Design Report

File Adventurer

By Lukas Rickard
Nicholas Howes
Joseph Grant
Connor Morales
William Trok

Table of Contents

Summary of Changes

Project Objectives:

- SSH Command Line (Scrapped from initial release)
- UI Elements:
  - File Info (Not viewable)
  - Refresh/Reload button (Refreshing should be automatic)
  - Delete Button (removed from the toolbar)

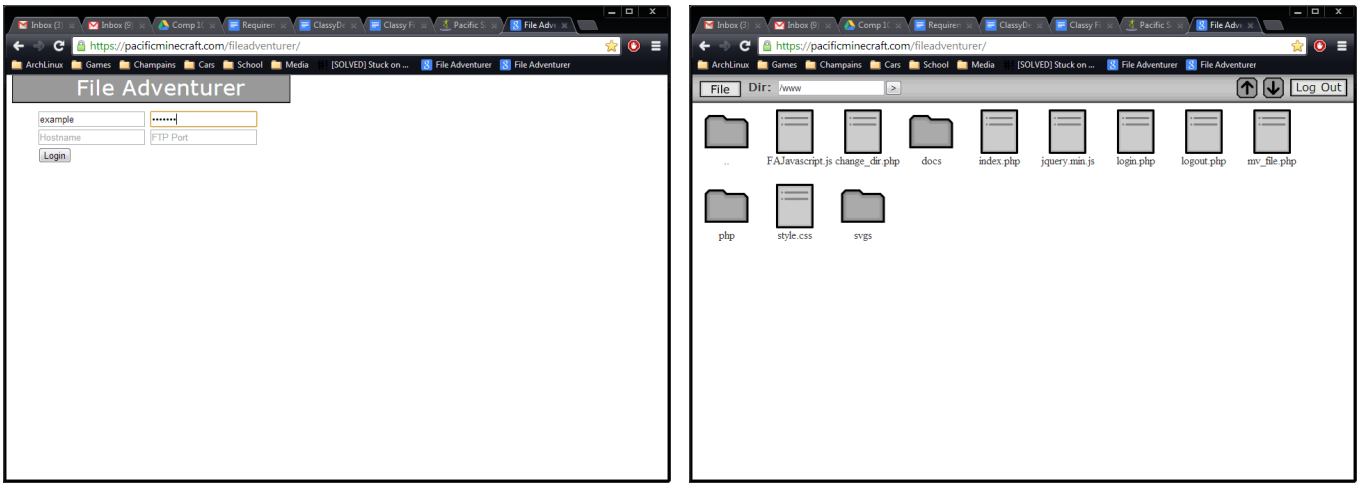
System Design:

- Abstract Use Case
- Class Diagram
- Short Use Case Descriptions:
  - Logging In
  - Downloading a File
  - Delete a File

Customer Requirements

File Adventurer is an application anyone can use to access files on a linux server with an FTP connection. Anyone with an account on a server running File Adventurer should be able to simply go to the URL where it's hosted and enter their username and password to browse the files in their home directory. In addition to this, anyone can access their files on a server with an FTP connection using File Adventurer, their username and password, along with the hostname and FTP port of the server they want to access.

Any website owner can install this application on their site to gain access to their files from anyone's computer through a supported browser. It gives them the convenience of accessing their server with having to carry anything around, not even a flash drive.
This is what it currently looks like:



Login Screen, Left. File Browsing, Right.
Users may use this application very similarly to how they would use other file explorers and operating systems to manage their files. Files can be drug around and standard actions can be carried out, such as renaming, deleting, and downloading. The convenience this application can provide is proportional to inconvenient the user finds carrying around a device to access their server. Even if you don't mind carrying a flash drive around, File Adventurer gives you the security of being able to access your server from practically, because it supports Firefox and Google Chrome. There are alternative ways to access one's server, but none of them provide as much accessibility as the web. Currently people use Filezilla, scp in ssh, or Winscp to transfer and manage files on their server, but you can use File Adventurer in addition to any or all of these tools… And it's free.

Glossary of terms

HTTPS - Standing for Hypertext Transfer Protocol Secure, HTTPS is an encrypted internet communication protocol used for transferring website data from server to client.
SSH - Standing for Secure Shell, SSH is an encrypted internet communication protocol that allows remote access of a server's command line.
FTP - Standing for File Transfer Protocol, FTP is a protocol used to transfer files between hosts.
PHP5 - Standing for PHP: Hypertext Preprocessor, PHP is a server-side scripting language that is used on the backend of File Adventurer.
POST - A method of data transfer from client to server that is utilized in File Adventurer.
Ajax - Standing for Asynchronous Javascript and XML, Ajax is utilized to activate PHP scripts and retrieve data from the server in the background without refreshing the page.
Javascript- A client side scripting language that runs inside internet browsers to make them more interactive and to manipulate HTML.
jQuery - jQuery is an open sourced javascript library that is designed to simplify the process of client-side scripting
JSON - Standing for Javascript Object Notation, is a data interchange format used to share data between different programming languages and javascript.
(DOM) Event - These are events that javascript sees occurring on different HTML elements such as divs and images. The DOM being the HTML item that can be changed, manipulated, and checked of events.

Functional Requirements Specifications

Stakeholders-

- Server Owners will have an easier time accessing their server with this software.
- Server Admins will also be helped by this tool.
- Programmers who would like to share a files or server server space would benefit to use this software.
- Anyone who stores their files on a remote file server could potentially benefit from this software.

Actors and Goals-

Main Actors without goals

Server User - Initiating
Server Files - Participating
HTTPS Server - Participating

Abstract Actors and goals

Initiating actors

Server User - This is anyone who has an account on the same server that File Adventurer is running on.

Participating actors

Input Verifier - This actor is the javascript we have running on our application, client side. It checks the validity of input and confirms with the user if he would like to carry out certain tasks.

Server Code - This actor is the PHP functions on the server. This code does most things from Verifying login to initiating downloads to manipulating file on the server.

HTTPS Server - This is our website itself which we are utilizing to allow people to use our application from a web browser

Short Use Case Descriptions-

Logging In: The user enters a hostname, FTP port, and their username and password on the web client. The server validates the information and starts an FTP session with the user and logs them in.

Downloading a File: After a user has logged in, they select a file and double click to download it, then their browser attempts to open the file in a new window.

Uploading a File: A user logs in, and clicks the upload file button, the upload dialog comes up in which they can input the destination of the file on the local machine that they wish to upload, after that they input that destination and the file(s) there are uploaded to the current directory.

Renaming a file: After a user has logged in, they select a folder/file and hit the rename button. The name of the file they selected becomes editable and is renamed when they hit enter. An error may occur if they attempt to rename to an illegal name.(eg. '/' in file name, explicit file names, names that are death threats to the president.)

Delete a file: A logged in user, selects a folder/file or folders/files and hits the uses the file adventurers delete button in the file dropdown.

Moving a file: A logged in user selects a file and drags it to the folder where they want to move it. It disappears from under the mouse and is now in the folder they dragged it to.

Navigating to a directory: A logged in user double clicks on a folder and the the application displays the contents of that file as the currently observed directory.

Using SSH terminal to create: A logged in user, opens up the ssh window, enters the command "sort > list.txt", and then enters the words "carrot","vegetable","duck" followed by Ctrl+d. The command is passed directly onto the the shell, and the file list.txt is created in the current working directory with the contents "carrot duck vegetable".
NOTE: This feature has been worked on, but only basic backend functionality exists. Thus, it does not exist in our release.

Full Use Case Descriptions-

Renaming a File

Precondition: The user has File Adventurer running on their web server at a certain URL. They are in either Google Chrome or Firefox, they have navigated to the URL where it is hosted, and they have logged in.

Main Flow of Events: The user clicks a file and selects Rename from the File menu. The name of the file comes into focus and becomes editable. They changed the name to the one they desire and hit enter.

Exceptional Flow of Events: The user clicks away from the file before clicking Rename, thus unselecting the file. Upon clicking Rename, an error box appears, informing the User that nothing was selected. The user may also select multiple files before hitting rename. If they do this they receive an error telling them that they cannot rename multiple files.

Exceptional Flow of Events: When choosing the new file name, the User enters an illegal character or name. Currently the user will get no error, but the file name will not be changed once they exit the current directory or logout.

Postcondition: The file name is changed, and the directory listing is refreshed.

Moving Files

Precondition: The user has file adventurer running on their web server at a certain URL. They are in either Google Chrome or Firefox, they have navigated to the URL where it is hosted, and they have logged in.

Main Flow of Events: The User clicks and drags a file from its previous place to over a folder. The file disappears and in now in the folder they dragged it to.

Exceptional Flow of Events: The user tries to drag the file into another file. It simply returns to its original place and no action is performed.

Exceptional Flow of Events: The user drags a file they do not have permissions to into another folder. They should receive and error and no action will be performed.

Postcondition: The files are moved into the chosen folder, and the directory is reloaded.

Deleting a File

Precondition: The user has file adventurer running on their web server at a certain URL. They are in either Google Chrome or Firefox, they have navigated to the URL where it is hosted, and they have logged in.

Main Flow of Events: The user clicks one or more files and/or folders and selects Delete from the File menu. They click OK on the prompt asking them if they want to delete the file(s). A request is sent to the server, and upon receiving an affirmative response, the file is removed from display.

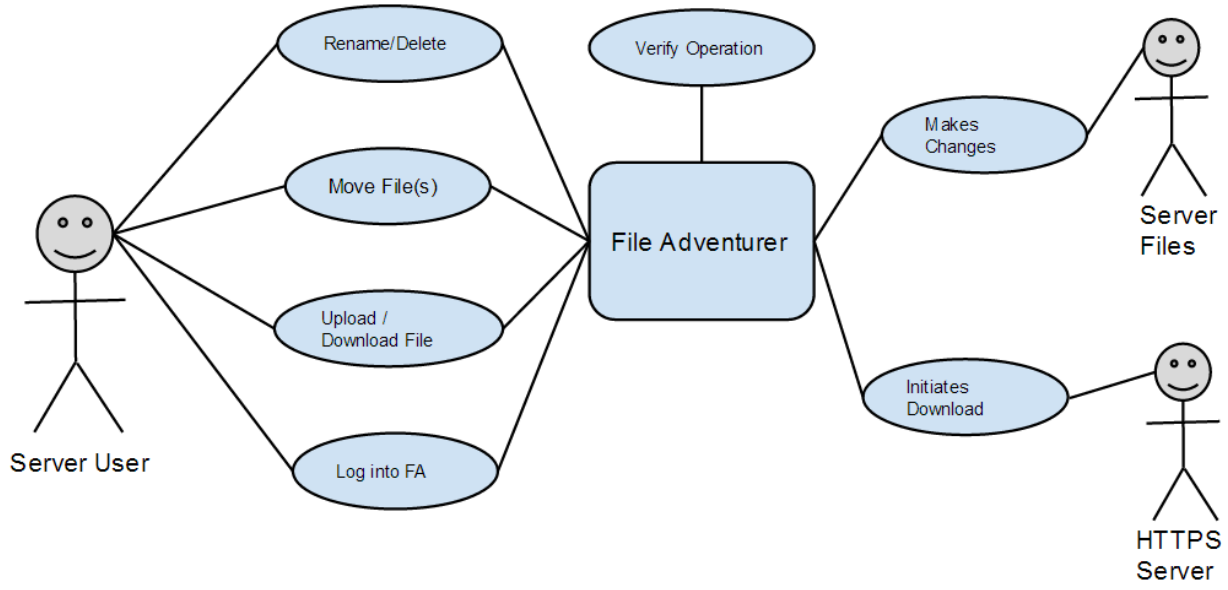
Exceptional Flow of Events: The user clicks away from the file before clicking Delete, thus unselecting the file. Upon clicking Delete, the file menu closes and nothing happens.

Exceptional Flow of Events: The request to the server returns false, possibly because the user does not have permissions, and the file in not deleted.

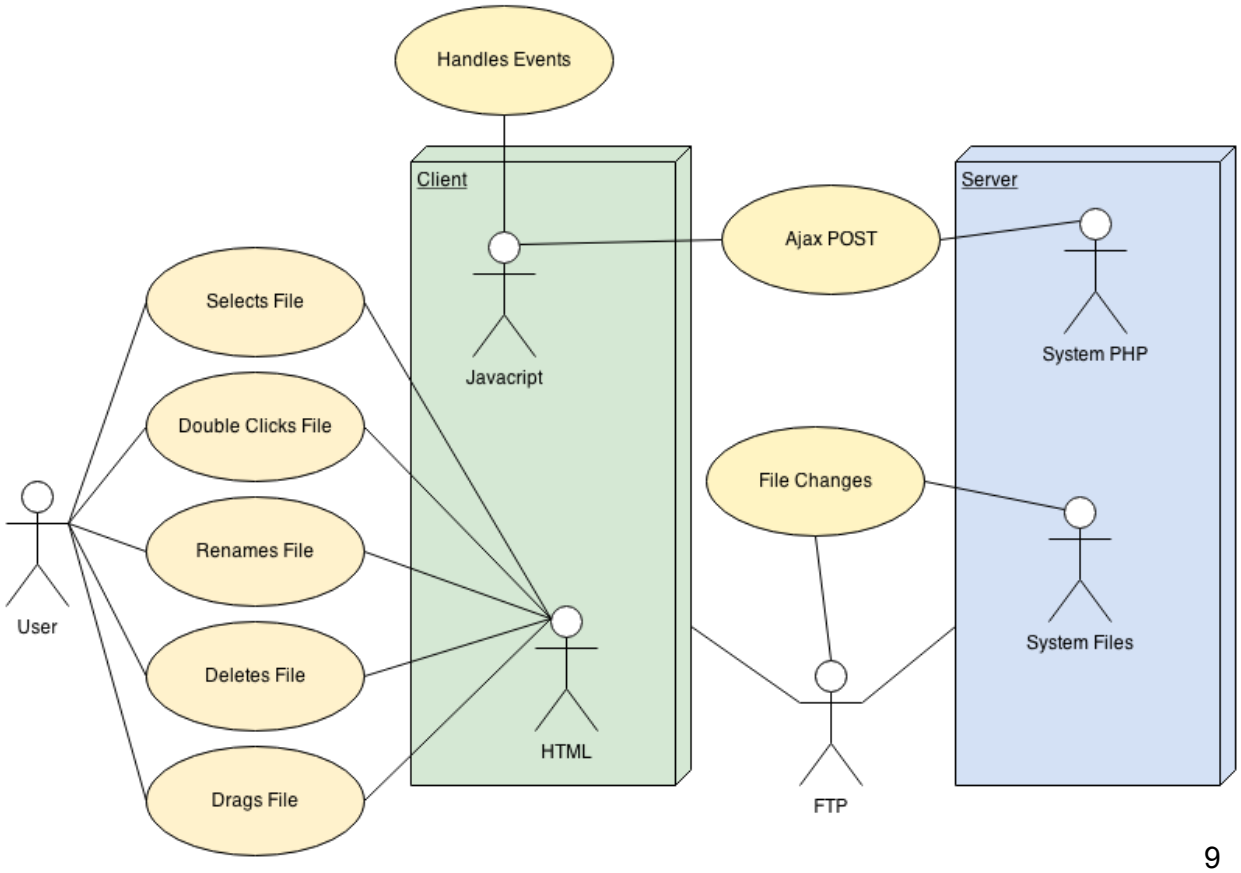
Postcondition: The files are deleted, and the directory is reloaded.

Use Case Diagrams-

Basic Diagram

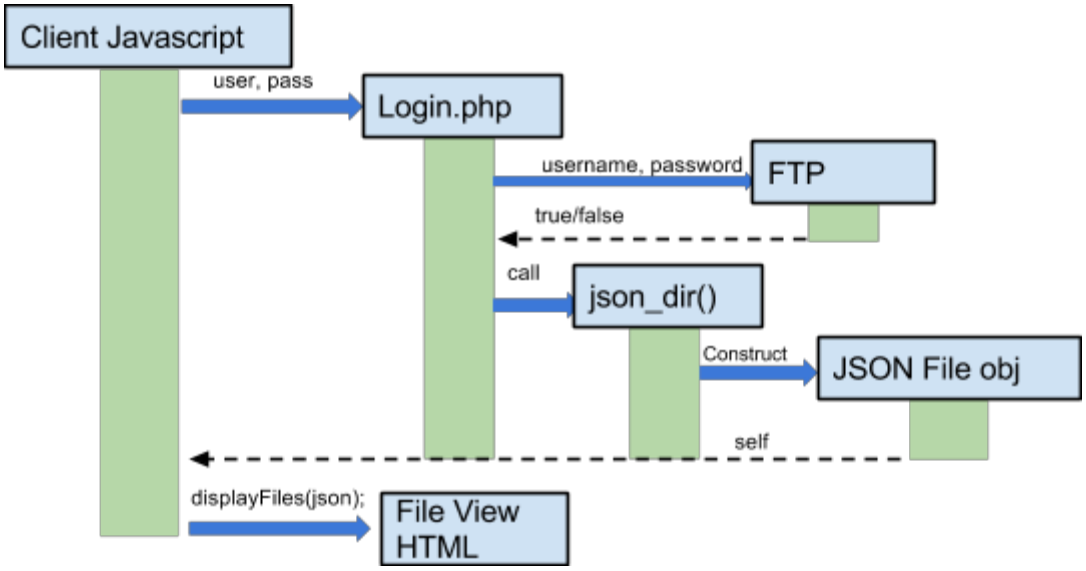


Abstract Diagram



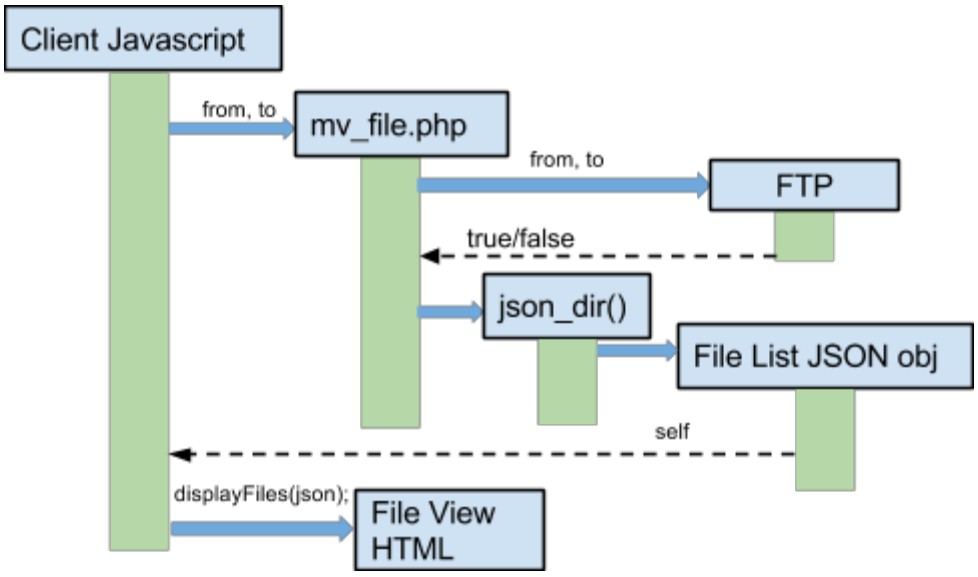
Interaction Diagrams

Logging In



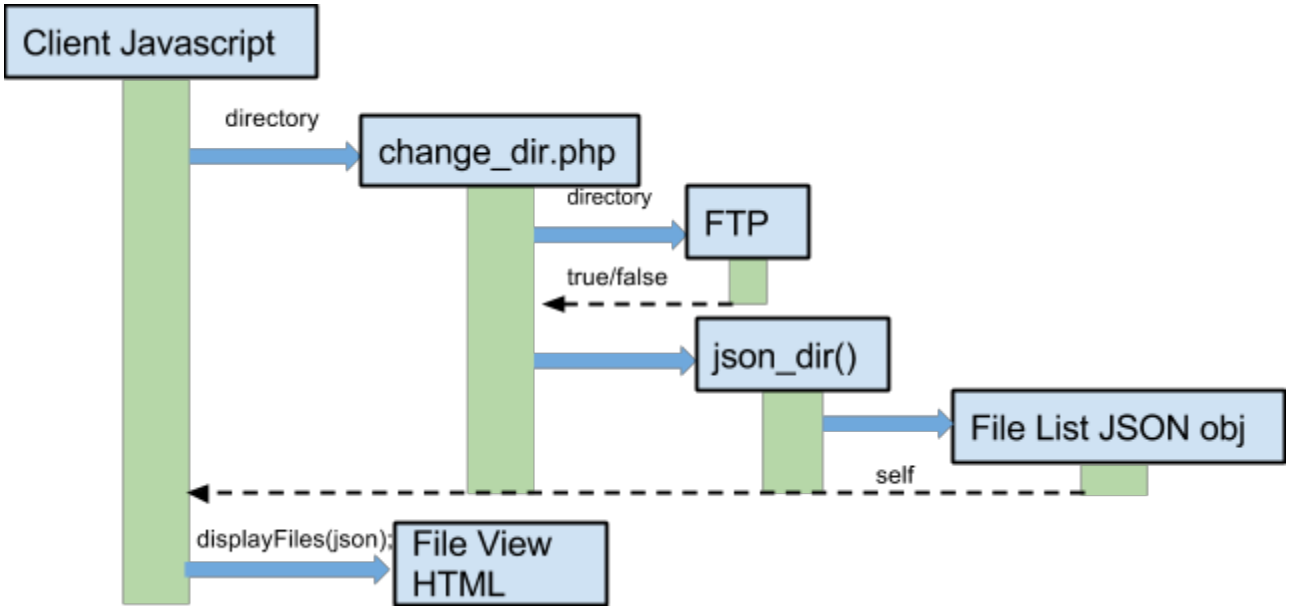
We decided that each function such as Login.php should call the json_dir function to update the file object. During login and when most other php function are called from javascript, we call displayFiles() right after to update what is shown on the screen. We also utilize FTP for all our file changes.

Moving a File



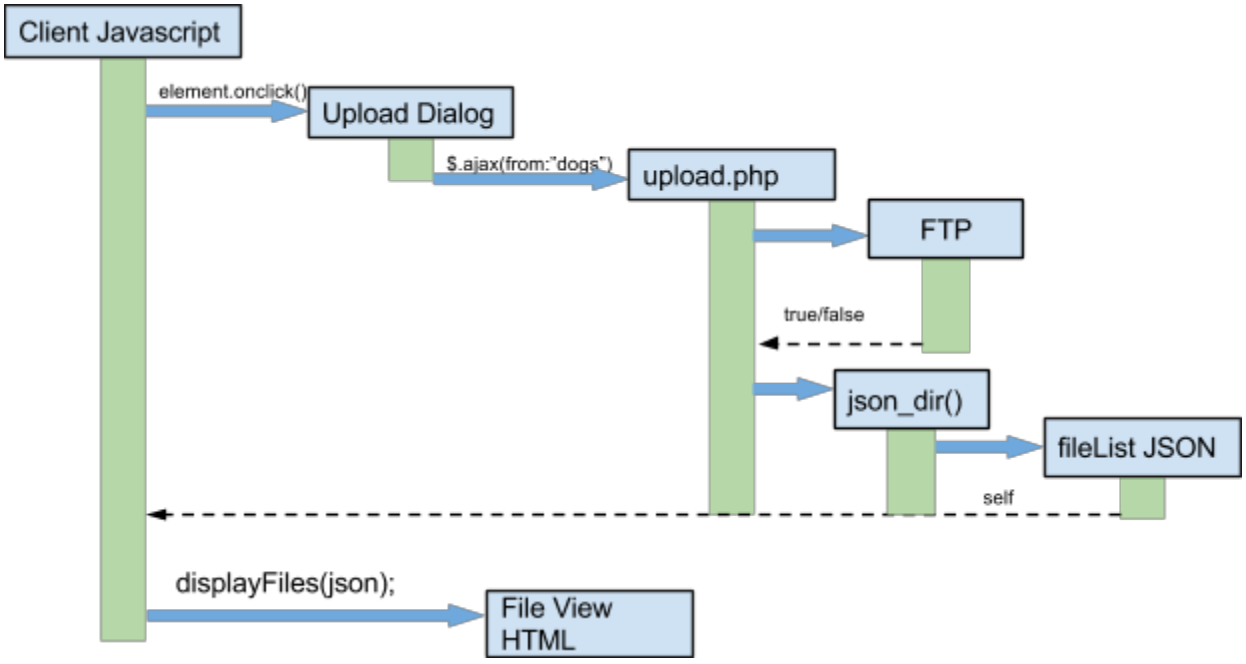
We thought it best to parse and reject bad input going to both mv_file.php and then again to system in order to ensure that requests failed as rarely as possible.

Navigating to a Directory



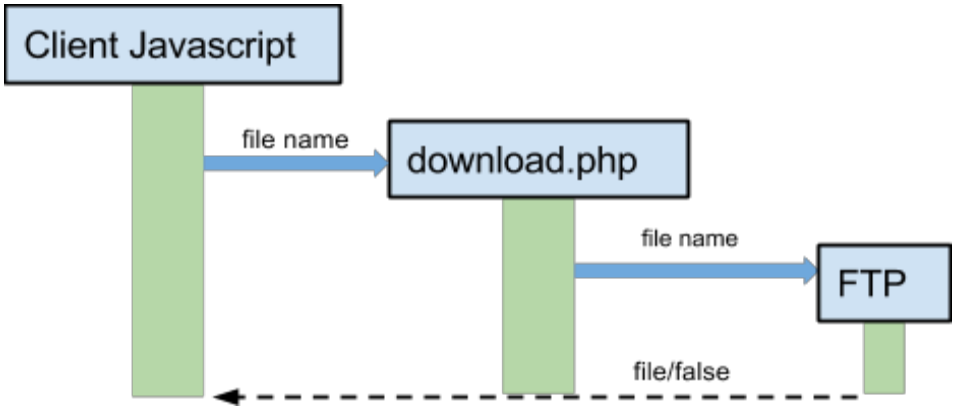
Navigating is very similar to moving a file, but we verify with FTP whether or not the current user is allowed to navigate to that directory and then update the currently displayed files as normal.

Uploading



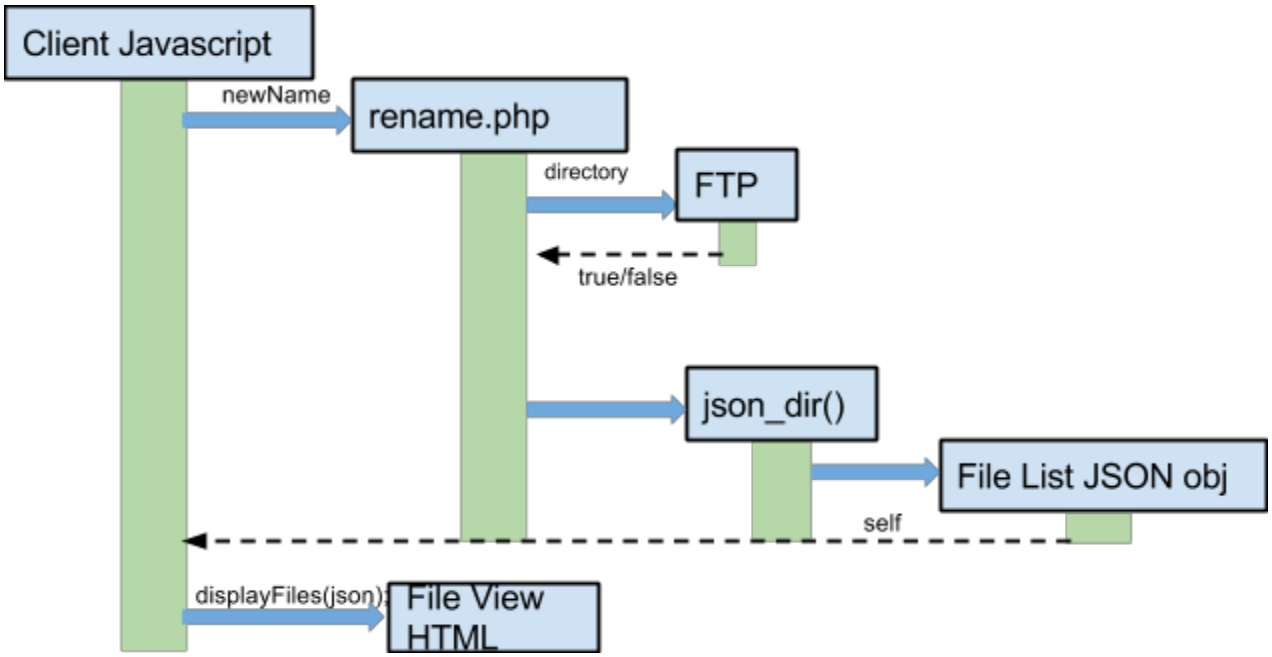
Upload is similar to most file interactions but it requires the user to wait before they continue navigating because the current directory will be updated upon the upload completing.

Downloading a File



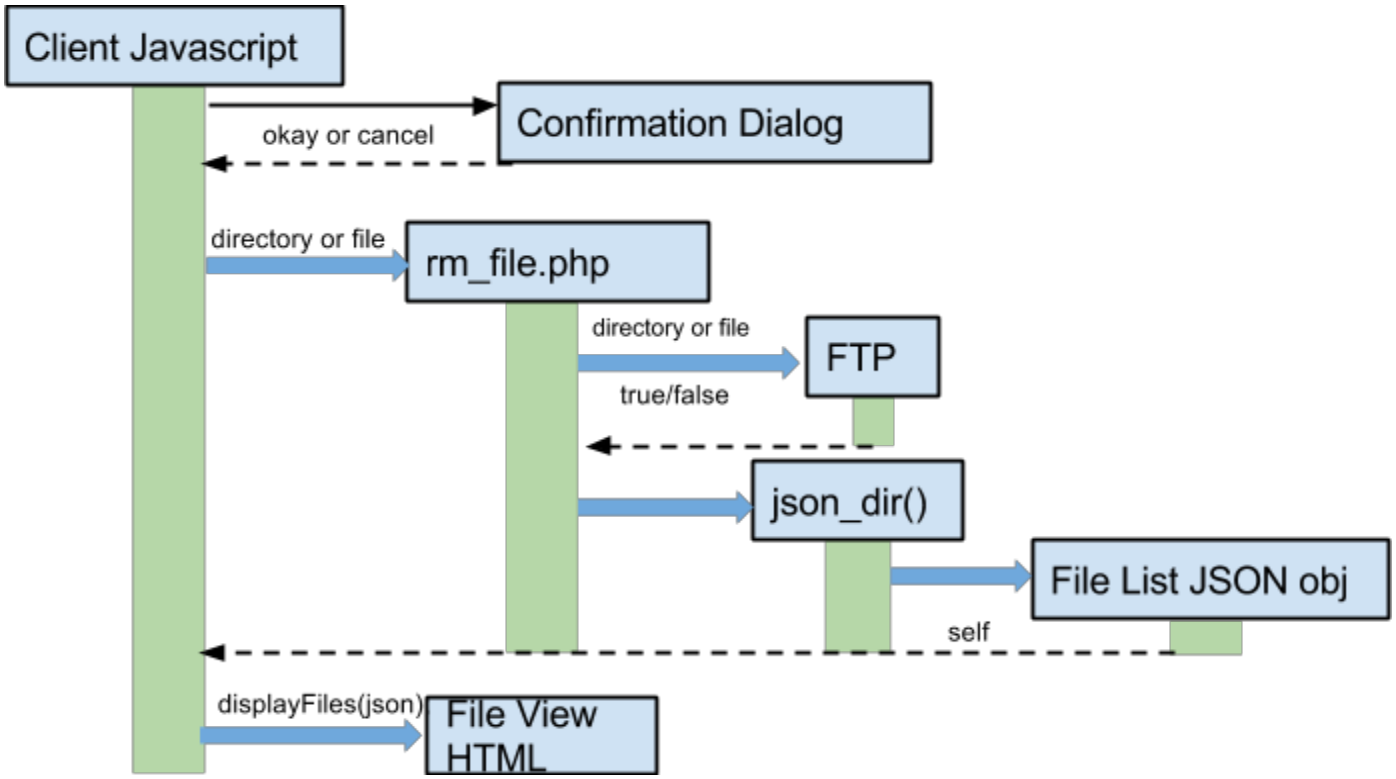
Downloading a file is trivially simple when an ftp connection is present with server. We will have a failure window pop up if the download fails.

Renaming a file



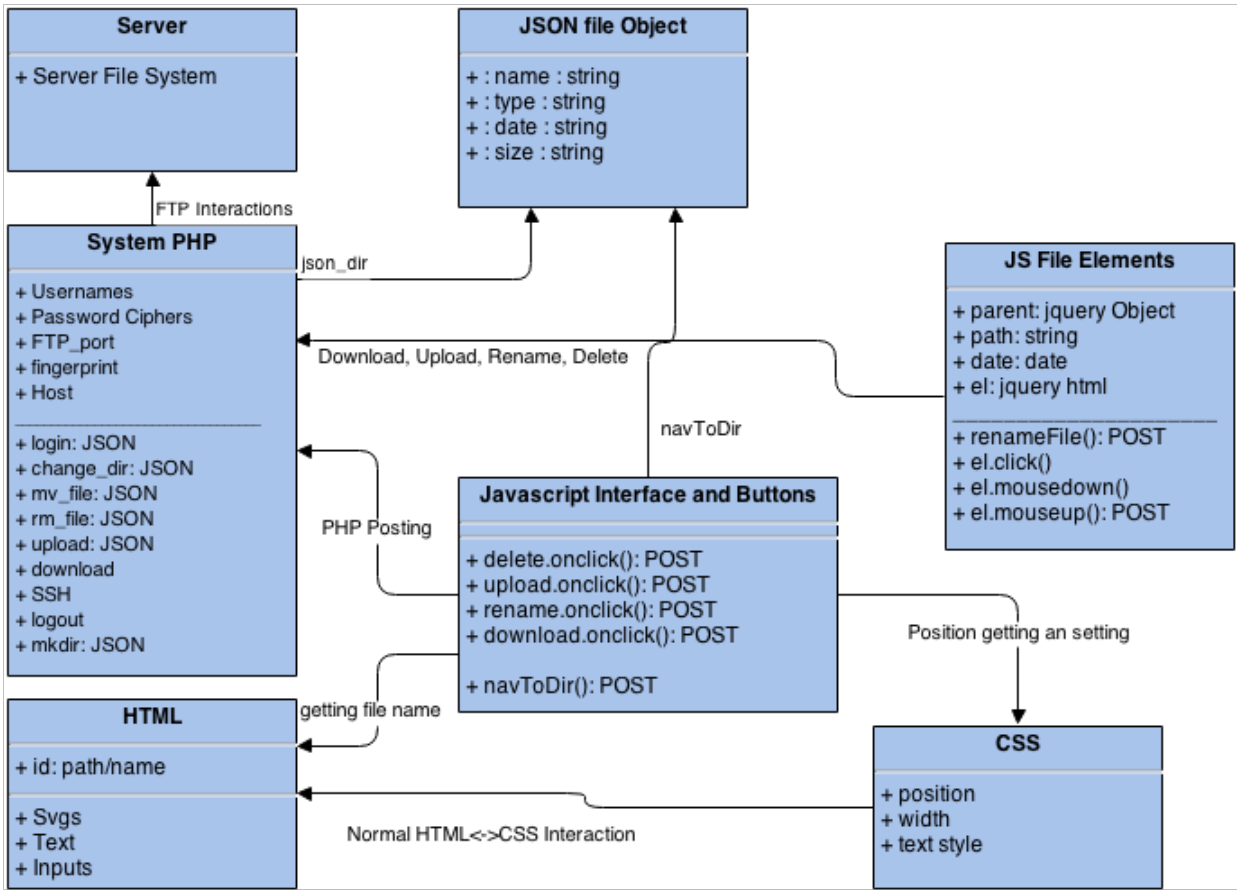
Renaming works almost exactly the same as changing a directory, except the current directory does not change and a file changes its name. We simply use a different FTP argument.

Deleting a file



Similar to the other direct file editing interactions, but this action in confirmed by a popup box, before it continues with the operation. The confirmation dialog specifies if one or more file are being deleted. It also displays the progress of the files being deleted.

Class Diagram and Interface Specification



Main Class Diagram

Data Types and Operation Signatures-

Operation signatures are detailed in the class diagram above.

jQuery Objects - We use these to store information about files, including their parent for easy access from functions that trigger on events including those files.

JSON Objects - We use one JSON object to store organized information about the current and the parent directory. This is edited by PHP and Javascript.

HTML Divs - We store some vital information in the HTML in the browser. HTML elements are also used as the focus of event triggers. For these reasons, one could consider the HTML divs in our applications to be.

A Folder of PHP functions

System Architecture and System Design

Mapping and Subsystems

Server

Any Linux flavor
Apache2, SSL module
PHP 5.3+, pcntl module

Client

Chrome 30+, Firefox 15+, javascript & cookies enabled

Network Protocol

Definitions:

Target Server - The server that the user "logs in" to, when first logging in the user enters the ip address as well as credentials for this server.

Pacific Minecraft Server- This is the server used by Team Classy to run (and test) the web application.

HTTPS - Hypertext Transfer Protocol Secure, Sensitive data being communicated

FTP - *File Transfer Protocol,* Universal

SSH - *Secure Shell,* Universal

UNIX Sockets - Fast communication for processes

The following protocols are used: SSH, FTP, HTTPS, and standard unix sockets, all of which are defined briefly above. They are also detailed in the Glossary of Terms. The use of each protocol is spelled out below.

HTTPS: Accessing general data from the website, Ultimately as far as the user is concern all data is coming to them through this protocol. More specifically the user uploads and downloads all data including files from this protocol.

FTP: This protocol is used to transfer files from the pacific minecraft server to whichever server the user may be logged in to. Having FTP enabled on the target server is a requirement on the user's part. If a user uploads a file it will first be uploaded to pacificminecraft.com server, then from there FTP will be used to get the uploaded file to the destination server.

In addition to using this protocol for file transfers, this protocol is also used to obtain all data about files stored on the target server. FTP is also used along with SSH to verify that the users credentials are valid.

SSH: This protocol is used to allow the user to send shell commands to the server. When a user initially opens the ssh window, a ssh shell is created, though from then on sockets are used to interact with the shh shell. SSH is also used when the user initially logs to verify that the users credentials are correct.

UNIX Sockets: Used along with ssh to quickly pass commands and receive output from the remote shell.

Recommended Hardware Requirements

SERVER

CPU: No testing has been done but something with 2 or more GHz should be okay
RAM: At least 512 mb
HDD: All our resources should be less than 1Mb

Algorithms and Data Structures

Algorithms

-Rijndael

This cipher, more commonly known as AES, is used to encrypt the user's password when it is stored in the SESSION variables.

Data Structures

-JSON File List Object

This is a generic javascript object, containing either one or two arrays, each containing instances of the File class. We chose to use JSON instead of an alternative like XML because it is a more compact data type and the ease with which we could parse through it with javascript.

- Javascript File element, Jquery Object

This javascript script class object handles the events that occur on the file elements that are created from the JSON File List Object. Objects of this class hold all the information about the divs that represent the files.

- HTML

HTML is has the file path that each file SVG represents stored in its ID

User Interface Design and Implementation

Initial Implementation

Initially, Lukas wanted a stylized and metaphorical interface design that made the user feel like they were using the technology of the future. The shortcomings of the some of the design ideas for this were very apparent after the whole team looked at them. We were forced to make some design decisions in the process of revising the user interface which really made the interface design the first pushing factor that made us decide what we really wanted from the project. Although the design was ambitious at first, it was always designed for ease of use.

Look and Feel Design

From the beginning we wanted the interface to be as simple possible, but also reminiscent of other file interaction software, so our users would not have to learn anything new. We made our file and folder icons similar to those anyone might see in an operating system or another file navigation tool. We put the operations you could perform on a file in a dropdown menu in a way that users should already be familiar with. We named all of actions in the traditional way and we allowed the use of keyboard buttons to perform actions that we knew our users would expect. Examples of this would be selecting a file and pressing the delete key on the keyboard would immediately bring up a confirmation dialog for the deletion of the selected file. The feel of the design is business-like and snappy.

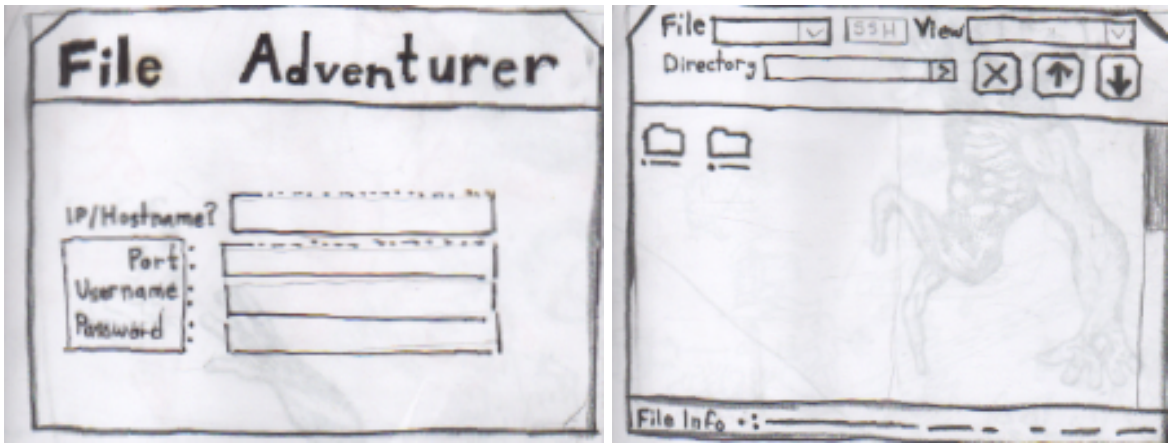
Minimizing Interface

The most important aspect of our interface design was simplicity. We know that with simplicity can come ease and speed of use. We expect our users to perform routine actions with our software, such and moving and deleting files, on a regular basis. Because our users will have to do the same things quite often, we wanted those actions to take as few steps as possible. Most actions in File adventurer do not take more than 4 clicks to execute. Deleting, renaming, downloading, and moving can all be performed in 4 actions or less. The interface we made is not flashy or confusing. We even removed our delete button on the toolbar when we discovered people were confusing it with an exit button. We removed it and put a logout button in its place, where people seem to expect it to be. In the current implementation we

have also removed the SSH button because we could not get that feature implemented. There are only a couple visible buttons in our interface by default and when dropdowns and dialogs are expanded. We feel that the simplicity in our interface's layout and functionality provide a maximized "ease-of-use".

Design and Features

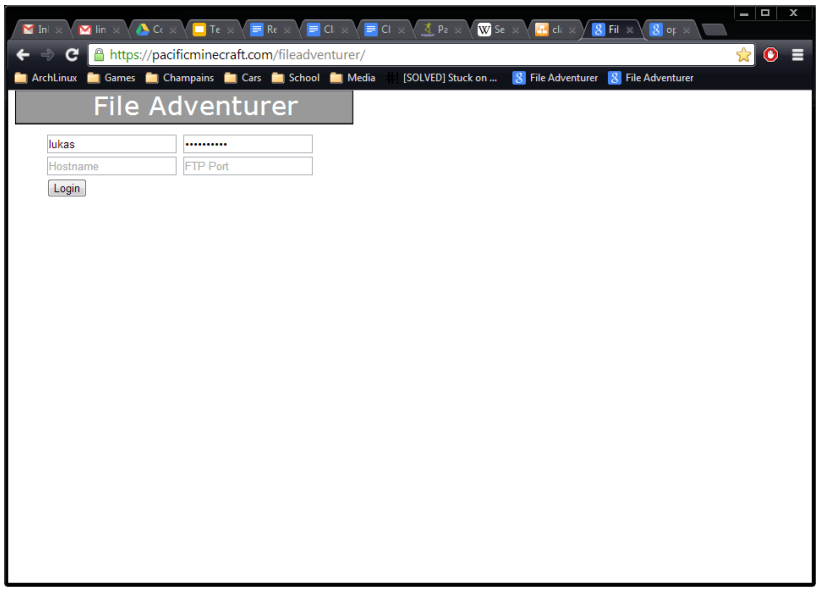
This is one of the first designs we had planned. In all our development we did not diverge very far from the original idea we had for our project.



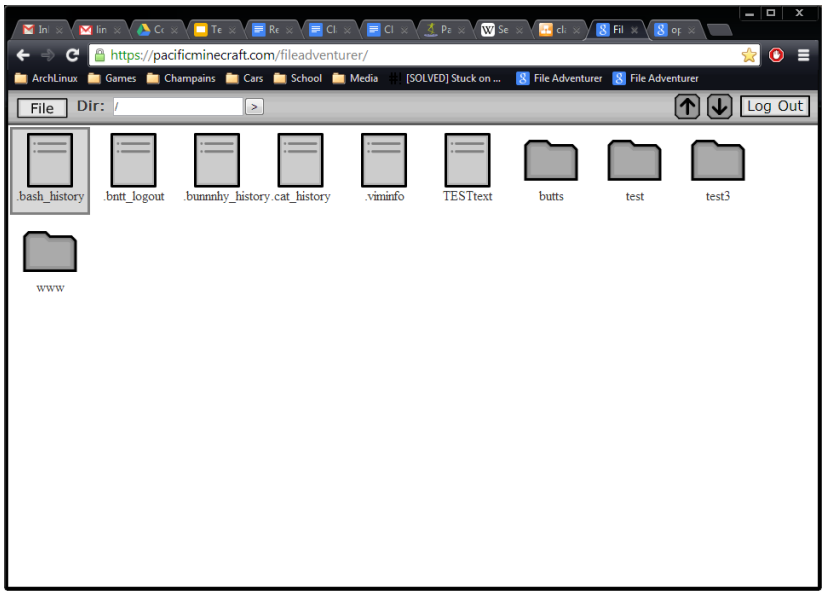
Here you can see that we originally planned to have different file view types, but that was set aside as a secondary feature. Minimal, and functional in what we have.

Interface use examples:

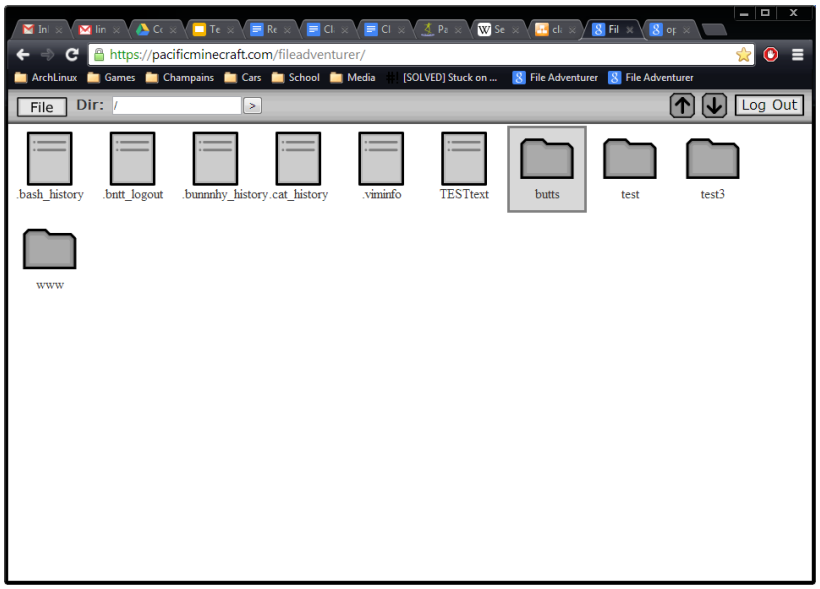
On the following pages I will show how the current interface is used.



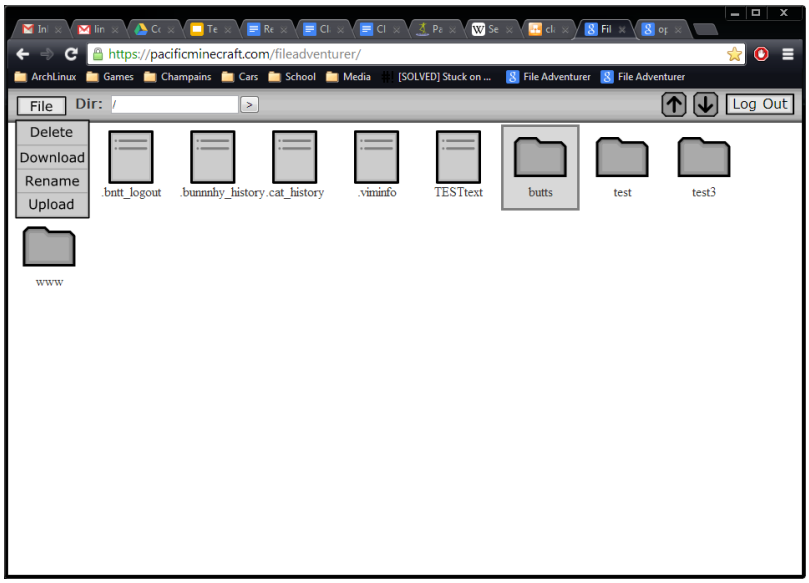
Here is where you login after entering your username and password. It is as simple as logging into your server normally, if not simpler.



Once you are logged in you can click any file to select it.

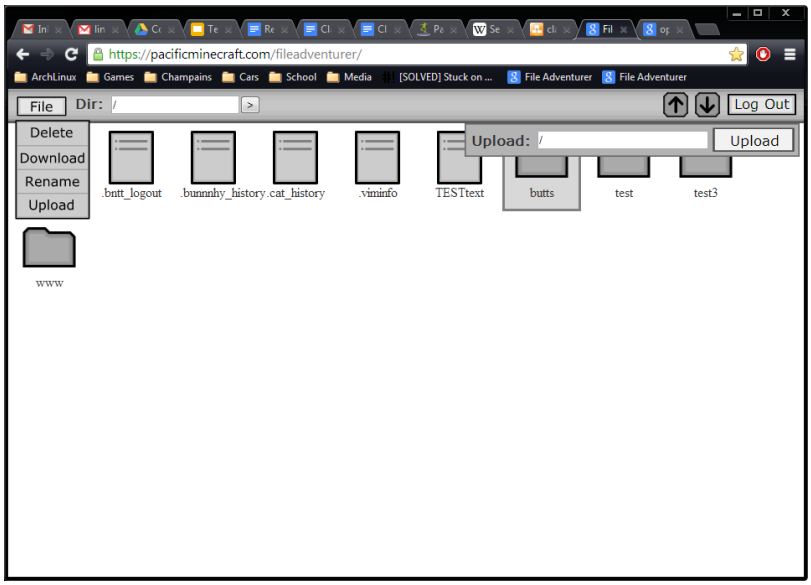


You can double click any folder to navigate into its directory.

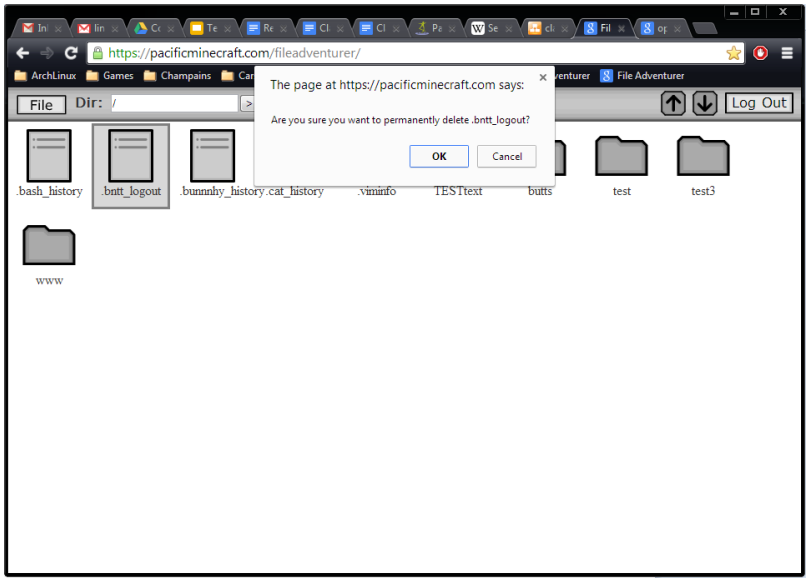


With any file selected you can select any of the options in the file menu.

Renaming will make the file text editable. Then it only requires pressing the enter key to change the name.



Clicking the upload button will open a dialog that will allow you to enter the path of the file you wish to upload to your current directory. In the future we would like to implement a system browse button.



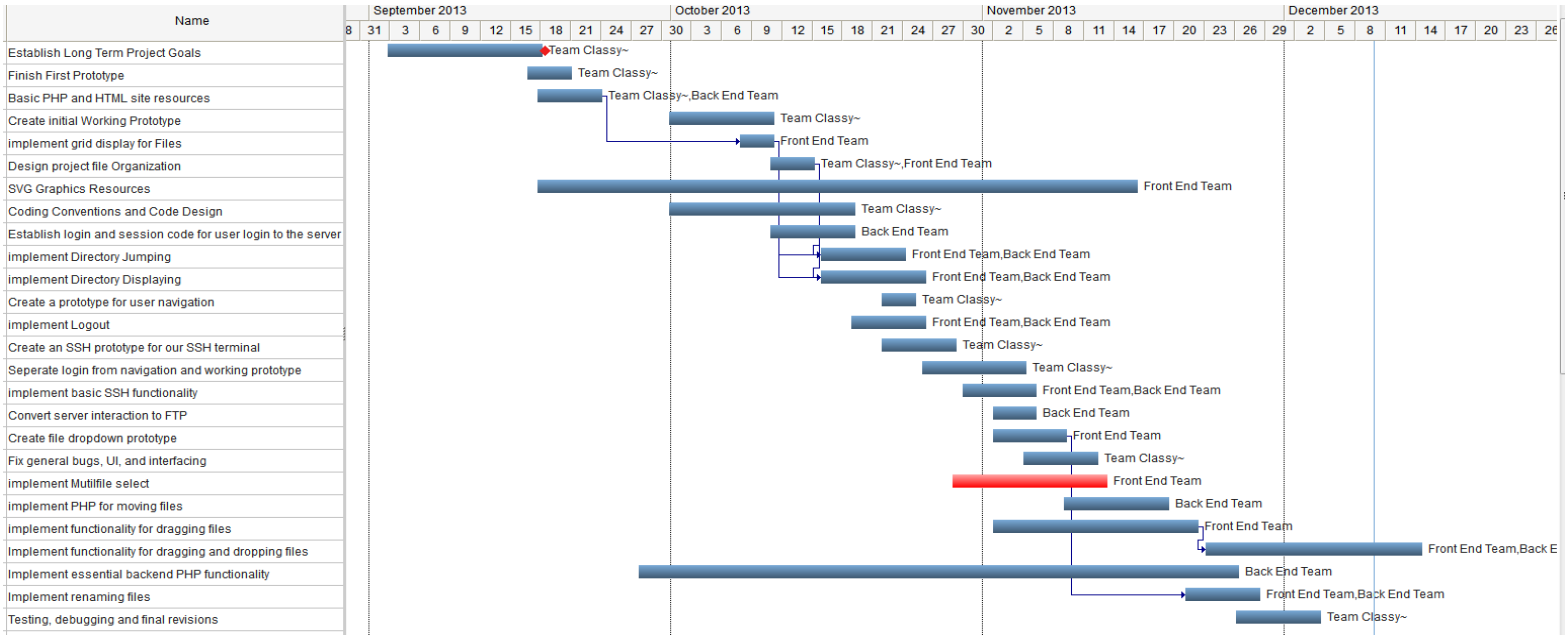
Clicking the delete button will open a confirmation dialog that will tell you if you are deleting multiple files and confirm your desire to have them deleted.

History of Work

Order of Events

- Finished long term goals September 17th
- Basic Prototype September 20th
- Basic PHP and site code September 30th
- Working prototype October 10th
- Finished floating SVG implementation of “grid”
- Designed project file organization October 13th
- Finished Most UI SVGs October 15th
- Designed conventions and finished initial design planning. October 18th
- Basic login and session code Implemented and organized October 19th (Backend implement)
- Directory Jumping October 19th
- Full Directory displaying October 22nd
- Prototype Navigation October 24th
- Logout October 25th
- SSH prototype October 27th (Finished interactive prototype)
- Separated login from navigation and Working prototype Nov 3rd
- Backend basic SSH Functionality November 5th
- Switched functionalities to FTP Nov 5th
- File Dropdown prototype November 9th
- Fixed bugs and interface. (Semifinal Prototype) November 10th
- File movement PHP finished November 17th
- Finished dragging November 21st
- Dragging and dropping kinda works November 23rd
- Essential Backend PHP finished November 25th
- Renaming implemented November 27th
- Deleting and multi file select implemented December 3rd
- MANY BUGS FIXED Almost finished product bugs maintained December 3rd

Gantt Chart



Team Classy~: Combination of anyone on the team
Front end team: Lukas Richards, Nicholas Howes, Joseph Grant
Back end team: Connor Morales, Bill Trok

Accomplishments

- Made User interface SVGS
- Designed JSON file list format
- Made login.php
- Made list_dir.php
- Made separate javascript and css files
- Implemented click handlers on files
- Made jQuery file objects
- Polished UI and removed confusing elements
- Created session code
- Made mv_file.php
- Implemented directory displaying
- Implemented directory jumping
- Implemented directory navigation by clicking
- Implemented file selection
- Conceded to FTP
- Got file dragging to work
- Implemented file renaming
- Implemented file deleting
- Implemented file downloading
- Got uploading working

Conclusion and future work

Connor Morales -

Technical Challenges - Before this project, I hadn't used PHP much. It was simple to learn the syntax and basics, but a language with as many functions as PHP is very difficult to master, not that I have any desire to do so. There are many problems with PHP that I would not expect to see from such a popular language that is mostly used for web development. Some of these problems are lack of/bad support for MIME, FTP, and SSH. It is not possible to keep connections such as FTP and SSH persistent over more than one script.

Skills Learned - I have more experience with PHP now, for whatever that's worth. I have never done process forking with a scripting language before. It was fairly straightforward. I can't really say I learned much else from this project, as I've worked on group projects before.

What would have helped - Perl or Ruby

Future Work - I really want to get SSH working. Other than that, there is quite a lot of code reorganization I could do.

Joseph Grant -

Technical Challenges - Having no prior experience with anything website or server related besides a joke of a class in High School on web design using dreamweaver was my largest hurdle in participating in the project. A little more than normal programming, there are a lot of methods that can be employed to reach the same goal in javascript. Trying to determine what method was the easiest and simplest way to reach a certain goal was usually harder than actually implementing it, as long as you chose the way that was simplest and easiest.

Skills Learned – Learning javascript and in general about how to utilize javascript for dom manipulation and event handling was what this project has given me. I do not doubt that there is a lot I still do not know, however, in general javascript seems like a language where you pick different functionalities almost as often as you implement them and learning how to deal with that was helpful to me. Becoming familiar with CSS and HTML5 was also neat although that all seemed pretty straightforward and a robust knowledge of the two seems unnecessary to accomplish most everything. Also even though I was not responsible for them I found the backend to frontend interaction and coding convention docs were hugely helpful to me understanding the basic functionality of the features of our application before they were implemented and while they were being implemented.

What would have helped – I wish I knew how to use Sublime Text and then maybe I could have done better at keeping up with our coding conventions. Other than that what really would have helped was previous experience of javascript and a more robust understanding of its role in interaction with servers, users, and the webpage document. I also think if a better way of testing our project could have been very helpful. Our only way of testing our javascript along with its interaction with PHP was to actually implement our changes on the server. However we used GIT to catch any complications so we were not nearly as bad of as the Tiger Gamez group.

Future Work – Our current visual are for the most part intuitive and simple, bar the

redundant and to some confusing upload and download buttons. However, it would be nice to have better visuals and tool tips like most applications have for the functionality of our features. We already have the necessary information being passed in to add in things such as our File Info for example, and we could add context menus for things such as double clicking files to download them because it is not something that is completely obvious. Past updating these things to address some of the comments we got about our presentation I would want to learn how to use Inkscape to create graphics and make other changes to the interface of our site like making the svg images not draggable and not selectable. I will probably never learn Inkscape, it is less intuitive and functional than GIMP is and I have no clue if it accepts any sort of plug ins that would make it acceptable. PHP also seems like a valuable thing to learn if I hope to implement anything further than prettiness on the front end.

Lukas Rickard -

Technical Challenges - Dragging and dropping with storing element coordinates for each file was a challenge as we found out. To defeat this problem we ended up storing the name and path of each file in the SVG that was generated to represent them. In general it was also difficult to implement features along with the code that Nicholas wrote. Because Nicholas had much more experience with javascript than me he wrote functions and statements I didn't fully understand. It was difficult at a few points to merge my code with someone elses, but we stayed very organized and used both a server and Git to keep our files organized, so it was not too messy when I was doing merges, which I am thankful for.

Skills Learned - I knew nothing about javascript before this class, but it was interesting to learn it as my first scripting language. I am currently in the process of learning python for another class and I can see how there are parallels in event distant scripting languages. I learned a lot about managing my server as well from Connor. I learned about security and how it can help force organization and order. It was nice to finally learn how to use git. I had used mercurial before, so it was easy to use the same commands, but there are some differences in branching and merging between the two that is vital. I did not learn very much about PHP, but I did learn some. It's not a very pretty language and I may try to avoid it in the future.

What would have helped - It would have helped to have had any prior knowledge of HTML or CSS going into making this app, but there was not too much we had to do with it, beyond the initial design. I think this project would have benefitted if we had known that FTP was the best idea and I could have learned more about it to help on the back end. In general I believe knowledge about how to make element bound tracking code would have been very helpful for many projects with intersecting box detection. If we had used bounding boxes or something to detect which file was over which it would have been simple to extend our system to dragging and dropping multiple files.

Future Work - I think I would like to change the dragging and dropping code in the future to allow the HTML to be cleaner and to allow the interface to show which file the user is hovering their current file over. This could be done using by storing element bounds and checking them on mouse move. When this system is implemented it would also be natural to continue work on the interface by writing code that would allow users to drag a transparent box over multiple files to select many at once. If I work on this project more I would also like to add

more color and view options. Future work can be done on the aesthetics or on the smoothness and proper implementation of the interface.

Nicholas Howes -

Technical Challenges - The in-browser SSH feature ended up being more complicated than I had anticipated, and ended up needing to use features of browsers that do not have standard implementations in javascript. Were I to return to this in the future, I would look for a text manipulation library that abstracted all of that away.

Skills Learned - Although I had previous javascript experience, I had never used jQuery, which is an important javascript tool, as it simplifies many common tasks. I learned a little bit about how to use linux, although I was not involved with much of the back-end work. I also learned how to use git.

What would have helped - Using jQuery UI would have made several things much simpler.

Future Work - It would be nice to give the whole thing a visual spruceing up, and improve the user experience by adding things like context menus that allow them several different options for doing certain tasks.

William Trok -

Technical Challenges - The biggest challenge was getting started in the right direction, there were often too many ways to approach each little problem, and the resources available often proscribed different paths to do basically the same thing. One example of this was downloading, while downloading a file from the server to the client side using HTTPS is actually quite straightforward, the goal of our project was for our server to access a remote server and pass on the files to client. There were many ways to do this, additionally while clearly it would be ideal if we could get a download going from FTP -> Client as opposed to FTP -> Server -> Client, there were many challenges and gigantic issues with all ways I attempted to do this, in the end I went with the less efficient, but more straightforward way.

Other big challenges where getting the server to run in a manner that did not leave you tearing out your ears, (eyes, etc.) while waiting for it to load. Out initial version of it, took well over 2 minutes to load the files in the home directory on my laptop. We were not completely successful in get the loading time down as much as we may have wanted, but did succeed in speeding it up roughly 4-5 times getting it down to about 30 seconds.

Skills Learned - Learning git was a much bigger undertaking then I had anticipated. Learned quite a bit about both php, unix directory structure and JSON, as well as general web development.

What would have helped - Finding and using pre-exisiting libraries, This is one of the biggest issues. There were serveral instances where me and connor both saw a problem and thought “Oh, this is easy I can write a couple functions to do this no problem.” We would write the functions then spent hours debugging them, only for them to break weeks later. Eventually we ended up replacing the functions we wrote with functions from some php library that did

exactly what we wanted. This is in addition to the fact that the other person had to spend time learning what each function did instead of simply looking at the php documentation. Long story short, looking for pre-exisiting libraries and using them as much as possible would have saved a ton of development time.

Future Work - Spend time speeding up file data retrieval, it still takes 30 seconds to get and return all the files in the home directory on my laptop. I believe changing the way we handle time can greatly speed things up. Additionally, some effort should be made to get the file adventurer working with ftp servers that run different operating systems, such as windows.

References

Our Github Repository - https://github.com/TeamClassy/fileadventurer

FTP- File Transfer Protocol - http://tools.ietf.org/html/rfc959

jQuery- http://jquery.com/

Gantter App- https://www.smartapp.com/

PacificMineCraft Server - https://pacificminecraft.com/fileadventurer/

Use case Diagram drawing software - https://www.draw.io/

Javascript and CSS learning Resource - http://www.w3schools.com