Objective

Develop a simple JavaScript Grid component with generic CRUD logic.

Specification

- Component should accept two sets: data format specification and a data set itself.
- Single data format specification has the following fields:
  - Name – property name, i.e. Login, Password, BirthDate, RegistrationDate, Age if User entity should be shown in the grid.
  - DataType – one of the following: string, int, double, date, datetime, time, bool, enum.
  - CalculateFrom – array of property names used for calculable fields
  - Calculate – delegate to a function which calculates value from fields specified in CalculateFrom.
  - IsRequired – specifies if field is required while creating/updating an entity.
- Data item contains properties with their values.
- Input data example:
  var dataFormats =

```
[
    {
        Name: "Login",
        DataType: DataTypes.String,
        IsRequired: true
    },
    {
        Name: "BirthDate"
```

```
            DataType: DataTypes.Date,
            IsRequired: false
        },
        {
            Name: "Age"
            DataType: DataTypes.Int,
            IsRequired: false,
            CalculateFrom: ["BirthDate"],
            Calculate: function (birthDate)
            {
                // Calculates age in years
                (Date.now() - birthDate) / (1000 * 60 * 60 * 24 * 365)
            }
        }
];

var data =
[
        {
            Login: "User",
            BirthDate: new Date("01/01/2000") // Age calculates automatically
        }
```

];

- Data format set should be validated when specified. Validation includes all possible checks. I.e. Name, DataType and IsRequired should be specified; CalculateFrom cannot be present if Calculate is not; Calculate function should have the same amount of parameters as CalculateFrom provides; no additional fields should be present; etc.
- Data set should be validated in similar way. In addition, it should be validated on data format accordance (i.e. data types).
- Validation errors should be logged by using specific logger service instance. Two default loggers should be available: console and alert. Logger should contain at least three message levels: error, warning, info. Validation errors log according to their importance: if component could not continue work, then it uses error level; if could (i.e. there are some strange additional fields that does not affect component work) – warning. Other useful messages shows as info: i.e. debugging information. Logger instance should be provided by using Dependency Injection.
- Another DI component should be AJAX one. User should have possibility to replace real REST API calls with a stub.
- The last column represents actions: edit and delete. Above or under the grid should be “Add” button.
- Edit or add opens a form with fields specified by a data format specification. Input validation for required fields, as well as data type validation should be present.
- Component should support local storage – added items should be shown after page reload. The best way to do that is to use AJAX stub.
- Component should be configurable. Configuration options includes:
    - Data source for format and data themselves. Could be REST service URLs or direct JSON specification.

    - Templates customization. Component should provide default templates for grid, every row, cell counting DataType (every DataType should have its own cell template). Default templates could be overridden.
    - Dependencies configuration (logger and AJAX services).
- Component should be covered by all necessary unit tests.

Technical requirements

AngularJS

TypeScript

Jasmine

Process

This task is especially designed to be undoable (or very hardly doable) within 8 hours. However, result is *not* completed task; result counts as amount of work done within 8 hours. Please do not spend much more time.

Bonuses (will be highly appreciated):

- Delivery in 24 hours after receiving
- Smart questions
- Deployment on test server
- GitHub or any other free service commit

In you delivery, please provide full source codes for review and how much time you actually spent.