Boxinator

Before you start coding, please let us know when you think you are ready to send us the result. This is not a test to see how fast you can solve the task, however it is usually a good idea to have a target date to aim at.

This specification describes a tiny application for calculating shipping costs of boxes to specific locations around the world. The application consists of two simple views, one table and one form.

View A - Form (Add box)

- Form Containing four fields:
    - Name of receiver (input text)
    - Weight (input number) in kilograms
    - Box color (color picker component / text) produces colors in RGB-format i.e. field value becomes (255, 255, 255)
    - Save button the data to the database)
    -
    - Destination country (Dropdown /select list) of the following choices:
        - Sweden (See multipliers below for value)
        - China (See multipliers below for value)
        - Brazil (See multipliers below for value)
        - Australia (See multipliers below for value)

    - Save button the data to the database)

If you try to save the form without filling every field, the user should be presented with some type of error message, indicating what fields are required. Likewise, if you try to enter “negative values” into the weight input field, the user should be presented with an error

describing that negative values are not permitted for weight, it should the default to the value zero - “0”.

Bonus points: it should be impossible to select any type of blue colour in the color picker, either through validation or a limitation in the color picker. This applies to all shades of the color blue.

View B - dispatch list
This view consist of a table that has four columns, receiver, weight, box color and shipping costs. The first column simply displays the name of the receiver. The second one has the weight of the box in kilogram units, the third one has no display value in text, instead it shows the actual color you picked,when creating the box. the final column i the shipping cost of the coury you selected when creating the box from the “Destination” dropdown. The shipping cost will be a calculation of box weight * multiplier, wher multiplier is different for each country (see below).

The multipliers of the countries are the following (pre defined constants):

- Sweden (1.3)
- China (4)
- Brazil (8.6)
- Australia (7.2)

See the following example of how it might look

Receiver	Weight	Box color	Shipping cost
Example box name	22 kilograms		n SEK

Finally below the table a summary of total shipping cost and total weight should be shown, as you add more boxes this is updated automatically together with the above table. this information should be pulled from the same database.

The application should be set up so that when you browse to (http://localhost:8080/#/addbox) you end up in the box form view, and when you change to (http://localhost:8080/#/listboxes) you end up in the table view.

This task should be solved using some form of MVC -style design pattern, where presentation and business logic is somewhat separated, it should be a single-page application using ajax OR websockets when communicating with the backend. The backend should be designed as a REST-api or RPC-style api if using websockets. With at least two endpoints/commands (One for saving boxes, the other for retrieving the list).

The following technologies should be used to complete the task:
● MySQL
● Java
● HTML
● CSS
○ Using “less”
● Javascript
○ React together with Redux (or similar)

Unit / Component Tests
Write unit / component tests by using something like jUnit (Java) and
Jest or Mocha together with something like Enzyme and any assertion library of your choice for testing React components. We value high coverage and clean easy-to-read tests.
If required, please supply a readme.md file if there is any manual setup required to launch your completed project or execute the tests.

Share your solution as a Github project. Please remove the project once it has been been reviewed by us.

Thanks and best of luck //Fortnox.