# Technologies Stack

The technology stack for a SaaS platform that provides a drag-and-drop interface for users would depend on a number of factors, such as the specific requirements of the platform, the desired performance and scalability, and the development team's preferences and expertise. However, a common tech stack for such a system might include the following components:

* Frontend client: A JavaScript library or framework **Angular**, along with HTML and CSS for the user interface.
* Backend server: **Django** web framework.

Database:

* Influx time series DB (Fastest solution, runs on any cloud)

Depends on what kind of operations we are going to execute more (CRUD).

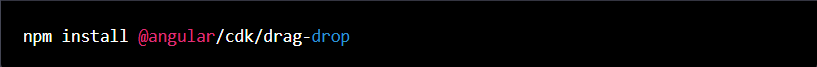
* Web API: A framework such **as Django REST framework** to handle the communication between the client and server.
* Machine learning pipeline: A framework such as **TensorFlow** or **PyTorch** for implementing the machine learning models, along with any necessary data preprocessing and postprocessing tools.

# Front-end (Drag&Drop UI) detailed development with **Angular**

To build a drag and drop interface with Angular, you can use the **@angular/cdk/drag-drop** module, which provides you with a way to easily and declaratively create drag-and-drop interfaces, with support for free dragging, sorting within a list, transferring items between lists, animations, touch devices, custom drag handles, and previews.

Here's a basic example of how you can use this module to create a drag and drop list:

1. First, you'll need to install the `@angular/cdk/drag-drop` module:



1. In your Angular component, you can then import the `DragDropModule` from the @angular/cdk/drag-drop module and add it to the imports array of your module:

Text

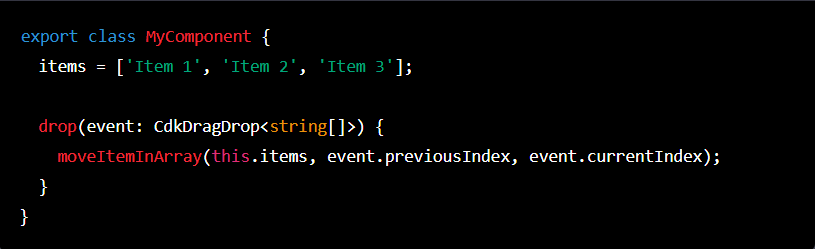
Description automatically generated

1. In your template, you can then use the `cdkDropList` directive to define a container for a list of draggable items, and the `cdkDrag` directive to make the items themselves draggable:

Text

Description automatically generated

1. In your component class, you can define the `items` array and the `drop()` method to handle the drop event:



<https://material.angular.io/cdk/drag-drop/overview>

## Create a linear connection (arrow) between the DnD containers

To create a linear connection between each drag and drop element, you can use the @angular/cdk/overlay module, which provides a way to create floating panels, connected to an anchor element, that can be used to display additional content on top of an existing element.

Here's an example of how you can use this module to create a linear connection between each drag and drop element:

1. First, you'll need to install the @angular/cdk/overlay module:



1. In your Angular component, you can then import the OverlayModule from the @angular/cdk/overlay module and add it to the imports array of your module:

Graphical user interface, text

Description automatically generated

1. In your template, you can use the cdkConnectedOverlay directive to create a floating panel that is connected to an anchor element. In this case, the anchor element will be each drag and drop element:

Text

Description automatically generated

1. In your component class, you can define the items array and the positions array, which specify the positions of the floating panel relative to the anchor element:



<https://material.angular.io/cdk/overlay/overview>

## Pass data between containers, declare start and end point for the data

1. First, you'll need to set up a drag and drop interface using the @angular/cdk/drag-drop module, as described in my previous answer.
2. To display a line or pointer between the starting and ending points, you can use the @angular/cdk/overlay module to create a floating panel that is connected to the starting point (i.e., the source container) and the ending point (i.e., the target container). You can use the **cdkConnectedOverlay** directive to create the floating panel, and the **cdkConnectedOverlayOrigin** and **cdkConnectedOverlayConnectedTo** inputs to specify the starting and ending points, respectively.
3. In your template, you can use the cdkConnectedOverlay directive to create the floating panel and specify the starting and ending points:

Text

Description automatically generated

1. In your component class, you can define the sourceItems and targetItems arrays, and the drop() method to handle the drop event and pass the data between the containers. You can also define a showOverlay property to control whether the floating panel is displayed:

A screenshot of a computer

Description automatically generated with medium confidence

# Back-end

1. Connect Django with Angular
2. Django with PyTorch or TensorFlow, decide which is better
3. Django with the DB

## Connect Django with Angular details

1. Install Django and create a new Django project using the Django web framework.
2. Create an Angular app using the Angular CLI.
3. Set up a Django backend by creating a Django app and defining models, views, and serializers to handle data storage, retrieval, and formatting.
4. Set up a Django REST framework (DRF) endpoint to expose the backend functionality via a REST API.
5. In the Angular app, use the HttpClient module to make HTTP requests to the DRF endpoint and retrieve or update data as needed.
6. Use Angular components, services, and templates to display and manipulate the data in the frontend.
7. If desired, you can also set up CORS (Cross-Origin Resource Sharing) to allow the Angular app to make requests to the Django backend from a different domain.