In this assignment, you will create a manager dashboard application using Next.js. This app will serve as a platform for admins to manage challenges, allowing them to create, edit, and delete challenges. Since the backend for managing challenges has not yet been implemented, you will use a basic JSON server to simulate CRUD operations on challenges.

Initial Setup

NextJS

Before you start, make sure to create a NextJS application and configure it to run on an unused port (8080 for example).

JSON Server

Install json-server. It is a Node.js package that allows you to quickly set up a RESTful API with CRUD (Create, Read, Update, Delete) operations based on a JSON file. Here's the data file that we are going to use:

```json
{
  "challenges": [
    {
      "id": "1",
      "title": "Palindrome Checker",
      "category": "Strings",
      "description": "### Problem Statement:\nWrite a function that checks whether a given string is a palindrome or not. A palindrome is a word, phrase, number, or other sequence of characters that reads the same forward and backward.",
      "level": "Easy",
      "code": {},
      "tests": [],
      "createdAt": "2024-04-02"
    },
    {
      "id": "2",
      "title": "FizzBuzz",
      "category": "Logic",
      "description": "### Problem Statement:\nWrite a program that prints the numbers from 1 to 100. But for multiples of three, print 'Fizz' instead of the number, and for the multiples of five, print 'Buzz'. For numbers that are multiples of both three and five, print 'FizzBuzz'.",
      "level": "Easy",
      "code": {},
      "tests": [],
      "createdAt": "2024-04-02"
    },
    {
      "id": "3",
      "title": "Binary Search",
      "category": "Algorithms",
      "description": "### Problem Statement:\nImplement the binary search algorithm to efficiently find the position of a target value within a sorted array. The algorithm compares the target value to the middle element of the array and continues narrowing down the search until the target value is found or the search space is empty.",
      "level": "Moderate",
      "code": {},
      "tests": [],
      "createdAt": "2024-04-02"
    },
    {
      "id": "4",
      "title": "Merge Sort",
      "category": "Algorithms",
      "description": "### Problem Statement:\nImplement the merge sort algorithm to efficiently sort an array of elements. The merge sort algorithm divides the array into two halves, recursively sorts the sub-arrays, and then merges the sorted halves.",
      "level": "Hard",
      "code": {},
      "tests": [],
      "createdAt": "2024-04-02"
    }
  ]
}
```

Then run:

```
json-server --watch db.json
```

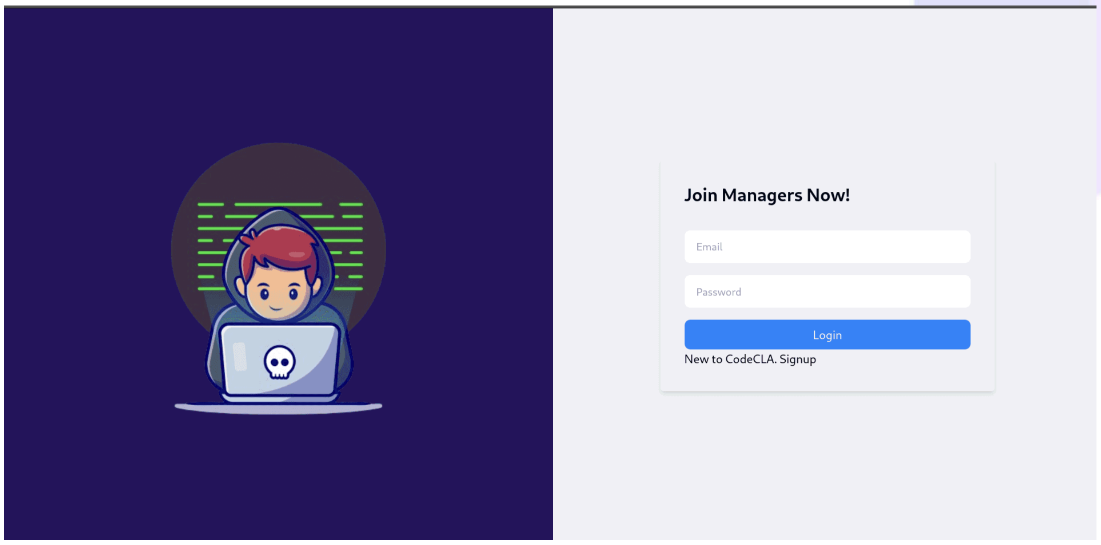
ShadcnUI

In this application, you are going to use the shadcn/ui design system. It's a rich components library. It has a variety of ready, modern and customizable React components. Please refer to the documentation for more details of how to use it.

1. Authentication

First, you're asked to develop the UI components and pages, setup redux and api for authentication.

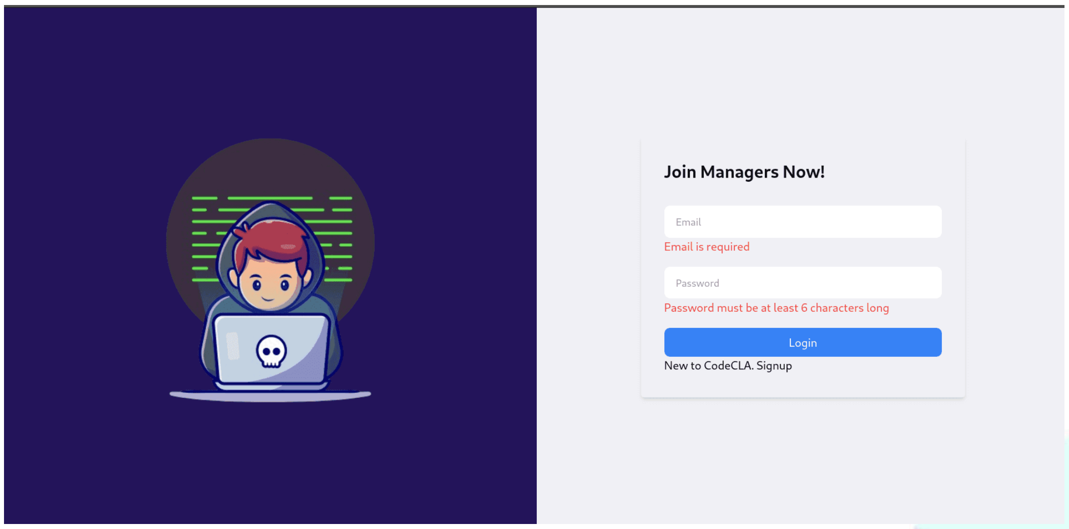
1.1. Singin Page



- Develop the UI component.
- Setup routing to the signup page if the Signup link is clicked.

Use react-hook-form and zod to validate the data

- The email should be a valid email.
- The password should be at least 6 characters long.
- If the data is not valid, error messages should appear bellow each non-valid input as shown in this picture:


Join Managers Now!

Email

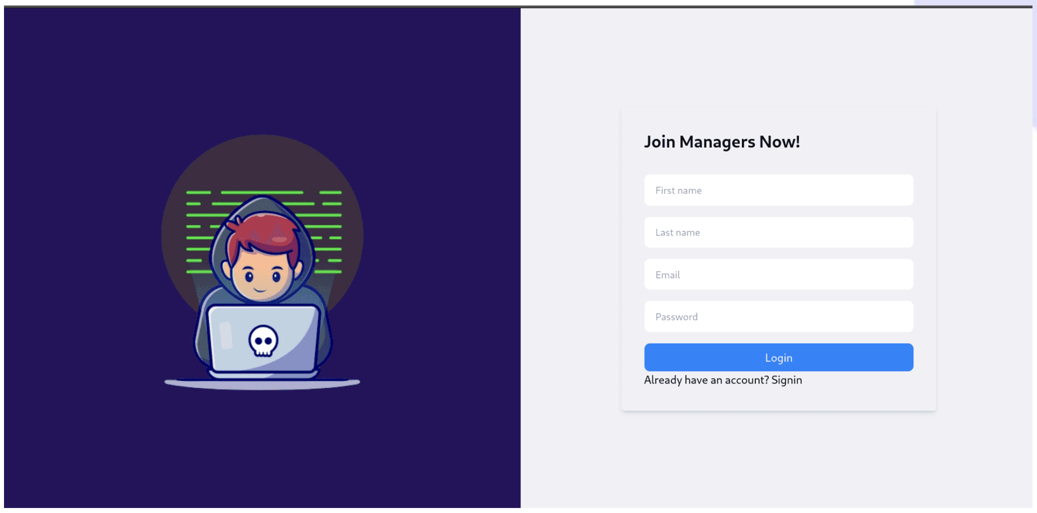
Email is required

Password

Password must be at least 6 characters long

New to CodeCLA. Signup

1.2. Signup Page


Join Managers Now!

First name

Last name

Email

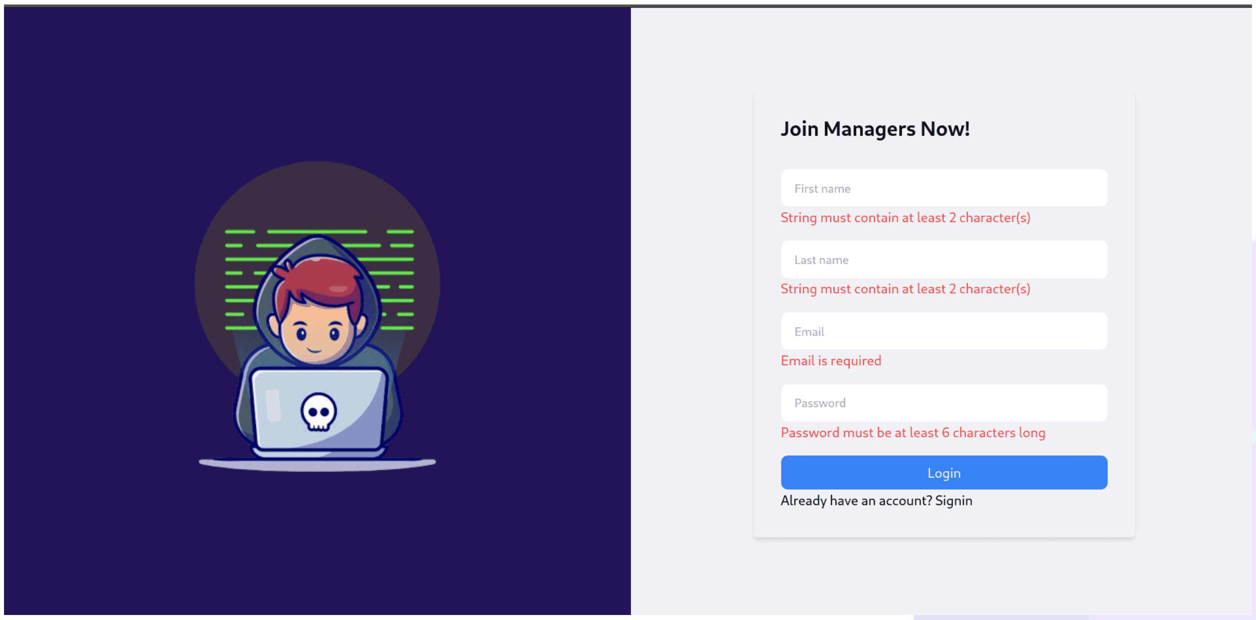
Password

Already have an account? Signin

- Develop the UI component
- Setup routing to the signin page if the Signin link is clicked.

Use react-hook-form and zod to validate the data

- The first name should be at least 2 characters long.
- The last name should be at least 2 characters long.
- The email should be a valid email address.
- The password should be at least 6 characters long.
- If the data is not valid, an error message should appear below each non-valid input, as shown here:



Join Managers Now!

First name

String must contain at least 2 character(s)

Last name

String must contain at least 2 character(s)

Email

Email is required

Password

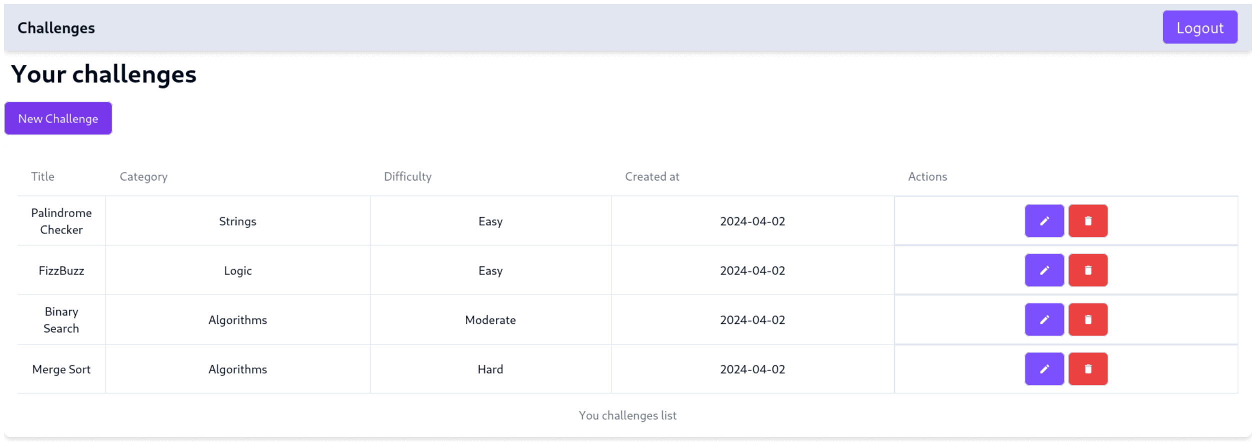
Password must be at least 6 characters long

Login

Already have an account? Signin

2. Dashboard

For the dashboard page, you'll need to create a Next.js component that includes a navigation bar component and a table to display the created challenges.



Your challenges

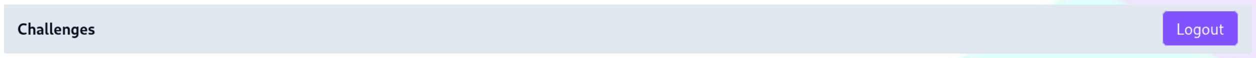
New Challenge

Title	Category	Difficulty	Created at	Actions
Palindrome Checker	Strings	Easy	2024-04-02	
FizzBuzz	Logic	Easy	2024-04-02	
Binary Search	Algorithms	Moderate	2024-04-02	
Merge Sort	Algorithms	Hard	2024-04-02	

You challenges list

2.1. Navbar Component

- Create a navbar component.
- It should contain a link to the home page (Challenges) and Logout button.

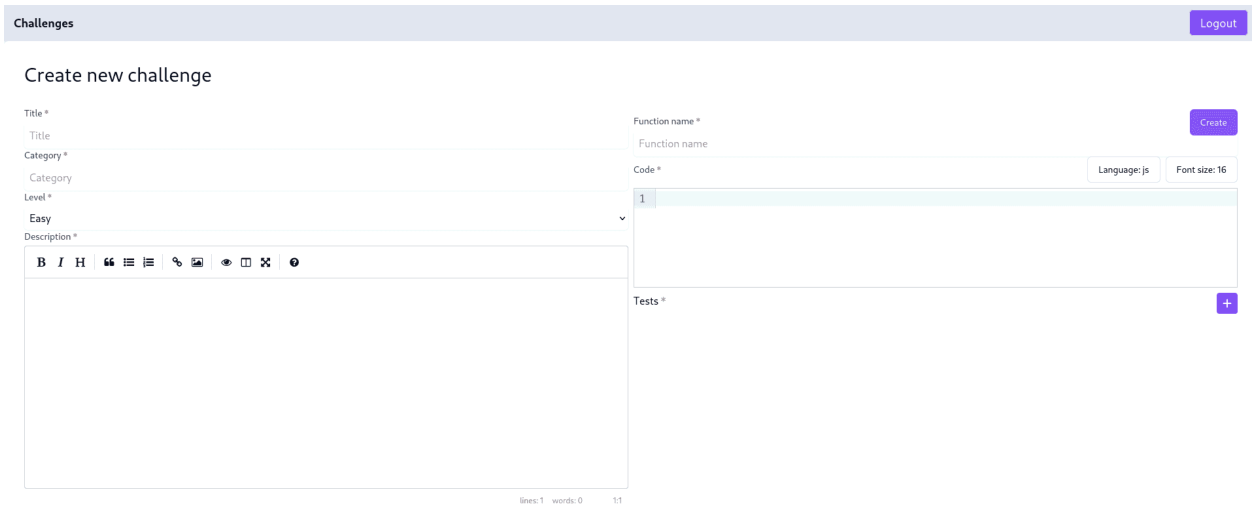


2.2. ChallengesList Component

- Create a ChallengesList component that contains the table of challenges.
- Create a server action that fetches the list of all the challenges, it should invoke json-server get all challenges endpoint.In the challenges listing component, load the challenges and render them using shadcn tables.
- Each entry of the table contains two actions, Edit and Delete.The Edit action is a link that navigates the user to the ChallengeEdit page.
- The Delete action is a button that deletes the entry. (Use server action and revalidatePath)
- Implement a "New Challenge" link component. When clicked, it should redirect the user to a page for creating a new challenge, which displays the ChallengeForm.

2.3 ChallengeForm Page

Now, you are going to implement the page that displays the ChallengeForm used to create a new challenge.

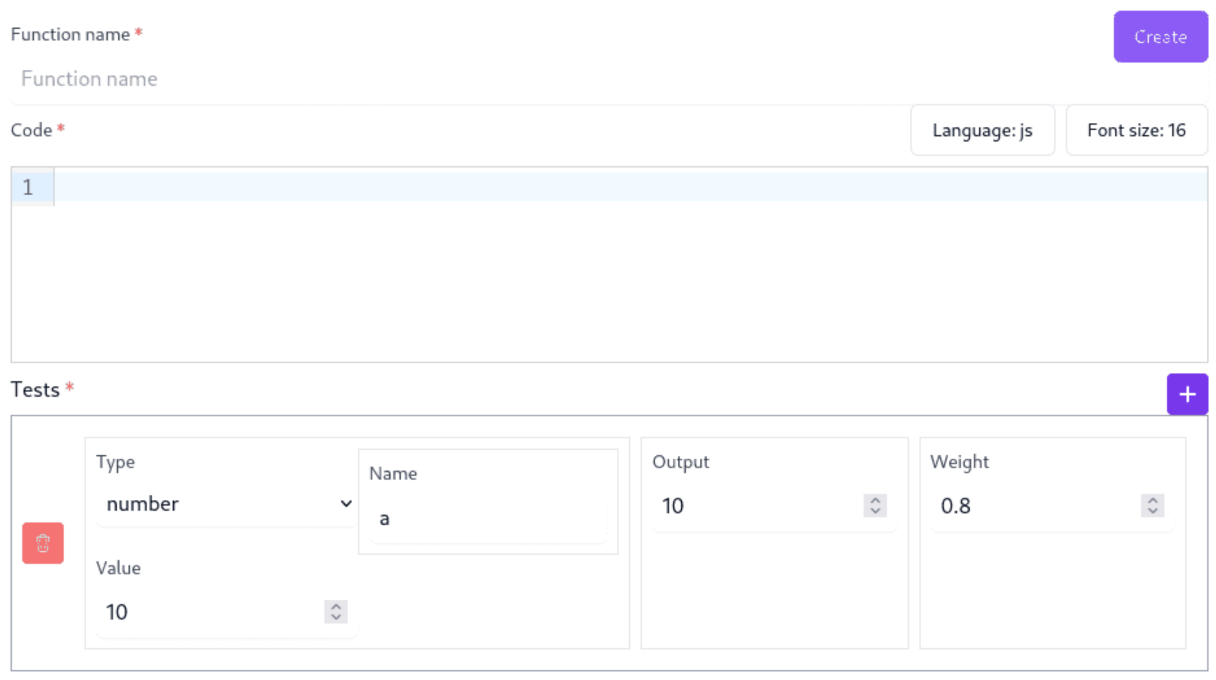


Note: For inputs make sure to use shadcn inputs.

- Design a form divided into two halves: left and right.
- On the left side, include inputs for title, category, and level, where level can have one of the following values: Easy, Moderate, or Hard.
- Integrate a markdown previewer on the left side using the react-simplemde-editor library, available here. Install and utilize this library for its simplicity and power.
- On the right side, we have the function name input of the coding challenge.
- Just bellow it, we have the code mirror that you used previously in the Workspace of your coder dashboard react app. You can follow the same process.
- Similar to the previous react application, the user can configure the programming language (we support js and py languages only) and the font size which should be stored in the centeralized redux state for global access.

- Create a LanguageMenu and FontSizeMenu components to achieve this. You can make use of shadcn dropdown components.
  Note: For the functions, to simplify the UI and focus on the concepts, we support functions with only one argument of two types: number and string.

The user can then add test cases by clicking on the plus button.



Function name *
Function name
Code *
Language: js
Font size: 16
Tests *
Type: number
Name: a
Value: 10
Output: 10
Weight: 0.8

A test case contains:

- A type which is the type of the input (number or string).
- A name which is the name of the function argument.
- A value which is the value of the function argument in the testcase.

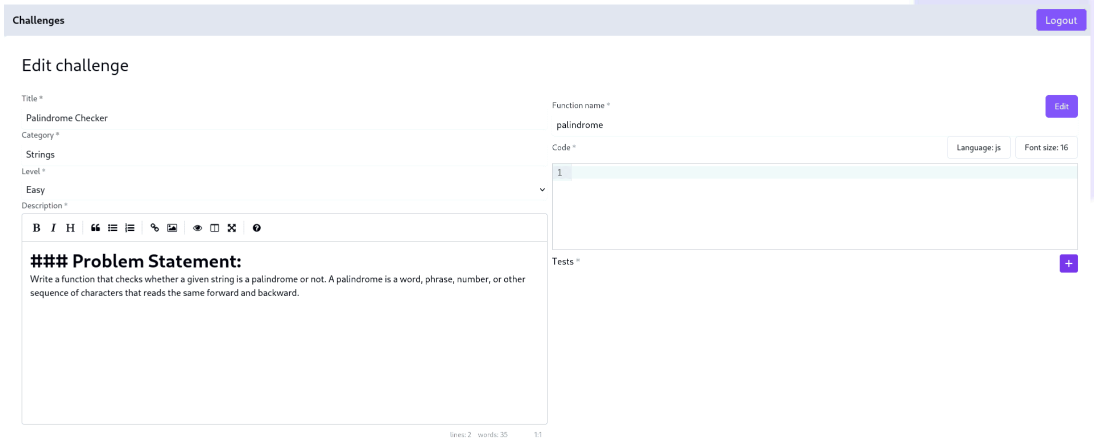
- An output which is the expected output of the testcase (the return value of the function).
- A weight which is a float number between 0 and 1 indicating the importance of the testcase.

The user can delete the testcase by clicking on the red delete button.

- Once the user clicks on the Create button, the form is first validated and submitted, and a post request should be sent to the json-server to create the challenge.
- Upon successful response, ensure to display an alert indicating the status of the operation.
- Upon receiving an error response, ensure to display an alert indicating the status of the operation.
  Note: For alerting, you can utilize shadcn-toast

2.4. ChallengeFormEdit Page

When user click on the edit link on the home page, they will be redirected to the ChallengeFormEdit page. The edit challenge page is straightforward; it should utilize the same ChallengeForm. However, the form should receive the challenge data as props, and all the state should be initialized with the props data as follows:



Edit challenge

Title *
Palindrome Checker
Category *
Strings
Level *
Easy
Description *

Problem Statement:
Write a function that checks whether a given string is a palindrome or not. A palindrome is a word, phrase, number, or other sequence of characters that reads the same forward and backward.

lines: 2 words: 35 1:1

Function name *
palindrome
Code *
Language: js
Font size: 16
Tests *

- The user can make updates to the form fields and submit the new challenges to an edit challenge API endpoint (That you need to add as a server action), which triggers the JSON-server update challenge endpoint.
- Upon successful response, ensure to display an alert indicating the status of the operation.
- Upon receiving an error response, ensure to display an alert indicating the status of the operation.

2.5. Delete Challenge

When the user clicks on the delete button on the home page, the challenge should be deleted.

- The delete button should trigger a server action that deletes the item.
- Upon successful response, ensure to display an alert indicating the status of the operation.
- Upon receiving an error response, ensure to display an alert indicating the status of the operation.
- Ensure to reload the data after a successful deletion using revalidatePath.