

URL Shortener

Waseem Aboliel 312378771

Majdi Aboliel 312314719

URL shortening is a technique in which a URL may be made substantially shorter and still direct to the required page. This is achieved by using a redirect which links to the web page that has a long URL. For example, the URL

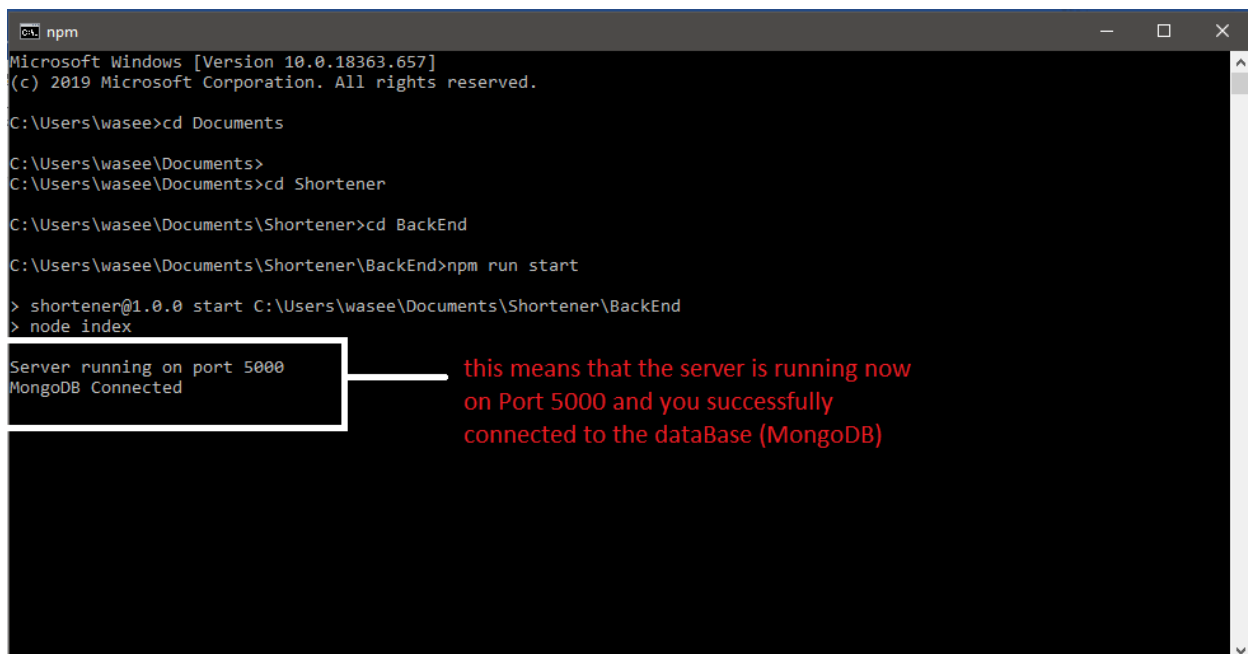
"https://example.com/assets/category_B/subcategory_C/Foo/" can be shortened to "https://example.com/Foo".

A friendly URL may be

- desired for messaging technologies that limit the number of characters in a message (for example SMS).
- for reducing the amount of typing required if the reader is copying a URL from a print source.
- for making it easier for a person to remember.
- And many more..

How to use:

1. Extract the zip file, you will see 2 folders (FrontEnd & BackEnd).
2. Save them on a simple folder so you can access them from the CMD.
3. Go to CMD (on the search panel search for CMD) then go to the folder BackEnd (by using cd)
For example, if your folder is in the Documents, You need to write cd Document/Something/BackEnd.
4. Now you need to write **npm run start** (make sure that you already downloaded npm and nodejs in your computer).
5. Now you will see something like this:



```
npm
Microsoft Windows [Version 10.0.18363.657]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\wasee>cd Documents

C:\Users\wasee\Documents>
C:\Users\wasee\Documents>cd Shortener

C:\Users\wasee\Documents\Shortener>cd BackEnd

C:\Users\wasee\Documents\Shortener\BackEnd>npm run start

> shortener@1.0.0 start C:\Users\wasee\Documents\Shortener\BackEnd
> node index

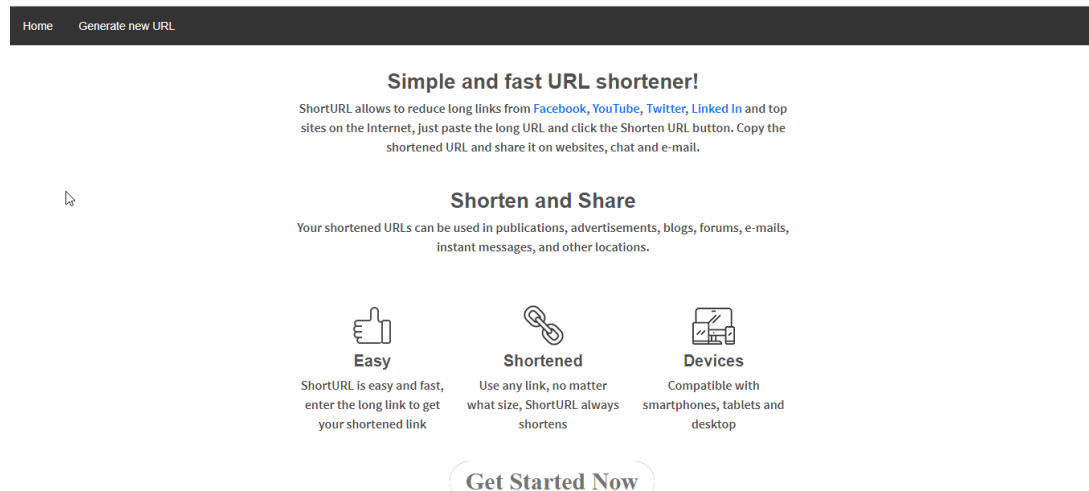
Server running on port 5000
MongoDB Connected
```

this means that the server is running now on Port 5000 and you successfully connected to the DataBase (MongoDB)

Now you are connected to the server and to the database.

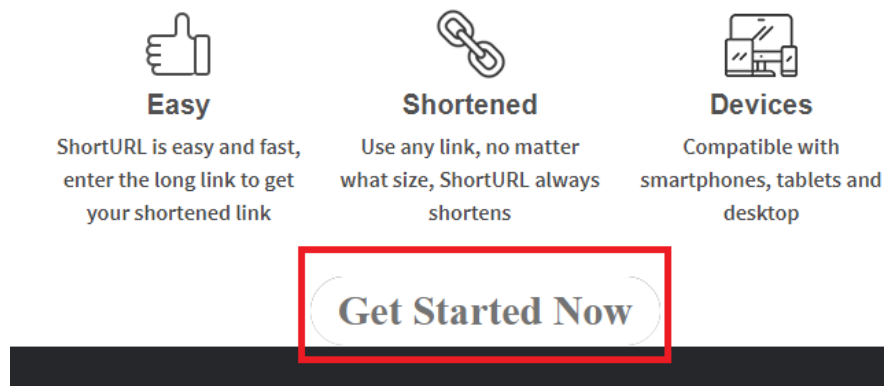
6. Now leave the server open and go to the second folder FrontEnd and open the html file (main.html).

7. You will see something like that:



This is the HomePage of the site. You can check it out!

8. Click on the Button (Get Started Now)



You will be redirected to the second page to generate a shortener URL.

9.

The screenshot shows a dark-themed web interface for a URL shortener. At the top, it says "GENERATE NOW!". Below this is a text input field with the placeholder "Enter long URL". To the right of the input field is a red instruction: "1. Add your URL here!". Below the input field is a white "Generate" button. To the right of the button is a red instruction: "2. Click on Generate". Below the button is a text area with the placeholder "Your short URL here". To the right of the text area is a white "Copy URL" button. Below the text area is a red instruction: "3. if your URL is a valid one, your Short link will show here". To the right of the button is a red instruction: "4. once you saw your Short link, click on Copy URL to copy the URL to your clipboard".

GENERATE NOW!

Enter long URL

1. Add your URL here!

Generate

2. Click on Generate

Your short URL here

Copy URL

3. if your URL is a valid one, your Short link will show here

4. once you saw your Short link, click on Copy URL to copy the URL to your clipboard

For example:

The screenshot shows the same dark-themed web interface for a URL shortener. At the top, it says "GENERATE NOW!". Below this is a text input field containing the text "google.com". Below the input field is a white "Generate" button. Below the button is a text area containing the text "http://localhost:5000/xkuAuU". To the right of the text area is a white "Copy URL" button.

GENERATE NOW!

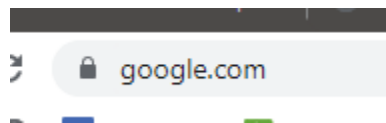
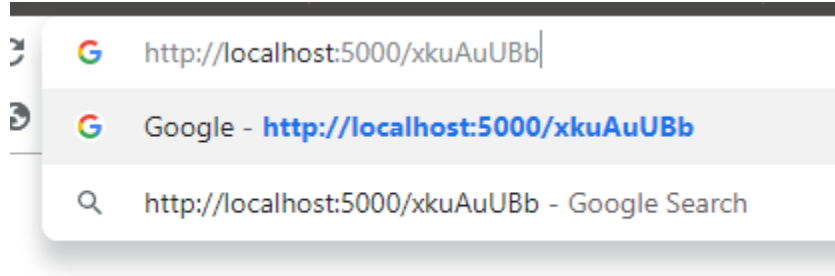
google.com

Generate

http://localhost:5000/xkuAuU

Copy URL

10. Now the URL is in your clipboard, just go to new tab and paste it and you will be redirected to the original URL!



Behind the scene:

what we do is that we send the original URL as a json from using fetch function with POST request.

```
async function sendData() {  
  await fetch('http://localhost:5000/api/url/shorten', {  
    method: "POST",  
    headers: {  
      "Content-Type": 'application/json'  
    },  
    body: JSON.stringify(POSTdata)  
  }).then((resp) => resp.json())  
    .then(function (response) {  
      GETdata = response.shortUrl;  
      return response;  
    });  
  
  if (GETdata) {  
    jQuery("[name='Short_URL']").val(GETdata)  
  }  
}
```

Then in the BackEnd we handle this POST request and check if the URL is already in our database then we just get the short URL, if not, we generate new code and add new json to the database that contain the original URL the code we generated and the short URL that we created (base URL + Code). Then we send as a response to the request another json that contain the original URL and the short URL.

When you try to use the short URL this mean that you sent Get request so in the BackEnd we handle this request by checking if the URL you used is on the database, if it's in the database then we redirect you to the original one if not in the database then you get a message (URL not found).

We used some application frameworks to help us:

- express- web framework we used to create our routes.
- config- a package that allow us to get virables from the json file.
- mongoose- very useful abstract to the data base mongoDB.
- shortid- help us to generate the URL code.
- valid-url- to validate the URLs that are sent to the API.
- nodemon- when we update something in the code, nodemon help us to reconnect to the server. (you can also connect to the server using npm run dev instead of npm run start).
- Cors- to override the policy headers.

Issues we faced and the solution

- The first one was the hardest one, when we try to send a POST request we get an error (Access to fetch has been blocked by Cors policy headers). Then after a while we added Cors to our project to override this.
- When we try to access the data base we get an error that we can't access it, then after a while we found the issue, the mongoDB let us use the data base only in a specific IPs that we added, when you try to access the data base from another IP you will get the error. The solution was to whitelist this IP 0.0.0.0/0
That means you can access the data base from every where