**Overview of project objectives: What problem(s) is the project trying to solve/address? What are you trying to do? For whom are you doing this? Why do they need it? What is the link to the corresponding challenge?**

**Script 1:**

This project aims to build a web-based system that can detect and predict Harmful Algal Blooms or HABs using remote sensing data and deep learning models.

The main issue we’re trying to solve is that current HAB detection methods, like lab-based sampling, are slow, manual, and very limited in both space and time. This makes it hard to act quickly when blooms are about to happen.

HABs can seriously impact the environment, human health, and the economy. They’ve caused mass fish deaths, toxic seafood, and even beach closures, with millions lost annually. And they’re becoming more frequent because of things like nutrient pollution and rising water temperatures.

So, with this project, we want to automate and improve how we track and predict these events, especially before they happen. That way, authorities and coastal industries can take action earlier and reduce damage.

Our work is based on HABNet, which uses spatiotemporal “datacubes” basically cubes of satellite data over time to train neural networks. It achieved over 90% accuracy in detection and solid predictive power up to 8 days in advance.

Our goal is to take this core idea and turn it into a functional, easy-to-use web tool that helps environmental agencies, researchers, and policymakers better manage HAB risks.

**• Technology stack: What are the key technologies you are using and why are they the best choices**

**for your project? What system architecture/infrastructure do you have?**

**Technology Stack**

Our technology choices were guided by the project’s needs and our goal of building a modular, scalable, and user-friendly HAB Detection System.

**1. Data Collection & Processing**

We used **Jupyter Notebooks** for initial data exploration and pipeline development due to their flexibility and strong support for Python libraries.

• For satellite data, we used the earthaccess library to connect with NASA’s MODIS datasets, and processed .nc files using netCDF4.

• Ground truth data was handled using pandas, and we serialised the resulting data cubes using pickle. While pickle worked well during prototyping, we plan to switch to HDF5 in future iterations for better performance (e.g., selective loading and concurrency).

**2. Machine Learning Framework**

Our data cube—a 15x15x5 structure capturing 5 days of chlorophyll anomalies—enabled temporal modelling.

• We experimented with tree-based models (e.g., **XGBoost**, which reached 94.7% accuracy) and deep learning models like **CNN + LSTM**, which capture both spatial and temporal patterns.

• Our final choice, **CNN + LSTM**, uses TimeDistributed layers to extract spatial features daily and an LSTM to track their evolution. This hybrid approach provides high accuracy and generalizability across unseen data points.

**3. Backend API (Model Inference)**

We deployed our trained model using a lightweight **Flask** API. Flask is ideal for single-endpoint microservices, making it a good fit for our REST interface.

• It receives user input (lat/lon, date), triggers the datacube generation and model inference, and responds with a simple JSON indicating whether the region is *Toxic* or *Non-Toxic*.

• It’s also container-friendly and ready for deployment on platforms like **Render** or **AWS**.

**4. Frontend (User Interaction)**

We built the user interface using **Streamlit**, as it allows for rapid development of interactive dashboards without requiring HTML or JavaScript.

• The app supports both manual input and map-based selection (via streamlit-folium), which makes it accessible to both technical and non-technical users.

• Once the user selects a location or enters data, the app formats this into a JSON request and sends it to the Flask backend for prediction.

This end-to-end stack ensures our HAB system is modular, scalable, and easy to maintain, from satellite data ingestion to interactive risk prediction.

**Data Sources: What are the key data resources you are using, and how are you accessing them, and**

**how will they support your application? How are you organising and storing data?**

**Ground Truth Data**

Initially, for our ground truth data, we considered using the HAEDAT, HAIS and Bloomin' Algae datasets, but we found that all of them primarily record positive HAB events. We decided to use the HABSOS Database, mainly because it covers the Karenia brevis algae in the Gulf of Mexico with their cell count, spatial and temporal data, just like the HABNet data had. It also has both positive and negative labelled samples, unlike the others, and Access to the data is very easy; we just need to download the CSV from their website. So far, we have processed 1368 events to use for our MVP.

**Satellite Remote Sensing Data**

We use our ground truth events longitude, latitude and date to get Remote Sensing Data from NASA MODIS-Aqua, which we are accessing using the Earth Access api, which has a rate limit of 1k requests a minute, but we only use a maximum of 500 per hour. The data from the satellite comes in granules that are saved as NC files on Daniel’s local system, and are cached so we can reprocess them later”

**DataCubes:**

Just like the HabNet paper, we build datacubes based on the satellite data. For now, we are only extracting the Chlorophyll-a concentration data from the granules and processing them into 30x30km x5 days datacubes leading up to an event, with a 2km resolution. Later, we will add other modalities such as PAR and rsr bands to them. We discard datacubes that have less than 60% coverage to have at least 3 days’ worth of data for training, and the datacubes are about 10kb in size

**• Evaluation: How will you evaluate your system? What question(s) are you trying to answer with**

**Your application, and how will you evaluate if it is successful?**

1. **Is Our HAB System Successful?**

“To evaluate our system, we broke the problem down into clear questions and then designed our metrics and tests to answer each of them. We’re not just checking whether the model works, we’re testing whether the whole system is robust, accurate, and helpful to real users.”

1. **What Question Are We Trying to Answer?**

We focused on four key evaluation questions:

1. Is our system accurately detecting toxic water?

This is the core goal: distinguish toxic vs. non-toxic regions reliably.

2. Can we predict future toxicity, not just detect existing ones? Important for early intervention and planning.

3. Is the system fast, scalable, and able to handle missing or noisy data?

Needed for real-world use.

4. Does our model perform better than traditional methods? Proves the value of using machine learning over simple thresholds.

1. **How Are We Evaluating It?**

1. Model Accuracy & Prediction Power (Technical Evaluation)

We compared multiple models using a labelled test set (from our held-out HABSOS data):

Takeaway: All ML models beat the traditional Chlorophyll-a thresholding method. CNN+LSTM gives the best temporal awareness. XGBoost gives the best raw accuracy, but CNN+LSTM generalises better across space and time.

1. **Real-World Functionality (System Evaluation)**

Datacube Flexibility:

Users can choose 1-day or 5-day predictions. We pad shorter sequences to allow consistent input.

Adaptable for casual users and researchers alike.

Prediction Speed:

Inference through Flask API takes under 3 seconds per location.

Can be optimised further using model quantisation or batched predictions.

Handling Missing Satellite Data:

If less than 60% of a cube is available, we discard it. This prevents garbage-in-garbage-out learning.

UI Flow (Streamlit Frontend):

Inputs get converted to JSON, and predictions are shown clearly as Toxic/Non-Toxic.

Usable without a technical background.

1. **Comparison to Baselines**

We also tested our models against the basic Chlorophyll-a threshold rule (e.g., flagging values > 1.0 as toxic):

Problems with the threshold method:

Misses subtle patterns (false negatives)

Overwarnings in normal zones (false positives)

Our CNN+LSTM model reduced both issues by learning spatial and temporal changes instead of hard-coded thresholds.

1. **User-Centred Success (What Makes It a “Good” System?)**

Early Detection: A 5-day forecast helps users and authorities act before a bloom hits.

Modular Design: The model, API, and frontend are decoupled, easy to scale or swap models.

Future Expansion Ready: We can add temperature, turbidity, or new regions without retraining the whole stack.

Feedback Loop: Future versions could use field feedback to improve predictions continuously.

1. **What Does “Success” Look Like?**

• If our system:

• Predicts toxicity 3–5 days in advance with ≥90% accuracy

• Handles live or missing satellite data gracefully

• Can be used by anyone with a map and a question

• Outperforms the traditional methods clearly…

Then, yes, we consider the system successful.

**Current status: Describe the progress you have made so far and explain why this constitutes a**

**MVP.**

As of now, we’ve successfully completed all the critical components that make up our Minimum Viable Product. The datacube generation pipeline is fully functional; we’re able to extract and structure chlorophyll-a data into 15×15×5 spatio-temporal blocks for each ground-truth event using real MODIS satellite data. These are being serialised efficiently and reused across our model training workflow.

On the machine learning front, we’ve trained and evaluated multiple models, including tree-based methods like XGBoost and deep learning architectures like CNN + LSTM. The best-performing CNN+LSTM model reaches over 90% accuracy and has shown strong generalizability across test events, which gives us confidence in its robustness.

Our backend is live and can take user input, trigger inference, and return a toxicity prediction. The Streamlit frontend allows users to either manually input coordinates or select locations via an interactive map. The system ties together seamlessly and is fully automated, from data to prediction.

This constitutes our MVP because every key functionality is working end-to-end: data collection, preprocessing, model inference, and user interface. While it may not be production-grade yet, the core proof of concept has been validated successfully and is ready for further iteration.