AudiEyes

Ayoub Maimmadi

# 1. Project Inception

## 1.1. Framing the Business Idea as an ML Problem

* Business case description

Audieyes is a groundbreaking project aimed at enhancing the independence and safety of visually impaired individuals through advanced machine learning technologies. The service utilizes image and video captioning to provide real-time, detailed descriptions of environments and objects, improving navigation and accessibility. This addresses a critical need within the underserved assistive technology market, offering both significant social impact and commercial potential. By focusing on inclusivity and technological innovation, Audieyes not only opens new markets for businesses but also profoundly enhances the quality of life for its users. For more details click [here](https://github.com/AyoubMaimmadi/Audieyes?tab=readme-ov-file#business-case).

* Business value of using ML

Utilizing machine learning in the Audieyes project enables precise and adaptive recognition of visual data, greatly enhancing the service's capability to assist visually impaired users. ML algorithms optimize the accuracy and speed of image and video captioning, ensuring real-time feedback that is critical for navigation and interaction. Moreover, continuous learning from user interactions and feedback refines the system's effectiveness, ensuring that the technology evolves to meet diverse user needs. This not only fosters user dependency and satisfaction but also positions Audieyes as an innovative leader in assistive technologies, potentially increasing market share and generating sustainable revenue streams. For more details click [here](https://github.com/AyoubMaimmadi/Audieyes?tab=readme-ov-file#business-value-of-using-ml).

* Data overview

Audieyes utilizes a rich dataset comprising image-text pairs from diverse sources such as COCO, flickr30k, vqa, and nlvr. These datasets provide a wide variety of visual scenarios and associated descriptions, essential for training the ML models to recognize and articulate the content of images accurately. The data includes everyday objects, people, scenes, and activities, ensuring comprehensive coverage and relevance to real-world situations faced by visually impaired users. This extensive data foundation enables the ML models to deliver precise and contextually appropriate captions, critical for the functionality of Audieyes. For More details click [here](https://github.com/AyoubMaimmadi/Audieyes?tab=readme-ov-file#data-sources).

|  |  |  |
| --- | --- | --- |
| Model checkpoints | Dataset trained on | Pretrained model |
| model\_base.pth | /export/share/junnan-li/VL\_pretrain/annotation/coco\_karpathy\_train.json | N/A |
| model\_base\_nlvr.pth | http://export/share/datasets/vision/NLVR2/ | https://storage.googleapis.com/sfr-vision-language-research/BLIP/models/model\_base\_nlvr.pth |
| model\_base\_retrieval\_coco.pth | Coco: http://export/share/datasets/vision/coco/images/ | https://storage.googleapis.com/sfr-vision-language-research/BLIP/models/model\_base\_retrieval\_coco.pth |
| model\_vqa.pth | http://export/share/datasets/vision/VQA/Images/mscoco/ | https://storage.googleapis.com/sfr-vision-language-research/BLIP/models/model\_base\_vqa\_capfilt\_large.pth |
| model\_base\_retrieval\_flickr.pth | Flickr: http://export/share/datasets/vision/flickr30k/ | https://storage.googleapis.com/sfr-vision-language-research/BLIP/models/model\_base\_retrieval\_flickr.pth |

A screenshot of a computer program

Description automatically generated

* Project archetype

Audieyes represents a hybrid of autonomous real-time systems and human-in-the-loop architectures. It combines the speed and scalability of automated machine learning for instant image and video captioning with the precision of human oversight to enhance accuracy and contextual relevance. This model ensures that the system can operate independently while still benefiting from periodic human intervention to refine outputs, making it highly effective for applications in dynamic and varied environments. The design caters specifically to the needs of the visually impaired, promoting independence through technology while maintaining the flexibility to adapt and improve with user feedback. For more details click [here](https://github.com/AyoubMaimmadi/Audieyes?tab=readme-ov-file#architectural-archetypes-for-vision-captioning).

## 1.2. Feasibility analysis

* + literature review

The development of "Audieyes" is underpinned by significant advancements in machine learning, particularly in the fields of image and video processing that cater to enhancing accessibility for visually impaired individuals. This literature review examines the relevant technologies and models that form the basis of the Audieyes project, focusing on their applicability and effectiveness in real-world scenarios.

1. **. Machine Learning Models for Image and Video Captioning**
   1. **. Salesforce's BLIP Model:**

The BLIP (Bootstrapped Language Image Pretraining) model from Salesforce is a cornerstone in the image captioning domain, known for its efficacy in generating accurate and contextually relevant descriptions from images. BLIP’s architecture is designed to leverage large-scale datasets combining images with textual annotations, which is ideal for the diverse inputs expected in Audieyes. This model's proven capabilities in visual question answering and image captioning provide a reliable foundation for developing Audieyes’ core functionalities.

* 1. **. Moondream:**

Moondream is a compact, open-source computer vision model characterized by its portability and efficiency. Its design allows it to run on a wide range of devices, making it particularly suitable for real-time applications in accessibility technologies. The model’s robust performance in various settings demonstrates its potential to be integrated into Audieyes, ensuring that the application is both scalable and adaptable across different hardware environments.

* 1. **. Hugging Face’s Vit-GPT2 Image Captioning:**

Hosted on the Hugging Face platform, the Vit-GPT2 model combines vision-transformer (ViT) and GPT-2 technologies to create a potent image-to-text system. This model excels in translating visual data into descriptive text, a key feature for Audieyes. The use of transformer architectures allows for nuanced understanding and generation of captions, which can significantly enhance the user experience for the visually impaired by providing detailed and accurate descriptions of their surroundings.

* + Model choice/ specification of a baseline

For the Audieyes project, we have chosen the BLIP (Bootstrapping Language-Image Pre-training) model as our baseline due to its comprehensive abilities in both vision-language understanding and generation, which are essential for real-time image and video captioning applications aimed at assisting visually impaired users.

**Key Features and Specifications of BLIP:**

**Architecture:** BLIP employs a Multimodal Mixture of Encoder-Decoder (MED) architecture, which is versatile in functioning as a unimodal encoder, an image-grounded text encoder, or an image-grounded text decoder. This flexibility allows it to adaptively handle various tasks, including image captioning and text retrieval, within the same framework.

**Pre-training Methodology:** BLIP uses a novel dataset bootstrapping technique, CapFilt, which involves generating synthetic captions for web images and filtering out noisy ones. This method improves the quality of training data, especially considering the prevalent noise in web-crawled datasets.

**Performance Metrics:**

**Image-text retrieval:** Improved recall@1 by +2.7% on average.

**Image captioning:** Enhanced CIDEr scores by +2.8%.

**Visual Question Answering (VQA):** Increased VQA score by +1.6%.

**Generalization:** BLIP demonstrates strong zero-shot generalization to video-language tasks, which could be beneficial for extending Audieyes to video captioning in the future.

**Advantages for Audieyes:**

**High Relevance and Accuracy:** BLIP's capability to generate contextually relevant captions directly supports the core functionality of Audieyes, enhancing the user's understanding of their surroundings.

**Scalability and Flexibility:** The model’s architecture and the CapFilt training approach allow for easy scalability and adaptation to various data environments, crucial for deployment across different geographic locations.

* + Metrics for business goal evaluation

To effectively measure the success and impact of the Audieyes project, it is essential to establish clear, quantifiable metrics that align with the business objectives and operational goals. The following metrics have been identified to evaluate the performance and business value of Audieyes:

**1. User Engagement Metrics:**

**. Daily Active Users (DAU) and Monthly Active Users (MAU):** These metrics will help track the usage frequency and retention of the application, providing insights into its acceptance and value to users.

**. Session Length:** Measures the average duration users interact with the app per session, indicating the application's utility and user reliance.

**2. Performance Metrics:**

**. Caption Accuracy:** Accuracy of the image and video captions generated by the BLIP model, assessed through human validation or comparison with benchmark datasets.

**. System Latency:** Time taken from image/video input to caption output, crucial for real-time performance. A lower latency ensures a seamless experience for users.

**3. Customer Satisfaction Metrics:**

**. Net Promoter Score (NPS):** This metric gauges user satisfaction and likelihood to recommend Audieyes to others, which is pivotal for organic growth in the consumer base.

**. User Feedback and Reviews:** Qualitative assessments from users, providing insights into the app's impact on their daily lives and areas needing improvement.

**4. Accessibility Impact Metrics:**

**. Tasks Completed:** The number of tasks users successfully complete with the aid of Audieyes, such as navigation in new environments or identification of objects.

**. Incidence of Accessibility Issues:** Tracking issues reported by users related to accessibility, helping to refine and enhance the app’s functionality.

**5. Economic Metrics:**

**. Customer Acquisition Cost (CAC):** The cost associated with acquiring a new customer, essential for evaluating the efficiency of marketing strategies.

**. Lifetime Value (LTV):** The total revenue expected from a typical customer over the life of their relationship with the app, indicating the long-term viability of the project.

# 2. ML Pipeline Development - From a Monolith to a Pipeline

## 2.1. Ensuring ML Pipeline Reproducibility (milestone 2, 15%)

* Project structure definition and modularity

The Audieyes project is structured into a modular architecture to enhance scalability, ease maintenance, and facilitate collaboration among development teams. This architecture is divided into several key layers: the Data Layer, responsible for data ingestion, preprocessing, and storage, ensuring that data remains consistent and accessible across the system; the Model Layer, which houses the machine learning models including the baseline BLIP model and any additional models for specific tasks; the Service Layer, which interfaces the model outputs with the application layer, handling API requests and responses efficiently; and the Application Layer, which consists of both the user-facing front-end interface and the back-end application logic that allows for user interactions and data visualization.

Each of these layers is designed as an independent module with well-defined interfaces for interactions with other modules. This modularity allows for individual components of the system to be updated or replaced without disrupting the entire system's functionality, which is crucial for implementing continuous integration and deployment practices effectively.

Furthermore, the modular design supports parallel development where different teams can work on separate modules without interference, thereby speeding up the development process and enhancing efficiency. It also simplifies testing and maintenance since modules can be individually tested and maintained, thus reducing complexity and minimizing the risk of introducing system-wide failures. Lastly, the approach promotes reusability, where common functionalities are abstracted into shared libraries or services that can be reused across different parts of the project or even in future projects.

Overall, this structured and modular approach not only ensures the robustness and reproducibility of the ML pipeline for Audieyes but also prepares the system for future expansions and adaptations, aligning with the project’s long-term vision and objectives.

* Code versioning

For the Audieyes project, we utilize GitHub and Docker Hub to ensure robust code versioning and management. GitHub serves as the central repository for all project code, including the application logic, machine learning models, and system configurations. This platform facilitates version control, allowing the team to track changes, revert to previous states, and manage branches for feature development and bug fixes efficiently. Collaborative features such as pull requests and code reviews help maintain code quality and consistency across the development lifecycle.

Additionally, Docker Hub is employed to manage the Docker images of the project. This integrates seamlessly with our development pipeline, enabling consistent environments from development through to production. By storing and versioning Docker images on Docker Hub, we ensure that any member of the team or the CI/CD pipeline can pull the exact versions of the environment needed to run the application, enhancing reproducibility and reducing "works on my machine" issues.

* Data versioning

For the Audieyes project, data versioning is managed through Data Version Control (DVC), which plays a crucial role in maintaining the integrity and reproducibility of the machine learning pipeline. DVC allows the team to track versions of data sets and models, ensuring that every experiment can be reproduced and every model training session can be traced back to its data source. This tool integrates seamlessly with existing version control systems like Git, but it manages large data files and machine learning models that Git cannot handle efficiently.

Using DVC, changes to datasets are tracked in a manner similar to source code, which enables the team to pinpoint the exact data version used for specific training runs. This capability is vital for debugging and improving models, as well as for team collaborations, where consistency in data usage needs to be maintained across different members and potentially remote environments. DVC also supports data storage in remote locations, providing flexibility in how and where data is accessed and stored, crucial for handling large-scale datasets typical in machine learning projects like Audieyes.

* Experiment tracking and model versioning

critical components in managing the machine learning lifecycle effectively. MLflow offers a centralized platform to monitor experiments, including tracking of parameters, metrics, and model artifacts across various stages of the ML pipeline. This functionality allows our team to compare different model versions objectively, facilitating the identification of the most effective configurations based on empirical data.

Model versioning in MLflow provides a systematic way to version and store models, ensuring that every iteration is cataloged and retrievable. This capability is crucial for rolling back to previous versions if needed and for auditing purposes, allowing us to maintain a detailed history of model development and changes over time.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

By integrating MLflow, the Audieyes project benefits from enhanced reproducibility and accountability in model development, enabling seamless transitions between model iterations and ensuring consistent results in deployment environments. This setup not only streamlines the development process but also supports robust testing and deployment strategies, crucial for delivering a reliable assistive technology solution.

* Setting up a meta store for metadata

In the Audieyes project, we've integrated MLflow to set up a meta store for efficiently managing metadata. MLflow's meta store centralizes and streamlines the storage of essential metadata such as model parameters, configurations, and experiment results. This setup not only enhances the reproducibility of our ML pipeline but also facilitates seamless access and analysis of metadata across the project's lifecycle. By using MLflow's meta store, we ensure robust management of metadata, enabling effective tracking of experiments and model versions, which is crucial for maintaining consistency and accelerating development cycles in our Audieyes system. For more details click [here](https://github.com/AyoubMaimmadi/Audieyes?tab=readme-ov-file#experiment--metadata-tracking-with-mlflow).

* Setting up the machine learning pipeline under an MlOps platform

In the Audieyes project, we have set up our machine learning pipeline using ZenML, an open-source MLOps framework that enhances the entire lifecycle of machine learning from data ingestion to model deployment. ZenML supports our goal of developing and testing the ML pipeline with a structured and scalable approach, emphasizing reproducibility, collaboration, and automation. Our pipeline includes key components such as data validation to ensure the integrity of input data, data ingestion from diverse JSON sources, and data splitting to segregate data into training and validation sets effectively. Additionally, we use ZenML for training our models using the BLIP architecture for image captioning and rigorously evaluate model performance on the validation set to pinpoint areas for improvement. To ensure our models perform reliably and meet quality standards, we integrate GitHub Actions for running behavioral model tests, which automatically validate changes in the codebase and maintain high standards of functionality and performance. This comprehensive setup under ZenML not only streamlines our ML pipeline but also embeds best practices of MLOps, ensuring that the Audieyes project achieves its technological and business objectives efficiently.

View all the details of the pipline [here](https://github.com/AyoubMaimmadi/Audieyes?tab=readme-ov-file#milestone-4-ml-pipeline-development-and-testing).

## 2.2. Pipeline Components

* + 1. Setup of data pipeline within the larger ML pipeline/ MLOps Platform
  + Data Validation and Verification

For the Audieyes project, an integral component of our ML pipeline setup within the MLOps platform is the robust process of Data Validation and Verification. We utilize the library Great Expectations, which plays a critical role in ensuring the integrity and quality of data flowing through our pipeline. This tool allows us to implement comprehensive data checks, including schema validation, data type verification, and constraint checks, which help prevent data corruption and inconsistencies that could adversely affect model training and performance.

Great Expectations enables us to automate the validation process, providing a structured framework to define and verify expectations for data quality. This setup helps maintain a high standard of data quality from the ingestion phase through to model training, making our ML pipeline more reliable and efficient. By embedding Great Expectations into our data pipeline, we ensure that all data meets the predefined criteria before it is processed, enhancing the reliability of our ML outputs and ensuring that the Audieyes project delivers accurate and impactful results to the end-users. View details [here](https://github.com/AyoubMaimmadi/Audieyes/blob/main/Audieyes/zemML/pipelines/validation/greate_expectation_validation.ipynb).

* + Feature Engineering store

In the Audieyes project, preprocessing and feature engineering play a critical role in preparing data for effective machine learning model performance. We employ Apache Cassandra as our feature store to manage and serve features due to its exceptional scalability, fault tolerance, and high availability, which are essential for handling extensive volumes of feature data. Our preprocessing steps involve transforming raw data into a structured format that is optimized for easy ingestion into Cassandra. This structured data is then meticulously engineered to extract meaningful features that significantly enhance the model's ability to learn and make accurate predictions. By integrating Apache Cassandra into our ML pipeline, we ensure seamless access to up-to-date feature data for continuous training and inference, thereby enhancing the overall efficiency and effectiveness of our AI models in the Audieyes project. View details here.

* + 1. Integration of model training and offline evaluation into the ML pipeline / MLOps Platform

I have used Automated tests in the pipeline to automatically run. View the file [here](https://github.com/AyoubMaimmadi/Audieyes/blob/main/.github/workflows/deployment.yaml).  
A screen shot of a computer code

Description automatically generated

* + 1. **Development of model behavioral tests**

I have used GitHub Actions for running behavioural model tests. These tests validate the functionality and performance of the image captioning model, ensuring that it meets the specified requirements and quality standards. The tests are executed automatically whenever changes are made to the model codebase, providing continuous feedback on the model's behavior and performance.

**1. Model checkpoints Test:** the test checks that the model checkpoints are generated correctly and contain the expected parameters and metadata. check the test [here](https://github.com/AyoubMaimmadi/Audieyes/blob/main/Audieyes/zemML/tests/checkpoints_tests.py).

**2. Business Metrics Test:** the test evaluates the model's performance against predefined business metrics, such as operational efficiency, revenue and more. check the test [here](https://github.com/AyoubMaimmadi/Audieyes/blob/main/Audieyes/zemML/tests/metrics_tests.py)

**3. Model Deployment**

For the deployment phase of the Audieyes project, we have strategically chosen Docker, Linode, and Kubernetes to ensure a robust and scalable deployment of our machine learning models. Docker containers are utilized to package the AI models along with their dependencies, ensuring consistency across different computing environments from development to production. This containerization facilitates the easy management of model deployments and versioning, allowing for quick updates and rollbacks without disrupting the entire system.

We deploy these Docker containers on Linode, a cloud hosting provider known for its simplicity and high performance. Linode enables us to scale our resources according to demand efficiently and ensures our application remains available with minimal downtime. This flexibility is particularly beneficial for handling the varying loads our service may experience as user numbers grow.

Kubernetes plays a crucial role in orchestrating these containers. It manages the deployment and scaling of our application, ensuring that we can seamlessly shift traffic between different model versions based on their performance. This orchestration is vital for maintaining high availability and performance, as Kubernetes can dynamically manage the application's load and distribute resources optimally across the containers.

Together, Docker, Linode, and Kubernetes form a powerful trio that underpins our model deployment strategy, ensuring that Audieyes delivers a reliable, scalable, and efficient service to assist the visually impaired community.

A screenshot of a computer

Description automatically generatedA black screen with white text

Description automatically generated

A screenshot of a person with curly hair

Description automatically generated

## 3.1 ML System Architecture

* Drawing with architecture highlights

A diagram of a computer

Description automatically generated

This architecture diagram outlines the workflow and interaction of various technologies in the deployment and operation of the Audieyes project. Here's a breakdown of each component and their interconnections:

1. React App: This is the front-end interface where users interact with the system. It sends HTTP requests to the deployed model and receives responses that are displayed to the user.
2. Deployed Model: This is the core AI model that processes incoming requests from the React app. It's responsible for executing the model's logic of image captioning.
3. Linode Server: Hosts the deployed model. It's the compute environment where the model runs and interacts with other components.
4. GitHub Actions: Used for continuous integration and deployment (CI/CD/CT). It automatically pulls the latest version of the model from the repository, builds it if necessary, and deploys it to the Linode server.
5. Kubernetes: Orchestration platform that manages the deployment of different model versions on the server. It ensures that the system scales efficiently and handles load balancing and auto-scaling.
6. Database: Stores and manages data, including user data, model inputs & outputs, and application logs. It's used by the system for persistent storage and retrieval of data needed for model validation and other processes.
7. Grafana: Monitors the model and system performance. It pulls metrics from the deployed model and other parts of the system to visualize performance and operational data.
8. Cassandra: Acts as the feature store. It stores and manages the features used by the machine learning models, enabling efficient feature retrieval for model training and inference.
9. MLflow: Manages the machine learning lifecycle, including experiment tracking, model versioning, and model storage. It saves trained models and interacts with other components to facilitate model deployment and monitoring.
10. Airflow: Automates and manages workflows, such as data preprocessing and feature engineering. It ensures that data is processed and made available to MLflow and Cassandra in the correct format.
11. Great Expectations: Validates and verifies data quality, ensuring that the data used throughout the system meets defined expectations before it's processed and used in model training.
12. DVC (Data Version Control): Manages and versions datasets, facilitating reproducibility of experiments and providing a mechanism to roll back to previous data versions if needed.

### Workflow:

* The React App sends a request to the Deployed Model for processing.
* The Deployed Model, hosted on a Linode Server and managed by Kubernetes, processes the request and sends back a response.
* The response data is stored in a Database and also sent back to the React App for display.
* Grafana monitors the system, using metrics from the model and other components.
* GitHub Actions deploys updates to the model automatically.
* MLflow manages the lifecycle of the model, saving new versions and tracking experiments.
* Cassandra stores features that are retrieved by Airflow during data processing tasks.
* Great Expectations ensures the data quality before it's processed and used.
* DVC manages the data versions used across the system, ensuring consistency and reproducibility.

## 3.2 Application development

* Model service development

In the Audieyes project, the development of the model service is crucial for integrating and efficiently serving our machine learning models within the application's architecture. This involves encapsulating the models within a service layer, designed to facilitate seamless interactions between the React frontend and the complex machine learning operations. The service layer effectively manages requests from the frontend, processes these through the deployed models, and ensures timely and accurate responses.

To enhance scalability and reliability, we employ Kubernetes to orchestrate the deployment and operation of these model services. Kubernetes allows us to manage multiple instances of the model service dynamically, handling load balancing and ensuring that the system scales effectively under different load conditions. Crucially, Kubernetes also plays a pivotal role in model performance management by monitoring various model versions in production and seamlessly routing traffic to the best-performing models based on predefined metrics. This capability not only improves the overall system performance but also enhances user satisfaction by providing the most accurate and efficient responses.

The integration of CI/CD practices using tools like GitHub Actions ensures that updates, whether they are new model versions or application updates, are consistently deployed to the Kubernetes-managed services without downtime. This robust setup forms the backbone of the Audieyes application development, ensuring it delivers high performance, reliability, and a seamless user experience. View Details [here](https://github.com/AyoubMaimmadi/Audieyes?tab=readme-ov-file#api-documentation).

* Front-end client development

The development of the front-end client is centered around React, a popular JavaScript library known for its efficiency and flexibility in building interactive user interfaces. We chose React because it enables the creation of a responsive and dynamic experience for users, essential for our application which requires real-time updates and interactive features. The React app is designed to be user-friendly, providing visually impaired users with an intuitive interface that allows easy interaction with the image and video captioning features. This front-end client communicates seamlessly with the back-end services through RESTful APIs, fetching data and displaying results efficiently. The use of React also facilitates the rapid development of new features and easy maintenance of the codebase, thanks to its component-based architecture and the wide ecosystem of development tools and libraries available. View Details [here](https://github.com/AyoubMaimmadi/Audieyes/blob/main/Audieyes/app/react_app/README.md).

## 3.3 Integration and Deployment

* Packaging and containerization

The integration and deployment are streamlined through the use of packaging and containerization. We utilize Docker, a leading containerization technology, to package our application and its dependencies into lightweight, portable containers. This approach ensures consistency across development, testing, and production environments, eliminating the "it works on my machine" problem. Docker containers can be easily deployed on any system that supports Docker, regardless of the underlying infrastructure, which simplifies deployment and scaling operations. This method not only enhances the reproducibility of our application but also aligns with modern DevOps practices, allowing for rapid, reliable, and continuous deployment cycles facilitated by Kubernetes, which orchestrates container deployment, scaling, and management. View details [here](https://github.com/AyoubMaimmadi/Audieyes/blob/main/Audieyes/docker/README.md).

* Integration with a CI/CD Pipeline

For the Audieyes project, integration with a CI/CD pipeline is implemented using GitHub Actions to automate the build, test, and deployment processes. This setup ensures that every pull request or push triggers a series of actions that maintain the code quality and deployment readiness of our application. The CI/CD pipeline is composed of multiple jobs, starting with linting and testing where the code is checked for style and functionality issues. Node.js and Python environments are set up for respective dependencies and testing routines.

Post-validation, the train\_evaluate\_deploy job kicks in, which is contingent upon successful linting and testing. This job handles the model training, evaluation, Docker image creation, and pushing the image to a Docker registry. Finally, the trained model is deployed to a Linode server using SSH, where it's pulled and run inside a Docker container to ensure consistent execution environments. This CI/CD pipeline not only automates the workflow but also ensures that each component of our application is tested and deployed systematically, reducing human error and speeding up production cycles. View details [here](https://github.com/AyoubMaimmadi/Audieyes/blob/main/.github/workflows/deployment.yaml).

* Hosting the application

For hosting the Audieyes application, we utilize Linode, a cloud hosting service known for its reliability and scalability. Linode provides a robust infrastructure that allows us to deploy our containerized application components, managed by Kubernetes, ensuring efficient load balancing, auto-scaling, and seamless deployment of updates. This setup guarantees high availability and performance, essential for delivering continuous service to our users. The choice of Linode as our hosting solution supports our needs for a flexible and cost-effective cloud environment, enabling us to maintain optimal service levels as user demand and system complexity grow. View details [here](https://github.com/AyoubMaimmadi/Audieyes?tab=readme-ov-file#3--ml-service-deployment-and-model-serving).

## 3.4. Model Serving and online testing

* Model serving runtime

Model serving and online testing are streamlined through a robust model serving runtime. This setup involves deploying our machine learning models within a Linode server environment that can handle requests in real time, ensuring that users receive immediate and accurate responses. The models are hosted on servers configured with the necessary computational resources to perform at scale, and they are accessible through a well-defined API that allows the React front-end to interact seamlessly with the model serving backend.

For online testing, we implement techniques such as A/B testing and canary releases, facilitated by our use of Kubernetes for orchestration. This approach allows us to continuously evaluate different versions of our models under real-world conditions, ensuring that only the best-performing models are deployed to all users. This dynamic testing and deployment strategy ensures that the Audieyes project remains at the forefront of technological reliability and accuracy, providing a dependable service to the visually impaired community.

* Serving mode (batch, on demand to a human, on demand to a machine)

The model serving approach is designed to cater to on-demand requests with a focus on delivering results directly to end-users, potentially with human intervention for verification or additional input when needed. This serving mode ensures that users receive real-time, accurate information with an option for human oversight to enhance reliability and trust in the system.

For the model serving runtime, we utilize a robust infrastructure that supports high availability and rapid response times. The deployment is managed through containerization with Docker, which allows the models to be served efficiently and scaled dynamically according to demand. This setup is orchestrated by Kubernetes, which manages the containers and ensures that the model serving runtime is always optimized for performance, handling load balancing and fault tolerance seamlessly. This combination provides a reliable and efficient environment for model serving, crucial for maintaining the responsiveness and accuracy required by the Audieyes application.

* Online testing (A/B Testing, Bandit)

Online testing, including A/B testing and Bandit algorithms, plays a crucial role in optimizing and ensuring the effectiveness of our machine learning models. We implement A/B testing to systematically compare different versions of our models or features using Kubernetes Services, directly measuring their impact on user experience and system performance. This approach helps us identify the most effective models or features based on real user interactions. Additionally, we use Bandit algorithms to dynamically adjust the traffic distribution among different model versions during the testing phase. This method not only accelerates the learning process about the best-performing models but also minimizes the risk and exposure from potentially less effective model variations. Both strategies are integrated seamlessly into our MLOps practices, allowing us to continually refine and enhance the Audieyes app.

# Monitoring and Continual Learning

## 4.1. Resource Monitoring

In the Audieyes project, resource monitoring is a vital component of our Monitoring and Continual Learning strategy, focusing on the efficient tracking and management of system resources to ensure optimal performance and reliability. We utilize Grafana and Prometheus for this purpose, where Prometheus continuously collects and stores metrics from various parts of our infrastructure, including server load, response times, and resource usage. Grafana then visualizes these metrics through dashboards that provide real-time insights into the system's health and performance.

This monitoring setup allows us to quickly identify and address potential bottlenecks or issues in the system, such as spikes in CPU usage or memory leaks, which could affect the user experience or system stability. By having detailed visibility into our resource utilization, we can make informed decisions about scaling and optimizing our application, ensuring that it remains responsive and stable under varying loads.

Moreover, the integration of monitoring tools into our Kubernetes-managed environment helps in automating some aspects of resource management, such as horizontal scaling based on predefined resource usage thresholds. This proactive approach to resource monitoring not only enhances system performance but also supports the continual learning aspect of our project by providing valuable data that can be used to refine and improve our models and infrastructure over time. View details [here](Milestone%207:%20Monitoring%20and%20continual%20learning).

## 4.2. Model Performance Monitoring or data distribution drift monitoring

Monitoring model performance and detecting data distribution drift are vital for ensuring the long-term reliability and accuracy of our machine learning models. To achieve this, we have implemented a comprehensive monitoring system using tools like Grafana and Prometheus. This system continuously tracks key performance indicators (KPIs) such as prediction accuracy, response times, and system throughput. Additionally, it monitors the distribution of incoming data to identify any significant drifts that might affect model performance.

By setting up alerts and thresholds within Grafana, we can proactively respond to any anomalies or performance degradations detected. This proactive monitoring allows us to maintain the system's integrity and effectiveness, ensuring that the models continue to perform well even as input data evolves over time. This approach not only helps in maintaining high standards of service for the end-users but also supports the ongoing improvement and tuning of models, keeping them aligned with the latest data trends and user needs. View details [here](Milestone%207:%20Monitoring%20and%20continual%20learning).

## 4.3. Continual Learning: CT/CD pipeline

We've implemented a Continual Learning (CL) pipeline integrated with Continuous Training (CT) and Continuous Deployment (CD) to ensure our models adapt to new data and evolving user needs effectively. This pipeline automates the process of retraining models on fresh data, evaluating their performance, and deploying improved versions without manual intervention. We utilize Apache Airflow to manage workflow automation, orchestrating tasks like data ingestion, preprocessing, model training, and validation. Once a new model version meets our performance criteria, it is automatically deployed to production using Kubernetes, which manages the rollout and ensures that there is no downtime or disruption to the service. This setup supports a dynamic learning environment where our models continually evolve and improve, maintaining high accuracy and relevance in real-world applications. View details [here](https://github.com/AyoubMaimmadi/Audieyes?tab=readme-ov-file#1--ml-system-architecture).

## 4.4. Pipeline orchestration

Pipeline orchestration is managed using Apache Airflow, which orchestrates complex workflows and ensures that each step of the machine learning pipeline, from data ingestion to model training and deployment, is executed in a timely and orderly manner. Airflow provides the flexibility to schedule and monitor workflows, handling dependencies and managing the execution of tasks across different stages of the pipeline. This automation not only streamlines operations but also enhances the reliability and efficiency of the entire system, allowing for scalable management of tasks and ensuring that resources are optimally utilized. By leveraging Airflow for pipeline orchestration, we ensure that the Audieyes project can maintain its operational efficiency and adapt quickly to new requirements as they arise.

# 5. Responsible AI

## 5.1 Evaluation Beyond Accuracy

* Audit Model for Bias

Recognizing the imperative of Responsible AI, we extend our evaluation beyond mere accuracy metrics to include comprehensive audits for bias in our models. This is crucial as the system is designed to assist a diverse group of visually impaired users, making it essential that the model's outputs do not favor one demographic over another.

To conduct effective bias audits, we employ a variety of techniques. First, we analyze the training data to ensure it represents a diverse range of scenarios and user groups, minimizing the risk of ingraining historical biases into the model. Next, we use statistical methods to detect any disparities in model performance across different groups defined by sensitive attributes like age, gender, and ethnicity.

We also implement fairness-aware machine learning algorithms that are designed to reduce bias by adjusting the model’s decision boundaries during training. To supplement these efforts, we engage with external audit teams who perform independent evaluations of the model to uncover any residual biases.

The findings from these audits inform continuous improvements in our training methodologies and model updates, ensuring that Audieyes remains a fair and equitable tool that enhances accessibility for all users, irrespective of their background. This commitment to auditing for bias underscores our dedication to upholding the principles of Responsible AI throughout the lifecycle of the project.

* Model Explainability and Interpretability

We prioritize model explainability and interpretability to ensure transparency and trustworthiness of our AI systems. This involves implementing techniques that make the inner workings of our machine learning models accessible and understandable to both technical and non-technical stakeholders. We utilize methods such as feature importance scores and model-agnostic tools like LIME (Local Interpretable Model-agnostic Explanations) and SHAP (SHapley Additive exPlanations) to elucidate how different features influence predictions. This approach not only helps in debugging and refining the model by identifying which features are most impactful but also aids in communicating the reasoning behind model decisions to end-users. Enhancing interpretability is crucial for maintaining user trust and for meeting regulatory and ethical standards, ensuring that our technology remains accountable and its decisions justifiable.

# 6. Conclusion

* Summary of Achievements
* Lessons Learned
* Future Directions

# References

# Appendices

# A screen shot of a computer Description automatically generatedA screen shot of a computer code Description automatically generatedA screenshot of a computer Description automatically generatedA screen shot of a computer code Description automatically generatedA screenshot of a computer Description automatically generatedA black screen with white text Description automatically generatedA screenshot of a computer Description automatically generatedA screenshot of a computer Description automatically generatedA screenshot of a computer Description automatically generatedA screenshot of a computer Description automatically generatedA screenshot of a computer Description automatically generatedA diagram of a computer Description automatically generated