

Enter-connectivity

Preserving Relationships



November 18, 2023

Lawrence Technological University

By: Milan Jostes and Roman Higginson

Table of Contents

[Introduction 2](#_Toc152679696)

[Basic Operation 2](#_Toc152679697)

[Server 2](#_Toc152679698)

[Client 2](#_Toc152679699)

[Design Diagram 3](#_Toc152679700)

[Coding 4](#_Toc152679701)

[Server Side 4](#_Toc152679702)

[Contact.js 4](#_Toc152679703)

[Read-write.js 5](#_Toc152679704)

[Server-commands.js 6](#_Toc152679705)

[Client Side 6](#_Toc152679706)

[Add.html 6](#_Toc152679707)

[Contacts.html 6](#_Toc152679708)

[Edit.html 6](#_Toc152679709)

[Header.html 6](#_Toc152679710)

[Index.html 6](#_Toc152679711)

[Log-in.html 7](#_Toc152679712)

[Profile.html 7](#_Toc152679713)

[Sign-up.html 7](#_Toc152679714)

[Demonstration 8](#_Toc152679715)

[Difficulties 8](#_Toc152679716)

[Future Additions 8](#_Toc152679717)

[Conclusion 8](#_Toc152679718)

# Introduction

Enter-connectivity is a program that was developed to promote the continued strengthening of relationships. A user will be able to keep track of all the people they wish to interact with on a consistent basis. The program will update the user on birthdays, which people to contact, and how long it has been since the two individuals last interacted. The aim of this project is to help people stay in contact with their friends and family. The server holds all of the data for the users, so that they can access it from anywhere or on any device.

# Basic Operation

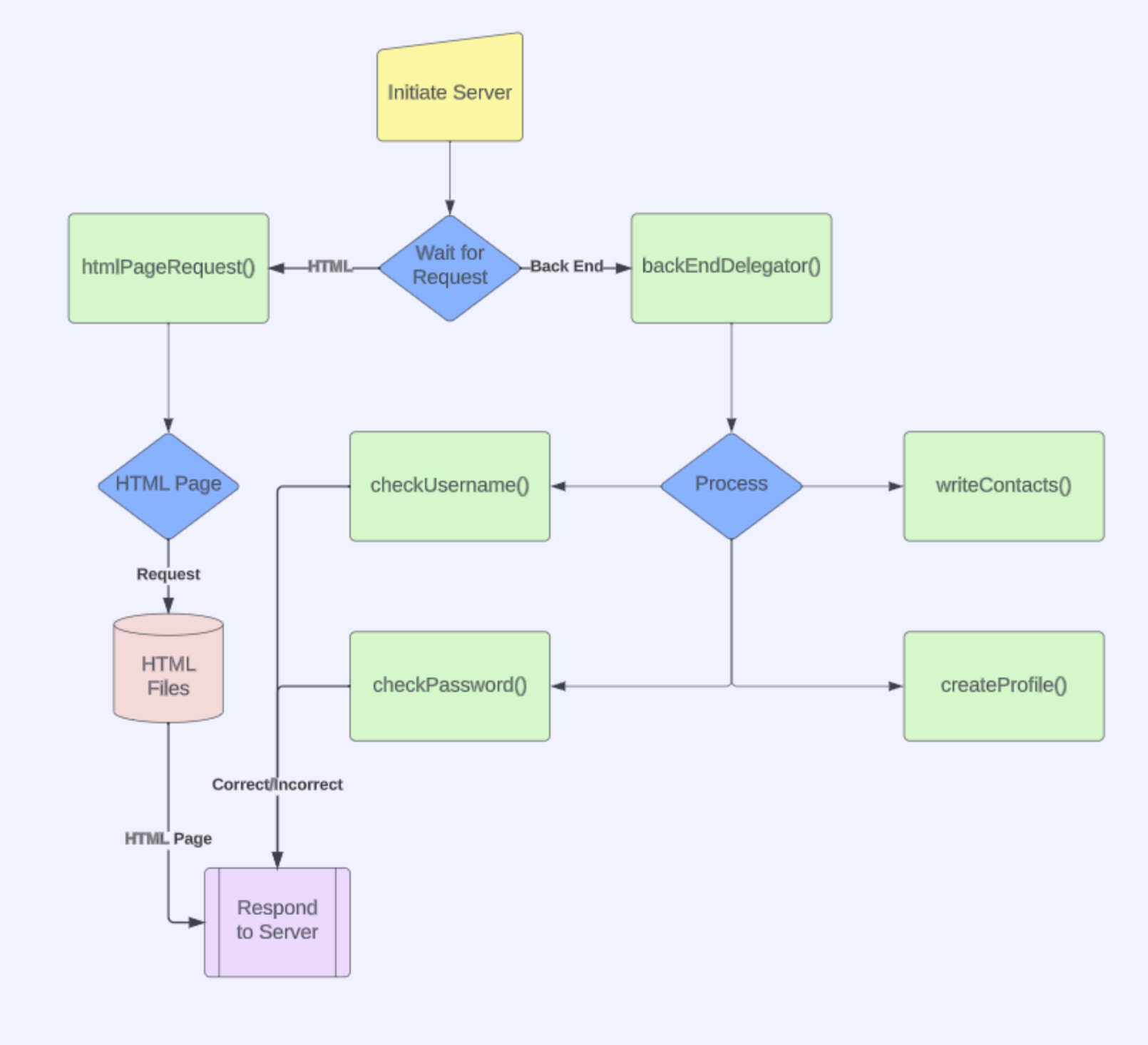
Enter-connectivity works in two segments, the Server side and Client side. The server deals with the logic of the program, reading and writing data, creating contact data variables, and storing of the information. The client side, written primarily in HTML, provides the User Interface of the program.

## Server

The server

## Client

# Design Diagram



# Coding

There are quite a few files used in this project, split into different files for the convenience of modularization.

## Server Side

### Contact.js

* contactCreate()
  + Parameters: Takes in a name, birthday, email, phone number, data of last contact, and how often the user would like to interact with that individual.
  + Function: Creates a contact variable that works similar to an object. It helps standardize how information is worked with and stored for ease of use.
  + Returns: A dictionary variable that contains all of the contacts information.
* dateToText()
  + Parameters: Takes in a date variable.
  + Function: Selects specific information from the date variable and reformats it into a more easily used format.
  + Returns: A string variable version of the date.
* textToDate()
  + Parameters: Takes in a string variable.
  + Function: Reformats the string date into a date variable.
  + Returns: A date variable.
* textContact()
  + Parameters: A dictionary variable for a contact.
  + Function: Using the input of the dictionary, it reformats the contact data into a string variable that can be used for writing contacts to storage in the read-write.js file.
  + Returns: A string version of the dictionary.
* separateContacts()
  + Parameters: An array of strings that are used to represent contacts and their information.
  + Function: The function will separate the list of strings into the individual contacts and their information. This takes the output from readContacts() in the read-write.js file and breaks the array of strings back into contacts so that the data can be handled in a reasonable manner.
  + Returns: An array of dictionaries the represent contacts and their information.
* outputContacts()
  + Parameters: An array of dictionaries that represent contacts and their information.
  + Function: Prints out all of the information to the console. Primarily used by server admins to debug and test the reading, writing, and other capabilities of the server.
  + Returns: Nothing.

### 

### Read-write.js

* readContacts()
  + Parameters: Takes in two strings, a username and a password.
  + Function: Takes the username to identify the correct file to open. It then checks whether the password is correct. If it is it will read all of the information in the file into an array of strings that can be used to load the contacts.
  + Returns: An array of strings that correlate to contact information.
* writeContacts()
  + Parameters: Takes in two strings, username and password, and an array that contains contact data.
  + Function: The function checks whether the user has a file and attempts to delete it. It then creates the file and writes the specified data to it.
  + Returns: Nothing.
* addContact()
  + Parameters: Takes in two strings, username and password, and a JSON contact variable.
  + Function: The function adds a new contact to the end of the user’s data file.
  + Returns: Nothing.
* editContact()
  + Parameters: Takes in two strings, username and password, an integer id and a JSON for a new contact.
  + Function: Gets the contact information from the file using the username/password and the readContacts() function. It then edits the information of the specified contacts and uses the writeContacts() function to write the data back to the file.
  + Returns: Nothing.

### Server-commands.js

## Client Side

### Add.html

### Contacts.html

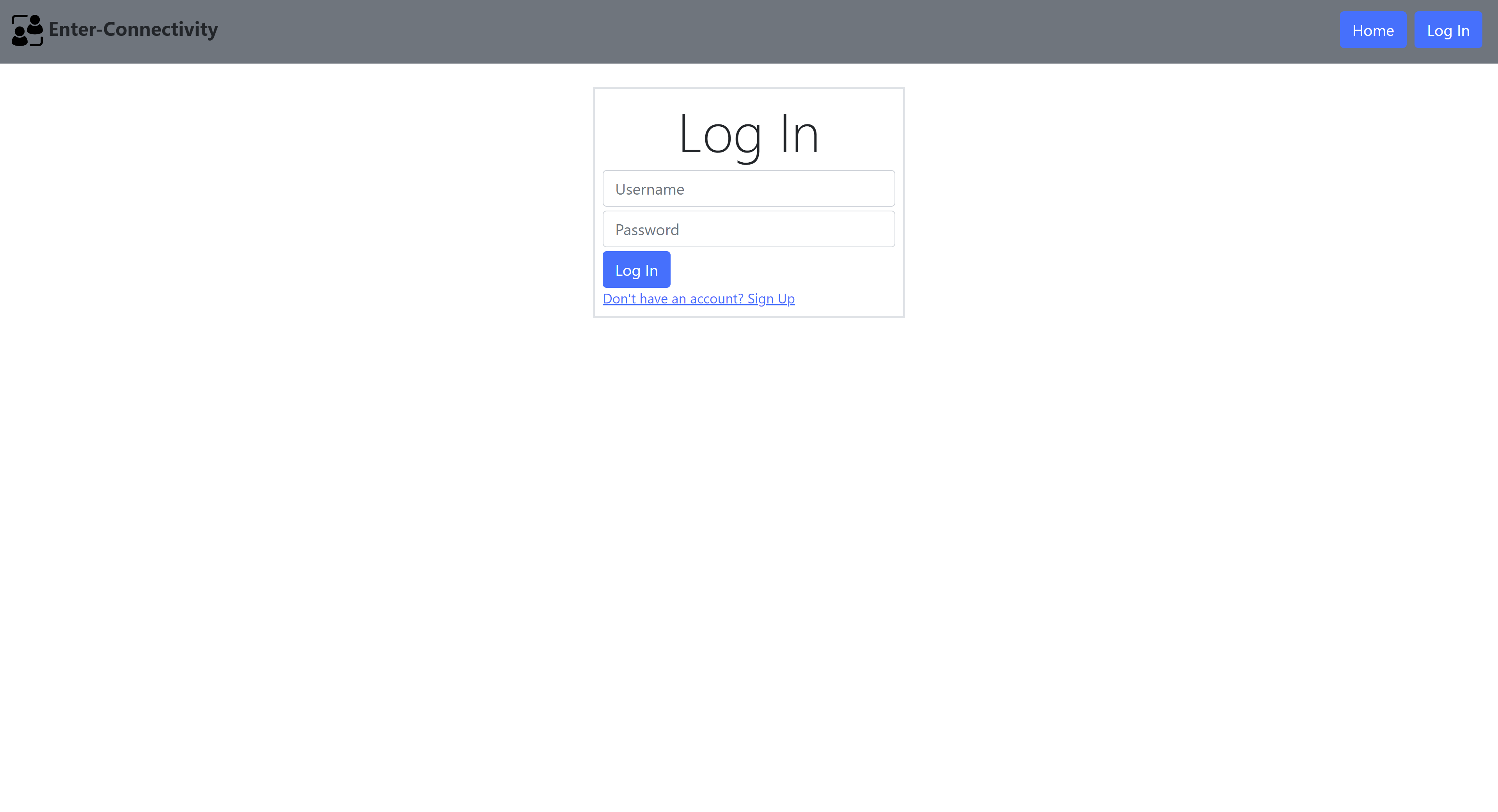
### Edit.html

### Header.html

### Index.html

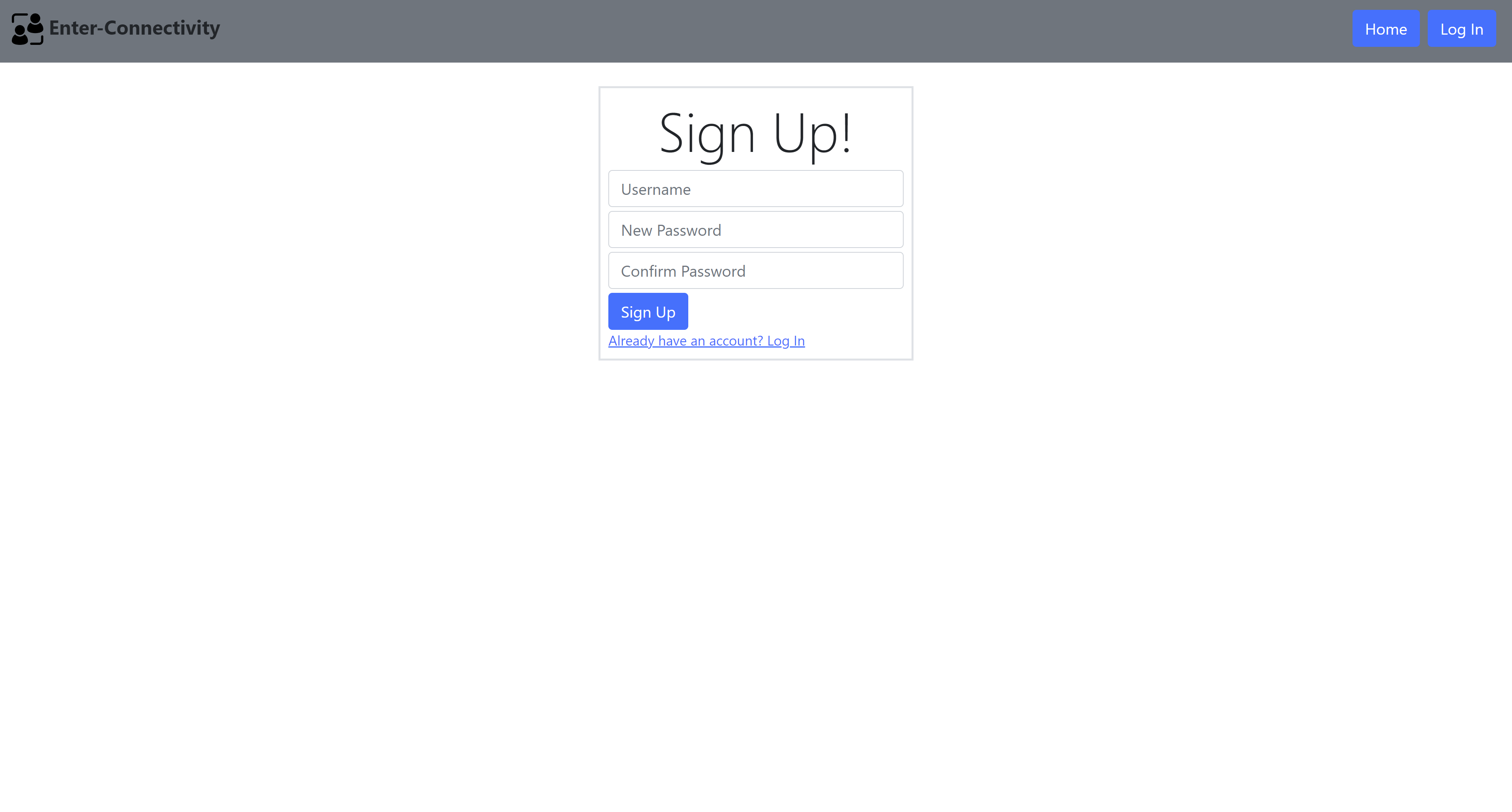


### Log-in.html



### Profile.html

### Sign-up.html



# Demonstration

# Difficulties

* Learning Javascript
* Getting a better understanding of HTML
  + Request/Response systems
* Port Forwarding
  + Sends traffic from a port on the router to the specified device.
  + Two NAT systems caused this to fail
* DMZ
  + A part of the network is made open to the public network/internet
  + Two NAT systems caused this to fail

# Future Additions

Desktop application

Allow adding of existing contacts from phone/other apps

Suggestions for possible outings

# Conclusion

The primary goal of this project was to familiarize ourselves with the interaction between different network systems. Firewalls, routers, NAT systems, and the internet all interact with each other in complex ways. Each step of the project helped us become more experienced with working with different network systems. Working with  the network gave us a better understanding of the request-response system between the front and back end.