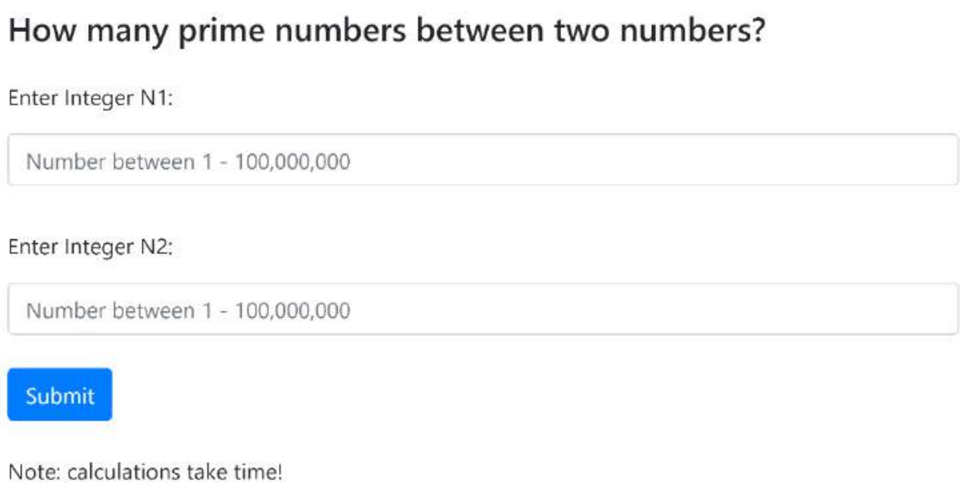
Please see the **demo video**

In this assignment you need to improve the WebWorker exercise. You'll be using a **Web Worker** and **IndexedDB** database.
The main HTML form is based on **Bootstrap4**:


How many prime numbers between two numbers?

Enter Integer N1:

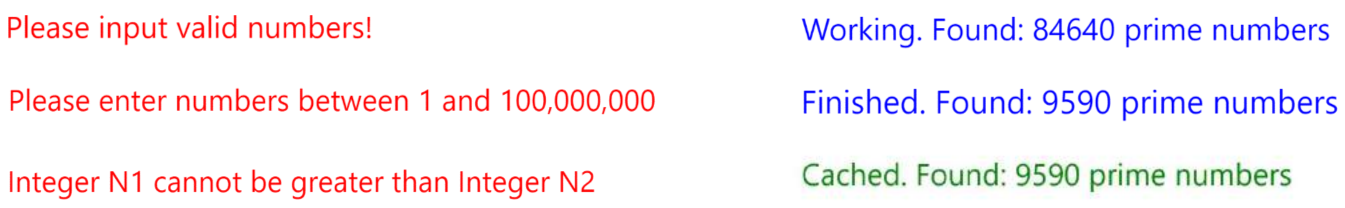
Number between 1 - 100,000,000

Enter Integer N2:

Number between 1 - 100,000,000

Note: calculations take time!

The data is sent to a Web Worker and the following messages are displayed:


Please input valid numbers!

Please enter numbers between 1 and 100,000,000

Integer N1 cannot be greater than Integer N2

Working. Found: 84640 prime numbers

Finished. Found: 9590 prime numbers

Cached. Found: 9590 prime numbers

Main Requirements:

1. Your solution must look and behave as close to the **demo version** as possible!
2. You must use the provided template (unzip it inside **public_html** folder). Please add your changes to 3 files: index.html, ./js/main.js and ./js/worker.js.
3. You must use **Bootstrap 4** classes and components in the **index.html**
4. The **Web Worker** code must be inside **worker.js** file
5. The main thread JS code must be inside **main.js** file.

6. You must use the **Web Worker API** to send data to the Web Worker.

7. **Web Worker** must return the object with the following keys: '**status**' ('error', 'working', 'done') and '**msg**' (red messages above or the total number of found prime numbers).

8. You must correctly process data returned by the Web Worker in the **main.js** file and display one of the messages shown above.

9. You must save (cache) in the IndexedDB the total number of found prime numbers for a given interval [Integer N1 – Integer N2].

10. You must correctly pull data from the IndexedDB and display it (see green message example above). Please use **primesDB** database with **primes** object store.

11. You need to take into account cases when IndexedDB API is not available in the browser (i.e. no caching) or Web Worker API is not available (form submission must be disabled).

12. Your solution must have optimal time and space complexity.

13. Your application should not have debugging/console messages.

14. You must submit **3 files: index.txt, main.js** and **worker.txt**
See "Submission Requirements" below.

Hints:

1. You might want to use number1@number2 key in the IndexedDB

2. To avoid duplicate code when you work with IndexedDB database you might want to use Promise wrappers https://0xmend.medium.com/understanding-indexeddb-and-creating-our-own-promise-based-api-19c425132d0c

Submission Requirements:

This is an individual assignment. Even partially copied code will be subject to regulations against academic integrity. Do NOT discuss or share your solution with anybody. Posting this assignment or solution on the Internet is a violation of the Student Code of Conduct.

- You must submit three files: **: index.txt, main.txt** and **worker.txt**
  (save **index.html**, **main.js** and **worker.js** files with **.txt** extension).
- You must upload **text files** to the Assignment Dropbox by the due date.
- Verify that **text files** you submitted can be opened in the notepad (Windows OS).
- MS Doc or files with rich-text formatting won't be graded.
  Archive files (.zip, .rar, etc) or files with incorrect name/extension won't be graded.

- Your submission must be unique or have references.
- On a scale **0-10** please **self-grade your solution** in the comments section of the Dropbox.
  You'll get a bonus mark if you correctly self-grade your own solution.

Grading Scheme:

- You will get **zero or minimal grade** if your code doesn't run at all.
- Compilation errors and console debugging messages are major mistakes.

- Comments are not required but recommended. However, "debugging code" or commented out "old code" is a minor mistake (unless it's clearly stated in the comments that you want me to take a look at such code).