Project Planning Phase
Technology Stack (Architecture & Stack)

Date	26 October 2023
Team ID	Team - 592719
Project Name	Project - Jungle Detectives: AI-Powered Image Classification of Wild Big Cats
Maximum Marks	4 Marks

Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

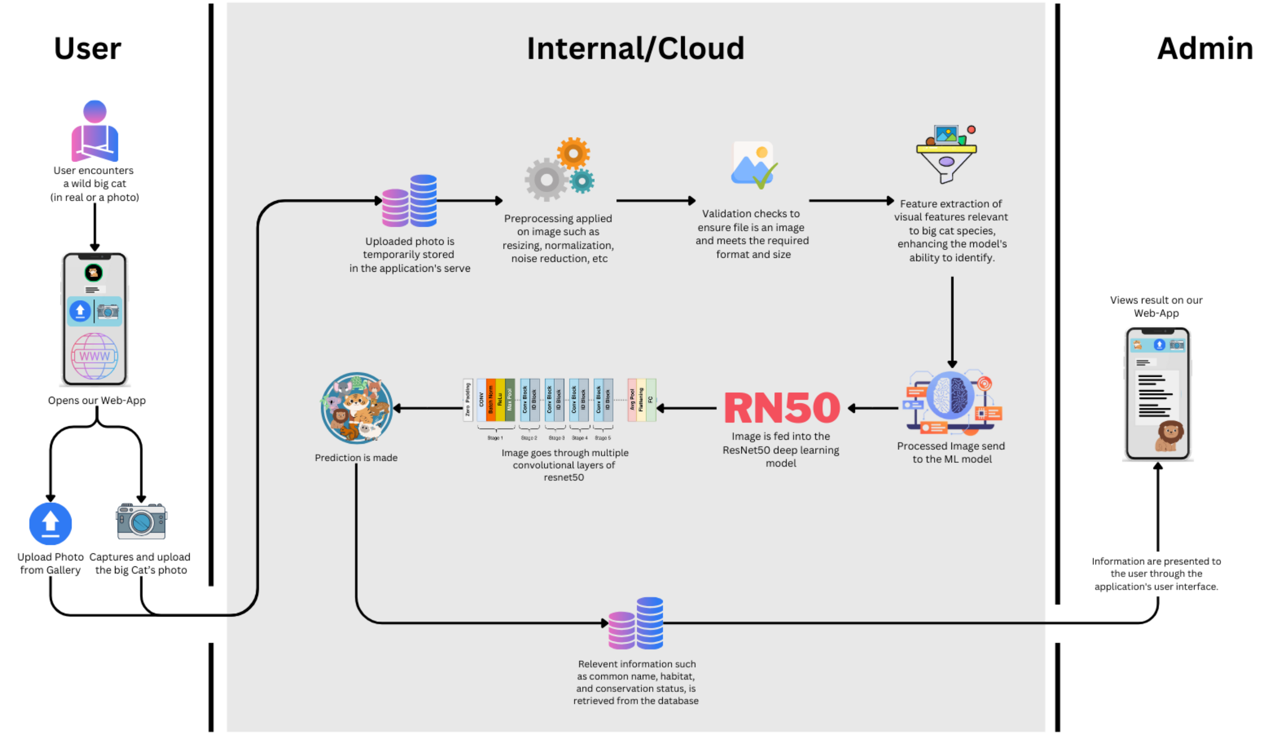
User
Internal/Cloud
Admin
User encounters a wild big cat (in real or a photo)
Opens our Web-App
Upload Photo from Gallery
Captures and upload the big Cat's photo
Uploaded photo is temporarily stored in the application's serve
Preprocessing applied on image such as resizing, normalization, noise reduction, etc
Validation checks to ensure file is an image and meets the required format and size
Feature extraction of visual features relevant to big cat species, enhancing the model's ability to identify.
Processed Image send to the ML model
RN50
Image is fed into the ResNet50 deep learning model
Image goes through multiple convolutional layers of resnet50
Prediction is made
Relevent information such as common name, habitat, and conservation status, is retrieved from the database
Information are presented to the user through the application's user interface.
Views result on our Web-App

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	The user login to Web-App and lands on the page where they get option to choose either to upload image by capturing directly from camera or import from gallery. After uploading and submitting the image they see information about the image like - species name, habitat, size, weight, type, etc.	HTML, CSS, JavaScript
2.	Application Logic-1 • Image Uploading	Through our frontend code we'll ask user to upload the image which will be send to be analysed by the model which is integrated with flask application.	Python Application, HTML
3.	Application Logic-2 • Pre-Processing of Image	To improve the image data to avoid unwilling distortions and enhances some image features important for further processing, along with resizing, normalization, noise reduction, geometric transformations of images like rotation, scaling, translation, etc.	Python Libraries (e.g.- ImageDataGenerator library, etc)
4.	Application Logic-3 • Image Identification	In our ML/DL model build on CNN, ResNet50 layers, the processed image goes though the convolution layers and the model then identify & classify the image and gives the name of the Big Cat as output for the flask application.	Python Libraries (e.g.- tensorflow, resnet50, etc)
5.	Application Logic-4 • Result Presented	After the Model predicts the Big Cat's Name, the identified big cat species, along with relevant information such as common name, habitat, and conservation status, is retrieved from the database, and displayed on the UI to the user.	MySQL, Python, HTML
6.	Database	Data Type- Image and Text	MySQL, Python etc.
7.	External API-1	Kaggle Dataset API is required for the training and testing of model on various images of Big Cats.	Kaggle, KDNuggets, etc

8.	Machine Learning Model	The aim of our Machine Learning Model is to preprocess and enhance the image, utilizing its convolution layers of the ResNet50 model to identify the processed image based on training with the provided dataset.	Image Recognition Model, ResNet50, etc.
9.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Deployment will be done on a free time bound hosting 3rd part server.	Localhost, Deployment

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Tensorflow PyTorch Keras	**TensorFlow** offers core components, TensorBoard for visualization, and TensorFlow Lite for mobile deployment. **PyTorch** features TorchScript for seamless integration, TorchVision for computer vision tasks, and TorchServe for production-ready model deployment. **Keras** provides a high-level API for efficient model building and Keras Applications for pre-trained models like VGG16 and ResNet50 and easy experimentation for computer vision task.
2.	Security Implementations	The system employs robust security measures, including encrypted data transmission protocols and secure socket layer (SSL) certificates to safeguard user interactions. Access control mechanisms restrict unauthorized entry, ensuring	e.g. Encryptions, RBAC, SSL etc.

		only authenticated users access sensitive data. Firewalls are deployed to filter incoming traffic and prevent unauthorized access to the server. Additionally, regular security audits, intrusion detection systems, and multi-factor authentication are enforced to maintain a secure environment, protecting user information and system integrity.	
3.	Scalable Architecture	**Three-Tier Architecture:** • **Presentation Tier**: Easily scalable front-end with horizontal scaling using web servers and load balancers for efficient user request handling without impacting core functionality. • **Application Tier**: Independent scaling of application servers and business logic, allowing addition of servers for distributed processing, ensuring responsiveness and high availability. • **Data Tier**: Vertical or horizontal scaling of database layer, maintaining data availability and read/write operations through hardware upgrades, sharding, or clustering as the dataset grows. **Microservices Architecture:** Fine-Grained Scalability: Microservices scale individually, meeting varied demands for tasks like image processing, model inference, and database operations. Agile Development: Enables seamless addition of new services, enhancing capabilities without disrupting existing ones, ensuring incremental, reliable growth.	HTML, CSS, JS SQL Server, Oracal MySQL
S.No	**Characteristics**	**Description**	**Technology**
4.	Availability	The application ensures high availability through strategic measures. Load balancers distribute user requests across multiple servers, preventing	**Load Balancers**: HAProxy balance incoming traffic across multiple servers,

		overload on any single server and ensuring continuous service even if one server fails. Distributed servers enhance fault tolerance; if one server experiences issues, others can seamlessly handle traffic. This setup minimizes downtime, enhances user experience, and guarantees uninterrupted access, vital for critical applications and user satisfaction	preventing overloads and ensuring consistent response times. **Distributed Servers:** Employing cloud platforms Azure is used for server deployment enhances fault tolerance, allowing seamless traffic handling across a network of geographically distributed servers.
5.	Performance	Designing for optimal performance involves managing high request volumes, utilizing caching mechanisms, and leveraging Content Delivery Networks (CDNs). Implementing efficient caching strategies reduces server load and speeds up response times. CDNs distribute content globally, reducing latency by serving data from servers closer to users, enhancing overall application speed and responsiveness.	Caching Mechanisms: Redis, Content Delivery Networks (CDNs): Cloudflare Load Balancers: HAProxy Platforms: Azure