The Home Task should be done using Typescript.

**You need to implement:**

“Currency Converter” application to display a certain Euro amount in various other currencies:

1. radio button to change modes:
   1. mode where the Euro amount is the same across all the panels with different currencies, so when changing the Euro amount/Currency amount for one of the panels, the rest should change accordingly;
   2. mode where the Euro amount/Currency amount is independent, so you can change amount for specific panel without affecting the rest of panels.
2. radio button to change views:
   1. view with text input fields (pic. 1);
   2. view with sliders as inputs (pic. 2);
3. panels with (design may be as simple as possible):
   1. currency title;
   2. currency exchange rate input field;
   3. euro amount input field;
   4. converted currency amount input field;
4. possibility to change Euro amount as well as amount of the respective currency. For mode A, the new currency value is used to change the value of Euro, which in turn must change the rest of currencies in all panels.

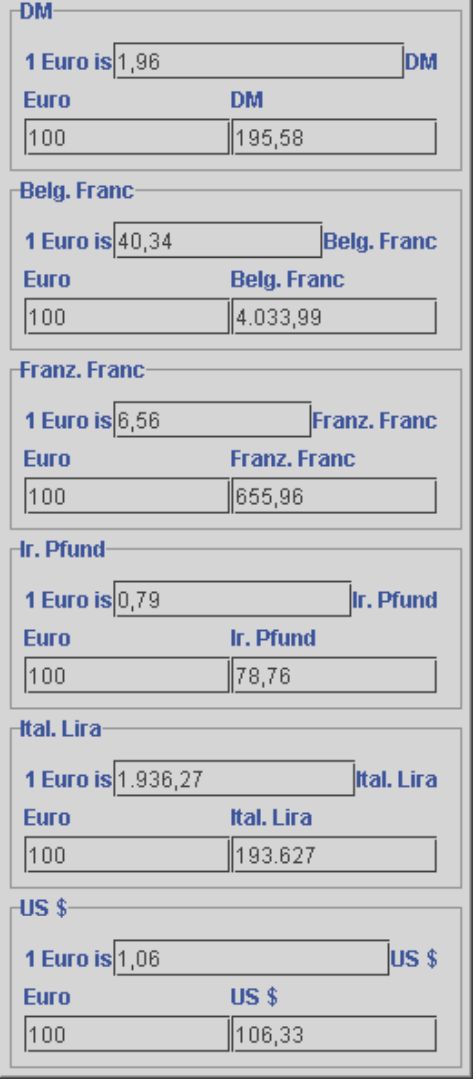
**Note**:

Think about the implementation details for both modes and do not forget to implement 3 components of MVC: Model, View and Controller.

For example, for mode A you can use Singleton to share Euro amount across the panels and Observer to notify all the Currency amounts for all panels when Euro amount (Subject) is changed.

Optionally, you can create each component as abstract class with basic functionality (e.g. Model as Observer or Pub-Sub) and inherit it in your application by adding specific logic.

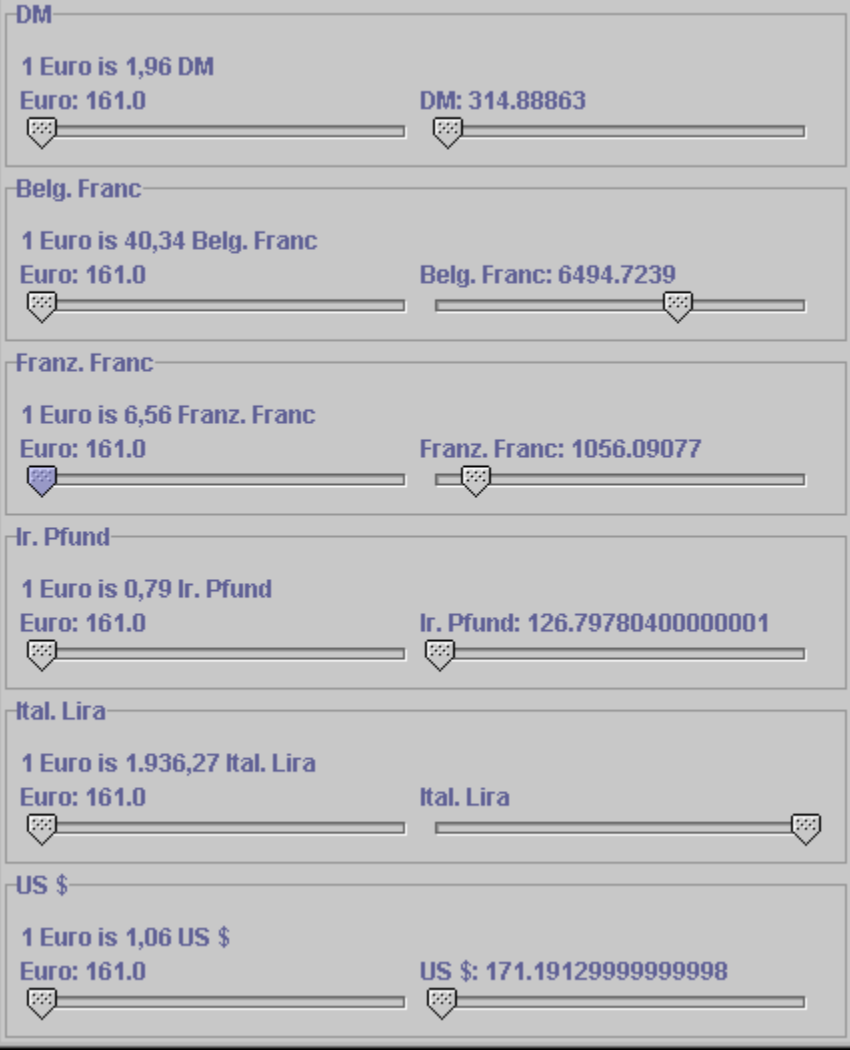
You can find a bit of inspiration in Backbone.JS library.



Pic 1. ExampleofCurrency Converter. You can use any exchange rates and currencies.

You need to fetch exchange rate for all currencies from mocked server (you can create simple server for that or just create *rates.json* file and make ajax requests to fetch data from it).

Use the previous results to design and implement a different appearance of the converter panels: the values should not be entered as text but by using sliders.

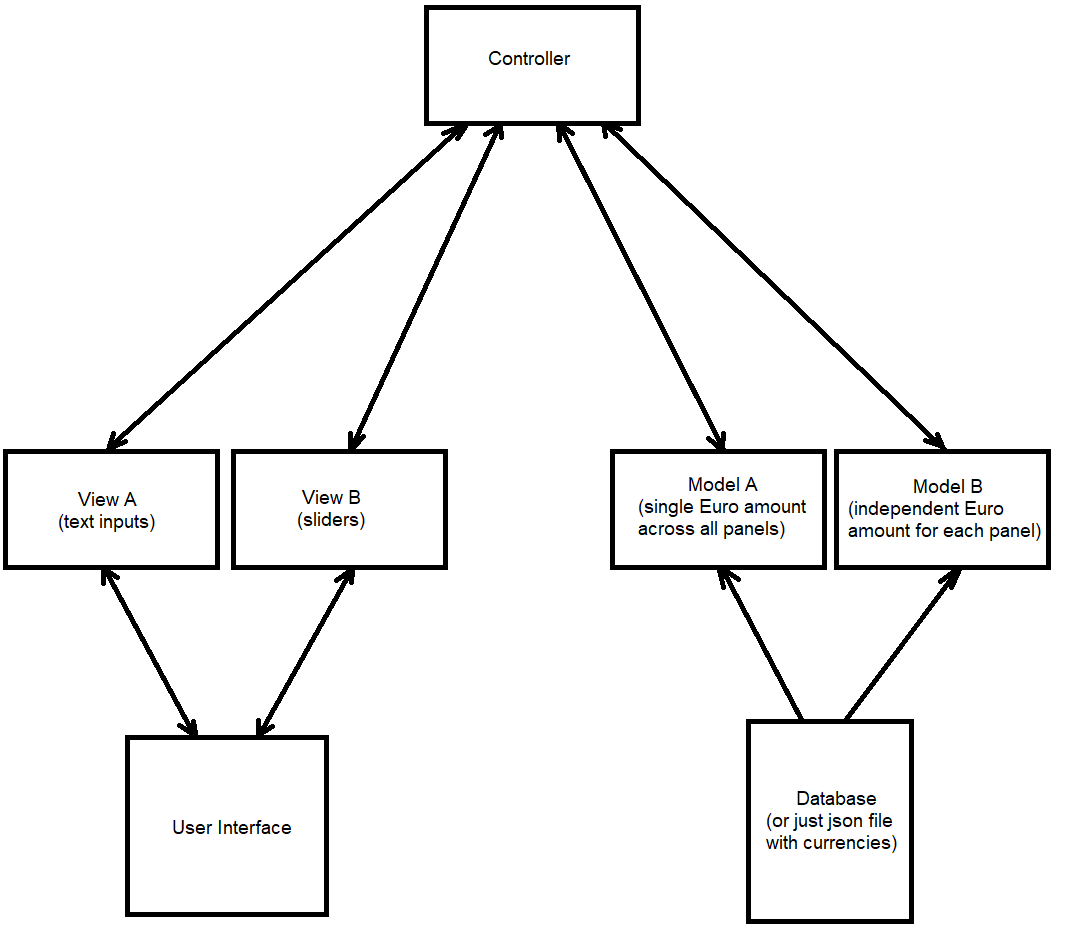
****

Pic 2. Another view for the same application with the same domain logic.

Would be a plus to have ability to use values of type double for view with sliders.

Think about how you want to handle displaying of different view types (view with inputs and view with slider): via the tabs, or via the dialog window at the start of application, so it’s up to you. Dynamic transition between different views (tabs) is optional.

You may use a scheme from a pic. 3 as a reference for your implementation, note that you do not need to strictly follow not the components provided below nor their relations as MVC implementations may vary.



Pic 3. MVC implementation scheme

**Important Notes:**

1. do not use any framework;
2. you can use some template engine to render view(s) (handlebars, jade, etc.);
3. in case you decide to use observer/pub-sub, implement it by yourself;
4. you can use real server to fetch currency rate data or just mock json file.

**Evaluation criteria:**

More than one requirement is not implemented or major MVC violations/MVC is not implemented.

One requirement is missed or some part of MVC is missed.

Minor issue with some of the requirements or minor MVC violation.

All requirements are met, MVC pattern is fully implemented.