

1time2fa

≡ Menu

One time Two Factor Authentication

Background & Problem Description

Most people now access all the important areas of their life—banking, shopping, insurance, medical records, and so on—simply by sitting at their computer and typing a username and password into a website. In theory, this kind of protection should be reasonably secure; but in practice, it's less trustworthy.

Nowadays, insecure WiFi connections can collect what you write into your browser, hackers can perform man-in-the-middle, dictionary attacks, etc. The security risk is greater when people use the same password for multiple services and websites, so if their password is stolen, they could have not one, but multiple accounts compromised.

On the other hand, there's a risk if you use cyber cafes or computers at colleges or other public places, that you forget to log out or you could unconsciously store your password on a machine someone else will use.

Banks and organizations are aware of cybercrime and sometimes require you enter not only your username and password, but also your birthday, the town you were born, or any personal piece of information. However, it doesn't overcome the mentioned problems attacks since those answers can get easily stolen by insecure Wi-Fi connections or man-in-the-middle and dictionary attacks.

The problem with two factor authentication and one time passwords by themselves is that they do not provide a full level of security to a user's account. The problem with two factor authentication is that, if someone gets your username or password, which are required for the two factor process, they will be able to access your account if you decide to deactivate two-factor authentication system. The problem with one-time password is that this removes the requirement of knowing a password, so if someone gets access to your phone or email address, they will get this PIN and the intended security is broken.

Goal

The goal of our project is to develop an alternative, more secure login method which allows users to use the one-time two-factor authentication for logging in to their account. This will ensure that no one will be able to access your account without 1) knowledge of your password and 2) access to your physical phone.

Solution (Non-Technical)

Description of our solution

Our solution is a one-time two-factor authentication system that, instead of requesting the entire password, only asks for the last 3 characters of the password and the username. Entering the 3 last digits is an advantage because even if a hacker gets what you wrote on the keyboard (using the man-in-the-middle attack, for example), they still won't know your full password. It also involves sending a code via SMS to users to allow them to access their accounts via two-factor authentication. This way, if someone gets access through the 2FA, they still will need access to your mobile phone, which they, in theory, won't have.

One functionality we have added is that the user can ask for a new code to be sent via SMS as many times as he/she wants. However, if the user does not enter the code in 300 seconds after logging in, he/she will be exited from the page (for security reasons) and will have to login again. We have provided functionality to allow the user to register on the same website as well.

Relevant features

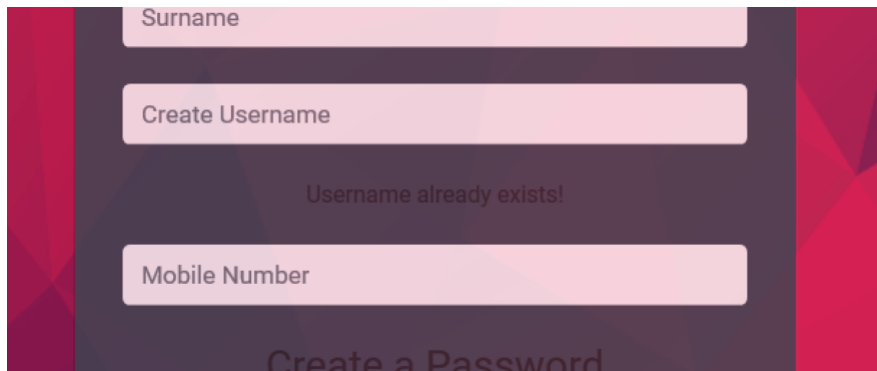
The landing page is a regular login page (that requests username and password) in which the user has two options: one for accessing through a two-factor authentication login and another one for registering, in case they don't already have an account. It looks like this:

A login form on a dark purple background with a red geometric pattern. At the top is a white square containing a black silhouette of a person's head and shoulders. Below this are two white input fields: the first is labeled 'Enter Username' with a small person icon, and the second is labeled 'Enter Password' with a small lock icon. Below the password field is a red button with a white right-pointing arrow and the text 'Log in'. At the bottom of the form are two blue links: 'Create an Account' and 'Log in using 2FA'.

Registration page requires the user creates a username and password and that he/she introduces his/her phone number:

A 'Create Account' form on a dark purple background with a red geometric pattern. The title 'Create Account' is in white. Below it is the section 'Enter Your Details' in white. This section contains four white input fields: 'First Name', 'Surname', 'Create Username', and 'Mobile Number'. Below these is the section 'Create a Password' in white, which contains two white input fields: 'Enter Password' and 'Confirm Password'. At the bottom of the form is a red button with a white right-pointing arrow and the text 'Submit'. Below the button is a blue link: 'Log into Existing Account'.

If username already exists, the next message will be shown:



Surname

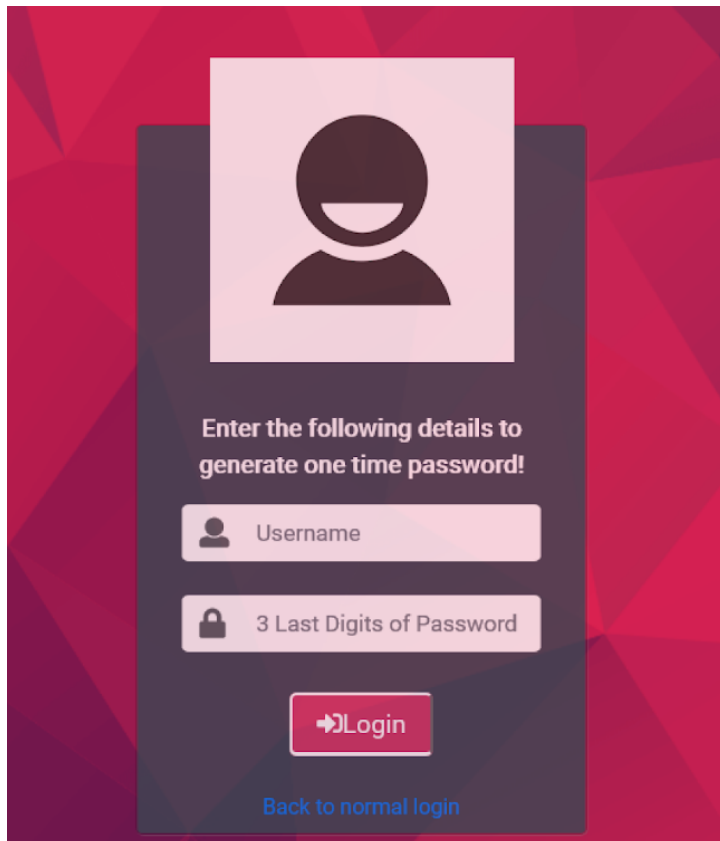
Create Username

Username already exists!

Mobile Number

Create a Password

If user chooses the two-factor authentication login, this will appear:



Enter the following details to generate one time password!

Username

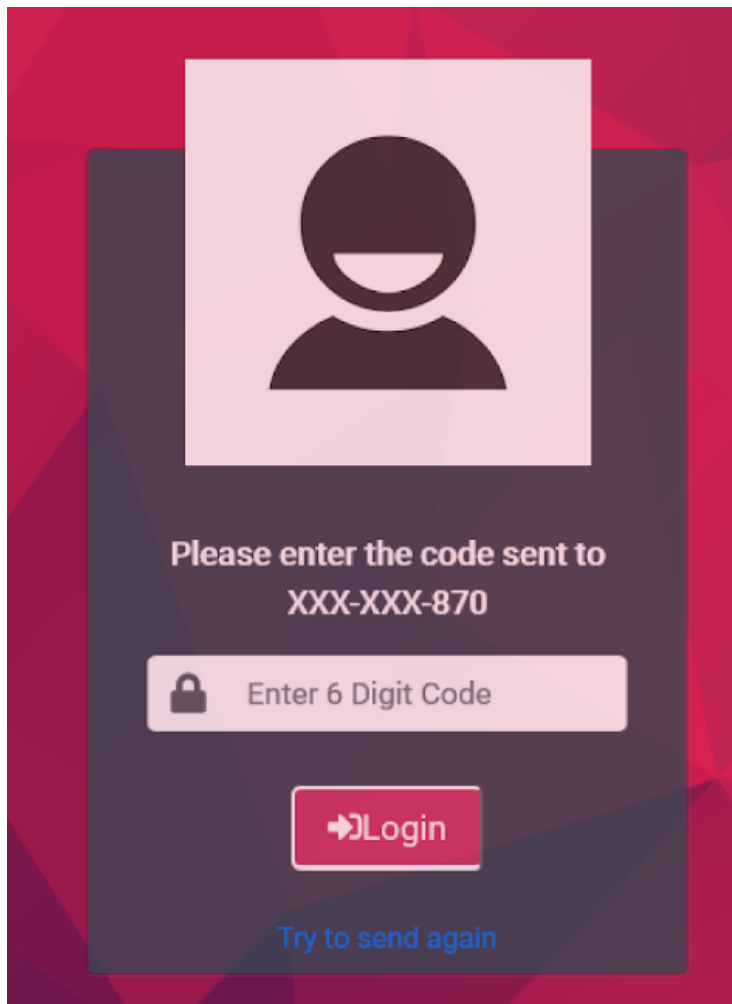
3 Last Digits of Password

Login

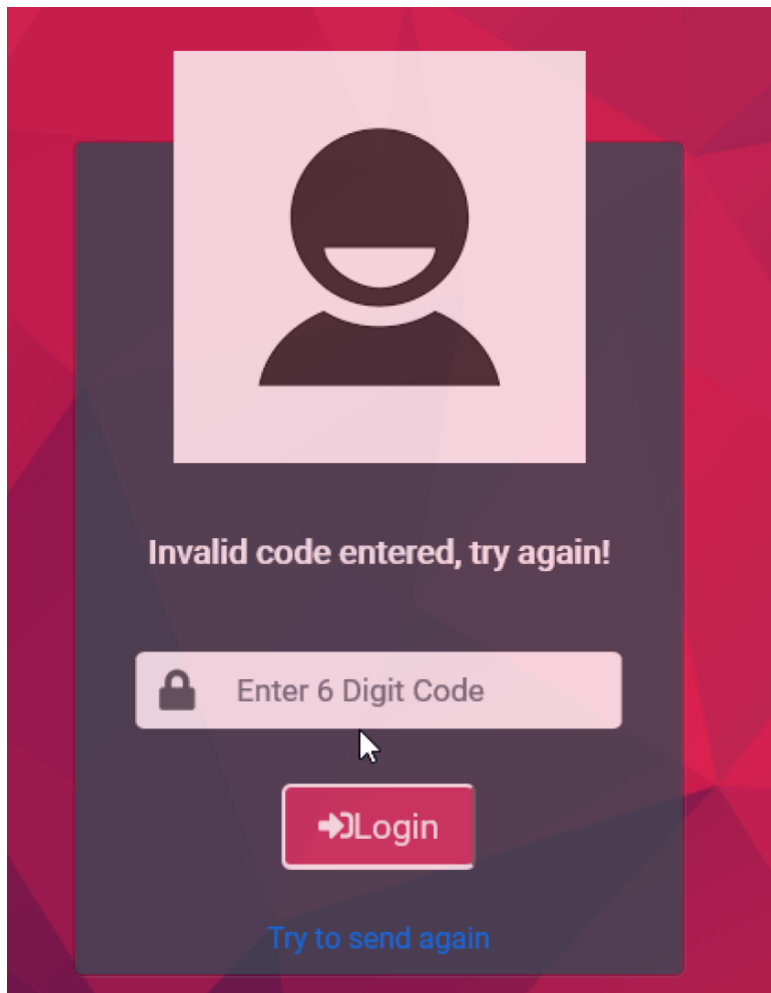
[Back to normal login](#)

But they have the option to go back to regular login.

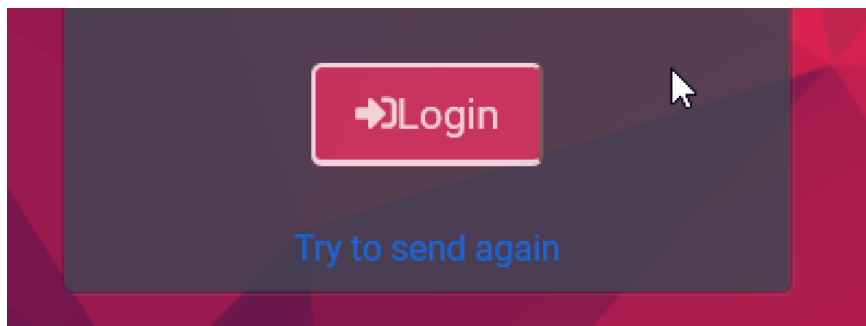
Once they have logged in with their username and three last characters, this page will be displayed, asking for a code:



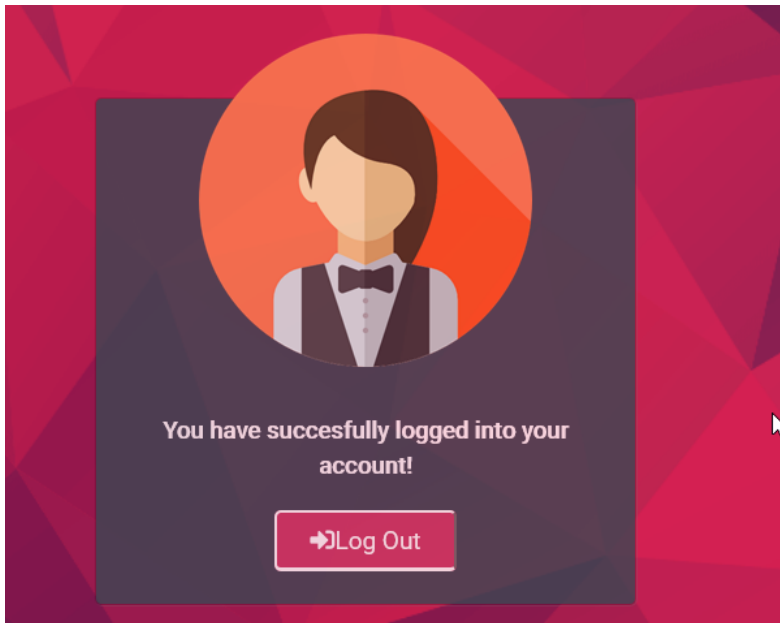
If user doesn't enter the correct PIN, this message will appear:



However, if he/she struggles for entering the code, the user can ask for a new one.



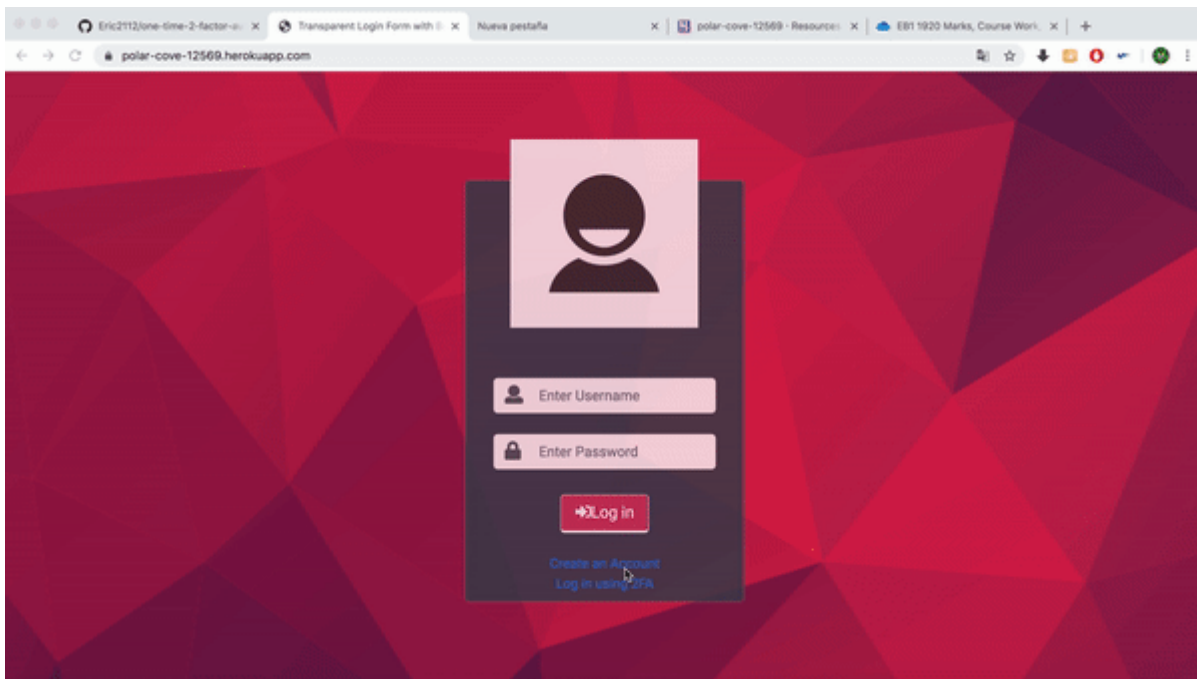
The final page is accessed when correct PIN is introduced:



Link to our demo: <https://polar-cove-12569.herokuapp.com/> (<https://polar-cove-12569.herokuapp.com/>).

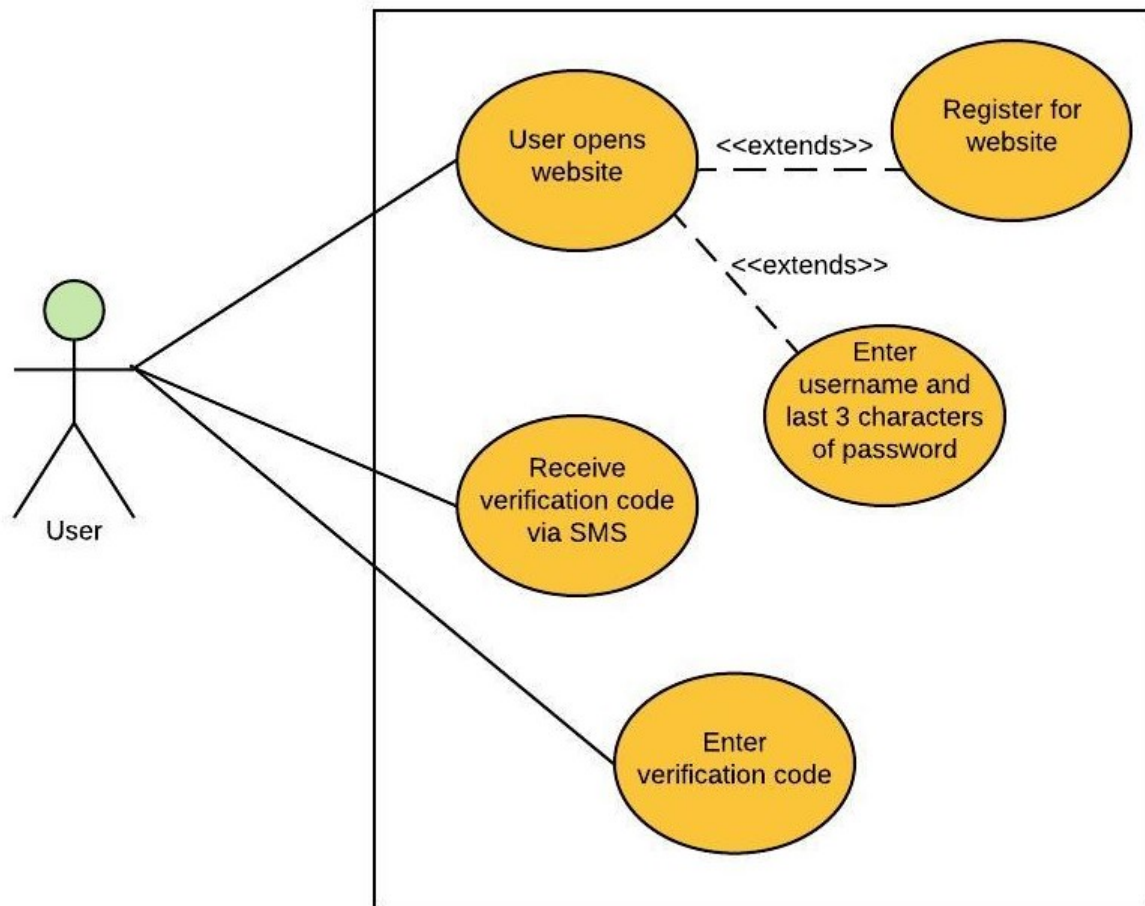
Here is a link to the repository that contains the source code for our project:
<https://github.com/Eric2112/one-time-2-factor-authentication> (<https://github.com/Eric2112/one-time-2-factor-authentication>).

Here is a video which shows our project in action:



Solution (Technical)

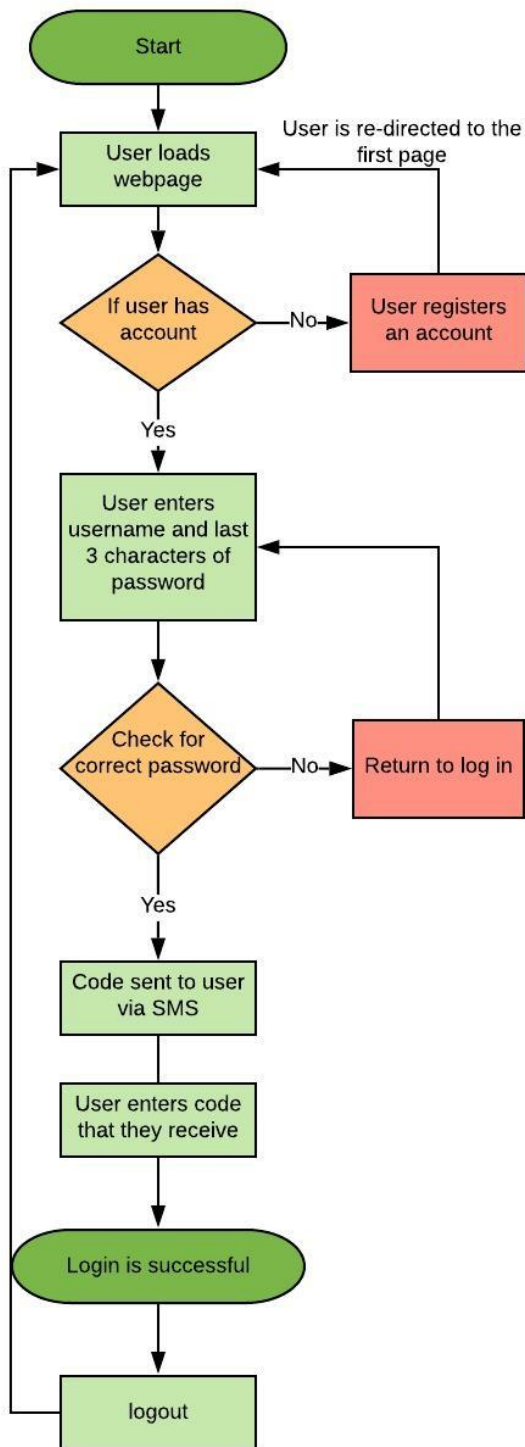
Use Case Diagram



Description:

The main purpose of this use case diagram is to show how the users will interact with our prototype. It contains the main methods that the user will participate in. The only actor for this use case is the user. The first method the user carries out is opening the website. This method is extended by either registering in the website or entering with the username and last 3 characters of password. Another method is going to be receiving a verification code via SMS. The user will check their phone for the code which they have received via text. This way, the last method is entering the verification code.

Flowchart Diagram

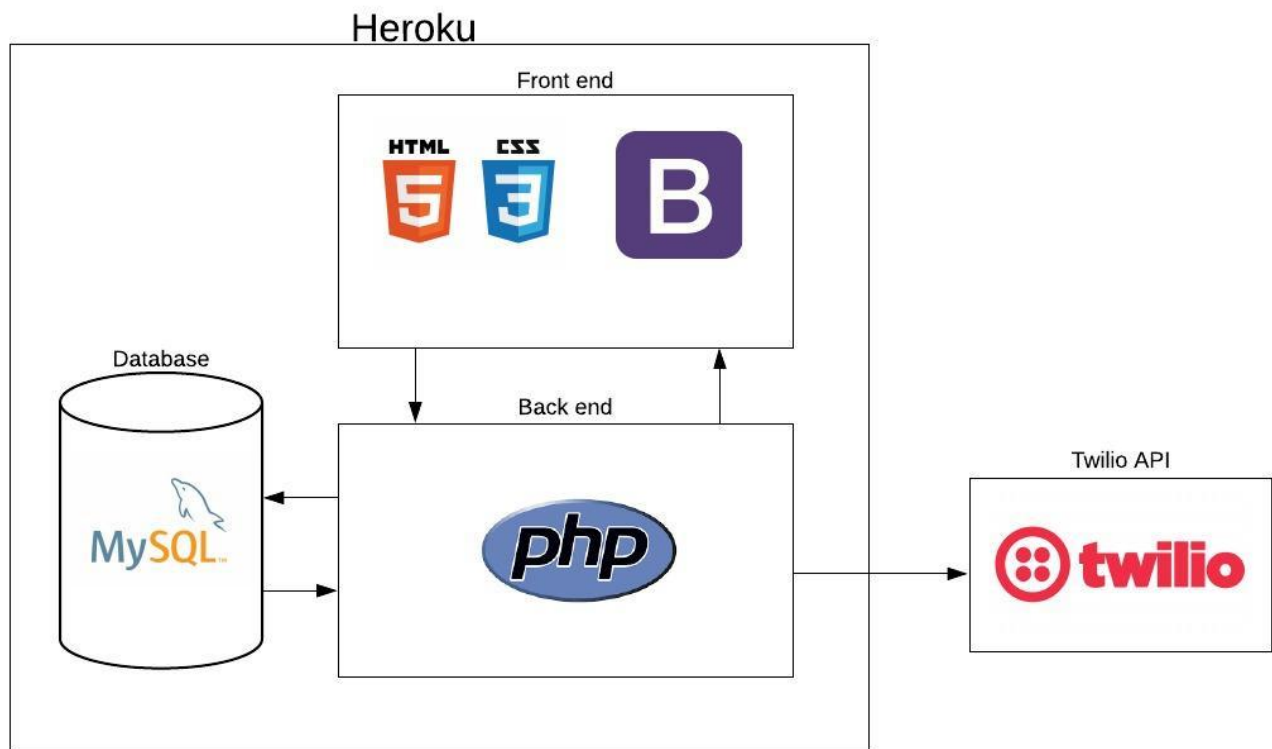


Description:

This flowchart shows the main flow of operations in our program for a user wishing to log in to their account with one-time two-factor authentication. The first step is that the user loads the webpage. If the user does not have an account they will need to register first to be able to log in using two-factor authentication. If the user has an account they need to enter their username and the last 3 characters of their password. The system will check if the correct password has been entered and, if it has not, the user will be returned to the login screen, displaying a message that informs the details were not well provided. If the password is correct the user will

be sent a code via SMS to their phone. When they enter the code they will have logged in successfully. Finally, there is the log out option that redirects the user back to the first login page.

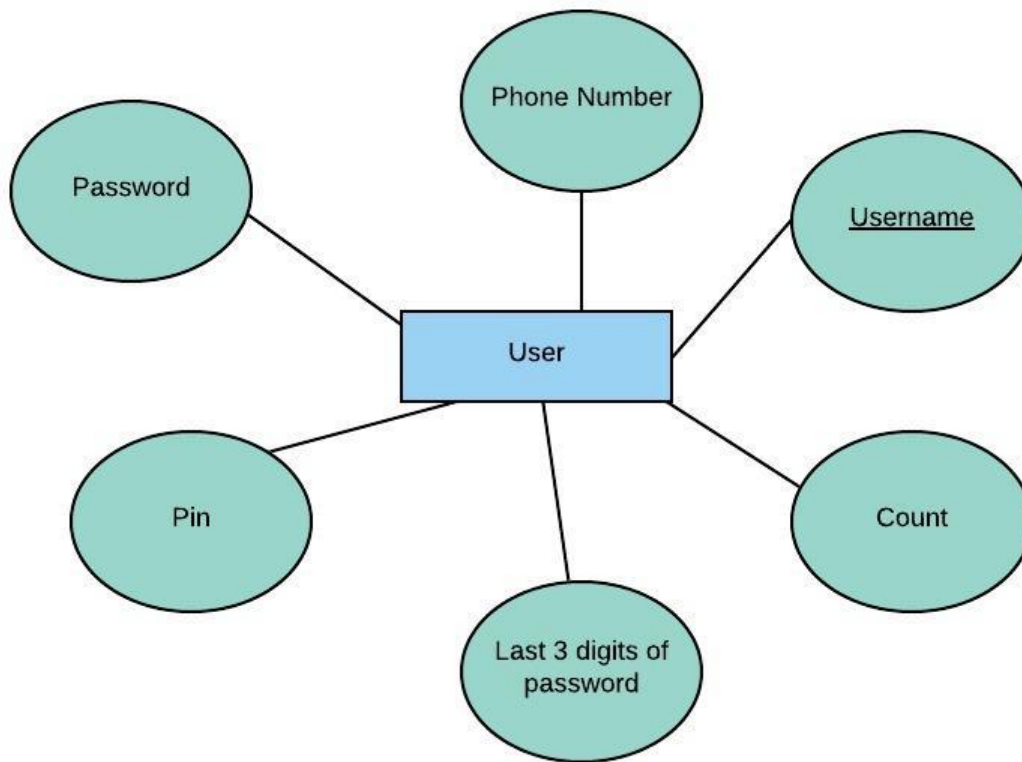
Architecture Diagram



Description:

This architecture diagram contains the main components that are in our program and how they run together. The cloud platform Heroku is being used to host our project, thus it contains the front end, back end and database. The front end sends and receives data to and from the back end. It is developed with HTML, CSS and Bootstrap. The back end receives input from the front end and sends data to the front end and it can access the database. It is developed with PHP. The database is stored on clearDB. The database can send data to the back end. MySQL is used for managing the database. The Twilio API is accessed by the back end to send the SMS to the user.

Entity Relationship Diagram



Description:

This diagram shows the main attributes that make up the user entity in our database. The primary key is the username attribute. Firstly, the phone attribute is used to store the user's phone number. The password attribute is used to store the full length password that the user has entered. The PIN field is used to store the verification code which is sent to the user's phone. The last 3 digits of password attribute is used to store the last 3 characters of password that is going to be checked for the two factor authentication. Finally, the count attribute is used to check whether a user has requested a code. If count = 1 they will receive a code, if count is =0 the user will be ignored.

Description of technical solution in detail

The programming languages we have used to develop this project are HTML, CSS and PHP. For managing the database we decided to use MySQL because it was easier to be integrated with the PHP code.

For the front end of our project, the Bootstrap library was used to help us design the user interface for the project. This library has given our HTML pages more structure and a more uniform design for the appearance of our project. Bootstrap also helps the pages to appear the same across multiple browsers.

To send the message over SMS we have decided to use the Twilio API. Twilio is a service which allows sending an SMS to a chosen users phone. We decided to generate the random code on our server, as it gives us more control over how it is created and stored in our project. The

Twilio API receives an HTTP request from our server which then triggers it to send the SMS message we require to be sent. To use the API token they need to be kept in environment variables so people cannot see our private details for our account.

To get this API to work we needed to use some libraries. We decided to use the composer tool for dependency management of this API as it is written in PHP. This allows us to call libraries needed to run the Twilio API into our project and it also updates and installs these libraries as required, saving us the task of manually loading each library every time we want to run the project. The composer files autoload all of the code necessary to run the Twilio software development kit.

We are hosting our website on the cloud platform Heroku. This permits us to keep the front end, back end and database together in one location and the overall project together. clearDB hosts the database on Heroku.

Limitations & Outlook

This is only the first version of the project. For future versions, we would improve:

- When logging in, the user could watch his/her name displayed in the successful page saying "Welcome NAME SURNAME, you've successfully logged into your account".
- Even though this system is for secure login, also allowing the regular login (with username and the full password).
- Right now, if the user goes back from the page that requests the PIN and another user tries to log in, an error will occur because the other user didn't properly "log out".
- Sending an SMS to the user once he/she has registered, to verify the information he/she entered and that he/she has access to the phone number they introduced.
- Unfortunately, we were not able to get the Twilio API to send SMS with the Heroku cloud application program, we could only get it to send SMS with the local host

Advertisements



Earn money off your WordPress site

WordAds

Monetize your WordPress blog!

WordAds

LEARN MORE

REPORT THIS AD

REPORT THIS AD

☐ ericted . Uncategorized ☐ Leave a comment ☐ November 20, 2019November
21, 2019 ☐ 7 Minutes

[Blog at WordPress.com.](#)