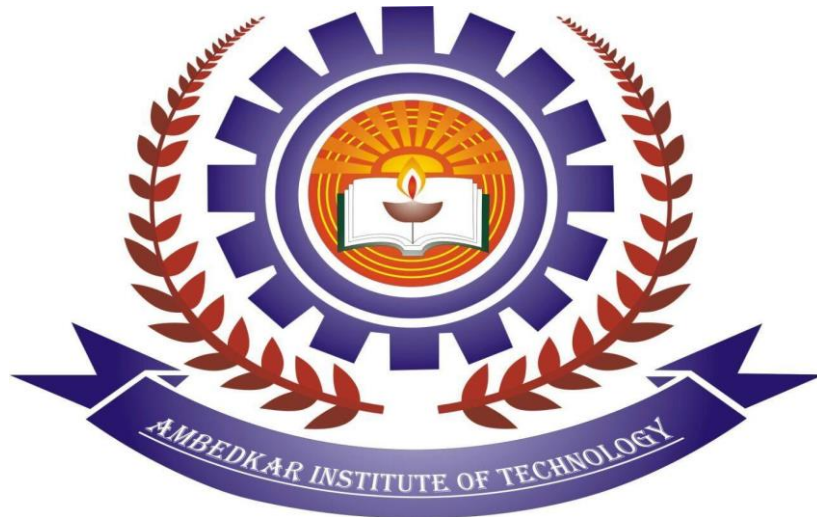


AMBEDKAR INSTITUTE OF TECHNOLOGY

ISO 9001:2015 CERTIFIED INSTITUTE

Shakarpur, Near Nirman Vihar Metro Station, Delhi-110092.

Website- www.ambp.in



BCA DEPARTMENT

A Report on Minor project

WEBSITE

SESSION: 2020-23

SUBMITTED TO:

Mr Shivam Gupta

(BCA DEPARTMENT)

SUBMITTED BY:

GAURAV BHANDARI

UNNIKRISHNAN. V

ACKNOWLEDGEMENTS

On this great occasion of accomplishment of our project on Website Designing (Narrow Links), we would like to sincerely express our gratitude to **Mr.SHIVAM GUPTA** and **Ms. APURVA VASHISTHA**, who have been supported through the completion of this project.

We would also be thankful to our principal **Ms. NEERU MEHTA** for providing all the required facilities in completion of this project.

Finally, as one of the team members, I would like to appreciate all my group members for their support and coordination. I hope we will achieve more in our future endeavours.

Gaurav Bhandari

(02024802020)

Unnikrishnan. V

(07324802020)

NARROW LINKS

*--Shorten your link with ease
and share your link via SMS or
scan QR code.--*

Table of content

Introduction.....	5
Why it is needed ?	6
Features and Advantages.....	7
Languages used	8
Apis used.....	11
Home page.....	12
How it works	13
Source code	15
Client side.....	15
Folder structure.....	16
Home page code.....	23
Server side.....	26
Folder structure.....	26
Conclusion.....	32
Bibliography.....	33

INTRODUCTION

This project is an attempt to make a URL Shortener Service(Narrow-Links): it creates a short, unique URL that will redirect to the specific website of the user's choice.

It is secure and has the HTTP protocol and data encryption.

As a result, if someone wants to share a link or include a link in their profile, they can. There are times when that link is too long and takes up the entire space as well as a lot of characters, which is a problem when there are only a limited number of characters to type. As a result, our project solves the problem. We shorten those long URLs and provide a copy button to quickly copy the short URL.

It also has a QR Code Generator, which generates a QR code of the link pasted by the user and redirects to the Website link that the user pasted by scanning this URL.

Another recent addition is a simple one-click button for sending the link to whomever you want in the form of an SMS. Twilio's SMS API is used for this purpose; simply enter the number and hit send, and the shortened link will be sent via SMS to that number.

WHY IT WAS NEEDED?

Many times what happens is we have to share some kind of URL to someone but sometimes the link is too long that it is difficult to share that link. Many social media sites etc have character limits in to be shared and hence if we take a long URL it takes most of the characters and we can't write anything else.

Also if these kind of long URL looks very unclean and dirty like this

https://www.google.com/search?q=happens&sxsrf=AJOqlzUuOabU8OO1bmTyfML2vNsuRcrOXw:1673370081383&source=lnms&tbm=isch&sa=X&ved=2ahUKEwib_Zyqvb38AhVTjuYKHVbcA78Q_AUoAXoECAEQAw&biw=1366&bih=657

As a result, our project provides a facility to shorten these kinds of long URLs into short, easily shareable URLs. Our project also has some extra features like create a QR code of the URL and share the URL as an sms to someone's phone number.

Features and advantages of Narrow links

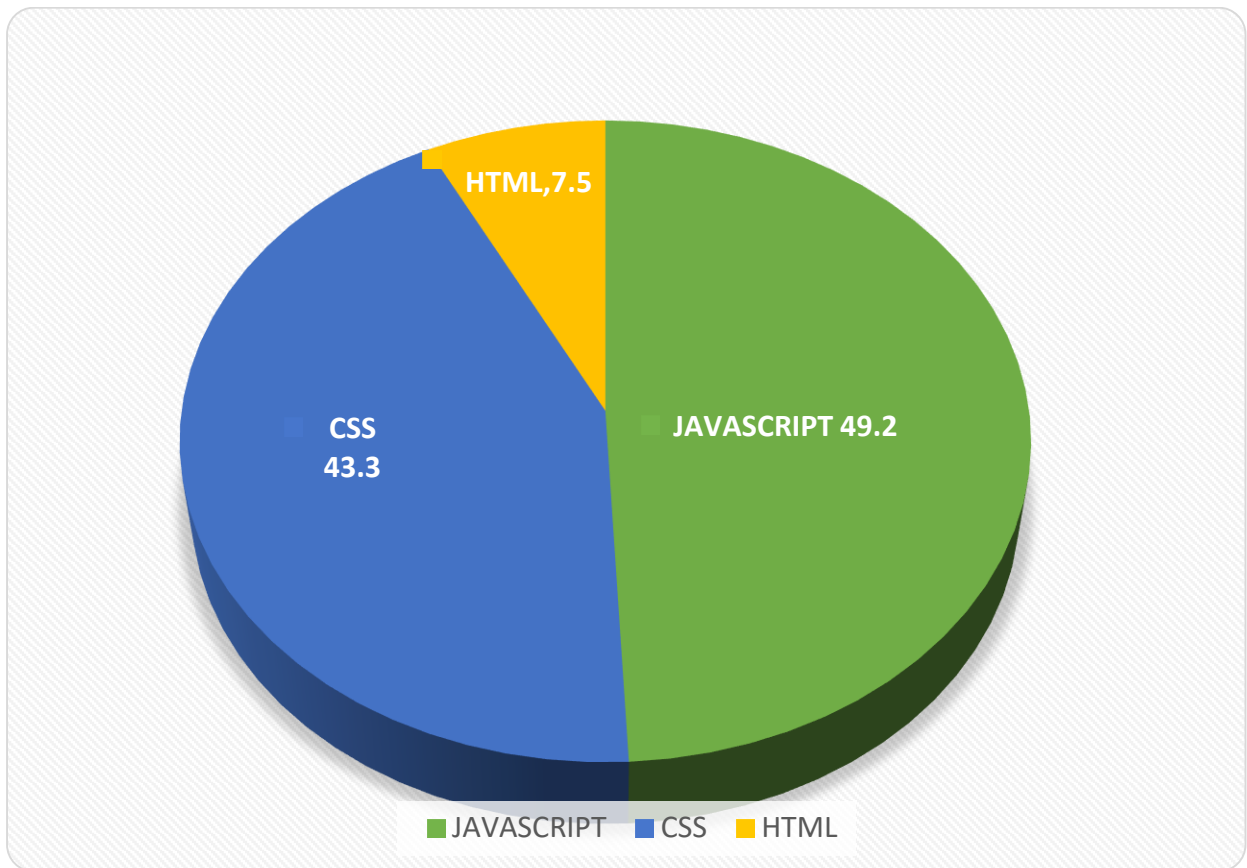
FEATURES

- ❖ It reduces the size and length of the long gusty URLs
- ❖ Easy to share short links with COPY LINK button available.
- ❖ It can also Converts the link in QR code.
- ❖ Links can also be shared via SMS.

ADVANTAGES

- ❖ Shortens your long links address
- ❖ It uses less number of characters
- ❖ Good for memory management
- ❖ QR code feature becomes handy when as we can put the QR code in posters and notices
- ❖ We can easily share the links

LANGUAGES USED



Technologies Used

React.js

The React. js framework is an open-source JavaScript framework and library developed by Facebook.

It's used for building interactive user interfaces and web applications quickly and efficiently with significantly less code than you would with vanilla JavaScript

The website's frontend interface is built using React library. React makes it simple to use component based architecture which makes it easier to design independent components in the frontend

Node.js

Node. js (Node) is an open source, cross-platform runtime environment for executing JavaScript code.

Node is used extensively for server-side programming, making it possible for developers to use JavaScript for client-side and server-side code without needing to learn an additional language.

The website's backend is built using Node.js. Node has many in built functions for doing many backend operations in a simple way. Node.js helps us to understand and code many complex functions in the backend.

Express.js

Express is a node js web application framework that provides broad features for building web and mobile applications.

It is used **to build a single page, multipage, and hybrid web application**. It's a layer built on the top of the Node js that helps manage servers and routes.

MongoDB

MongoDB Atlas is a fully-managed cloud database that handles all the complexity of deploying, managing, and healing your deployments on the cloud service provider of your choice (AWS , Azure, and GCP).

MongoDB Atlas is the best way to deploy, run, and scale MongoDB in the cloud.

Bootstrap

What is Bootstrap used for?

Bootstrap is **a free, open source front-end development framework for the creation of websites and web apps**.

Designed to enable responsive development of mobile-first websites, Bootstrap provides a collection of syntax for template designs.

Most of the frontend components are made using bootstrap templates which makes it easier to design components even faster and customise according to your choice

APIs Used

API (Application Programming Interface)

Application Programming Interface (API) is a software interface that allows two applications to interact with each other without any user intervention. API is a collection of software functions and procedures.

API is defined as a code that helps two different software's to communicate and exchange data with each other.

We have used these APIs in our project some of them are-----

Twilio SMS API

Twilio's SMS API **helps you send and manage messages programmatically**: To send an outbound SMS, WhatsApp, or Channels message with the API, POST to the Message resource. You'll also use the Message resource to fetch messages and list messages associated with your account.

The SMS feature in the website is implemented through this API.

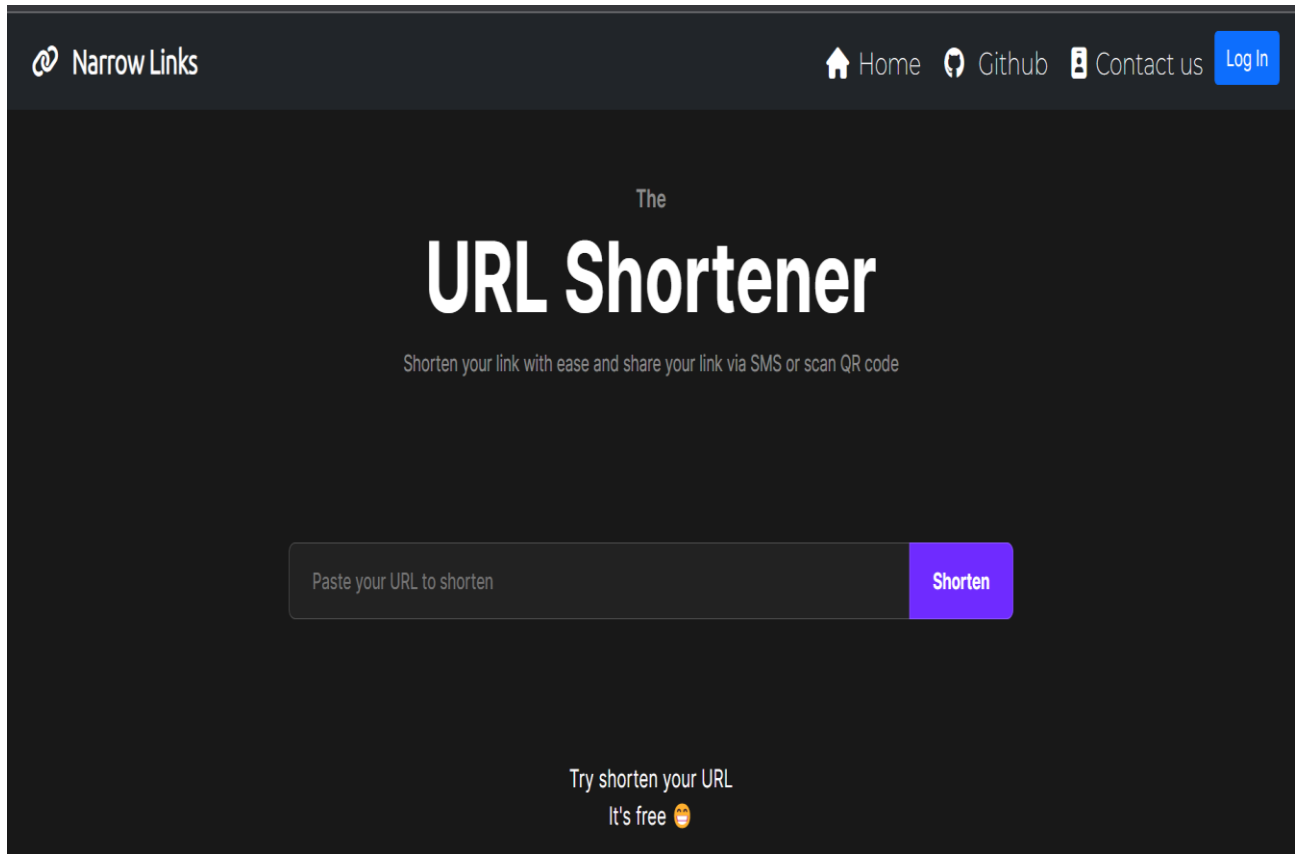
QR Code API

QR Code API is an application programming interface that you can integrate with your app or website to create customized QR Codes of your choice.

The QR code feature in the website is implemented through this API..

HOME PAGE

This is how the home page of our website looks like

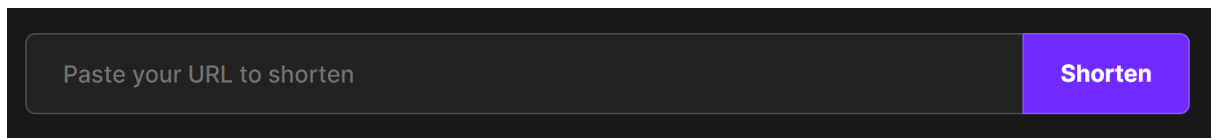


HOW IT WORKS

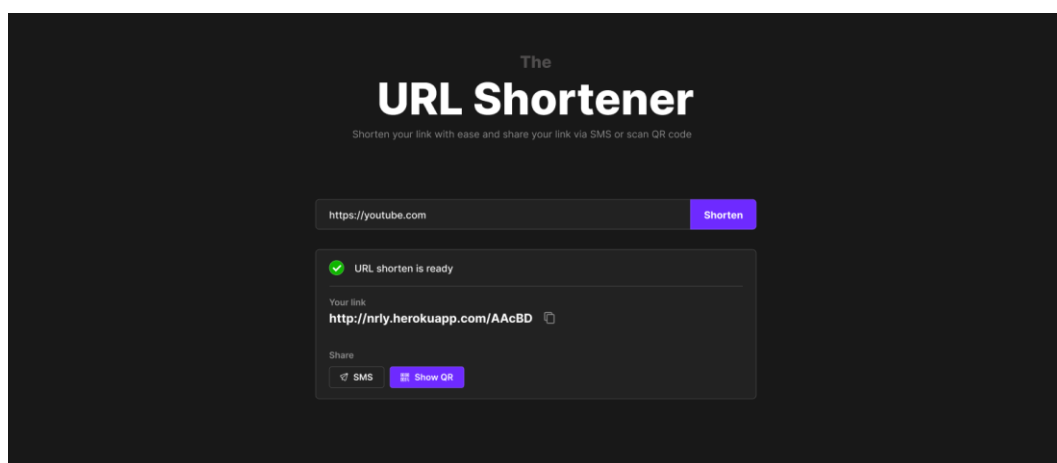
Lets have a look at how narrow links works

--The main objective of the app is to help to reduce long Urls and links into short URLs so that it becomes easier to share it with others.

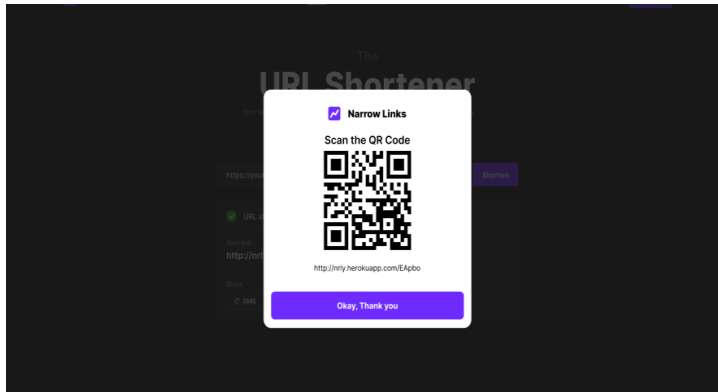
- *Step 1 : Go to the input field and enter any URL you want to shorten.*

A screenshot of the URL Shortener app's input field. It features a dark gray background with a light gray rounded rectangle containing the placeholder text "Paste your URL to shorten". To the right of this rectangle is a purple button with the text "Shorten" in white.

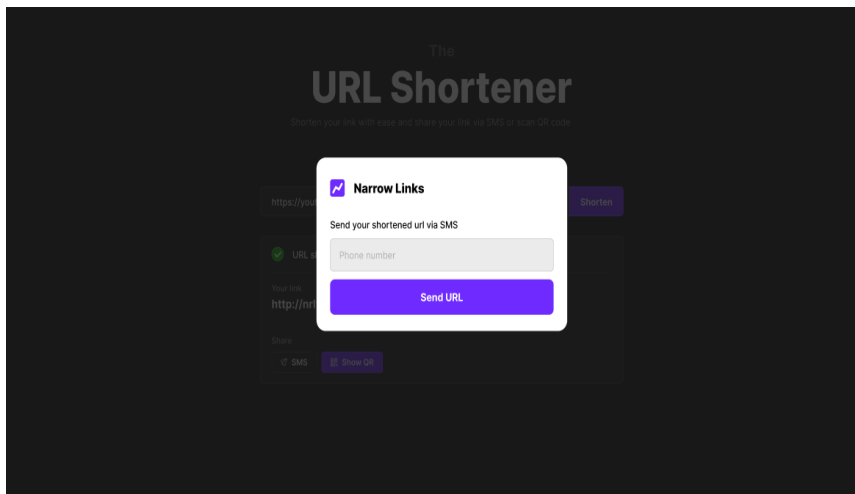
- *Step 2 : Enter the Shorten button to shorten the URL.*
- *Step 3 : The Shortened URL will be now shown below the input field which you can copy and share.*

A screenshot of the URL Shortener app's main interface. At the top, it says "The URL Shortener" in white text, with a subtitle "Shorten your link with ease and share your link via SMS or scan QR code". Below this is an input field containing "https://youtube.com" and a purple "Shorten" button. Underneath the input field, a green checkmark icon is followed by the text "URL shorten is ready". Below this, the "Your link" is displayed as "http://nrly.herokuapp.com/AACBD" with a copy icon. At the bottom, there is a "Share" section with an "SMS" button and a "Show QR" button.

- *Step 4 : You can also generate the QR code for the URL using the Show QR button.*



- *Step 5 : You can also easily share the URL to someone as an sms into their phone number using SMS button below the shortened URL.*



SOURCE CODE

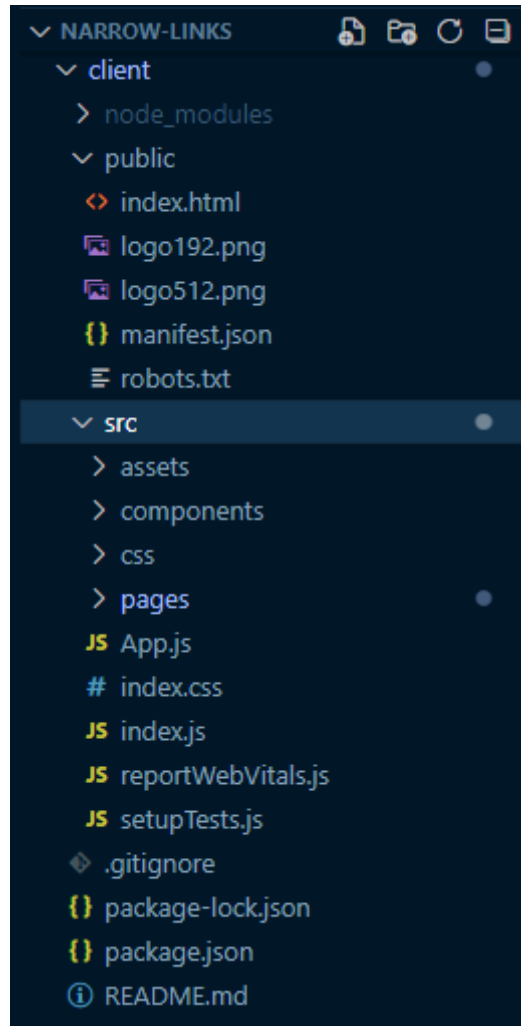
Our project is mainly divided into two sections - Client side and Server side.

Client side

In the client side we have done all the frontend design. All the frontend code elements are in this section and this section communicates with the backend data and populates the data in the frontend. We have used a Javascript framework called React.js to write all the design code. React.js uses components based architecture for the application. React helps in writing the code quickly and efficiently. It has a predefined template and a clear folder structure to be worked upon.

Component based architecture is a process where we create various components in the website and store it in separate files. This method becomes very useful when we have to style every component differently and independently. Every component is stored in separate files for easy access and easy error detection. We can easily find in which files are the error and fix them quickly. All the components are imported to the main page file of the website. Using this approach is also useful when you don't have to write all your code in a single file

FOLDER STRUCTURE



The above image shows the folder structure of the client folder. The client side folder is mainly divided into two subfolders which are public and src. In the public folder, the main file is index.html. index.html is the main entry point of the website where our javascript files are mounted. Now next comes the source folder which is the main section of the client side where all our source code of the frontend is written. In source folder we have 4 subfolders - assets, components, css and pages. The assets folder contains all the custom fonts, images, icons etc. The component folder has all the

components used in the website. Components are independent elements which are stored in separate files for the ease of writing code and accessibility. In the Css folder we have written all the custom css codes of the components and pages. In the last folder which is the pages folder we have stored all the pages in our website in separate files and the components are imported into these pages and shown in the output. Apart from the subfolders you can see some extra files like index.js, app.js etc. These files are nothing but the inbuilt files by React.

CODE

Let's understand the different components of the application first. We have five components in the application which are Navbar, InputURL, Results, ShowQR and ShowSMS

Navbar component

The Navbar component is the header part where all the necessary page links are present like the home page, contact page, github link, login button etc.



The code of the Navbar is given below :

```

1  import React from 'react';
2  import { useAuth0 } from '@auth0/auth0-react';
3  import '../css/navbar.css';
4  import LoginButton from './Login';
5  import LogoutButton from './Logout';
6
7  export const NavBar = () => {
8    const { isAuthenticated } = useAuth0();
9    return (
10     <nav className='navbar navbar-expand-lg navbar-dark bg-dark'>
11       <div className='container-fluid'>
12         <span className='navbar-brand mx-auto' href='#shortener'>
13           &nbsp;<i className='fa-solid fa-link'></i>
14         </span>
15         &nbsp; 
16         <a className='navbar-brand' href='#shortener'>
17           Narrow Links
18         </a>
19         <button
20           className='navbar-toggler'
21           type='button'
22           data-bs-toggle='collapse'
23           data-bs-target='#navbarSupportedContent'
24           aria-controls='navbarSupportedContent'
25           aria-expanded='false'
26           aria-label='Toggle navigation'
27         >
28           <span className='navbar-toggler-icon'></span>

```

```

29     </button>
30     <div className='collapse navbar-collapse' id='navbarSupportedContent'>
31       <ul className='navbar-nav ms-auto'>
32         <li className='nav-item'>
33           <a className='nav-link active' href='/'>
34             <i className='fa-solid fa-house'></i> Home
35           </a>
36         </li>
37         <li className='nav-item'>
38           <a
39             className='nav-link active'
40             target='_blank'
41             rel='noreferrer'
42             href='https://github.com/unnikrishnan2002/Narrow-Links'
43           >
44             <i className='fa-brands fa-github'></i> Github
45           </a>
46         </li>
47         <li className='nav-item'>
48           <a className='nav-link active' href=''>
49             <i className='fa-solid fa-id-badge'></i> Contact us
50           </a>
51         </li>
52         <li className='nav-item log-in'>
53           {isAuthenticated ? <LogoutButton /> : <LoginButton />}
54         </li>
55       </ul>
56     </div>
57   </div>

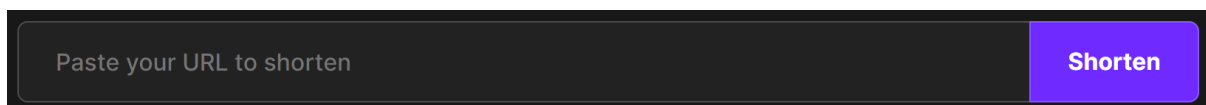
```

We have used Bootstrap Navbar component and then we have added custom css to make it attractive.

The Navbar component has Home page, contact, Github and login links

InputURL component

The Input URL component is the component which takes input URL to be converted.



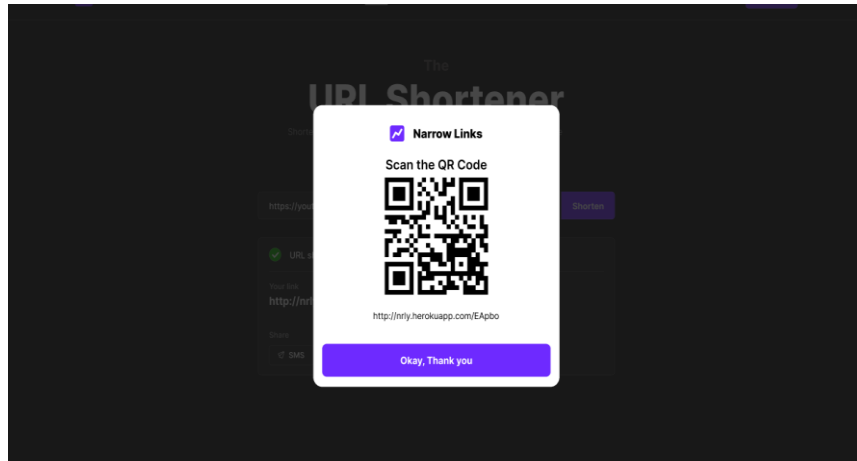
The screenshot shows a dark-themed user interface. On the left, there is a dark gray rounded rectangular input field with the placeholder text "Paste your URL to shorten" in a light gray font. To the right of the input field is a bright purple rectangular button with the word "Shorten" in white text.

The code of the InputURL is given below :

```
1 import '../css/animated.css';
2 import '../css/InputURL.css';
3
4 export default function InputURL({ url, URLInput, HandleEnter, ShortenURL }) {
5   return (
6     <section className='input-container h-25 animated-fadeIn'>
7       <div className='input-form'>
8         <input type='text' aria-label='Paste your URL to shorten' id='inputUrl' placeholder='Paste your URL to shorten' value={url} onChange={URLInput} onKeyDown={HandleEnter} />
9         <button type='submit' onClick={ShortenURL} data-testid='narrow' aria-label='Narrow Link'> Shorten </button>
10      </div>
11    </section>
12  );
13 }
```

ShowQR component

This component helps in generating the QR code from the URL given in the input.



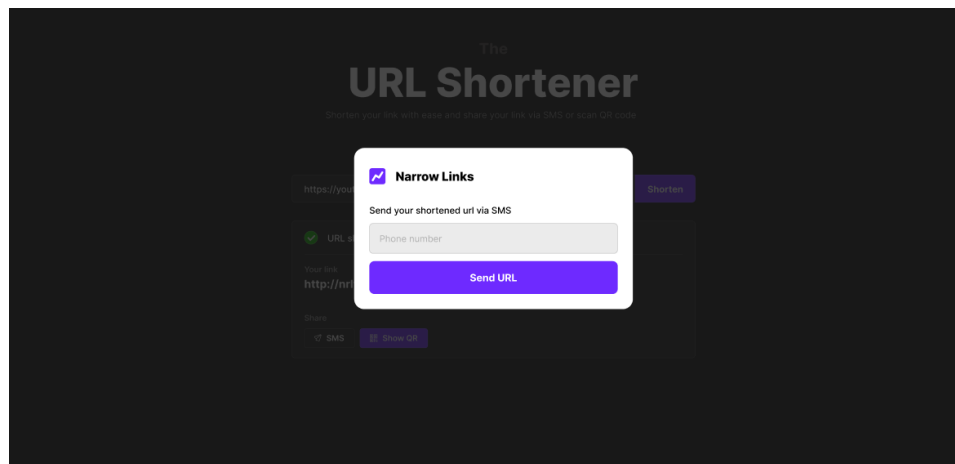
The code for the this component is given below :

```
1  import Logo from '../assets/logo.svg';
2  import '../css/home.css';
3
4  export default function ShowQR(props){
5
6      const { narrowURL, setShowQR } = props;
7
8      function ShowQRCode(){
9          return `https://api.qrserver.com/v1/create-qr-code/?size=600x600&data=${narrowURL}`;
10     }
11
12     return (
13         <section className="qr-container animated-fadeIn">
14             <div className="qr-modal">
15                 <div className="brand">
16                     <img src={Logo} alt="logo" />
17                     <h1>Narrow Links</h1>
18                 </div>
19
20                 <div className="scanQR">
21                     <h1>Scan QR Code</h1>
22                     <img src={ShowQRCode()} alt="QR Code" className="qr-code" />
23                     <p className="url">{narrowURL}</p>
24                 </div>
25
26                 <button className="done" onClick={() => setShowQR(false)}>Okay, Thanks</button>
27             </div>
28         </section>
29     )
30 }
```

We have used QR code api to implement the feature and produce the QR code. Then after creating the QR code the output is returned to the main home page to display the Qr code.

ShowSMS component

This component is used to implement the send SMS feature in the website.



The code for the ShowSMS component is given below :

```

1  import { useState } from 'react';
2  import Logo from '../assets/logo.svg';
3  import LoadingAnimated from '../assets/loading-animated.svg';
4  import '../css/home.css';
5
6  export default function ShowSMS(props){
7
8      const { narrowURL, setShowSMS } = props;
9      const [ phoneNumber, setPhoneNumber ] = useState("");
10     const [ sendProcess, setSendProcess ] = useState(false);
11
12     function InputPhoneNumber(element) {
13         setPhoneNumber(element.target.value);
14     };
15
16     async function SendSMS(e) {
17         e.preventDefault();
18         setSendProcess(true);
19
20         const response = await fetch("https://nrly.herokuapp.com/api/sms/twiliosms", {
21             method: "POST",
22             headers: {
23                 "Content-Type": "application/json"
24             },
25             body: JSON.stringify({
26                 phoneNumber: phoneNumber,
27                 data: narrowURL
28             })

```

```

29     }
30     );
31
32     const SMSRes = await response.json();
33     alert("Message send successfully");
34     setPhoneNumber("");
35     setSendProcess(false);
36     setShowSMS(false);
37 };
38
39 return (
40     <section className="sms-container animated-fadeIn">
41         <div className="sms-modal">
42             <div className="brand-container">
43                 <div className="brand">
44                     <img src={Logo} alt="logo" />
45                     <h1>Narrow Links</h1>
46                 </div>
47
48                 <button onClick={() => setShowSMS(false)}>Close</button>
49             </div>
50
51             <div className="form">
52                 <p>Send your shortened url via SMS</p>
53                 <input type="number" placeholder="Phone Number" className="input-phone" value={phoneNumber} onChange={InputPhoneNumber} />
54                 <button className="done" onClick={SendSMS}>
55                     { !sendProcess && <div>Send URL</div> }
56                     { sendProcess && <img src={LoadingAnimated} className="loading"/> }

```

```

57         </button>
58     </div>
59 </div>
60 </section>
61 )
62 }

```

The component has another input field inside it in which we have to provide the phone number where the sms is to be send. It also has buttons etc.

Home Page code

Now after writing independent components it is time to import all these components to the home page to collectively display all the components.

```

78 <>
79   {showQR && <ShowQR narrowURL={narrowURL} setShowQR={setShowQR} />}
80   {showSMS && <ShowSMS narrowURL={narrowURL} setShowSMS={setShowSMS} />}
81
82   <section id="shortener">
83     <section className="headline flex flex-col items-center justify-center text-center animated-fadeIn">
84       <p className="the">The</p>
85       <h1 className="big-headline">URL Shortener</h1>
86       <p className="description">
87         Shorten your link with ease and share your link via SMS or scan QR
88         code
89       </p>
90     </section>
91
92     <InputURL
93       url={url}
94       URLInput={URLInput}
95       HandleEnter={HandleEnter}
96       ShortenURL={ShortenURL}
97     />
98
99     {status === "idle" && (
100       <section className="idleState animated-fadeIn">
101         <p className="title">
102           Try shorten your URL
103           <br /> It's free 😊
104         </p>
105       </section>

```

```

106     })
107
108     {status === "loading" && (
109         <section className="loadingState animated-fadeIn">
110             <h1 className="title">Shortening your url ...</h1>
111             <img
112                 src={animatedLoading}
113                 alt="loading-animated"
114                 className="loading-animated"
115             />
116         </section>
117     )}
118
119     {status === "done" && (
120         <Result
121             narrowURL={narrowURL}
122             setShowSMS={setShowSMS}
123             setShowQR={setShowQR}
124             isValid={validURL}
125         />
126     )}
127 </section>
128 </>
129 );
130 }

```

Here we have integrated all our components to the html page.

Now for the main operations we have used useStates to store variable data's . useState are a concept of React, this function is used to update a variable data during the runtime.

```

12     const [url, setUrl] = useState("");
13     const [narrowURL, setNarrowURL] = useState("");
14     const [status, setStatus] = useState("idle");
15     const [showQR, setShowQR] = useState(false);
16     const [showSMS, setShowSMS] = useState(false);
17     const [validURL, setValidURL] = useState(true);

```


These are the useStates we have used in the application.

Now we have the main operations of the frontend where we have shortened the Url and displayed it to the user. In this part we are also calling our backend api so that we can shorten the URL and save it in the database.

We are also checking if the URL is valid or not.

```
48     if (!isValidUrl(url)) {
49         setValidURL(false);
50         setStatus("done");
51     } else {
52         const response = await fetch(
53             "https://nrly.herokuapp.com/api/url/narrowurl",
54             {
55                 method: "POST",
56                 headers: { "Content-Type": "application/json" },
57                 body: JSON.stringify({
58                     originalUrl: !url.startsWith("https://") ? "https://" + url : url,
59                 }),
60             }
61         );
62
63         const jsonResponse = await response.json();
64         console.log(jsonResponse.originalUrl, jsonResponse.shortUrl);
65         setUrl(url);
66         setNarrowURL(`http://nrly.herokuapp.com/${jsonResponse.shortUrl}`);
67         setStatus("done");
68         setValidURL("true");
69     }
70 }
```

We are fetching the json response from the backend and then we are setting the data into the useState.

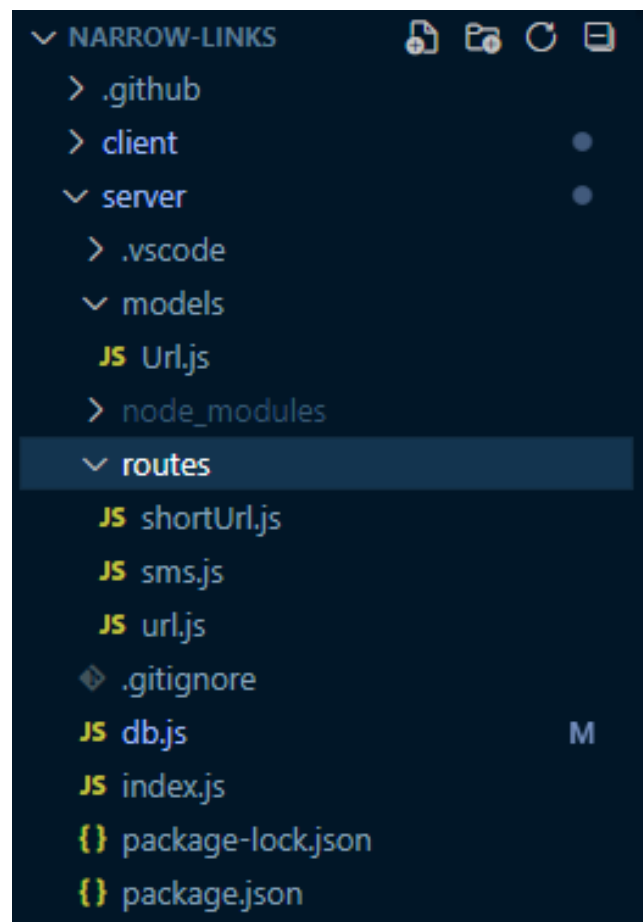
```
35     const isValidUrl = (url) => {
36         var urlCheck = new RegExp(
37             "^(https?:\\/?\\/?)" +
38             "((([a-z\\d]([a-z\\d-]*[a-z\\d])*)\\.)+[a-z]{2,})|" +
39             "([\\d]{1,3}\\.){3}[\\d]{1,3})" +
40             "(\\:\\d+)?(\\/[-a-z\\d%_.~+]*)" +
41             "(\\?[;&a-z\\d%_.~+=-]*)?" +
42             "(\\#[-a-z\\d_]*)?$",
43             "i"
44         );
45         return !!urlCheck.test(url);
46     };
```

Here we are checking the validity of the URL

Server side

In the backend we have written all the logic for shortening the URL, sending the responses, etc. Another javascript framework Node.js is used for the backend. Node.js helps in simplifying many backend concepts using simple functions. In Node.js there is a concept of file based routing which we have used here. We have defined various routes for different functionalities like QR code, Send SMS, and URL shortening.

FOLDER STRUCTURE



The above image shows the folder structure of the server folder. The server side folder is mainly divided into two subfolders which are models and routes. In the models folder, there is Url.js which is the model of our database. Model

is basically a template of json format where we specify how do we want to save the data. Now next comes the routes folder where all the routes are defined . Routes as we discussed earlier are for defining various functionalities in various endpoints. After that we have 2 more files, which are db.js and index.js. index.js is the main file where the execution starts. db.js is the file used to connect to the mongodb database.

CODE

Index.js file

This file is the main entry point of the backend application. In this file we will do all the function imports and route defining

```
1  const connectToDb = require('./db');
2  const express = require('express');
3  const cors = require('cors');
4  require("dotenv").config();
5
6  connectToDb();
7
8  const app = express();
9  const port = process.env.PORT || 5000;
10
11 app.use(cors());
12 app.use(express.json());
13
14 app.use('/api/url', require('./routes/url'));
15 app.use('/', require('./routes/shortUrl'));
16 app.use('/api/sms', require('./routes/sms'));
17
18 app.listen(port, () => {
19   console.log(`Server is running on port: ${port}`);
20 })
```

As you can see, we have imported all the necessary packages and we have defined two routes for shortURL and sms.

After that we have wrote the command to start the server

Db.js file

This file is used to connect the server to the database.

In the below code snippet you can see that we have connected the server to the database using a connection string.

We have also logged a console statement to ensure that the database is connected

```
1  const mongoose = require('mongoose');
2  require('dotenv').config();
3
4  const dbURI = process.env.MONGODBURI;
5
6  const connectToDb = () => {
7    mongoose
8      .connect(dbURI, {
9        useNewUrlParser: true,
10       useUnifiedTopology: true
11      })
12      .then(() => console.log("Database connected!"))
13      .catch(err => console.log(err));
14  }
15
16  module.exports = connectToDb;
```

Url.js (Model file)

As we discussed above this file is for defining the database model. We will define a template for the database and data will be stored in that way only.

```
1  const mongoos = require('mongoose');
2  const Schema = mongoos.Schema;
3
4  const UrlSchema = new Schema({
5    originalUrl: {
6      type: String,
7      required: true
8    },
9
10   shortUrl: {
11     type: String,
12     required: true
13   }
14 });
15
16  module.exports = mongoos.model('Url', UrlSchema);
```

We have given two fields which are the originalUrl and ShortURL

Route files

We have three route files which are shortURL.js, url.js and sms.js

The shortURL.js file where we are actually redirecting the shortened url to original URL.

```
1  const express = require('express');
2  const router = express.Router();
3  const Url = require('../models/Url');
4
5  const host = "http://localhost:5000/"
6
7  router.get('/:shortUrl', async(req, res) => {
8      const url = await Url.findOne({shortUrl: req.params.shortUrl});
9      if (url) {
10
11          const shortUrl = host + url.shortUrl;
12
13          console.log(shortUrl);
14          const updatedUrl = new Url({
15              originalUrl: url.originalUrl,
16              shortUrl: shortUrl
17          });
18
19          updatedUrl.save()
20
21          res.redirect(url.originalUrl);
22          // res.json({newUrl});
23
24          return
25      }else{
26          return res.status(404).json({ error: "The shortUrl does not exist already please try again !!" });
27      }
```

Next file is the url.js file where we are actually creating the shortURL endpoint which is of 5 random character appended at the end of the base URL.

```
1  const express = require('express');
2  const router = express.Router();
3  const Url = require('../models/Url');
4
5
```

```

6  router.post('/narrowurl', async (req, res) => {
7
8      const { originalUrl } = req.body;
9
10     const characters = 'abcdefghijklmnopqrstuvwxyz';
11
12     let shortUrl = '';
13     for (let i = 0; i < 5; i++) {
14         shortUrl += characters.charAt(Math.floor(Math.random() * 10));
15     }
16
17
18     let urlExists = await Url.findOne({ shortUrl });
19     if (urlExists) {
20         return res.status(400).json({ success, error: "Sorry ! A user with the same email exists already" });
21     }
22
23     const url = new Url({
24         originalUrl,
25         shortUrl
26     });
27
28     url.save()
29         .then(() => {
30             res.json(url);
31             // res.redirect({originalUrl});
32         })

```

Here we have written logic for creating a string of random characters which needs to be appended at the end of base url

We are also checking if the URL exist already in the database or not

Then after appending all the string we are saving the URL and then sending it as a response.

Lastly we have sms.js file which is used to carry out the send SMS feature.

```

1  const express = require('express');
2  const router = express.Router();
3  require('dotenv').config();
4
5  router.post('/twiliosms', async (req, res) => {
6
7      const { phoneNumber, data } = req.body;
8
9      const accountSid = process.env.TWILIO_ACC_SID;
10     const authToken = process.env.TWILIO_AUTH_TOKEN;
11     const client = require('twilio')(accountSid, authToken);
12
13     client.messages
14         .create({
15             body: `Hey user, this message is send from Narrow-Links here is the link you shared to your number right now : \n ${data}`,
16             from: process.env.TWILIO_NUMBER,
17             to: `+91${phoneNumber}`
18         })
19         .then(message => console.log(message.sid));
20
21     res.json({ success: "Message send successfully" });
22 })
23
24 module.exports = router;

```

Here we are calling the Twilio SMS api to send the SMS. we are sending a json data body to the API so that they can send the sms in the provided number.

Conclusion

This project was an attempt to create a URL shortening service like Bitly.com and we have tried our best to include all the basic requirements in the website. We constructively divided our project to two different parts - frontend and backend so that we could work on the parts where we had the best knowledge. The logic for the application was mostly done in the backend part and the data was populated to the frontend. We also tried our best to include a wide range of services, technologies and APIs so that we could understand how to work on various tech stack and new technologies. Working with APIs helped us understand many of the backend concepts and methodologies and best coding standards. We created the frontend in such a way that it is easy to understand, and easy to operate. The operations are simple and also we included some additional features like QR code and SMS feature. The project becomes very useful whenever we have to share long gusty URLs in simple format. Thus we have successfully completed our minor project within the given time. Finally we would like to thank all our friends, mentors, teachers and all the online resources who have helped us.

Bibliography

- <https://www.geeksforgeeks.org/reactjs-basics-concepts-complete-reference/>
- <https://www.youtube.com/watch?v=z-Z5radvnOA&list=PLwGdqUZWnOp3aROg4wypcRhZqJG3ajZWJ&index=15>
- <https://www.techtarget.com/searchapparchitecture/application-program-interface-API>
- Advanced web development by Mehul Mohan.
- https://www.youtube.com/watch?v=ExcRbA7fy_A&list=PL4cUxeGkcC9h77dJ-QJlwGlZITd4ecZOA
- <https://www.youtube.com/watch?v=zb3Qk8SG5Ms&list=PL4cUxeGkcC9jsz4LDYc6kv3ymONOKxwBU>
- <https://www.twilio.com/docs/sms/api>
- <https://goqr.me/api/>