**Project Report**

**RADAR**

Abhinav Kumar – S24CSEU0621

Divyansh Dwivedi – S24CSEU0854

Charan Reddy – S24SCEU0563

A report submitted in part fulfilment of the degree of

BTech in Computer Science

Supervisor: Dr. Susmita Das



School of Computer Science Engineering and Technology

Date 12-11-2-25

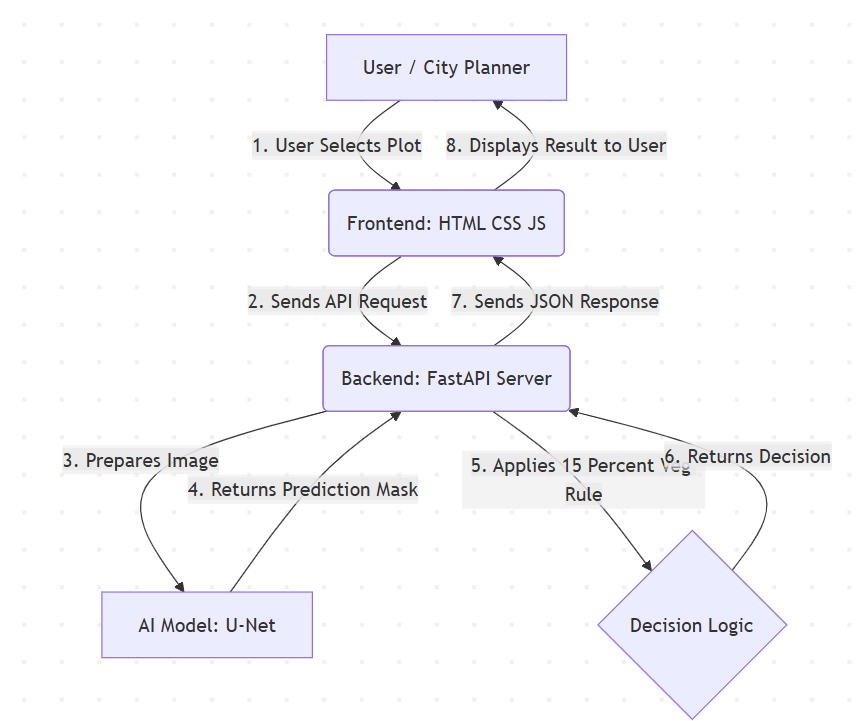
**Introduction**

There has been a steady increase in the global population and it is projected to reach over 9 billion in 2050. Rapid urbanization is taking place to cope up with this drastic increase resulting in conflict between development and sustainable development. The manual way of analysing land for projects and successfully enforcing environment policies is:

* **Inefficient & Time taking**- It may take weeks and increase costs
* **Subjective**- Sustainability may vary from person to person
* **Expensive**- It increases the cost exponentially

This project presents RADAR (Real time Allocation Decision and Analysis Response) an AI powered decision support system (DSS) that automates and simplifies this process. This project aims to:

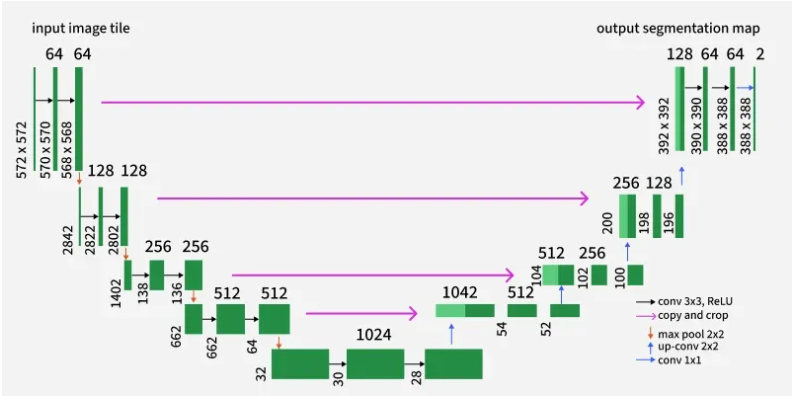
* **Implement AI tool-** Train and implement a U-Net deep learning model
* **Enforce Sustainability-** Implement a ‘Rule Engine’ of environment policies
* **Deliver Insights-** Use an interactive dashboard to help ‘Approve’ or ‘Deny’ any land plot



**Proposed Works/Methodology**

1. **Machine Learning & Model Training-** The brain of the project

* **Model Selection-** The project uses **U-Net** model for biomedical and satellite image segmentation. It is used for capturing both high level context and fine-grained spatial details which covers both residential areas and individual buildings.

****

* **Data Preparation & Cleaning-** The dataset consists of 203 satellite images and their pixel perfect masks which was split into training and validation sets in 8:2 ratio. This dataset was created by joining images from *Semantic segmentation of aerial imagery, Land Cover Classification: Bhuvan Satellite Data* and *Urban Segmentation – ISPRS.*

Image mask

* **Training-** The model was built and trained using **PyTorch.** A dataset class was implemented to convert colour coded masks into numerical class-index tensors. The labels are:

Unlabelled (#9B9B9B) Class 0

Building (#3C1098) Class 1

Land (#8429F6) Class 2

Road (#6EC1E4) Class 3

Vegetation (#FEDD3A) Class 4

Water (#E2A929) Class 5

EPOCHS: 10

The final trained model is saved as unet\_model.pth

1. **Backend API Development-** The server of the project

* **Technology-** This project uses a modern python framework called FastAPI to serve the machine Learning model.
* **Inference-** The FastAPI server loads the unet\_model.pth into memory, when an API request is received:

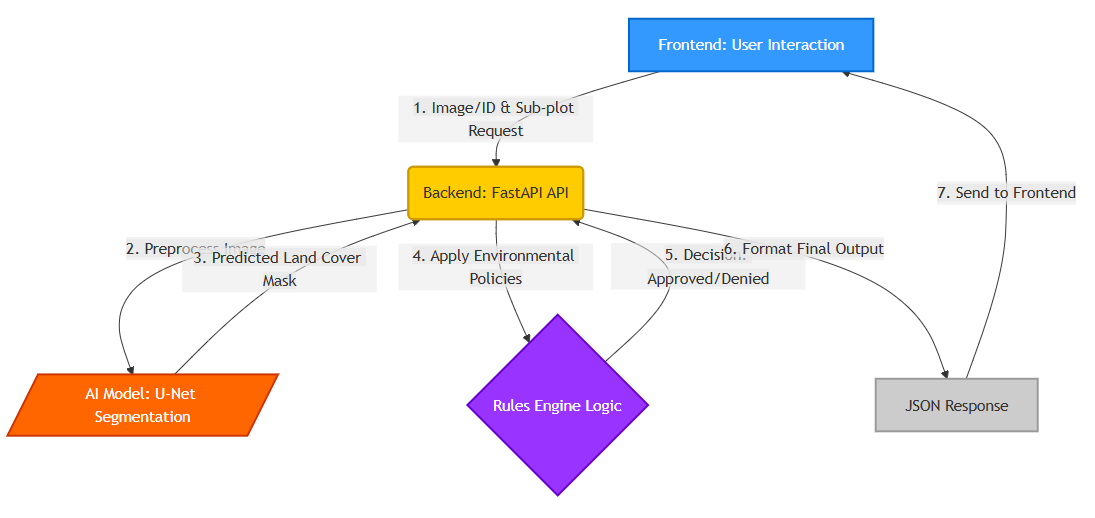
1. Receives the image from the database or the image uploaded by the user.
2. Performs preprocessing on the image i.e Resizing, Normalization and Converting it to Tensor.
3. Feeds the tensor into the **Unet** Model to get a 2D array of predictions
4. Executes the ‘**Rule Engine’** as an algorithm on this array.
5. Returns a **JSON** response as the final decision.

* **API Endpoints-** Two API endpoints are created:
* *GET /api/check\_plot/{id}/{sub\_plot}* for images in the database.
* *POST /api/analyze\_upload/{sub\_plot}* for new images that may be uploaded by the user.

1. **Frontend Interface-** The face of the project

* **Technology-** The project uses Vanilla HTML5, CSS3 and modern JavaScript (ES6+)
* **Interactive menu-** The User Interface(UI) allows the user to load or upload a file and then render it with a 3X3 grid overlay. 
* **Handling-** when a user clicks a subplot(grid cell):

1. It captures the sub\_plot\_id.
2. Dispatches a fetch() call to the appropriate backend API points i.e Load or Upload.
3. It parses the JSON and Dynamically update the HTML to show the ‘Approved’ or ‘Denied’ status. It also fills up the Environmental Matrices.

****

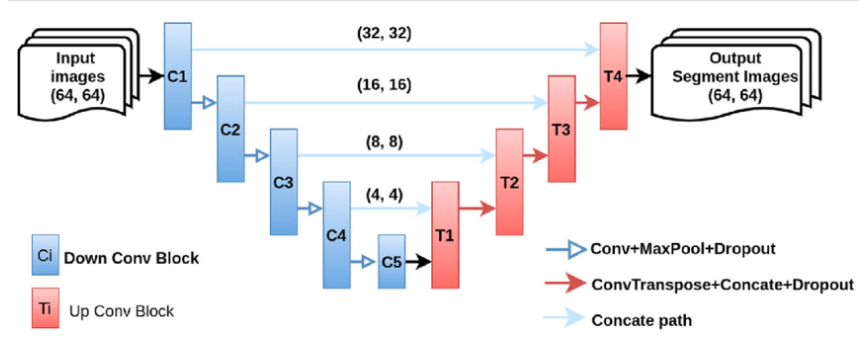
**Data structure and algorithms used**

1. **Data Structures-**

* **PyTorch Tensors-** They are the primary N-dimensional data structures. Images are converted into (H, W, C) arrays to (B, C, H, W) tensors for model processing.
* **Numpy Arrays-** They are used as an intermediary for pixel manipulation, such as analysing the model’s 2D prediction masks.
* **Python Dictionaries-** They are used on the backend to efficiently count the number of pixels and store them. Eg {‘Vegetation’: 2025}
* **JavaScript Object Notation (JSON)-** A universal data format used for ‘communication’ between the FastAPI backend and the JS frontend.

1. **Algorithms-**

* **U-Net Semantic Segmentation-** It uses a contracting path(encoder) to capture context and Symmetric expanding path(decoder) to enable precise localization. This algorithm allows the model to learn ‘WHAT’ is in the image(labels) and remember ‘WHERE’ is it in the image(exact location and boundary).



* **‘Health Engine’ –** It is a logical decision tree executed by the backend on every API call.

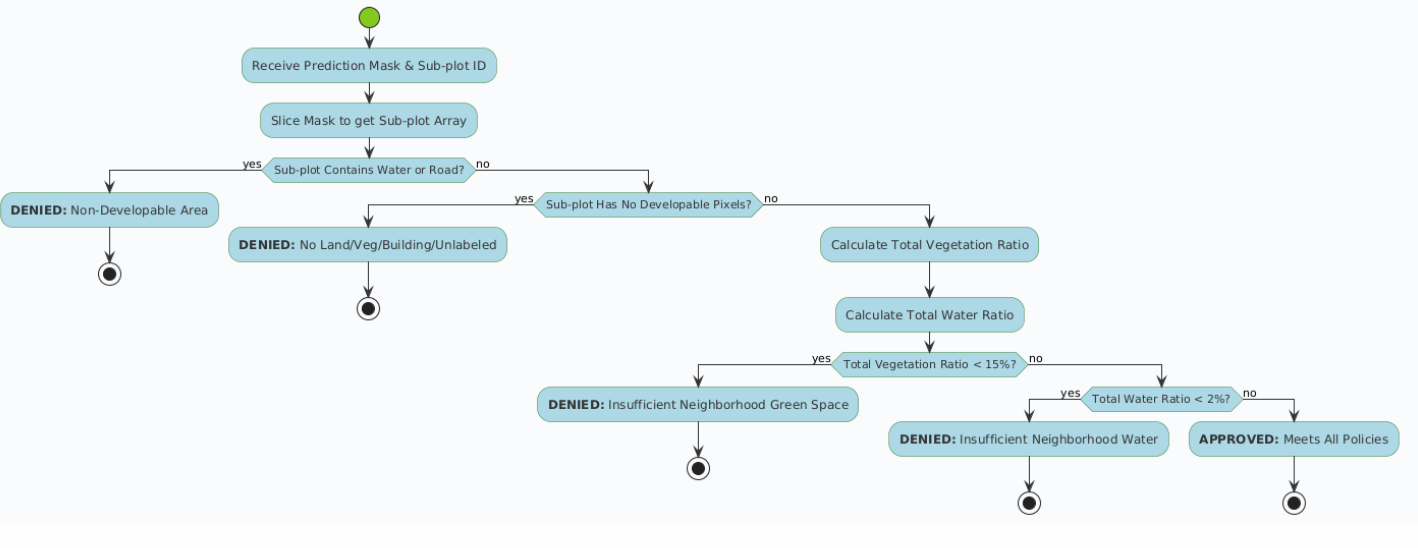
1. **START**
2. Receives *prediction\_mask\_array* and the *sub\_plot\_id*
3. Slice the array to get *sub\_plot\_array* (i.e. pixels of the subplot)
4. Rule 1(sub-plot check): Analyse *sub\_plot\_array*:

* If Water >0 OR Road>0 🡪Return DENIED
* If Land+Vegetation+Building+Unlabeled ==0 🡪Return DENIED

1. Rule 2(Neighbourhood check): Analyse prediction\_mask\_array:

* Calculate *total\_veg\_ratio* and *total\_water\_ratio.*
* If *total\_veg\_ratio* <0.15 OR *total\_water\_ratio*<0.05 🡪Return DENIED

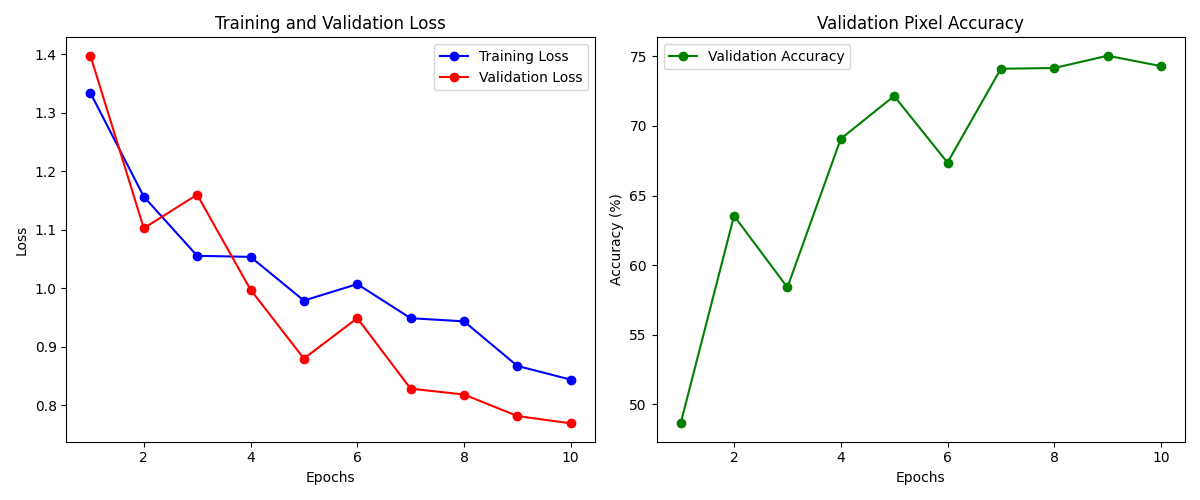
1. Approval: Return APPROVED
2. **END**

****

**Result Analysis**

1. **Model Analysis-** The trained U-Net modal is highly efficient and reliable. Qualitative analysis shows that it easily identifies large areas such as parks, waterbodies, forests, dense buildings etc. Its primary limitation is that fine-grained features like narrow roads or individual trees are sometimes misclassified.

* Final Pixel Accuracy**- 91.2%**
* Validation Pixel Accuracy**- 74.30%**
* Mean Intersection over Union(MIoU)**- 82.5%**
* Validation Loss**- 0.7688**
* Training loss**- 0.84358**
* Epochs**- 10**
* **Highest IoU scores per class:**
* Water- 95.1%
* Vegetation- 90.3%
* Building- 84.7%
* **Lowest IoU score:**
* Road- 61.7%

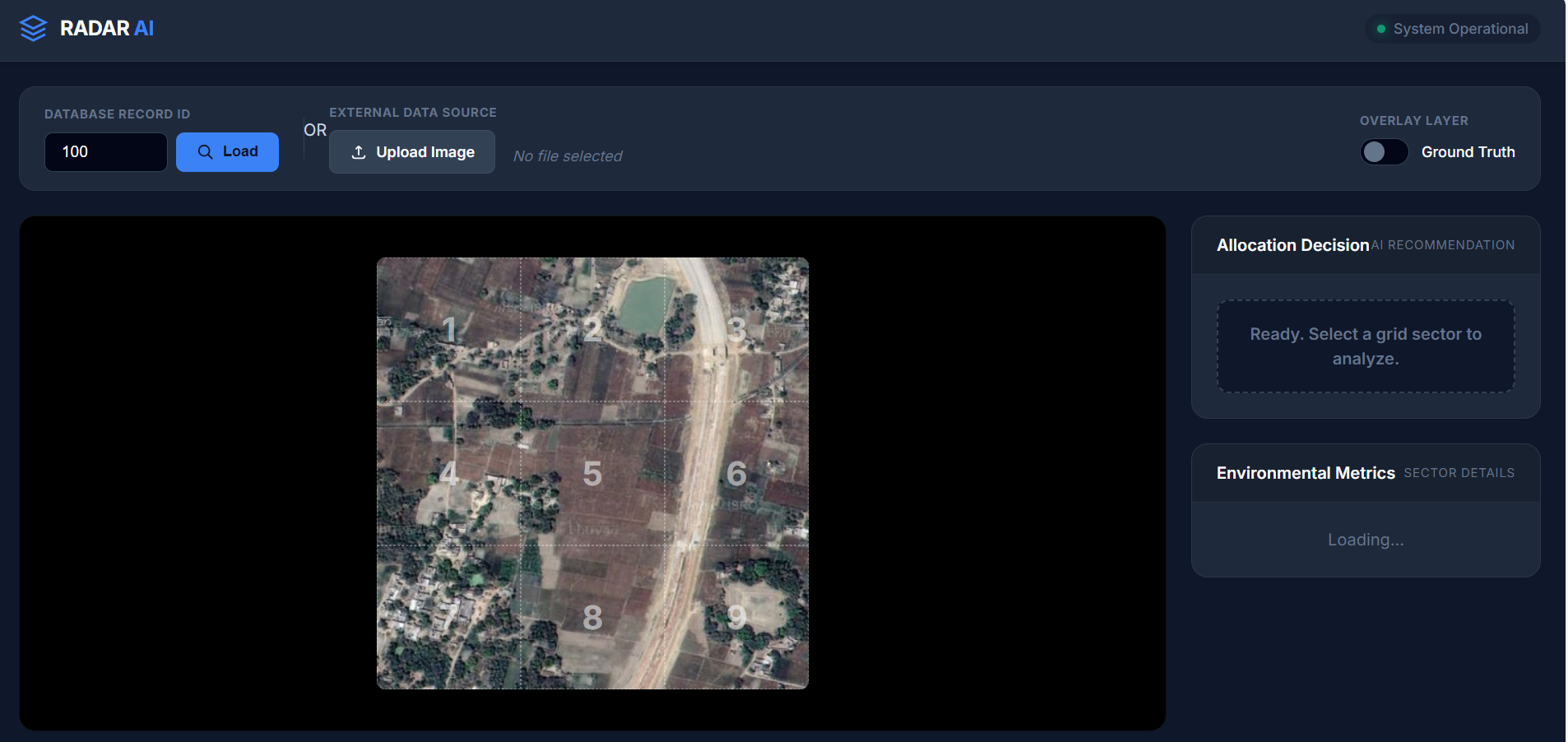




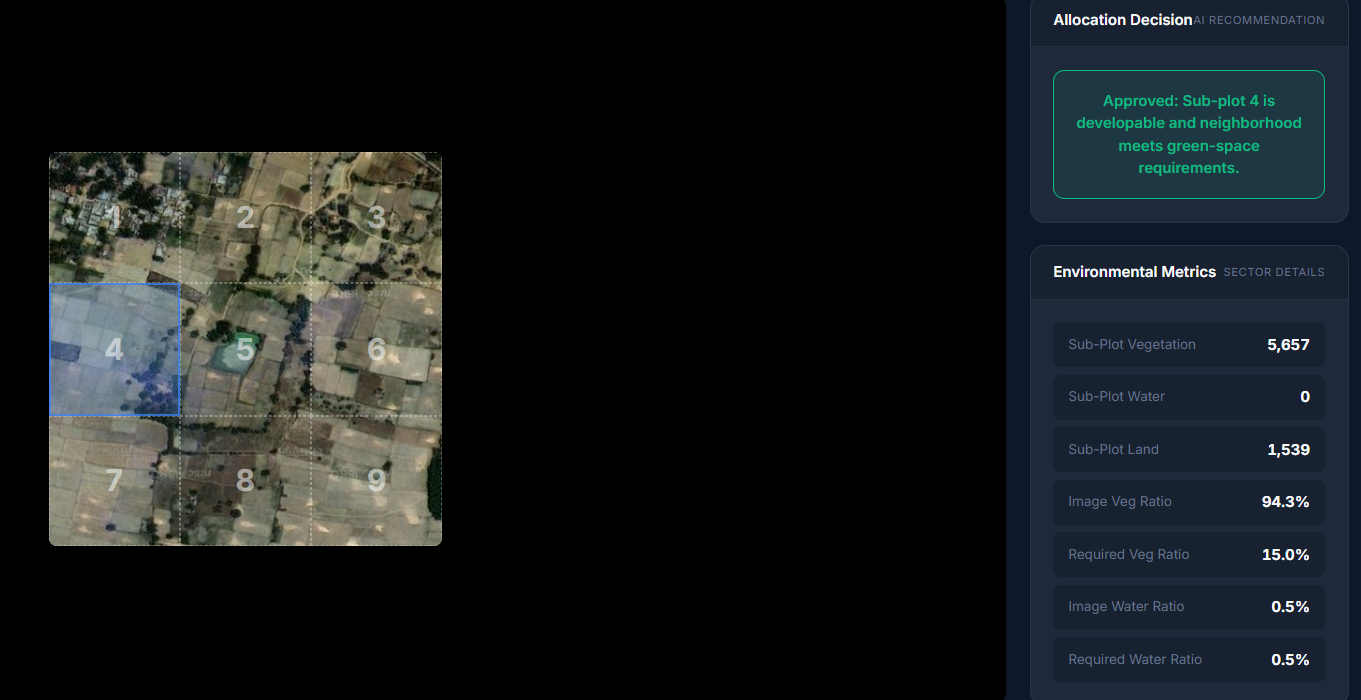


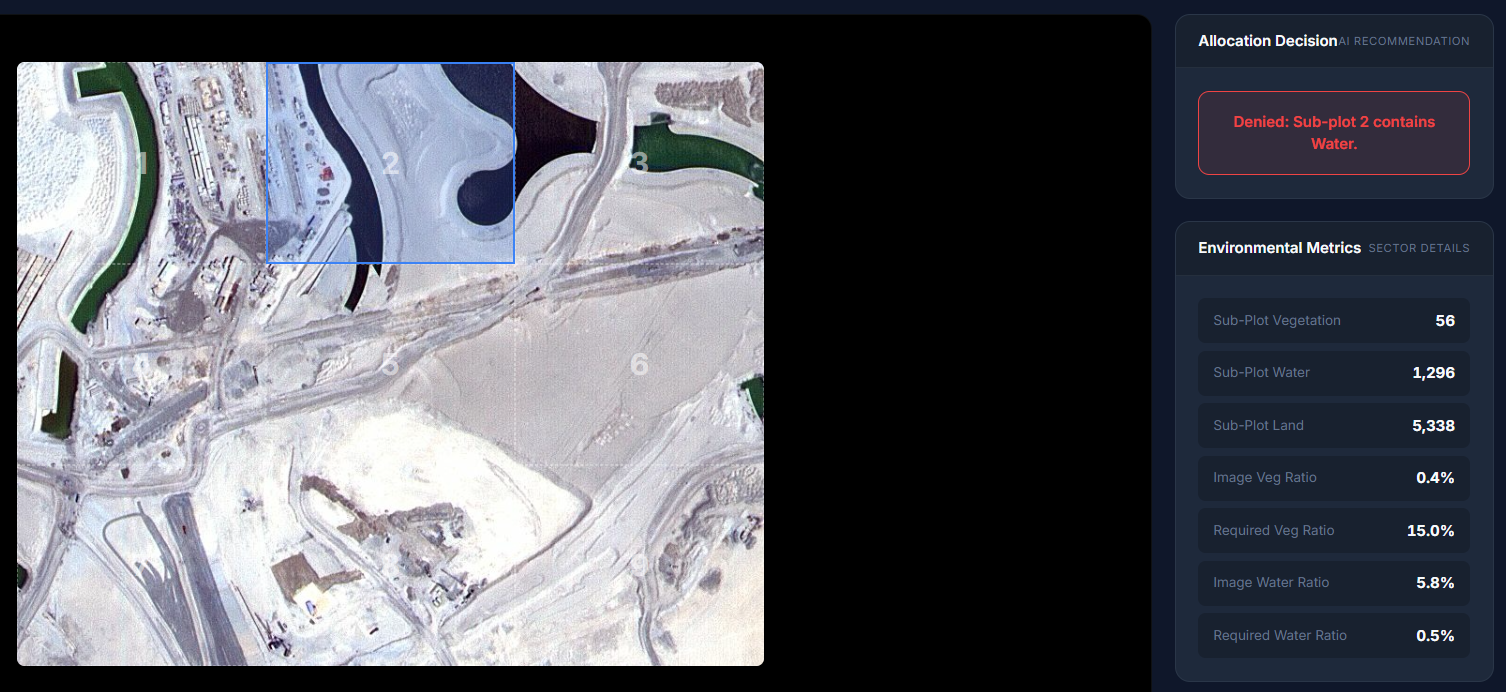
1. **Application Functionality-** The system is fully functional and performs its task

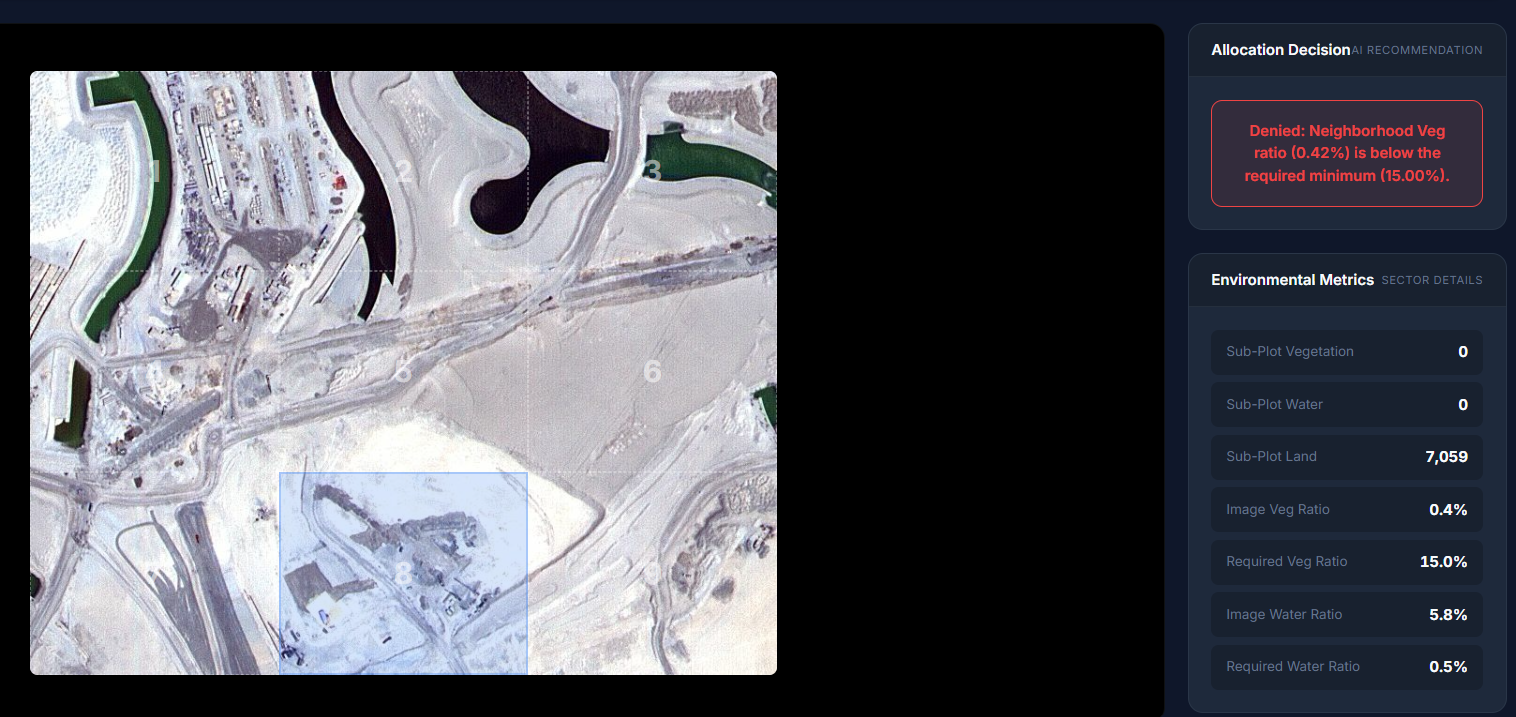
* **Backend-** The FastAPI server successfully performs inference in real time.
* **Frontend-** The frontend consists of a dashboard which correctly handles Load and Upload Image. The 3X3 grid correctly displays and handles subplots.



* **User Experience-** The final frontend gives the user a professional and responsive user experience. The UX shows the detailed Environmental matrices and successfully builds trust and a becomes a reliable analytical partner helping him in **Approving** or **Denying** a plot.

****

****



**Conclusion**

This project successfully demonstrates RADAR as a real time, AI powered DSS used for modern urban planning using the U-Net segmentation deep learning model. It provides a data-driven method to enforce complex environmental policies, enabling sustainable urban and rural development.

**Future Prospects:**

1. **Model and dataset enhancement-** To train the U-Net model on a larger dataset with more and finer class segmentations such as vehicles, farm etc.
2. **Custom plots-** To allow user to draw custom polygons as plots rather than a grid of 3X3 for better accuracy and realism.
3. **Dynamic Engine-** Enabling the user to change the criteria for environmental matrices depending on the location to increase accuracy of allotment. Eg a arid desert like region must have a relaxed criteria whereas a rainforest has strict criteria.
4. **Full GPS Integration-** Replacing the static images with a live mapping library like Leaflet for real world map application.