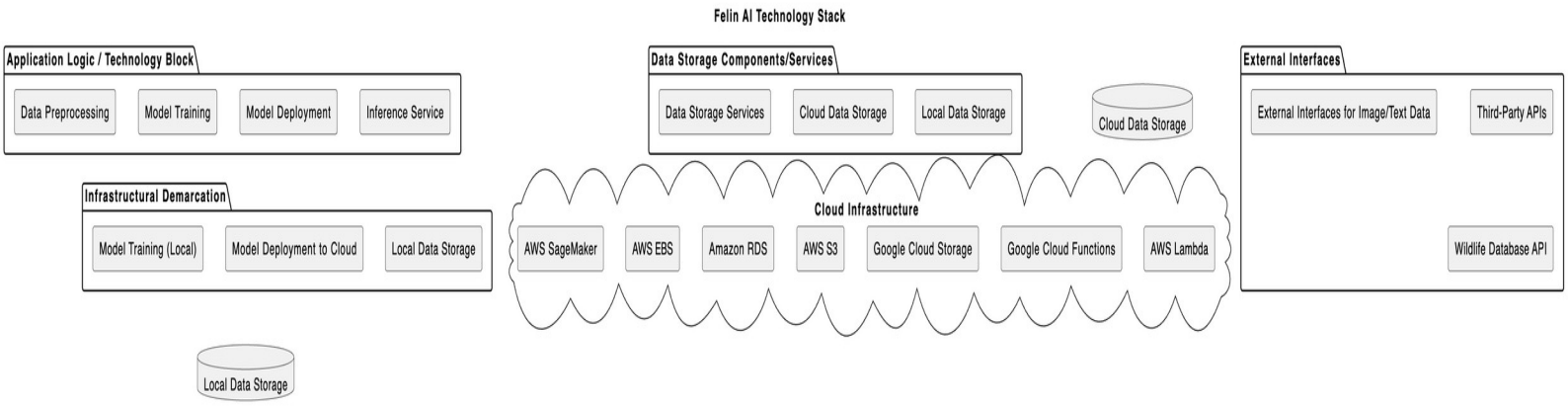
Project Planning Phase

Technology Stack

Date	28 October 2023
Team ID	Team-592716
Project Name	FELINAI: HARNESSING ARTIFICIAL INTELLIGENCE FOR FELIS TAXONOMY CLASSIFICATION
Maximum Marks	4 Marks

Technical Architecture:



Components and Technologies:

S.No	Component	Description	Technology
1.	Data Collection & Preprocessing	Collecting and preparing Felis taxonomy-related data for use in machine learning models.	Python (Pandas, NumPy), Web Scraping Tools, Data Cleaning Libraries.
2.	Feature Extraction	Extracting relevant features from the collected data for use as input to machine learning models	OpenCV (for image features), NLP Libraries (for text features).
3.	Machine Learning Models	Developing and training machine learning models for Felis taxonomy classification.	TensorFlow, PyTorch, Scikit-Learn.
4.	Model Deployment to Cloud	Deploying trained models to cloud services for scalability and accessibility.	AWS Lambda, Google Cloud Functions, Azure Functions.
5.	API and User Interface	Providing users with the ability to interact with the classification system through APIs and user interfaces.	RESTful APIs (Flask, FastAPI), HTML, CSS, JavaScript (for web UI).
6.	Post-processing and Visualization	Processing model outputs and creating visualizations for user understanding.	Matplotlib, Seaborn, Data Visualization Libraries.
7.	Feedback Loop	Collecting user feedback and data to improve the classification models.	Database (e.g., PostgreSQL, MongoDB), User Feedback Forms.
8.	Third-Party APIs	Integrating external APIs to access additional data sources and information.	API Integration Libraries, SDKs.
9.	Local Data Storage	Storing raw and preprocessed data locally during the early stages of development.	File Systems, Local Databases (SQLite).
10.	Cloud Data Storage	Using scalable cloud storage for large datasets and model checkpoints.	AWS S3, Google Cloud Storage

Application Characteristics:

S.no	Characteristics	Description	Technology
1.	Open Source Frameworks	Utilizing open source frameworks to leverage pre-built tools and libraries, promote collaboration, and reduce development time and costs.	Django, Flask, TensorFlow, PyTorch, and FastAPI for various components in the stack.
2.	Security Implementations	Implementing security measures to protect sensitive data, ensure data privacy, and prevent unauthorized access to the system.	HTTPS for secure communication, encryption algorithms, access control lists, and security audits.
3.	Scalable Architecture	Designing the architecture to be scalable, enabling the system to handle increased load and user interactions without compromising performance.	AWS Elastic Beanstalk, Kubernetes
4.	Availability	Ensuring high availability to minimize downtime and provide uninterrupted service to users.	AWS Availability Zones, redundancy in database systems, and auto-scaling to maintain system availability.
5.	Performance	Optimizing the system for efficient and responsive performance to meet user expectations and response time requirements.	Performance optimization can involve caching mechanisms (e.g., Redis), profiling tools for code optimization, and load testing for identifying bottlenecks.