**IAIDB Frontend Documentation**

**Version 1.4.1**

**Index**

1. **Objective** ................................................................. Page 3
2. **Frontend Structure** ................................................ Page 3
   1. User Interface & Navigation
   2. Loading Indicators
   3. Google Maps Initialization
   4. Unauthorized Access Handling
   5. System Crashes
   6. Technology Stack & Versions
3. **List of Endpoints** ................................................... Page 6
   1. Onboarding & Password Recovery
   2. Open to All Authenticated Users
   3. Data Management
   4. User Management
4. **Authorized User – Full Layout Structure** ........... Page 7
5. **Onboarding & Password Recovery** ..................... Page 10
   1. Login
   2. Sign Up
   3. Forget Password
   4. Reset Password
6. **Open to All Authenticated Users** ......................... Page 19
   1. MapView
   2. Open Data
7. **Data Management** ................................................. Page 33
   1. AIM Data
   2. NOTAM Data
   3. Shared Files
8. **User Management** ................................................. Page 52
   1. External User
   2. Internal User

## **Objective**

The document provides an overview of the IAIDB project **(version 1.4.1)**, detailing the front-end implementation and the APIs it interacts with. It outlines key functionalities, API endpoints, and their role in ensuring a seamless user experience. Additionally, it covers system integration, UI considerations, and performance optimization methods.

## **Frontend Structure**

The front-end application is built using **React** with **Vite**, a fast and efficient development tool. React enables seamless interaction with API calls, manages routes, handles user roles, and efficiently reacts to events.

Key aspects of the front-end structure include:

### **User Interface & Navigation**

After logging in, users can easily navigate between different sections using the top bar and sidebar for quick access. The sidebar, located on the left side of the screen, features the ADL logo and allows users to switch between pages depending on their role. The footer contains a copyright notice: *Copyright © 2023-24 ADL Coordinates Pvt. Ltd. All rights reserved*. It also includes a hyperlink to the ADL Coordinates website for easy navigation. The content section dynamically updates based on the user role and API responses.

Pages that are unauthorized do not include any layout elements.

### **Loading Indicators**

For improved user experience, whenever a React component requires additional time to load data, a blue-colored circular loader is displayed. This loading indicator is consistently available across all pages of the website, ensuring that users are aware of ongoing data processing.

### **Google Maps Initialization**

As the application uses Google Maps, it requires initialization to access the maps. To optimize performance, the Google Maps API is initialized when the user visits the website for the first time, ensuring smooth and efficient map interactions throughout the user session.

### **Unauthorized Access Handling**

If an unauthorized user attempts to access a restricted page, they are automatically redirected to the */auth/404/*page, preventing unauthorized access and ensuring proper redirection.

### **System Crashes**

In case of React failures due to issues such as data retrieval errors or logical problems, a dialog box is displayed to inform users about the crash. The dialog prompts users to re-login to restore functionality.

### **Technology Stack & Versions**

The following are the key technologies, and their versions used in the project, ensuring a stable and efficient development environment:

* **Node.js**: v20.12.2
* **React**: v18.2.0
* **Vite**: v4.5.5
* **Material UI (MUI)**: v5.10.16
* **MUI Icons Material**: v5.8.4
* **MUI X Tree View**: v7.21.0
* **Axios**: v1.7.7
* **JSZip**: v3.10.1
* **JWT Decode**: v4.0.0
* **Lodash**: v4.17.21
* **Proj4**: v2.12.1
* **Shapefile**: v0.6.6
* **Google Maps API**: Version alpha (Libraries: maps3d, drawing)

## **List of Endpoints**

### **Onboarding & Password Recovery**

1. LOGIN – Handles user authentication **(/)**.
2. SIGNUP – Registers new external users **(/signUp)**.
3. FORGOT PASSWORD – Allows users to reset their passwords **(/forgotpassword)**.
4. RESET PASSWORD – Confirms and updates new passwords **(/resetpassword)**.
5. CREATE USER – Sign Up for Internal Users (/createUser).

### **Open to All Authenticated Users**

1. MAPVIEW – Displays 2D and 3D maps to the viewers data **(/v1/mapView)**.
2. OPENDATA – Provides access to publicly available data **(/v1/openData)**.

### **Data Management**

1. AIM DATA – Handles Aeronautical Information Management (AIM) data **(/v1/aimData)**.
2. NOTAM DATA – Handles NOTAM (Notice to Airmen) data **(/v1/notamData)**.
3. SHARED FILES – Manages shared files depending on category **(/v1/sharedFiles)**.

### **User Management**

1. EXTERNAL USER – Manages external users **(/v1/externalUserManagement)**.
2. INTERNAL USER – Manages internal users **(/v1/internalUserManagement)**.

## **Authorized Users – Full Layout Structure**

For all authorized users, every screen includes a top header, a left sidebar, a main content area in the center, and a bottom footer, ensuring seamless navigation and a consistent user experience.

### **Header -**

The header provides a structured and user-friendly navigation experience. It consists of three key sections:

#### **Centered Title**

* The title “IAIDB” is centrally positioned for a balanced layout.

#### **User Information Display**

* The logged-in user’s first name, last name, and role are displayed on the right side.

#### **Sign-Out Option**

* A power icon button provides access to a dropdown menu.
* Clicking the Sign Out option logs the user out, clears session data, and redirects them to the login page.

### **Sidebar -**

The sidebar provides structured navigation, allowing users to access different sections of the website based on their roles. It consists of the following sections:

#### **Logo**

At the top, an image of ADL is displayed as branding.

#### **Navigation Options**

**Common Navigation**: These options are available to all users, regardless of their role.

**Dynamic Navigation**: Additional links appear based on the user's assigned role, ensuring role-based access control.

##### **Common Navigation:**

These options are available to all users, regardless of their role:

1. Map View – navigate to */v1/mapView*
2. Open Data – navigate to */v1/openData*

##### **Dynamic Navigation:**

The navigation options dynamically adjust based on the logged-in user's role, ensuring access to only relevant features.

###### **Role-Specific Options:**

**Originator Role:**

Dropdown "Import Data" section includes access to:

1. Import AIM Data - navigate to */v1/aimData*
2. Import NOTAM Data - navigate to */v1/notamData*
3. Import Shared Files - navigate to */v1/sharedFiles*

**Validator Role:**

Dropdown " Validate Data" section includes access to:

1. Validate AIM Data - navigate to */v1/aimData*
2. Validate NOTAM Data - navigate to */v1/notamData*
3. Validate Shared Files - navigate to */v1/sharedFiles*

**Admin Role:**

Dropdown "Manage Data" section includes access to:

1. NOTAM Data - navigate to */v1/notamData*
2. Shared Files - navigate to */v1/sharedFiles*

Gains access to an additional "Manage User" section, which includes:

1. External User - navigate to */v1/externalUserManagement*
2. Internal User - navigate to */v1/internalUserManagement*

These options dynamically update depending on the role, ensuring users only see what is relevant to them.

### **Main Content -**

The main content area dynamically displays relevant information based on the user's role and the current URL. It occupies the majority of the screen, ensuring a seamless and responsive experience. All page-specific data updates within this section, adapting to user interactions and role-based access.

### **Footer -**

This section represents the footer of the application. It is centered and includes a copyright notice for the years 2023-24. Additionally, it provides a clickable link to the **ADL Coordinates Pvt. Ltd** website which is <https://www.adlcoordinates.com/>. The footer maintains a structured layout with proper spacing and styling to ensure clear visibility.

## **Onboarding & Password Recovery**

### **LOGIN (/) -**

**Access By:** This page is accessible by every user, including non-registered users.

**Overview:** The Login page serves as the entry point to the system, requiring users to provide their credentials. It includes fields for **Username** and **Password**, along with essential validations. Users are redirected based on their roles after logging in.

#### **UI Layout**

The page layout consists of the following sections:

* **Left Section**: Displays a background image for visual appeal.
* **Right Section**: Contains the login form with two input fields, a Login button, and navigation options for Sign Up and Forgot Password. Users can toggle password visibility for ease of input. If credentials are incorrect, they receive a clear error notification above the login button. It also checks if the fields are empty when the login button is clicked. The logo is displayed along with the text "Welcome To IAIDB" at the top. At the bottom of the form, the current version of the application is displayed.

#### **Login Functionality**

The login process includes the following key elements:

* Users must enter both their **Username** and **Password** to proceed.
* If login details are incorrect, an error message is displayed.
* Upon successful login, user details are stored, and navigation is performed based on the assigned role, with a token used for authentication.
* If there is a **server issue** or **connectivity problem**, an error message is shown.
* Fields marked as required have an asterisk (\*) to indicate **mandatory inputs**. In this scenario, both fields are mandatory.

#### **API calls and Navigation**

1. **Login Button:**

* Sends a request to */api/Users/login* on port 6002 for authentication.
* Displays a loading indicator while processing the user's credentials.
* Upon successful login, users are redirected based on their role:
  + **Admin**: Redirected to the Internal User Management page at */v1/internalUserManagement*.
  + **Non-Admin**: Redirected to the Map View page at */v1/mapView*.
* If the login details are incorrect, an error message will prompt the user to try again.

1. **Sign Up Link:**

* Takes the user to the Sign-Up page to create a new account.

1. **Forgot Password Link:**

* Redirects the user to the password recovery page.

### **SIGN UP (/signUp) -**

**Access By:** This page is accessible by every user, including non-registered users.

**Overview:** The Sign-Up page enables new external users to create an account by providing their personal details and login credentials. It includes fields for First Name, Last Name, Username, Email, Password, and Confirm Password. The system ensures accuracy and security by validating the entered information.

#### **UI Layout**

The page layout consists of the following sections:

* **Left Section**: Displays a welcome message or branding with a brief description of the platform’s benefits.
* **Right Section**: Contains the sign-up form with input fields for First Name, Last Name, Username, Email, Password, and Confirm Password, along with a "Sign Up" button. Users can toggle password visibility for the Password field for ease of input, but the Confirm Password field does not have this feature. If the user enters incorrect details, an error notification is displayed above the "Sign Up" button. The page also includes navigation options for Sign in and Forgot Password.

#### **Sign Up Functionality**

The login process includes the following key elements:

* Users must enter all required details: **First Name, Last Name, Username, Email, Password, and Confirm Password.**
* Clicking the "Sign Up" button triggers the validation process.
* Ensures the email format is valid.
* Password must meet the minimum-security criteria (at least **8 characters**).
* Confirms that "Password" and "Confirm Password" fields match.
* If validation fails, an error message is displayed.
* Upon successful sign-up, user details are stored, and navigation to the login page occurs.
* If there is a **server issue** or **connectivity problem**, an appropriate error message is shown.
* Fields marked as required have an asterisk (\*) to indicate **mandatory inputs**.

#### **API calls and Navigation**

1. **Sign Up Button**:

* Sends a request to */api/Users/ExternalUserSignup* on port 6002 with the data from the respective input fields.
* Displays a loading indicator while processing the request.
* Upon successful sign-up, users are redirected to the Login page *(/).*

1. **Sign In Link**:

* Takes the user to the Login page to access an existing account.

1. **Forgot Password Link**:

* Redirects the user to the password recovery page.

### **FORGET PASSWORD (/forgotpassword) -**

**Access By:** This page is accessible by every user, including non-registered users.

**Overview:** The Forgot Password page allows users to reset their password by entering their registered email address. Upon submitting, a loading indicator is shown, and a success or error message is displayed based on the outcome.

#### **UI Layout**

The form is placed in the middle of the screen, with the "Submit" button directly below the text field to enter Email. While the user submits the request, a loading indicator appears on the button. Additionally, a snackbar at the top-right corner notifies the user of the success or error status.

#### **Forget Password Functionality**

* When a registered email is entered and the submit button is clicked, a request is sent to the respective API to handle the password reset.
* If the email is valid and the reset request is successful, a success message is displayed, and the user is redirected to the login page after a brief delay.
* If an error occurs, an error message is shown to the user.
* A loading indicator is displayed while the request is being processed.

#### **API calls and Navigation**

1. **Submit Button:**

* Sends a request to */api/Users/SendForgotPasswordLink?email=<email here>* on port 6002 to initiate the password reset process.

### **RESET PASSWORD (/resetpassword?email=<email>) -**

**Access By:** The Link provided by the email to access the reset password.

**Overview:** The Reset Password page allows users to reset their password by entering a new Password and Confirm Password. Upon submitting, a loading indicator is shown, and a success or error message is displayed based on the outcome.

#### **UI Layout**

The form is placed in the middle of the screen, with the "Submit" button directly below two text fields labelled as New Password and Confirm Password. While the user submits the request, a loading indicator appears on the button. Additionally, a snackbar at the top-right corner notifies the user of the success or error status.

#### **Reset Password Functionality**

* Two text fields "Enter new password" and "Confirm new password" are present to add the respected value.
* If both fields match, an API request is sent; otherwise, the message "Passwords do not match" is displayed.
* If successful, the message "Password reset successfully" is displayed, and the user is redirected to the login page after 3 seconds.
* If an error occurs, the message "An error occurred while resetting the password" is displayed.

#### **API calls and Navigation**

1. **Submit Button:**

* Sends a request to */ api/Users/ResetPasswordByMail* on port 6002 to initiate the password reset process.

**5. CREATE USER (/createUser?email=<email here>) -**

**Access By:** The Link provided by the email to access to fill internal user details.

**Overview:** The Create User page is designed to allow new internal users to register and create their account within the system. It provides a simple and intuitive form for users to enter their personal and account details. Upon submission, the system processes the information and provides feedback to the user through success or error messages.

#### **UI Layout**

The page is visually divided into two main sections:

##### **Welcome Section (Left Side):**

* A warm greeting is displayed at the top:
* Hello! Welcome to IAIDB. Kick-start your journey to the IAIDB system!
* Below the greeting, there is a prompt for existing users:
* Already have a User Account? Login here
* The Login here text is clickable and redirects users to the login page.

##### **Form Section (Right Side):**

* A clean and bordered form is displayed, containing the following fields:
* First Name: A text input field for the user to enter their first name.
* Last Name: A text input field for the user to enter their last name.
* Enter Username: A text input field for the user to create a unique username.
* Email Address: A pre-filled, non-editable field displaying the user's email address (taken from the link).
* Enter Password: A password input field for the user to set their account password.
* Designation: A text input field for the user to specify their designation.
* At the bottom of the form, there is a CREATE button. Clicking this button submits the form and initiates the account creation process.

##### **Feedback Messages (Snackbar):**

* After submitting the form, users will see a feedback message at the top-right corner of the screen.
* If the account is created successfully, a success message is displayed, and the user is automatically redirected to the login page after a few seconds.
* If there is an error (e.g., invalid input or server issue), an error message is shown with details about the problem.
* The feedback message automatically disappears after a few seconds, but users can also dismiss it manually.

#### **Create User Functionality**

##### Process the detail filling:

* The user must fill in all required details, including **First Name, Last Name, Username, Email Address, Password, and Designation**.
* The **Email Address** is automatically pre-filled from the URL, which follows the format:  
   */createUser?email=<email>*.
* When the **"CREATE"** button is clicked:
  + The system first checks if all fields are filled.
  + If any field is missing, the user is prompted to complete it before proceeding.
* Once all required details are provided, the system sends a request to create the account.

##### Response of the API and redirecting to login:

* After clicking the **"CREATE"** button, the system processes the request to create a new user account.
* If the account is created successfully:
  + A **greenish notification** appears at the top-right corner with the message: **"User created successfully!"**
  + After **3 seconds**, the page automatically redirects to the **login screen**.
* If there is an issue with the provided details (e.g., email already exists):
  + A **red notification** appears with the specific error message from the system.
* If the process fails due to a technical issue:
  + A **red notification** appears with the message: **"Error creating user."**
* Notifications disappear **automatically after 6 seconds** or can be closed manually.

#### **API calls and Navigation**

1. **Create Button:**

* Sends a request to **<fill-the-api>** on **port 6002** to initiate user creation.
* The request includes the following details in the payload:

{

"firstName": "<user's first name>",

"lastName": "<user's last name>",

"userName": "<chosen username>",

"email": "<user's email>",

"password": "<user’s password>",

"designation": "<user's designation>"

}

* Once the request is processed, the system handles the response and navigates accordingly.

## **Open to All Authenticated Users**

### **MAP VIEW (/v1/mapView) -**

**Access By:** All the authenticated users can access this page.

**Overview:** This page provides both 2D and 3D interactive maps, allowing users to visualize data dynamically. Users can draw shapes like circles, rectangles, and polygons directly on the map, as well as upload .shp, .kml, or .kmz files to plot values. A filtering feature enables users to selectively display specific data points on the map.

#### **UI Layout**

Let's divide the UI in four sections for better understanding:

##### **Map Menu content**

The content is located at the top left of the page. Clicking the button will reveal two sections: Filter and My Preferences.

###### **Filter section**

* With the title **Selection Based on Geographic Extent**, a file upload option is provided. Under the "Browser" label, there is a "Choose File" button for inserting the file. Initially, the button shows "Upload AOI" to initiate the upload. If a file has already been uploaded, the button changes to "Reset" to remove the file, and the name of the uploaded file is displayed in place of ‘No file chosen,’ along with its extension.
* With the title **Selection Based on Aeronautical Features**, a list of predefined categories, such as 'Airspace', 'Aerodrome', 'Navaid', 'ATS and Other Routes', and 'Obstacles', is presented to the user with checkboxes. All options are initially selected, and each feature is displayed in a grid layout. When a feature is selected, clicking it again will deselect it, allowing users to filter the data by selecting or deselecting options.
* At the bottom of the Filter section, there are three buttons:
* **Add Preference Button**: Clicking this button opens a dialog box with a text field for entering the file name. After entering at least one character, the "OK" button becomes enabled, and the "Cancel" button is available to close the dialog without saving.
* **Export Button**: Clicking the Export button reveals three options at the bottom right of the button itself. These options are AIXM, Spreadsheet, and GeoJSON.
* **Switch Data File / View Approved List**: For **non-admin** users, the 'Switch Data File' option is available. When clicked, a dialog opens asking for confirmation to switch to the previous version. For **admin** users, the 'View Approved List' option appears. Upon selection, a list of all approved data is displayed. Initially, the current file is shown as the selected option. When an item from the list is clicked, the UI’s focus shifts to that data. Each entry shows the file name along with the updated date and time. The dialog for both non-admin and admin users includes 'Cancel' and 'Switch' buttons. If there are no approved files, 'No Files Available' is displayed for non-admin users, and the button is disabled if the file list contains fewer than two entries.

###### **My Preference section**

* A loader is displayed until the API retrieves all the data from the backend. The text 'Loading Preferences' is shown, and a loading style is applied. Once the file names and data are retrieved, they are presented as a list in this section, with a delete button on the right. When an item is selected, a tick mark appears next to it, indicating which one the user has clicked, and the selection is also reflected on the map.

##### **Drawing tools for 2D map**

At the top-center of the 2D map, there is a Drawing Options toolbar featuring three shape options: Circle, Rectangle, and Polygon, each represented by an icon. When a user clicks on any shape, it becomes selected, indicated by a subtle highlight (a semi-transparent white background) around the icon, making it clear which tool is active. Only one shape can be selected at a time. Once a shape is selected, the mouse cursor changes to a '+' sign, allowing the user to draw the chosen shape on the 2D map. After completing the drawing, the toolbar updates to display a Reset button labeled "Reset Filter" in place of the shape options. The Reset button has a blue background and white text, and clicking it clears the drawn shape and restores the original shape options. The toolbar has a clean, modern design with a dark blue background and white icons, ensuring high visibility and usability. Hovering over the shape options or the Reset button triggers a slight color change, providing visual feedback for interactivity.

##### **Toggle between 2D and 3D**

At the bottom-left, there is a button to switch between the 2D map and 3D map. When switching from 2D to 3D, the button text displays "Show 3D Map", and when switching from 3D to 2D, it displays "Show Map".

##### **Map to view**

The maps are built using Google Maps components, which display data on the screen. Initially, a 2D map is used to visualize the data, with line, point, and polygon data sourced from the API response. Each item updates the map based on specific actions. The map spans the entire screen and serves as the background for all the buttons mentioned above. To display information on the 2D map, we use the Advanced Marker option, which is essentially a short HTML code placed at specified locations. This helps present relevant information effectively. For the 3D map, an alert is used, with the color determined by the section and the information based on the corresponding trigger. The polygon data includes only the stroke color for the edges, omitting the fill color. This approach enhances the visibility of smaller polygons nested within larger ones.

#### **Map View Functionality**

To better understand the functionalities, we will explore them in terms of user actions on features and their corresponding reflections on the map. Every interaction, such as adding, removing, or modifying data, dynamically updates the map in real time. Since all functionalities are directly related to the map, any changes made by the user will be immediately reflected in the view.

##### **Initial Map Setup and Key Norms:**

When a user first visits the endpoint, we retrieve a list of all approved files. From this list, we identify the most recently updated file and fetch **line, polygon,** and **point** data while loading both the 2D and 3D maps in parallel. As the data is received, we use the **lodash** library to group it by gml\_Id, ensuring efficient organization and processing. This grouped data is then plotted on the map, with colors assigned based on the section and its retrieval source, using gml\_Id as a reference for consistency and accuracy.

The colors used for differentiation are either provided by the API or predefined within the code. These colors are directly applied to Google Maps for both the 2D and 3D views. For 2D maps the markers use Advanced Marker styled as pins to precisely indicate their location at specified latitude and longitude points, while polygons and polylines are rendered to represent area-based and linear data, respectively. Additionally, the Advanced Markers component is used to show detailed information about the data when a user clicks on a specific marker, polygon, or polyline. To ensure optimal visibility, LatLngBounds is employed to adjust the map view so that all data is visible to the user upon initial load.

The predefined color codes used in the map are as follows:

|  |  |  |
| --- | --- | --- |
| **Category** | **Representation** | **Color Code** |
| Aerodrome | Marker | #1976D2 |
| Navaid | Marker | #FFC107 |
| ATS | Marker | #E1231E |
| Obstacles | Marker | #000000 |
| Airspace | Polygon | rgba(255, 0, 255, 0.8) |
| Routes | Polygon | #FFFF00 |

One of the **key concepts** in Google Maps is that the unit of measurement is **meters**. Therefore, values such as the radius in NOTAM (measured in KM) or the Lower Limit in lines (measured in FL or FT) must be converted to meters.

For the 3D map, since users can tilt the view, additional detailing is required to define the altitude at which data should be displayed. In the current 3D implementation, we use two altitude modes: **ABSOLUTE** and **RELATIVE\_TO\_MESH**.

* **ABSOLUTE**: Used for plotted file data (.shp, .kml, .kmz) and for representing route lines. This mode takes altitude from Mean Sea Level (MSL) and displays it accordingly on the 3D map. For routes we are using the Lower limit of the line to represent it on the map.
* **RELATIVE\_TO\_MESH**: Used for Airspace data with an altitude of 10,000 meters. This mode positions objects relative to the highest surface available whether it's the ground, buildings, or water. Over water, the reference is the water surface and over land, it is the highest point, either a building or the terrain if no buildings are present.

##### **Toggle between 2D and 3D:**

The button displayed at the bottom left shows "Show 3D Map" when the 2D map is active and "Show Map" when the 3D map is active. When clicked, it ensures that only one map is visible at a time, seamlessly switching between the 2D and 3D views through applied styling options.

##### **Map Interactions:**

We have three types of data plotted on the map: polygon, line, and marker. When a user clicks on the 2D map, an information window appears with the details of the clicked item. For the 3D map, currently, only marker clicks are supported.

In the 3D map, when a user clicks on a marker, an alert window is generated. The position of the alert is determined based on the mouse location when it was triggered. We calculate the pixel coordinates of the click and position the alert relative to the top-left of the clicked area. The information displayed corresponds to the clicked marker, which was triggered, and the background color is determined based on the section where the marker is located. After four seconds, the alert is removed, and it will also disappear if the click occurs outside the alert.

In the 2D map, there are three distinct triggers to display data for polygons, lines, and markers, each activating based on the user's last click. The information is displayed through an information window style, which is part of the Advanced Marker component and provides detailed data. For polygons, the information window opens at the exact location where the user clicks. For lines, the window opens at both endpoints of the line. For markers, the window opens directly above the marker. These positions are dynamic and tied to latitude and longitude coordinates, ensuring the information is displayed accurately. Below are the triggers used in the 2D map:

###### **Click on the Polygon:**

When a user clicks inside a polygon, a trigger activates to determine whether the click is on a polygon or a line edge. The process prioritizes checking for polygons first. If no polygon is found, the system then checks for line data at the same clicked coordinates. This two-step approach ensures accurate data retrieval, as lines often overlap with polygons.

**Polygon Check:**

* The system checks within a tolerance range of 0.001 to 0.004 for both latitude and longitude.
* It evaluates all polygon points to identify the one closest to the user's click.
* If the point corresponds to multiple gml\_Ids, the system creates an array and adds the details to the information window.
* A switch button is activated when more than one gml\_Id is found. Initially, one polygon is highlighted and displayed in the information window. Clicking the switch button updates the window to highlight the next polygon.

**Line Check:**

* If no polygon data is found, the system checks for line data within a tolerance of 0.008.
* It verifies if the line has been triggered by the same latitude and longitude on both endpoints.
* If the line has two or more gml\_Ids, the system adds the corresponding information to the information window.

###### **Click on the line:**

This trigger comes from the Line itself, which detects the line on the 2D map and identifies the triggered line by checking the latitude and longitude of both its endpoints. If multiple lines share the same latitude and longitude for both ends, they are grouped together, following a process similar to the Airspace trigger. The output displays two information windows—one at each endpoint of the line like the Airspace trigger of line when the clicked point is close to the line.

Additionally, when no polygon is present around a line, the cursor changes to a clicking hand icon, providing a visual cue to the user and improving the click interaction.

###### **Click on the marker:**

This trigger functions independently, accurately detecting markers even if they are inside a polygon. When a user clicks on a marker, the corresponding information window opens directly at the clicked marker's location.

##### **Drawing:**

We can draw on the map using 2D maps, but for 3D maps, drawing is unavailable. The drawing options for 2D maps are enabled through the DrawingManager component, which allows us to draw circles, polygons, and rectangles. These drawing tools are triggered based on user interactions. Since the DrawingManager appears as a tool over the map, we created a custom styling option over the existing one to make the drawing tool trigger slightly larger than the default size for better usability.

Only one drawing option is active at a time, allowing users to enable or disable it as needed. Once a user draws on the map, a Reset button replaces the three images. Clicking the Reset button restores the images and removes the drawn elements from the map.

When a user draws on the 2D map and completes the drawing, the same shape is also displayed on the 3D map. However, the 3D map only serves as a viewer, meaning users do not have access to the Reset button or the three drawing tool icons.

##### **File Upload:**

The file upload option can be found by clicking on the menu content option at the top left, where the title **Selection Based on Geographic Extent** is displayed. When selecting the **Choose File** option, users can upload files in .shp, .kml, or .kmz formats. These formats will be highlighted while browsing their file directory, as the upload is restricted to them.

Once a file is uploaded, the **No file chosen** text will be replaced with the file name, which will be displayed with a maximum of 25 characters, including the extension. If a user attempts to upload a file with an unsupported format, an error message **Please upload a valid KML, KMZ, or SHP file.** will be displayed with an **OK** button.

Upon clicking the **Upload AOI** button for a valid file, the system will first verify the file format and then process it accordingly.

The **KML processing** begins by parsing the XML content to extract coordinate data from the *<coordinates>* tags. Each coordinate string is trimmed, split, and further broken down into latitude and longitude values. These values are then converted into floating-point numbers, and any invalid entries are skipped. For each valid coordinate, a structured data object is created, including attributes such as a unique identifier, latitude, longitude, metadata details, section name, and a predefined color code. Since multiple coordinates can be found within a single KML file, the extracted data is flattened into a single list and filtered to remove any null values. Finally, the processed coordinate data is stored and made available for further use, ensuring seamless plotting on the map.

When handling a **KMZ file**, the process begins by extracting its contents using the **JSZip** library. The KMZ file, which is essentially a compressed ZIP archive, is first loaded asynchronously. Once the file is loaded, the system searches for the KML file inside the archive by identifying the file with a .kml extension. The name of the KML file is then used to retrieve its contents, which are extracted as a text string. This KML content is then passed through the same processing flow as a regular KML file. The overall process flows from KMZ to KML to valid plotting output.

When handling a **shapefile upload**, the process begins by extracting the coordinates from the shapefile. The initial data contains coordinates that need to be transformed to the appropriate zone for the region, currently using UTM Zone 40N, which covers the Middle East and is specifically required for the **Oman region**. The transformation is performed using the **proj4 library**, which is suited for the region's geodetic system. During this transformation, any invalid or empty coordinates are filtered out to ensure data accuracy. Afterward, the coordinates are structured with relevant metadata, such as a unique identifier, details, and a color for visualization purposes. Once processed, the structured data is made available for plotting on the map, allowing it to be visualized with additional contextual information.

After the shapefile, KML, or KMZ file is successfully uploaded and processed, the user will see the data plotted on the map. The "Upload AOI" button will change its text to "Rest," allowing the user to remove the current data from both the map and the underlying values. This action will also clear the file name from the display. Once the data has been removed, the user can upload a different file if needed.

It is important to note that the system restricts users from uploading only one file at a time to ensure smooth operation and prevent any conflicts during the upload and processing stages.

##### **Filtering the Category:**

The S**election Based on Aeronautical Features** option is available in the menu and allows users to filter based on features like 'Airspace', 'Aerodrome', 'Navaid', 'ATS and Other Routes', and 'Obstacles'. Initially, all features are selected by default, and users can remove them by clicking on the checkboxes. Each feature is displayed as a clickable option with a checkbox. The styling adjusts based on selection, with hover effects and transitions enhancing the user experience.

##### **Preferences:**

To view or add preferences, the user must first draw one of the available shapes: circle, rectangle, or polygon. To add a preference, there is a button labeled 'Add Preference' in the bottom left of the content menu. To view saved preferences, a separate tab is available at the top of the content menu.

When the user clicks the **Add Preference** button, a dialog opens where they are required to enter the name of the preference they want to save. The text field only requires at least one character to enable the 'OK' button. Clicking 'OK' will send the value to the API. After clicking either the 'Cancel' or 'OK' button, the dialog will be removed from the map.

When the user views the **My Preferences** section, a loader is displayed while the data is being loaded. If no preferences are found, a message saying "No preferences found" will be shown. If there are preferences, the name of each file along with its last modified date will be displayed. Each file has a delete icon on the right side of the block, and on the left, there's a select point that triggers the drawing on both the 2D and 3D maps.

##### **Export:**

The Export button is available in the content menu, and its position depends on the number of files in the approved list. For non-admin users, it is visible only if there are fewer than two files, in which case it appears at the bottom right. For admins, or when there are more than two files, the Export button is positioned at the bottom center.

After clicking on the Export button at the bottom right, the user is presented with options to export the data as AIXM, Spreadsheet, or GeoJSON. Clicking on any of these options triggers the respective API call to retrieve the data

The exported data shared with the API depends on both the file data and the drawn data. The response from the API contains a link to download the file. Once the link is received, the file is automatically downloaded to the end user's device. In cases where both the file and the drawn data are present, the drawn data takes priority over the file data

##### **Data File Switching:**

The functionality allows changing the map data to a different data file. For non-admin users, the button is disabled if the approved list has no values. If the approved list contains fewer than two entries, the button remains hidden.

When the approved list contains two or more entries, the button for **non-admin users** is enabled at the bottom right. Clicking on it displays a dialog with the message: 'Do you want to switch to the previous data file?'. If the user clicks 'OK', the other ID from the approved list is fetched, along with its line, polygon, and point data. Only two IDs exist in the approved list, so retrieving all data is unnecessary.

For **admin users**, there are no disabled criteria. If there is no approved list, the dialog displays the message: 'No Files Available. There are no files to display at the moment.' If at least one file exists, the currently displayed item on the map is selected by default. Users must choose a different file to switch. If the user selects the same file that is already in use, no API call is made. However, if a different file is selected, the system triggers the same API calls used initially to fetch polygon, line, and point data, but only for the newly selected file ID. The API is triggered when the user clicks the 'Switch' button. Both the 'Switch' and 'Cancel' buttons will close the dialog. However, the 'Switch' button will only close the dialog after all the data has been successfully fetched.

When fetching the data, the existing data currently displayed on the map is removed first. Once the new data is retrieved from the response after the switch, it is then loaded onto the map to replace the previous data.

#### **API calls and Navigation**

##### **Initial call:**

* We will retrieve the user's file list using the */ADL\_AIM\_DB/GetImportFileListByUser* API from port 6002.
* Pass the file ID to the following endpoints to fetch the respective data of port 6002
  + GET\_POLYGON: /api/MapData/GetPolygonData?ImportFileDetailId=
  + GET\_LINES: /api/MapData/GetLinesData?ImportFileDetailId=
  + GET\_POINTS: /api/MapData/GetPointsData?ImportFileDetailId=

These API calls will return all the relevant data associated with the selected file.

##### **Add Preferences:**

* The API call */api/AIXMFile/AddUserPreference* of port 6002 is triggered to Add preference. The payload sent with the request contains the file name entered by the user.

##### **My preferences:**

* To load the preferences of a specific user, we use the userId in the API endpoint */api/AIXMFile/GetUserPreferencesByUserId?userId=* on port 6002.
* To delete a preference, we use */api/AIXMFile/DeleteUserPreference?id=*, which requires the preference ID to be specified for deletion.

##### **Export button:**

The export functionality supports three different APIs, all of which require the same payload structure. The following fields must be included in the payload to generate the desired output:

**Required Payload Fields:**

* **shape** – Defines the geometric shape type:
* "P" for Polygon
* "R" for Rectangle
* "C" for Circle
* "NONE" for no shape
* **selectedFeatures** – List of features to be exported.
* Circle Parameters (Required when shape is "C")
* **circleLat**, **circleLng** – Center coordinates.
* **circleRadius** – Radius of the circle.
* Rectangle Parameters (Required when shape is "R")
* **rectLatmin**, **rectLngmin** – Bottom-left (southwest) corner coordinates.
* **rectLatmax**, **rectLngmax** – Top-right (northeast) corner coordinates.
* Polygon Parameters (Required when shape is "P")
* **polygonPath** – List of polygon points in the format:

POLYGON((lat1 lng1, lat2 lng2, lat3 lng3, lat1 lng1))

**API Endpoints:**

* AIXM Export: /ADL\_AIM\_DB/ConvertGeoJsonToExportFile (Port: 6002)
* Spreadsheet Export: /api/MapData/GetGeographicalDataByShape (Port: 6002)
* GeoJSON Export: /api/AIXMFile/GeoJsonDataExportToSpredSheet (Port: 6002)

##### **Data Switch:**

* The initial data call and switching mechanism remain unchanged. We only need to retrieve the ID of the selected value and reuse the existing Polygon, Line, and Point APIs to fetch the corresponding data efficiently.

### **Open Data (/v1/openData) -**

**Access By:** All the authenticated users can access this page.

**Overview:** This page gives open data to the users depending on the category they have selected. This page is used to download the files as well.

#### **UI Layout**

##### **Common Page Header**

* A breadcrumb-style navigation showing the current section.
* "Open Data" serves as a section title to indicate the displayed information with an icon.

##### **Category Selection**

* A dropdown for selecting a category, with options dynamically populated.
* The selected category determines the data displayed in the table.

##### **Data Table**

Displays a list of approved shared files in a table format with columns:

* Sr. No. (serial number)
* File Name (name of the shared file)
* Date & Time (formatted approval date)
* Action (a download icon for file download)

If no data is available, a "No data found" message is shown.

##### **Pagination**

* A pagination control allows navigation between pages.

##### **Error on Fetch Modal**

Displays error messages for:

* Page fetch failure: "Error fetching files for the selected page."
* Shared file fetch failure: "Error fetching Imported Shared Files."
* Download failure: "Download failed. Please try again later."
* File path issues: "There is an issue with the file path. Please try again later."

#### **Open Data Functionality**

The page loads with a dynamically set title and subtitle. It includes a dropdown that fetches categories from an API. Below the dropdown, a table displays data with a title. If no data is available, a "No data found" message appears. Page navigation is positioned at the bottom right of the table.

From the category list, we take *categoryName*, which is displayed as selectable options in the dropdown below the "**Select Category:**" label. The default value is **Select** which remains disabled to indicate the initial state.

When the user selects a category from the dropdown, an API call is made to fetch the list of approved shared files. From the response, *fileName*, *approvedDate*, and *filePath* are extracted and displayed in the table.

The table dynamically populates data as follows:

* **Sr. No.** increments automatically for each row.
* **File Name** is taken directly from *fileName* in the response.
* **Date & Time** is formatted from *approvedDate* into **DD/MM/YYYY, HH:MM AM/PM** format.
* **Action** contains a Download button linked to the corresponding *filePath*.

When the **Download** button is clicked, the system attempts to fetch the file from the provided file path. If successful, the file is downloaded to the user's device using a temporary URL, but if an error occurs or the file path is invalid, an error message is displayed to the user.

A **temporary URL** is required because the fetched file is in the form of a blob (binary large object), which cannot be directly downloaded. Browsers do not allow direct access to fetched binary data as a downloadable file. By creating a temporary URL using window.URL.createObjectURL(blob), we convert the blob into a downloadable link that the browser can recognize. This allows the user to download the file seamlessly. After the download is triggered, the temporary URL is revoked to free up memory.

**Pagination** allows users to navigate through approved shared files efficiently by displaying 10 records per page. Clicking the next or previous button updates the page number accordingly and fetches new data from the API. If an error occurs during this process, an error message is displayed to the user.

#### **API calls and Navigation**

1. **Category list:**

To get the category used in our system we fetch */ADL\_AIM\_DB/GetCategoryList* of port 6002

1. **Category Data:**

The following is the complete API call required:

*/api/AIXMFile/GetApprovedSharedFiles?startRowIndex=${page}&pageSize=${pageSize}&categoryId=${selectedCategoryDetails}*

In this API call, we need to specify startRowIndex, pageSize, and categoryId:

* **startRowIndex**: Initially set to 1 for the first fetch, then increment or decrement as needed.
* **pageSize**: Defines the number of records to fetch starting from startRowIndex. We typically use 10 for optimal performance.
* **categoryId**: GetCategoryList provides categoryId’s, among which we need to provide one.

## **Data Management**

### **AIM DATA (/v1/aimData) -**

**Access By:** Authenticated users with the roles of **Originator** and **Validator**.

**Overview:** This page allows users with the Originator role to upload AIXM files, verify data based on categories, and either reject them or submit them for approval. Users with the Validator role can then review the submitted data and choose to approve or reject it.

#### **UI Layout**

##### **Common Page Header**

* A breadcrumb-style navigation displays the current section with the text "Imports AIM Data" for the Originator role and "Validate AIM Data" for the Validator role.
* "Import Data" serves as a section title for the Originator, while "Imported File List" indicates the displayed information with an icon.

##### **File Upload**

* Only users with the **Originator** role can upload files, and the upload section is positioned on the left side.
* The left side section starts with the label "Select File", followed by a "Choose File" button, a designated space for current file status, which initially displays "No file chosen" when no file is selected, and an "Upload" button next to it.
* A text indicator below these elements specifies that the file should be “AIXM 5.0 or above”.
* When a file is uploaded, its name replaces "No File Chosen", displaying the first 25 characters along with the file extension.
* A loader appears on the Upload button, indicating that the request is being processed.
* If an invalid file is uploaded, an error message appears, and a dialog box display "Invalid file type. Please upload an AIXM file."

##### **AIM File List**

* Depending on the user role, which is Originator or Validator, the position of the file list varies. For an **Originator**, it is displayed on the right-hand side next to the File Upload section, labeled "Uploaded File List" with an icon. For a **Validator**, since file upload is not required, it is displayed on the left next to the label "Imported File List" with an icon.
* A loader appears until the data is fully loaded.
* If the file list contains no data from the API, "No Files Found" is displayed.
* If files are found, each entry is displayed with a serial number, file name, last updated timestamp, and uploader details. If the file name exceeds 55 characters, it is truncated for readability while preserving its extension. For a better user experience, file selection and hover effects are implemented.

##### **Imported Data Summary and Tree Structure**

* The **Imported Data Summary** section, labeled at the start, provides an interactive overview of available data based on the selected file. It dynamically updates to display different feature categories, each accompanied by a count indicating the available data. Users can click on any feature to trigger which is highlighted as well.
* A loader is displayed until the API response is received, ensuring users are informed of ongoing data retrieval.
* The displayed tree format adapts based on the selected category. A key element in this structure is the title / label, which helps users identify the selected item. The data is presented in a hierarchical tree format, designed for clarity and ease of navigation. Clicking on any element expands the relevant details, allowing users to explore the available information efficiently.

The Following is the structure and the details it shows in the tree:

|  |  |  |  |
| --- | --- | --- | --- |
| **Category** | **Title** | **Attributes (First Section)** | **Additional Details (Second Section)** |
| **Aerodrome** | name | gml\_Id, Designator, Type, Latitude, Longitude, ARP Elevation, ARP Elevation UOM | Not available |
| **Airspace** | name | gml\_Id, Designator, Type, Latitude, Longitude, Airspace Volume Upper Limit, Airspace Volume Upper Limit UOM, Airspace Volume Upper Limit Reference, Airspace Volume Lower Limit, Airspace Volume Lower Limit UOM, Airspace Volume Lower Limit Reference | Not available |
| |  | | --- | | **Navaid** |  |  | | --- | |  | | name | gml\_Id, Designator, Type, Latitude, Longitude, Elevation, Elevation UOM | Not available |
| **Obstacles** | name | gml\_Id, Type, Latitude, Longitude, Elevation, Elevation Uom | Not available |
| **ATS and other Routes** | Designator Second Letter - designator Number | Route Segment <incremental number> | Latitude, Longitude, Route Formed Title, Lower Limit, Lower Limit UOM, Upper Limit, Upper Limit UOM, Path Type, Length, Width Left, Width Left UOM, Width Right, Width Right UOM |

##### **Button Actions and Confirmation Flow**

* For the Validator, the buttons available are **Approve** and **Reject**, while for the Originator, they are **Import** and **Cancel**.
* Each button performs a specific action in the code. When clicked, a dialog opens with two options, asking users to confirm whether they want to proceed. The options provided are **OK** and **Cancel**. If the user clicks **OK**, another dialog appears displaying a success message for that step.

#### **AIM Data Functionality**

##### **Initial Page Loading**

For both the Originator and Validator roles, when the page loads, we retrieve the Uploaded File List and Imported File List respectively. A loader is initialized while calling these APIs. Based on the response, the UI is updated accordingly. If no files are found, a "No File Found" message is displayed.

##### **File Upload**

When the user clicks on the Choose File button and uploads a file, a loading process is initiated until the API retrieves a response. If the response is invalid, a dialog is displayed rejecting the file due to issues. This also clears the file name from both the UI and the system.

If the upload is successful, another API is called to fetch the updated file list.

##### **AIM File List**

Depending on the user's selection from the file list, we initiate the **Imported Data Summary** section. The styling is also adjusted based on the selected file's ID, allowing users to easily identify which file has been clicked.

##### **Imported Data Summary and Tree Structure**

When a user selects a file from the file list, the **Imported Data Summary** UI is triggered, displaying categories such as Aerodrome, Airspace, Navaid, Obstacles, and ATS & Other Routes. The respective count for each category is retrieved from the "TableCount" in the File List API. In rare cases where the count is unavailable, a default value of "0" is assigned.

When a user clicks on any displayed category, an API call is made to fetch the tree structure data. Based on the response and requirements, values are extracted dynamically. The tree follows a predefined structure, implemented using the *<RichTreeView/>* component, where we only need to provide the structured data for display.

The sample structure we provide to *<RichTreeView/>* is:

*{*

*"id": "Category-Index",*

*"label": "Category Name",*

*"children": [*

*{ "id": "Category-Index-Property1", "label": "Property1: Value1" },*

*{ "id": "Category-Index-Property2", "label": "Property2: Value2" },*

*{ "id": "Category-Index-Property3", "label": "Property3: Value3" },*

*{ "id": "Category-Index-Property4", "label": "Property4: Value4" },*

*{ "id": "Category-Index-Property5", "label": "Property5: Value5" }*

*]*

*}*

This structure represents a hierarchical tree where each category has an ID, a label, and multiple properties dynamically populated from the API response.

##### **Button Actions and Confirmation Flow**

When a user selects a file from the list, action buttons are enabled based on their role:

* **Originator**: Cancel and Import positioned at the bottom center.
* **Validator**: Reject and Approve positioned at the bottom center.

Whenever a user performs an action on the button, a confirmation dialog appears with two options: OK and Cancel. The dialog ensures that the user confirms their decision before proceeding.

* **Import (Originator)**: Opens a dialog with the message "Are you sure you want to import the data?" Clicking OK proceeds with the import process.
* **Cancel (Originator)**: Opens a dialog with the message "Are you sure you want to delete the data?" Clicking OK triggers an API call to remove the data from the list.
* **Approve (Validator)**: Opens a dialog with the message "Are you sure you want to approve the data?" Clicking OK proceeds with the approval process/
* **Reject (Validator)**: Opens a dialog with the message "Are you sure you want to reject the data?" Clicking OK triggers an API call to remove the data from the list.

Each action follows the same confirmation process, ensuring that users explicitly confirm before making changes.

#### **API calls and Navigation**

1. **Upload AIXM File:**

When a user uploads an AIXM file and clicks on the Upload button we call the */ADL\_AIM\_DB/UploadAIXMFileAndHasmemberCount* on port 6002 with form Data. Form data includes the ‘file’ word in a string, the file itself, and its name.

1. **Uploaded File List:**

To get Uploaded File List we call /ADL\_AIM\_DB/GetUploadedFileList of port 6002.

1. **Imported File List:**

To get Imported File List we call /ADL\_AIM\_DB/GetImportededFileList of port 6002.

1. **Import to Approve:**

To approve the imported file, we use /*ADL\_AIM\_DB/ProcessImportToApprove* on port 6002, where we send the imported file ID. The endpoint with the imported file ID becomes: */ADL\_AIM\_DB/ProcessImportToApprove?ImportFileDetailId=${selectedFileId}.*

1. **Change File Status:**

To update the file status, we modify the values of isImported, isRejected, isApproved, and isDeleted. These values, along with the fileId, are sent to */ADL\_AIM\_DB/UpdateUploadedFileStatus* of port 6002. The table below provides insights into how the data changes based on different actions.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **File Action** | **isImported** | **isRejected** | **isApproved** | **isDeleted** | **Role used** | **Action button** |
| **Importing file** | true | false | false | false | Originator | Import |
| **Cancelling file** | true | false | false | true | Originator | Cancel |
| **Approving file** | true | false | true | false | Validator | Approve |
| **Rejecting file** | false | true | false | false | Validator | Reject |

1. **File Feature information:**

To retrieve specific details of the selected category for a particular file, we call the */api/AIXMFile/GetImportedDataDetails* endpoint on port 6002. This request includes the feature name and the import file ID.

The complete endpoint format:

*/api/AIXMFile/GetImportedDataDetails?featureName=${featureName}&importfileDetailId=${selectedFileId}.*

**Note:** To approve a file, we call two APIs: */ADL\_AIM\_DB/ProcessImportToApprove* and */ADL\_AIM\_DB/UpdateUploadedFileStatus*.

### **NOTAM DATA (/v1/notamData) -**

**Access By:** Authenticated users with the roles of **Originator,** **Validator** and **Admin**.

**Overview:** This page allows users with the Originator role to fetch NOTAM data from different NOTAM applications and import it for validation. The Validator reviews and approves the NOTAMs shared by the Originator. Both the Validator and Admin can view all approved NOTAMs. Users can also visualize NOTAMs on a map in both 2D and 3D views before approving or importing them.

#### **UI Layout**

##### **Common Page Header**

* A breadcrumb-style navigation displays the current section with the text "Imports NOTAM Data" for the Originator role and "Validate NOTAM Data" for the Validator role and "Manage NOTAM Data" for the Admin role.
* 'Configured Data List' serves as a section title to indicate the displayed information.

##### **Load NOTAM Button**

* For the Originator role, there is a label "Import NOTAMs (Published):" with a Fetch button that calls the API. A loader is displayed on the button while fetching data.
* For the Validator, there are "Approved List" and "Back to Unapproved List" buttons, which toggle based on the NOTAM data the user wants to retrieve.
* For the Admin, there is no button required to load the data.

##### **View Grid and View Map**

* The **"View Grid"** and **"View Map"** buttons are enabled only when at least one data entry is available after fetching. These buttons allow users to toggle between a table and a map view. Each button includes an icon that visually represents its function, highlighting the current section the user is viewing.

##### **Table and Pagination**

* The table displays Sr. No., Location, NOTAM No., and NOTAM Text across all lists. Additionally, it includes Selection (with checkbox) for Non-Imported and Unapproved Lists, Imported By and Imported Date for the Unapproved List, and Approved By and Approved Date for the Approved List (without a checkbox).
* **Pagination** is positioned at the bottom right of the table and is available only for the Validator and Admin roles to navigate between Unapproved and Approved NOTAMs.

##### **2D and 3D map views**

* When the user clicks on the View Map button, two additional buttons - **2D Map View** and **3D Map View**, appear below the View Grid and View Map buttons. By default, the 2D Map View is displayed. The maps are positioned directly below these buttons.
* The currently active map view button is disabled to prevent redundant calls. Initially, the 2D Map View is selected and disabled, and when switched to 3D Map View, the 2D button becomes active while the 3D button is disabled. The highlighting and hover effects enhance the user experience.
* The View Grid and View Map buttons remain accessible at all times, allowing users to switch between grid and map views as needed.

##### **Button Actions and Confirmation Flow**

* For the **Validator**, the Approve and Reject buttons are available only in the Unapproved List.
* For the **Originator**, in the Unimported List, the available buttons are Import and Cancel.
* Each button triggers a specific action in the code. When clicked, a dialog appears with two options, prompting users to confirm whether they want to proceed. The options provided are OK and Cancel. If the user clicks OK, another dialog appears displaying a success message for that action.
* **Action Buttons (Approve, Reject, Import, Cancel)** are only triggered when a checkbox is selected within the grid. Without selection, no actions will be performed.
* If the **Reject** or **Cancel** button is clicked while a checkbox is selected, it will simply uncheck the item without performing any other action.
* If no checkboxes are selected, the buttons remain **clickable**, but no action will be executed.
* When a checkbox is selected and the **Import** or **Approve** button is clicked, the corresponding API call is triggered.
* If the **Approve** or **Import** button is clicked without selecting an item, a dialog appears with the message: **"Please select items to approve."**

#### **NOTAM Data Functionality**

##### **Initial Page Loading**

For the **Originator**, the UI loads without making an initial API call. Instead, we proceed with the available interface. If data is unavailable or not fetched, the table displays a "No data found" message in the corresponding row.

For the **Validator**, the UI loads with Imported NOTAM data from the start. The data is retrieved immediately, and the tables are populated accordingly. If the table contains data, the "View Grid" and "View Map" options are available. If no data is present, a "No data found" message is displayed.

For the **Admin**, the process is similar to the Validator, but the API fetches the Approved List instead. The admin can view the table initially, but no action buttons are displayed in this section.

##### **Table and Pagination**

The Table and Pagination will come under the View Grid part. Each data depending on the List it is been displayed depending on it we are fetching it from the response of the API. This following section clarifies the lists, specifying their column names and relevant functionalities:

* For **Non-Imported NOTAMs** (Originator role), the table displays the following columns: Sr. No., Location, NOTAM No., NOTAM Text, and Selection, with a checkbox available for each row. Pagination is not applicable as there is no size limit enforced.
* For the **Imported / Unapproved List** (Validator role), the table displays the following columns: Sr. No., Location, NOTAM No., NOTAM Text, Imported By, Imported Date, and Selection, with a checkbox available for each row. The data is displayed in batches of 10, fetching 10 records at a time.
* For the **Approved List** (Validator and Admin roles), the table displays the following columns: Sr. No., Location, NOTAM No., NOTAM Text, Approved By, and Approved Date. No checkbox is available in this list. The data is also displayed in batches of 10, fetching 10 records at a time.

**Key Takeaways:**

* Dates are displayed in DD/MM/YYYY format.
* For Location, the value is extracted from NOTAM Text, specifically the text between ‘A)’ and ‘B)’.

##### **NOTAM Map**

Maps are enabled when user clicks on the View Map, with 2D maps loading by default. NOTAM data retrieved from the API is plotted on the map.

**Key points in Map are**

* **Radius Measurement**: The radius is provided in kilometers and is converted to meters for accurate map scaling.
* **Coordinate Conversion**:
  + Since NOTAM data originates from different applications, coordinates are stored in different formats.
  + Conversion from Degrees and Minutes (DM) to Decimal Degrees (DD) is applied using the formula:

decimalDegrees = degrees + (minutes / 60)

* + South (S) and West (W) coordinates are converted to negative values.
* **Circle Calcuation in 3D Map**:

Since circles are not natively supported in the 3D map, the circular area must be converted into a polygon to accurately represent it. This is achieved by calculating multiple latitude and longitude points along the perimeter of the circle. The latitude is adjusted based on the circle’s radius and the Earth’s curvature, while the longitude is modified accordingly, factoring in the cosine of the latitude to maintain accuracy. These calculated points form a polygon that visually represents the circular area on the 3D map.

**Formulas Used:**  
  
1. latitude=circleLat+((circleRadius/ earthRadius) × (180 / π) ×sin(angle))

2. latitude=circleLng+((circleRadius/ earthRadius) × (180 / π) ×cos(angle)) / cos ((circleLat \* π)/180)

The angle represents the incremental rotation in radians for each point around the circle, ensuring an even distribution of points to form a polygon that approximates a circle on the 3D map. The earthRadius is set to 6371000 meters (Earth’s radius). The angle starts from 0 and increases in equal steps up to 2π radians (360°), covering the full circumference of the circle. This ensures a smooth and evenly distributed polygon representation on the 3D map.

##### **NOTAM Approval Workflow**

The NOTAM approval process begins with the Originator, who selects NOTAMs from the View Grid and submits them for approval. Upon submission, two dialogs appear: a Warning Dialog to confirm the selection and a Confirmation Dialog to display a success message. The submitted NOTAMs then move to the Unapproved NOTAMs List.

The Validator reviews the Unapproved NOTAMs List and selects NOTAMs using checkboxes. They can either approve the NOTAMs, moving them to the Approved List, or reject them, sending them back to the Originator. The same Warning Dialog and Confirmation Dialog appear during approval or rejection to ensure user confirmation. Multiple NOTAMs can be selected and processed at once.

Once approved, the NOTAMs become accessible in the Approved List for both Validators and Admins.

#### **API calls and Navigation**

1. **Fetch button call:**

We first fetch data from */api/AIXMFile/GetListOfExistingNotams* on port 6002. The response is then sent to */api/notam/GetActiveNotams* on port 5001, which is an unauthorized endpoint. The response from this endpoint is used as the NOTAM data for Originator role.

1. **Import Notam:**

To import NOTAMs, we are using */api/AIXMFile/ImportNotams* API endpoints of port 6002.

To ensure the correct data is passed and to prevent issues, the following payload structure is used when importing NOTAMs:  
{

"notamId": 0,

"latitude": item.latitude ? String(item.latitude) : "",

"longitude": item.longitude ? String(item.longitude) : "",

"notamNo": item.notamNo ? item.notamNo : 0,

"notamText": item.notamText !== null && item.notamText !== undefined ? item.notamText : item.notamNo,

"radius": item.radius || 0

}

This structure ensures:

* Latitude and longitude are always strings to maintain consistency.
* NOTAM number defaults to 0 if not provided.
* NOTAM text falls back to the NOTAM number if null or undefined.
* Radius defaults to 0 if missing.

These validations help prevent errors during the import process.

1. **Import and Approve Notam:**

To approve NOTAMs, we are using */api/AIXMFile/ApproveNotams* API endpoints of port 6002. The following payload ensures that the NOTAM is correctly approved within the system.

{

"id": notam.id,

"latitude": notam.latitude,

"longitude": notam.longitude,

"notamNo": notam.notamNo,

"notamText": notam.notamText,

"radius": notam.radius

}

1. **Imported Notams List:**

To retrieve the imported NOTAMs, we call */api/AIXMFile/GetImportedNotamsForValidator* on port 6002. The API also requires *startRowIndex* and *pageSize* as parameters.

The complete API endpoint is:

*/api/AIXMFile/GetImportedNotamsForValidator?startRowIndex=${startRowIndex}&pageSize=${pageSize}*

In this API call, we need to specify startRowIndex, pageSize:

* **startRowIndex**: Initially set to 1 for the first fetch, then increment or decrement as needed.
* **pageSize**: Defines the number of records to fetch starting from startRowIndex. We typically use 10 for optimal performance.

1. **Approved Notams List:**

To retrieve the approved NOTAMs, we call /api/AIXMFile/GetApprovedNotams on port 6002. This API also requires startRowIndex and pageSize as parameters.

The complete API endpoint is:

/api/AIXMFile/GetApprovedNotams?startRowIndex=${startRowIndex}&pageSize=${pageSize}

In this API call, we need to specify:

* **startRowIndex**: Initially set to 1 for the first fetch, then incremented or decremented as needed.
* **pageSize**: Defines the number of records to fetch starting from startRowIndex. Typically, a value of 10 is used for optimal performance.

### **SHARED FILES (/v1/sharedFiles) -**

**Access By:** Authenticated users with the roles of **Originator,** **Validator** and **Admin**.

**Overview:** This page allows users with the Originator role to upload files categorized by type and import them for validation. The Validator reviews and either approves or rejects the uploaded files. Once approved, both the Validator and Admin can access the uploaded files.

The approved section here directly reflects in the OpenData section, meaning that once data is approved, it becomes accessible to all users in OpenData. This ensures that only validated and authorized files are publicly available.

#### **UI Layout**

##### **Common Page Header**

* A breadcrumb-style navigation displays the current section with the text "Imports Shared File" for the Originator role and "Validate Shared File" for the Validator role and "Manage Shared File" for the Admin role.
* ‘Import Shared File’ for Originator and ‘Shared File Data Details’ for Validator and Admin serves as a section title to indicate the displayed information.
* All three have a label ‘Select Category:’ with a dropdown for selecting a category, with options dynamically populated.

##### **Toggle Button of Unapproved List to Approved List**

* For the **Validator** role, buttons are available to toggle between the Unapproved List and Approved List. These buttons are positioned next to the ‘Shared File Data Details’ label.
* When clicked, the text changes from ‘Approved List’ to ‘Back to Unapproved List’, reflecting the data the user wants to view.

1. **File Upload**

* For the **Originator** role, only the file upload option is available. The left-side section begins with the label "Select File", followed by a "Choose File" button, a designated space for the current file status, which initially displays "No file chosen" when no file is selected, and an "Upload" button next to it.

1. **Table and Pagination**

* The table displays Sr. No., Category, and File Name across all lists. In the **Uploading Files** section (available for the Originator role), it also includes Date, with no pagination. In the **Unapproved List**, additional columns include Imported By, Imported Date & Time, and Selection, where Selection has a checkbox. In the **Approved List** (accessible to Validator and Admin), additional columns include Approved By and Approved Date & Time, without a checkbox.
* Both the **Unapproved List** and **Approved List** have a pagination option at the bottom right of the table.

1. **Button Actions and Confirmation Flow**

* For the **Originator**, the uploaded file section includes **Cancel** and **Import** buttons, which are activated only after the user has successfully uploaded at least one file.
* For the **Validator**, the **Reject** and **Approve** buttons are available only in the **Unapproved List**. These buttons are active only when the user is viewing the **Unapproved List**.
* Additionally, warning and confirmation dialogs are available for both the **Approve** and **Import** buttons. If the user attempts to proceed without selecting any checkbox, a dialog appears with the message: **"Please select items to Approve / Import."**

#### **Shared Files Data Functionality**

##### **Initial Page Loading**

On page load, the API is called to fetch Category items, which are then populated in the dropdown. Additionally, all table headers are initialized.

For the **Originator** role, the File Upload section is displayed.

For both the **Validator** and **Admin** roles, no category is selected initially. As a result, the Unapproved List (for Validator) and the Approved List (for Admin) display the message "No data found" or "No approved files available", respectively. The table shows 0-0 of 0 in the pagination section.

##### **File Upload**

For the **Originator** role, only the File Upload option is available. The user selects a file, and its name is truncated to 25 characters while retaining the file format. Upon clicking Upload, the file is stored in the system.

Before uploading, two validation checks are performed:

If **no category** is selected, a warning dialog appears with the message "Please select a Category to Upload."

If **no file** has been uploaded, a warning dialog displays "Please upload a Valid File."

Once the file is successfully uploaded, the table becomes active, enabling action buttons for further processing.

Users can upload multiple files, but only one at a time. Each uploaded file is added to the table, updating its content with every import.

##### **Dropdown, Table, and Pagination**

For the Originator role, once the file upload is successful, the table displays the following columns: Sr. No., Category, File Name, and Date, where the selected category, file name, and the current date are shown.

For Validator and Admin, the user must select a category from the dropdown. Upon selection, if data is available, it is displayed in a table format. The API call is made based on the section the user is viewing.

* In the **Unapproved List** (Validator), the table includes Sr. No., Category Name, File Name, Imported By, Imported Date & Time, and Selection (checkbox).
* In the **Approved List** (Validator & Admin), the table includes Sr. No., Category Name, File Name, Approved By, Approved Date & Time, and Action (not for Validator).

Both lists support pagination, with 10 entries per page, and data is fetched dynamically based on the current page selection.

**Key Points in table data are:**

* **Date and Time Format**: All date and time entries follow the DD/MM/YYYY, HH:MM AM/PM format.
* **Action Button**: The Delete action button is available only for the Admin role and is represented by a delete icon. Clicking it triggers a confirmation dialog before calling the delete file API upon confirmation.
* **File Preview**: In both the Unapproved List and Approved List, the file name is a clickable link that opens the file in a new tab, allowing users to view it without downloading.

##### **Shared File Workflow**

The Shared Files approval process begins with the Originator, who uploads a file by selecting a category and choosing a file. Upon upload we will get two buttons Import and Cancel. On click of Import button we have Warning Dialog and Confirmation Dialog appear to ensure user confirmation. The Cancel button will remove the file data from the system as well as table is also removed as no data will be present. If the user has Imported, it then API call will be used to add it to the Imported / Unapproved list.

The Validator reviews the Unapproved List and selects files using checkboxes. They can either approve the files, moving them to the Approved List, or reject them, removing them from the validation process. The same Warning Dialog and Confirmation Dialog appear during approval or rejection to ensure user confirmation. Multiple files can be selected and processed at once.

Once approved, the files become accessible in the Approved List for both Validators and Admins. Additionally, approved files are also available in the Open Data section, making them visible to all users.

#### **API calls and Navigation**

1. **Category List:**

To get the category used in our system we fetch */ADL\_AIM\_DB/GetCategoryList* of port 6002.

1. **Fetch Imported Shared File by category:**

The */api/AIXMFile/GetImportedSharedFilesForValidator* API on port 6002 is used to fetch imported shared files based on a selected category. It requires three parameters: categoryId, startRowIndex, and pageSize. The complete API endpoint follows the format:

*/api/AIXMFile/GetImportedSharedFilesForValidator?categoryId=${categoryId}&startRowIndex=${startRowIndex}&pageSize=${pageSize}*

* **categoryId**: Identifies the selected category and is provided from the category list.
* **startRowIndex**: Initially set to 1 for the first fetch and adjusted dynamically as needed.
* **pageSize**: Determines the number of records to retrieve, typically set to 10 for optimal performance.

1. **Fetch Approved Shared File by category and Page:**

The */api/AIXMFile/GetApprovedSharedFiles* API on port 6002 is used to fetch imported shared files based on a selected category. It requires three parameters: categoryId, startRowIndex, and pageSize. The complete API endpoint follows the format:

*/api/AIXMFile/GetApprovedSharedFiles?categoryId=${categoryId}&startRowIndex=${startRowIndex}&pageSize=${pageSize}*

* **categoryId**: Identifies the selected category and is provided from the category list.
* **startRowIndex**: Initially set to 1 for the first fetch and adjusted dynamically as needed.
* **pageSize**: Determines the number of records to retrieve, typically set to 10 for optimal performance.

1. **Import Shared File:**

The Import Shared File API */api/AIXMFile/ImportSharedFile* on port 6002 requires a FormData payload containing the file, category ID, and category name. The file is appended as 'File', the category ID as 'CategoryId', and the category name as 'CategoryName'.

1. **Approve Shared File:**

The */api/AIXMFile/ApproveImportedSharedFiles* API on port 6002 is used to approve imported shared files. It requires a JSON payload containing a list of selected files, where each file includes the following parameters:

* **id**: The unique identifier of the file.
* **categoryId**: The category ID associated with the file.
* **categoryName**: The name of the category.
* **fileName**: The name of the file being approved.

1. **Delete Approved Shared File:**

The */api/AIXMFile/DeleteApprovedSharedFile* API on port 6002 requires the unique ID of the file to be deleted. The request includes a params object containing the file's ID, which is passed in the delete request to remove the specified file.

## **User Management**

### **External User (/v1/externalUserManagement) -**

**Access By:** Authenticated user of role **Admin**.

**Overview:** This page allows Admin users to monitor and manage external user access to the project. The Admin can view users, change their status, and edit or delete their information.

#### **UI Layout**

##### **Common Page Header**

* A breadcrumb-style navigation displays the current section with the text “External User Management”.
* ‘External User List’ serves as a section title to indicate the displayed information.

##### **Table & Pagination**

* The table consists of six columns: First Name, Last Name, Username, Email, User Status, and Action.
* The Action column includes only one icon, displayed as an edit icon.
* The User Status column features a dropdown menu that allows users to update the status.
* The pagination option is available at the bottom right of the table.

##### **Alerts**

* API calls trigger alerts at the top-right of the screen.
* Success messages appear in green, while errors appear in red.
* Alerts automatically close after a few seconds or can be manually dismissed.

##### **Edit option**

* The Edit User form opens directly at the center of the screen, allowing admins to modify user details efficiently.
* It includes labeled text fields for First Name, Last Name, Email Address, Country Code, Mobile Number, Short Name, Display Name, and Login Username, all pre-filled with existing data and editable.
* At the bottom, there are two action buttons: Update Record to save changes and Close to exit without making modifications.

#### **External User Functionality**

##### **Initial Page Loading**

On page load, an API call is made to fetch all external users. The retrieved data is then used to populate the table, and pagination is set accordingly to manage the user list efficiently.

##### **Change Status**

The User Status column includes a dropdown menu that reflects the status of each user. The available options are Pending\_For\_Approval, Approved, Suspended, and Rejected. When a user selects a new status, an API call is triggered to update the status accordingly.

##### **Edit user information**

All required fields in the Edit User form must be properly filled before submitting. If any field is missing, a red alert appears at the top right, displaying the error message from the API response. Once all fields are correctly entered and the update is successful the form automatically closes.

#### **API calls and Navigation**

1. **User List:**

The Get Users List API */api/Users/GetUsersList* on port 6002 is used to fetch a paginated list of users. Since pagination is required, the API is structured as follows:

*/api/Users/GetUsersList?userRoleId=${roleId}&startRowIndex=${pageIndex}&pageSize=${pageSize}*

* **userRoleId**: Currently, it is passed as a string "1", which represents external users.
* **startRowIndex**: Defines the starting index for fetching users. Initially set to 1 for the first page and dynamically updated based on pagination.
* **pageSize**: Determines the number of records retrieved per request. It is typically set to 10 to balance performance and usability.

1. **Update User Status:**

The Update User Status API */api/Users/UpdateUserStatus* on port 6002 is used to modify the status of a specific user. The payload sent to this API contains the following key details:

* **userId**: The unique identifier of the user whose status needs to be updated.
* **userStatusId**: The corresponding ID of the selected status from the dropdown.
* **statusName**: The name of the new status being assigned to the user which is among the Pending\_For\_Approval, Approved, Suspended, and Rejected.
* **emailAddress**: The user's registered email address, ensuring the update is associated with the correct user.

1. **Update User information:**

The */api/Users/UpdateUser* endpoint on port 6002 is used to update user information. This API requires a JSON payload containing essential user details. The payload structure includes the following fields:  
*{*

*userId: '',*

*firstName: '',*

*lastName: '',*

*emailAddress: '',*

*countryCodeMobile: '',*

*mobileNumber: '',*

*shortName: '',*

*loginUserName: '',*

*displayName: '',*

*password: '',*

*}*

Each field must be correctly populated to ensure a successful update.

1. **Internal User (/v1/ internalUserManagement) -**

**Access By:** Authenticated user of role **Admin**.

**Overview:** This page allows Admin users to monitor and manage internal user access to the project. The Admin can view users, change their status, and edit or delete their information. Admins have the capability to add new internal users directly within the system.

#### **UI Layout**

##### **Common Page Header**

* A breadcrumb-style navigation displays the current section with the text “Internal User Management”.
* ‘Internal User List’ serves as a section title to indicate the displayed information.

##### **Table & Pagination**

* The table consists of six columns: First Name, Last Name, Username, Email, User Status, and Action.
* The Action column includes only one icon, displayed as an edit icon.
* The User Status column features a dropdown menu that allows users to update the status.
* The pagination option is available at the bottom right of the table.

##### **Alerts**

* API calls trigger alerts at the top-right of the screen.
* Success messages appear in green, while errors appear in red.
* Alerts automatically close after a few seconds or can be manually dismissed.

##### **Edit option**

The Edit User form opens directly at the center of the screen, allowing admins to modify user details efficiently. It includes labeled text fields for First Name, Last Name, Email Address, Country Code, Mobile Number, Short Name, Display Name, and Login Username, all pre-filled with existing data and editable. At the bottom, there are two action buttons: Update Record to save changes and Close to exit without making modifications.

##### **Create User**

The dialog is titled "Create a New User" and includes two input fields: "Select Role" with options Originator and Validator, and "Enter Email-id" for user input. A Submit button is placed below to the email field, while a Close is positioned at the bottom right.

#### **Internal User Functionality**

##### **Initial Page Loading**

On page load, an API call is made to fetch all internal users. The retrieved data is then used to populate the table, and pagination is set accordingly to manage the user list efficiently.

##### **Change Status**

The User Status column includes a dropdown menu that reflects the status of each user. The available options are Pending\_For\_Approval, Approved, Suspended, and Rejected. When a user selects a new status, an API call is triggered to update the status accordingly.

##### **Edit user information**

All required fields in the Edit User form must be properly filled before submitting. If any field is missing, a red alert appears at the top right, displaying the error message from the API response. Once all fields are correctly entered and the update is successful the form automatically closes.

##### **Create User**

Clicking the Create User button opens a dialog titled "Create a New User." If the user attempts to submit without selecting a role or entering an email, an error message "Please fill in all fields" appears at the top right. The Role dropdown provides two options: Validator and Originator. After selecting a role and entering a valid email, clicking Submit triggers the API call, displaying an alert with the message "Invite Sent Successfully."

#### **API calls and Navigation**

1. **User List:**

The Get Users List API */api/Users/GetUsersList* on port 6002 is used to fetch a paginated list of users. Since pagination is required, the API is structured as follows:

*/api/Users/GetUsersList?userRoleId=${roleId}&startRowIndex=${pageIndex}&pageSize=${pageSize}*

* **userRoleId**: Currently, it is passed as a string "2,3,4", which represents internal users.
* **startRowIndex**: Defines the starting index for fetching users. Initially set to 1 for the first page and dynamically updated based on pagination.
* **pageSize**: Determines the number of records retrieved per request. It is typically set to 10 to balance performance and usability.

1. **Update User Status:**

The Update User Status API */api/Users/UpdateUserStatus* on port 6002 is used to modify the status of a specific user. The payload sent to this API contains the following key details:

* **userId**: The unique identifier of the user whose status needs to be updated.
* **userStatusId**: The corresponding ID of the selected status from the dropdown.
* **statusName**: The name of the new status being assigned to the user which is among the Pending\_For\_Approval, Approved, Suspended, and Rejected.
* **emailAddress**: The user's registered email address, ensuring the update is associated with the correct user.

1. **Update User information:**

The */api/Users/UpdateUser* endpoint on port 6002 is used to update user information. This API requires a JSON payload containing essential user details. The payload structure includes the following fields:  
*{*

*userId,*

*firstName,*

*lastName,*

*emailAddress,*

*countryCodeMobile,*

*mobileNumber,*

*shortName,*

*loginUserName,*

*displayName,*

*password,*

*}*

Each field must be correctly populated to ensure a successful update.

1. **Create User:**

The */api/Users/SendInviteLink* endpoint on port 6002 is used to invite the new user to it. This API requires a JSON payload containing essential user details. The payload structure includes the following fields:

*{*

*emailAddress,*

*userRoleId,*

*userRoleName,*

*}*

Each field must be correctly populated to ensure a successful update.