Task description:

The task is about a small web application for managing **(Flights)** information.

API Integration:

1- Install @musalasoft/flights-server
2- Add it to the scripts section of your package.json:

```
"server": "flights-server"
```

Run it with the command “ npm run server “

- Check carefully the API documentation and request parameters. The documentation is self-explanatory

Tech to use:

Reactjs and you can choose whatever library will help to complete the task (keep it minimal as much as possible)

Requirements:

Story 1:

1. Create a web page to display the first page and 10 items of the list of flights showing `Code`, `Capacity` and `Departure Date` of each flight using a Table component.

2. Add a navigation control that allows to change table pages and also to change the number of items per page.

3. Change the navigation element to update the URL of the page when clicked and to show the correct page and page size when the browser page is refreshed.

4. Validate the parameters just added in the URL to match the format accepted by the server for page and page size and redirect to a "Bad request" page in case the parameters are invalid.

5. Add a column to display an icon in the cell that, when clicked, shows a preview of the image or nothing in the flight has no photo.

Story 2:

1. Add a page and display a form that allows to creation of flights with Code, Capacity and Departure Date inputs.

2. Make sure the `code` is unique before creating the flight.

3. Add a new control to this form with an optional file upload for a `photo` field. Only images are allowed to be uploaded.

4. Add a preview to show the photo before submitting when the user changes the value of this control.

Story 3:

1. Add a column with a delete button on it that allows you to delete flights by `id` in the flights table, and you must ask for confirmation before deleting.

2 Add an edit button in the same column as the delete button that allows to edit the properties of a flight.

3. Add a search input on the flight list page that allows filtering by `code` and Update the URL as well to persist the user filter and validate it. If the user types an invalid code show an error on the search input and no results on the table.

Story 4:

1. Add a register screen that allows to creation of new users with `name`, `email` and `password` fields.

2. Add a login screen that allows to authenticate the user with `email` and `password`.

3. Enable security on the server and make all your requests authenticated. Do not allow users who are not registered to access the application.

4. Add a toolbar on top of your authenticated page showing the authenticated user with a menu with an option to logout. Make sure error pages also have this toolbar.

Story 5:

1. For mobile view, display your data in cards instead of tabulated view.

2. Create your custom component for the cards. This component must be generic and allow for the specification of any custom content for the body. Build your styles for this component. Also, use scalable units based on font size.

Nice to have:

1. When the security credentials are expired redirect the user to the login screen.

3. Automatically refresh the tokens when they expire without logging out the user.

Notes:

- Focus on doing fewer stories but doing them right.
- Make sure to include input validations and request notifications for the user.
- Use whatever styled component library or basic CSS styling we don't look for a great design but make it decent.

Good luck 😊