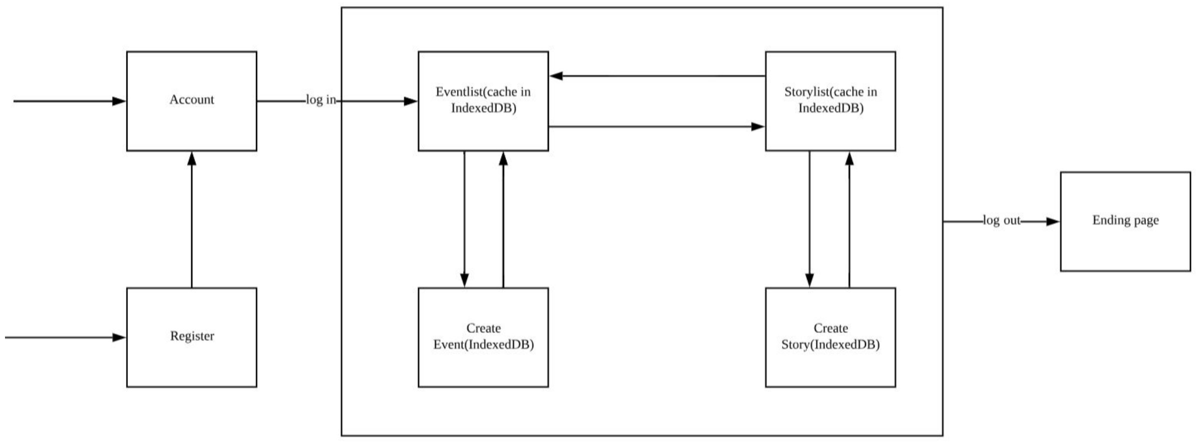
Introduction

Through a period of time, our term develop a simple progressive Web App with a local database and a nodejs webserver. The purpose of the App is to record events and write stories at music festivals. That is the first part of our work. The work is a responsive web which allow people to use it in computer or mobile. The app allow to:

- Create a new event inclusive of event name, event author and location(haven't achieved).
- Return the newly created event and existing events in a eventlist.
- Create a new story inclusive of story title, story author and photos(haven't associate with event).
- Return the newly created story and existing events in a storylist.
- Search events and stories though key words.
- Support offline function.

Diagram



For Each Task

1. Interface in insert and search data via forms

Challenges: At this part, we need to develop a responsive web app, which means not only our pages can be implemented on a PC but also provide a great user experience on a smaller screen. At first, we try to build the web page without using css/js libraries. During a long time trial, we find difficult to make the page responsive, it can not write all the style code by ourselves in style.css. The other challenge is we need to search different types of data, and it will decrease the look of page if we use more than one search bars, so we need to change the data type in a search bar. Otherwise, it is difficult to show multiple photos like Facebook.

Solution: We design about seven pages including log in; register; log out; eventlist; createevent; storylist; createstory and design the jump between pages. In order to achieve responsive layouts, we use bootstrap libraries, which support responsive web page, we just need to modify some relative code in bootstrap.css. After that, the pages can satisfy our special demand and achieve responsive pages. And we write a function to change the input type between text and date. And we make a great number of style to improve the look of the web (e.g a navigation bar to show the theme website image). What's more, it is easy to use carousel to show multiple photos.

Requirements: The design achieves a responsive webpage and provides the page to achieve our app features. It meets all of the requirements.

Limitation: The page is too simple because we do not have so many features. And we can also improve the look of this page.

2. PWA – caching of the app template using a web worker

Challenges: The web worker is new for us. So we have to learn about how it works, what kind of browser it could support, what the requirement is, and why it is convenient. The biggest challenges is that we are not familiar to it.

Solution: After learning some knowledge about it, we know that that we should install it first and then activate it. So it can cache some files we defined and provide them when users are offline.Besides, it need the HTTPS secure connection and latest versions of browsers. Concretely, we confirm what we should cache first and write them into service-worker.js. Then install it when the indexedDB is initialized. Now it could cache the certain files automatically. It would be better to add two event listeners to detect whether the users are offline or not and inform them. When users are offline, we can get files from web worker.

Requirements: It meets all the requirements. When the users are offline, we can still show the data we cached in web worker although it is stale.

Limitation: It need latest versions of browsers and https connection.Not suitable for all situation.

3. PWA-caching data using IndexedDB

Challenges: The challenges of this part can be summarized in one sentence: "We are not familiar with indexedDB." Because we have never used this database before, so we have to learn how to initialize it, what the structure it is, how it stores data and In what way it stores data. Besides, we have to decide what to store and how to store(in what datatype).

Solution: After learning the indexedDB, we have a clearer understanding of this database. We need to initialize it first and create two object stores, which are similar to the tables in SQL database. One of the object stores is to store the information of events, another one is to store the information of stories. Then we get the event data from frontend which includes “event name”, ”event date”, ”event location” and “event description” and store them into the database using object store. Similarly, we store “story author”, ”story title”, ”creating date”, ”story description” and the “picture paths” of the pictures uploaded by users. In particular, “creating date” is not inputted by users. Instead, it is generated by our code automatically. And the “picture paths” is returned by the server when we are uploading the pictures. After that, we could get data from the database and show it to users if necessary.

Requirements: It meets all requirements because it can store data and provide data normally and successfully.

Limitation: We haven’t considered the speed of storing and getting data. So if too many users are operating, it would be a little bit delay.

Conclusion

During the development of this web app, we learning how to build a webpage, and know the knowledge of javascript, ajax and PWA. Although the function in this App is simple, it build a powerful architecture which should be applied in all of the Web App. In addition to programming skills, we know the importance of work division. If the division is not clear, our work will be confused.

Division of Work

Xiongjia Zhou lead the implementation of 3 and wrote the corresponding documentation.
Xuanyi Zhou lead the implementation of 1 and wrote the corresponding documentation .
Yiwei Xu lead the implementation of 2 and wrote the corresponding documentation.

Extra Information

It need multer and fs-extra dependencies. We have added them in package.json.