2025

AI-Powered Tool for Annotating Satellite Images

Dr.Mohamed

EGSA

1/1/2025

DDSS.03

First Version

January 1, 2025

Project Name

AI-Powered Tool for Annotating Satellite Images

University Name

Helwan University

Report On

Monthly report

PREPARED BY	Ahmed Fatthi	28/01/2025
REVIEWED BY	Eng.Mazen Hesham	Date
APPROVED BY	Eng.Tamer Mohamed	Date
AUTHORIZED BY	Dr.Mohamed El-Hady	Date
PROF	PROJECT MANAGER	

Contents

Contents

1. List of Figures

1. List of Tables

2. Document Versions

V0.01	

3. ACRONYMS

Name	Description
CNNs	convolutional neural networks

4. Introduction

The rapid advancement of satellite technology has led to an explosion of data, offering valuable insights across various domains, from environmental monitoring to urban planning. However, one of the major challenges faced by industries and researchers utilizing satellite imagery is the time-consuming and labor-intensive process of annotating these images to extract meaningful information.

This project aims to develop an AI-powered tool designed to automate the annotation of satellite images. By leveraging deep learning techniques, specifically convolutional neural networks (CNNs), the tool will be able to accurately identify and label features such as buildings, roads, vegetation, and bodies of water. The ultimate goal is to reduce the time and effort required for manual image annotation, improving the efficiency of analysis while maintaining high accuracy.

In this report, we will discuss the methods employed in training the AI model, the datasets used for model validation, and the challenges encountered during development. This tool not only holds the potential to expedite satellite image analysis but also represents a significant advancement in the application of AI to geospatial data.

5. Methodology

5.1. **Overview of the Approach**

The primary objective of this project is to develop an AI-powered tool for annotating satellite images by leveraging a semantic segmentation model, DeepLabV3+. The methodology involves data collection, model selection and training, fine-tuning the model for enhanced accuracy, and integrating it into a web-based application using Python (FastAPI) for the backend and ReactJS for the frontend.

5.2. **Data Preparation**

- **Data Source:** Satellite images are sourced from Earth Explorer. The images are in TIFF format with RGB bands.
- **Testing:** The pretrained model will be evaluated directly on the downloaded images to assess its segmentation performance.
- **Fine-Tuning (If Needed):** In case the pretrained model requires domain-specific adaptation, additional annotations may be created for fine-tuning using the collected satellite images.

Figure 1: Screenshot of satellite imagery from Earth Explorer, selected for testing the DeepLabV3+ model.

Why Earth Explorer and TIFF Format with RGB Bands?

- **Earth Explorer:** This platform provides access to a wide variety of high-resolution satellite imagery, making it an ideal source for obtaining diverse datasets for testing and fine-tuning. Its user-friendly interface and availability of public datasets ensure accessibility for academic projects.
- **TIFF Format:** TIFF files are widely used in remote sensing due to their ability to store high-quality, georeferenced images without data loss. This format ensures that the full detail of satellite imagery is preserved, which is crucial for accurate semantic segmentation.
- **RGB Bands:** The use of RGB bands aligns with the pretrained DeepLabV3+ model's input requirements, simplifying the integration process. RGB bands also represent the visible spectrum, which is essential for many practical applications, such as land use classification and environmental monitoring.

5.3.**Model Utilization**

5.3.1. Introduction

Semantic segmentation is a critical task in computer vision, particularly in applications such as satellite imagery analysis, where precise object detection and boundary delineation are essential. For our graduation project, we evaluated several state-of-the-art models and ultimately selected **DeepLabv3+** for its superior performance, flexibility, and robustness in handling complex segmentation tasks. This document explains our rationale for choosing DeepLabv3+, highlights its advantages, and justifies its suitability for our specific use case.

5.3.2. Overview of DeepLabv3+

“DeepLabv3+” is a semantic segmentation model introduced by Google Research. It builds on the strengths of its predecessor, DeepLabv3, and incorporates an encoder-decoder structure to refine the segmentation process. Key features of DeepLabv3+ include:

1. Atrous Spatial Pyramid Pooling (ASPP): Efficiently captures multi-scale contextual information by applying atrous (dilated) convolutions with different rates.
2. Encoder-Decoder Architecture: Improves boundary segmentation by recovering spatial information lost during downsampling.
3. Atrous Separable Convolutions: Enhances computational efficiency without sacrificing accuracy.
4. Flexibility: Can be fine-tuned for various datasets and adjusted to balance accuracy and speed.

5.3.3. Why We Chose DeepLabv3+

1. Superior Performance

DeepLabv3+ consistently ranks among the top-performing models in semantic segmentation benchmarks, such as PASCAL VOC and COCO datasets. It achieves high mean Intersection over Union (mIoU) scores, demonstrating its capability to handle complex segmentation tasks with precision.

2. Multi-Scale Contextual Understanding

The ASPP module in DeepLabv3+ allows the model to capture information at multiple scales, making it particularly effective for satellite imagery, where objects of interest can vary significantly in size (e.g., buildings, roads, and ships).

3. Robust Boundary Detection

The encoder-decoder design ensures that fine details, such as object boundaries, are accurately preserved, addressing a critical requirement in our project.

4. Efficiency

By using atrous separable convolutions, DeepLabv3+ reduces the computational cost compared to traditional convolutional operations, making it feasible to process large satellite images efficiently.

5. Proven Applications in Remote Sensing

DeepLabv3+ has been successfully applied in remote sensing tasks, including land cover classification, urban mapping, and disaster assessment. These real-world applications validate its effectiveness in handling satellite imagery.

5.3.4. Comparison with Other Models

- **PSPNet**

While PSPNet (Pyramid Scene Parsing Network) is another strong candidate for semantic segmentation, we opted for DeepLabv3+ for the following reasons:

  - Boundary Precision: DeepLabv3+ outperforms PSPNet in capturing fine details and object boundaries due to its encoder-decoder structure.
  - Efficiency: Atrous separable convolutions in DeepLabv3+ offer a more efficient alternative to PSPNet’s pyramid pooling module.
  - Flexibility: DeepLabv3+ provides greater flexibility for fine-tuning, making it more adaptable to our specific dataset.

- **UNet**

UNet is widely used for medical imaging but lacks the multi-scale contextual understanding provided by ASPP in DeepLabv3+. Additionally, UNet struggles with large-scale images due to its simpler architecture.

5.3.5. Fine-Tuning DeepLabv3+

1. To achieve optimal performance, we plan to fine-tune DeepLabv3+ on our satellite imagery dataset. Key steps include:
2. Dataset Preparation: Preprocess the satellite images and annotations to match the input requirements of the model.
3. Transfer Learning: Use pre-trained weights (e.g., on COCO or ImageNet) to leverage prior knowledge and accelerate training.
4. Hyperparameter Optimization: Adjust learning rates, batch sizes, and other parameters to optimize performance on our dataset.
5. Augmentation: Apply techniques such as rotation, flipping, and scaling to improve model robustness.

5.3.6. Conclusion

DeepLabv3+ is a state-of-the-art model that aligns perfectly with the requirements of our graduation project. Its superior performance, efficiency, and adaptability make it the ideal choice for semantic segmentation in satellite imagery. By leveraging its advanced features and fine-tuning it for our dataset, we aim to achieve accurate and reliable results, setting a strong foundation for the success of our project.

5.4. **Tool Development**

- **Backend Development:** Python with FastAPI is used to develop an API that processes input images and returns segmentation results.
- **Frontend Development:** ReactJS is used to create a user-friendly interface for uploading images and visualizing the output.

Why FastAPI for Backend and ReactJS for Frontend?

- **FastAPI for Backend:** FastAPI was chosen due to its high performance and ease of use. It supports asynchronous operations, which is important for handling large satellite images efficiently. FastAPI also provides automatic API documentation (Swagger UI), speeding up the development process.
- **ReactJS for Frontend:** ReactJS is ideal for creating a dynamic and responsive user interface. Its component-based architecture allows for easy reuse of UI elements, and its virtual DOM ensures fast rendering, which is important for displaying large annotated images smoothly.

5.5. Comparison between DeepLabv3+ and Sam Segmentation Models

To determine the most suitable segmentation model for our system, we compared **DeepLabV3+** and **SAM (Segment Anything Model)** based on their segmentation outputs.

1. **Full Segmentation with DeepLabV3+:**
    - o DeepLabV3+ performs full-image segmentation, classifying each pixel into a category.
    - o It segments both the object and its background, with no transparency.

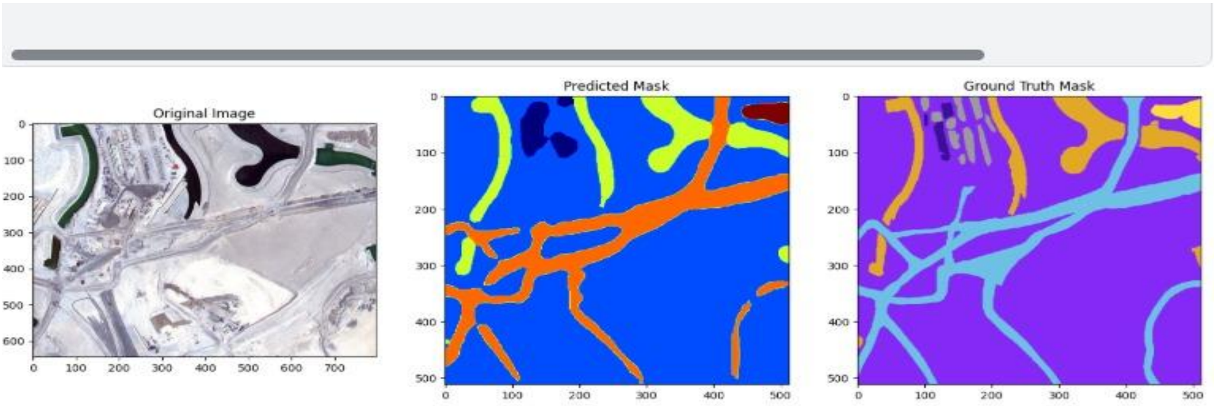


Figure 2 Full Segmentation using DeepLabv3+ – The entire image is segmented, including both the object and background, with no transparency.

2. **Full Segmentation with SAM:**
   - SAM focuses specifically on the selected object, segmenting it without including the background.
   - The segmentation result maintains transparency, preserving finer details of the object.

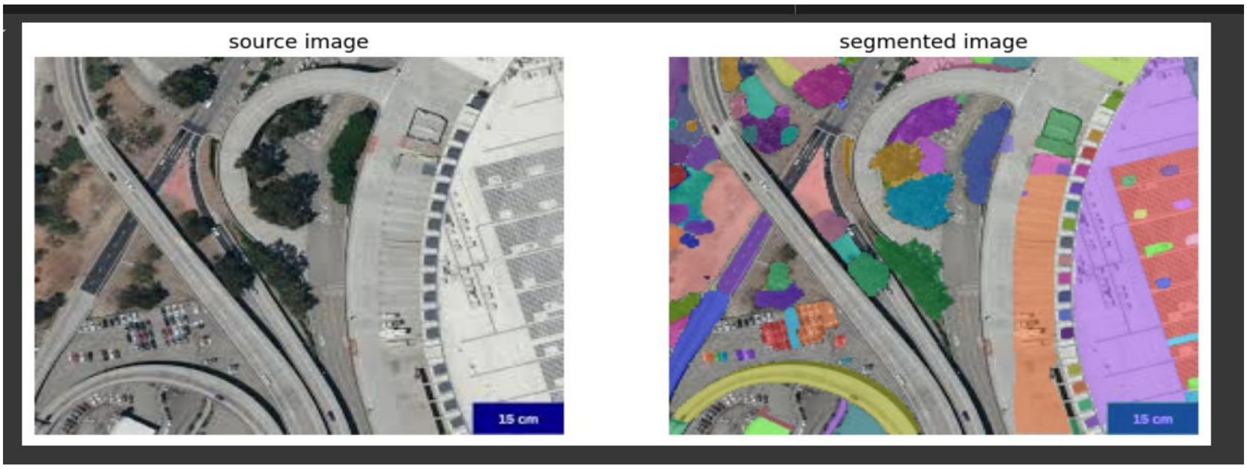


Figure 3 Full Segmentation using SAM (Segment Anything Model) – The object is segmented with transparency, excluding the background.

3. **Object Segmentation with Bounding Box:**
   - When a user selects an object by drawing a bounding box, SAM accurately segments it.
   - However, small objects may not be segmented as precisely, requiring further optimization to improve SAM's accuracy for such cases.

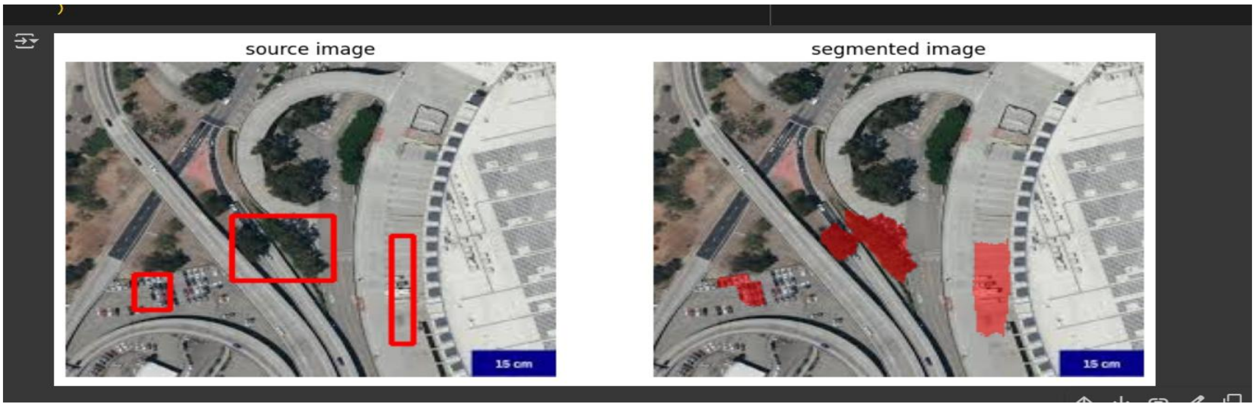


Figure 4 Object Segmentation with Bounding Box – The user selects an object, and SAM accurately segments it, except for small objects that may require accuracy improvements.

5.6. **Implemented API Endpoints**

This section outlines the key API endpoints implemented in the system for handling image uploads and retrieval.

1. Root Endpoint

This endpoint serves as the entry point for the API, providing a welcome message.

Request:

```
curl http://localhost:8000/
```

Expected Response:

```
{
  "message": "Welcome to the Satellite Image Annotation Tool"
}
```

2. Upload an Image

This endpoint allows users to upload an image to the system. The image is stored, and its metadata is returned.

Request:

```
curl -X POST http://localhost:8000/api/upload-image/ \
  -H "Content-Type: multipart/form-data" \
  -F file=@./data/test_image.jpg
```

Expected Response:

```
{
  "success": true,
  "message": "File uploaded successfully",
  "image": {
    "file_name": "test_image.jpg",
    "file_path": "uploads/b1030d37-9fa8-4823-b5ad-43ee4cd22e5c.jpg",
    "resolution": "540x360",
    "source": "user_upload",
    "image_id": 1,
    "capture_date": "2025-02-26T19:43:14.121927",
    "created_at": "2025-02-26T19:43:14.121927"
  }
}
```

3. Retrieve All Images

This endpoint retrieves a list of all uploaded images along with their metadata.

Request:

curl http://localhost:8000/api/images/

Expected Response:

```
    [
  {
    "file_name": "test_image.jpg",
    "file_path": "uploads/b1030d37-9fa8-4823-b5ad-43ee4cd22e5c.jpg",
    "resolution": "540x360",
    "source": "user_upload",
    "image_id": 1,
    "capture_date": "2025-02-26T19:43:14.121927",
    "created_at": "2025-02-26T19:43:14.121927"
  },
  {
    "file_name": "test_image.jpg",
    "file_path": "uploads/d8defaa1-f5c9-400f-86e7-f2ffc4ce0d3c.jpg",
    "resolution": "540x360",
    "source": "user_upload",
    "image_id": 2,
    "capture_date": "2025-02-26T20:42:35.066164",
    "created_at": "2025-02-26T20:42:35.066164"
  }
]
```

6. Project Design

The AI-powered tool for annotating satellite images is designed for **interactive and efficient segmentation**. It consists of a **ReactJS frontend, FastAPI backend, and a DeepLabV3+ model** for processing. Users upload images, select areas for annotation, and receive segmented results. This section outlines the **system architecture and data flow**, with room for future enhancements.

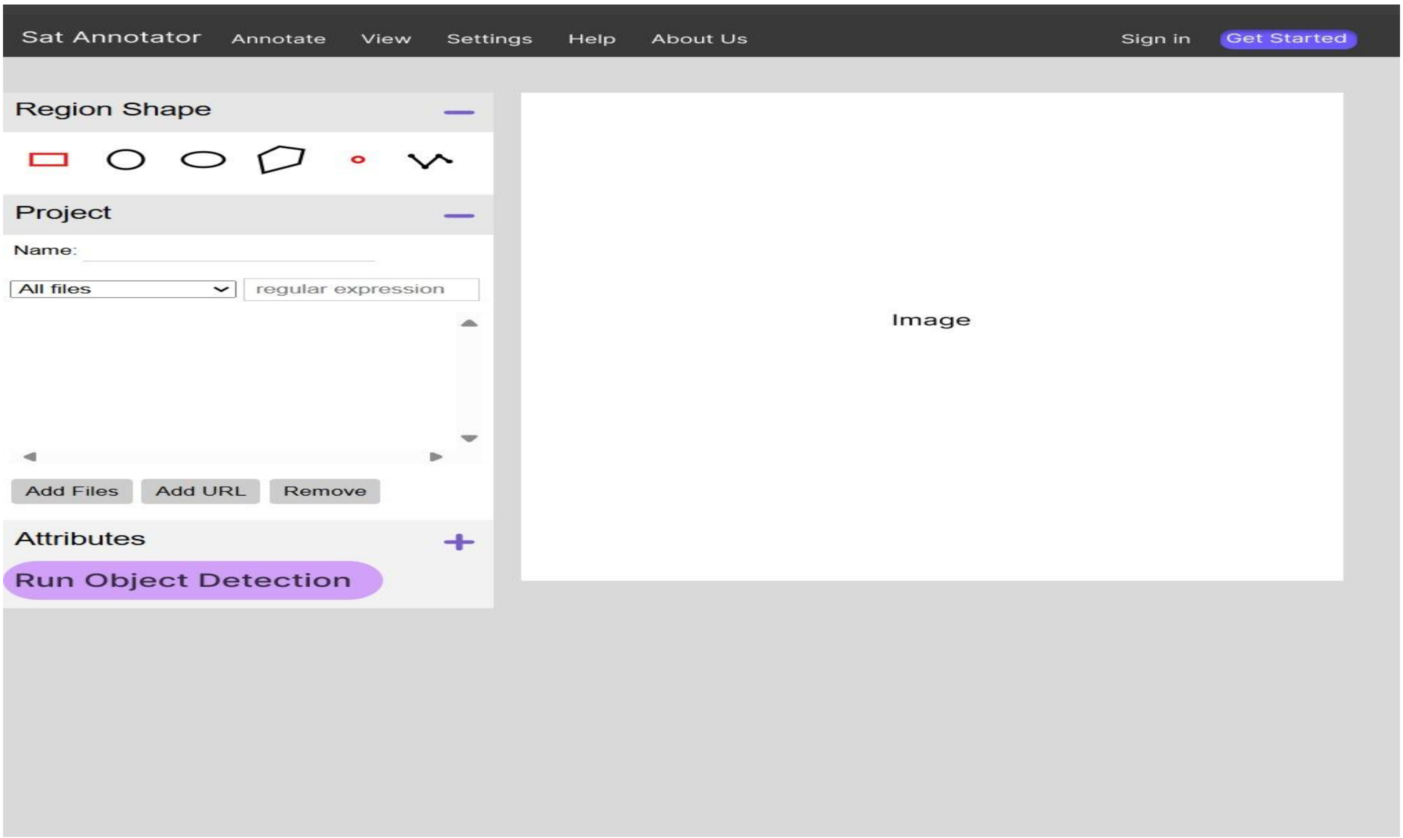


Figure 5

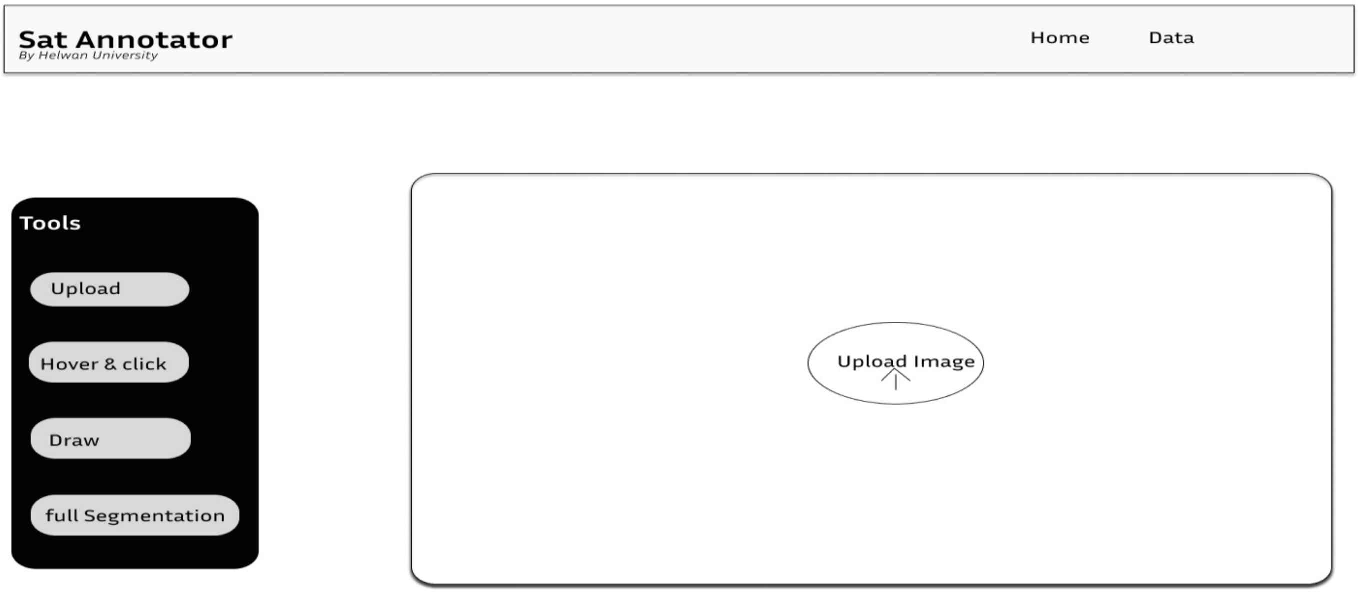


Figure 6

a. System Architecture

The system architecture of the AI-powered tool for annotating satellite images follows a modular design, enabling user interaction, efficient processing, and scalability. It consists of three main components: **frontend, backend, and deep learning model integration**. While this design is the current approach, it may evolve as the project progresses.

1. Frontend (User Interface)

- Developed using **ReactJS** for an interactive and user-friendly experience.
- Allows users to **upload satellite images** and interact with them before annotation.
- Users **click on points or draw a rough boundary** on the uploaded image to guide the model.
- Sends the selected region and image to the backend via **RESTful APIs**.
- Displays the segmented output returned by the backend.

2. Backend (API and Processing Layer)

- Implemented using **FastAPI**, ensuring fast response times and scalability.
- Handles requests from the frontend, including image uploads and user-defined regions.
- Processes the input and forwards it to the deep learning model for segmentation.
- Stores and retrieves annotated results as needed.

3. Deep Learning Model (Segmentation and Annotation)

- Uses a **pretrained DeepLabV3+ model** for semantic segmentation of satellite images.
- Accepts **TIFF images with RGB bands**, obtained from Earth Explorer.
- Instead of segmenting the entire image, the model **focuses on user-selected regions**, improving precision.
- Outputs segmentation masks highlighting boundaries of the selected objects or regions.

4. Data Flow and Interaction

1. **User uploads a satellite image** through the frontend.
2. **User selects points or draws a rough boundary** on the image to guide the segmentation process.
3. The frontend sends the **image and selected region** to the backend.
4. The backend processes the request and sends the data to the **DeepLabV3+ model**.
5. The model generates segmentation masks **focusing on the selected area**.
6. The backend receives the segmented result and sends it back to the frontend.
7. The frontend displays the **annotated image with clear boundaries**.

5. Possible Future Enhancements

- **Fine-tuning the model** on specific datasets for improved performance.
- **Implementing interactive tools** for users to refine boundaries before processing.
- **Optimizing API performance** to handle real-time interactions smoothly.

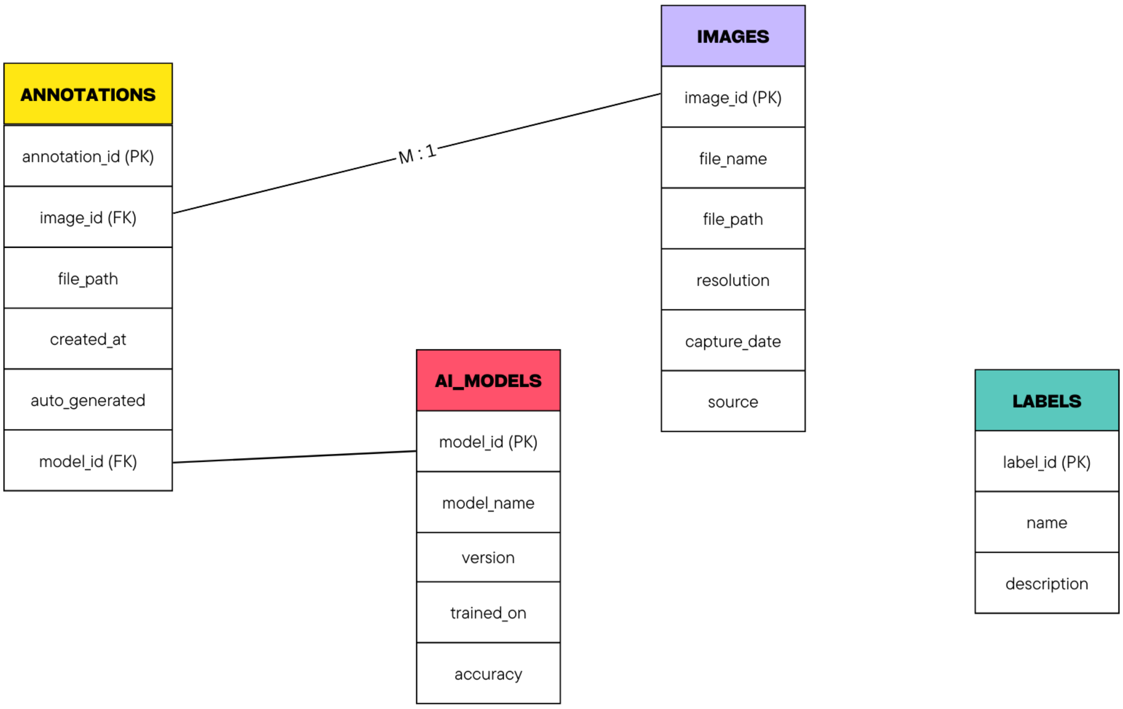
- **Adding authentication and user management** for better access control.
- **Integrating cloud storage** for managing large datasets efficiently.

6. Database Design

The system utilizes a relational database to store and manage images, annotations, AI model metadata, and labels efficiently. The core entities include:

- **Images:** Stores metadata about uploaded satellite images, including file path, resolution, and source.
- **Annotations:** Links images with segmentation results, storing annotation files and tracking whether they were auto-generated.
- **AI Models:** Maintains records of different AI models used for segmentation, including model name, version, training data, and accuracy.
- **Labels:** Stores classification labels and their descriptions, which can be used to categorize segmented objects.

This structured design ensures efficient data retrieval, traceability of model-generated annotations, and seamless integration between different system components. The following ERD illustrates the database schema:



7. Project Progress

This project can be summarized in the following stages as listed in table 1.

Table 1. The progress of the project.

Stage No.		Progress (%)	Remarks	Month
1	Project registration with the department	100%	Registered the project in the college.	December 2024
2	Project Planning and Requirements	100%	Specified project requirements, and Determined technologies (FastAPI, ReactJS) and model (DeepLabV3+).	January 2025
3	Backend Development and Initial Testing	100%	- Begin backend development with FastAPI. - Gather satellite images from Earth Explorer. - Test the pretrained DeepLabV3+ model on selected images. - Analyze model performance. - Decide if fine-tuning is needed.	February 2025
4	Model Fine-Tuning and Backend Development	--	- Fine-tune DeepLabV3+ model (if needed) with additional data. - Continue backend development (API for model inference). - Evaluate model's performance after fine-tuning.	March 2025
5	Backend and Frontend Development	--	- Continue backend development and API integration. - Begin frontend development with ReactJS for user interface.	April 2025
6	Frontend-Backend Integration	--	- Continue backend and frontend development. - Integrate backend with frontend. - Implement functionality to upload and display satellite images.	May 2025
7	Final Testing and Project Delivery	--	- Finalize backend and frontend integration. - Conduct extensive testing of the full system. - Prepare project documentation and presentation.	June 2025
8	Evaluation	--	--	--
9	Project submission	--	--	--

8. Current Month Activities

Table 2. Current month activities

Stage No.		Progress (%)	Remarks
1	Begin Backend Development (FastAPI)	100	Start building the backend using FastAPI, setting up APIs for future model integration and data processing.
2	Gather Satellite Images from Earth Explorer	100	Download satellite images from Earth Explorer in TIFF format with RGB bands for testing and evaluation.
3	Select Model (DeepLabV3+)	100	Test the pretrained DeepLabV3+ model on the gathered satellite images to assess its initial performance.
4	Analyze Model Performance	100	Evaluate how well the pretrained model performs on the images and decide if fine-tuning is necessary.
5	Decide on Fine-tuning (if required)	0	If needed, plan the fine-tuning process for the model to improve its performance.

9. Next Month Activities

Table 3. Next month activities

Stage No.		Progress (%)	Remarks
1	(Optional) Fine-tune DeepLabV3+ or SAM model	0	Train with additional satellite image data if performance improvements are needed.
2	Develop API for object segmentation	0	Implement an API that takes **user click coordinates** and returns the boundaries of the clicked object only.
3	Start frontend implementation	0	Set up the ReactJS project structure and install necessary dependencies.
4	Design UI layout	0	Create the interface for **image upload and interactive object selection**.
5	Implement image upload feature (frontend)	0	Develop the component for users to upload satellite images.
6	Connect frontend with backend	0	Establish communication between the ReactJS frontend and FastAPI backend.
7	Implement image preprocessing	0	Prepare uploaded images for model inference (e.g., resizing, normalization).
8	Validate processed images	0	Ensure uploaded and preprocessed images meet model input requirements.

Conclusion

In this first phase of the project, we have successfully established the foundational aspects necessary for the development of the AI-powered tool for annotating satellite images. We have clearly defined the project requirements, selected the technologies (FastAPI for the backend, ReactJS for the frontend), and chosen the DeepLabV3+ model for image segmentation. Additionally, we have identified Earth Explorer as a reliable source for obtaining satellite images in TIFF format with RGB bands.

Our initial efforts have focused on setting up the backend and performing preliminary tests with the pretrained DeepLabV3+ model. The results of these tests will guide us in determining whether fine-tuning is required to enhance model performance. Going forward, we will continue to refine the system, integrating both the backend and frontend components, and ensuring that the tool delivers accurate annotations for satellite imagery.

The upcoming months will be dedicated to improving the model's performance, building out the full functionality of the system, and conducting extensive testing to ensure that the final product meets the project's objectives. We are confident that with continued progress and effort, the AI-powered tool will provide valuable insights and capabilities for satellite image analysis.

References

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. European Conference on Computer Vision (ECCV). [arXiv](https://arxiv.org/abs/1802.02611)

Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2017). *Pyramid Scene Parsing Network*. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [arXiv](https://arxiv.org/abs/1612.01105)

Chen, L.-C., Papandreou, G., Schroff, F., & Adam, H. (2017). *Rethinking Atrous Convolution for Semantic Image Segmentation*. [arXiv](https://arxiv.org/abs/1706.05587)

PapersWithCode. (n.d.). DeepLabv3+. Retrieved from https://paperswithcode.com/method/deeplabv3

Satellite Imagery Analysis with DeepLabv3+. (2020). IEEE Geoscience and Remote Sensing Letters

[https://www.researchgate.net/publication/343289493_Farm_Area_Segmentation_in_Satellite_Images_Using_DeepLabv3_Neural_Networks]

Appendices

APPENDIX A

DEVELOPED SOFTWARE

APPENDIX B

HARDWARE DESIGN