How to use the widget:

In your index.html, paste the following,

```
   <div id="busCalculator"></div>
```

Following properties can be changed in regards to styling,
Note that the styles have to be in a separate script tag.

```
   <script>
     let fontFamily = "Times New Roman";
     let backGroundColor = "f5f5f5";
     let backGroundColorTab1 = "e7e7e7";
     let backGroundColorTab2 = "ddd";
     let borderColor1 = "00f";
     let borderColor2 = "00f";
     let buttonColor = "224c98";
     let buttonHoverColor = "4682b4";
     let headingBackgroundColor = "224c98";
   </script>
```

In the backend, you would have to create a route that will host the script.js file in the zip. You would have to have the script.js file in the filesystem and copy the path and paste it in the fs.readfile() method, in the first argument.

```
const http = require('http');
const fs = require('fs');

const server = http.createServer((req, res) => {
   if (req.url === '/script.js') {
       fs.readFile('./script.js', (err, data) => {
           if (err) {
              res.writeHead(500);
              return res.end('Error loading script.js');
           }
           res.writeHead(200, {'Content-Type':
'application/javascript'});
           res.end(data);
       });
   } else {
       res.writeHead(404);
       res.end();
   }
});
```

Now you can use the following line with your api endpoint, in this case it is a http://localhost:8000,

```
   <script src="http://localhost:8000/script.js">
```

In case of customer validation, the server will have some additional code.

```
 <script src="http://localhost:8000/script.js?key=abcd-efg">
```

The key will be taken as a query and the server will send a request to the provided api, and the api will return true or false. If false we will throw an error if true we will continue with our rendering of the widget.