Blog Post Reference

# Official Data Declaration Syntax

This section is talking about the syntax/declaration I have added to declare specific data types/data entries.

## Data Types vs Data Entry Types

Currently my system accepts certain “types”. It can accept data types like “date” which is a “data type” but it will accept data entry types like “select” which is a data entry type.

This can be seen as confusing as the question “what does the JSON declare, data types or data entry types”. I hope to combat this later on in development. For now, I’ll just be referencing Data Types/Data Entries as “Data Declaration”.

### Currently Supported Data Declaration

Single line text

Date

Select

### Specialized Data Declaration Explanation

In this section, I will be stating what is meant by data declarations like “Select”.

**Select**

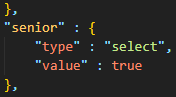
This would be used for “Yes” or “No” fields. For example, for Boolean data fields. You can see an example of data fields that asks the user if they are senior or not.

**Date**

Currently right now, the date picker that I am using requires a time to be entered in. Anyway, this is a self-explanatory data type. Just needed to declare why, it requires a time.

## The Syntax

Below you can see a typical declaration of it inside an Employee JSON.



To declare the String Value Pair should be an Object using curly brackets. Secondly, “Type” should be the first element in the Object, if it isn’t the system won’t recognize it as a “Data Declaration”.

As you can see this is the what the system rendered out and the default value was “Yes” which correlates to the value that was defined above.

## Further Development to be Done

- Allow for more complex data entry types

- Real time verification

- Used for password checks like symbols, length, and capitalization

- More robust declaration

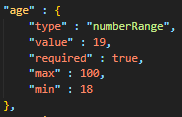
- The “Type” does not have to be declared first

- Case does not matter

# Adding Real Time Validation

This section is talking about how to use the React Hook Form Errors to enforce real time validation. Ill be going through an example of adding the required attribute and max for a number entry. You will be seeing this type of validation being implemented throughout my code.

## JSON Declaration



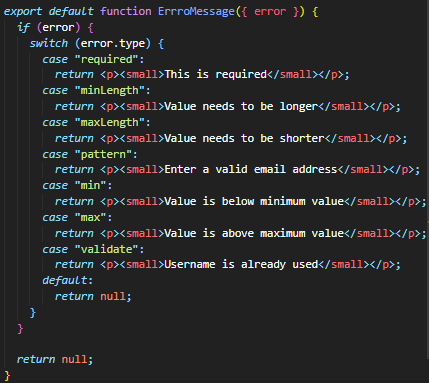
You can see above what I have declared in a JSON file to describe a employee age. The maximum age is 100 and it is a required field meaning the form cannot continue without that field being completed.

## Data Entry Type Declaration

This section talks about how the react rendering component sees this json and renders the corresponding element. Below you can see the code.

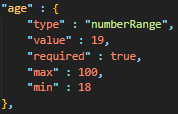
C:\Users\Justin\Documents\University\Semester 2\Computer Science Challenges\Main Code\WebAutomationResearch\main-app\Blog Post\images\Validation Overview.png

The code above if for a number input. The input can enforce max and min values and if the field is required or not. Each time the input catches an error it will send that said error to an ErrorMessage component which renders the corresponding message dependent on the violation. The photo below is a snippet of the ErrorMessage component and how it recognizes what message to display dependent on the error.

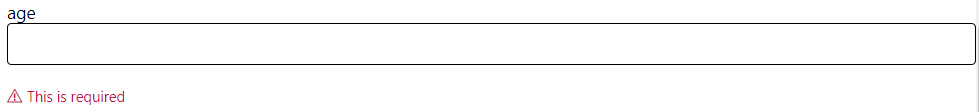


# Example

Below I’ve declared a number input with the constrictions that it’s a required field, has a max value of 100 and the minimum value is 18. This is for an age.

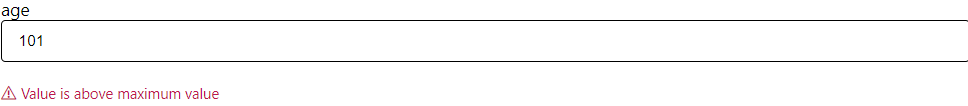


## Error: Required



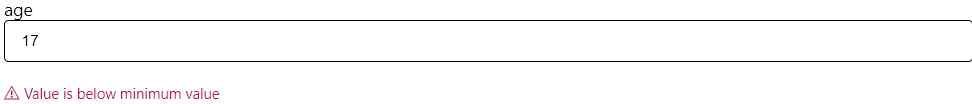
When I press the submit button when the field is empty. This is what appears. When you press submit your cursor will be focused on the field until it’s filled.

## Error: Max



This error happens in real time during your typing. As you can see the max was 100 and I went above that by one.

## Error: Min



This error happens in real time during your typing. The min value was 18.

## Further Development to be Done

- Aim to add more validation types like two field verification

- Maybe add some robot verification

# What Next

In this section, I’ll be talking about what I’ve done and what needs to be done next. This is for those who want to continue my work.

## Work That Has Been Done

- Able to take a JSON and recursively parse through it

- Take data from the parser and render it through React

- Have specialized data types for date, number, entrybox, email, dropdown and select

- Validation for specialized data types, like required, min, max, maxLength, minLength

- An Express Back End that can deliver a JSON file to the React

## RoadMap

Ability to get the data from the form and save the instance inside the system

Ability to upload their own JSON files into the system

Create a MongoDB to store all the JSON files

Start working on a Login Page

Start Working on Permissions for a Single User

Create syntax in the JSON to declare for permissions

Incorporate another user into the system with different permissions

Allow for them to interact with each other, edit, delete and add data

Now start to create database tables for each user. For example for Staff it would be their sales

Allow users to interact with the new tables

Create syntax in the JSON to permissions on certain tables

Start to develop the visual aspect of the application

# The High-Level End Goal of The Project

If you want to continue the project right to the end, you must be able to visualize the finished product. The project must be able to create multiple views for each JSON object that is placed inside the application. For example, would be LinkedIn, you would create Users, Employers and Staff. The user would declare certain permissions or views for each object declared. For example, Staff can delete others posts if inappropriate. Employers can add users if they work for them, etc.

Remember the reason for creating this project was because the most common type of commercial application is CRUD Web applications, and we are trying to help automate this process. This process has some very boiler plate code that is done over and over again. We aim to create a basic CRUD Web Application using just JSONs, whilst having the application to be further developed by the user if needed to be.