ReactJS Task For Recruitment

The candidate should create a single page application(SPA) with React and Redux which must implements the following:

1. React & Redux Setup with Webpack without using Create React App.
2. Any framework for styling can be used with the condition that no ready-made React components can be used, only styles.
3. Index page (Table) with the data from `data.json` loaded dynamically
    - The table must be sortable only by status, created_at, merchant_name, type, error_class, card_holder, card_number and amount fields.
    - All dates must be displayed in a specific format: yy-mm-dd HH:MM (Example: 17-07-26 07:11).
    - Transaction types like SaleTransaction, Sale3dTransaction, AuthorizeTransaction must be displayed as Sale, Sale3d, Authorize.
    - error_class must be formatted to show only System, Remote or Unknown instead of the entire class name
    - amount must be visualized as 'amount + currency' (example: 0.50 USD). Note that in the date.json the value for the amount is 50. You must convert that value properly before visualized.
4. The `data.json` file, must be wrapped in a class that provides data access methods. Imagine SDK/API library. The provided data should be accessible only via that class (or classes) and it's methods.
    - (optional) Connect the SDK to NodeJS Express server (or another framework by choice) for extra points.
5. All transactions records must have a details page:
    - The created_at table field must be a link to the details page
    - The details page must contain all transaction details, including those displayed on the index page.
    - Error messages must be visible only at the details page of the transaction
    - The full error class must be visible on the details page
6. Above the Table from `3.` there must be a search area:
    - Should be able to search transactions by date range (from, to) including hours and minutes.
    - Must have drop-down with all table columns(without created_at) named Filters
    - Once the filter is selected and visible, it should be removed from the drop-down with the filters (in other words, avoid having the same filter appear twice in the search area).
        - When a user selects a value from the drop-down new filter must appear. The new filter contains two fields, one drop-down named 'match by'(with values: equal, starts with, ends with, contains) and one text input with label 'value'.
        - In the new filter, the user must be able to search for a value entered in the text input using the matching type selected from the filter's drop-down.
        - There must be button near the new filter to remove it
    - The user must be able to add filter for all values(table columns) in the Filters drop-down
7. Provide unit tests with Jest for the implemented "SDK" and for the UI components.
    - (optional) Add e2e tests with Cypress or similar tools for extra points.
    - Points will be taken if no tests are provided at all.
8. Use Linters for the code and for the styles.
9. Write documentation in Markdown (README.md) as simple setup guide.
10. The application must look like finished MVP aka. should look like finished concept.
    - Points will be taken for unfinished look.
11. It is required to upload the code to a private GitHub repository. Access should be provided only to us. Git commits should be grouped in the following way:
    - All subsequent commits should be logically organized, reflecting the steps you've taken developing the application.
    - Neither one large commit with all changes nor a multitude of smaller commits for every little tiny change.