**Smart Attendance and Access Management System (SAAMS)**

**Channel-View Component Documentation**

Revision History:

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Prepared / Edited By** | **Changes** |
| 0.1 | 03/03/2025 | Vasanth | Initial Draft |

**Introduction** :

**Purpose**: This document describes the Channel-View React component, which provides a user interface for viewing, adding, updating, and deleting communication channels (TCP/IP or Serial) in an Electron-based application.

**Scope**: Covers the component’s functionality, UI elements, data structure, and integration with Electron APIs. Does not cover backend implementation or the Channel model definition.

**Overview :**

The Channel-View component renders a data grid displaying channel details and includes buttons for refreshing, adding, modifying, and deleting channels. Modals handle input for adding and updating channels, with validation for IP addresses and ports.

These buttons are part of the initial Stack layout and trigger the core functionality of the view.

1.**Refresh Button:**

* **Label**: "Refresh" (with a <Refresh /> icon)
* **Use**: Reloads the list of channels displayed in the data grid.

2.**Add Button:**

* Label: "Add" (with an <Add /> icon)
* Use: Opens a modal to create a new channel.

After clicking on Add button it display a new pop up which consists of

different text fields, dropdowns, etc..

2.1Channel Name:

* Label: "Channel Name"
* Type: Text

2.2 Channel Code:

* Label: "Channel Code"
* Type: Text

2.3 Channel Type:

* Label: "Channel Type"
* Type: Select (dropdown)

2.4 Channel Fields:

* IP Address:
  1. Label: "IP Address"
  2. Type: Text
* Port:
  1. Label: ”Port”
  2. Type: Text
* COM Port:
  1. Label: “COM port”
  2. Type: Text
* Baud Rate:
  1. Label: “Baud Rate”
  2. Type : text
  3. Validation: Must be numeric.
  4. LTS Checkbox:
* Label: "LTS"
* Type: Checkbox
* Default: Unchecked (false).
  1. Cancel:
* Use: Closes the modal without saving.
  1. Add:
* Use: Submits the form to create a new channel.

**3.Modify Button:**

* Label: "Modify"
* Use: Opens a modal to edit a single selected channel.

**4. Delete Button:**

* Label: "Delete"
* Use: Deletes one or more selected channels from the list.

**Use Cases:**

**1. Viewing Existing Channels**

**Description**: Users can view a list of all configured channels in a tabular format.

**Preconditions**:

* The application is running.
* Channels have been previously created and stored in the backend system accessible via the API.

**Steps:**

1. The user opens the application.
2. On initial load, the application automatically fetches all channels by invoking window.electronAPI.getChannels() (via handleRefreshButtonClick).
3. The channels are displayed in a DataGrid with columns: ID, Name, Code, Type, Value, LTS, Created At, Updated At, IP Address, Port, COM Port, and Baud Rate.
4. The user can scroll through the list or use pagination (fixed at 5 rows per page).

**Postconditions**: The user sees an up-to-date list of channels.  
**Exceptions**:

* If the API call fails, an error modal appears with the message "Error fetching data."

**2. Refreshing the Channel List**

**Description**: Users can manually refresh the channel list to ensure it reflects the latest data from the backend.

**Preconditions**: The application is running.

**Steps**:

1. The user clicks the "Refresh" button.
2. The application calls handleRefreshButtonClick, which invokes window.electronAPI.getChannels().
3. The DataGrid is updated with the latest channel data. **Postconditions**: The channel list reflects the current state of the backend.  
   **Exceptions**:

* If the API fails, an error modal displays "Error fetching data."

**3. Adding a New Channel**

**Description**: Users can add a new channel with specific configuration details.

**Preconditions**: The application is running.

**Steps**:

1. The user clicks the "Add" button.
2. A dialog modal opens with fields for:
   * Channel Name (text)
   * Channel Code (text)
   * Channel Type (dropdown: TCP/IP or Serial)
   * Conditional fields based on type:
     + TCP/IP: IP Address (text) and Port (text)
     + Serial: COM Port (text) and Baud Rate (text)
   * LTS (checkbox, default unchecked)
3. The user fills in the required fields and selects the channel type.
4. The user clicks "Add."
5. The form validates:
   * For TCP/IP: IP address must match a valid format (e.g., 192.168.1.1), and port must be numeric.
   * For Serial: Baud rate must be numeric.
6. If valid, the data is sent to window.electronAPI.createChannel() with a constructed Channel object.
7. A success or error modal appears based on the API response.
8. The channel list refreshes automatically.

**Postconditions**: A new channel is added to the backend and reflected in the UI.  
**Exceptions**:

* Invalid IP address or port/baud rate triggers an error message in the form.
* API failure shows an error modal.

**4. Updating an Existing Channel**

**Description**: Users can modify the details of an existing channel.

**Preconditions**:

* The application is running.
* At least one channel exists and is selected.

**Steps:**

1. The user selects exactly one channel from the DataGrid using the checkbox.
2. The user clicks the "Modify" button.
3. A dialog modal opens, pre-filled with the selected channel’s details (name, code, type, connection details, LTS).
4. The user edits the desired fields.
5. The user clicks "Update."
6. The form validates the input (same rules as adding a channel).
7. If valid, the updated Channel object is sent to window.electronAPI.updateChannel().
8. A success or error modal appears, and the channel list refreshes.

**Postconditions**: The selected channel is updated in the backend and UI.  
**Exceptions**:

* Selecting multiple channels or no channels triggers a modal with "Select only one item to edit" or "Select an item to edit."
* Validation or API errors are displayed in modals.

**5. Deleting a Channel**

**Description**: Users can remove one or more existing channels.  
  
**Preconditions**:

* The application is running.
* At least one channel exists and is selected.

**Steps:**

1. The user selects one or more channels from the DataGrid using checkboxes.
2. The user clicks the "Delete" button.
3. For each selected channel, the application calls window.electronAPI.deleteChannel() with the channel’s ID.
4. Upon completion, the channel list refreshes.
5. A success or error modal appears if any deletion fails.

**Postconditions**: The selected channels are removed from the backend and UI.

**Exceptions**:

* If no channels are selected, a modal displays "Select an item to delete."
* If any deletion fails, an error modal shows the failed ID.