**Description**

In this project I was asked to design and create a URL shortener. I built a full-stack application which allowed users to shorten long URLs so that they can be included in abbreviated links in

character-limited posts, I called the website T1ny URL.

**You will find the deployed app here:** [T1ny URL](https://t1ny-url.netlify.app/)

**Timeframe**

* Timeframe:
  + 5 days

**Technologies used**

* Node.js
* Flask
* PostgreSQL
* Python
* JavaScript
* React
* Axios
* GitHub
* HTML
* CSS

**Brief**

* Have an interface where I can paste/write a URL and process it at the click of a button, so that I can easily provide the URL(s) I want to shorten.
* The shortened URL is returned quickly so that I can use the link elsewhere with minimal delay.
* I can view all URLs previously shortened so that I can quickly reuse a processed URL if need be.
* I can edit and delete previously shortened URLs so that I only have a list of active shortened URL’s
* I can see data on how often each shortened URL has been clicked so that I can evaluate how often each shortened URL is used.
* I am redirected to the original URL when I paste the shortened URL in my browser so that I can utilise a shorter, working URL shortened URLs.

**Planning**

I did some research into how sites like Bitly work, I found an [article](https://stackoverflow.com/questions/4572734/how-does-a-url-shortener-work) on Stack Overflow which explained in detail the logic involved for shortener URL’s.

Using my research, I planned out the basic logic for my application:

* When a full URL was inputted, a post http method containing the full URL would be sent to my API.
* A record would be created in the database. The full URL would be added, and a 6-digit number would be automatically added to that database entry.
* I would also add a clicks column in the database which would be set as 0 for each new URL.
* The full URL can be accessed using the shortened link by typing in the hostname for my site and the 6-digit code, for example, hostname/645875.
* A get http method containing the 6-digit code will be sent to my API and the full URL associated with the 6-digit code will be returned, the user will then be redirected to that site.

Once I had worked out the logic that I wanted to implement for my URL shortener, I thought about how my site would look, I created a wireframe for my homepage.

*Homepage*

Text, letter

Description automatically generated

I also created a Jira board for the project, this helped me to keep track of all outstanding tasks.

Graphical user interface, application, Teams

Description automatically generated

**Build Code Process**

Technical Setup and Back-end (days 1 to 2)

For the technical setup I installed all the dependencies required for the front-end and back-end, including, flask-sqlalchemy, flask\_cors, flask\_marshmallow, flask, React and react-router-dom.

The back-end was created using Python, I used the Django web framework with a PostgreSQL database, in total I had 2 models to create. Initially, I created the base model, which was included in in my links model, this allocated each entry a unique ID and noted when the entry was created and updated, I also added in two methods which allowed me to save or remove a model object.

I then created the link model, which defines the structure of the SQL database and the serializer, which uses marshmallow to convert data.

The link controller file contains a set of functions, one for each route that they are required to handle, for example, my POST /links route will create a new record in the database and add a random 6-digit code to the record.

Once everything had been created, I began to test each of the endpoints using Insomnia, to confirm I was receiving the responses that I expected.

***Featured code: Links model and controller***

The link model was very simple, it contained columns which contain the full URL, the 6-digit code (short) and the number of clicks, which as default is set to 0.

***A screenshot of a computer

Description automatically generated***

***post\_new\_link function***

* Creates a new record in the database with the new full URL.
* In an infinite while loop, a random 6-digit number is created, then the database is checked in case this 6-digit number has already been associated with an existing URL.
* If it hasn’t then the 6-digit number is added to the object’s short column and the database object is saved, converted to JSON and returned.
* If the 6-digit number has been used before we loop round again with a new random number until it finds a 6-digit number that hasn’t been used before.

Text

Description automatically generated

Front-End (days 3-4)

React was used for the front-end; the useState and useEffect React hooks were utilised. I used Axios for http requests to my API.

Three components were created for the front-end:

***Home Component***

The Home component contains the input where a user posts their URL to be shortened and the table including the headers

A picture containing bar chart

Description automatically generated

I used useEffect to manage the side effects in this component, the getData function gets an array of all links that have already been shortened when the component first renders, anytime that a link is created, clicked on, deleted or edited an updated array of shortened links is requested from the back-end, this is due to the dependencies that have been set in the useEffect hook.



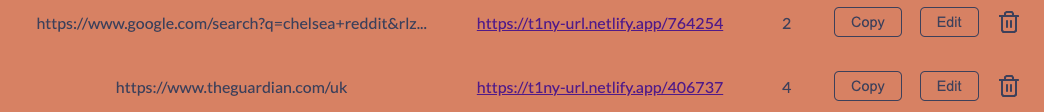
The links array is then iterated through using the map method and props are sent to the LinkData child component which displays the links that have already been shortened.

Graphical user interface, text, application, chat or text message

Description automatically generated

***LinkData Component***

The LinkData component adds links that have been shortened to the table, it also allows the user to copy, edit and delete the link.



If you click on the edit button it sets the showEditInput state as true and will then show the edit input section, this can be hidden again if you press the cancel button.

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated



***Redirect Component***

Finally, the Redirect component is called when a user inputs a shortened URL, for example, <https://t1ny-url.netlify.app/406737> into a browser, the component then sends a get http request to the API which returns the full URL and the component re-directs the user to that address.

Text

Description automatically generated

Front-End (day 5)

I styled the site using CSS, I centred the table using Flexbox and used the Lato font from Google Fonts. I used [coolers.co](https://coolors.co/) to pick a colour scheme, I spent too long trying different colour schemes but was happy with the colours I implemented in the end.

**Key Learnings**

* I learnt how a URL shortener works; I initially thought it would be more complicated than it was. I felt that by doing research and making a detailed plan it allowed me to achieve my targets with little complication.

**Bugs**

* When I click on a shortened URL instead of opening in a new tab it navigates to the webpage, leaving <https://t1ny-url.netlify.app>.

**Future Improvements**

* Make the site responsive.
* Add error handling, for example, I would like my application to display a message if a link has already been shortened.
* Check that the link being inputted into our input box is valid.
* Add a header and a t1ny-url logo.