**1. Project Objectives**

The goal of this project is to develop an interactive web dashboard that visualizes Harmful Algal Bloom (HAB) data and integrates a Large Language Model (LLM) such as GPT-4 to provide human-readable explanations for bloom alerts. HABs are a growing threat to marine ecosystems, aquaculture operations, and public health, yet interpreting bloom predictions remains challenging for many stakeholders due to the technical nature of the data.

This project addresses that gap by combining dynamic visualizations with natural language explanations. The system will allow users—such as aquaculture managers and environmental researchers—to interact with time-series data (e.g., Chlorophyll-a levels, sea surface temperature) and spatial bloom alerts through an intuitive dashboard interface. The integration of an LLM will enable users to ask questions such as “Why is this site at high risk today?” and receive detailed explanations of contributing factors (e.g., rapid temperature rise or turbidity changes), along with possible mitigation steps.

The core deliverables of the project include:

* Interactive visualizations for time-series trends and geospatial risk mapping
* A conversational chat interface powered by an LLM for interpretability
* Options to download visual data (as PNG, JPG, etc.)
* A fully containerized frontend (React) and backend (Flask/Express) deployed using Docker

Ultimately, this dashboard aims to enhance the transparency, usability, and actionability of HAB prediction models. By making complex environmental data more understandable and accessible, the system supports informed, real-time decision-making for sustainable aquaculture and ecosystem management.

**2. Sprint Plan**The project will follow agile development principles, divided into weekly sprints aligned with module deadlines.   
  
Week 1–2: Team formation, project scoping, and data source research  
Week 3: Initial UI mockups and backend API structure draft  
Week 4: Build time-series and map visualizations using dummy data  
Week 5: Integrate LLM interface and prompt engineering  
Week 6: MVP completion, usability test with peers, and interim presentation prep  
Week 7–8: Refine UI/UX, handle real-time data slices, add download options  
Week 9: Technical and user evaluation setup and testing  
Week 10: Final presentation and demo prep  
Week 11–12: Report writing, fine-tuning, deployment and submission

Sprint tracking will be done via Trello and GitHub

**3. Roles**

* **Project Manager** – Harshal More  
  Coordinates team meetings, maintains sprint tracking, and ensures deadlines are met.
* **Frontend Developer** – Anuj Lamba  
  Responsible for building the dashboard interface using React, Plotly.js for time-series visualizations, and Leaflet.js for spatial alert maps.
* **Backend Developer** – Shikhar Singh Negi  
  Sets up the Flask/Express backend, implements REST APIs, and integrates with the frontend and LLM components.
* **LLM Integration Lead** – Khush Poddar  
  Crafts LLM prompt templates, handles communication with GPT-4 or similar models, and ensures accurate, meaningful explanations.
* **Data Engineer** – Omkar Sanwatsarkar  
  Manages data collection, preprocessing, merging, and cleaning scripts to prepare structured inputs for the dashboard and LLM prompts.
* **DevOps & Deployment** – All Members  
  Collaboratively manage Docker setup for frontend and backend, CI/CD via GitHub Actions, and deployment to Vercel/Render.

**4. Architectural Setup**

* **Frontend**: React app hosted on Vercel
* **Backend**: Flask API deployed on Render (handles LLM queries, data endpoints)
* **LLM**: OpenAI GPT-4 API with limited free credits and cost controls; fallback to Hugging Face-hosted LLMs
* **Data Storage**: Static CSVs for sensor data; MongoDB Atlas (free tier) for query logging
* **Visualization**: Plotly.js for time-series, Leaflet.js for spatial alerts
* **Containerization**: Docker for both frontend and backend
* **CI/CD**: GitHub + GitHub Actions for auto-deployment on push to main

A diagram of a process flow

AI-generated content may be incorrect.

**Figure 1: Architectural Diagram**

**5. Data Plan**

This project will use publicly available environmental datasets that contain parameters linked to Harmful Algal Bloom (HAB) events. These datasets are sufficient for both building feature-based explanations via LLMs and powering visual insights on the dashboard. No private or sensitive data will be used.

**Data Sources**

* **HAEDAT** (Harmful Algal Event Database):  
  Provides records of bloom occurrences with location, date, and description.  
  🔗 <https://haedat.iode.org>
* **HAIS** (Harmful Algal Information System):  
  Offers time-series data on environmental variables such as Sea Surface Temperature (SST), Chlorophyll-a (Chl-a), salinity, and turbidity.  
  🔗 <https://data.hais.ioc-unesco.org>
* **Bloomin’ Algae**:  
  Community-reported observations of bloom events, including photos, GPS location, and severity indicators.  
  🔗 <https://www.ceh.ac.uk/our-science/projects/bloomin-algae>

**Data Format and Structure**

The data will be compiled in **CSV/JSON format** and structured with the following fields:

* timestamp (date)
* latitude, longitude
* chlorophyll\_a (µg/L)
* sea\_surface\_temperature (°C)
* turbidity (NTU)
* bloom\_label (1 = bloom, 0 = no bloom)

Processed datasets will be used to generate:

* Time-series charts for Chl-a, SST, and turbidity
* Spatial bloom risk maps
* Prompt inputs for the LLM to generate explanations

**Data Generation and Processing**

Python scripts will be used to:

* Clean and merge datasets
* Handle missing values
* Normalize and prepare fields for plotting and LLM use
* Simulate “live” daily data slices by extracting time windows from historical data

**Evaluation Data Usage**

The processed data will directly support evaluation tasks:

* **LLM Accuracy**: Each explanation must reference the top 1–2 environmental contributors (e.g., "High SST and turbidity").
* **Usability**: Testers will ask the chatbot “Why is this site at high risk?” and must receive interpretable, actionable outputs in <3 minutes.
* **Performance**: System must retrieve and explain a prediction in <5 seconds on average.

**Ethics and Sufficiency**

All data is **publicly accessible** and used strictly for non-commercial, educational purposes. The datasets span multiple regions and years, ensuring sufficient variation for robust testing, realistic dashboard simulation, and reliable feature-based analysis.

**6. GitHub Usage**

* Central GitHub repo for all code, updated regularly
* Commit standards and branch workflows will be enforced
* Team members must push design docs, minutes, diagrams, code, and testing data continuously
* Repository Link : <https://github.com/KhushPoddar11/CloudForgeX6-LLM-Dashboard-for-HAB-Alerts>

**7. Team Management**

* **Meetings**: Twice-weekly standups + ad hoc pair programming
* **Tools**: GitHub Projects, Zoom/Google Meet
* **Minutes**: Shared via Google Docs and committed weekly to GitHub

**8. Evaluation Plan**

* **Usability**: Peer testing with a timed task (e.g., ask for explanation at site A)
* **Performance**: Dashboard loads <2s; LLM response <5s
* **Accuracy**: LLM outputs must cite top 2 driving features per site, verified against known HAB drivers
* **Engagement**: Feedback will be collected to iteratively improve explanations

**9. Risk and Mitigation**

|  |  |  |
| --- | --- | --- |
| Risk | Impact | Mitigation |
| API costs (OpenAI) | Medium | Use free tokens for development, monitor usage, switch to Hugging Face if limits exceeded |
| Data access delays | Low | Use cached datasets and simulate live slices |
| LLM misinterpretation | Medium | Refine prompts; add fallback explanations |
| Team scheduling | Low | Fixed standup slots and async comms tools |