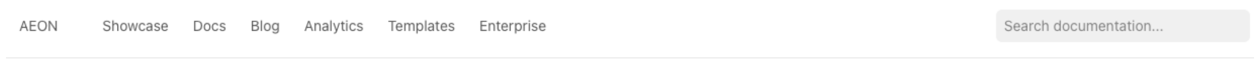
Challenge 1: Build a Basic Navbar with Next.js

Objective: Create a simple web application using Next.js that includes a functional navigation bar (navbar). Focus on layout and basic interaction without adding backend functionality.

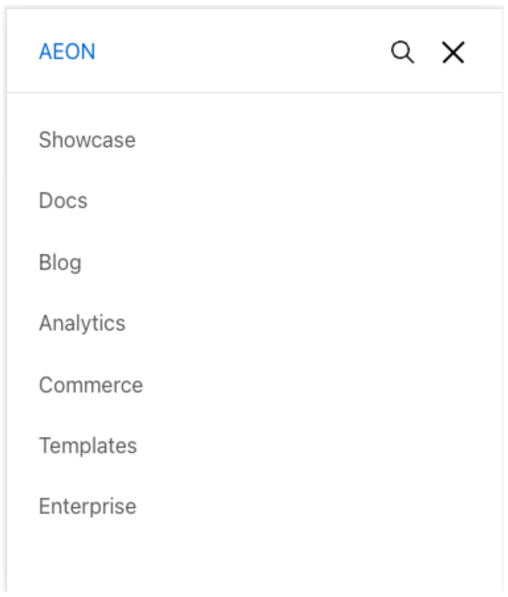
Requirements:

1. **Navbar:**
   - The navbar should have a title, which can be any name of your choice.
   - Include a search input field (no functionality needed).
   - The navbar should be collapsible:
     - By default, it is closed.
     - A hamburger icon should be used to open it.
     - When open, it can be closed by clicking an [X] icon (replace this with a hamburger icon when the navbar is closed).
     - Add [Login] button that should navigate to Challenge 2

Desktop View



Mobile View



Challenge 2: Build a Simple Login Flow with Next.js

Objective:
Create a basic login system using **Next.js** where the user enters a username and password. The login flow will include multiple steps and demonstrate interaction with a mock API.

Requirements:

1. Username Input (Step 1):

- Create a simple form where the user can input their **username**.
- Once the user submits the username, trigger an API call to a mock API.

2. Mock API (Step 2):

- Use **Next.js API routes** to create a mock API endpoint (e.g., `/api/getSecureWord`) that returns a static secure word, such as `"secure123"`.
    - The API does not need to validate the username, just return the secure word for any input.

3. Display Secure Word (Step 3):

- After the user submits their username and the mock API returns the secure word, display the secure word on the page.
- Provide a button labeled **"Next"** for the user to proceed to the password input step.

4. Password Input (Step 4):

- When the user clicks **Next**, prompt them to enter a **password**.
    - The password should be entered in a password input field where the text is **masked** (obscured with asterisks or dots).

5. Encrypt Password (Step 5):

- Before submitting the password to the mock API, **encrypt** it using any hashing library
    - This should ensure the password is **never sent in plaintext** to the API.

6. Final Submission (Step 6):

- Send the **encrypted password** and username to the API (e.g., `/api/login`).
    - The mock API will simply accept the encrypted password and return a success response.
- On submission, display a message (e.g., "Login successful") without revealing the actual password.

7. Final Submission (Step 7):

- Navbar should be cleared once login

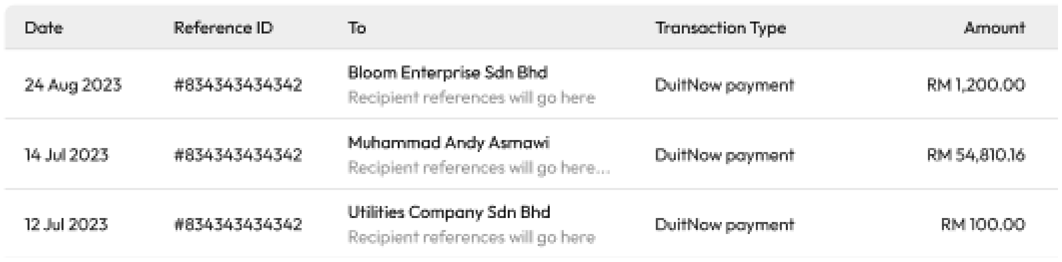
Challenge 3: Create a Simple Table with Data Fetching from a Mock API

1. After a successful login, the user is redirected to a page displaying a simple table.
2. The table should display data fetched from a mock API (e.g., `/api/transaction-history`).

Mock API:

- Create a mock API in the Next.js that returns a static JSON response.
- The API should return an array of objects.

Build a simple table as below;



Date	Reference ID	To	Transaction Type	Amount
24 Aug 2023	#834343434342	Bloom Enterprise Sdn Bhd Recipient references will go here	DuitNow payment	RM 1,200.00
14 Jul 2023	#834343434342	Muhammad Andy Asmawi Recipient references will go here...	DuitNow payment	RM 54,810.16
12 Jul 2023	#834343434342	Utilities Company Sdn Bhd Recipient references will go here	DuitNow payment	RM 100.00

Challenge 4: Unit tests (Optional)

Objective:
Create unit tests for above challenges

Notes:

1. Provide us with how to run your code in a readMe file
2. Please complete the test given in **Javascript or Typescript**
3. Please use **NextJs** for the framework
4. Option 1: send us the answers in a zipped file via Google Drive link or Option 2: you can also provide your answers through Github. Please provide us with the repository link.
5. If you are shortlisted, we are expecting you to **demo and walk us** through your solution from your machine during the interview