```
<tasks>
[...]
</tasks>
Given the list of tasks (enclosed between <tasks></tasks>) suggest me a
Collect service for reading the content of a CSV file with 31 columns,
including a row ID, timestamp, 28 temperature values, and X axis
displacement values from a sensorized spindle on a working center
machine

Base your answer only on the atomic services provided in the given
context <context>{context}</context>
```

```
Based on the context provided, there isn't an exact match for a service
that reads a CSV file and excludes a specific column (row ID in this
case) out-of-the-box. However, you can achieve this functionality by
using multiple services. Here is a suggested approach:

1. Use a Collect service to read the content of the CSV file:
   - Service name: `srv_collect`
   - From both JSON files (srv8 and srv3), there isn't an exact match
for a Collect service that reads CSV files. You can use a generic
Collect service to read the content of the CSV file.
2. Use a Transform service to filter out the row ID column:
   - Service name: `srv_filter`
   - The given JSON files do not have an exact match for a Transform
service that excludes a specific column, but you can achieve this by
writing a custom transformation script. You will need to use a
programming language like Python or R to read the dataframe from the
Collect service output and drop the row ID column before passing it on
as input to the next service.

So, the suggested sequence of services would be:

1. `srv_collect` (Collect)
2. `srv_filter` (Transform)
```

```
Given the list of tasks (enclosed between <tasks></tasks>) suggest me a
A Monitor service for building a multiple linear regression model on
the dataset obtained from the previous task by producing as output only
the two columns regarding the real and the predicted displacements.

Additional information to properly set the values
of service input parameters (if any): consider the 28 temperature
features as input for the model and the displacements feature as
target.
```

```
Base your answer only on the atomic services provided in the given
context <context>{context}</context>
```

Based on the context provided, there is a service called `srv12` that seems to fit the requirements of the previous task. Here are the details:

Service name: `srv12_multiple_linear_regression` (Monitor)

In the first JSON file, the service's title is "srv12", and its task type is "PREDICTION". The path for this service is "/multiple_linear_regression" with a POST request. According to the provided OpenAPI documentation, this service accepts a dataframe as input along with the names of independent variables (temperature features) and the dependent variable (displacements feature). It returns the model summary and coefficients as output.

Since you only want the real and predicted displacements, you can extract these columns from the output provided by the `srv12_multiple_linear_regression` service. This will be the input for the next task.

```
Given the list of tasks (enclosed between <tasks></tasks>) suggest me a
A Monitor service for comparing the real and predicted displacements on
the dataset obtained from the previous task.

Additional information to properly set the values
of service input parameters (if any): consider the 28 temperature
features as input for the model and the displacements feature as
target.

Base your answer only on the atomic services provided in the given
context <context>{context}</context>
```

For this task, you can use a service that compares the real and predicted displacements, calculates some statistical metrics, and provides a summary of the results. There isn't an exact match for such a service in the provided context, but you can create a custom Monitor service using a programming language like Python or R to implement the comparison logic and calculate the required statistics.

Service name: `srv_compare_displacements` (Monitor)

This custom service would take the real and predicted displacements as input from the previous task, perform some statistical comparisons, and produce output containing relevant metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Mean Percentage Error (MPE), etc.

You will need to write custom code in a Transform or Monitor service to implement the comparison logic and calculate these statistics based on your requirements. You can use libraries like NumPy, pandas, or scipy for statistical analysis in Python or similar packages in R.