

Plain Data

Michael Grieco

Project Description



Table of Contents

1. [Introduction](#)
2. [Overview](#)
 - 2.1. [Server-Client Model](#)
 - 2.2. [Server Body](#)
 - 2.3. [Client Body](#)
3. [Demo](#)
 - 3.1. [Accounts](#)
 - 3.2. [Files](#)
 - 3.3. [Boards](#)
4. [Conclusion](#)
 - 4.1. [Implications](#)
 - 4.2. [Future Edits](#)



1. Introduction

Plain data is an online platform that encourages public contribution and communication. On this application, users can ask questions and publicize their research; they can upload work in order to seek public opinion regarding it. This opinion comes from other users who have the opportunity to contribute their opinion or give advice on the topic. The general purpose of Plain Data is to encourage progress, data transparency, and most importantly, public contribution. As more and more people gain access to such public files and data, there is a much greater chance of finding a solution to the original problem or issue. Along with this, making data public can help increase awareness of current events. For example, one user can upload a set of data, and another can make it interpretable by the public, allowing more people to be exposed to it. This application, even if it is a prototype, has the potential to do just that.



2. Overview

2.1. [Server-Client Model](#)

2.2. [Server Body](#)

2.3. [Client Body](#)



2.1. Server-Client Model

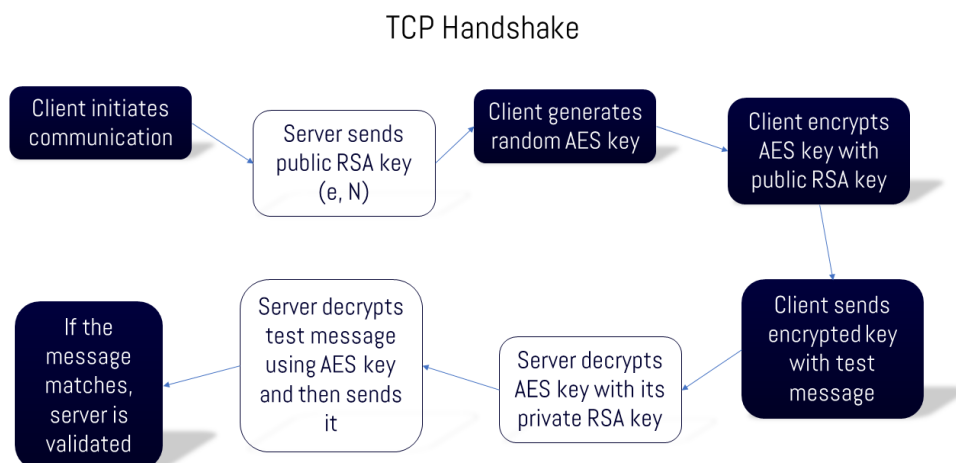
This application consists of a server written in Python and a desktop client written in JavaScript.

For the server, I have a TCP networking library that is available on GitHub [here](#). I chose Python for its simplicity and comprehensible syntax. Specifically, the socket modules integrated perfectly into a class context, allowing the server class to be inherited from seamlessly.

As of now, I have implemented one client, built for the desktop. I used the Electron.js framework for the Node.js platform. This is the first application I designed using JavaScript for the desktop. The reason why I chose this framework was that it allowed me to use HTML and CSS to design the GUI for the application.

Communication Algorithm

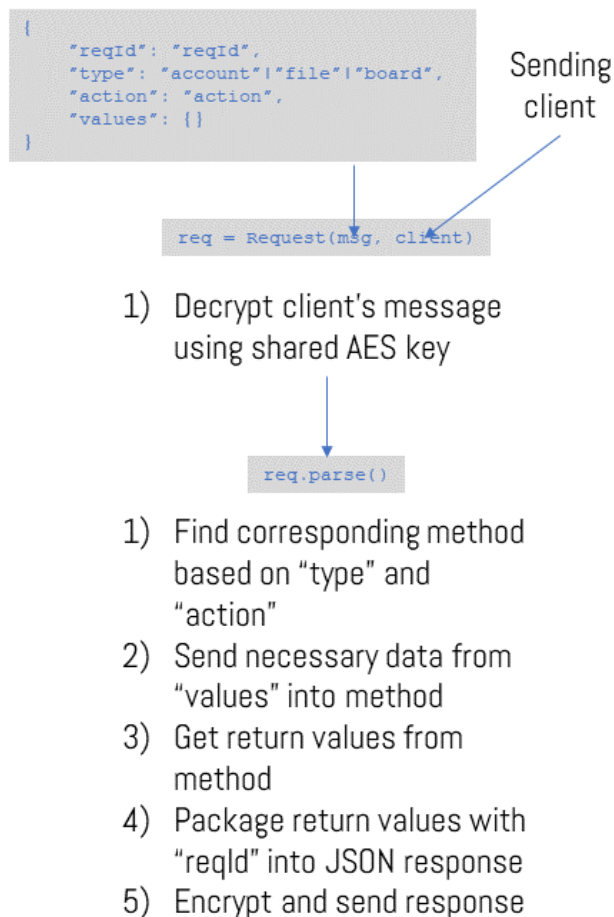
The server and client communicate using AES encryption. The validation process is identical to the standard TCP Handshake used in most modern-day applications. The first step once the client has initiated communication is that the server sends its public RSA key to the client. The client then uses that RSA key to encrypt its AES key and a test message encrypted with that AES key. The client sends the encrypted data to the server, which then decrypts it using its private RSA key. After this, the server decrypts the test message and sends it back to the client. If they match up, the client knows that the server can be trusted. Now the server and client are validated, they send messages and data encrypted with the AES algorithm using the client's AES key, which is more efficient than the RSA algorithm.





2.2. Server Body

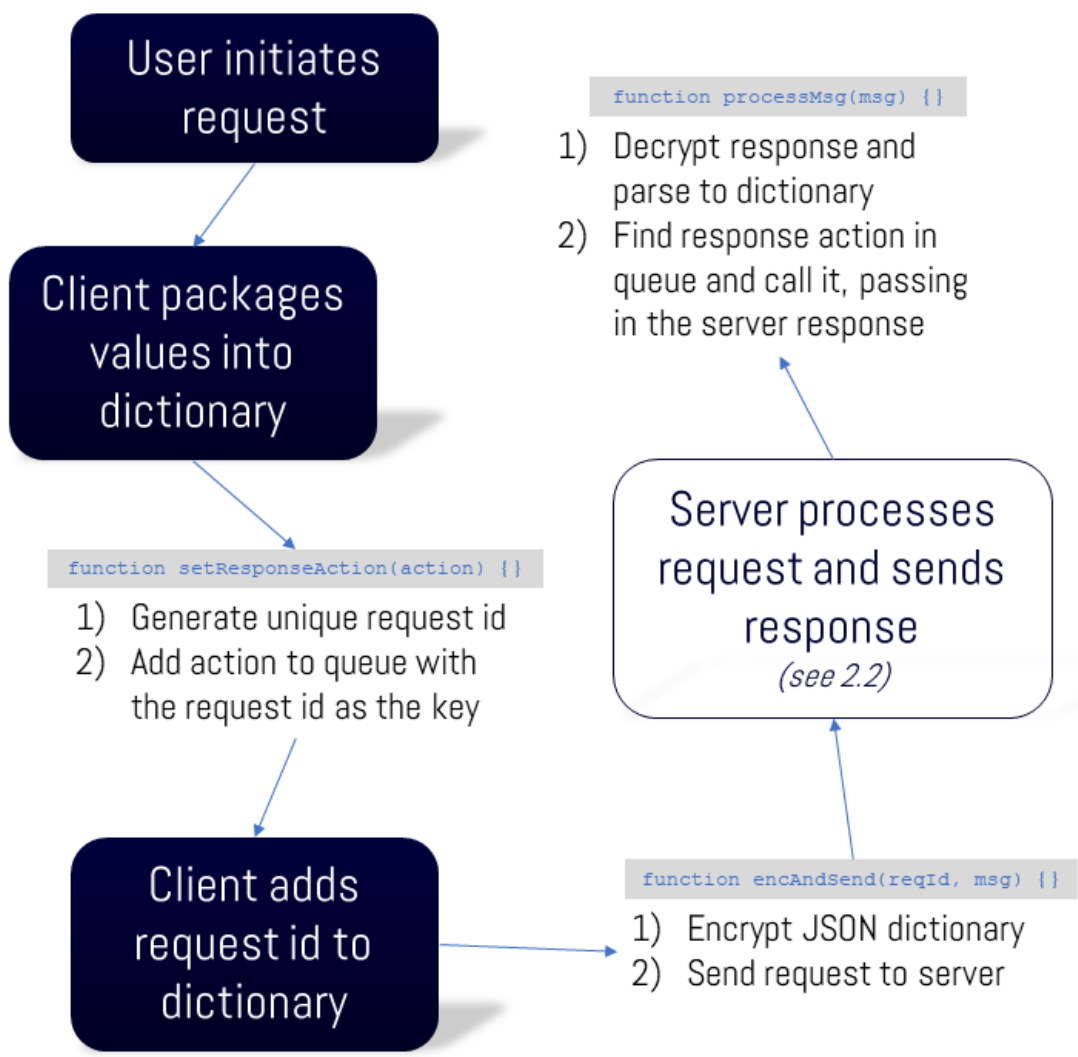
Once the server has validated a client, it then routes each message through a parser. It first decrypts the message — which was encrypted with the shared AES key — and then packages it in an object of type `Request`, which is a class on the server. The `Request` class has a method called `parse`, which first loads the message into a dictionary (with the JSON module) and then assigns attributes to its own object. After this, it finds the correct method to call, based on the type and action of the request. These methods are stored in the `models` folder, with a different file for each type (user, board, and file). The action methods return values in the form of a dictionary, with the main key as `"result"` and its corresponding value as `true` or `false`, to indicate whether the request yielded a successful result. After that, it converts the return values into a JSON string, encrypts it with the shared AES key, and then sends it to the client.





2.3. Client Body

On the other side of the application, the client uses the same validation process for the server. And once the server has been validated, the client first decrypts the server’s message using the shared AES key, like the server. However, the client does use a system of callback methods linked to unique request ids to process responses, which is different from the server. For each request the client sends to the server, it generates a unique request-id and adds it to a request queue with its corresponding response method. When it receives a response, bearing a request id, it looks for the callback method within the queue with that id. It passes in the response — which was parsed into a dictionary using the JSON library — to the callback method it found.





3. Demo

3.1. [Accounts](#)

3.2. [Files](#)

3.3. [Boards](#)

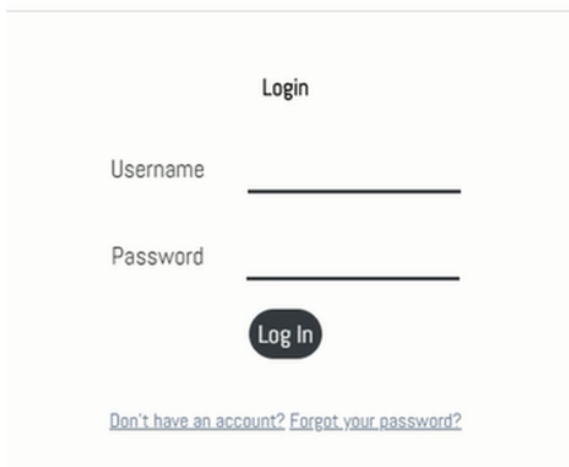


3.1. Accounts

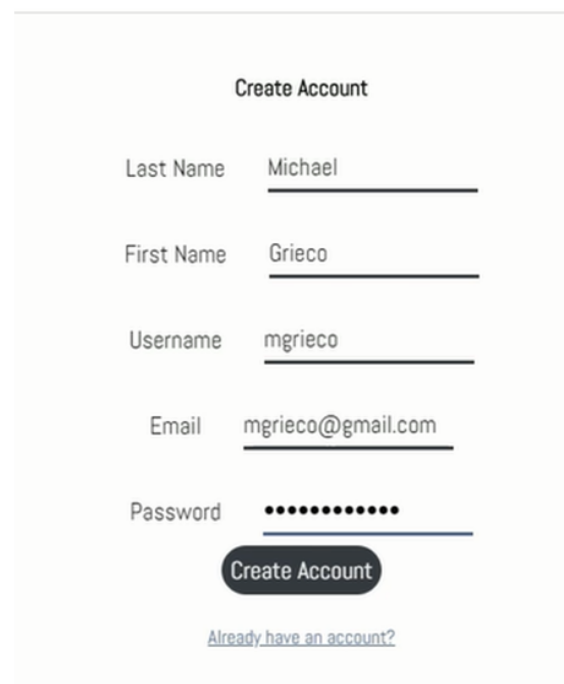
Each user has an identity on this platform and must have an identity to upload files or create boards. Accounts are stored server-side on a local SQL server. For the time being, the data can be left unencrypted since the database is not made public.

Upon starting the application, the user has two options — to create a new account or log in with an existing one. The default one is to log in with a username and password. After completing this form, the client sends a login request to the server who then compares the credentials with the database. If the server finds a match in the database, it returns the user id, email, and name corresponding to the entered data.

To create an account, the client must click on a link in the default screen which then takes them to a different form. They enter their username, password, email, and name. The email and username are checked to make sure there are no existing accounts with those credentials and the password validation consists of ensuring the password is strong enough. If it passes those tests, the user is added to the database and the client is sent back to the login screen to input their new account information.



The login form is titled "Login" and contains two input fields: "Username" and "Password". Below the password field is a "Log In" button. At the bottom of the form, there are two links: "Don't have an account?" and "Forgot your password?".



The "Create Account" form is titled "Create Account" and contains five input fields: "Last Name" (with the value "Michael"), "First Name" (with the value "Grieco"), "Username" (with the value "mgrieco"), "Email" (with the value "mgrieco@gmail.com"), and "Password" (with masked characters). Below the password field is a "Create Account" button. At the bottom of the form, there is a link: "Already have an account?".



3.2. Files

Uploading

After logging in, the user is taken to a dashboard where they can choose from several actions. One of those actions is to upload a file. This is a principal function on this application — allowing users to upload files to integrate with their posts. In the upload form, the user enters basic information about the file, such as its author and category. The client then submits a request to the server with a physical copy of the file. Once the server receives the request, it assigns a unique numerical id to the file. After that, it writes the file contents to a file under that id. It stores the rest of the information in a JSON manifest file.

Downloading

Along with being able to upload a file, another core aspect of this application is allowing other users to access the files uploaded to the server. Since everything is public, the client has the ability to download these files. On the client machine, the user can search for files with several parameters. There is a link that users can click on, which allows them to open that file. The file is downloaded to a temporary location on the client machine, where they can then access it.

The 'Upload File' form contains the following elements:

- Author:** A text input field.
- Title:** A text input field.
- Category:** A dropdown menu.
- Tags:** A text input field with the placeholder text 'eg: calculus,derivative:'.
- Select File:** A button to choose a file from the local system.
- Upload File:** A button to submit the file to the server.
- Cancel:** A link to cancel the operation.

The 'File Search' form contains the following elements:

- Title:** A text input field.
- Author:** A text input field.
- Filetype:** A text input field.
- Category:** A dropdown menu.
- Tags:** A text input field with the placeholder text 'Separate each term by a comma'.
- Search:** A button to execute the search.
- Cancel:** A link to cancel the operation.



3.3. Boards

The next major section of this application is creating boards. Boards allow people to physically ask questions about files they uploaded. The purpose of this application would be defeated if users could not do anything with the files. With boards, users can link files to the question and seek public opinion. On the dashboard, users can select to create a new board using an HTML interpreter. As of now, the only way to design a board is with HTML code; a graphical designer will be included in the future.

Linking Other Elements

To link files, users must use the anchor tag in the form: `label`, where `file_id` is the numerical id assigned to the file by the server. When a user clicks on this link, it sends a request for the file with that file id. The server returns all the metadata in the manifest file along with a temporary copy of the file.

The way users can respond to these boards is through comments. On a board page, there is a button where a user can enter a comment. It uses the same HTML interpreter to design the comment. Users can link their own files using the same syntax, and they can also link other boards with relevant information that could contribute to a solution. This can be done through a tag that is almost identical to the file anchor: `label`, where `board_id` is the numerical id assigned to the board by the server.

Create Board

Title

Category

Tags

Content

Create Board

Post a comment

Subject

Post Comment



4. Conclusion

4.1. [Implications](#)

4.2. [Future Edits](#)



4.1. Implications

This application has the potential to help people across the world. It can contribute to the solution of a problem as small as a personal project or something as broad as foreign policy. By allowing anybody to add their opinion, there will be a diverse set of answers to each issue. As a result, there is a good chance that one of these unique solutions will be that one idea that helps fix the problem. People in the world are brought up in different areas and as a result, they have a different value or belief set. And these people can bring some extraordinarily rare idea that could be very simple but could hold the solution. This application encourages this sort of contribution, and with more people in the world that are able to help and input their own ideas, there is a larger probability that users can find solutions to a vast range of issues and challenges.



4.2. Future Edits

In the future, I wish to make the application more user-friendly. To start, I want to add an easier way to find boards and files. There is already a search algorithm in place, but it may become hard if a user wants to repeatedly find the same file or board. To combat this, I wish to add a function where a user can bookmark certain items for future reference.

Additionally, the current element that allows users to design their board or comment is a simple text area that interprets whatever is entered as HTML text. In future versions, I will implement a graphical design element that allows users to create boards and comments without having to know any HTML or CSS code. For example, the only way to link files or other boards to a comment or a board is through anchor tags that require a unique attribute with the key, 'tar.' This may become tedious or outright confusing for many. With this graphical element, users would be able to drag components into the area in order to design the board or comment.

One of the concerns I had with the concept of this application was that I am giving too much trust in the users not to abuse the publicity of the data and files. This application could potentially store a large amount of user-created work. And there are no measures in place to prevent people from simply stealing that work. So, as a future update, I could add guidelines on how to cite a file, or how to use it elsewhere while giving credit where it is due, in order to encourage people to responsibly use the data.