2025

EarthScape - User Guide

**A tree in a bubble

AI-generated content may be incorrect.**

Table of Contents

[**Student Information** 4](#_Toc208242444)

[**Management** 4](#_Toc208242445)

[**1. User Guide** 6](#_Toc208242446)

[**1.1 Authentication and Authorization** 6](#_Toc208242447)

[**Registering as a User (Analyst)** 6](#_Toc208242448)

[**Logging In** 7](#_Toc208242449)

[**Access Control** 9](#_Toc208242450)

[**Logging Out** 10](#_Toc208242451)

[**1.2 Data Ingestion** 10](#_Toc208242452)

[**Purpose** 10](#_Toc208242453)

[**Supported Dataset Types** 11](#_Toc208242454)

[**Steps for Data Ingestion** 11](#_Toc208242455)

[**Dataset Management** 16](#_Toc208242456)

[**1.3 Data Storage** 19](#_Toc208242457)

[**Implementation** 19](#_Toc208242458)

[**Optimization Strategies** 21](#_Toc208242459)

[**1.4 Data Processing** 23](#_Toc208242460)

[**1.5 Real-time Data Processing** 27](#_Toc208242461)

[**Steps** 27](#_Toc208242462)

[**Access Control** 30](#_Toc208242463)

[**Error Handling** 30](#_Toc208242464)

[**1.6 Anomaly Training** 30](#_Toc208242465)

[**Steps** 30](#_Toc208242466)

[**1.7 Forecast Training** 33](#_Toc208242467)

[**Steps** 33](#_Toc208242468)

[**Access Control** 36](#_Toc208242469)

[**Error Handling** 36](#_Toc208242470)

[**1.8 Correlation Analysis** 37](#_Toc208242471)

[**Features** 37](#_Toc208242472)

[**Access Control** 40](#_Toc208242473)

[**Error Handling** 40](#_Toc208242474)

[**1.10 Weather Details** 40](#_Toc208242475)

[**Features** 40](#_Toc208242476)

[**Access Control** 42](#_Toc208242477)

[**Error Handling** 42](#_Toc208242478)

[**1.11 Weather ChatBot** 42](#_Toc208242479)

[**Features** 43](#_Toc208242480)

[**Access Control** 44](#_Toc208242481)

[**1.12 Admin Management** 44](#_Toc208242482)

[**Features** 45](#_Toc208242483)

[**Access Control** 48](#_Toc208242484)

[**Error Handling** 48](#_Toc208242485)

[**1.13 Feedback Management** 48](#_Toc208242486)

[**Features** 48](#_Toc208242487)

[**Access Control** 52](#_Toc208242488)

[**Error Handling** 52](#_Toc208242489)

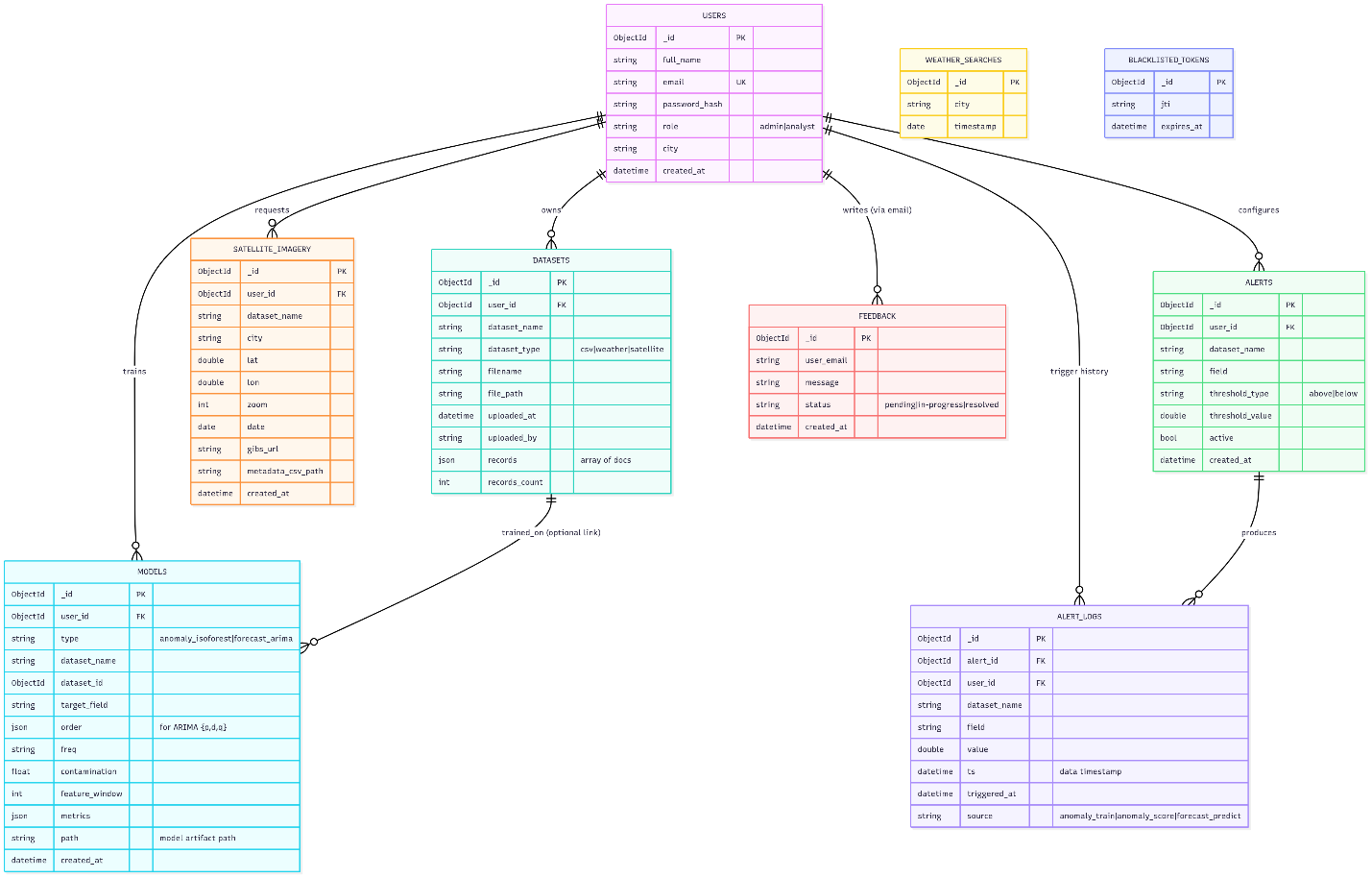
# **Student Information**

|  |  |
| --- | --- |
| Student ID | Student Name |
| Student1420981 | **Muhammad Saad Karim** |
| Student1421220 | **Tayyaba** |
| Student1368985 | **Mahnoor** |
| Student1418404 | **Laraib** |
| Student1335767 | **Lamia Raza** |
| Student1418410 | **Albina** |

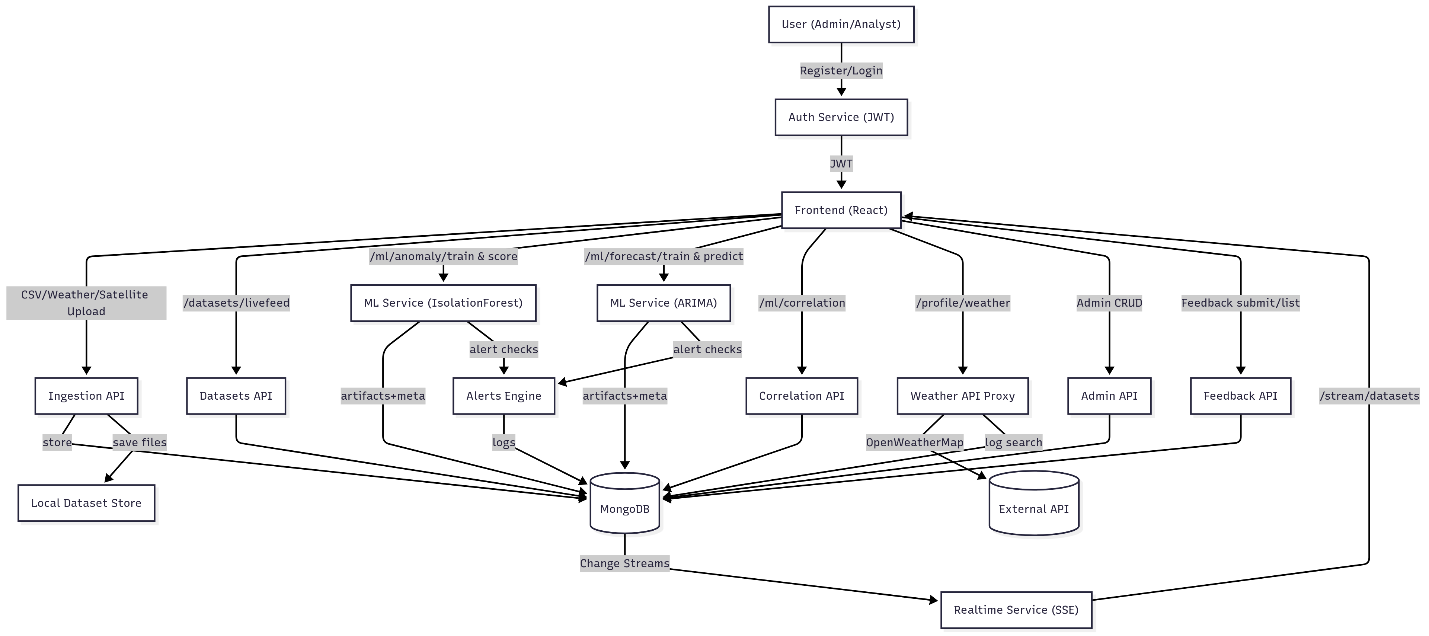
# **Management**

|  |  |
| --- | --- |
| Role | Student Name |
| Faculty | **Mam Wajiha** |
| Coordinator | **Mam Sana** |

**ERD**

****

**System Flow**

****

**Real-Time SSE Sequence**

**A diagram of a computer

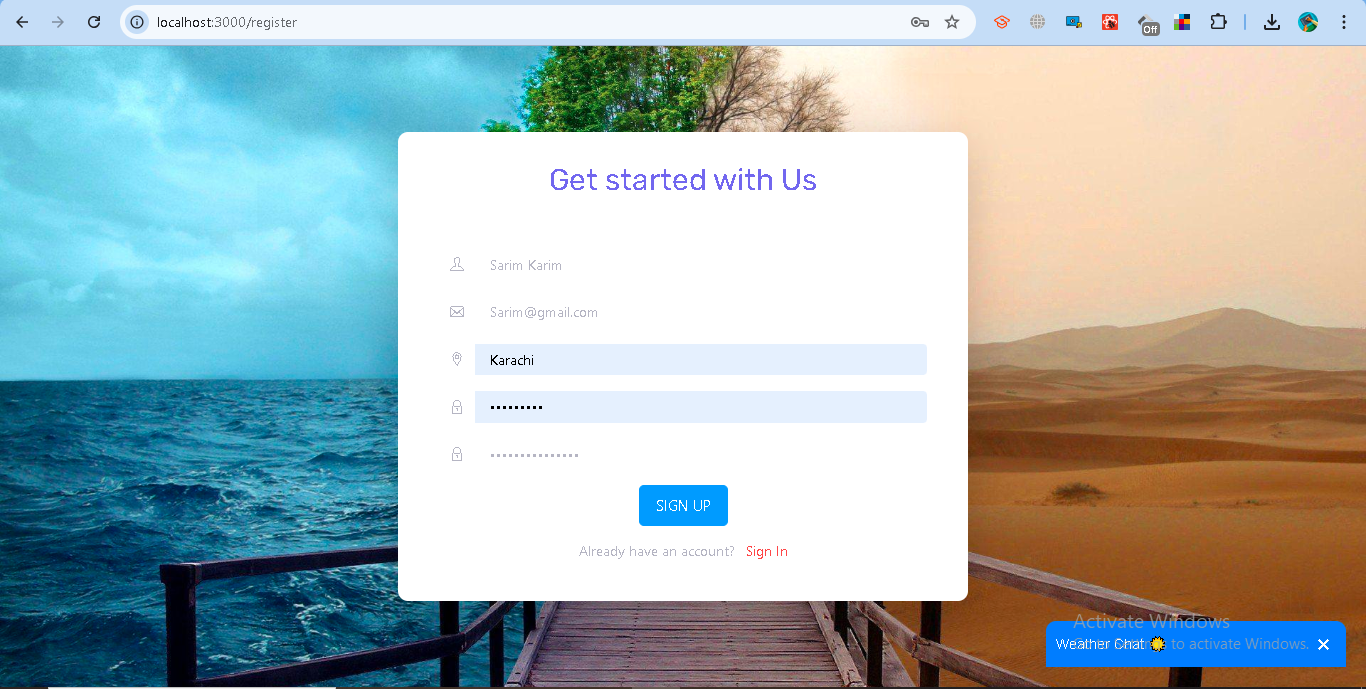
AI-generated content may be incorrect.**

# **1. User Guide**

## **1.1 Authentication and Authorization**

The system uses a secure authentication process with role-based access control (RBAC). It ensures that administrators have full rights, while analysts (normal users) have restricted access.

### **Registering as a User (Analyst)**

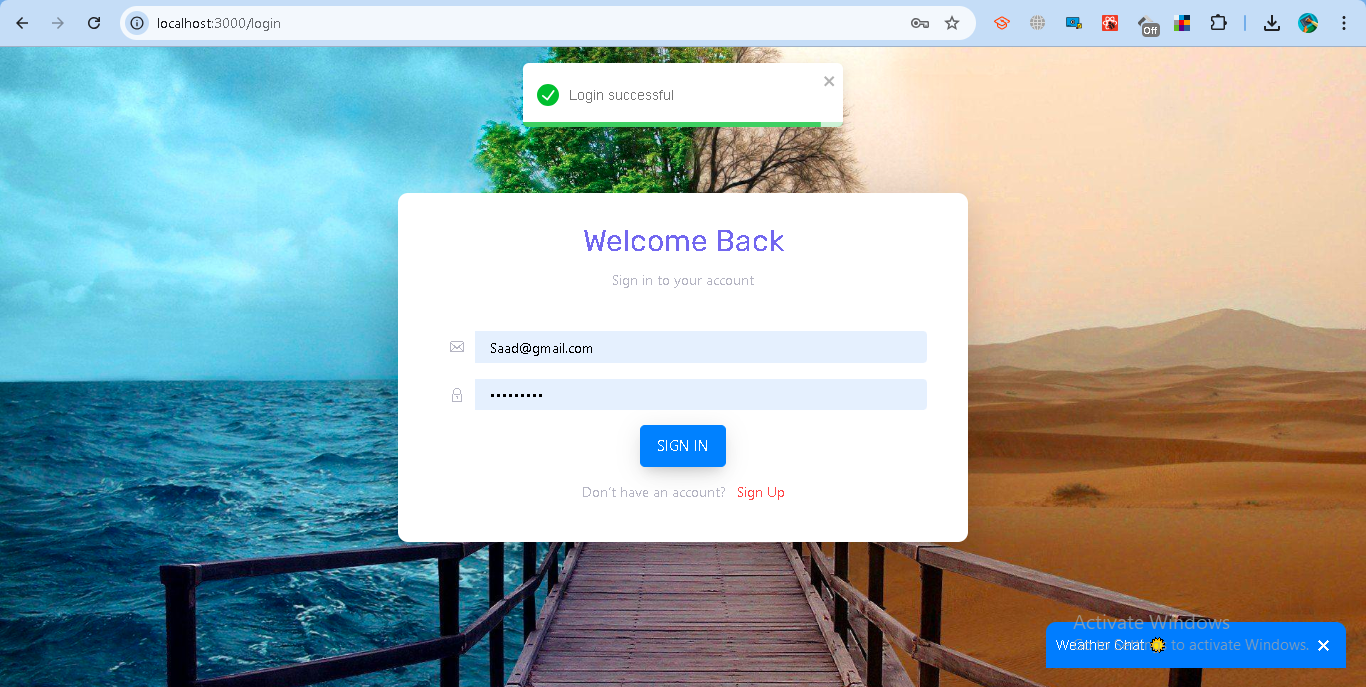


* Go to the registration page.
* Provide:  
  + Full Name
  + Email Address (must be valid format, e.g., user@example.com)
  + Password (minimum 6 characters, stored securely with bcrypt hashing)
  + City (for weather API integration)
* After successful registration:  
  + Your account is created with the role analyst.

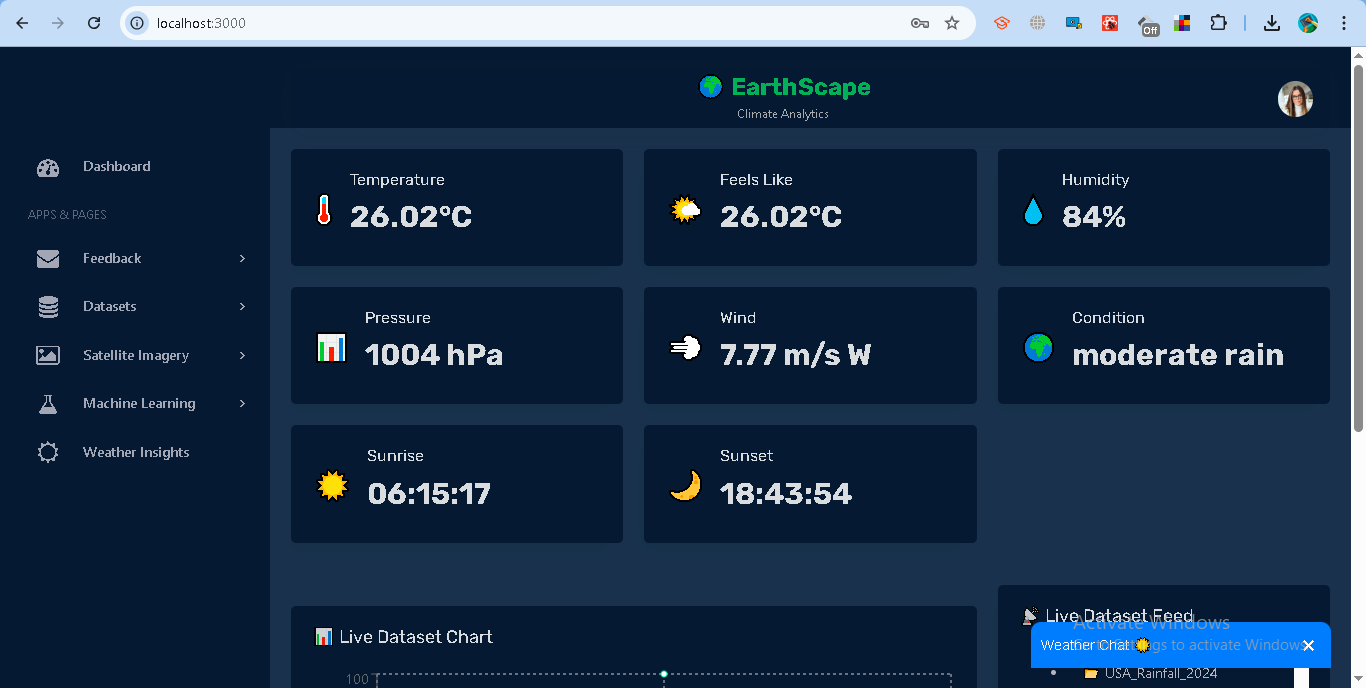
**A confirmation message is shown:**  
  
 Analyst registered successfully

By default, new users are not administrators. Only system-defined admins have elevated privileges

### **Logging In**

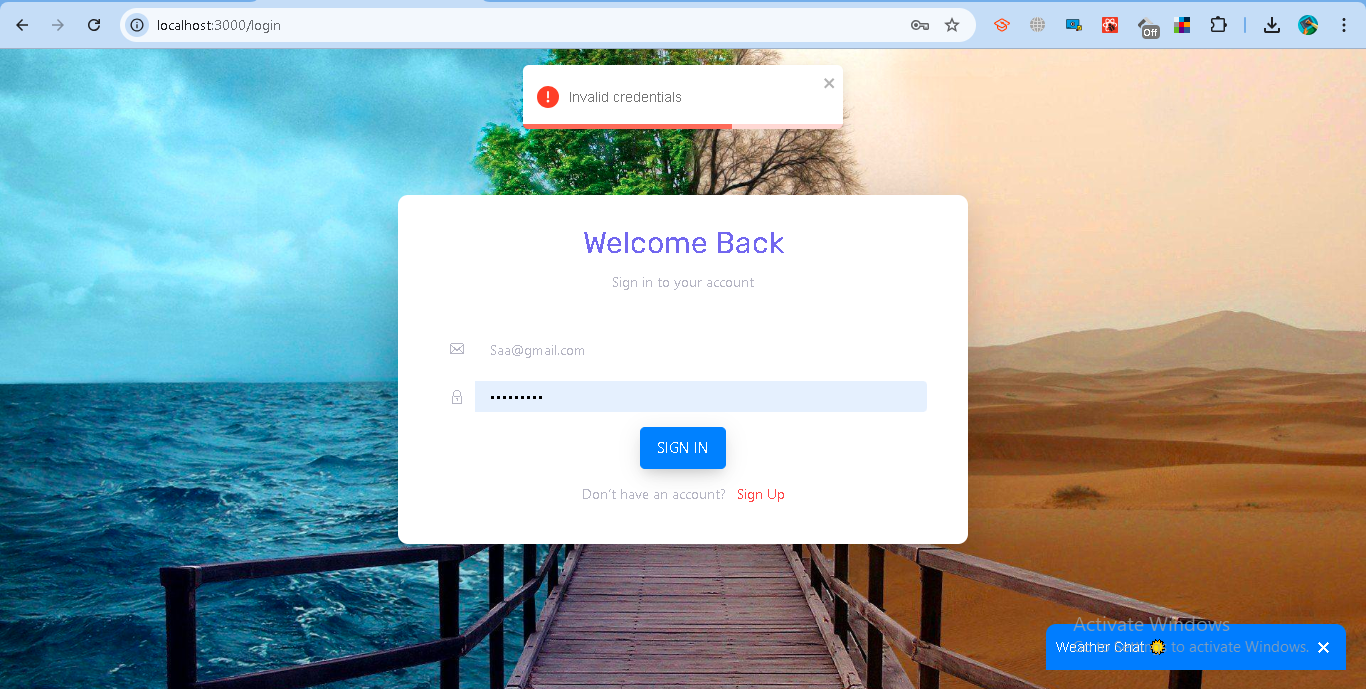


* Go to the login page.
* Enter your registered email and password.
* If valid:  
  + The server generates a JWT token with:  
    - user\_id (your unique ID)
    - email
    - role (admin or analyst)
    - exp (expiry: 2 hours)
  + You will be redirected:  
    - Analyst → Dashboard (/)
    - Admin → Admin Panel (/)



* If invalid:

**Error message is shown:**  
  
 Invalid credentials

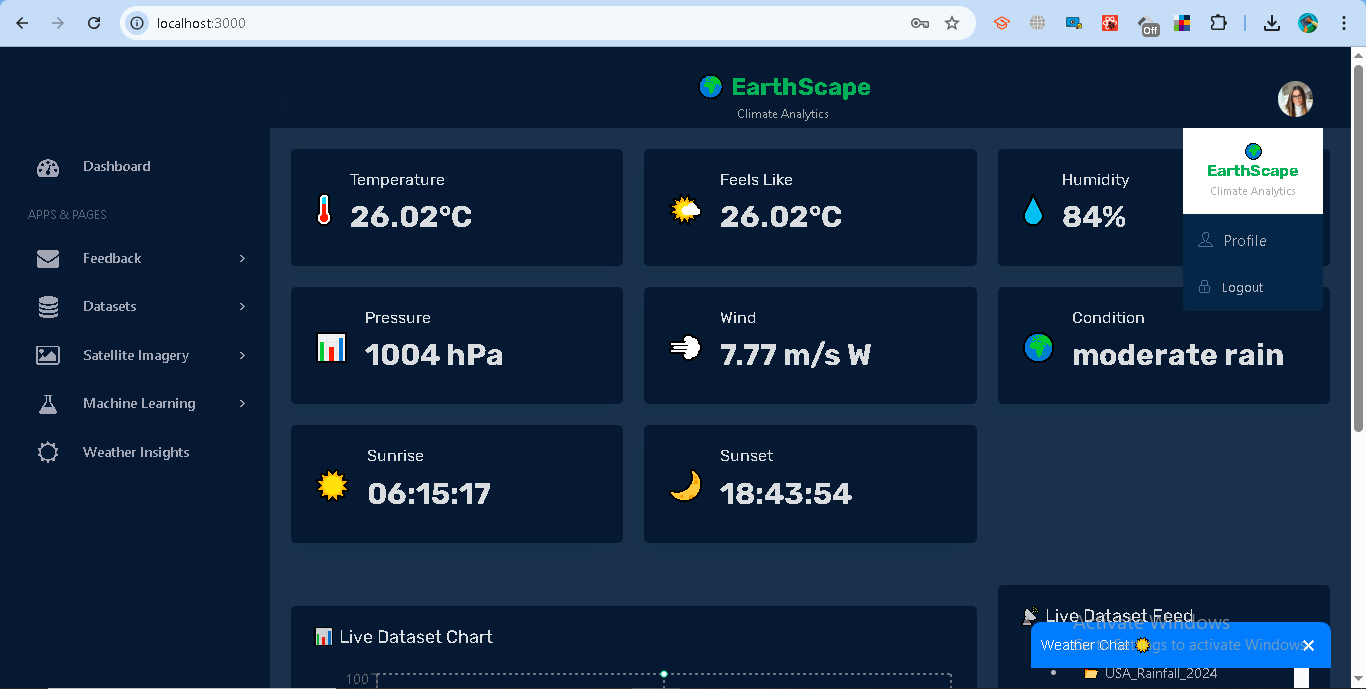


### **Access Control**

* Analyst (Normal User)  
  + Can log in and view their profile.
  + Can fetch weather details based on their city.
  + Can update their own profile information (name, email, password, city).
  + Cannot manage users or datasets.
* Administrator (Admin)  
  + Has all rights, including:  
    - Managing users (add/remove/update).
    - Accessing the Admin Panel.
    - Adding datasets.
    - Configuring alerts.
  + Can also perform all analyst actions.

### **Logging Out**

* When logging out:  
  + Your JWT token is blacklisted and cannot be reused.



**Response from backend:**  
  
 Logged out successfully

This ensures secure login, session management, and role-based permissions. Users cannot escalate privileges since role is assigned at registration and enforced in the backend.

# **1.2 Data Ingestion**

### **Purpose**

The data ingestion module allows users to upload and manage climate-related datasets such as CSV files, satellite imagery, and weather records. This ensures that both historical and real-time data sources can be stored, processed, and later analyzed.

### **Supported Dataset Types**

1. **CSV Upload** Users can upload their own structured datasets in .csv format.
2. **Satellite Imagery** Users can fetch and upload remote sensing imagery based on location (latitude/longitude) and zoom level.
3. **Weather Data (Historical)** Users can pull climate records (temperature, humidity, wind speed, precipitation, etc.) from the Meteostat API by specifying a city and date range.

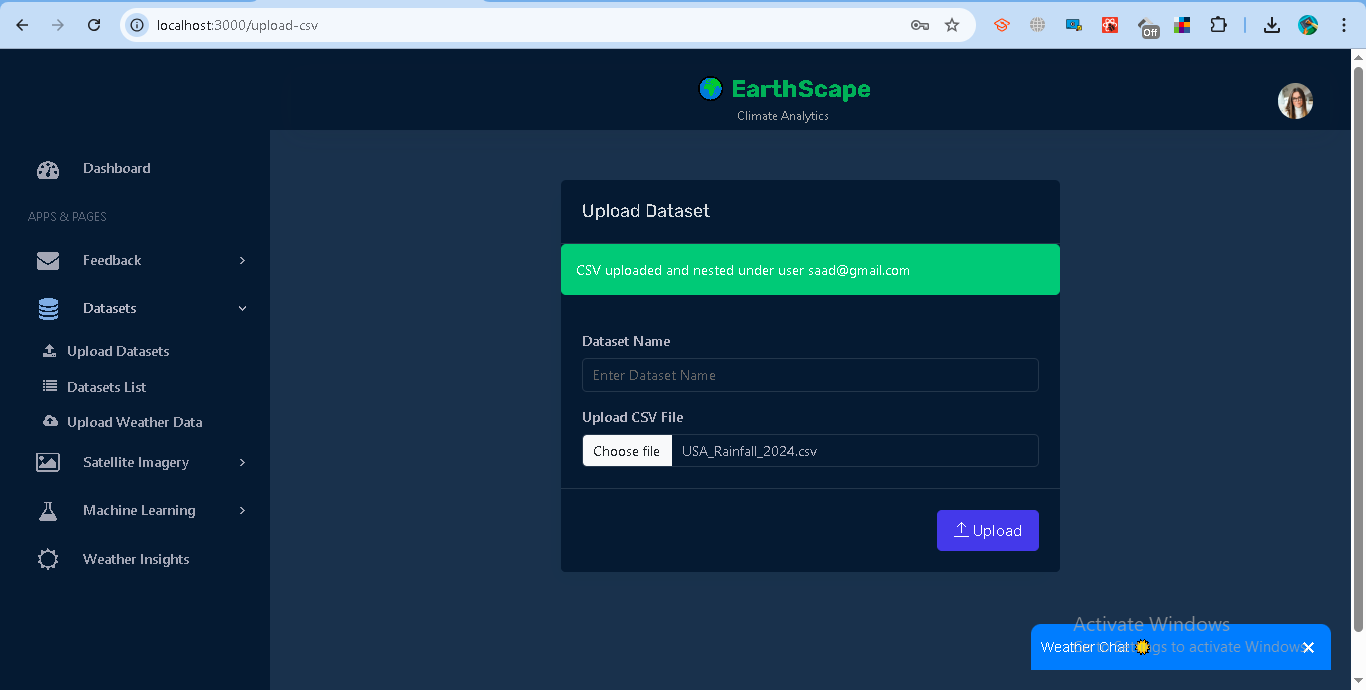
### **Steps for Data Ingestion**

#### **A. Uploading a CSV Dataset**

1. Login to the system using your credentials.
2. Go to **Dataset → Upload Dataset → CSV.**
3. Enter a dataset name (example: USA\_Rainfall\_2024).
4. Select and upload a .csv file containing your climate data.

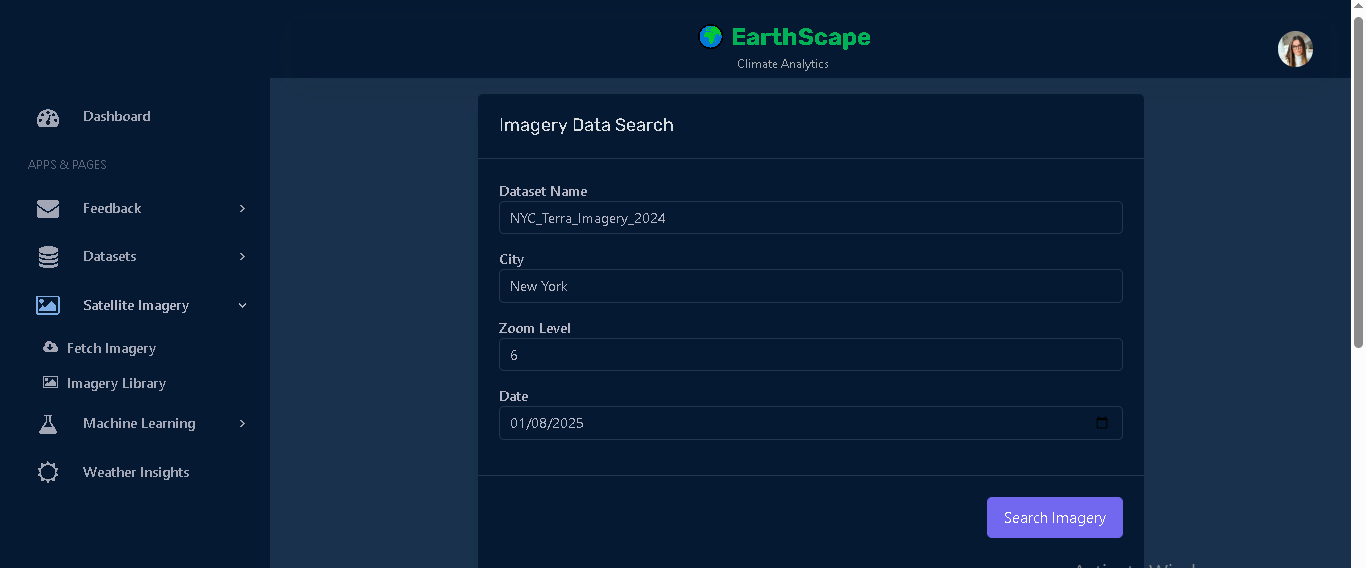


1. The system will:  
   * Validate the file.
   * Save records in the database.
   * Provide feedback on the number of records uploaded.
2. A confirmation message will be displayed:  
     
     
    *“CSV uploaded and nested under user [your email].”*

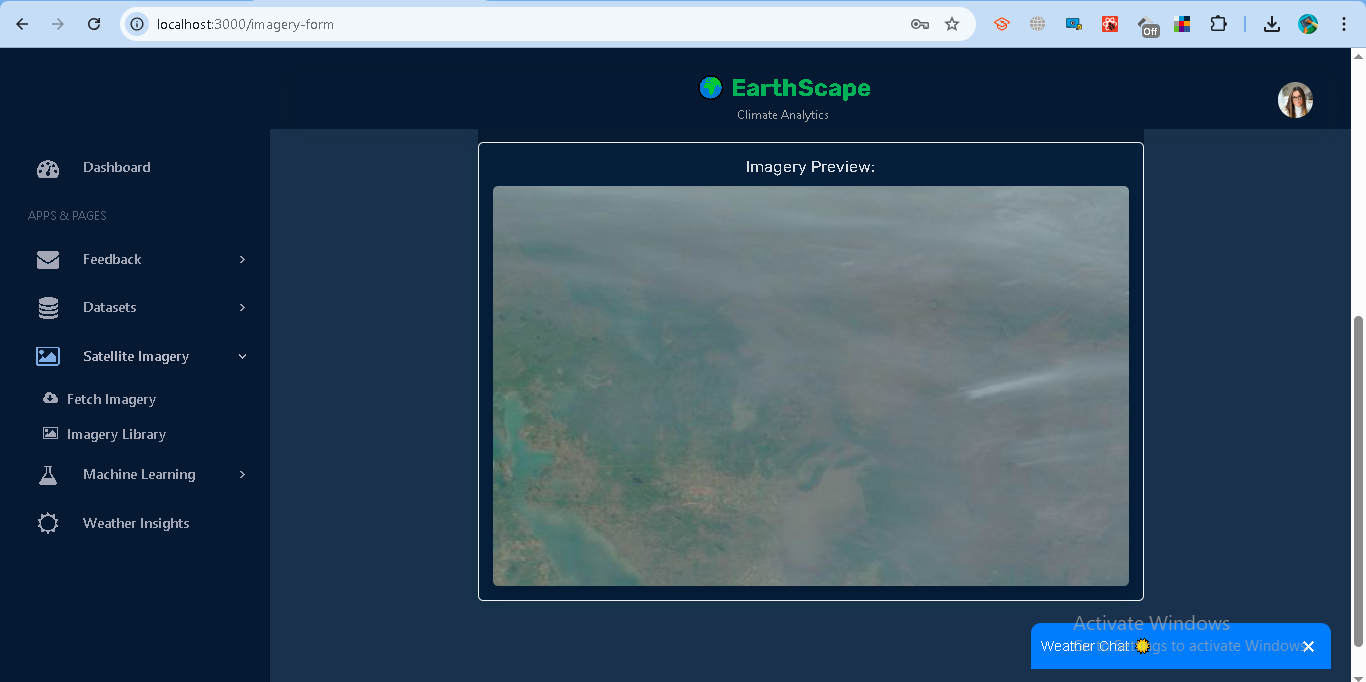
**

#### **B. Uploading Satellite Imagery**

1. Go to **Satellite Imagery→ Fetch Imagery.**
2. Provide:  
   * Dataset name.
   * City name.
   * Latitude and Longitude coordinates.
   * Zoom level (higher = more detail).
   * Date (or leave blank to use today).



1. The system converts your location into WMTS tile coordinates and fetches an image from NASA’s GIBS API.
2. Data is stored with metadata (location, date, zoom, and GIBS preview URL).
3. A CSV file with metadata is saved, and a confirmation message appears:

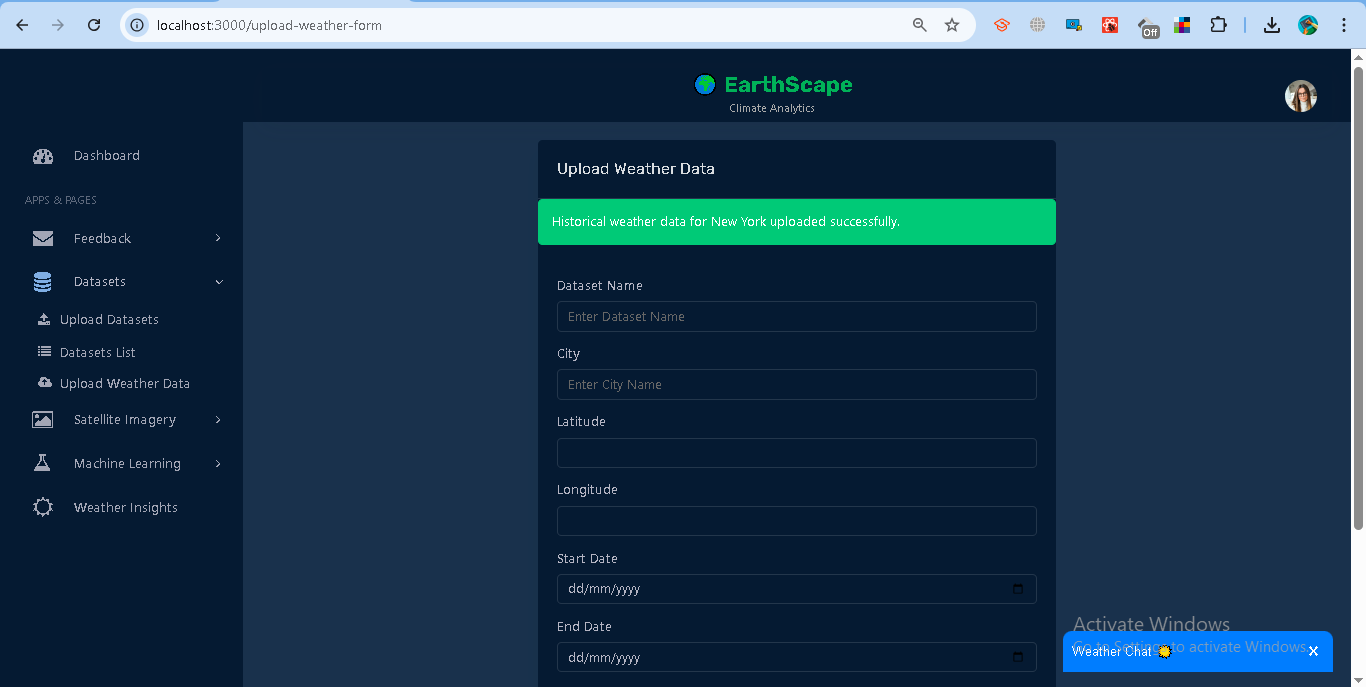
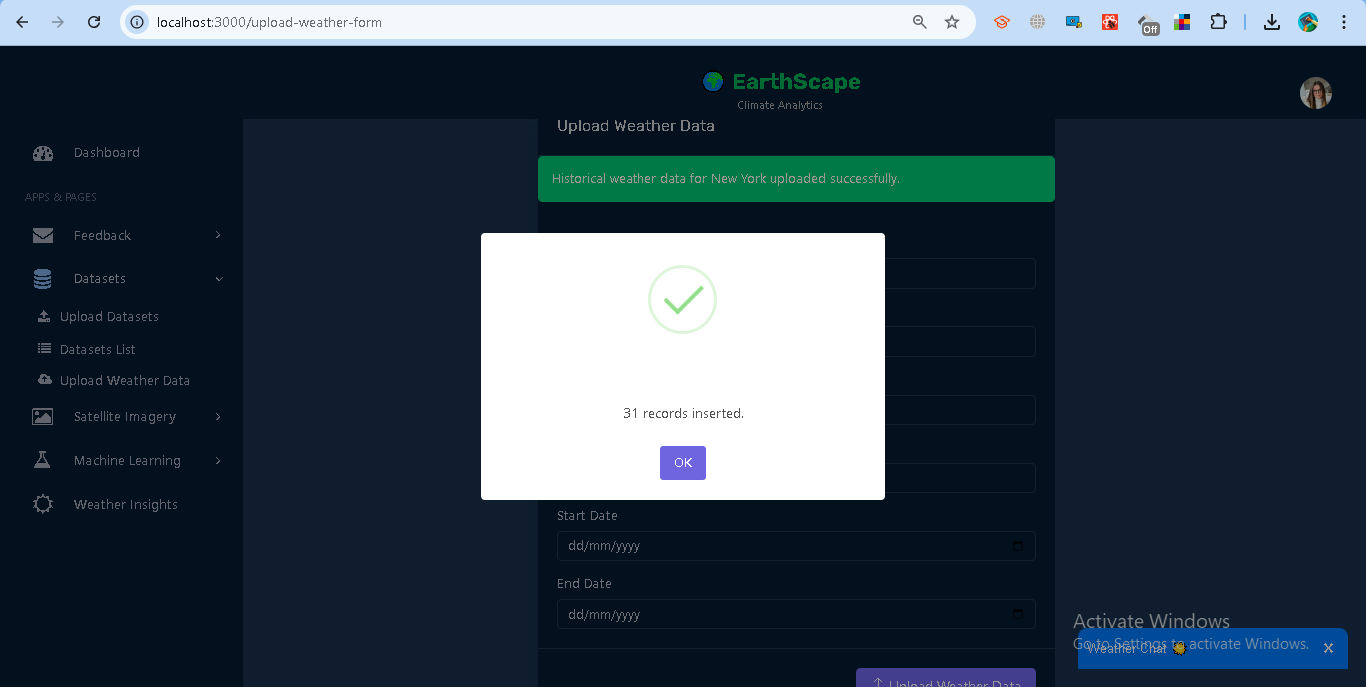


#### **C. Uploading Weather Data**

1. Go to **Dataset → Uploade Weather Data**.
2. Provide:  
   * Dataset name.
   * City name.
   * Start and End date.

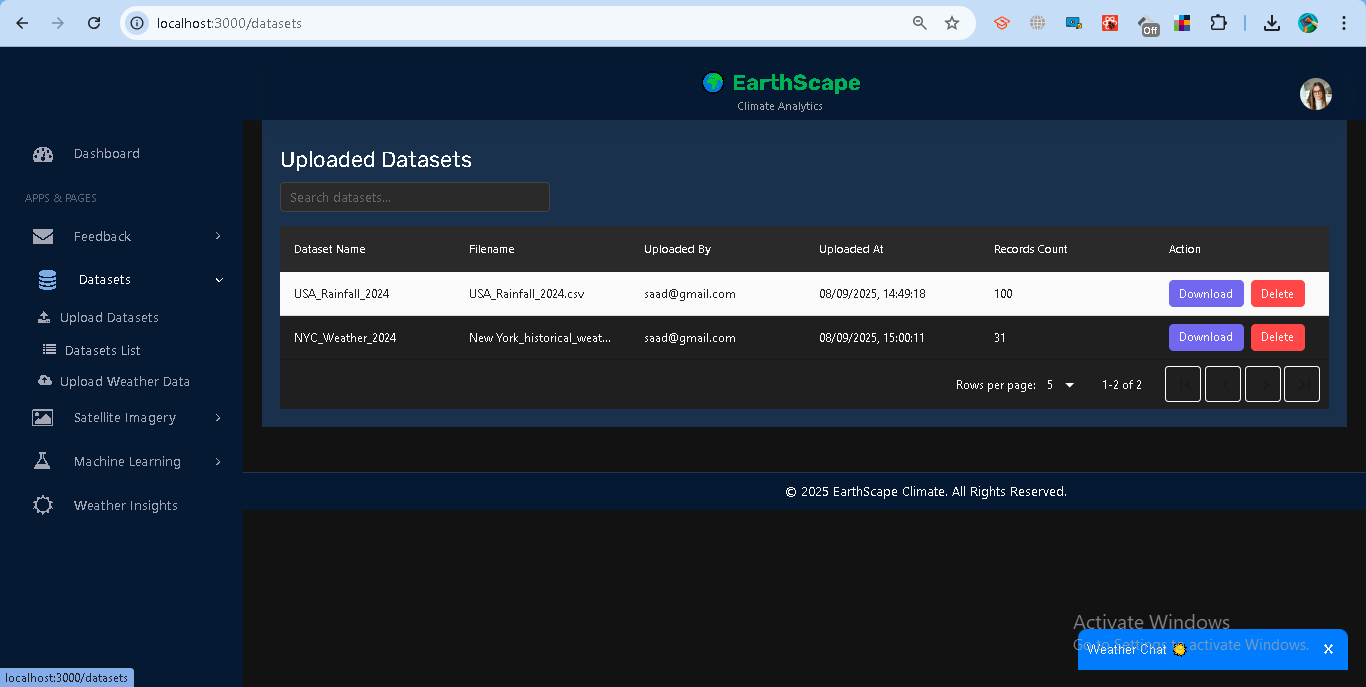


1. The system queries Meteostat for daily climate records.
2. Records include: temperature, humidity, pressure, wind speed, and precipitation.
3. A .csv file with the results is stored and available for download.
4. Confirmation message appears:

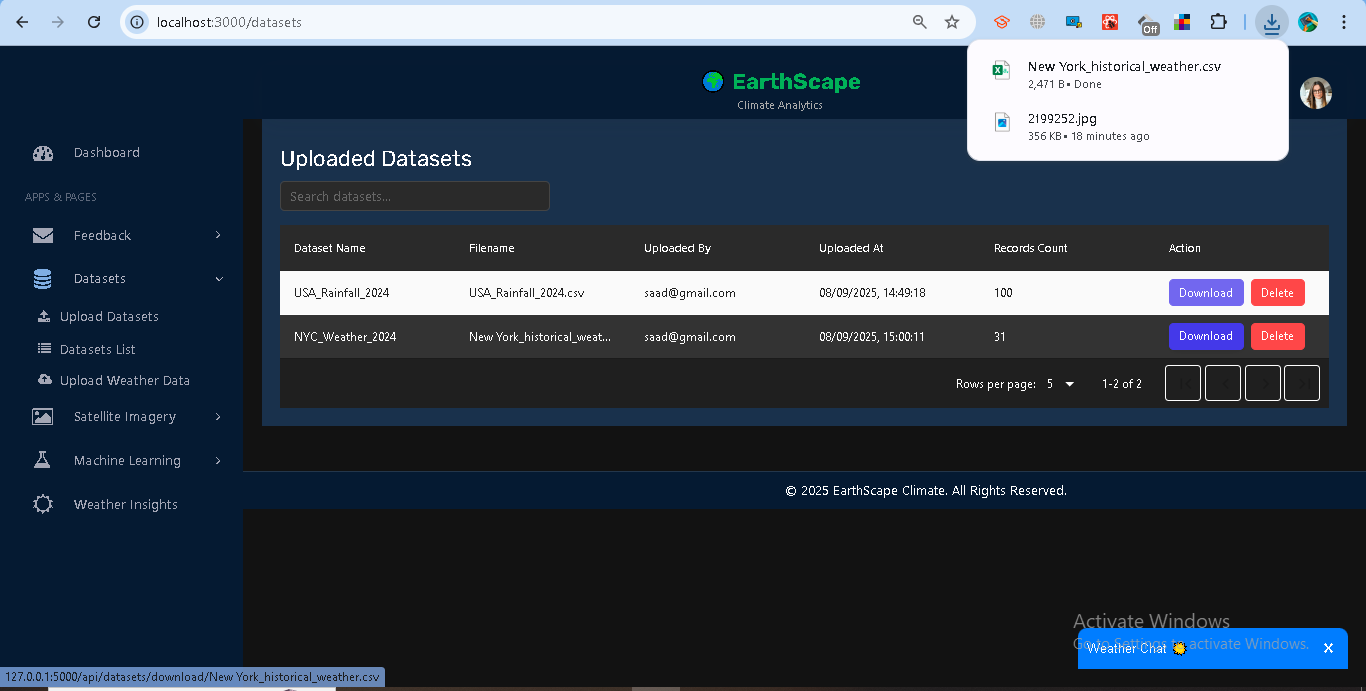


### **Dataset Management**

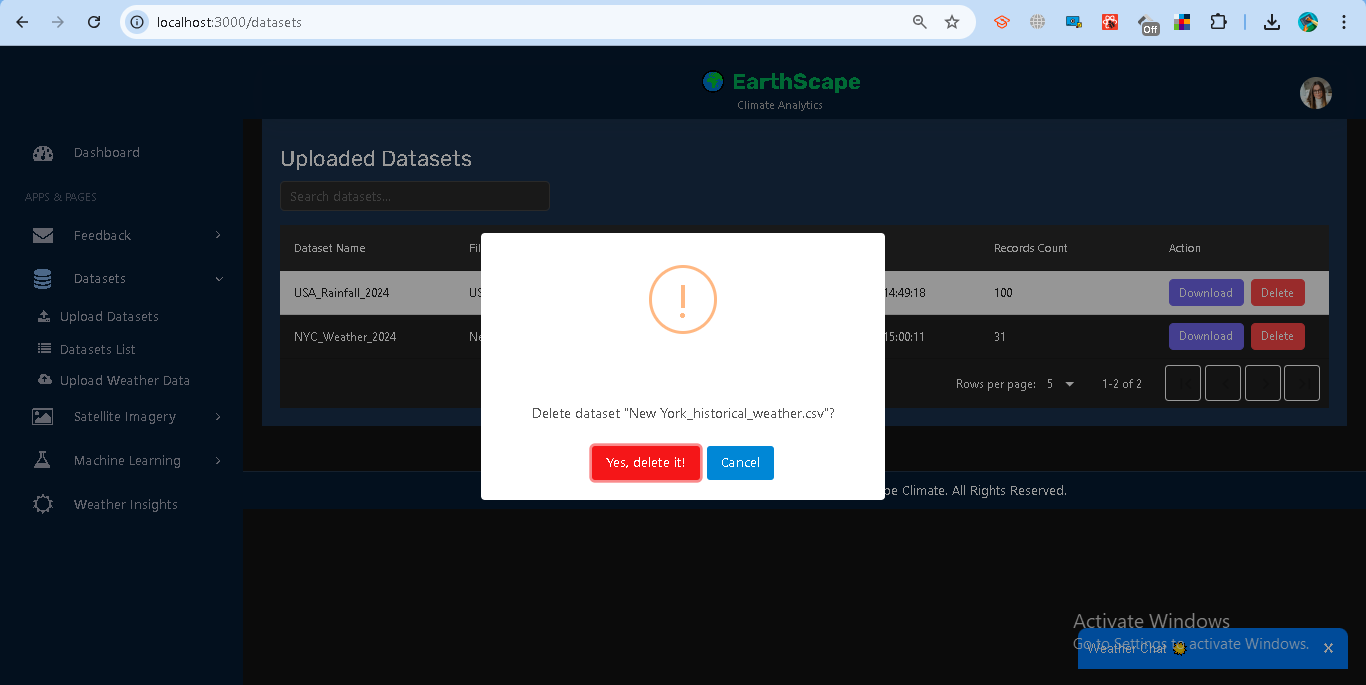
* **View Datasets:** All uploaded datasets can be viewed under **Dataset → *Datasets lists*.**

****

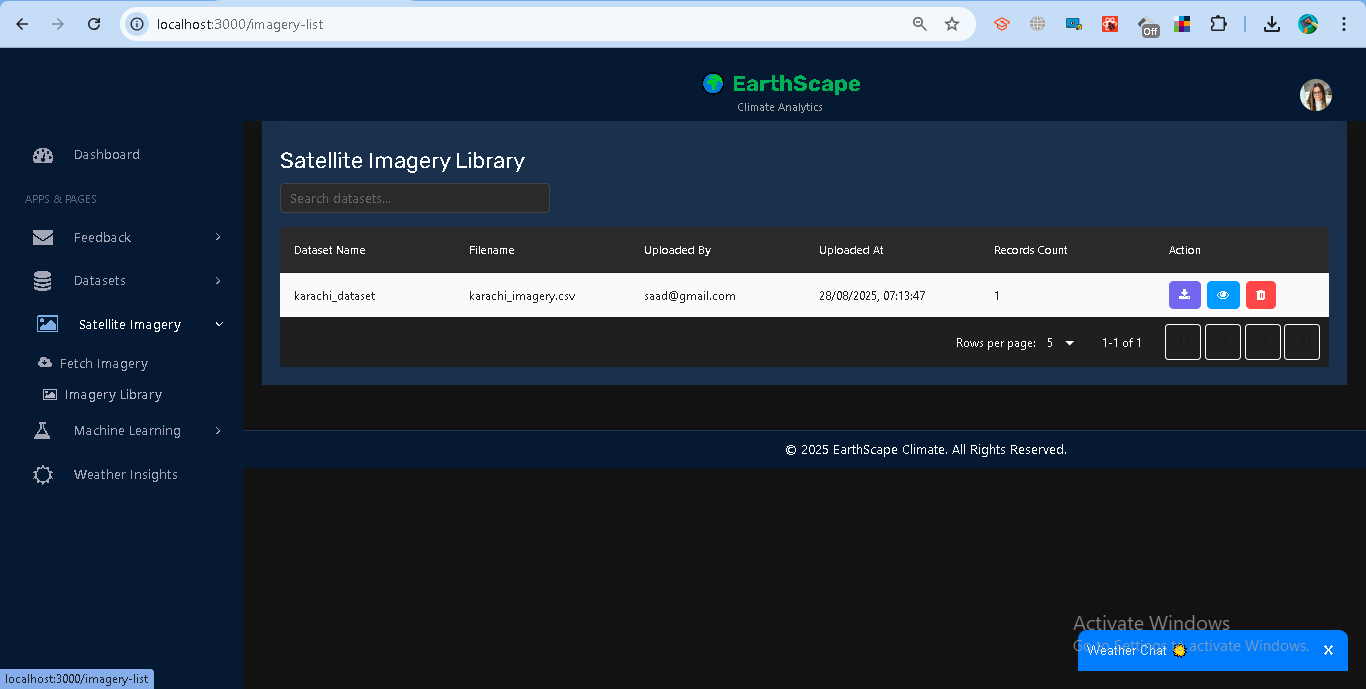
* **Download:** Each dataset provides a direct download link for .csv files.

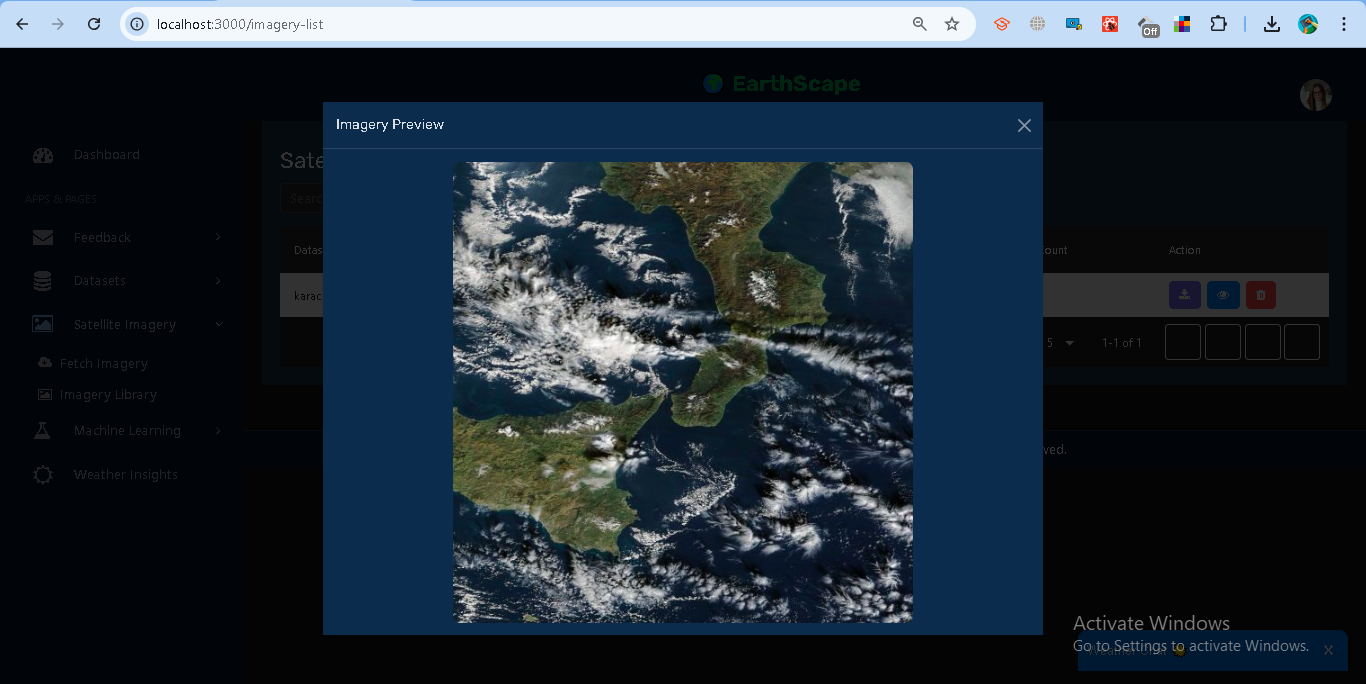


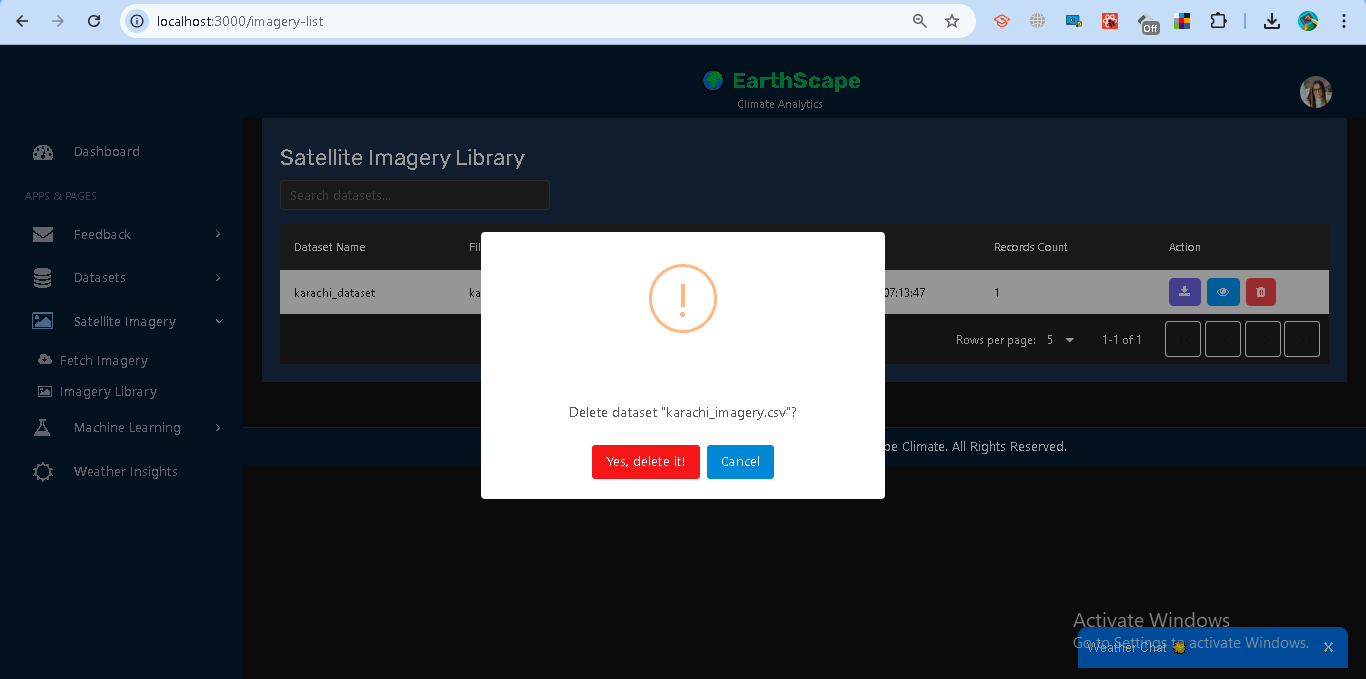
* **Delete:** Users may delete their own datasets; admins can delete all datasets.



* **View Datasets:** All uploaded datasets can be viewed under **Satellite Imagery → *Imagery Library*.**

****

****

****

## **1.3 Data Storage**

### **Implementation**

Instead of using Hadoop Distributed File System (HDFS), this project leverages **MongoDB Atlas** for storing climate-related datasets in a **scalable and fault-tolerant** way. MongoDB’s document-oriented design provides flexibility for handling diverse climate datasets such as CSV uploads, satellite imagery metadata, and weather station records.

* **Datasets Collection (datasets)**
  + Stores uploaded CSV datasets along with metadata (dataset name, file path, uploader, records, record counts).

Example structure:  
  
 {

"user\_id": "65ab3e...",

"datasets": [

{

"dataset\_name": "NYC\_Weather\_2024",

"filename": "nyc\_weather.csv",

"file\_path": "/dataset/nyc\_weather.csv",

"uploaded\_at": "2025-01-12T08:30:00Z",

"uploaded\_by": "admin@example.com",

"records\_count": 100,

"records": [ { "date": "2024-01-01", "temperature": 5, "humidity": 80 } ]

}

]

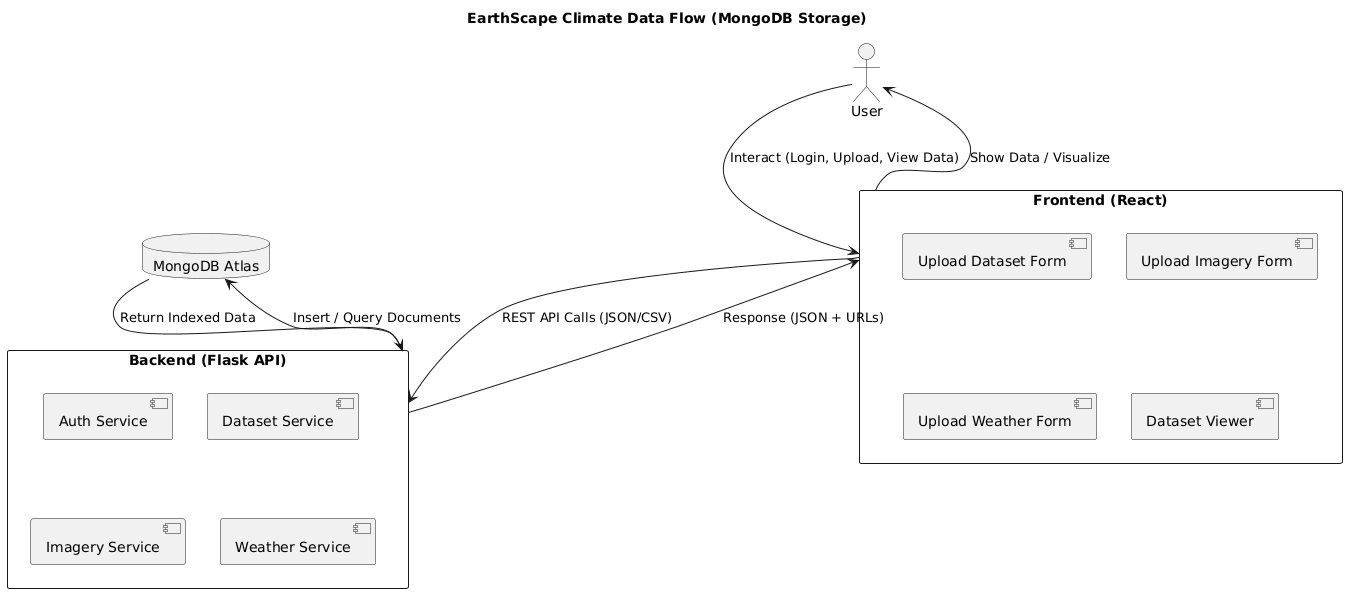
}

* **Satellite Imagery Collection (satellite\_imagery)**
  + Stores metadata about imagery requests (city, lat/lon, zoom level, NASA GIBS URL, generated CSV file, etc.).
* **User Profile Collection (users)**
  + Stores user credentials, roles, and cities, enabling role-based access control.

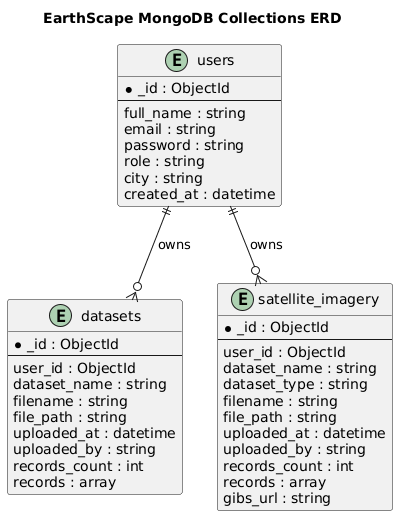
### **Optimization Strategies**

* **Indexing**
  + MongoDB indexes are created on fields like user\_id, datasets.filename, and satellite\_imagery.dataset\_name to speed up queries.
* **Partitioning / Sharding**
  + MongoDB supports **sharding** for horizontal scaling when datasets grow large. This ensures queries and storage are distributed across multiple nodes.
* **Document Nesting**
  + Datasets are stored as **nested arrays under each user document**, ensuring quick retrieval of all datasets for a given user.
* **File Storage**
  + Uploaded CSVs and imagery metadata are stored locally under /dataset and referenced in MongoDB with **download URLs** for retrieval.

**Data Flow**

****

**ERD**

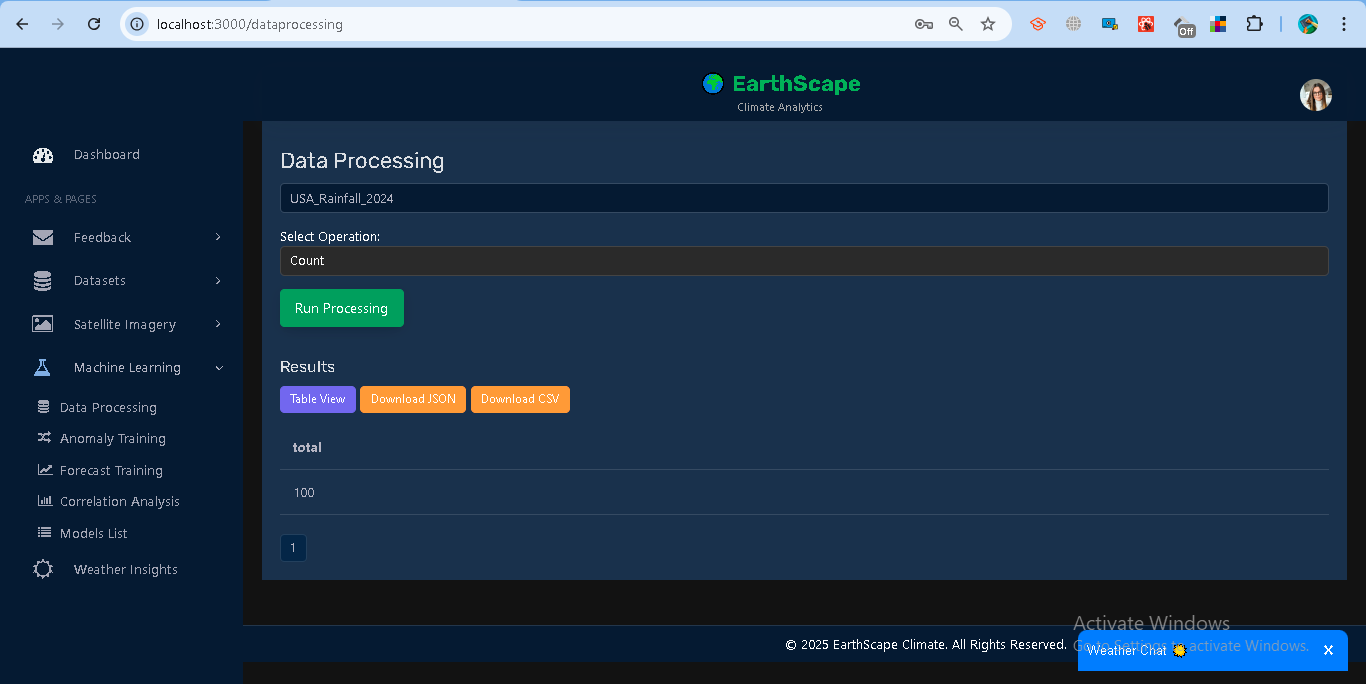
****

### **1.4 Data Processing**

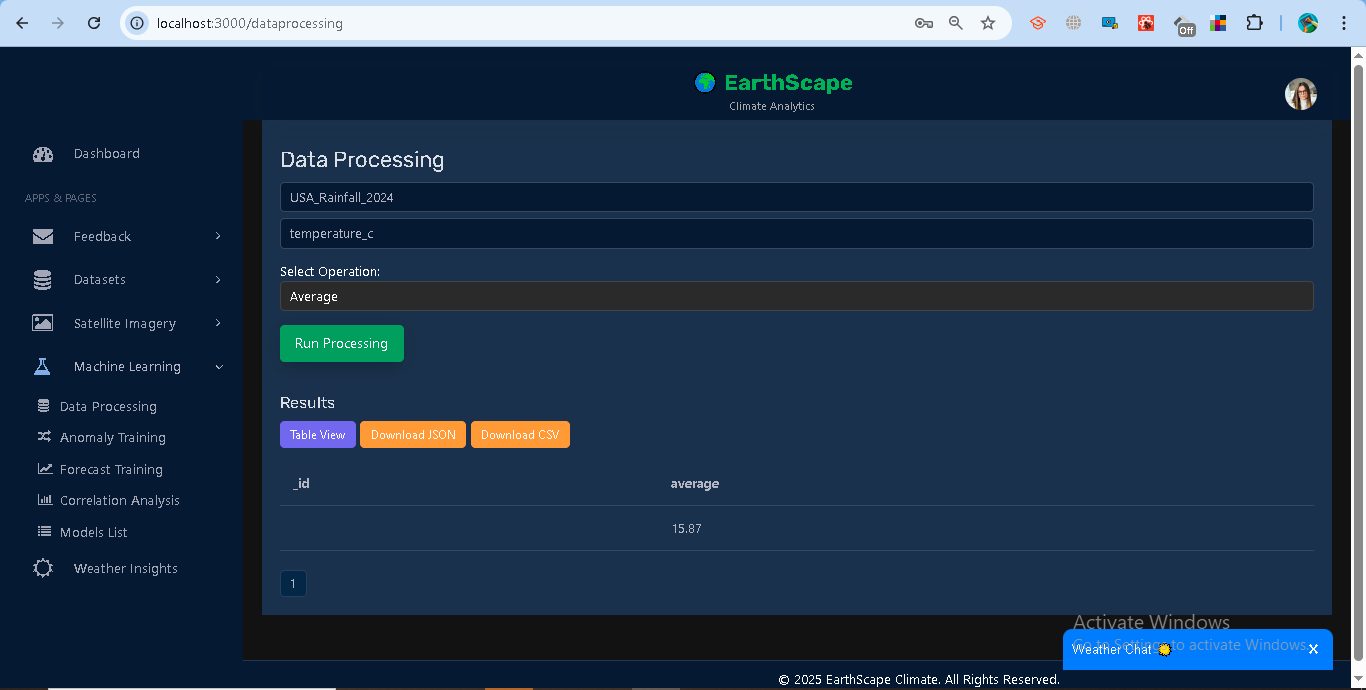
**Purpose** Allows users to analyze climate datasets (CSV, Weather, Satellite) stored in MongoDB. Operations include record counts, averages, groupings, and simple temporal pattern detection.

**Available Operations**

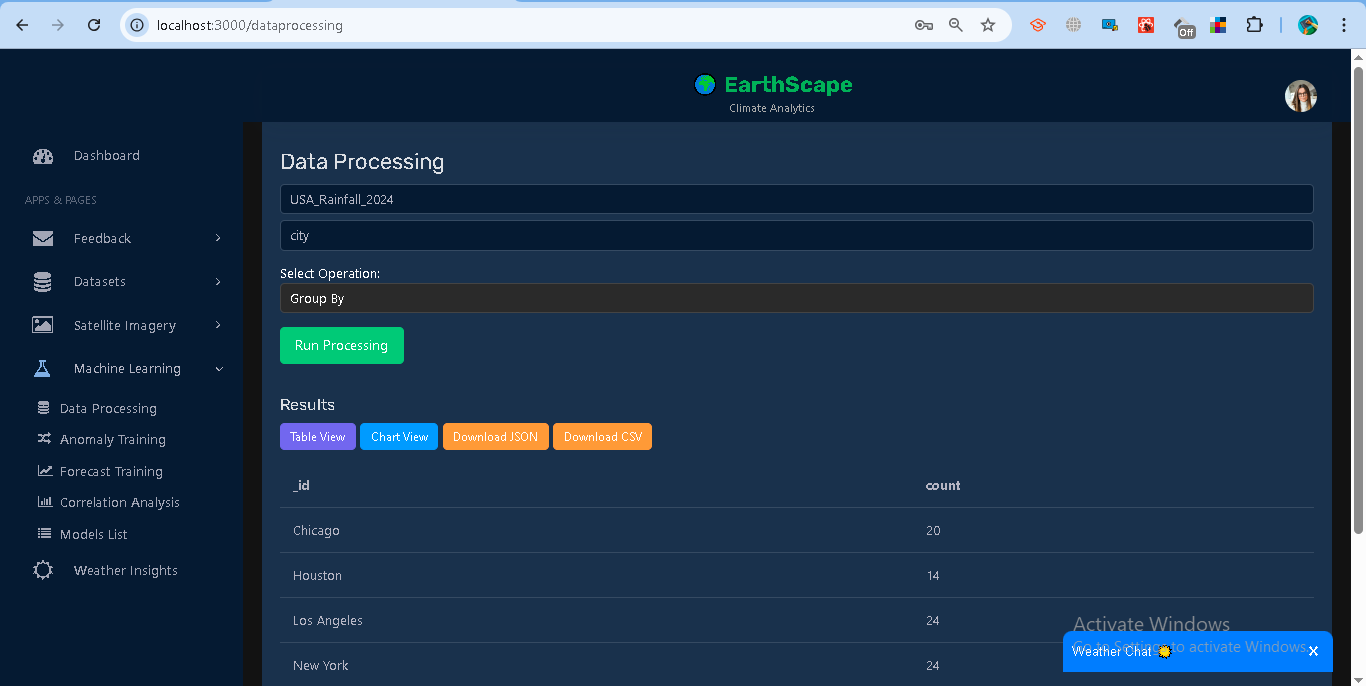
* **Count** → total records.



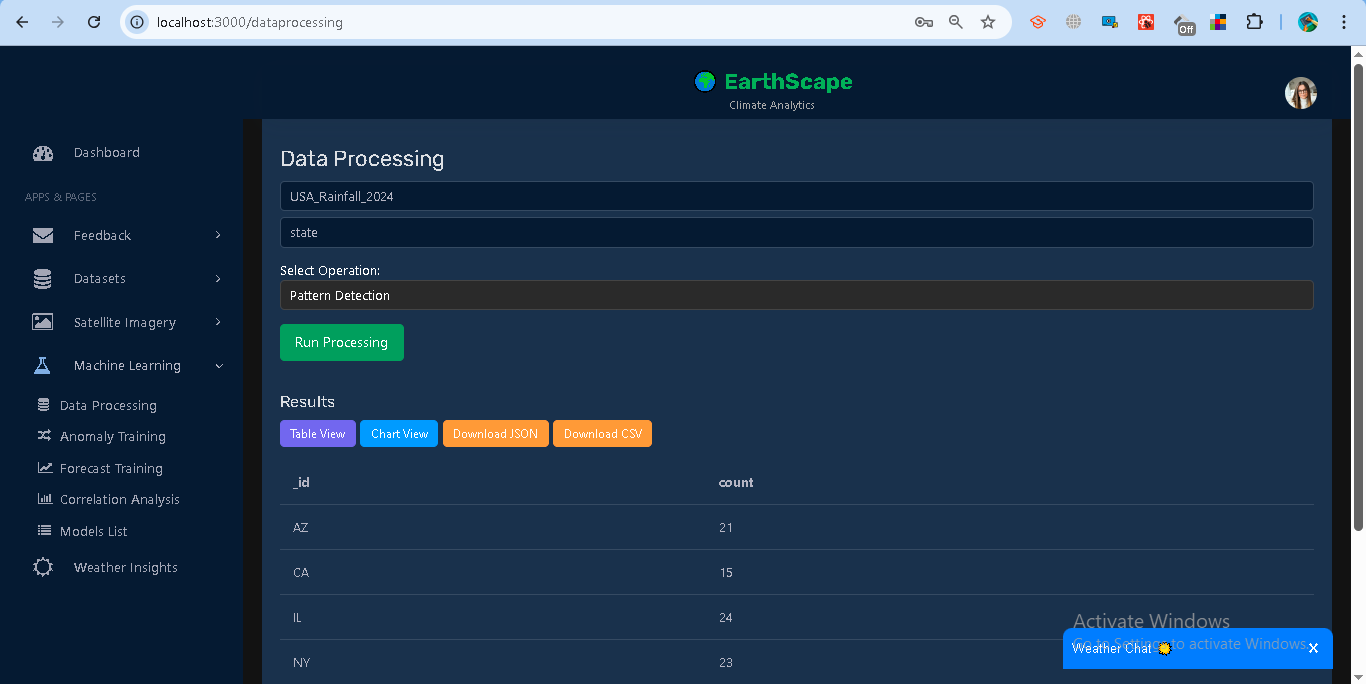
* **Average** → mean of numeric field.



* **Group By** → group by field (city, value bins).

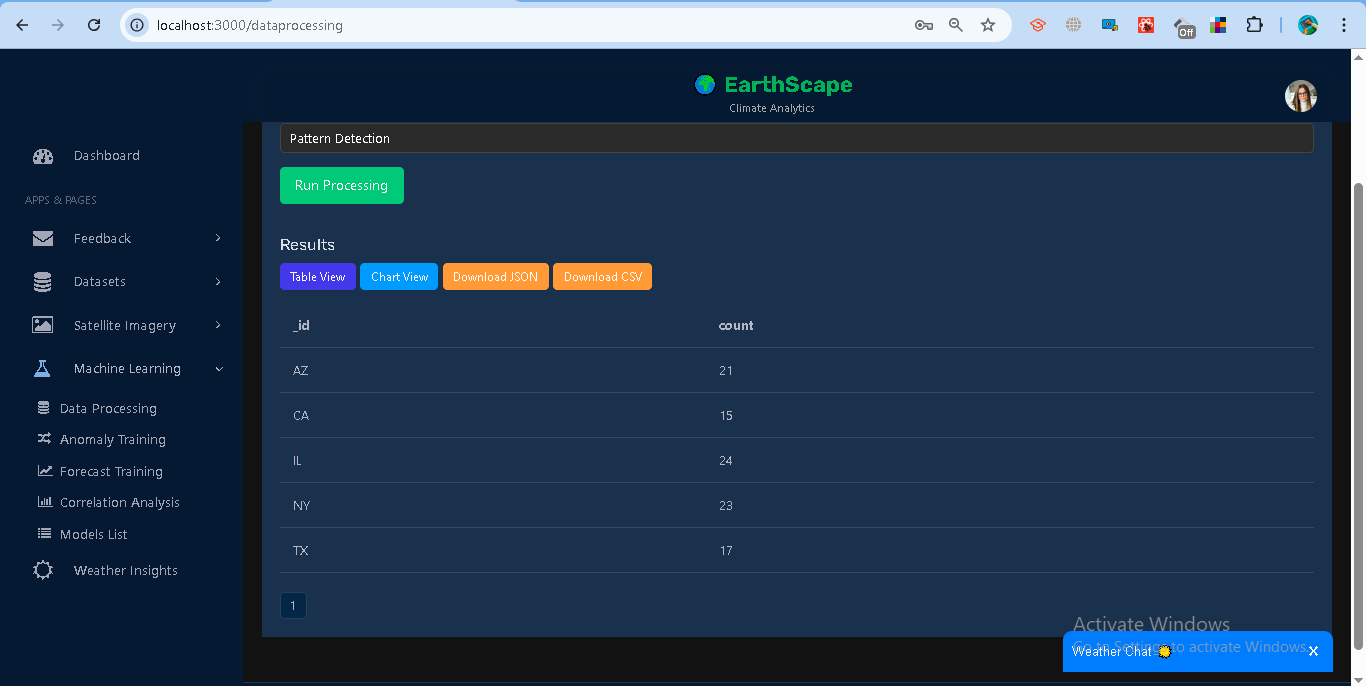


* **Pattern** → detect seasonality (group by month if date).

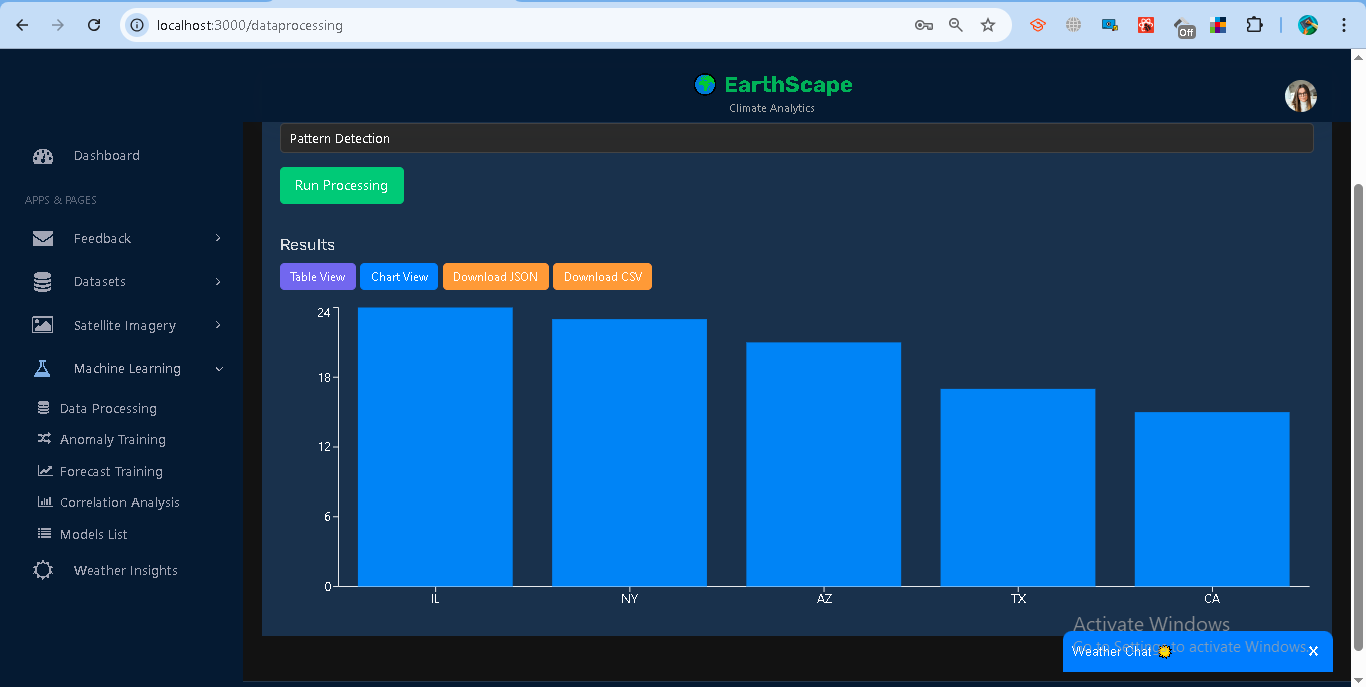


**How to Use (UI)**

1. Navigate to *Machine Learning → Data Processing*.
2. Select a dataset.
3. Choose an operation:  
   * For *average / group by / pattern*, select a field.
4. Click **Run Processing**.



1. View results as **Table** (with pagination) or **Chart** (Bar/Line).



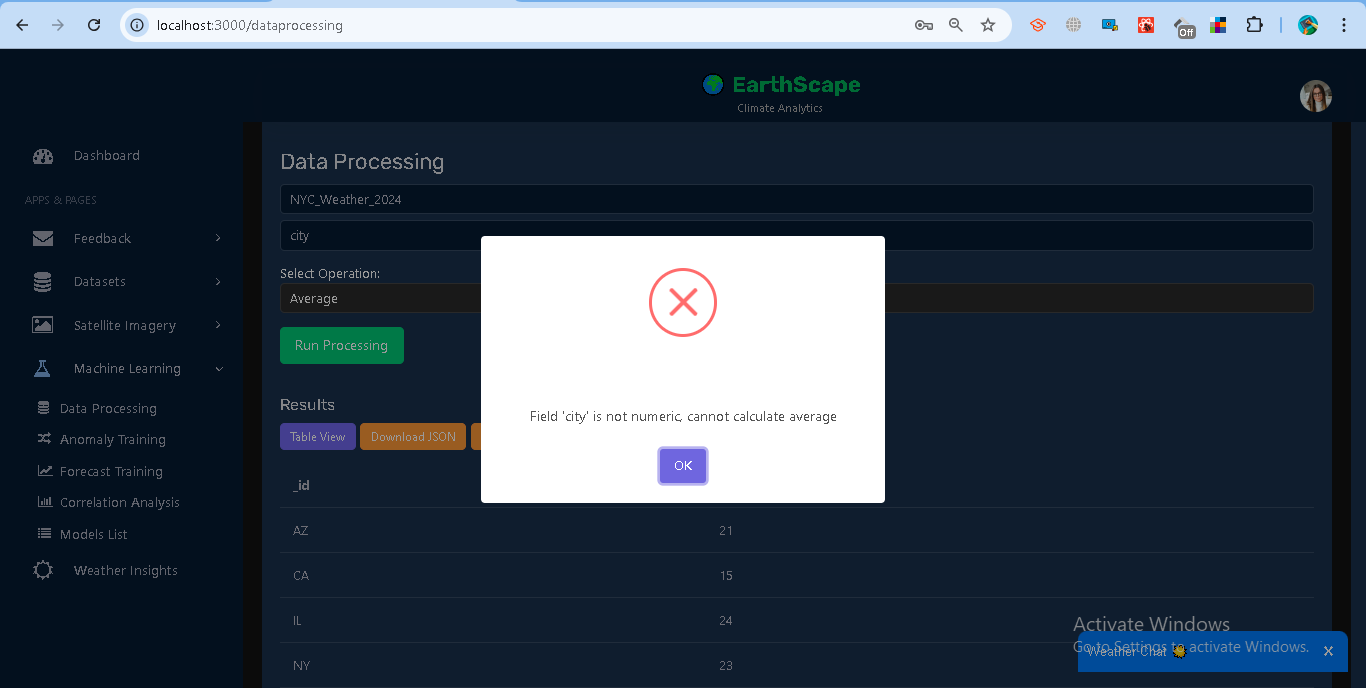
1. Download results as **JSON** or **CSV**.

**Examples**

* Count on NYC\_Weather\_2024 → { total: 1000 }
* Average on temperature → { average: 17.42 }
* Group By city → [{ \_id: "New York", count: 123 }, ...]
* Pattern on datetime → monthly counts { \_id: 1, count: 34 }

**Error Handling**

* Invalid field → "Field not numeric"
* Missing dataset → "Dataset not found"
* Expired/invalid token → "Unauthorized"



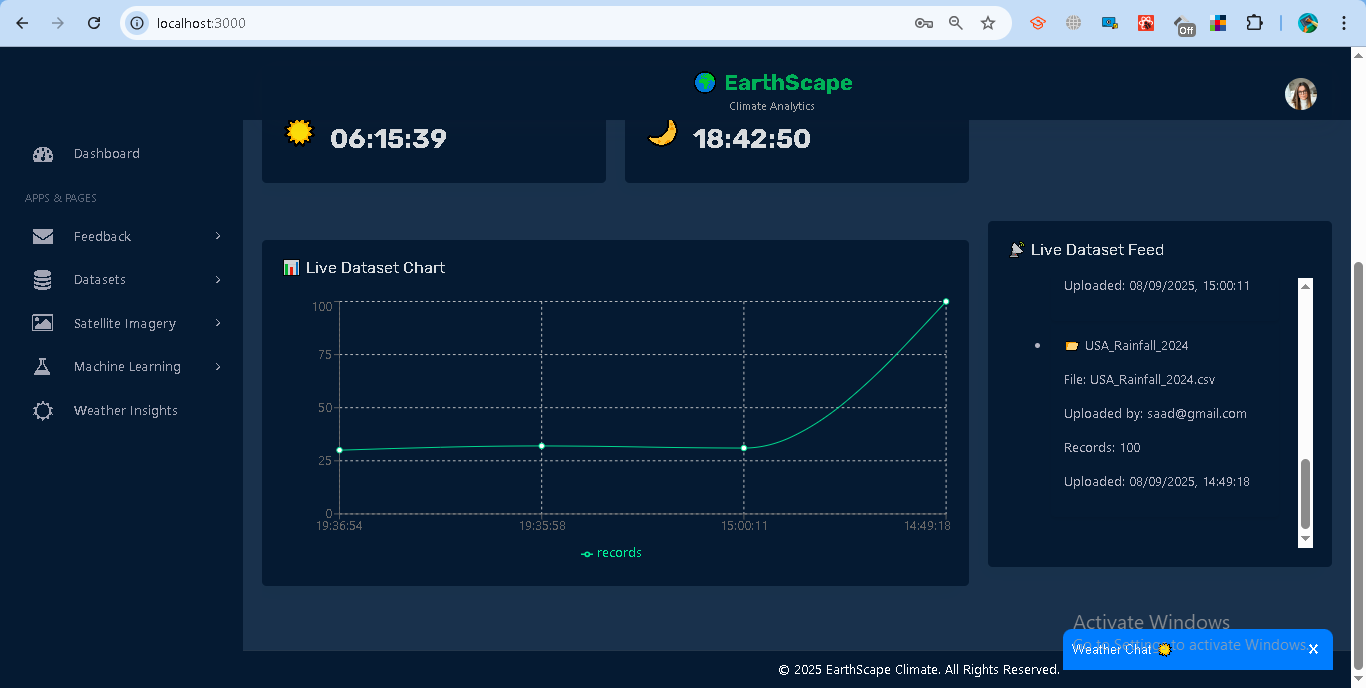
## **1.5 Real-time Data Processing**

The system integrates **real-time data streaming** with **batch data** for complete analysis. Users can view live updates on datasets and temperature records without refreshing the page.

### **Steps**

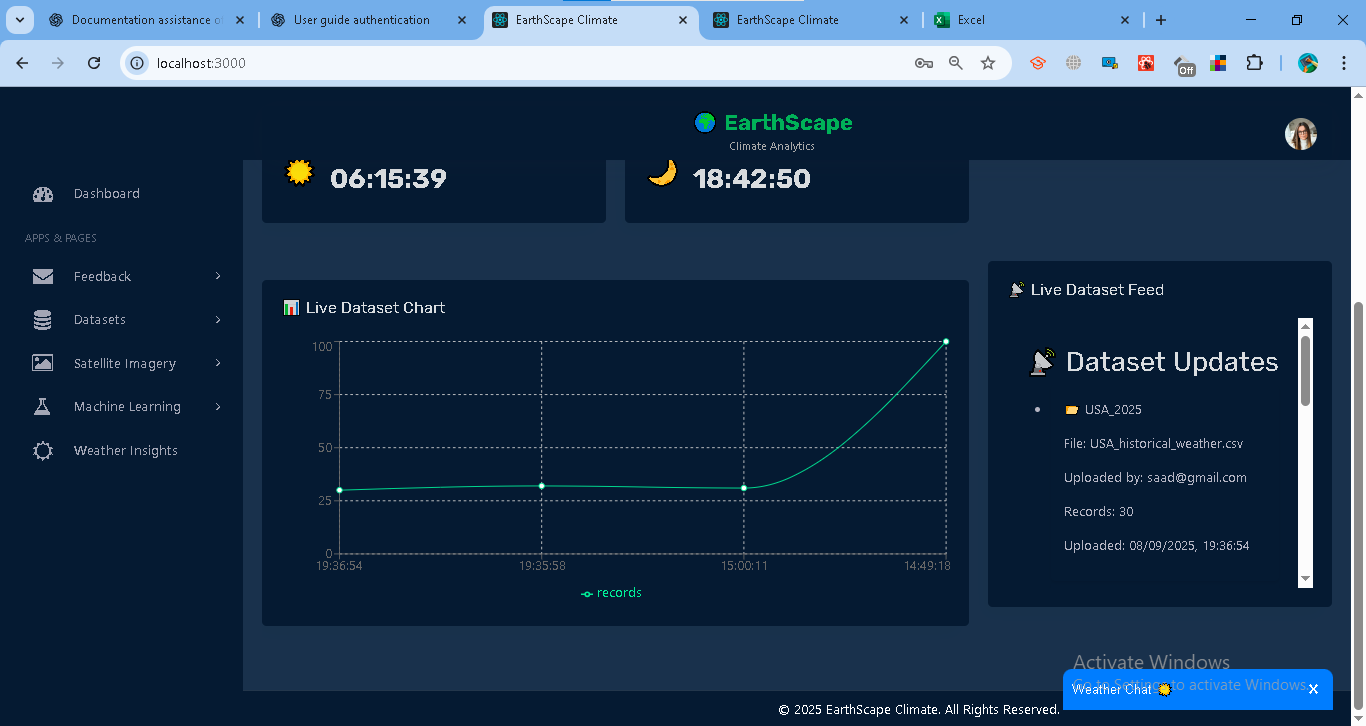
**Live Dataset Chart**

* Shows number of records over time.
* Loads historical data from /datasets/livefeed.
* Subscribes to /stream/datasets?token=... for new updates.



**Live Dataset Feed**

* Lists the latest 10 datasets.
* Displays file, uploader, record count, and timestamp.
* Option to expand and view sample records.



**Live Temperature (per City)**

* Plots temperature trends across multiple cities.
* Loads recent records from /datasets/latest?token=....
* Streams updates via /stream/datasets?token=....
* Cities can be selected from a dropdown.



### **Access Control**

* Analysts: see only their datasets.
* Admins: see datasets for the owner of the latest upload.

### **Error Handling**

* **401 Unauthorized** → Invalid or expired token.
* **404 No dataset found** → No datasets uploaded yet.
* **No updates** → Check SSE connection, token, and database stream settings.

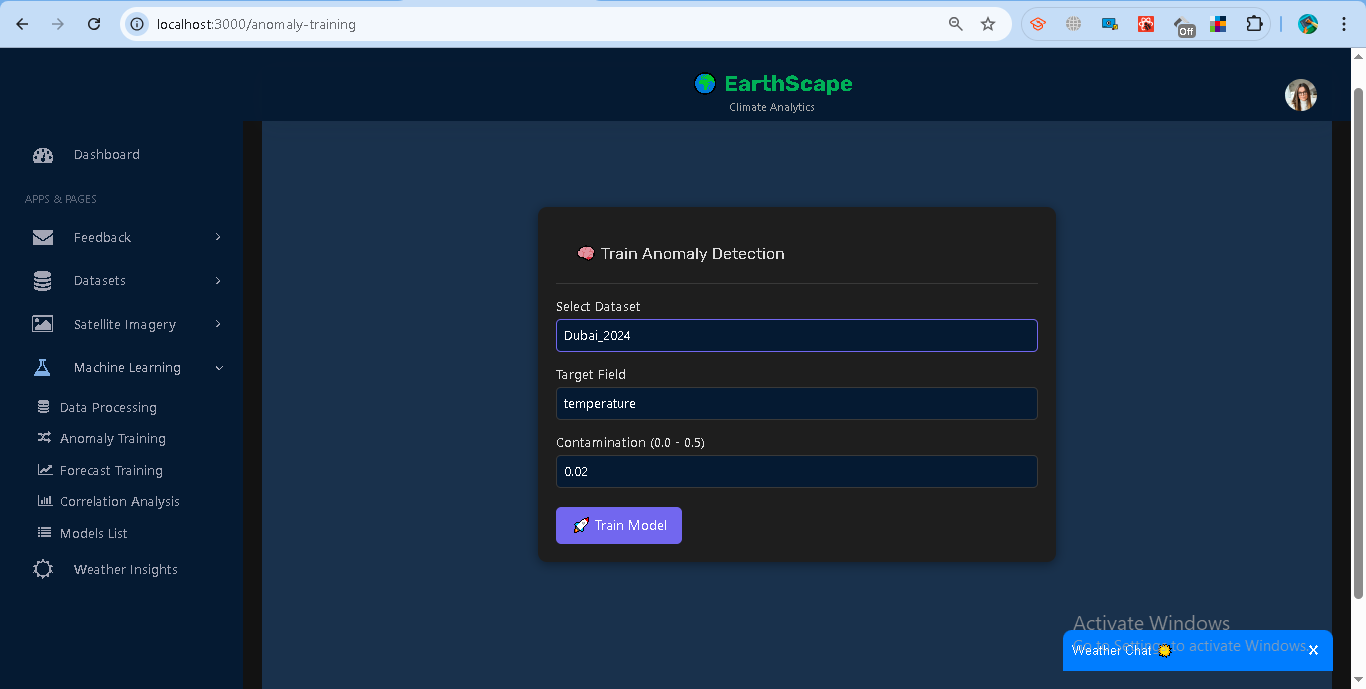
## **1.6 Anomaly Training**

The system provides anomaly detection using the **Isolation Forest model**. This helps identify unusual climate values (e.g., abnormal temperatures) in uploaded datasets.

### **Steps**

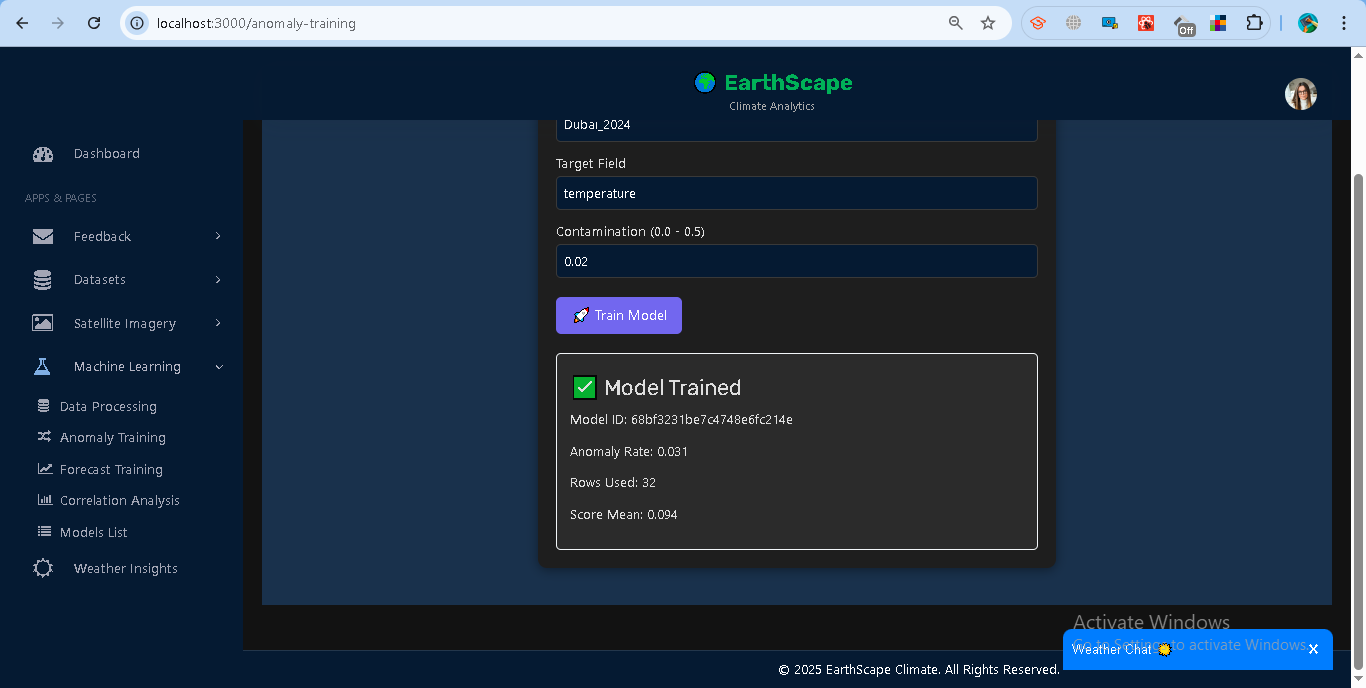
**Train a Model**

1. Go to **Machine Learning → Anomaly Training**.
2. Select a dataset (e.g., *Dubai\_2024*).
3. Enter the **Target Field** (numeric column, e.g., *temperature*).
4. Set **Contamination** (expected anomaly proportion, default = 0.02).
5. Click **Train Model**.



**Results**

* Model ID is generated and stored.
* Training summary shows:  
  + **Anomaly Rate** → percentage of detected anomalies.
  + **Rows Used** → records used in training.
  + **Score Mean** → average decision score.

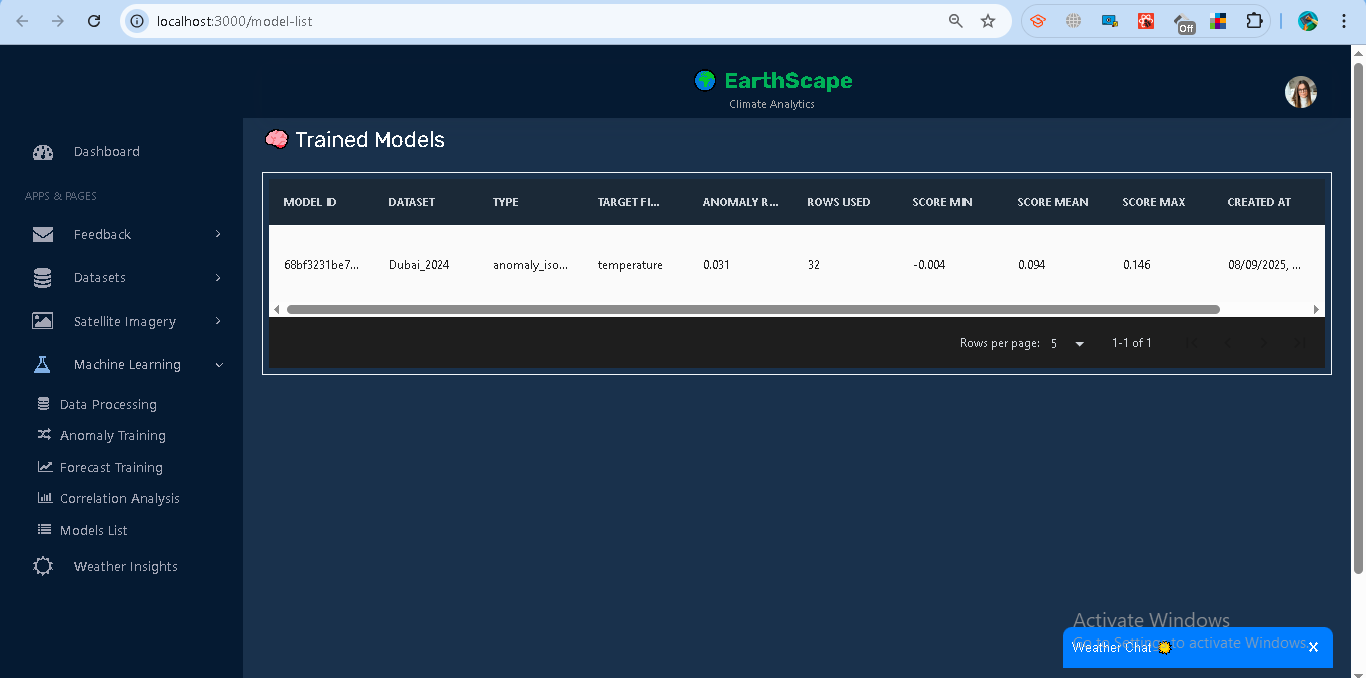


**Access Control**

* Analysts: Can only train and use models on their datasets.
* Admins: Can view and manage all users’ models.

**Error Handling**

* **400** → Missing dataset/field or insufficient records (<20).
* **401** → Invalid/expired token.
* **403** → Accessing another user’s model without admin rights.
* **500** → Training failure (invalid data format, missing numeric values).



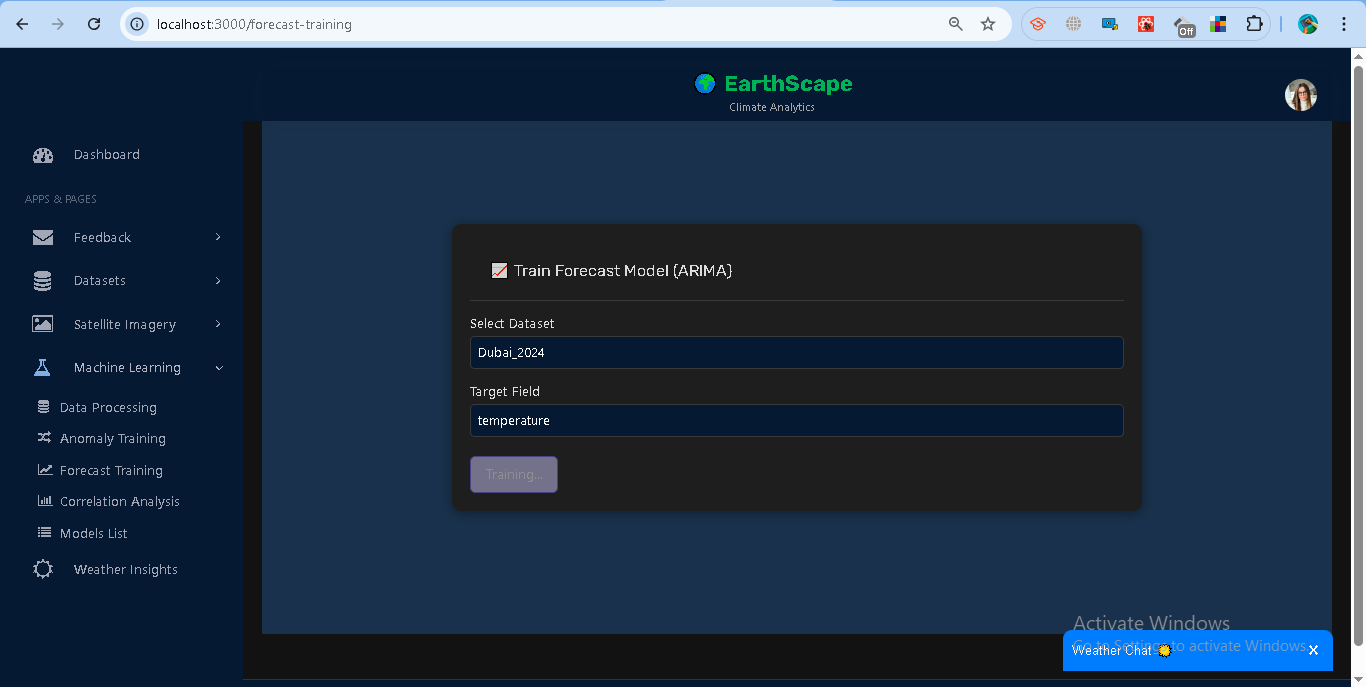
## **1.7 Forecast Training**

The system supports **time-series forecasting** using the **ARIMA model**. This allows predicting future values (e.g., temperature trends) from uploaded datasets.

### **Steps**

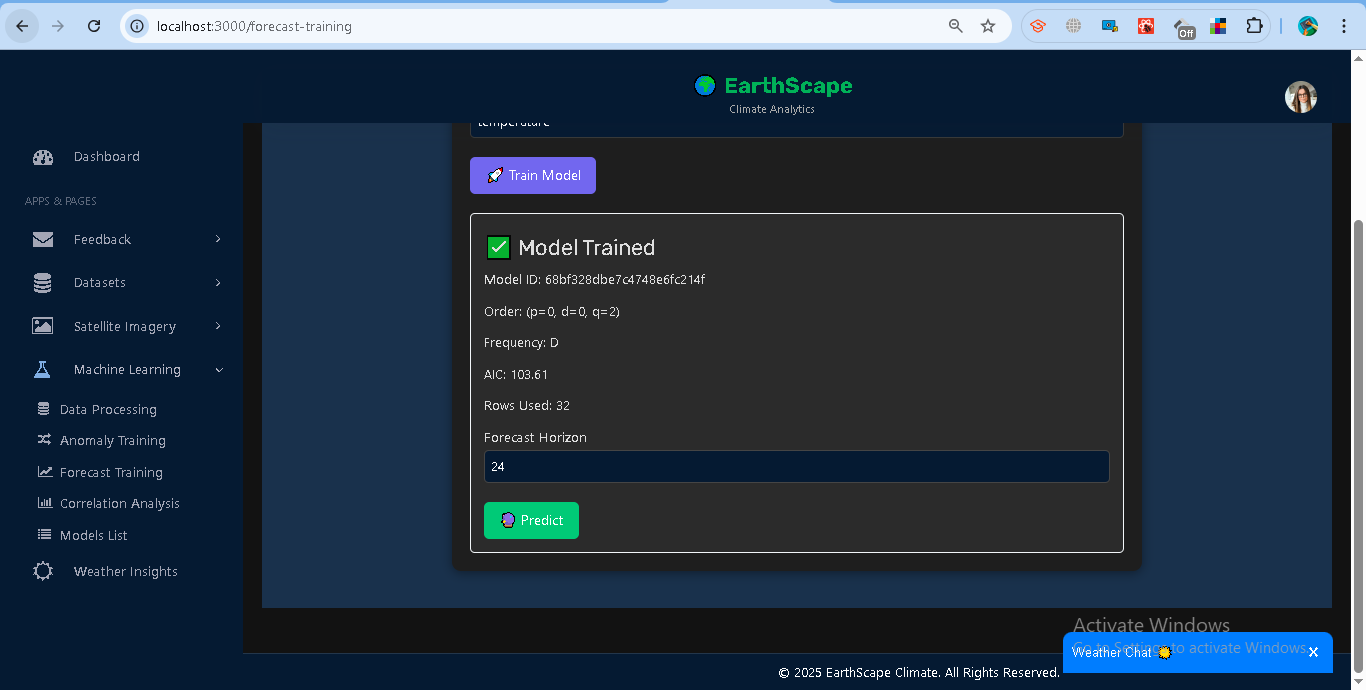
**Train a Forecast Model**

1. Go to **Machine Learning → Forecast Training**.
2. Select a dataset (e.g., *Dubai\_2024*).
3. Enter the **Target Field** (numeric column, e.g., *temperature*).
4. Click **Train Model**.



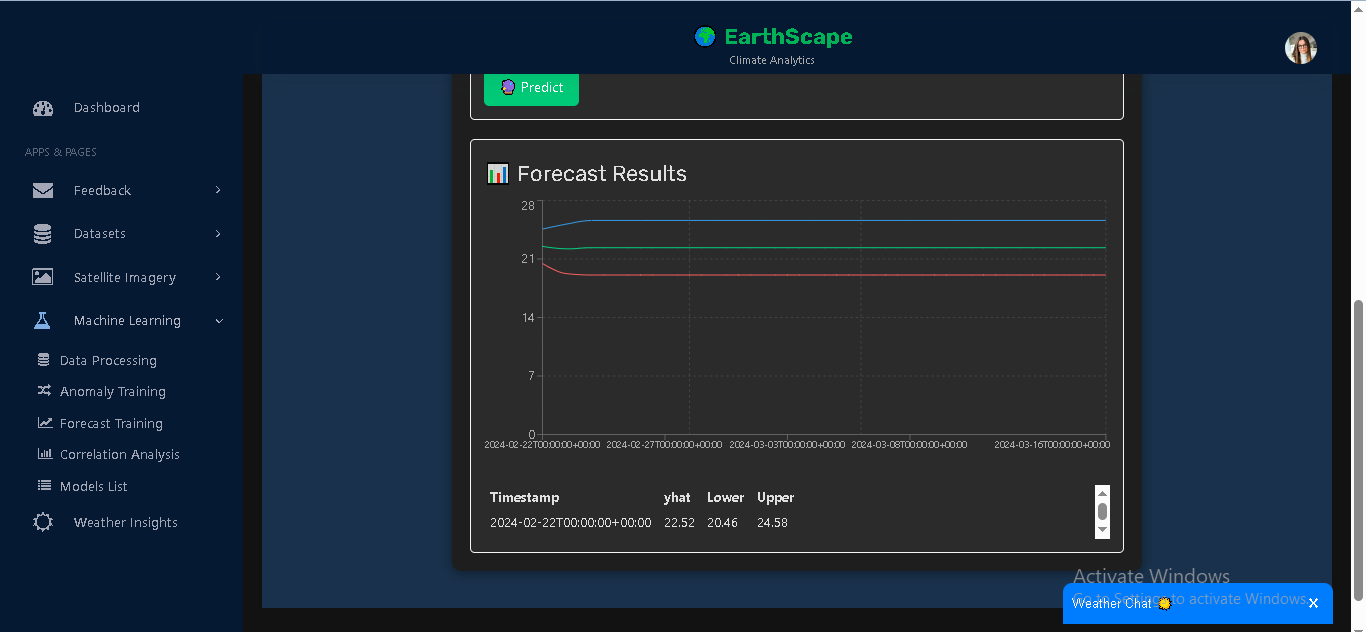
**Training Output**

* Model ID (saved in system).
* ARIMA Order (p,d,q).
* Frequency (detected time interval).
* Metrics: AIC score and training rows used.



**Generate Forecasts**

1. Enter a **Forecast Horizon** (number of future steps, e.g., 24).
2. Click **Predict**.
3. View results in:  
   * **Chart**: Predicted line with confidence bounds.
   * **Table**: Timestamp, prediction (yhat), lower and upper bounds.

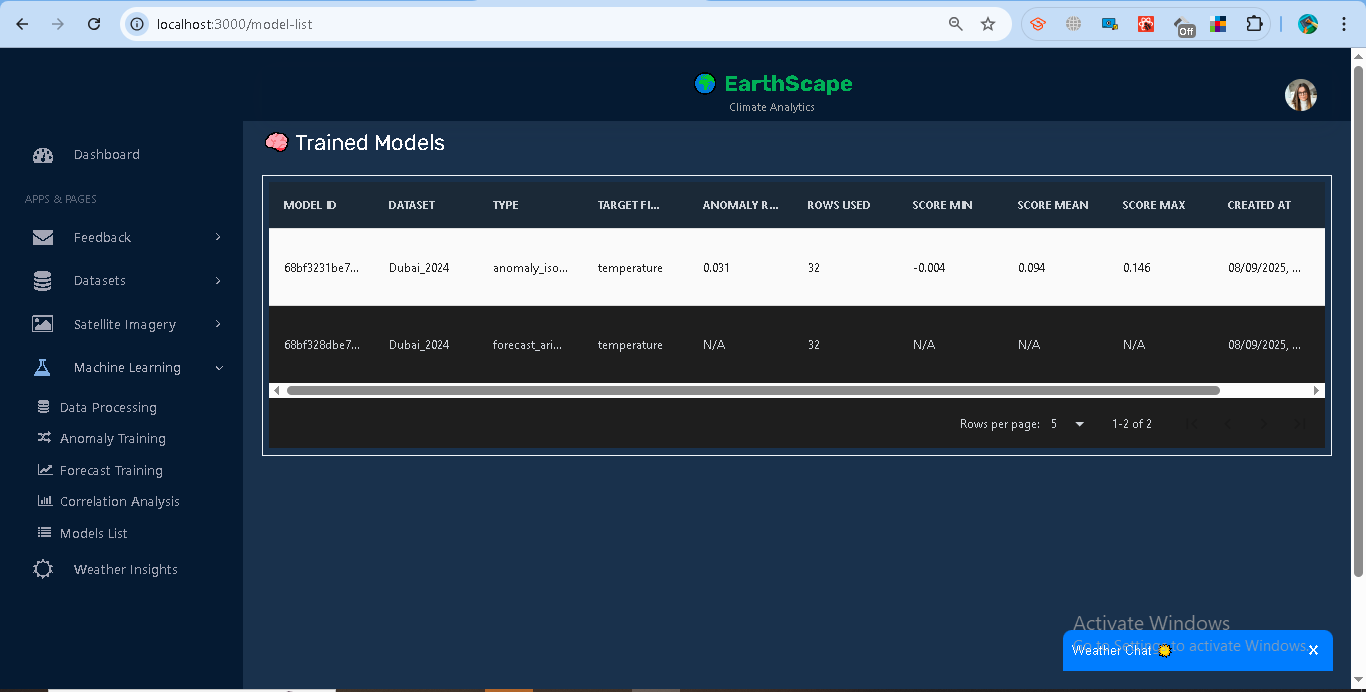


### **Access Control**

* Analysts: Train/predict only on their datasets.
* Admins: Can access and manage models across users.

### **Error Handling**

* **400** → Missing dataset/field or insufficient records (<20).
* **401** → Invalid/expired token.
* **403** → Not authorized to use another user’s model.
* **500** → Training/prediction error (data quality, ARIMA fit failure).



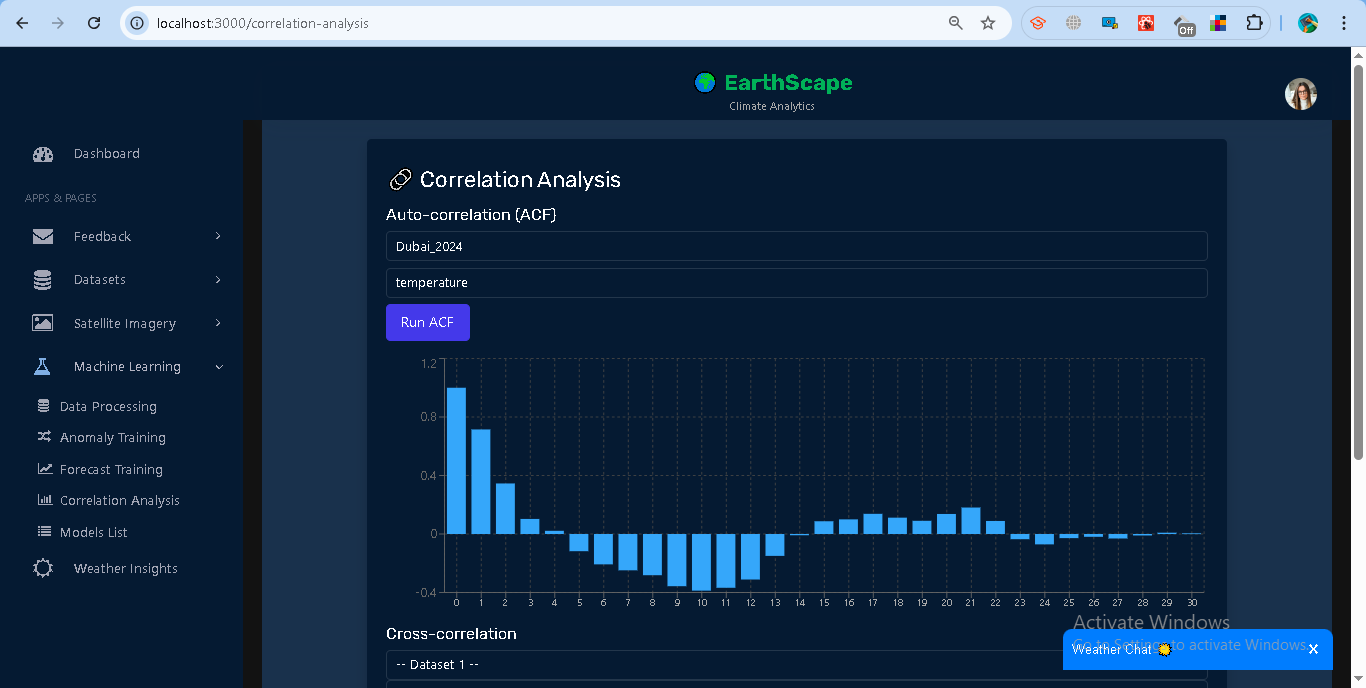
## **1.8 Correlation Analysis**

The system provides correlation tools to help understand relationships within and between climate datasets. Analysts can compute **Auto-correlation (ACF)**, **Cross-correlation**, and **Correlation Matrices**.

### **Features**

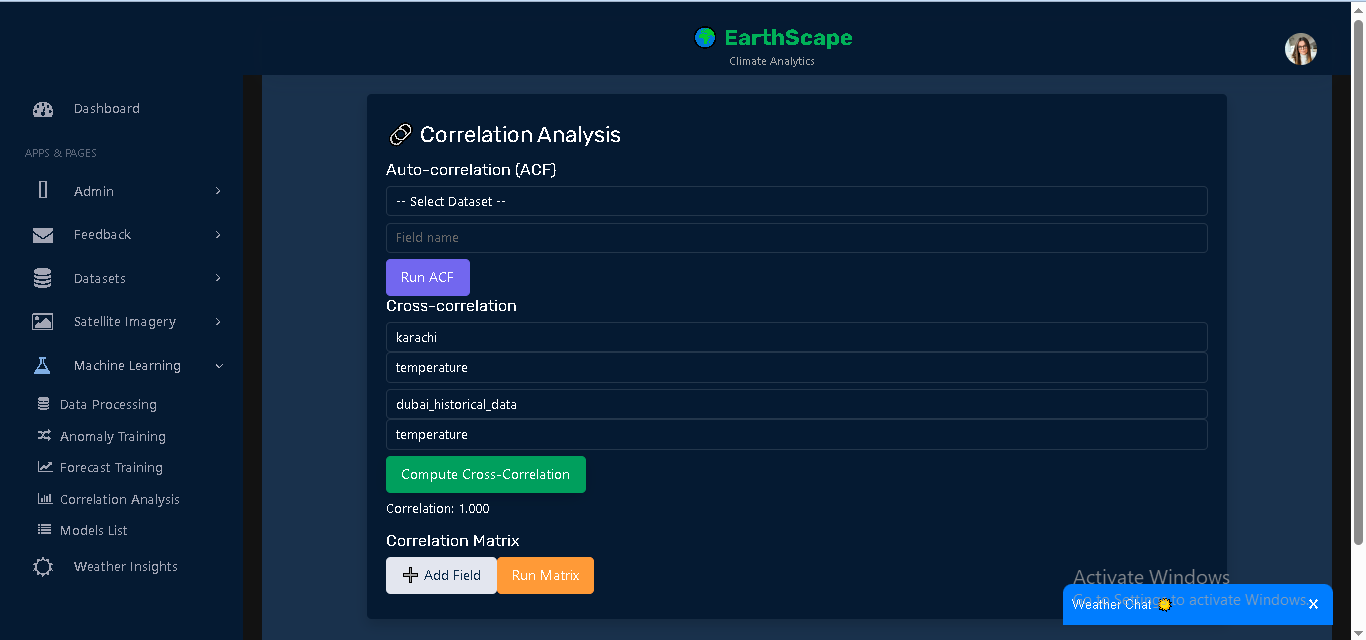
**Auto-correlation (ACF)**

* Measures how a field correlates with its past values (lags).
* Steps:  
  1. Select a dataset.
  2. Enter a **Target Field** (numeric column, e.g., *temperature*).
  3. Run ACF.
* Results: Bar chart of lag vs correlation.



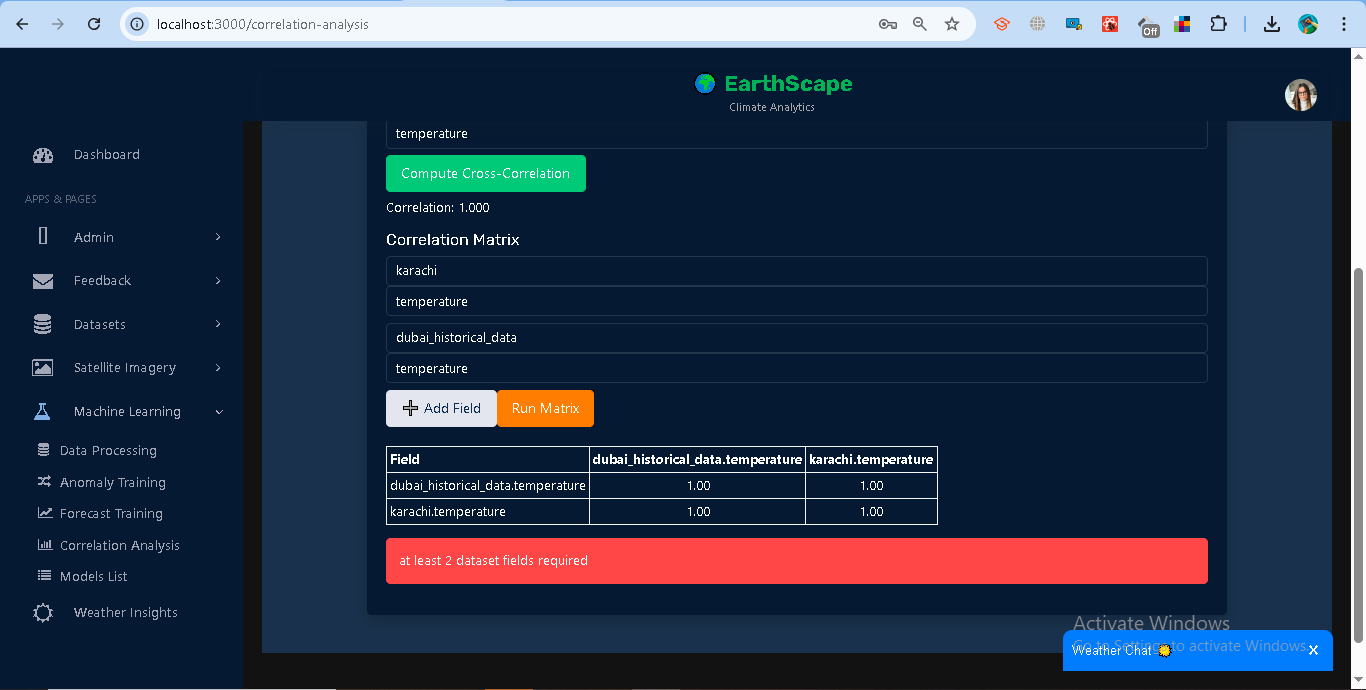
**Cross-correlation**

* Compares similarity between two dataset fields.
* Steps:  
  1. Select **Dataset 1 + Field**.
  2. Select **Dataset 2 + Field**.
  3. Compute correlation.
* Results: Single correlation score (between -1 and 1).



**Correlation Matrix**

* Shows correlation between multiple dataset fields at once.
* Steps:  
  1. Add datasets + fields (minimum 2).
  2. Run Matrix.
* Results: Table of pairwise correlation values.



### **Access Control**

* Analysts: Can analyze only their datasets.
* Admins: Can run correlations across all datasets.

### **Error Handling**

* **400** → Missing dataset/field inputs.
* **401** → Invalid/expired token.
* **403** → Accessing another user’s data without admin rights.
* **500** → Calculation error (e.g., missing numeric data).

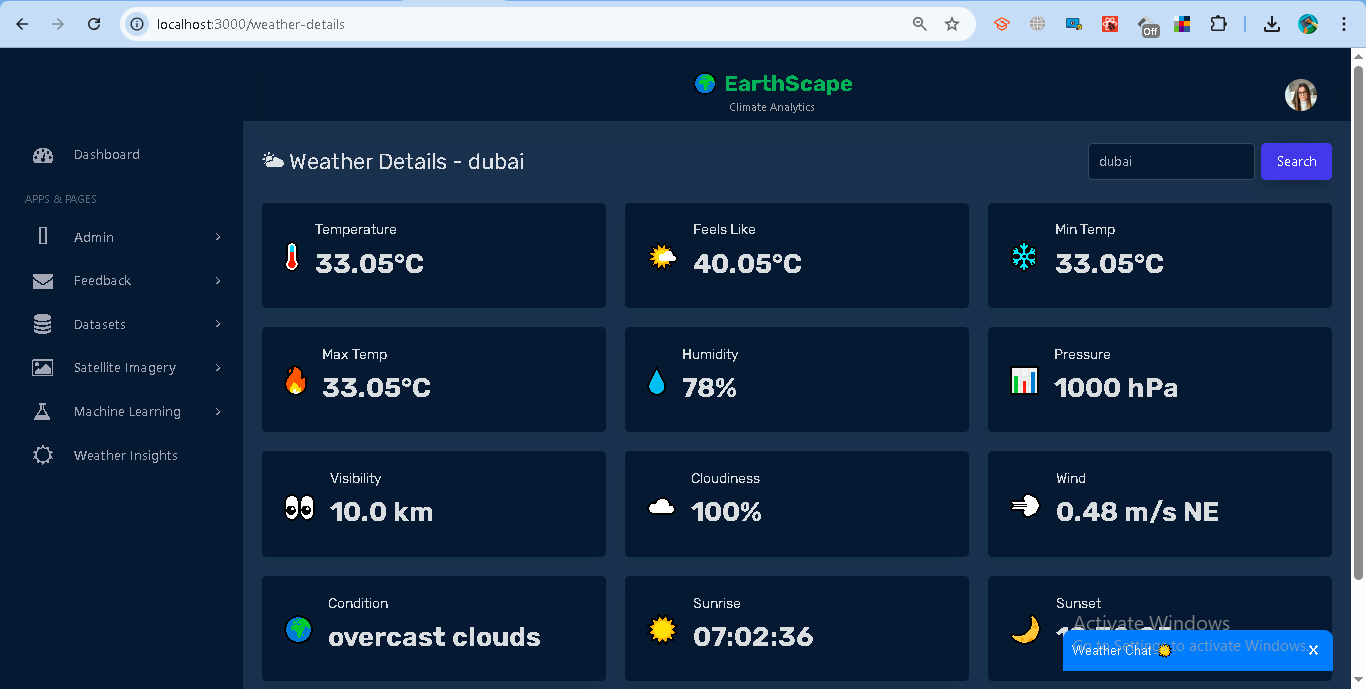
## **1.10 Weather Details**

The Weather Details module provides **real-time weather insights** using the OpenWeatherMap API. Users can search for a city and view key weather indicators in a dashboard-style card layout.

### **Features**

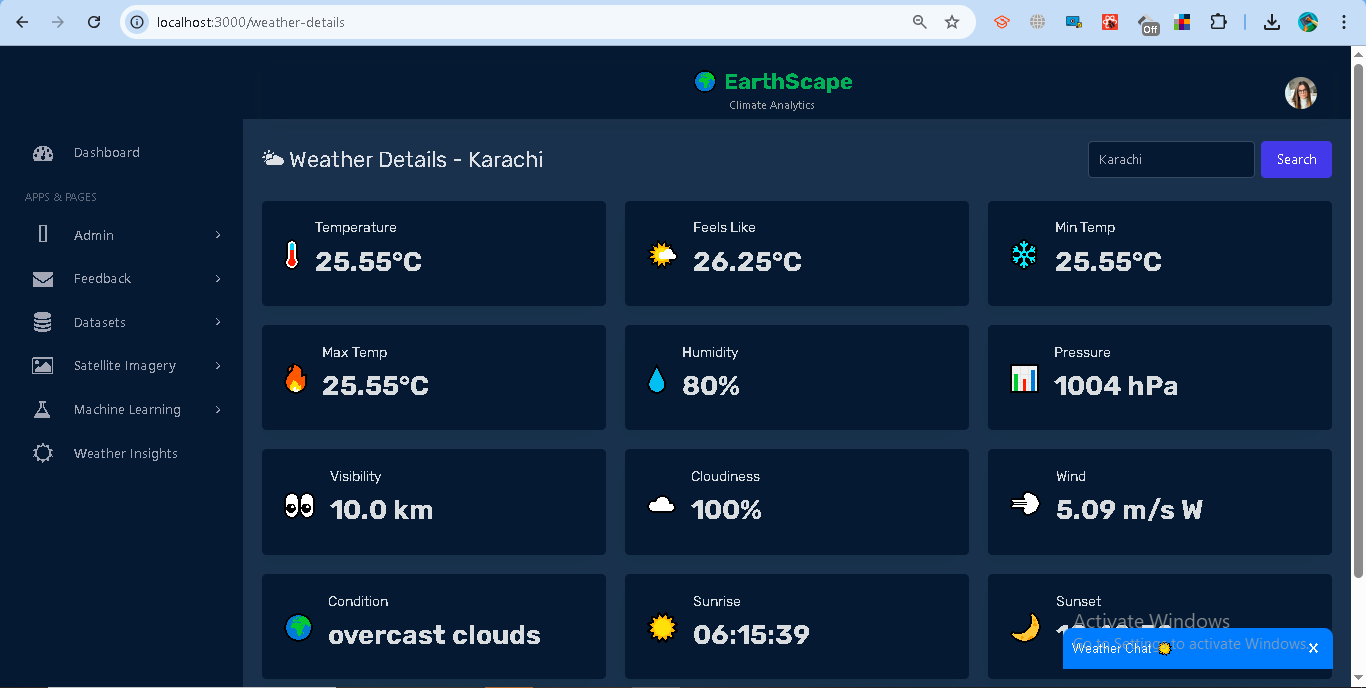
**Search and Fetch Weather**

1. Go to **Weather Insights** in the sidebar.
2. Enter a **city name** (e.g., *Dubai*) and click **Search**.
3. The system fetches live data from OpenWeatherMap.
4. The query is stored in MongoDB with a timestamp.



**Displayed Metrics**

* **Core:** Temperature, Feels Like, Min Temp, Max Temp.
* **Atmosphere:** Humidity, Pressure, Visibility, Cloudiness.
* **Wind & Condition:** Wind speed + direction, Weather condition.
* **Sun Info:** Sunrise and Sunset times (local).
* **Rain/Snow:** If available, rain/snow amount is displayed.



* **Errors:**
  + **400** → No city provided.
  + **404** → City not found.
  + **500** → External API/server error.

### **Access Control**

* Requires authentication (Bearer <token>).
* Searches are stored per user in MongoDB.

### **Error Handling**

* Toast notifications are shown for invalid city names or server errors.
* Weather card shows "Loading weather..." until data is available.

## **1.11 Weather ChatBot**

The Weather ChatBot provides an interactive way for users to query weather conditions for any city or country. It acts as a conversational assistant, fetching live data from the backend weather API.

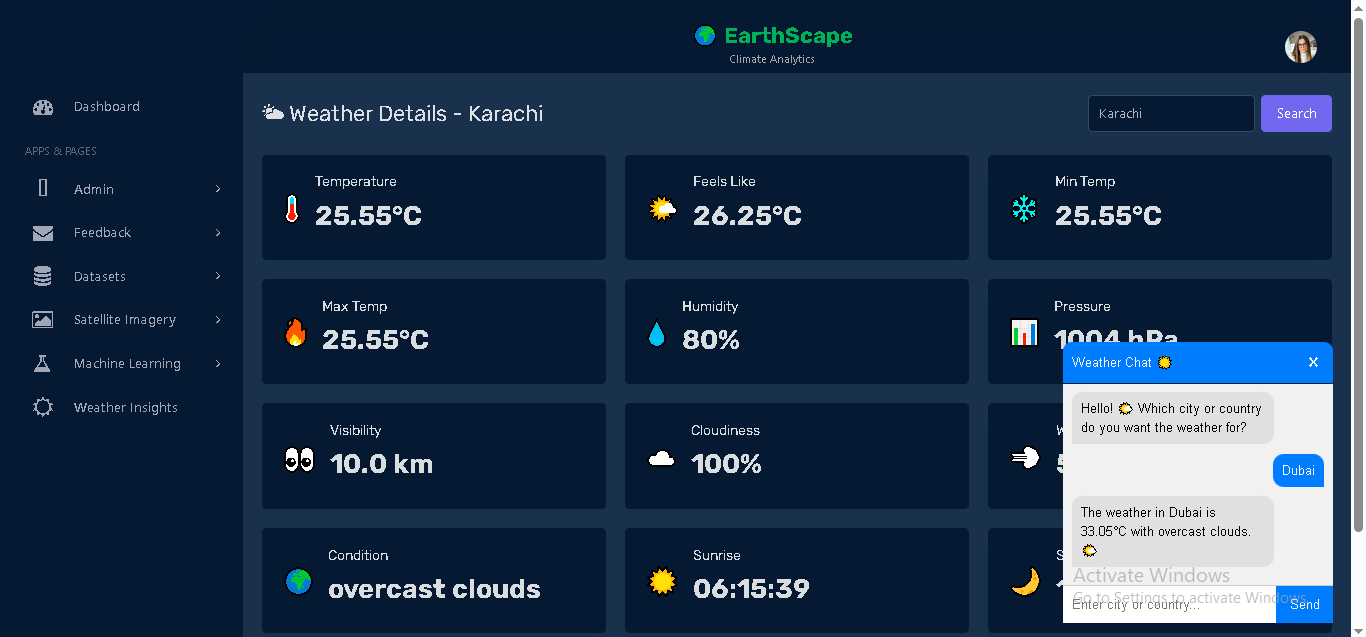
### **Features**

**Interactive Messaging**

* The bot greets the user and prompts for a city or country.
* Users type queries into the chat input.
* The bot responds with weather conditions such as **temperature** and **description**.

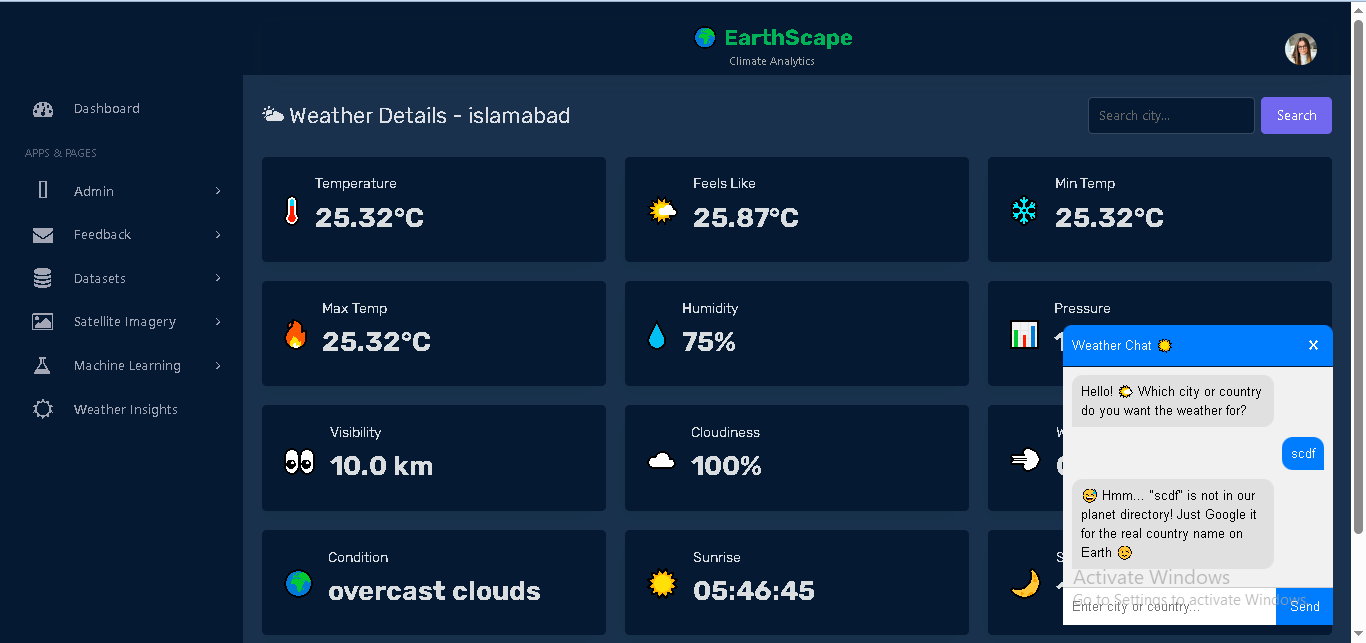
**Weather Query**

* Input: City or country name (e.g., *Dubai*, *USA*).
* Output: A conversational response, e.g.:  
   *“The weather in Dubai is 33°C with clear sky 🌤️”*

**

**Error Handling**

* If an invalid city/country is entered, the bot replies with a playful error message:  
   *“😅 Hmm... ‘X’ is not in our planet directory! Just Google it for the real country name on Earth 😉”*

**

**UI Behavior**

* The ChatBot is docked as a widget on the page.
* Clicking the header toggles the chatbot open/close.
* Messages auto-scroll to the bottom.
* Previous conversation is preserved while the session is active.

### **Access Control**

* Public feature, no login required.
* Requests go through the weather API with API key protection.

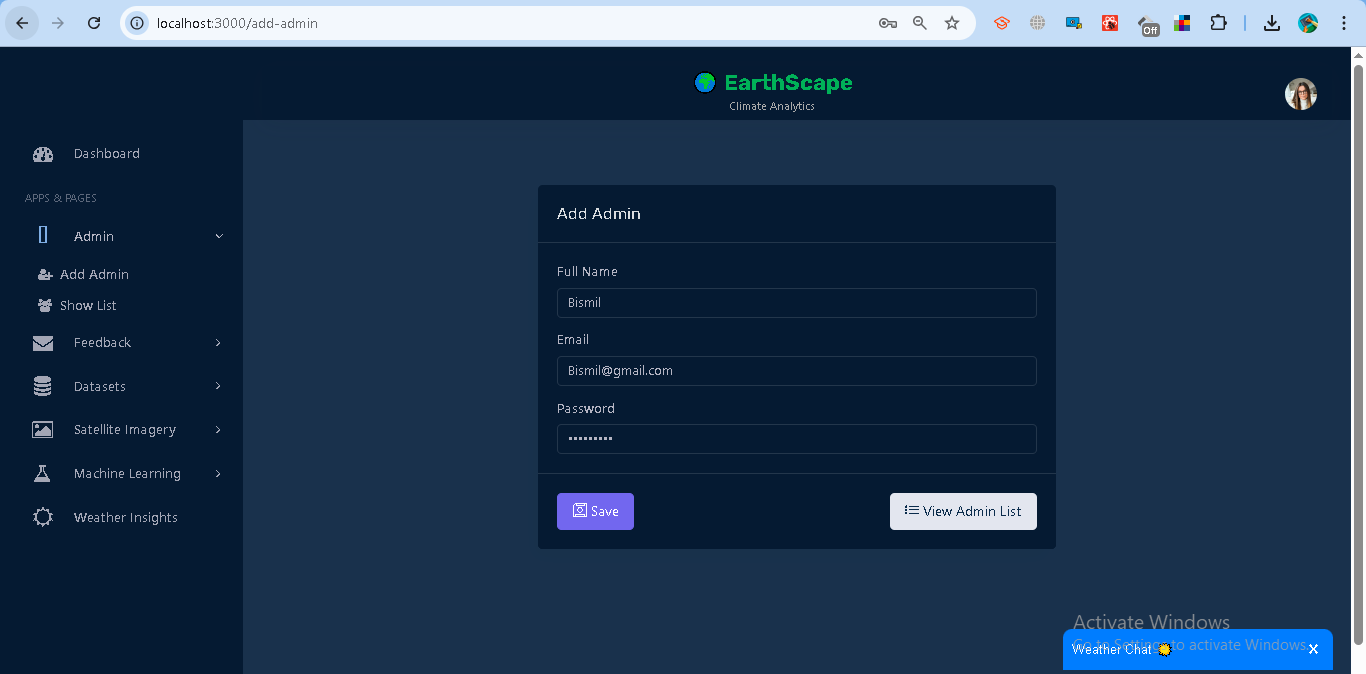
## **1.12 Admin Management**

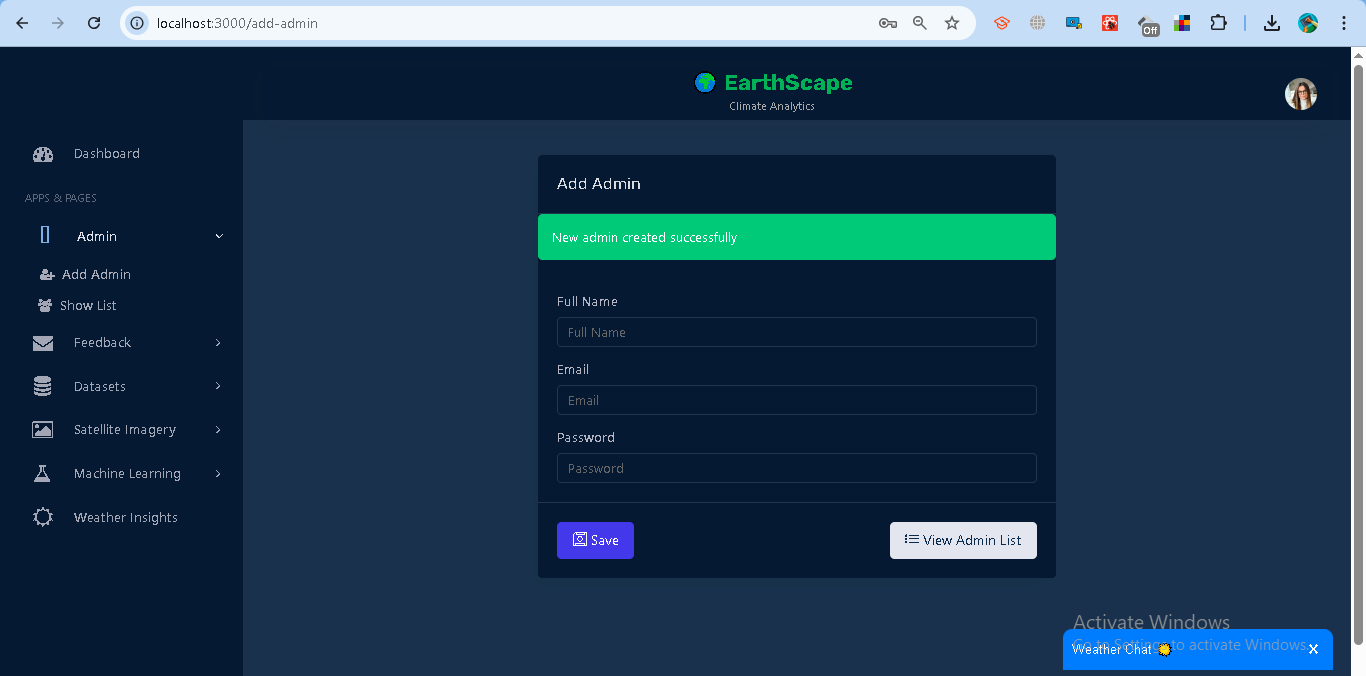
The Admin Management module allows only users with the **admin role** to manage administrator accounts.

### **Features**

**Add Admin**

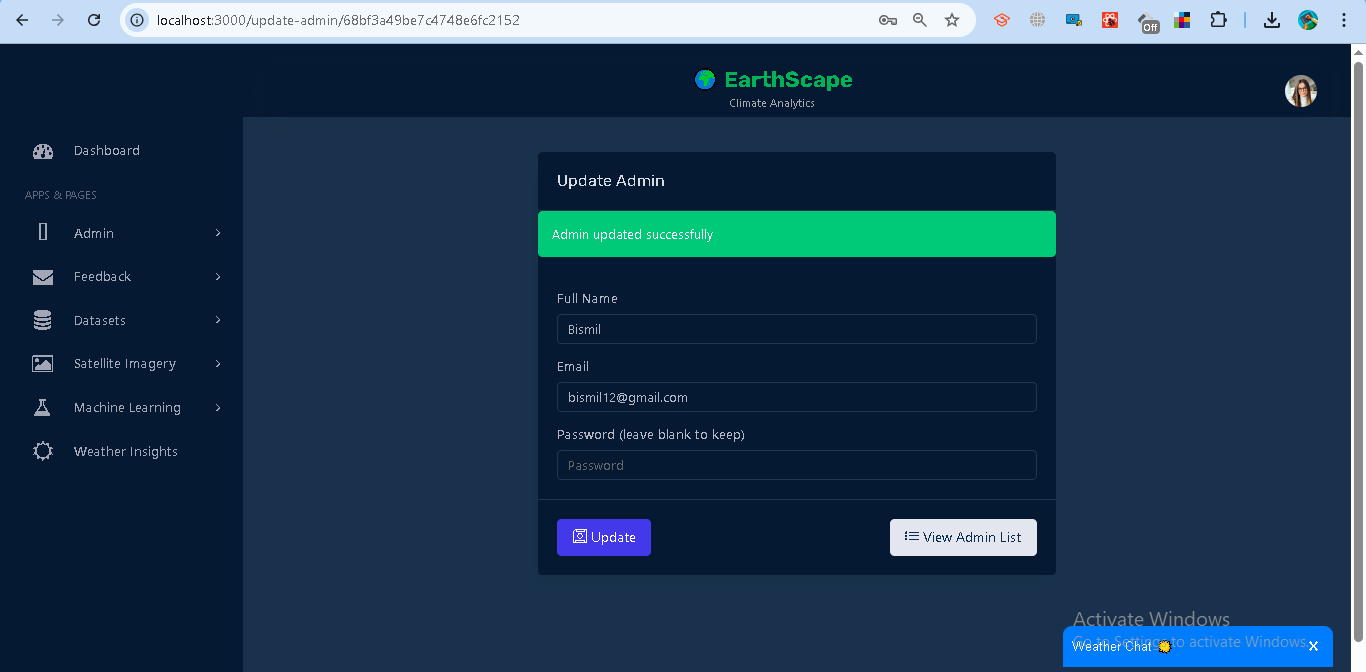
* Accessible only to admin users.
* Enter **Full Name, Email, Password**.
* On success, the new admin is stored in the system and a confirmation is displayed.





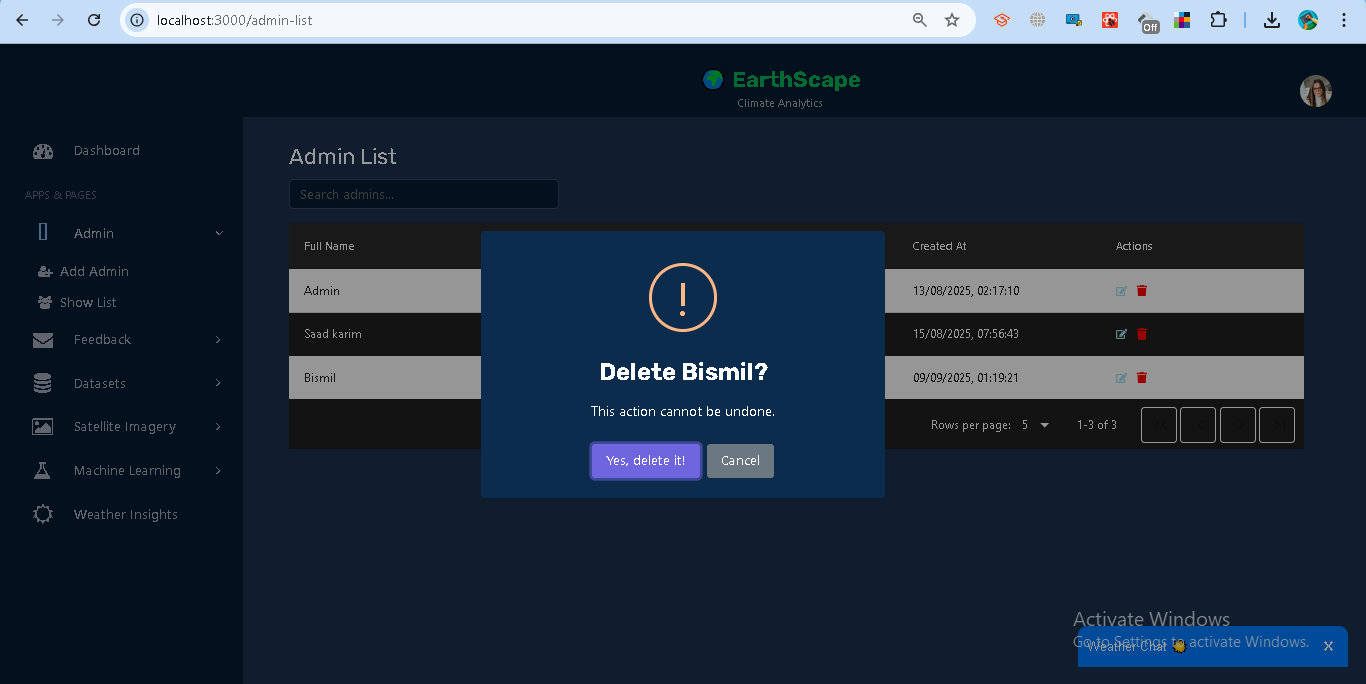
**Update Admin**

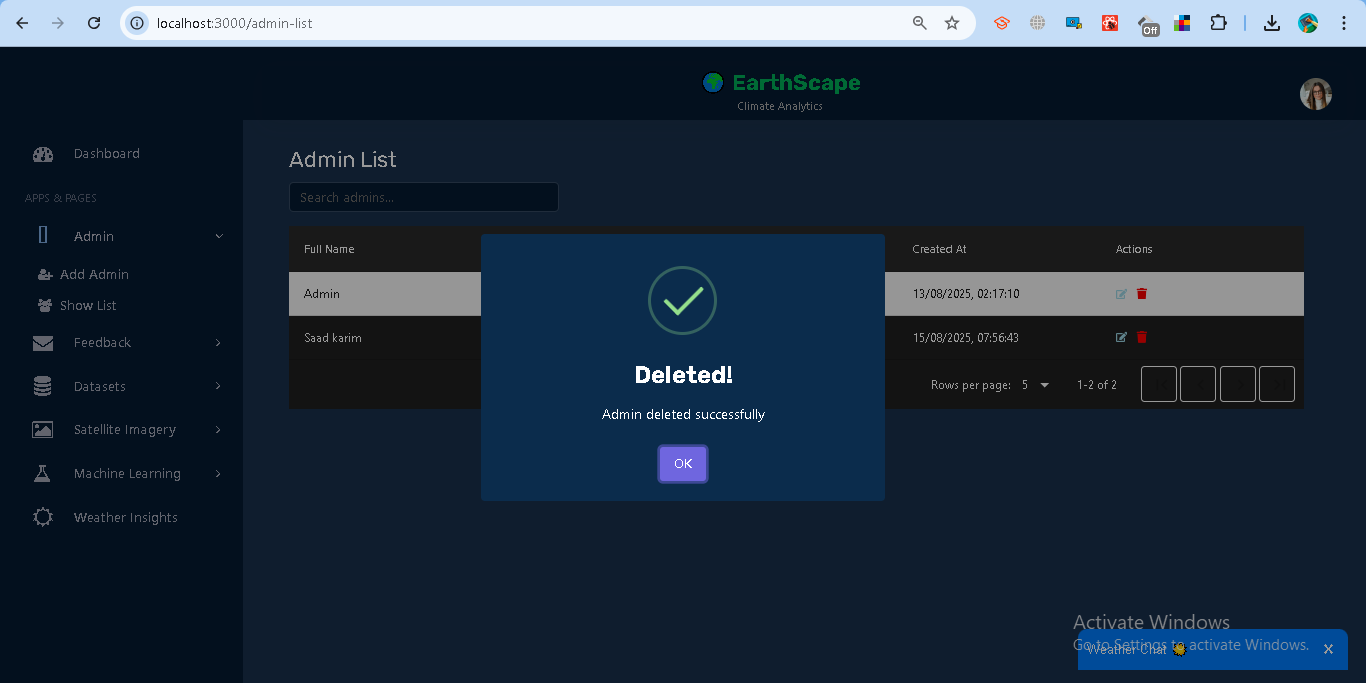
* Admins can update existing admin details such as **Full Name, Email, Password**.
* Requires selecting an admin from the list and saving changes.



**Delete Admin**

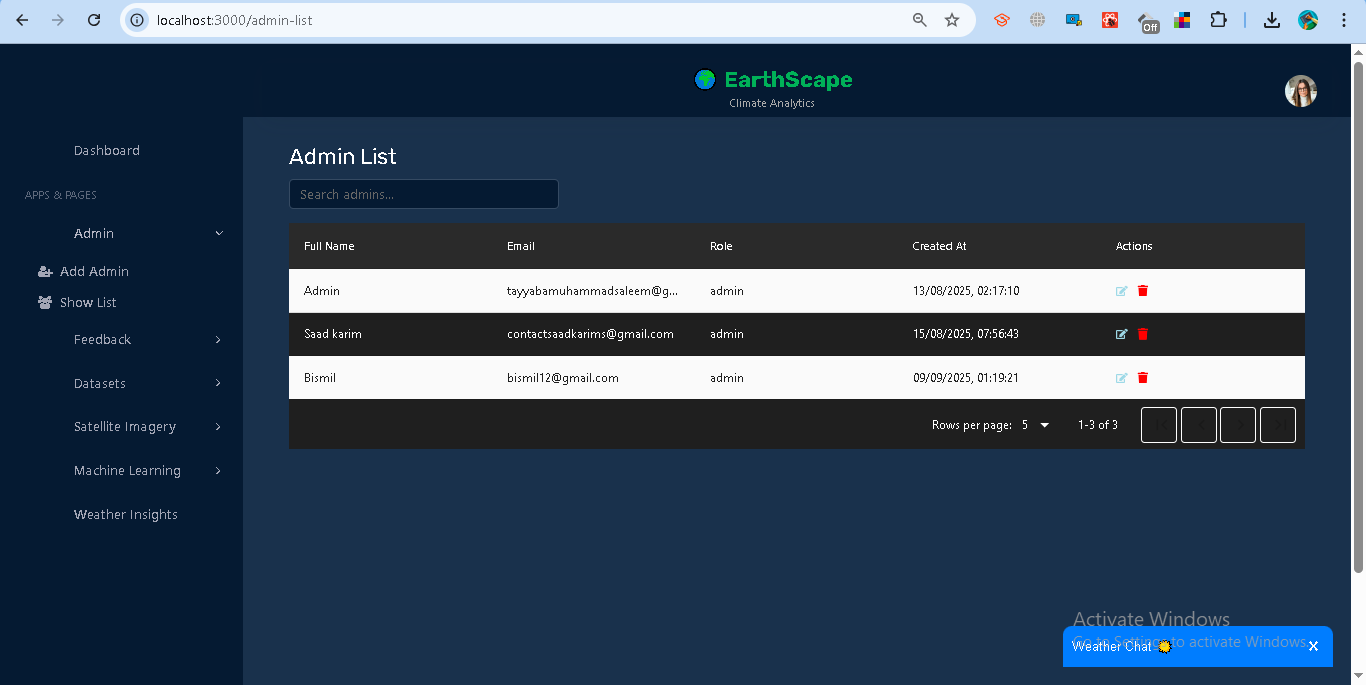
* Admins can remove an existing admin from the system.
* Requires confirmation before deletion.





**View Admin List**

* Displays all registered admin users.
* Provides options to update or delete each admin.



### **Access Control**

* Only users with role = **admin** can access this module.
* Non-admins are redirected and shown an **Access Denied** warning.

### **Error Handling**

* **Missing Fields** → Prompts user to fill all required inputs.
* **Unauthorized** → User must log in to access.
* **Forbidden** → Only admins can perform this action.
* **Server Errors** → Displayed with appropriate error messages.

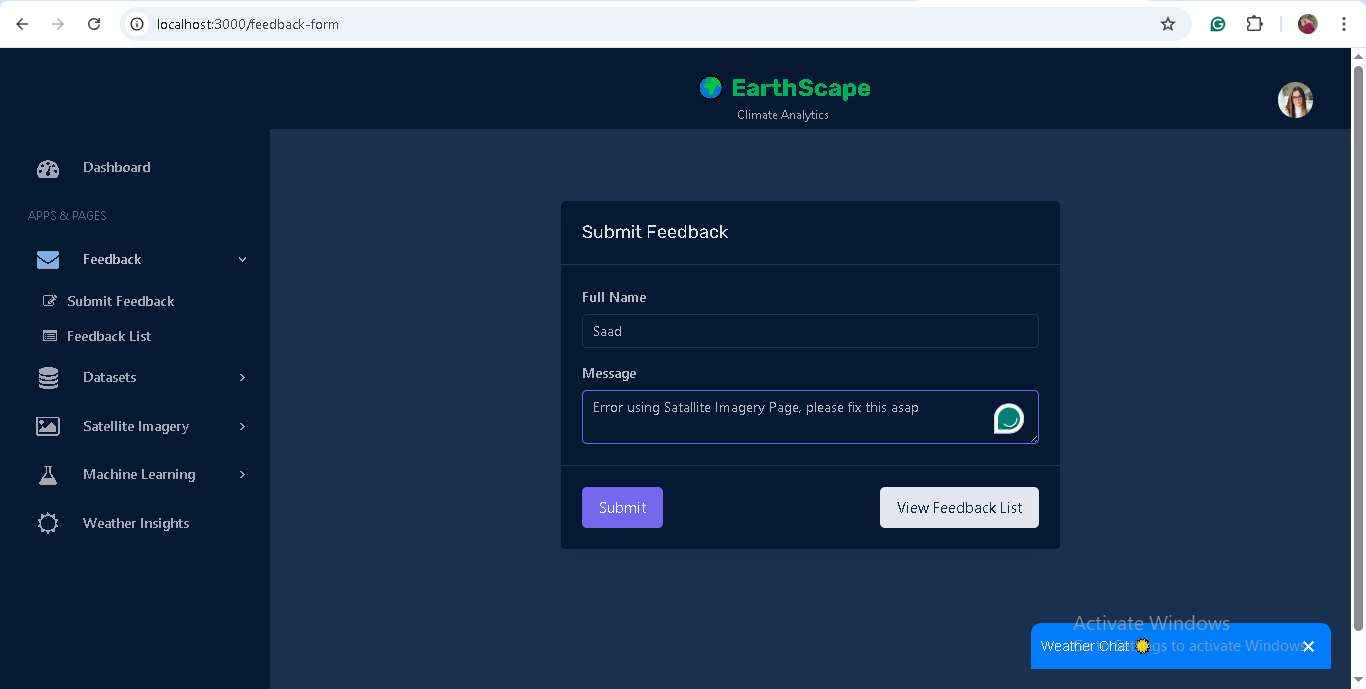
## **1.13 Feedback Management**

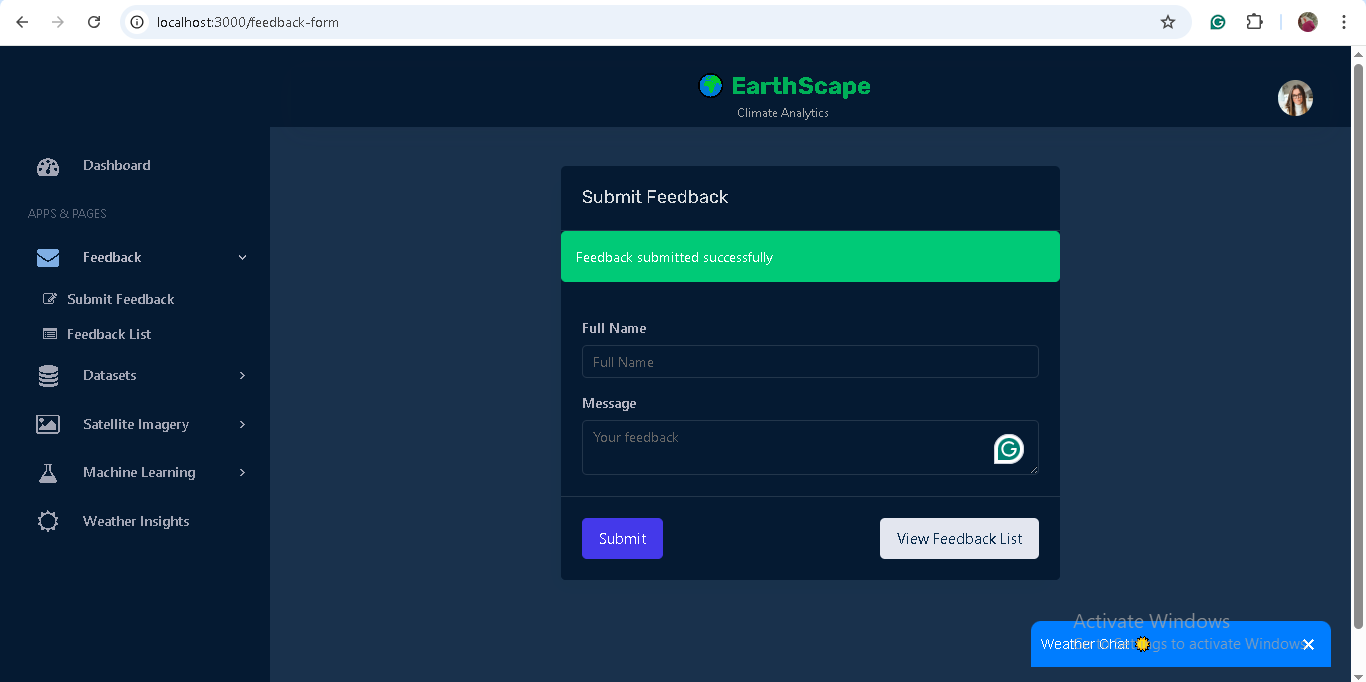
The Feedback module allows users to submit feedback and track its status, while administrators can manage and resolve feedback entries.

### **Features**

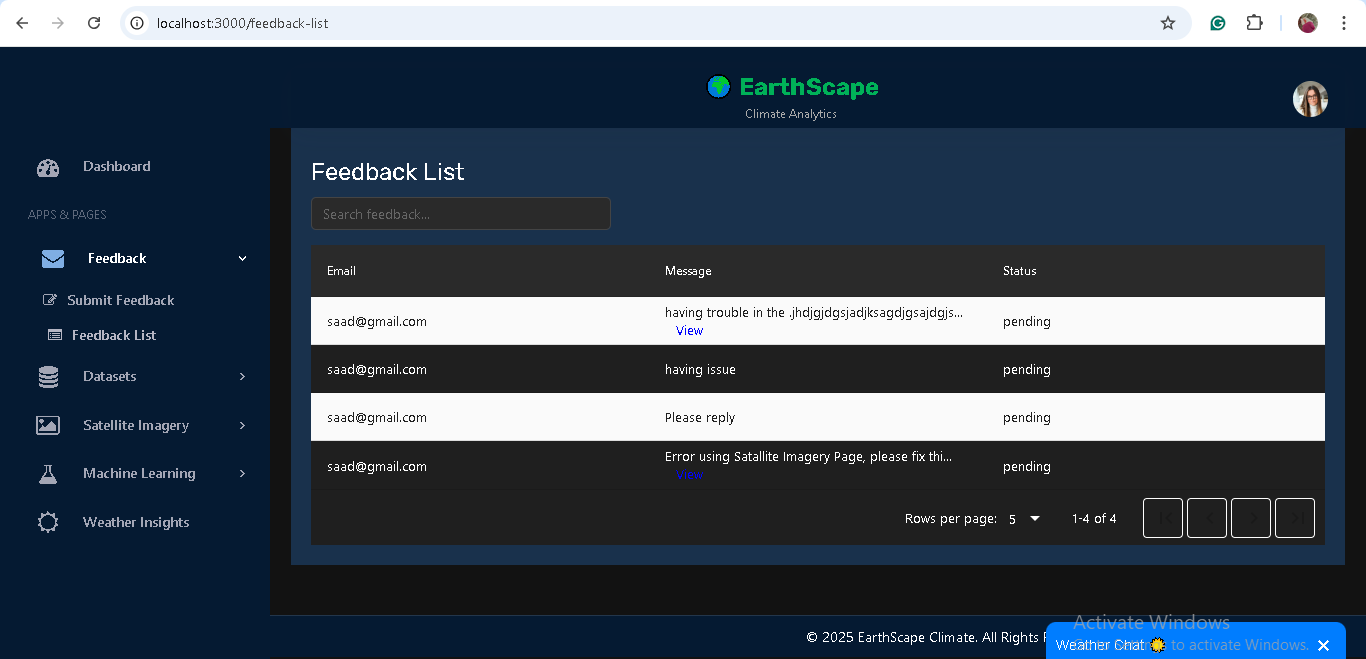
**For Analysts (Normal Users):**

* **Submit Feedback**
  + Enter **Full Name** and your **Message**.
  + The system automatically attaches your logged-in email.
  + On success, you see: *“Feedback submitted successfully.”*

**

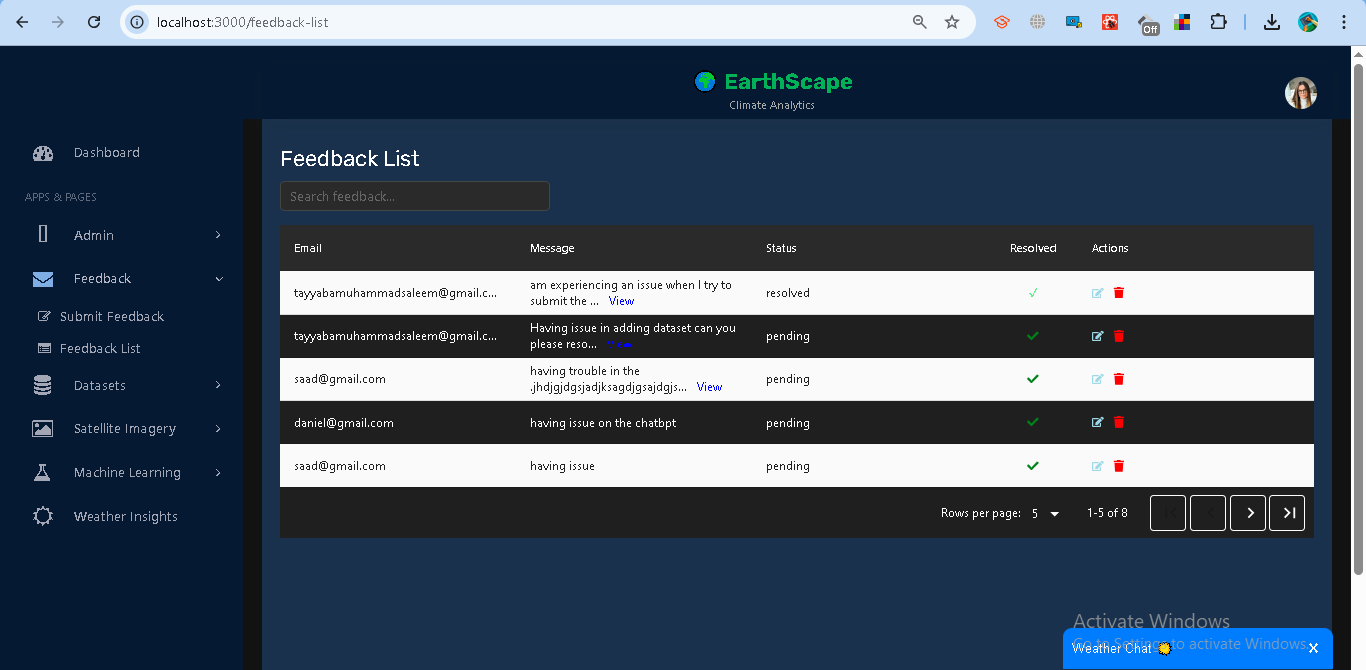
**

* **View Feedback List**
  + Shows all feedback you have submitted.
  + Displays **message, status, and date submitted**.
  + Status may be *Pending, In-Progress,* or *Resolved*.

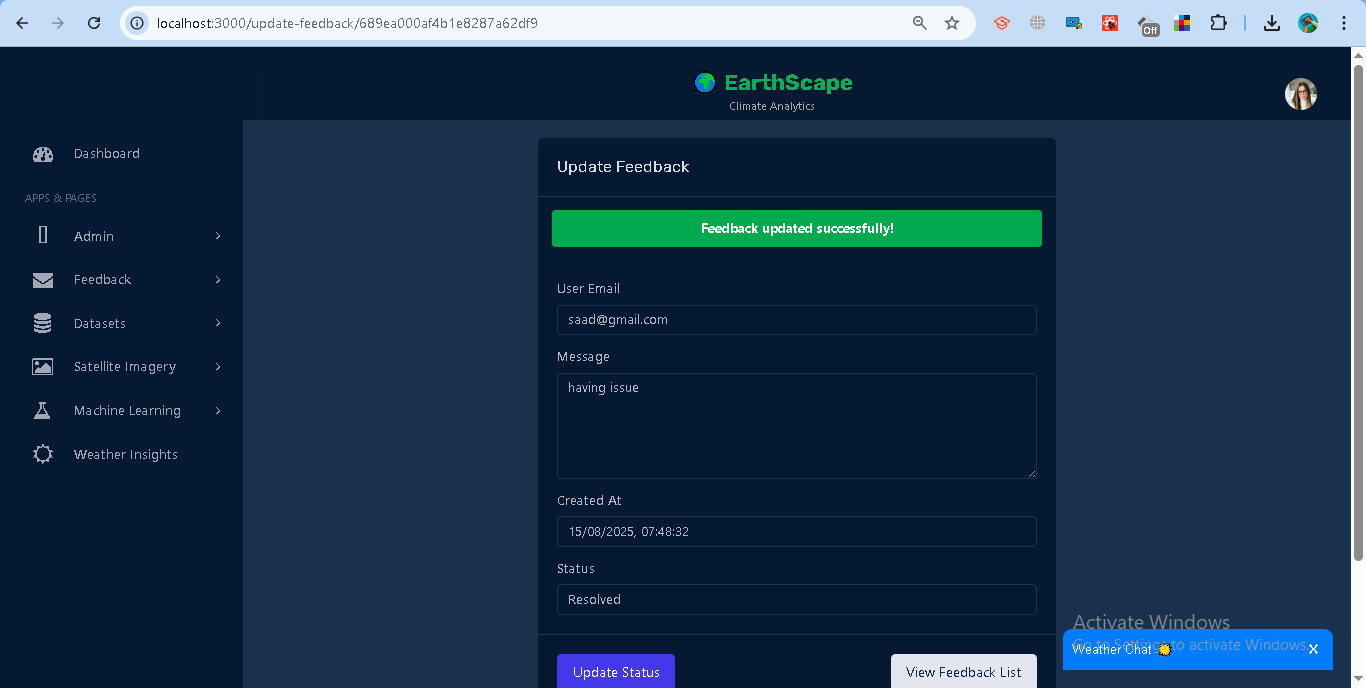


**For Administrators (Admin Role):**

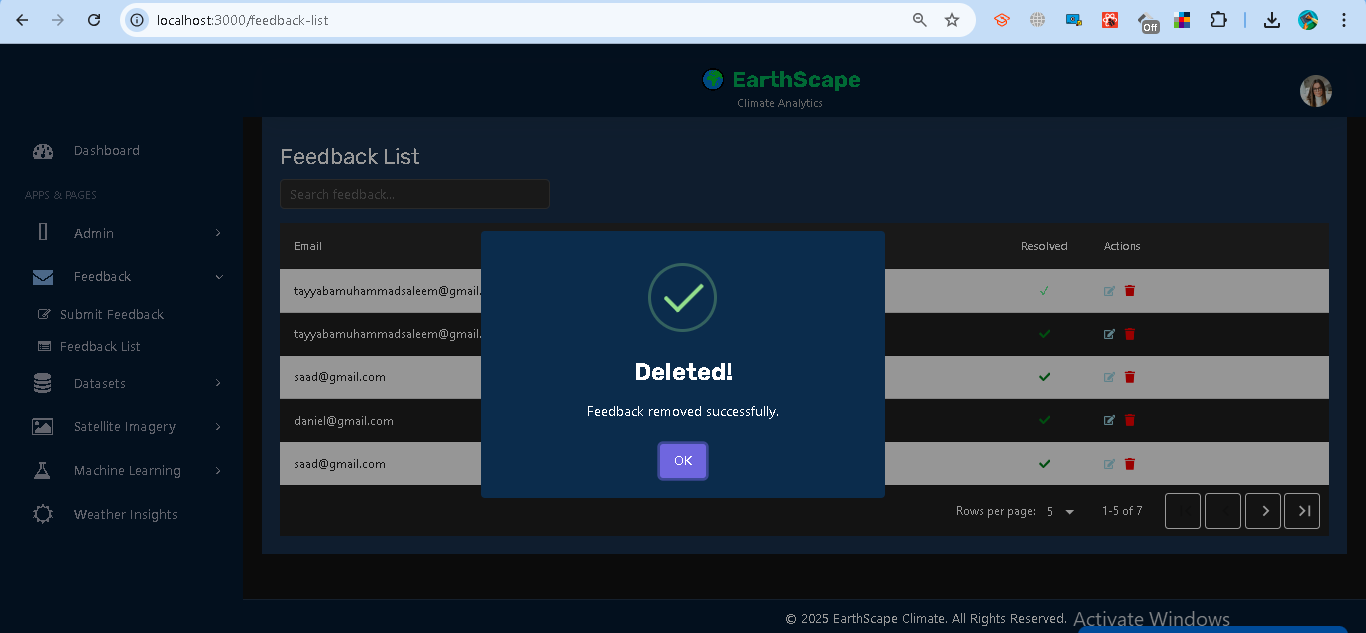
* **View All Feedback**
  + Access to all feedback submitted by any user.
  + Useful for monitoring system-wide feedback.



* **Update Feedback Status**
  + Change status of feedback to *Pending, In-Progress,* or *Resolved*.
  + Helps track progress of reported issues or suggestions.



* **Delete Feedback**
  + Remove feedback entries when necessary.
  + Requires confirmation to prevent accidental deletions.



### **Access Control**

* **Analyst** → Can submit feedback and view their own submissions.
* **Admin** → Full access (view all, update status, delete).

### **Error Handling**

* **Missing Fields** → Prompts users to fill all required fields.
* **Unauthorized** → Requires login to submit or view feedback.
* **Forbidden** → Only admins can manage feedback beyond their own.
* **Invalid ID / Server Errors** → Displayed with appropriate error messages.