

This digital transition required not only skilled developing teams but first and foremost IT architects. In their roles as IT strategists and planners, they map out a target architecture and make sure that all IT decisions align with business goals and requirements. This is largely due to the highly dynamic nature of IT, and its widespread adoption throughout all industries and businesses that have developed their own practices.

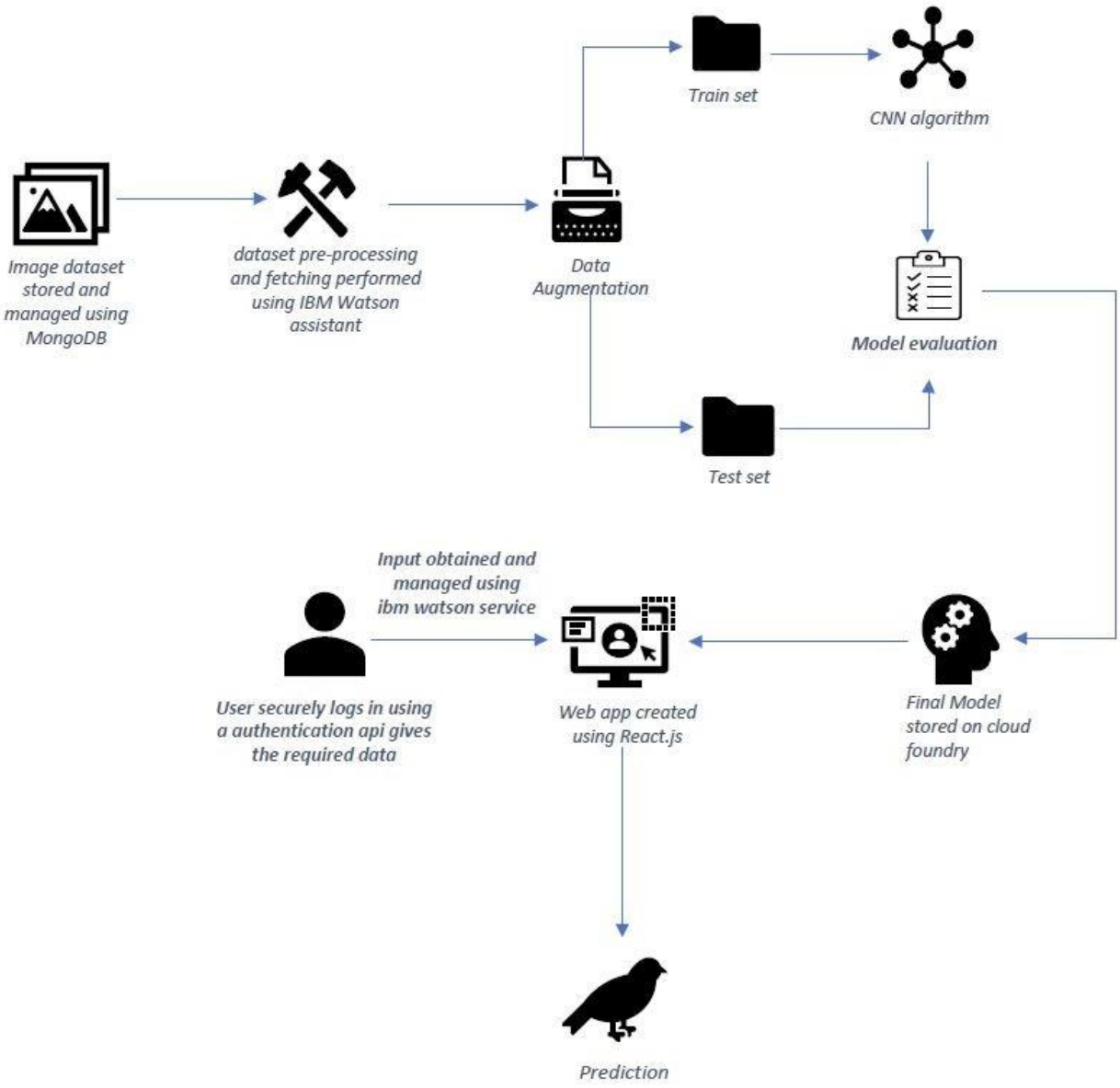

Train set
CNN algorithm
Image dataset stored and managed using MongoDB
dataset pre-processing and fetching performed using IBM Watson assistant
Data Augmentation
Model evaluation
Test set
Input obtained and managed using ibm watson service
User securely logs in using a authentication api gives the required data
Web app created using React.js
Final Model stored on cloud foundry
Prediction

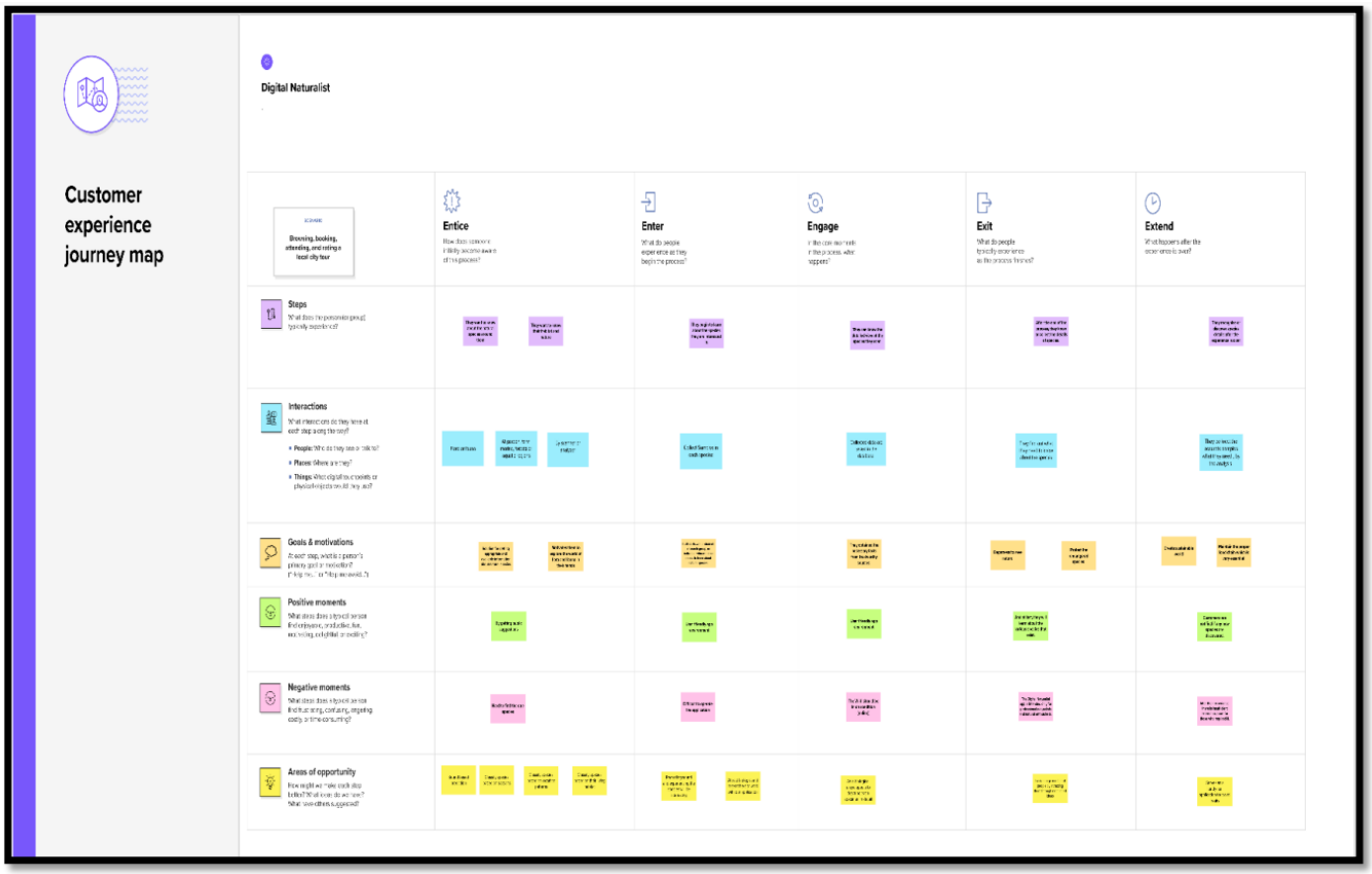
Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	Web UI or Website or Web app	HTML, CSS, React.js
2.	Application Logic-1	Model building and training	PY
3.	Application Logic-2	Getting image or text data from user for prediction	IBM Watson STT service
4.	Application Logic-3	Fetch the relevant data from the database andproject them to user	IBM Watson Assistant
5.	Database	Image and text data of all the species along withdetailed view of each species	NoSQL (MongoDB)
6.	Cloud Database	Fetch data from database and feed them to model for prediction and also used to retrieve the data required for user.	IBM Cloudant
7.	File Storage	Image data, login credentials, code (backendand frontend) and API keys	IBM Block Storage
8.	External API-1	To get data from the database when user givethe image input	IBM Storage API
9.	External API-2	To get the username and password of the specific user	Secure Authentication API
10.	Machine Learning Model	To predict the species (flora or fauna) through the image input and also it gives detailed view of the particular species	SDIM
11.	Infrastructure (Server / Cloud)	To deploy our application in cloud server	Cloud Foundry

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Application is built by using flask	WSGI fíamewoík (Web Seíveí Gateway Inteíface)
2.	Security Implementations	For authenticating the user data and protectingthe data about species in database	SHA-256 and Encryptions
3.	Scalable Architecture	To scale our application in server side by supporting clients including desktop browsers,mobile browsers etc	IBM Auto Scaling
4.	Availabil ity	To make application available both online andoffline and also 24/7 service.	IBM Cloud load balancer
5.	Performance	Designing an application that can handle wide range of requests at a time without any delay and to provide accuracy in pred	IBM instance

7.3: **Customer Journey Map**



8. PROJECT PLANNING & SCHEDULING

8.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Model Building Phase	USN-1	Collecting and digitalizing data for analysis	3	Medium	Vijaya Raghavan V
Sprint-1	Model Building Phase	USN-2	Data Augmentation and Feature Engineering	4	High	Ganesh K
Sprint-1	Model Building Phase	USN-3	Building the model using transfer learning approach	4	High	Kishore P
Sprint-1	Model Building Phase	USN-4	Evaluating the model to check the accuracy and precision	4	High	Vijaya Raghavan V
Sprint-1	Model Building Phase	USN-5	Class Prediction	3	Medium	Ganesh K
Sprint-2	Development Phase	USN-6	User database creation – contains the details of user	4	High	SudhakaranS
Sprint-2	Development Phase	USN-7	Web page Creation	4	High	Ganesh K
Sprint-2	Development Phase	USN-8	Login and register page creation - Login through email and password along with otp verification	3	Medium	Kishore P
Sprint-3	Development Phase	USN-9	Area to obtain user input	3	Medium	SudhakaranS
Sprint-3	Development Phase	USN-10	Model loading - API creation using flask.	4	High	Kishore P
Sprint-3	Development Phase	USN-11	Prediction page creation – shows prediction foruser input along with description about the species	2	Low	Ganesh K
Sprint-4	Deployment Phase	USN-12	Connecting the frontend and backend using API calls	4	High	Vijaya Raghavan V
Sprint-4	Deployment Phase	USN-13	Cloud deployment – Deployment of application using IBM cloud	4	High	Sudhakaran S
Sprint-4	Testing Phase	USN-14	Functional testing – Checking scalability and robustness of the application	3	Medium	Kishore P
Sprint-4	Testing Phase	USN-15	Nonfunctional testing Checking for user acceptance and integratio	3	Medium	Sudhakaran S

8.2 Sprint Delivery Schedule

Project Tracker, Velocity & Burndown Chart:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	18	6 Days	24 Oct 2022	29 Oct 2022	18	30 Oct 2022
Sprint-2	11	6 Days	31 Oct 2022	05 Nov 2022	11	5 Nov 2022
Sprint-3	9	6 Days	07 Nov 2022	12 Nov 2022	9	10 Nov 2022
Sprint-4	14	6 Days	14 Nov 2022	19 Nov 2022	14	15 Nov 2022

8.3 Reports from JIRA

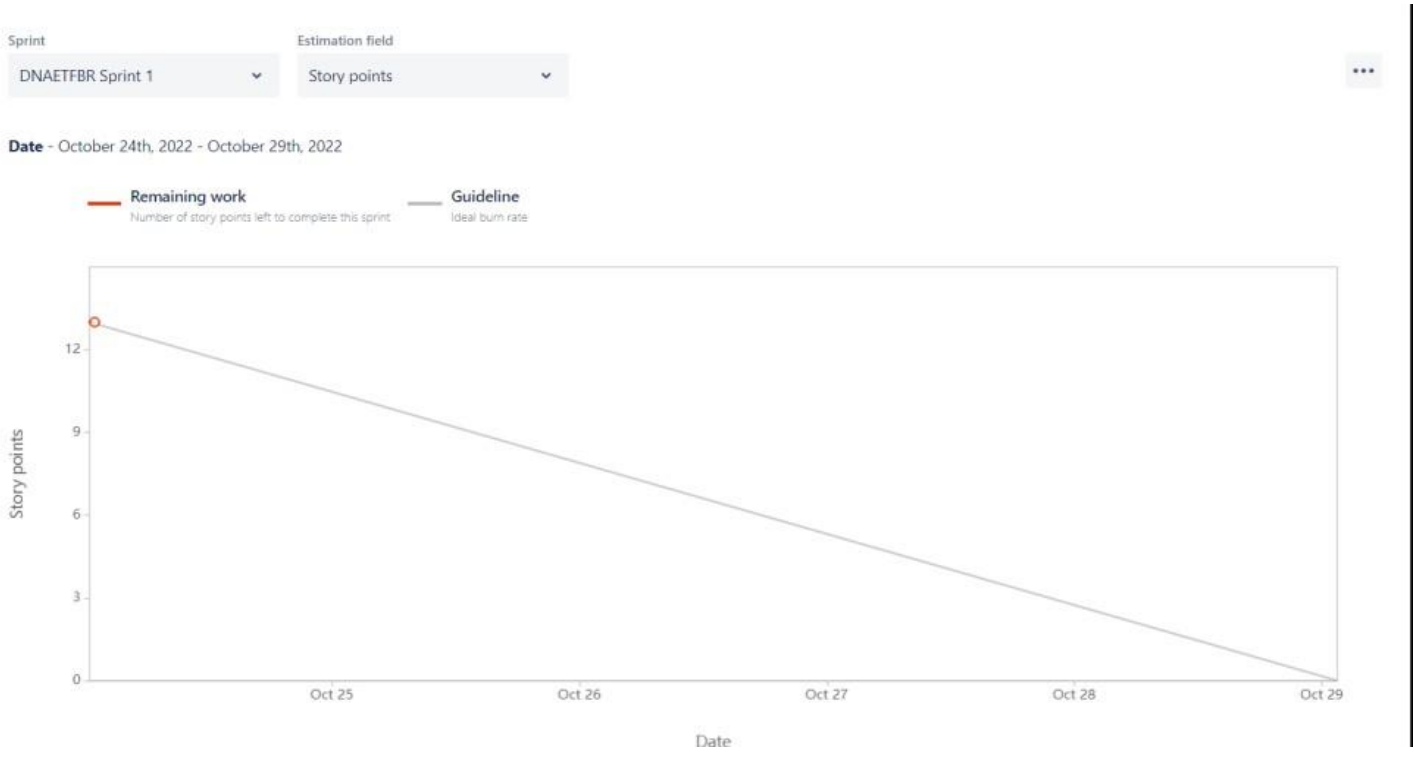
Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$\text{Average velocity} = 9/4 = 2.25$$

Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies suchas Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



9. CODING & SOLUTIONING (Explain the features added in the project along with code)

9.1 Feature 1

A CNN-based model which is trained up with the help of a pre-stored dataset of different species and performs with a high accuracy in predicting any new given restricted data to the model and the response/output from the model is delivered through a webpage for the user. Genuinely the model runs on a cloud platform called "IBM cloud" where the input files (i.e) dataset that are necessary for the model to predict properly are stored in the cloud as like the model itself. We used inception net pretrained network to train the model which helps in avoiding the overfitting issues and for efficient computation as well. It is then integrated with the flask application to allow the user to give input image-file to the model via a webpage in order to get knowledge about the species that they are looking for.

9.2 **Feature 2**:

A feature called upload option which is present in the webpage for the purpose of delivering the input image-file from the user to the model for the computation purpose of finding out what exactly the species is. This feature is linked up with a function from flask application whereby when a user clicks on this very upload button then the uploaded image-file is taken to the model where the image-file is stored locally and turned into an image array before the actual computation process begins and later sending back the response/output to the webpage for user's view.

10. TESTING

10.1 User Acceptance Testing

Introduction:

Effectively documenting incidents during the testing process is the key to improving software or processes before a system is released. Sometimes, the testers themselves document issues they encounter; but more often, a UAT coordinator verifies, consolidates, and classifies reported issues before assigning them to the appropriate groupto address. Then, that IT coordinator again validates and prioritizes the technical issues before handing them off to an IT developer to investigate further and resolve.

During the course of UAT, it is inevitable that issues will be discovered. It is shocking howoften documented issues contain insufficient data to facilitate a quick and thorough investigation.

Deliverables of UAT:

Every interviewer very quickly stated that UAT is to assure quality. Project managers also stated that it can double as a training exercise for business users as well as ensuring that the requirements set match the functionality that is desired from the system. People managers expressed thatone of the most important deliverables is the decision to go forward with the update or new system; the ”green” or ”red” light. Individual contributors expressed that UAT and inclusion goes hand in hand. That the testers feel included in the development and actually have a say in what works and what doesn't. Individual contributors stated that they felt that UAT has been done enough when the tests they are running are all success full but that it is a gut-feeling or intuition that says when they are content with the testing. They alsostated many perks of UAT such as: learning the new system, cooperation between departments, learning something new, feeling valued by the company and inclusion in decision making. Project Managers stated that the organization at large sometimes acted as though it had forgotten the purposeof UAT - to assure quality and usability of a release.

Data Mining:

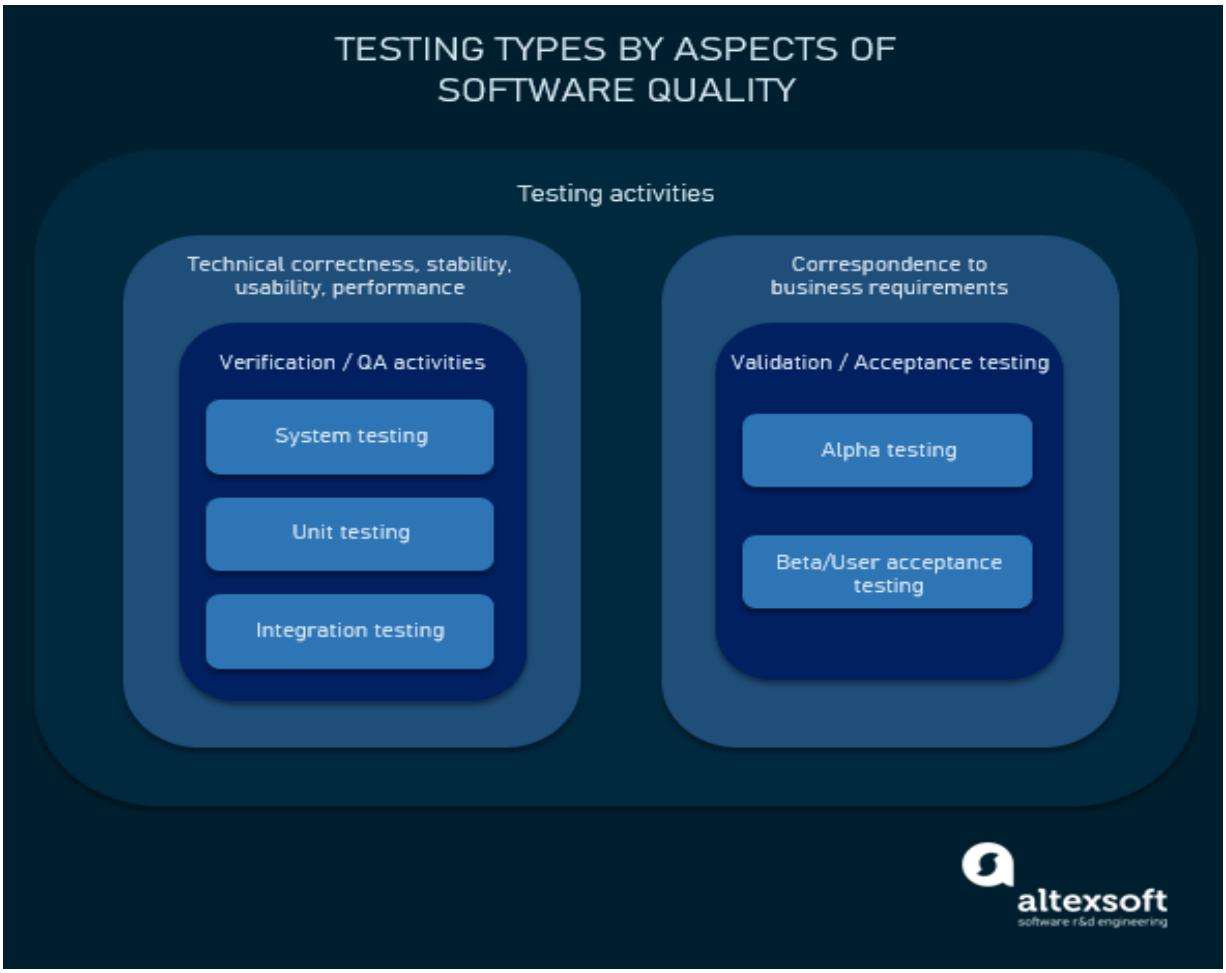
This section represents the actions pillar of the research. Here resultsbased on empirical insights from the system log files are presented. Results from the qualitative review of the testers use of test management tools are also presented in this section.

Time spent on Testing:

Because access was granted to the SUTs application logs it waspossible to track exactly how much time users spent on testing functionality inthe system. In the blow table 4.3.1 average times are detailed some of the users from the SUT

Test Quality:

From the production logs of the SUT a Markov-chain with 68 states(one for each application feature that was left after filtering out non-relevantstates) was created. Due to the fact that the SUT was a regular release of an existing system, and not a newly adopted software, a transition matrix could be made on a per-tester level for both the production system logs, as well as the test system logs. Variability due to changes in logging were taken into account by qualitatively examining the log files. As transition matrices for bothTEST and PROD had been computed, a similarity score could be computed to directly and in bulk estimate the quality of the testing.



Execution Flow

- C program (source code) is sent to preprocessor first.
- Expanded source code is sent to compiler which compiles the code andconverts it into assembly code.
- The assembly code is sent to assembler which assembles the code andconverts it into object code.

Usually, when possible, this testing happens in a conference or a war room sortof a set up where the users, PM, QA team representatives all sit together for a day or two and work through all the acceptance test cases.

Once all the tests are run and the results are in hand, the **Acceptance Decision** is made. This is also called the **Go/No-Go decision**. If the users aresatisfied it's a Go, or else it's a No-go. Reaching the acceptance decision is typically the end of this phase.

Conclusion:

UAT is not about the pages, fields or buttons. The underlying **assumption** even before this test begins is that all that basic stuff istested and is working fine. God forbid, the users find a bug as basic as that – itis a piece of very bad news for the QA team.

This testing is about the entity that is the primary element in the business.

```
class Program
{
    0 references
    static async Task Main(string[] args)
    {
        WriteLine("Please type the username for the desired user:");
        var username = ReadLine();

        var github = new GitHubClient(new ProductHeaderValue("MyAmazingApp"));

        try
        {
            var user = await github.User.Get(username);
            WriteLine($"The user {user.Name} was succesfully retrieved!");
            WriteLine($"{user.Name} has {user.PublicRepos} public repositories. Do you want to see the list? (y/n)");
            var response = ReadLine();

            if (string.Equals(
                "Y",
                response,
                StringComparison.InvariantCultureIgnoreCase))
            {
                var repos = await github.Repository.GetAllForUser(username);
                foreach (var repo in repos.OrderBy(x => x.CreatedAt))
                {
                    WriteLine($"{repo.CreatedAt:yyyy-MM-dd} | {repo.Name}");
                }
```

11. RESULTS

11.1 Performance Metrics

COLLECTION OF PERFORMANCE MEASUREMENTS

Managing application performance requires the continuous collection of data about all relevant parts of the system starting from the end user all the way through the system. This collected data is the basis for getting a holistic end-to-end and up-to-date view of the application state including the end-user experience. In this chapter, we will discuss what data to collect, and from where and how to collect the data in order to achieve this view Most application systems are implemented in a way that, in addition to the application logic executed at the provider's site

(Referred to as the back-end), parts of the application are executed at client'ssite. The client site usually constitutes a system tier accessing the back end

EXTRACTION OF PERFORMANCE-RELEVANT SYSTEM INFORMATION

The previous chapter focused on the collection of performance measurements from the relevant locations of the application system. This chapter focuses on the representation of higher the application system. While time series represent summary statistics (e.g., counts, percentile, etc.) over time, execution traces provide a detailed representation of the application-internal control flow that results from individual system requests.

From this data, architectural information, including logical and physical deployments and interactions (topology), can be extracted. For all cases, we will highlight examples and use cases in the context of APM level performance-relevant information about the system and their end-users that can be extracted from this data and that is used for APM visualization and reasoning, as detailed in the next chapters. Notably, we will focus on three commonly used representations, namely time series, execution traces, and augmented information about the architecture.

When depicting the number of users accessing a system, time series usually show a periodic pattern, e.g., based on the weekdays and the hours of the day. Other interesting patterns are spikes, for instance, indicating peaks in workload or hiccups.

EXECUTION TRACES

We concluded the previous section with the statement that time series are not suitable for analyzing individual requests. A data structure commonly used in APM for this purpose is an execution trace. Informally, an execution trace is a representation of the execution flow of a request through the system–ideally starting from the end user. As an example, Figure 3 depictsa schematic execution trace.

The execution trace starts with an operation called do Filter that is commonly found as an entry point in web-based applications. It can be observed that the execution of the do Filter operation includes a sequence of additional nested operation executions, until the list operation performs asequence of calls to a database.

12. **ADVANTAGES & DISADVANTAGES**

Advantages:

This system allows us to Identify and learn more about species automatically once an input is given. The input image is fed into a CNN which automatically analyses and produces an prediction. This project can be accessed from anywhere through the internet thus making our project portable.

Disadvantages:

The current web app is not appropriately scaled and hence won't be able to handle high traffic. Since the dataset used is not of wide variety, we will not be able to detect a wide variety of species.

13. **CONCLUSION**

In this project, we have deployed a website where we can upload an image of restricted set of species and the website will browse through thousands of images and will find every information it can regarding the being in the database.

14. **FUTURE SCOPE**

This application can be scaled widely to include a wide variety of species and also live detection systems placed in various places in areas where wildlife is widely present can be used to track and observe wildlife and help protect them.

15. APPENDIX

Source Code

diginature_app.py

```
from __future__ import print_function
from __future__ import division
import os

import numpy as np
import tensorflow as tf
from PIL import Image
from flask import Flask, redirect, render_template, request
from keras.applications.inception_v3 import preprocess_input
from keras.models import model_from_json, load_model
from werkzeug.utils import secure_filename
from keras.preprocessing import image

global graph
graph=tf.compat.v1.get_default_graph()
#this list is used to log the predictions in the server console
predictions = np.array(["Seneca White Deer",
                "Pangolin",
                "Lady's slipper orchid",
                "Corpse Flower",
                "Spoon Billed Sandpiper",
                "Great Indian Bustard"
                ])

#this list contains the link to the predicted species
found = np.array([
    "Seneca White Deer",
                "Pangolin",
                "Lady's slipper orchid",
                "Corpse Flower",
                "Spoon Billed Sandpiper",
                "Great Indian Bustard"
        ])
app = Flask(__name__)
model = load_model("model.h5")

@app.route('/', methods=['GET'])
def index():
     # Home Page
     return render_template("index.html")
```

```
@app.route('/predict', methods=['GET', 'POST'])
def upload():
  if request.method== 'GET':
    return("<h6 style=\"font-face:\"Courier New\";\">No GET request herd.....</h6
>")
  if request.method== 'POST':
    # fecting the uploaded image from the post request using the id 'uploadedimg'
    f = request.files['uploadedimg']
    basepath = os.path.dirname(__file__)
    #securing the  file by creating a path in local storage
    file_path = os.path.join(basepath, 'uploads', secure_filename(f.filename))
    #Saving the uploaded image locally
    f.save(file_path)
    #loading the locally saved image
    img = tf.keras.utils.load_img(file_path, target_size=(224, 224))
    #converting the loaded image to image array
    x = tf.keras.utils.img_to_array(img)
    x = preprocess_input(x)
    #converting the preprocessed image to numpy array
    inp = np.array([x])
    with graph.as_default():
      #loading the saved model from training
      json_file = open('DigitalNaturalist.json')
      loaded_model_json = json_file.read()
      json_file.close()
      loaded_model = model_from_json(loaded_model_json)
      #adding weights to the trained model
      loaded_model.load_weights("model.h5")
      #predecting the image
      preds = np.argmax(loaded_model.predict(inp),axis=1)

      #logs are printed to the console
      print("The predicted species is " , predictions[preds[0]])
  text = "The predicted species is " + found[preds[0]]
  return render_template("index.html", RESULT = text)


if __name__ == '__main__':
  #Threads enabled so multiple users can request simutaneously
  #debud is turned off, turn on during development to debug the errors
  #applications is binded to port 8000
  app.run(threaded = True,debug=True,port="8000")
```

Digtal_Naturalist_Model.ipynb

```
import os, gc, glob, random
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from PIL import Image
from tensorflow.keras.layers import Dense,Flatten
from tensorflow.keras.models import Model,model_from_json
from tensorflow.keras.applications.inception_v3 import InceptionV3, preprocess_input
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from os import listdir
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes
    your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='8gfltFjx9VEkeuxy-yjna6atnHeb6X6cPwVjRYFdARxc',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'digitalnaturalistcnn-donotdelete-pr-lqvznzucfzdltw'
object_key = 'Digital Naturalist Dataset.zip'

streaming_body_2 = cos_client.get_object(Bucket=bucket, Key=object_key)['Body']

# Your data file was loaded into a botocore.response.StreamingBody object.
# Please read the documentation of ibm_boto3 and pandas to learn more about the
    possibilities to load the data.
# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
# pandas documentation: http://pandas.pydata.org/
from io import BytesIO
import zipfile
unzip = zipfile.ZipFile(BytesIO(streaming_body_2.read()),'r')
file_paths = unzip.namelist()
for path in file_paths:
    unzip.extract(path)
import os
filenames = os.listdir('/home/wsuser/work/Digital Naturalist Dataset/Augmented
    Dataset')
dirName = '/home/wsuser/work/Digital Naturalist Dataset/Augmented Dataset'
folders = listdir(dirName)

def getListOfFiles(dirName):
    listOfFile = os.listdir(dirName)
    allFiles = list()
    for fol_name in listOfFile:
        fullPath = os.path.join(dirName,fol_name)
        allFiles.append(fullPath)
```

```
    return allFiles

Folders = getListOfFiles(dirName)
len(Folders)
subfolders = []
for num in range(len(Folders)):
    sub_fols = getListOfFiles(Folders[num])
    subfolders+=sub_fols
X_data=[]
Y_data=[]
id_no=0

found=[]

for paths in subfolders:
    files = glob.glob(paths + "/*.jpg")

    for myFile in files:
        img=Image.open(myFile)
        img=img.resize((224,224),Image.ANTIALIAS)
        img=np.array(img)
        if img.shape==(224,224,3):
            X_data.append(img)
            Y_data.append(id_no)
    id_no+=1
X=np.array(X_data)
Y=np.array(Y_data)

X=X.astype('float32')/255.0
y_cat=to_categorical(Y_data,len(subfolders))



X_train,X_test,y_train,y_test=train_test_split(X,y_cat,test_size=0.2)
print("The model has "+str(len(X_train))+" inputs")
#importing inceptionV3 a pretrained model
base_model = InceptionV3(input_shape=(224,224,3),include_top=False)
#setting the weights learnt by the pretrained model to false to halt learning
for layer in base_model.layers:
  layer.trainable = False
#flattening the output layer from the pretrained model
basemodel_output = Flatten()(base_model.output)
basemodel_output = Dense(units = 6, activation = 'sigmoid')(basemodel_output)
nn_model = Model(base_model.input,basemodel_output)
nn_model.compile(optimizer               =               'adam',loss               =
    keras.losses.binary_crossentropy,metrics=['accuracy'])
nn_model.summary()
# Creating a model checkpoint which monitors the accuracy of the model and saves the
    checkpoint
mc = ModelCheckpoint(filepath = "./model.h5",
                     monitor = 'accuracy',
                     verbose = 1,
                     save_best_only = True)

# Creating a earlystopping object which stop training once the model performance stops
    improving on a hold out validation dataset
es = EarlyStopping(monitor = "accuracy",
                   min_delta = 0.01,
                   verbose = 1)

call_back = [mc,es]
history      =      nn_model.fit(X_train,y_train,      steps_per_epoch=60,epochs      =
    30,callbacks=call_back,validation_data=(X_test,y_test))
# Exporting the model to json
```

```
model_json = nn_model.to_json()
with open("DigitalNaturalist.json", "w") as json_file:
    json_file.write(model_json)

# Exporting the model weights

nn_model.save_weights("DigitalNaturalist")
print("Saved model to disk")
#Model Evaluation
predictions = ["Corpse Flower",
               "Great Indian Bustard",
               "Lady's slipper orchid",
               "Pangolin",
               "Spoon Billed Sandpiper",
               "Seneca White Deer"
              ]


path       =       '/home/wsuser/work/Digital       Naturalist       Dataset/Augmented
    Dataset/Flowers/LS_Orchid_AUG/aug_download (1)_0_4894.jpg'
ime = tf.keras.utils.load_img(path,target_size=(224,224))

i = tf.keras.preprocessing.image.img_to_array(ime)
i = preprocess_input(i)
input = np.array([i])
pred = nn_model.predict(input)
plt.imshow(ime)
predictions[np.argmax(pred)]
```

GitHub & Project Demo Link

Github repo link: https://github.com/IBM-EPBL/IBM-Project-6013-1658822127.git

Project Demo Link: https://www.youtube.com/watch?v=O5F8cWbRBz0