Department of Computer Engineering

CENG350 Software Engineering

Software Requirements Specification for Koster Seafloor Observatory (KSO)

Group 90

By

Aneliya Abdimalik

Mukhammadrizo Maribjonov

Wednesday 9th July, 2025

Contents

List of Figures

List of Tables

Revision History

(Clause 9.2.1)

1. Introduction

1.1 Purpose of the System

The purpose of the Koster Seafloor Observatory (KSO) is to provide an open-source, modular software system for enhancing marine ecological research by efficiently analyzing large volumes of subsea movie data collected through autonomous technologies. The KSO's goal is to transform marine biodiversity monitoring process in Sweden's Kosterhavets National Park through 3 main objectives:

1. **Automated Video Processing:** The system systematically analyzes underwater footage gathered by Remotely Operated Vehicles (ROVs).Then replacing manual video review processes with scalable computations,which leads to increase in data processing efficiency.

2. **Citizen Science Integration:** The system engages global citizen scientists via Zooniverse platfor which allows volunteers to categorise footage. This is crowdsourced data validation approach and therefore requires an 80% consensus to make sure identification reliablity and improve the quality of ecological data.

3. **Machine Learning Deployment:** The KSO trains and uses algorithms for the automated extraction of species observations from subsea footage by utilizing advanced machine learning techniques, particularly YOLOv3 object detection models (with a demonstrated performance of F1=0.97 for target species). Additionally, the system provides an Application Programming Interface (API) to

establishes seamless access and utilization of these trained machine learning models.

1.2 Scope

The KSO system boundary:

Included:

- Video ingestion (MP4/MOV format, resolution $\geq$ 480p) and metadata extraction (GPS coordinates, timestamps).
- Citizen science classification interface, offering identification options for 27 marine species.
- YOLOv3 machine learning model training and evaluation (minimum mAP@0.5 $\geq$ 0.9).
- FastAPI prediction endpoints supporting configurable confidence thresholds.

Excluded:

- Real-time ROV telemetry processing.
- Direct integration with oceanographic sensors (e.g., temperature, salinity).
- Mobile application development (planned as a future extension).
- Development and physical deployment of autonomous underwater vehicles (AUVs) or ROVs.
- Physical maintenance of underwater recording equipment.
- Analysis of non-image-based data (e.g., water chemistry data).

The KSO software comprises several core modules:

- **Data Management Module:** It manages the storage, access, and processing of subsea movie data and metadata. Also it converts video segmentation to manageable clips to give optimized scientist review.
- **Citizen Science Module:** It uses the Zooniverse platform to ensure citizen scientists' annotation and classification activities due to supporting workflows for species identification through video clips and precise bounding box annotation. Human-generated annotations undergo a checking mechanism to ensure quality and reliability.
- **Machine Learning and High-Performance Computing Module:** It focuses on training, testing, and deploying YOLOv3 models to detect species automatically. It provides an FastAPI-based RESTful API for model predictions which allows researchers to extract ecological findings from new footage.

1.3 System Overview

1.3.1 System Perspective

The KSO operates as an integrated system interacting with several external entities:

- **External Entities:** ROV operators, citizen scientists, researchers
- **Data Flows:** Video uploads (100-500MB/min), classifications (5-10/sec), predictions

1.3.1.1 System Interfaces

- Inputs: ROV footage (H.264 encoded MP4, 1920×1080, 30fps)
- Outputs: Species detection reports (JSON), distribution maps (GeoJSON)

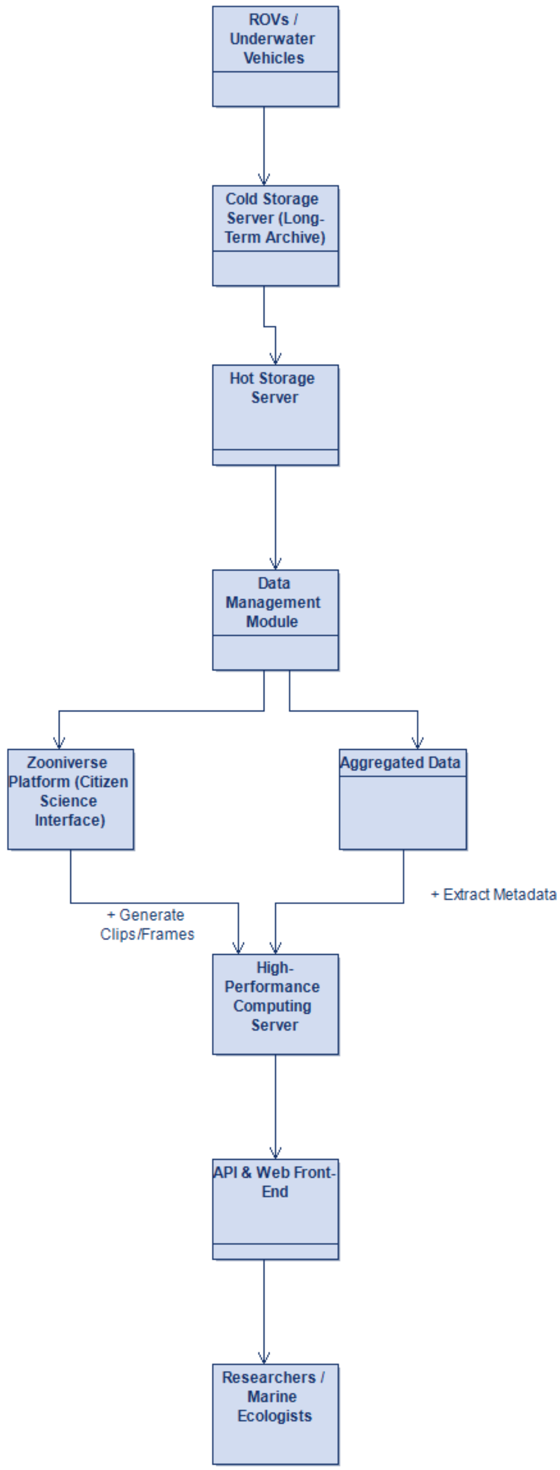

Figure 1.1: KSO System Context Diagram

1.3.1.2 User Interfaces

- Researchers: Python CLI for batch processing
- Citizen Scientists: Zooniverse web UI with tutorial videos

1.3.1.3 Hardware Interfaces

- Storage:
  - Cold Storage: 200TB NAS (Linux)
  - Hot Storage: 20TB RAID 10 SSD array
- Compute: NVIDIA GTX 2080Ti GPUs, dual 8-core Intel i9-9900 processors

1.3.1.4 Software Interfaces

- Zooniverse API: REST endpoints for clip management
- FastAPI: /predict endpoint with confidence threshold parameter

1.3.1.5 Communication Interfaces

- HTTPS/TLS 1.2 for external communications
- WebSocket for real-time training progress updates

1.3.1.6 Memory Constraints

- Minimum 16GB RAM for video processing
- GPU memory: 11GB GDDR6 (per GTX 2080Ti)

1.3.1.7 Operations

- Scheduled nightly backups (00:00 UTC)
- Model retraining triggered every 500 new annotations

1.3.2 System Functions

- Video Processing:
  - FFmpeg segmentation into 10-second clips
  - Metadata extraction via ExifTool
- Citizen Science Workflow:
  - Consensus algorithm (5 users per clip)
  - Anonymized data storage (GDPR compliant)
- Machine Learning Pipeline:
  - YOLOv3 model training (500 epochs)
  - Automated validation with mAP@0.5

1.3.3 Stakeholder Characteristics

Key stakeholders include

- **marine biologists**, who needs accuarte ecological data but has limited technical skills;
- **citizen scientists** who take benefit from clear instructions without engaging to variables;
- **IT personnel** who ensures system stability and operation in existing infrastructure.

1.3.4 Limitations

- Data Quality Constraints:
  - Analysis is limited to visible spectrum imagery, excluding ultraviolet (UV) and infrared (IR) ranges.

  - Minimum resolution of 480p is required, restricting identifying smaller or more mobile species.

- Model Performance Constraints:
  - Around 15% false-negative rate for species that are observed very rare.
  - Computational time is about 30 minutes per one hour of video, that can cause in delays when they have high data volumes.

- Operational and Storage Constraints:
  - Organizations or Governments can restrict data archiving functions.
  - Main data services have integration challenge that limits efficiency.
  - Short-term storage server capacity is limited, meaning that frequently accessed data must be managed effectively.

1.4 Definitions

- API: Application Programming Interface
- AUV: Autonomous Underwater Vehicle
- DwC (Darwin Core): Biodiversity informatics standard for sharing species occurrence data
- EBV: Essential Biodiversity Variable
- EMODnet: European Marine Data Archive
- ROV: Remotely Operated Vehicle (Sperre Subfighter 7500)
- SRS: Software Requirements Specification
- YOLO: You Only Look Once (object detection model)
- mAP@0.5: Mean Average Precision at 0.5 Intersection-over-Union (IoU) threshold, metric for evaluating object detection performance

2. References

- Aceves-Bueno et al. (2017). Bulletin of the Ecological Society of America
- Anton et al. (2021). Biodiversity Data Journal 9:e60548
- FFmpeg Developers (2022). FFmpeg 5.0 Documentation
- Germishuys et al. (2019). Koster ML GitHub Repository
- IEEE (2018). 29148-2018 - ISO/IEC/IEEE International Standard
- Redmon Farhadi (2018). arXiv:1804.02767

3. Specific Requirements

3.1 External Interfaces

Please refer to 3.1 for the diagram.

3.1.1 Zooniverse Platform Interface:

This interface connects the Koster Seafloor Observatory (KSO) system to the Zooniverse system so that video clips and image frames can be uploaded and classified, as well as annotated, by citizen scientists. It also gives the export of classification and annotation data provided by the citizen scientists with authentication, data transfer, and media and result synchronisation between the KSO system and Zooniverse.

3.1.2 High-Performance Computing (HPC) Server Interface:

This interface integrates the KSO software to the HPC server so that researchers can send machine learning training tasks, monitor progress, and download trained models. It facilitates transferring big data and model artifacts and allows for efficient computation for model training and inference.

3.1.3 Swedish National Data Archive Interface:

This interface enables the KSO system to save metadata of underwater films in the Swedish National Data Archive. It ensures that metadata are in archive-standard formats and ensures secure data transfer for long-term preservation and public access.

3.1.4 Application Programming Interface (API):

This interface provides researchers with access to the trained machine learning models via a web-based API. Built with FastAPI and a Streamlit user interface, the interface allows users to upload new video, make predictions, and adjust hyperparameters. The API also provides browsing pre-classified video and interactive model output exploration.

3.1.5 Database Server Interface:

This interface combines the KSO software with the SQLite database, managing the storage and retrieval of project data, including movies, sites, species, annotations, and model results. It manages data integrity, supports queries for data analysis, and enforces relationships between entities

3.1.6 Long-term Storage Server Interface:

This interface interconnects the KSO system to the cold storage server that holds raw underwater video. This interface allows software to access films on demand to be processed and analyzed, ensuring efficient access to large, infrequently accessed files.

3.1.7 Short-term Storage Server Interface:

This interface connects the hot storage server with the KSO software, storing frequently accessed movies ready for ongoing analysis. It enables the transfer of selected movies from cold storage to hot storage, such that access and processing are faster for ongoing research tasks.

3.1.8 Web Browser Interface:

This interface offers a user-friendly front-end for researchers and citizen scientists to interact with the KSO system. It makes access to the API and data visualization

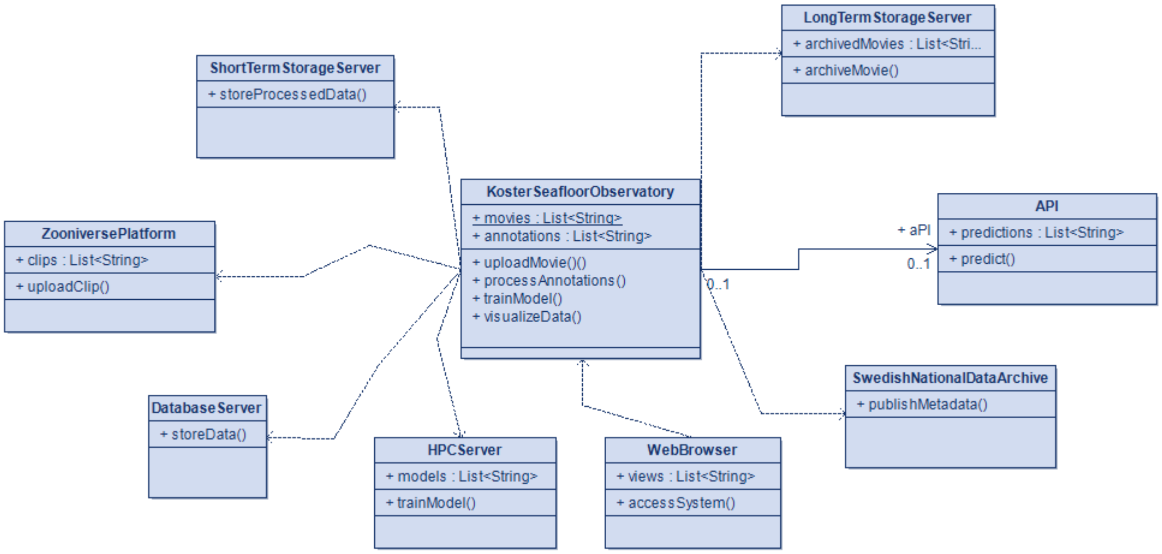

Figure 3.1: External Interfaces Class Diagram

software available to researchers. For citizen scientists, it interacts with Zooniverse for annotation and classification. It supports standard web browsers and has easy navigation.

3.2 Functions

Use case diagram is shown in Fig. 3.2 with their respective descriptions in Tables 3.1-3.13. Activity, Sequence, State diagrams are given in Figures 3.3, 3.4, 3.5 respectively.

3.3 Logical Database Requirements

Table descriptions of KSO Database 3.6.

- **Subject Table:** Stores filename, time intervals (for clips), frame numbers (for frames), workflow and subject set IDs, classification counts, retirement status, and creation timestamps. It also links to the associated movie via `movie_id`.
  Used when: The system uploads media to Zooniverse, tracks classification progress, or retrieves annotations for aggregation.

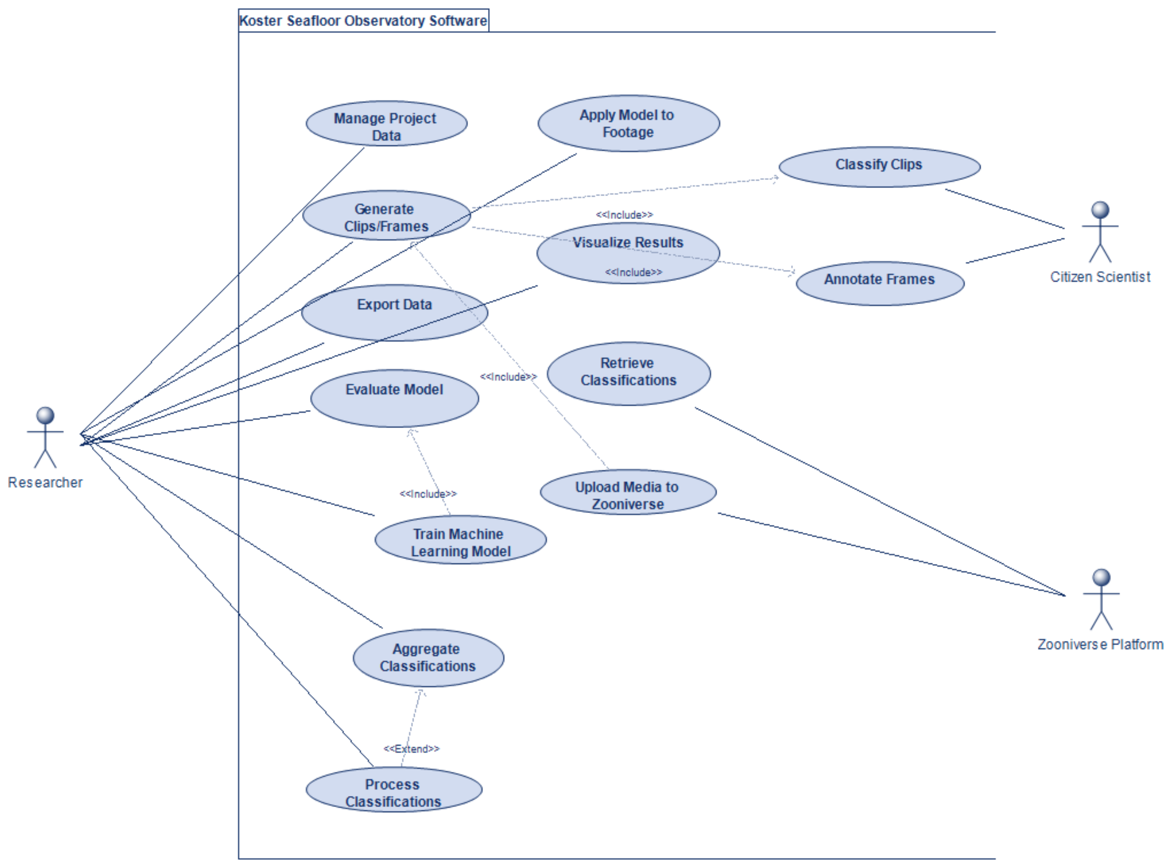

Figure 3.2: Use-Case diagram of KSO

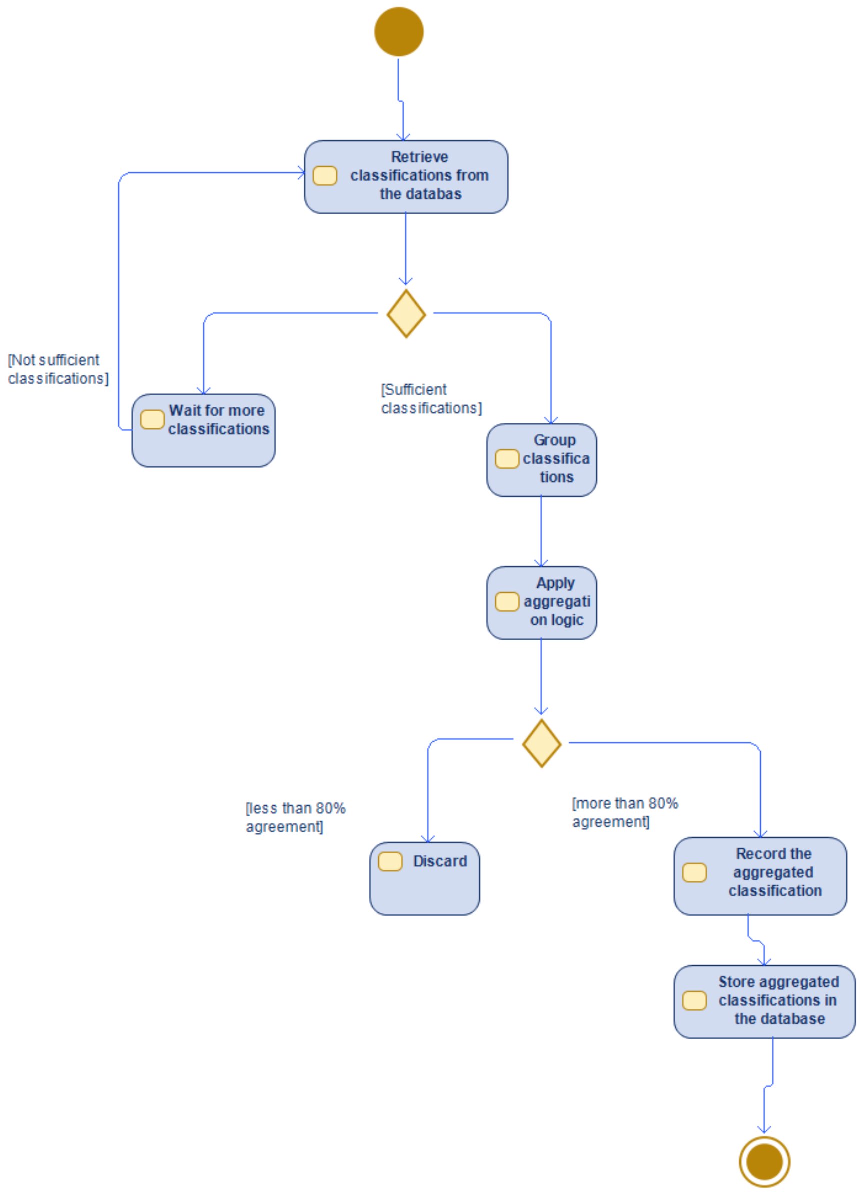

Figure 3.3: Activity Diagram for "Aggregate Classifications"

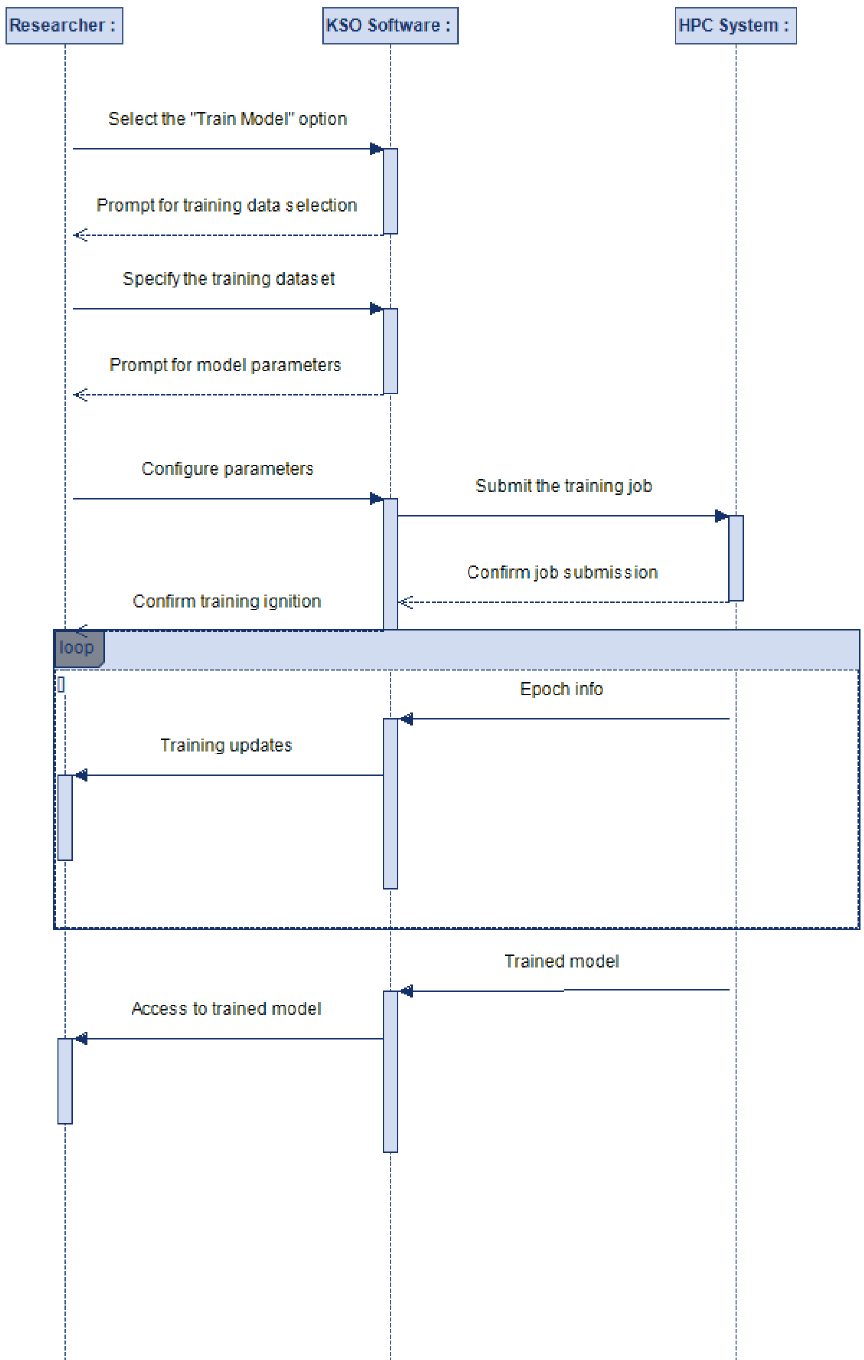

Figure 3.4: Sequence Diagram for "Train Machine Learning Model"

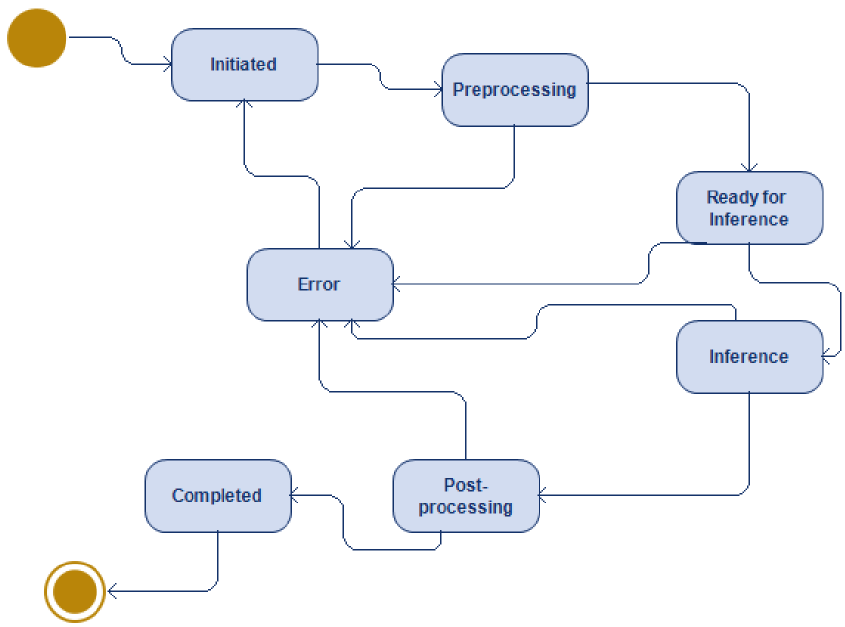

Figure 3.5: State Diagram for "Apply Model to Footage"

- **AggAnnotationsFrame Table:** Stores aggregated annotations for frames, including the species ID, position (`x`, `y`), dimensions (`width`, `height`), and the subject ID (linking to the frame).

  Used when: The system processes and aggregates frame annotations from citizen scientists, preparing data for machine learning training or analysis.

- **AggAnnotationsClip Table:** Stores aggregated annotations for clips, including the species ID, number of individuals observed (`how_many`), the time the species first appears (`first_seen`), and the subject ID (linking to the clip).

  Used when: The system processes and aggregates clip classifications from citizen scientists, preparing data for further analysis or model training.

- **ModelAnnotations Table:** Stores annotations generated by the machine learning model, including frame number, model version, species ID, movie ID, creation timestamp, and confidence level of the prediction.

  Used when: The system applies the trained model to new footage, storing predictions for analysis, visualization, or comparison with expert annotations.

Description of corresponding relations between tables.

- Movies to Sites: One site can have many movies (1 to many), linked by `site_id`.
- Movies to ModelAnnotations: One movie can have many annotations (1 to many), linked by `movie_id`.
- Movies to Subjects: One movie can have many subjects (1 to many), linked by `movie_id`.
- ModelAnnotations to Species: Many annotations can refer to one species (many to 1), linked by `species_id`.
- Subjects to AggAnnotationsFrame: One subject can have many frame annotations (1 to many), linked by `subject_id`.
- Subjects to AggAnnotationsClip: One subject can have many clip annotations (1 to many), linked by `subject_id`.
- AggAnnotationsFrame to Species: Many frame annotations can refer to one species (many to 1), linked by `species_id`.
- AggAnnotationsClip to Species: Many clip annotations can refer to one species (many to 1), linked by `species_id`.

3.4 System Quality Attributes

This section describes the quality attributes that are most important for KSO. In the order of priority, the attributes considered are Usability, Performance, and Dependability.

3.4.1 Usability

- **USR-01:** The system shall provide a clear, intuitive graphical user interface (GUI) with consistent navigation.

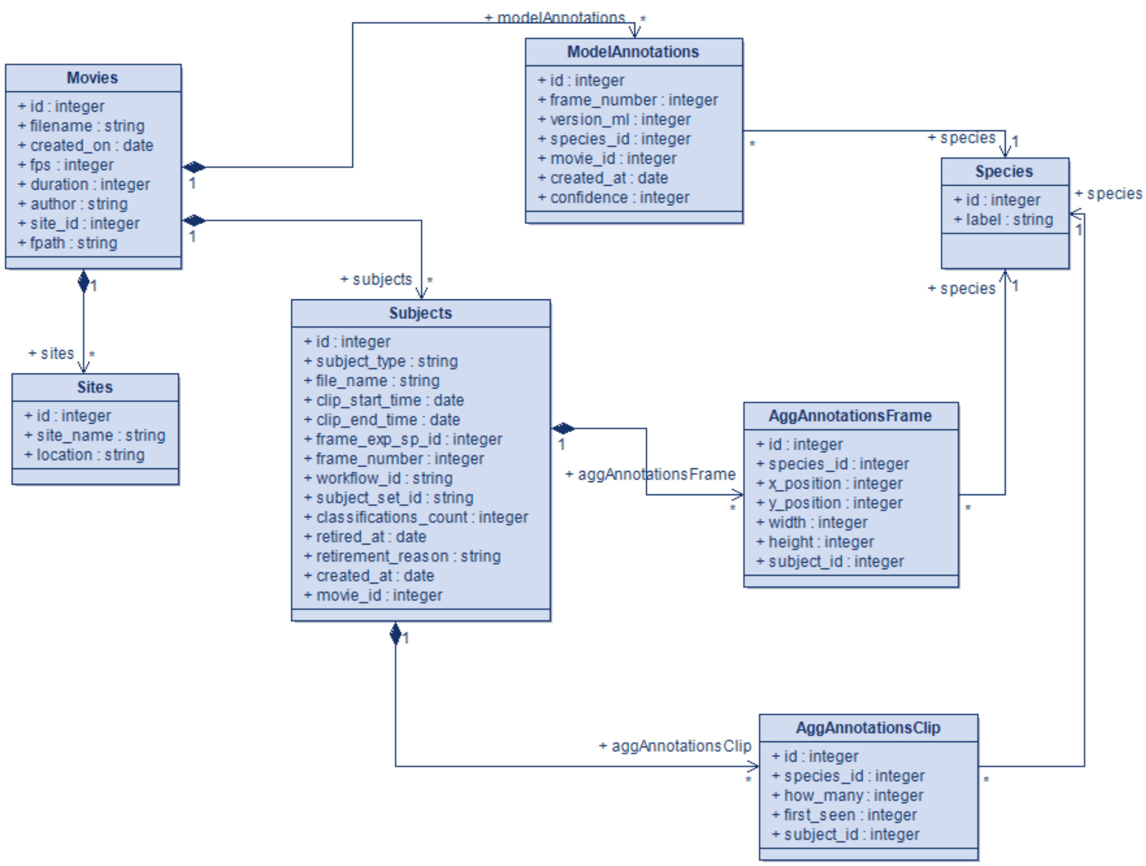

Figure 3.6: Logical Database Requirements Class Diagram

- **USR-02:** Interfaces shall be designed to minimize the learning curve for non-expert users, with context-sensitive help and tooltips.
- **USR-03:** User documentation and tutorials shall be available to assist new users.
- **USR-04:** The system shall offer real-time feedback on user actions, such as confirming uploads and providing error messages that clearly explain how to resolve issues.
- **USR-05:** Interactive visualizations (e.g., for machine learning model results) shall be easily understandable and modifiable by the user.

3.4.2 Performance

- **PER-01:** The system shall provide responses (e.g., loading of video clips, processing of annotations, model predictions) within a maximum delay of 3 seconds under normal operating conditions.
- **PER-02:** The system shall be capable of handling increasing amounts of data (e.g., high-definition movies and large numbers of citizen annotations) by employing efficient data management practices and scalable computing resources.
- **PER-03:** The system shall support a minimum of 50 concurrent users without degradation of performance, ensuring smooth interactions for both researchers and citizen scientists.

3.4.3 Dependability

- **DEP-01:** The system shall include mechanisms for error detection, logging, and recovery to ensure continuous operation even in the event of component failures.
- **DEP-02:** Automated backup procedures shall be in place to prevent data loss, especially for critical datasets like raw movies and citizen annotations.

- **DEP-03:** KSO shall have an uptime target of 99.5% during operational hours, with planned maintenance windows clearly communicated to users.

- **DEP-04:** The system architecture shall follow modular design principles to facilitate easy updates and integration of new functionalities (e.g., updated machine learning algorithms).

3.5 Design Constraints

- **CON-01:** The system must adhere to applicable national and international regulatory requirements for data privacy, security, and accessibility.

- **CON-02:** Compliance with standards such as the Web Content Accessibility Guidelines (WCAG 2.1) is mandatory to ensure usability for all users.

- **CON-03:** The system design must follow established protocols and best practices in software engineering and data management, ensuring interoperability with existing marine data repositories.

- **CON-04:** Budgetary and resource constraints may limit the selection of hardware and software components, necessitating the use of cost-effective and open-source solutions where possible.

- **CON-05:** The system must be designed to support the organizational workflows of the research teams and citizen science community, including compatibility with existing data archiving and analysis tools.

- **CON-06:** Management directives may impose specific timelines and milestones that influence the scope and scalability of the system.

- **CON-07:** The system must operate efficiently under varying network conditions and support remote access, given the distributed nature of the users (e.g., marine researchers and citizen scientists).

- **CON-08:** Limited on-site hardware resources, such as storage and processing power, require the system to be optimized for both performance and scalability.
- **CON-09:** The integration with external systems (e.g., data archives and high-performance computing platforms) is constrained by the available APIs and data formats.

3.6 Supporting Information

- KSO is an open-source marine data analysis system that leverages citizen science and machine learning to process and annotate underwater video data.
- In the event that video processing fails or outputs appear inconsistent, verify that the input files conform to the supported formats (e.g., MP4, MOV) and that the system's connectivity to both cold and hot storage servers is intact.
- If annotations from citizen scientists are not aggregating correctly, review the guidelines provided in the user documentation and ensure that the annotation thresholds are set as specified.
- The system is modular and designed for scalability. Should performance issues arise, check the integration of high-performance computing components (such as GPU acceleration) and review the backup and recovery procedures.
- Although there are no major hazards associated with operating KSO, proper training is essential. Users should be familiar with the data processing workflow and interpretation of machine learning outputs.
- A responsible administrator should oversee system initialization, updates, and ensure adherence to security protocols, including encrypted packaging of sensitive data.

Use Case Name	Manage Project Data
Actors	Researcher
Description	The researcher updates metadata related to the project, such as site information, movie details, and species lists.
Pre-Conditions	The project is initialized, and the researcher is logged in.
Data	Site metadata, movie metadata, species lists
Response	The system saves the updated metadata.
Stimulus	The researcher needs to update project information.
Normal Flow	1. The researcher selects the "Manage Project Data" option. 2. The researcher chooses which type of data to manage (sites, movies, species). 3. The researcher updates the relevant fields. 4. The researcher saves the changes.
Alternative Flow	The researcher might choose to import data from a file instead of manual entry.
Exception Flow	If the data entered is invalid (e.g., incorrect format), the system prompts the researcher to correct it.
Post-Conditions	The project data is updated in the database.
Comments	This use case is crucial for maintaining accurate project information.

Table 3.1: Tabular Description of *Manage Project Data*

Use Case Name	Generate Clips/Frames
Actors	Researcher
Description	The researcher creates video clips or image frames from the raw footage for uploading to Zooniverse.
Pre-Conditions	Raw footage is available, and the project is set up.
Data	Raw video files, parameters for clip generation (e.g., duration, frame rate)
Response	The system generates the clips or frames and saves them.
Stimulus	The researcher wants to prepare media for citizen science classification.
Normal Flow	1. The researcher selects the "Generate Clips/Frames" option. 2. The researcher chooses the raw footage to process. 3. The researcher sets parameters for clip or frame generation. 4. The system processes the footage and generates the clips or frames.
Alternative Flow	The researcher might choose to generate both clips and frames or apply specific filters.
Exception Flow	If the footage is corrupted or parameters are invalid, the system alerts the researcher.
Post-Conditions	Clips or frames are generated and ready for upload.
Comments	This step is essential for preparing data for citizen science involvement.

Table 3.2: Tabular Description of *Generate Clips/Frames*

Use Case Name	Upload Media to Zooniverse
Actors	Researcher, Zooniverse Platform
Description	The researcher uploads the generated clips or frames to the Zooniverse platform for classification by citizen scientists.
Pre-Conditions	Clips or frames are generated, and the project is linked to Zooniverse.
Data	Clips or frames, Zooniverse workflow settings
Response	The media is uploaded to Zooniverse.
Stimulus	The researcher wants to make the media available for classification.
Normal Flow	1. The researcher selects the "Upload Media to Zooniverse" option. 2. The researcher chooses the media to upload. 3. The researcher confirms the upload. 4. The system uploads the media to Zooniverse.
Alternative Flow	The researcher might select specific workflows or subjects.
Exception Flow	If the upload fails due to network issues or Zooniverse errors, the system retries or alerts the researcher.
Post-Conditions	Media is available on Zooniverse for classification.
Comments	This use case bridges the project with citizen science efforts.

Table 3.3: Tabular Description of *Upload Media to Zooniverse*

Use Case Name	Retrieve Classifications
Actors	Researcher, Zooniverse Platform
Description	The researcher downloads classification data from Zooniverse into the project software.
Pre-Conditions	Classifications are available on Zooniverse, and the project is linked.
Data	Classification data
Response	The system imports the classification data.
Stimulus	The researcher wants to collect classification results.
Normal Flow	1. The researcher selects the "Retrieve Classifications" option. 2. The researcher chooses the workflow or subject set. 3. The system downloads the classifications from Zooniverse.
Alternative Flow	The researcher might filter classifications by date or other criteria.
Exception Flow	If the connection to Zooniverse fails, the system alerts the researcher.
Post-Conditions	Classification data is stored in the project database.
Comments	This step is crucial for obtaining citizen science contributions.

Table 3.4: Tabular Description of *Retrieve Classifications*

Use Case Name	Process Classifications
Actors	Researcher
Description	The researcher prepares the retrieved classification data for aggregation, possibly cleaning or formatting the data.
Pre-Conditions	Classifications are retrieved and stored in the database.
Data	Raw classification data
Response	The system processes the data into a suitable format.
Stimulus	The researcher wants to prepare data for aggregation.
Normal Flow	1. The researcher selects the "Process Classifications" option. 2. The system applies processing steps (e.g., data cleaning, normalization). 3. The processed data is saved.
Alternative Flow	The researcher might manually adjust processing parameters.
Exception Flow	If the data is incomplete or corrupted, the system alerts the researcher.
Post-Conditions	Processed classification data is ready for aggregation.
Comments	This use case ensures data quality before aggregation.

Table 3.5: Tabular Description of *Process Classifications*

Use Case Name	Aggregate Classifications
Actors	Researcher
Description	The researcher combines classifications from multiple citizen scientists to produce consensus results.
Pre-Conditions	Processed classifications are available.
Data	Processed classification data, aggregation parameters
Response	The system generates aggregated classification results.
Stimulus	The researcher wants to obtain final classification outcomes.
Normal Flow	1. The researcher selects the "Aggregate Classifications" option. 2. The researcher sets aggregation parameters (e.g., agreement threshold). 3. The system runs the aggregation algorithm. 4. The aggregated results are saved.
Alternative Flow	The researcher might experiment with different aggregation methods.
Exception Flow	If there are insufficient classifications, the system alerts the researcher.
Post-Conditions	Aggregated classification data is available for further use.
Comments	This step is key for deriving reliable results from citizen science data.

Table 3.6: Tabular Description of *Aggregate Classifications*

Use Case Name	Train Machine Learning Model
Actors	Researcher
Description	The researcher uses aggregated classification data to train a machine learning model for automated classification.
Pre-Conditions	Aggregated classifications are available, and a baseline model is selected.
Data	Aggregated data, model training parameters
Response	The system trains the model and saves it.
Stimulus	The researcher wants to develop an automated classification tool.
Normal Flow	1. The researcher selects the "Train Model" option. 2. The researcher chooses the data and model type. 3. The researcher sets training parameters (e.g., epochs, batch size). 4. The system trains the model.
Alternative Flow	The researcher might perform cross-validation or hyperparameter tuning.
Exception Flow	If training fails due to data issues or computational errors, the system alerts the researcher.
Post-Conditions	A trained model is available for evaluation and use.
Comments	This use case leverages citizen science data for machine learning.

Table 3.7: Tabular Description of *Train Machine Learning Model*

Use Case Name	Evaluate Model
Actors	Researcher
Description	The researcher assesses the performance of the trained machine learning model using test data.
Pre-Conditions	A trained model and test data are available.
Data	Test dataset, evaluation metrics
Response	The system computes and displays evaluation results.
Stimulus	The researcher wants to check the model's accuracy.
Normal Flow	1. The researcher selects the "Evaluate Model" option. 2. The researcher chooses the test dataset. 3. The system runs the evaluation and displays metrics (e.g., precision, recall).
Alternative Flow	The researcher might compare multiple models.
Exception Flow	If the test data is incompatible, the system alerts the researcher.
Post-Conditions	Model performance metrics are available.
Comments	This step is crucial for validating the model's effectiveness.

Table 3.8: Tabular Description of *Evaluate Model*

Use Case Name	Apply Model to Footage
Actors	Researcher
Description	The researcher uses the trained model to classify new footage automatically.
Pre-Conditions	A trained model and new footage are available.
Data	New footage, model predictions
Response	The system generates classification results for the footage.
Stimulus	The researcher wants to automate classification of new data.
Normal Flow	1. The researcher selects the ”Apply Model” option. 2. The researcher chooses the footage and the model. 3. The system processes the footage and saves the predictions.
Alternative Flow	The researcher might adjust confidence thresholds.
Exception Flow	If the footage format is unsupported, the system alerts the researcher.
Post-Conditions	Classification results for the new footage are available.
Comments	This use case demonstrates the practical application of the model.

Table 3.9: Tabular Description of *Apply Model to Footage*

Use Case Name	Visualize Results
Actors	Researcher
Description	The researcher views graphical representations of the data and results, such as maps or charts.
Pre-Conditions	Data or results are available for visualization.
Data	Various data types (e.g., site maps, classification results)
Response	The system displays visualizations.
Stimulus	The researcher wants to explore or present the data visually.
Normal Flow	1. The researcher selects the ”Visualize Results” option. 2. The researcher chooses the type of visualization. 3. The system generates and displays the visualization.
Alternative Flow	The researcher might customize visualization parameters.
Exception Flow	If data is missing or incompatible, the system alerts the researcher.
Post-Conditions	Visualizations are displayed and can be saved or exported.
Comments	This use case aids in data analysis and communication.

Table 3.10: Tabular Description of *Visualize Results*

Use Case Name	Export Data
Actors	Researcher
Description	The researcher saves data or results in a specific format for sharing or further analysis.
Pre-Conditions	Data or results are available in the system.
Data	Export format, selected data
Response	The system generates and saves the exported file.
Stimulus	The researcher wants to share or analyze data outside the system.
Normal Flow	1. The researcher selects the "Export Data" option. 2. The researcher chooses the data to export and the format. 3. The system creates the export file.
Alternative Flow	The researcher might select specific subsets of data.
Exception Flow	If the export fails due to format issues, the system alerts the researcher.
Post-Conditions	The exported file is saved to the specified location.
Comments	This use case facilitates data sharing and integration with other tools.

Table 3.11: Tabular Description of *Export Data*

Use Case Name	Classify Clips
Actors	Citizen Scientist, Zooniverse Platform
Description	The citizen scientist views video clips on Zooniverse and provides classifications (e.g., identifying species).
Pre-Conditions	Clips are uploaded to Zooniverse, and the citizen scientist is logged in.
Data	Video clips, classification inputs
Response	The classification is submitted to Zooniverse.
Stimulus	The citizen scientist wants to contribute to the project.
Normal Flow	1. The citizen scientist logs into Zooniverse. 2. The citizen scientist selects the Koster Seafloor Observatory project. 3. The citizen scientist views a clip and enters classification data. 4. The citizen scientist submits the classification.
Alternative Flow	The citizen scientist might skip a clip if unsure.
Exception Flow	If a clip fails to load, the citizen scientist reports the issue.
Post-Conditions	The classification is recorded on Zooniverse.
Comments	This use case is external but critical for data collection.

Table 3.12: Tabular Description of *Classify Clips*

Use Case Name	Annotate Frames
Actors	Citizen Scientist, Zooniverse Platform
Description	The citizen scientist views image frames on Zooniverse and annotates them (e.g., marking species).
Pre-Conditions	Frames are uploaded to Zooniverse, and the citizen scientist is logged in.
Data	Image frames, annotation inputs
Response	The annotation is submitted to Zooniverse.
Stimulus	The citizen scientist wants to contribute to the project.
Normal Flow	1. The citizen scientist logs into Zooniverse. 2. The citizen scientist selects the Koster Seafloor Observatory project. 3. The citizen scientist views a frame and adds annotations. 4. The citizen scientist submits the annotation.
Alternative Flow	The citizen scientist might use different annotation tools.
Exception Flow	If a frame fails to load, the citizen scientist reports the issue.
Post-Conditions	The annotation is recorded on Zooniverse.
Comments	This use case is external but essential for detailed data collection.

Table 3.13: Tabular Description of *Annotate Frames*

4. Suggestions to Improve The Existing System

Areas for Suggested Improvements:

1. Establish Superior Protocols for Volunteer Data Reliability: Before using classifications from community scientists to train artificial intelligence, refined methodologies must be implemented to ensure the accuracy of these contributions.

2. Incorporate Intelligent Prioritization for Community Tagging Tasks: The system will use machine learning to figure out which parts of the data (like short video clips or pictures) are most important or confusing. It will then ask the volunteer researchers to classify those parts first.

3. Develop a More Comprehensive Analytical Web Portal: Create an deatiled user interface within the online application allowing researchers to visualize findings across space and time. Then compare the outputs of different models, and do statistical analyses.

4. Expand the Spectrum of AI Model Capabilities: Combine the ability to train and deploy many variety of machine learning models or to recognize a greater dataset.

4.1 System Perspective

Context Diagram: Updated KSO System Context Diagram 4.1.

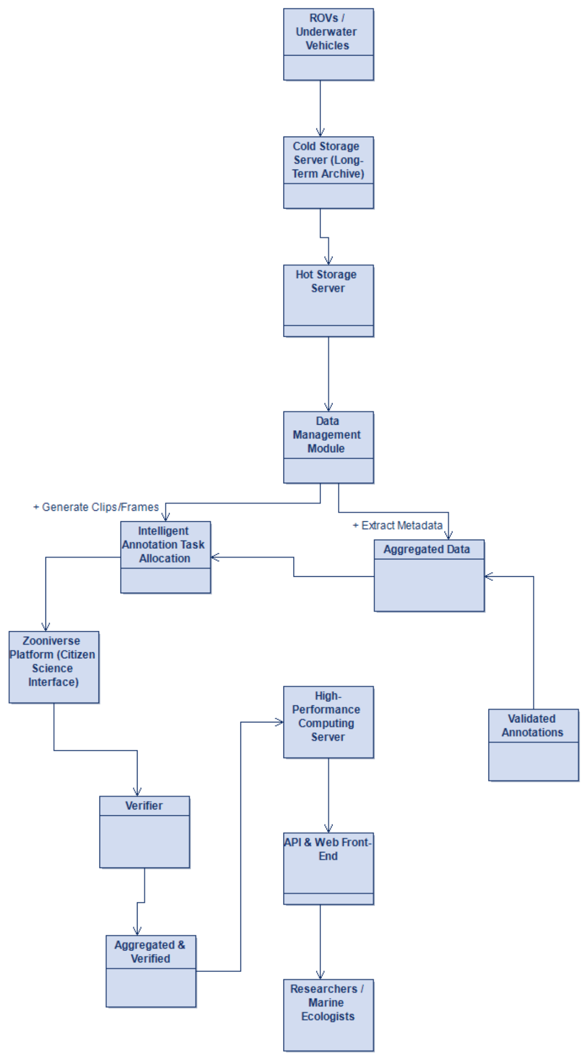

Figure 4.1: Updated KSO System Context Diagram

1. Broadened AI Algorithm Portfolio: The system will now complete the training and deployment of a wider array of machine intelligence algorithms like delineating objects in imagery (e.g., coral extent) and calculating biological mass.

2. Intelligent Annotation Task Allocation: A learning system will help the AI identify data segments that are either ambiguous or critical. These segments will then be prioritized for labeling by the volunteers. So reduce the total amount of data they need to annotate.

3. Consolidated Analytical Display: The web app will be upgraded to give researchers a better way to see patterns in location and time, compare models, and do simple statistical analysis.

4. Augmented Data Validation for Citizen Contributions: Before using the volunteer scientists' data to train the machine learning, a new user role, 'Verifier', will check its accuracy.

Interactions:

- New External Systems:
  - Geographic Data Utilities (WMS/WFS) for map-based visualizations within the analytical interface.
  - Statistical Computation Modules (SciPy/R) integrated into the dashboard to give immediate data analysis.
- Revised Operational Sequences::
  - The information from the volunteer scientists is checked for quality before it's used to be used the machine learning models
  - The labeling work is focused on the most important data points thanks to active learning

4.2 External Interfaces

Altered Interfaces:

- Geographic Visualization API: connects to a tool (GeoPandas) to get basic map information for showing data in the analysis tool.
- Classification Validation Module: creates a connection between the tool that checks data quality and the database (SQLite). It also adds places in the database to store if the data is verified and who checked it.
- Statistical Analysis Module: helps the main part of the website communicate with programs (SciPy or R) that do math calculations, so it can show important numbers.
- Intelligent Prioritization Communication: lets the machine learning part of the system talk to the Citizen Science Platform.

Modifications to Existing Connections:

- Zooniverse: adds feedback about the quality assurance status of annotations.
- Machine Learning API: defines outputs related to image segmentation and biomass estimation.

Updated diagram of External interfaces can be found in Fig 4.2.

4.3 Functions

Updated Use-Case diagram is shown in 4.3.

New Use Cases:

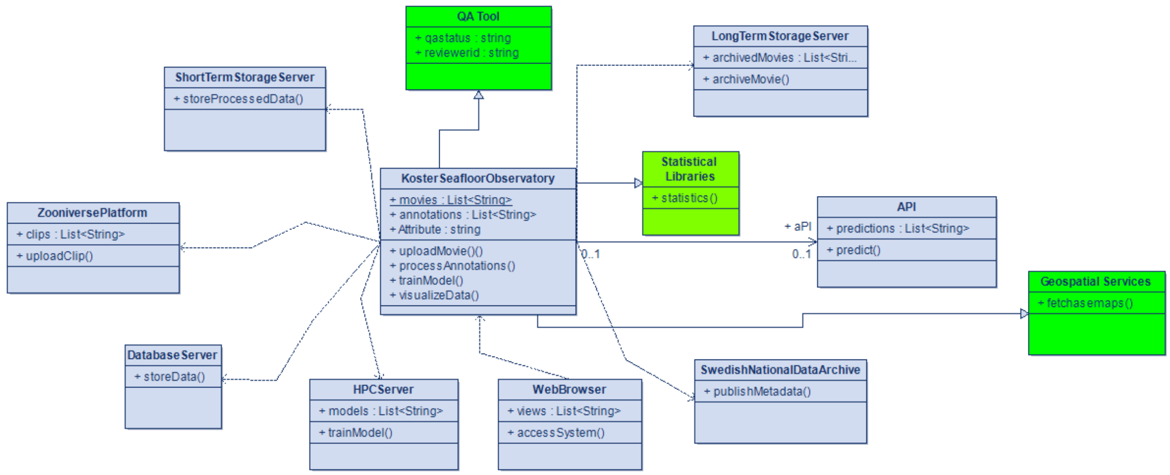

Figure 4.2: Updated KSO External Interfaces Class Diagram reflecting suggested changes (highlighting additions/modifications).

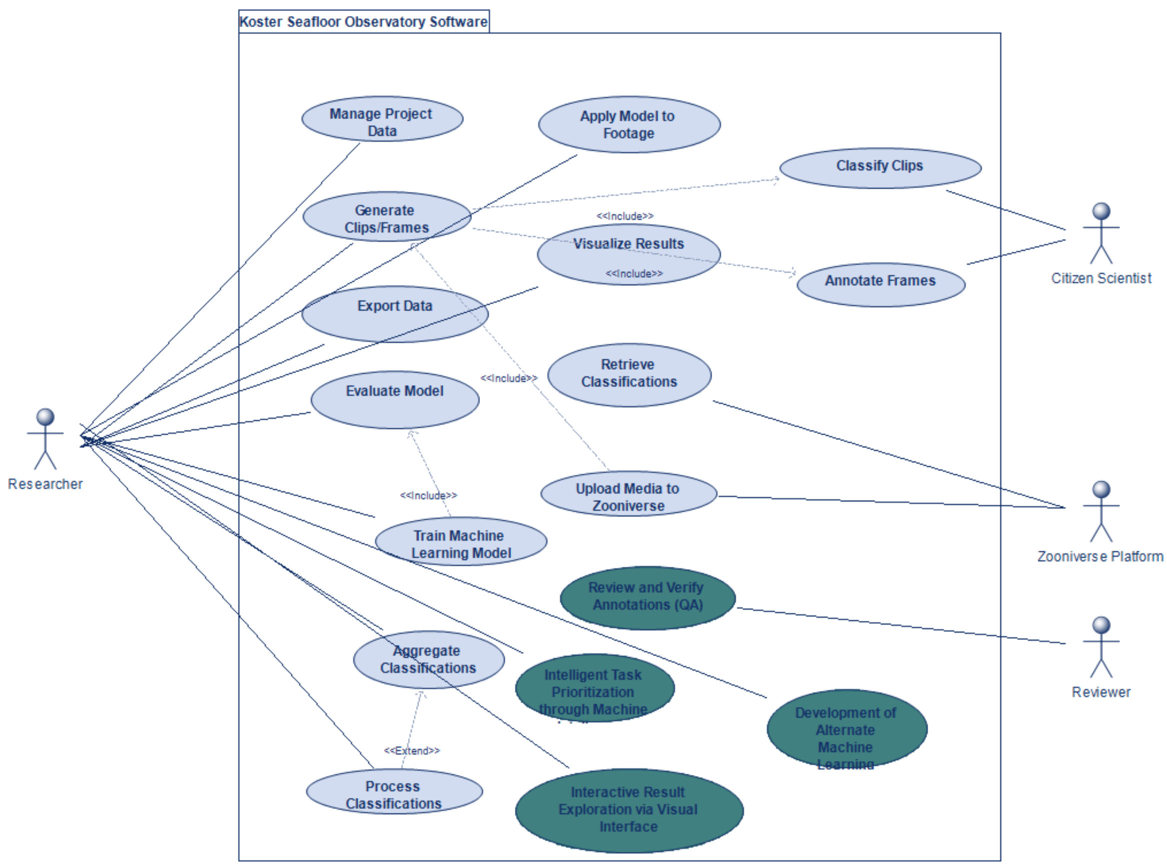

Figure 4.3: Updated Use Case Diagram

- Annotation Validation and Confirmation (Quality Assurance): Sequence Visualization: Details a cyclical verification process involving "Accept" or "Decline" stages 4.4 and Table 4.1.

- Intelligent Task Prioritization through Machine Intelligence: Process Flow Diagram: Depicts concurrent operations for assessing data segment importance and adjusting task priorities Fig.4.5 and Table 4.2.

- Development of Alternate Machine Learning Algorithms (Segmentation): Lifecycle Diagram: Outlines the stages of a model, from initial training to performance assessment and final implementation 4.6 and Table 4.3.

- Interactive Result Exploration via Visual Interface: Activity Flowchart: Illustrates the user journey of applying filters, executing queries, and generating visual representations of data 4.7 and Table 4.4.

Key Flows:

- Intelligent Learning-Driven Task Assignment: The machine learning system pinpoints data segments with lower prediction certainty and subsequently designates these as urgent tasks within the Zooniverse platform.

- Interactive Data Analysis Portal: Researchers can refine data based on species or timeframes and subsequently produce geographical visualizations and statistical summaries without requiring manual coding.

4.4 Logical Database Requirements

Updated database requirements can be found in Fig. 4.8

New Tables/Fields:

- Annotation QA: Within the agg annotations `agg_annotations_frame/clip` data storage: A new field, `qa_status`, will track the verification level of each annotation

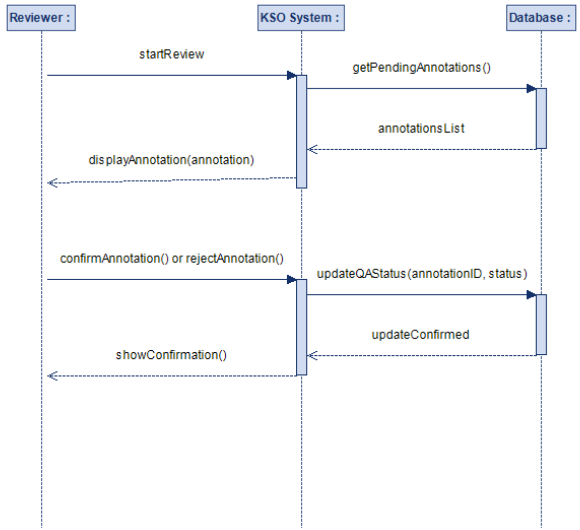

Figure 4.4: Review and Verify Annotations (QA). Sequence Diagram

(using a predefined set of values). A new field, `reviewer_id`, will identify the user who performed the quality check.

- Active Learning: Within the Subjects data storage: A new field, `priority_score`, will store a numerical value indicating the importance of a data item for annotation. A new field, `priority_status`, will categorize the annotation priority (using a predefined set of values).

- Expanded ML: `Segmentation_results` will store the locations of segmentation masks, connected to the relevant `species_id` and `frame_number`.`Biomass_estimates` will include a field, `biomass_value`, to represent the estimated biomass.

Associations: The Segmentation_results data is a specialized type of `model_annotations`.The Biomass_estimates data is linked to the species table.

4.5 System Quality Attributes

- Outlines key system qualities, prioritizing Usability and Performance.

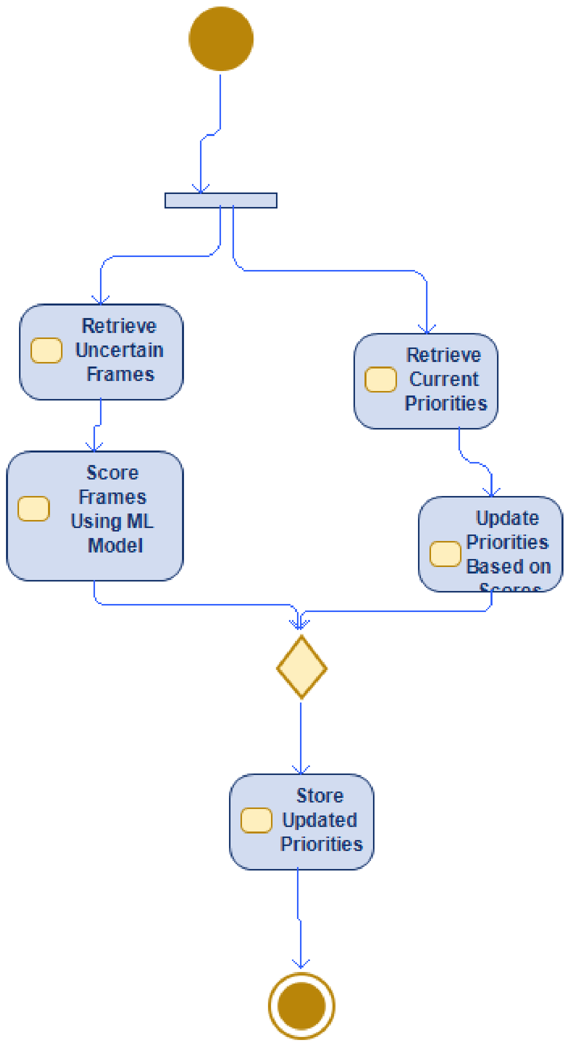

Figure 4.5: Activity Diagram for Prioritize Tasks via Active Learning

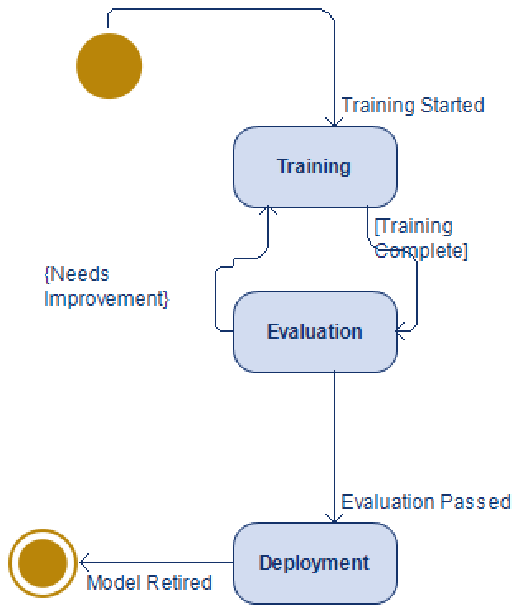

Figure 4.6: State Diagram for Train Alternative ML Model (Segmentation)

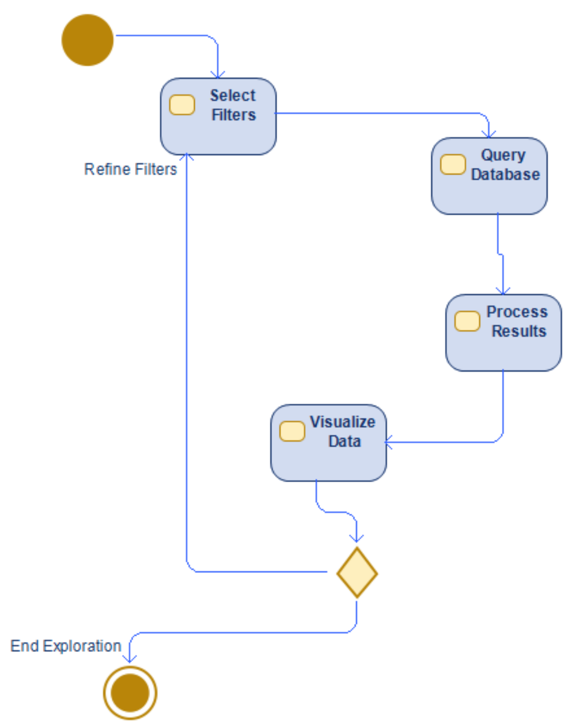

Figure 4.7: Activity Diagram for Explore Results via Dashboard

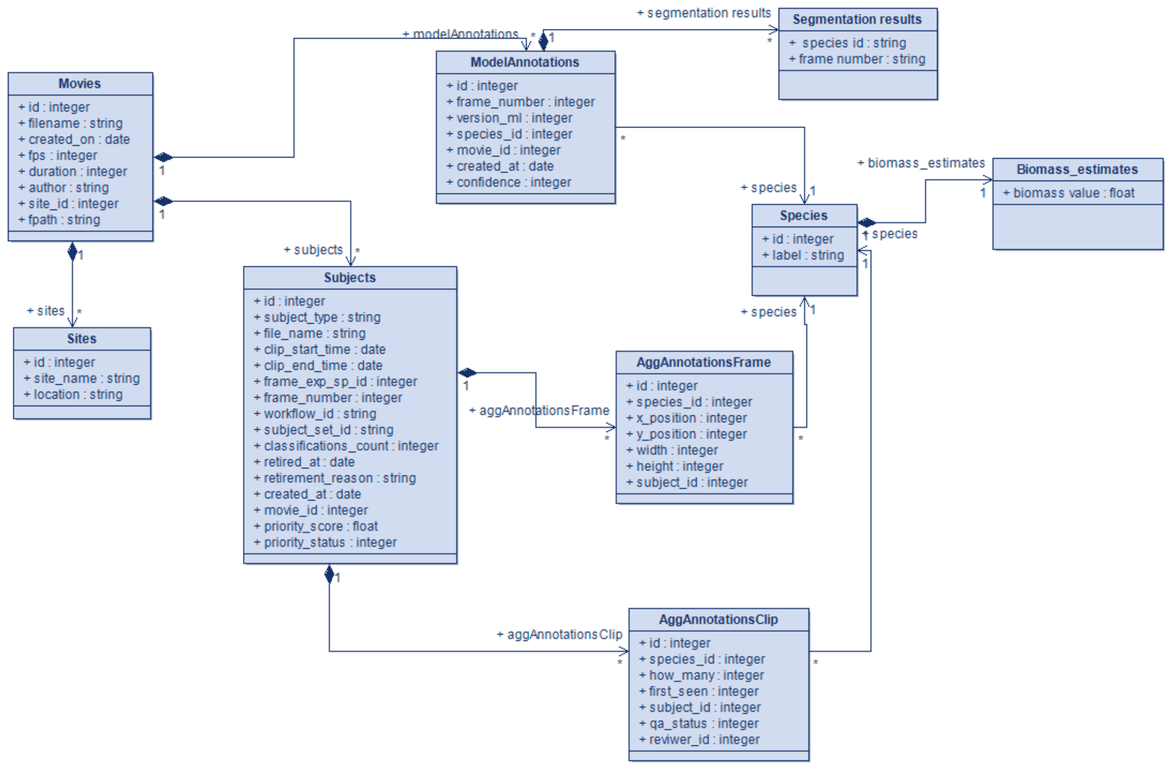

Figure 4.8: Updated KSO Logical Database Requirements

- **Usability**: Dashboard should allow filtering and map viewing in ≤ 3 steps, with clear visualization labels.
- **Performance**: Active learning prioritization for 1000 items should take ¡ 1 hour.
- **Accuracy**: New ML models (segmentation, biomass) must meet mIoU ≥ 0.70 and R-squared ≥ 0.80, respectively.
- **Efficiency**: Annotation review interface should support processing ≥ 10 annotations/minute.
- **Modularity**: ML pipeline should allow adding new model types via a standardized interface.

4.6 Design Constraints

- Details external limitations on system design.
- QA/Dashboard: May require standardizing Python libraries (e.g., Plotly, GeoPandas).
- Active Learning: May depend on specific frameworks (e.g., modAL) or require custom development.
- Expanded ML: Integrating different ML frameworks (e.g., TensorFlow, PyTorch) may increase complexity.
- General: Enhancements will likely require more development resources.

4.7 Supporting Information

- Lists resources relevant to the proposed system enhancements.
- Includes research on active learning for ecological analysis, documentation for relevant software, QA best practices for citizen science, and ML model benchmarks.

- This information will inform detailed design and implementation.

Use Case Name	Annotation Validation and Confirmation (QA)
Actors	Reviewer
Description	The reviewer verifies annotations made by citizen scientists, accepting or declining them based on quality in a cyclical verification process.
Pre-Conditions	Annotations are available for review, and the reviewer is logged in.
Data	Annotation data, QA status
Response	The system updates the annotation's QA status.
Stimulus	The reviewer wants to ensure annotation quality.
Normal Flow	1. The reviewer selects the "Review Annotations" option. 2. The system displays a pending annotation. 3. The reviewer evaluates the annotation. 4. The reviewer chooses to accept or decline it. 5. The system updates the QA status and moves to the next annotation.
Alternative Flow	The reviewer may request clarification from the citizen scientist if the annotation is ambiguous.
Exception Flow	If the annotation is unclear, the reviewer flags it for further review.
Post-Conditions	The annotation's QA status is updated in the database.
Comments	This ensures high-quality data for downstream processes like model training.

Table 4.1: Tabular Description of *Annotation Validation and Confirmation (QA)*

Use Case Name	Intelligent Task Prioritization through Machine Intelligence
Actors	Researcher
Description	The system uses machine learning to assess data segment importance and adjust task priorities concurrently, such as prioritizing frames for annotation.
Pre-Conditions	A trained ML model is available, and tasks (e.g., frames) are pending.
Data	Frame data, model uncertainty scores
Response	The system updates task priorities.
Stimulus	The researcher wants to optimize annotation efforts.
Normal Flow	1. The researcher initiates the "Prioritize Tasks" process. 2. The system retrieves pending tasks. 3. The ML model scores tasks based on importance (e.g., uncertainty). 4. The system updates task priorities in the database.
Alternative Flow	The researcher may manually override the priorities if needed.
Exception Flow	If the ML model fails to score tasks, the system falls back to a default priority order.
Post-Conditions	Task priorities are updated, guiding future annotations.
Comments	This leverages active learning to enhance efficiency.

Table 4.2: Tabular Description of *Intelligent Task Prioritization through Machine Intelligence*

Use Case Name	Development of Alternate Machine Learning Algorithms (Segmentation)
Actors	Researcher
Description	The researcher trains, assesses, and deploys a segmentation model through its lifecycle to analyze footage (e.g., coral coverage).
Pre-Conditions	Annotated segmentation data is available, and the researcher is logged in.
Data	Annotated frames, model parameters
Response	The system trains and deploys the segmentation model.
Stimulus	The researcher wants advanced analysis capabilities.
Normal Flow	1. The researcher selects the "Train Segmentation Model" option. 2. The researcher selects the dataset and model type. 3. The system trains the model. 4. The researcher assesses model performance. 5. If satisfactory, the researcher deploys the model.
Alternative Flow	The researcher may test alternative model architectures.
Exception Flow	If training fails, the system notifies the researcher to adjust parameters.
Post-Conditions	A trained segmentation model is deployed for use.
Comments	Supports detailed analysis like coral segmentation.

Table 4.3: Tabular Description of *Development of Alternate Machine Learning Algorithms (Segmentation)*

Use Case Name	Interactive Result Exploration via Visual Interface
Actors	Researcher
Description	The researcher applies filters, executes queries, and generates visual representations of data via an interactive dashboard.
Pre-Conditions	Data is available in the system, and the researcher is logged in.
Data	Filter criteria, query results, visualization settings
Response	The system displays visualizations based on user inputs.
Stimulus	The researcher wants to explore project results interactively.
Normal Flow	1. The researcher selects the "Explore Results" option. 2. The researcher applies filters (e.g., date, species). 3. The system queries the database. 4. The system generates and displays visualizations.
Alternative Flow	The researcher may save or export the visualizations for later use.
Exception Flow	If no data matches the filters, the system prompts the researcher to adjust them.
Post-Conditions	Visualizations are displayed, providing insights.
Comments	Enhances data analysis and decision-making.

Table 4.4: Tabular Description of *Interactive Result Exploration via Visual Interface*