Hey, thanks for agreeing to take up the coding exercise! This exercise will give us a good idea about your coding style and approach to building software. It would also form the basis of our future discussions in the hiring process. Please go through it and let us know the date by when you would submit the solution.

Exercise:

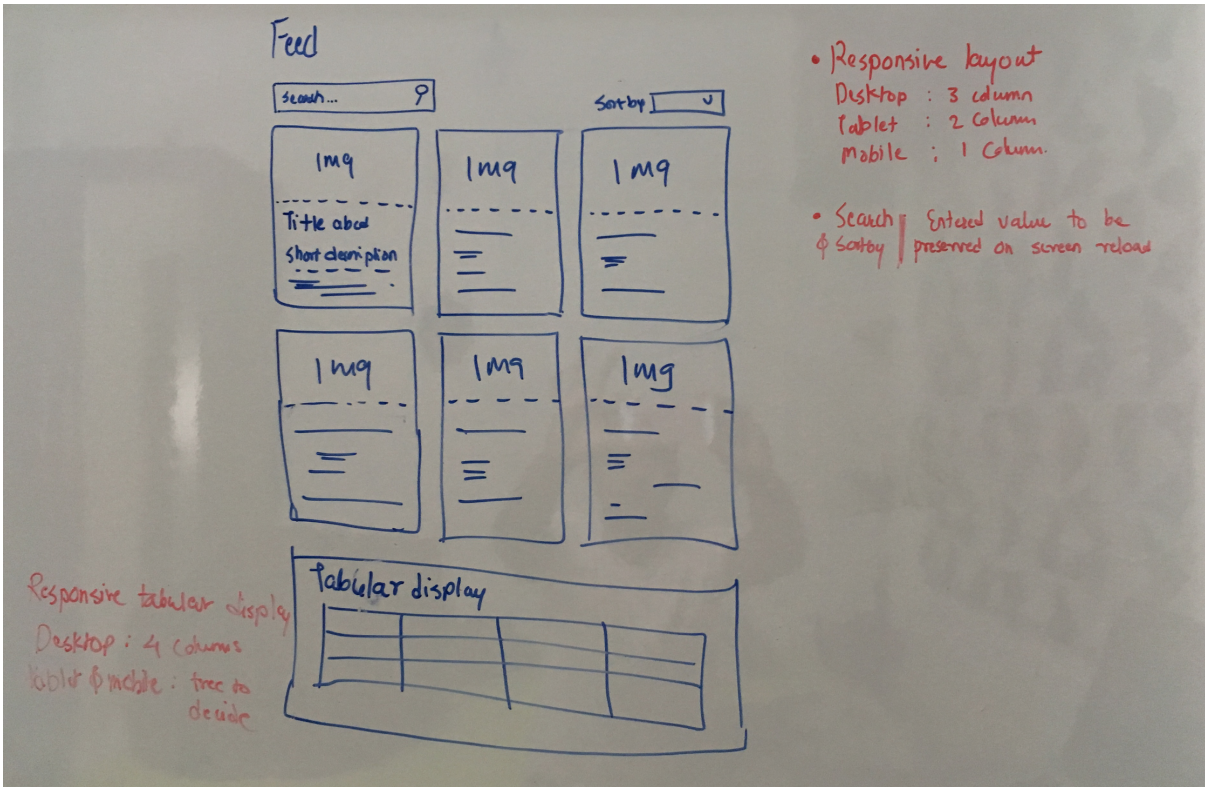
- Implement search and sort of a feed, with pagination.
  - Search:
    - should search in fields `name` and `description`
    - By default (i.e. for no search term) all posts should match the query.
    - Support exact match when the query contains a phrase within double quotes (like Google does)
    - Examples:
      - Given the following posts:
        - Post 1 with `name: "The Lord of the Rings: The Return of the King"`.
        - Post 2 with `name: "The Lion King"`.

Search Term	**Post 1 matches?**	**Post 2 matches?**
`the king`	Yes	Yes
`"the king"`	Yes	No

  - Sort:
    - allow sorting by `name` and `dateLastEdited`
  - Pagination:
    - Use 'page numbers' style of pagination
    - Include total count in the response matching the current query result

- **Backend:**
  - Use a strongly typed language like Typescript (Node.js) or any JVM based language, with or without an application framework.
  - The web APIs can be ReSTful or GraphQL-based.
  - Search should be implemented in-memory. Design it in such a way that it would be easy to replace it with a microservice in future.
  - Use the attached `mock_data.json` as seed data
  - Write meaningful unit and/or integration tests where appropriate, preferably following TDD.

- **Frontend:**
  - Wireframe



  - Responsive layout
    - The grid should show 3 columns on Desktop screens, 2 on Tablet screens, and 1 on Mobile screens.
    - The table below the grid should always have the same width as the grid. You're free to decide how to arrange columns inside the table
    - Use the mobile-first approach for designing the layout.
  - Search & Sort integration
    - Search terms entered in the input box should be passed to call to the web service implemented on the backend and display the results in the grid and table
    - Selecting the Sort option from the dropdown should call the web service accordingly to update the results.
    - Include pagination links to control which part of the search result should be displayed.
  - State Management
    - The Grid content should be paginated matching the search and sort criteria.
    - The Table rows should match the data currently shown in the grid.
    - UI state should be preserved when the web-page is refreshed
      - i.e. data loaded on refresh should be as per the value entered in the 'Search' field and the value selected for 'Sort by'.

        - bonus points if all UI states of the entire app are driven by routes/URLs

    - Other requirements
        - Use a component-oriented view library for implementation.
        - Don't use any 3rd-party styling/component framework.
        - Write **meaningful unit tests** where appropriate.

- The purpose of this exercise is to demonstrate the best practices you're aware of in terms of writing clean, modular, testable, and well-tested code.

- Provide a git repo link containing the project files.
    - If you follow a TDD approach, please commit in a sequence that helps demonstrate it.
    - If you're used to rebasing to make the commits history easier to follow, you can do so for this assignment too!

- Include instructions for running the app and executing the tests – preferably a single command for setting up dependencies and services, running the app, executing the tests, etc

Do let us know if you have questions at any time during the assignment by writing to tech-hiring@kodo.in

Good luck!