Frontend

View the Result

source code: https://github.com/KaiyuWei/brink_page

I deployed the frontend page on Vercel, see: https://brink_page_48if3yc6a_kaiyuweis_projects.vercel.app/

Bonus

✅ Sliders created for mobile phones
✅ JavaScript library used: Node.js, React.js, Next.js, react_elastic_carousel for creating the slider, and Tailwind for css.
✅ Next.js used
✅ **Bonus of Bonus**: This webpage not only works for desktops and mobile phones, I also designed and made the tablet mode.

How I developed it:

1. Actually making such a webpage was quite strange for me at first. In my current company Copernica, the webpages we made are not as complicated as this. Because we are a 2B platform, so we have a quite fixed webpage structure and style. At first I even thought I need to make two versions of it: one specially for the desktop and another one specially for the mobile devices XD. But Im quite proud that I have learned the skills I need to develop such a page in just a couple of days.
2. One thing I spent more time on was the constructing of the gallery flex boxes. I tried quite a couple of layouts of flex boxes and finally found the best one. It was helpful that Google Chrome had very good dev tools, by which I can observe the structure, styles, and make experiments.
3. I felt programming mind helped me a lot. E.g. the media query looked complicated at first, but it was simply an equivalent of `if...else` statement. So I quickly got it after determined the breakpoints for different media sizes.
4. The slider_making itself was not that hard. I used a library, "react_elastic_carousel", combining with the media query to make it only show up when the media are mobile devices. Btw, I learned this word "carousel" in this process XD.

Backend

source code: https://github.com/KaiyuWei/register

View the Result

By AWS deployment

I deployed the backend assignment on AWS with the help of docker, see: http://54.226.179.78:3000 (Ill turn off the instance on 10, October since I may pay for running it XD). In this web app users can register, login, log out, and reset the password in case they forget it.

By Docker Containers

The Docker images are available in docker hub by names **kykywei/register-server:v1.0**(serverside) and **kykywei/register-app:v1.0**(client side). Building from the docker images requires an Arm64 host machine, for I developed it on a Mac with M2 chip.

1. Pull the docker image:

```
docker image pull kykywei/register-app:1.0
docker image pull kykywei/register-server:1.0
```

2. Run the docker images:

```
docker run -it -d --name user-app -p 3000:3000 kykywei/register-app:1.0
docker run -it -d --name user-app -p 8000:8000 kykywei/register-server:1.0
```

3. visit `http://localhost:3000` and then you can see the login page.

The Database Management

I also deployed the MySQL database on AWS RDS, you can log in by `mysql -h brink-users.cjyitoqnpwnw.us-east-1.rds.amazonaws.com -u kaiyuwei -p`, and the password is `YqeTD07AkU8pitYEcIBa`.

After log in, choose the database `use brink`. There are two tables in the database: `users` and `sessions`. One for user information storage, and another one for login session storage. Note that though I was able to store and update the login time, as required by the bonus part, by default AWS RDS uses UTC time zone. Unlike most of the MySQL databases, it is hard to change this immediately (either by calling AWS customer service or waiting for their next maintenance time window), so the login time is 2 hours later than the actual time here in The Netherlands.

Bonus

✅ The reset password flow created. view it by the "forget password" link on the login page.
✅ Latest user login time implemented. Check it in the database on `users` table.
✅ **Bonus of bonus**: email service deployed on AWS SES. It helps validate users email address and reset the password. Youll receive and email after submitting the registration form, or after submitting the email when in the password resetting process. (If you do not receive the email, check the junk box).

How I developed

1. I started from creating some APIs, like the API for using us. The APIs you see in `route/auth.js` and the controllers in `controllers/auth.js` and something I made at first. After making each API, I tested it using Postman.
2. I needed two AWS services in this project: AWS SES for sending emails, and AWS RDS for deploying MySQL database. I just launched two instances and got connected to them by either SSH keys or username_password combination.
3. The user is authenticated by login sessions and user id in cookies. Every time a user logs in, a new session with the users user id is sent to the front end, and after the user logs out, the cookie is removed.
4. I used to work with MongoDB a lot, and I used to use jwt refresh tokens to do the user authentication. However, things changed in this project. Im required to use MySQL database. and defining a column with MySQL `VARCHAR` values requires declaring of the string length. I did some googling and did not find a specific answer about the longest length of the jwt token, which made it hard to determine how long should I define the `VARCHAR` if I wanted to store the token in the database. In this condition, I turned to using log in sessions and cookies to do the authentication with the help of Exress session.
5. After all the APIs were developed and tested, I started making the frontend page. I used the React router for page routing, and implemented conditional routing by React Outlet components. Some pages are protected from the non_logged in users, like the `/dashboard`.
6. Since it is just some pages with simple components, I did not have too many problems with them. Just used some basic Bootstrap components to fill the page. I spent a little bit more time to make it look better.
7. After testing it locally, I made docker images from this project and deployed this project on AWS, as you can see in the link at the beginning.