Suraj SN

Design Document

Corporate Action Data Ingestion

Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Description** | **Author** | **Status** |
| v.01 | 02-Feb-25 | First Draft Version | Suraj SN |  |
|  |  |  |  |  |

Contents

[1. Introduction: 2](#_Toc189769107)

[2. Purpose 2](#_Toc189769108)

[2.1 Scope 2](#_Toc189769109)

[3. Front End - Corporate Actions Dashboard 2](#_Toc189769110)

[Table Columns 3](#_Toc189769111)

[Filtering Mechanism 3](#_Toc189769112)

[User Flow 3](#_Toc189769113)

[3.1 Screen 2 - Historical Corporate Actions 4](#_Toc189769114)

[Overview 4](#_Toc189769115)

[Logic and Workflow 4](#_Toc189769116)

[User Flow 5](#_Toc189769117)

[3.2 Summary of Features 5](#_Toc189769118)

[4. Corporate Action Data Flow and Database Schema 6](#_Toc189769119)

[4.1 Table Names 6](#_Toc189769120)

[4.2 Data Flow Process 6](#_Toc189769121)

[4.3 Database Schema 7](#_Toc189769122)

[4.3.1 PostgreSQL Database Credentials 7](#_Toc189769123)

[4.3.2 Table Names 7](#_Toc189769124)

[4.3.3 Table Schemas 7](#_Toc189769125)

[4.3.4 API Endpoints 8](#_Toc189769126)

[Corporate\_Actions Table 8](#_Toc189769127)

[Scheduler\_Logs Table 8](#_Toc189769128)

[Action\_History Table 9](#_Toc189769129)

# Introduction:

Ingest corporate action data from multiple sources such as regulatory filings, exchange announcements, emails, and news feeds. Process unstructured and semi-structured formats to extract key corporate action details.

# Purpose

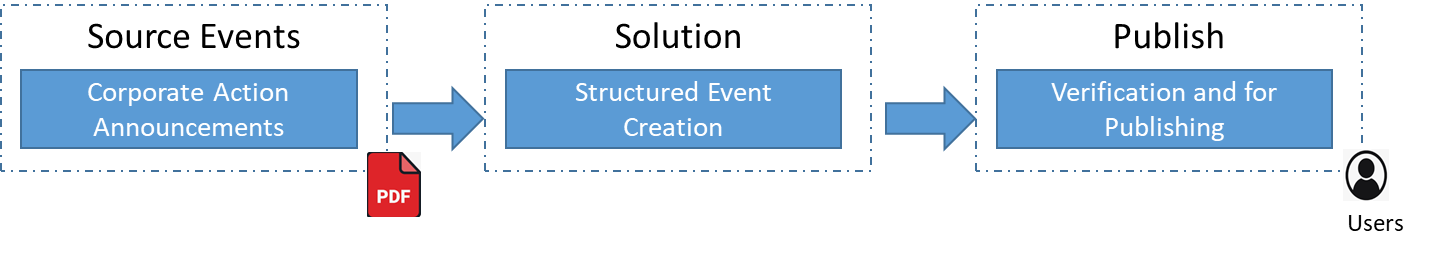
The purpose of this design specification is to describe solution in terms of its structure, implementation, and deployment. The intended core audience for this document is:

* Technical teams including Solution Architect
* Technical Project Management

It is intended to capture and convey significant architectural decisions required to commission the system.

## Scope

The scope of the proposed design covers the following key business requirement



## Front End - Corporate Actions Dashboard

#### Overview

Screen 1 serves as the primary interface where users can view and manage corporate actions. The data is dynamically populated from the backend using the get-records-today API, ensuring real-time updates on corporate actions.

#### Logic and Workflow

* **Data Retrieval:** The table fetches corporate action records from the backend via get-records-today API.
* **Dynamic Table Population:** The retrieved data is displayed with multiple columns, each providing specific details.
* **Popup Functionality:** Clicking the **Details** button opens a popup displaying extracted information in an editable format.
* **Record Modification:** Users can edit extracted information in the popup and submit changes, triggering the update-corporate-action API.
* **Verification Update:** Upon submission, the record's verification status updates from "Unverified" to "Verified".
* **Source Button:** Clicking the **Source** button opens either a webpage (if the source is a URL) or a PDF (if the source is a document in assets/master\_pdf).
* **Filtering Mechanism:** Corporate action buttons dynamically load from the backend, allowing users to filter records based on corporate action types.

### Table Columns

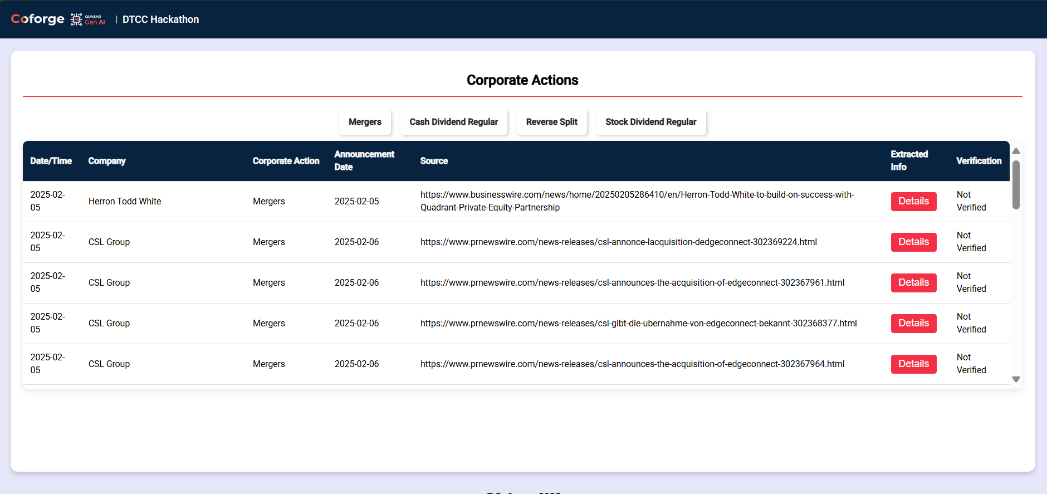
|  |  |
| --- | --- |
| **Column Name** | **Description** |
| **Date** | Date of record entry |
| **Company Name** | Name of the company associated with the action |
| **Corporate Action** | Type of corporate action (e.g., Merger, Forward Split) |
| **Announcement Date** | Official announcement date of the action |
| **Source** | Link to the official source (URL or PDF) |
| **Extracted Info** | Processed data extracted from the source |
| **Verification Status** | Status indicating whether the data has been verified |

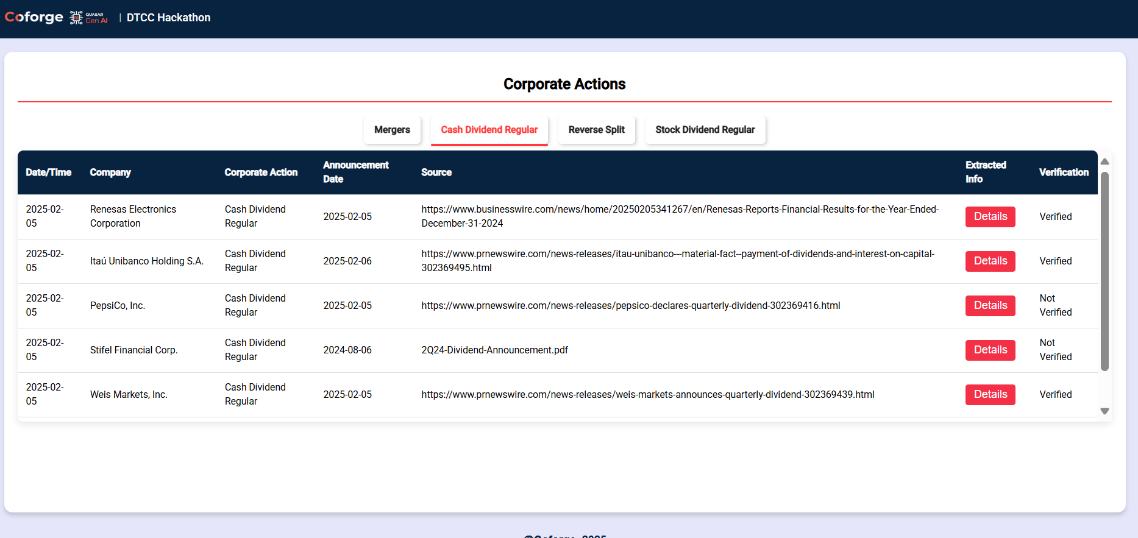
### Filtering Mechanism

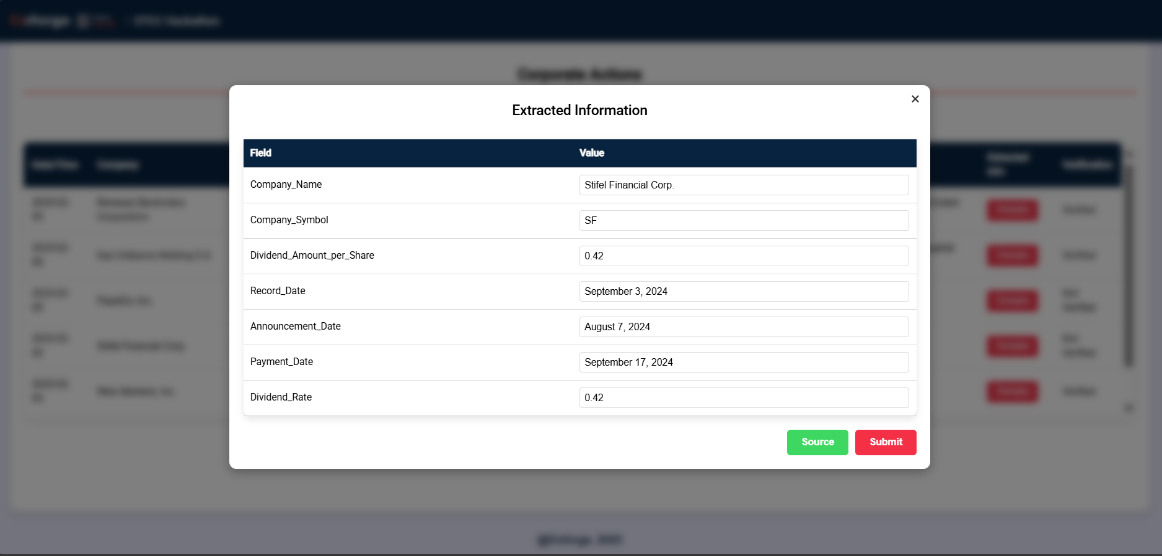
* **Dynamic Corporate Action Buttons:** Action buttons load dynamically to include any new corporate actions.
* **Click to Filter:** Clicking a corporate action button filters records to display only relevant entries.
* **Unselect to Show All:** Clicking the selected action again restores all records.

### User Flow

1. User lands on the dashboard and views corporate actions data.
2. User clicks the **Details** button to review extracted information in a popup.
3. User edits the data if necessary and clicks **Submit**, updating the backend and changing verification status to **Verified**.
4. User clicks the **Source** button to verify the information via a URL or PDF.
5. User applies filters to focus on specific corporate actions.







## Screen 2 - Historical Corporate Actions

### Overview

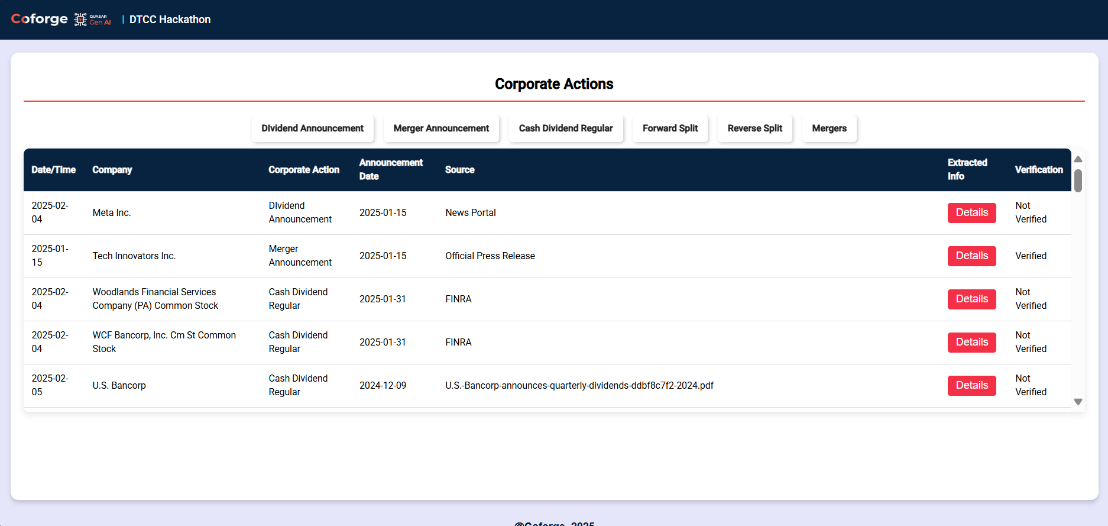
Screen 2 functions similarly to Screen 1 but focuses on historical corporate actions. Instead of displaying today's records, it retrieves past records using the get-records-before-today API.

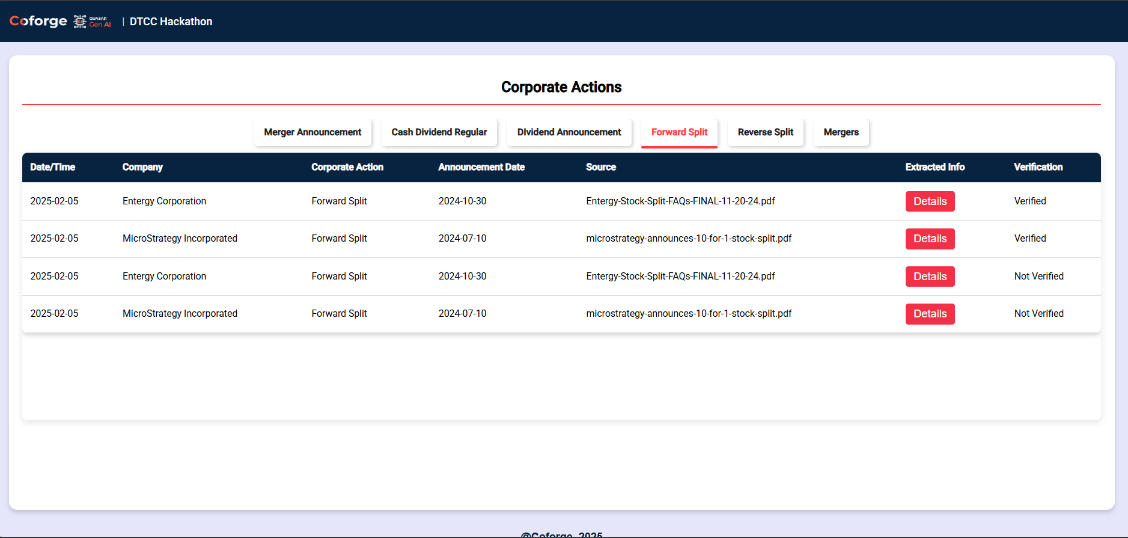
### Logic and Workflow

* **Data Retrieval:** The table fetches historical corporate action records using get-records-before-today API.
* **Table Population:** Data is displayed with the same columns as in Screen 1.
* **Popup for Details:** Users can view and edit extracted information using the **Details** button.
* **Update Process:** Users modify records, trigger an update request, and mark records as **Verified**.
* **Source Button Functionality:** Enables users to view the official source as a URL or a PDF.
* **Filtering Mechanism:** Works the same way as Screen 1, allowing users to filter records by corporate action type.

### User Flow

1. User navigates to the historical records dashboard.
2. User reviews past corporate actions and filters relevant data.
3. User clicks **Details** to review and update extracted information.
4. User verifies the source via URL or PDF.
5. User submits changes, ensuring historical data accuracy.





## Summary of Features

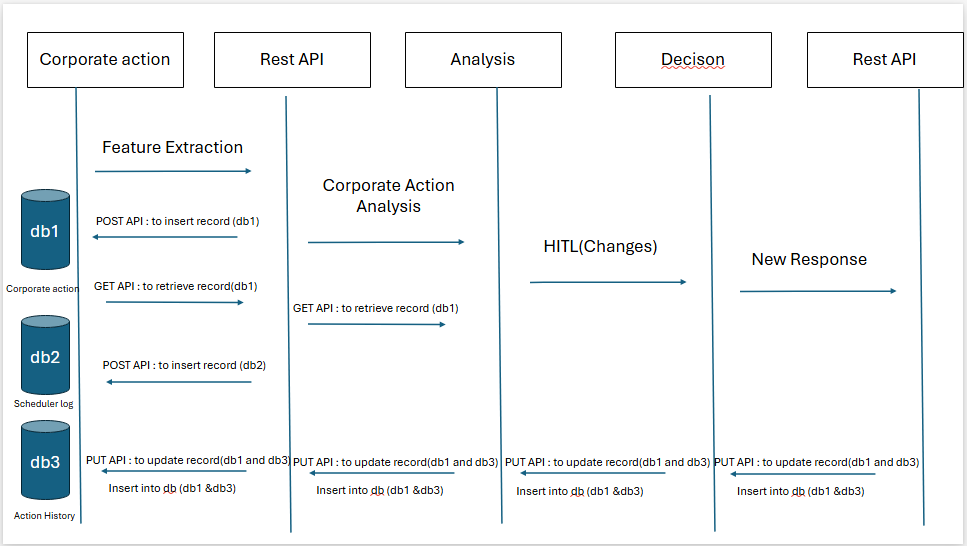
|  |  |
| --- | --- |
| Feature | Description |
| **Dynamic Data Fetching** | Retrieves records from APIs for real-time updates |
| **Interactive Filtering** | Allows filtering by corporate action type |
| **Editable Extracted Info** | Enables users to modify extracted information and submit updates |
| **Verification Status Update** | Automatically marks records as verified after edits |
| **Popup Window** | Displays extracted information in a structured format |
| **Source Validation** | Opens source links or PDFs in a separate window |
| **Seamless API Integration** | Connects with backend endpoints for data management |

This Angular-based corporate actions management system provides a structured, interactive, and user-friendly interface for tracking and verifying corporate actions.

## Corporate Action Data Flow and Database Schema

## Table Names

1. Corporate\_Actions
2. Scheduler\_Logs
3. Action\_History



## Data Flow Process

1. **Receiving Corporate Action**:
   1. When a new corporate action is received, the system extracts feature and sends them to the backend via a REST API to update the **Corporate\_Actions** table in the database (db1).
2. **Sending Metadata**:
   1. The system also sends metadata information related to the corporate action via a REST API to update the **Scheduler\_Logs** table.
3. **Retrieving Records**:
   1. The system retrieves the corresponding records via a REST API (GET API) for analysis and display on the UI.
4. **Context-Aware Analysis**:
   1. A context-aware analysis determines whether the corporate action should be verified or not by making necessary changes.
5. **Human in the Loop (HITL)**:
   1. To verify the analysis, a decision-making process involving human intervention (HITL) is introduced.
6. **Updating Records**:
   1. If changes are made, the system updates the **Corporate\_Action**table and **Action\_History** table via a REST API (PUT API).

## Database Schema

## PostgreSQL Database Credentials

* **Hostname**: localhost
* **Port**: 5432
* **User**: your\_username
* **Password**: your\_password
* **Database Name**: your*db*name

## Table Names

1. Corporate\_Actions
2. Scheduler\_Logs
3. Action\_History

## Table Schemas

#### Corporate\_Actions Table

CREATE TABLE Corporate\_Actions (  
 id SERIAL PRIMARY KEY,  
 Company VARCHAR(255),  
 Corporate\_Action VARCHAR(255),  
 Date\_Announcement TIMESTAMP,  
 Source VARCHAR(255),  
 Extracted\_Information TEXT,  
 Status VARCHAR(50) CHECK (Status IN ('Verified', 'Not Verified')),  
 Insertion\_Date\_Time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 Modified\_Date\_Time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
);  
**Scheduler\_Logs Table**

CREATE TABLE Scheduler\_Logs (  
 id SERIAL PRIMARY KEY,  
 Scheduler\_Name VARCHAR(255),  
 Source VARCHAR(255),  
 Date\_Time\_Scheduler\_Ran TIMESTAMP,  
 Total\_Records\_Fetched INT,  
 Time\_Taken INT,  
 Status VARCHAR(50) CHECK (Status IN ('Success', 'Failure'))  
);

**Action\_History Table**

CREATE TABLE Action\_History (  
 id INT,  
 Extracted\_Information\_Before TEXT,  
 Extracted\_Information\_After TEXT,  
 Modified\_Date\_Time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
);

## API Endpoints

### Corporate\_Actions Table

**POST**: <http://127.0.0.1:8000/insert-corporate-action/>

* + To enter data into the Corporate\_Actions table.
  + Example Request Body:

{  
 "Company": "Tech Innovators Inc.",  
 "Corporate\_Action": "Merger Announcement",  
 "Date\_Announcement": "2025-01-15T10:30:00Z",  
 "Source": "Official Press Release",  
 "Extracted\_Information": "Tech Innovators Inc. announced a merger with Future Solutions Ltd. to expand their market reach and product offerings.",  
 "Insertion\_Date\_Time": "2025-01-15T10:35:00Z",  
 "Modified\_Date\_Time": "2025-01-15T10:35:00Z"  
}  
**PUT**: <http://127.0.0.1:8000/update-corporate-action/>

* + To update records in the Corporate\_Actions table based on the primary key.
  + Example Request Body:

{  
 "id": 4,  
 "Extracted\_Information": "this is updated"  
}  
**GET**: <http://127.0.0.1:8000/get-records-today>

* + To select records where Insertion*Date*Time is today.

**GET**: <http://127.0.0.1:8000/get-records-before-today/>

* + To select records where Insertion*Date*Time is before today.

### Scheduler\_Logs Table

**POST**: <http://127.0.0.1:8000/insert-scheduler-log/>

* + To insert new values into the Scheduler\_Logs table.
  + Example Request Body:

{  
 "Scheduler\_Name": "Daily Data Fetcher",  
 "Source": "Internal API",  
 "Date\_Time\_Scheduler\_Ran": "2025-02-04T07:30:00Z",  
 "Total\_Records\_Fetched": 1500,  
 "Time\_Taken": 120,  
 "Status": "Success"  
}

**GET**: <http://127.0.0.1:8000/get-scheduler-logs/>

* + To select all records from the Scheduler\_Logs table.

### Action\_History Table

* **Note**: This table will be populated when the Corporate\_Actions table is updated. No need to insert data manually.
* **GET**: <http://127.0.0.1:8000/get-action-history/>
  + To select all records from the Action\_History table.