TNPG: JiangJiangWangWang
Roster: Kevin Wang (PM), Hui Wang, Marc Jiang, Ian Jiang
Soft Dev
2023-6-3

ABSTRACT:
Our project is called **Ducky Drive**. Our idea is to implement a copy of Google Drive, using the **lab machines** as the storage backend. This will enable easy access to your files on the lab machines, and functionality like uploading, downloading, viewing, editing, deleting, and creating files will make **file management as abstract as possible**. Additionally, Ducky Drive will have the option of opening an ssh terminal on the frontend for power users.

Preface
There are **TWO** concurrent, independent SSH connections
1. **Connection A**: One is created on the flask server and acts as the central **point of abstraction** for Ducky Drive
2. **Connection B**: The other is created by a node script and is responsible for creating a **terminal on the frontend** for power users to use. For example, if they want to use the CLI to edit their files, they will have the option of opening this terminal and directly communicating with the lab computers on the website.

Program List

Connection A
1. **__init__.py** - Basic driver with Flask. Create routes that allow authenticated users to perform different actions on their file system.

2. Fileman.py - handles opening ssh connection and sending commands to the ssh server. Communicates with __init__.py to perform actions. Utilizes the paramiko package to do so.

3. static/css/
   a. Style.css - Formatting that FEF doesn't take care of
   b. Xterm.css - formatting of the ssh terminal for frontend

4. static/js/ -
   a. Filemanager.js - Act as middleware to query flask server routes when a user demands it on the front-end, such as when a button is clicked.

5. templates/ - our HTML files for the respective pages we need
    a. File_system.html - displays the current file system
    b. Login.html - login functionality
    c. Search.html - see results of a search
    d. Files.html - contains all files in given directory
    e. Folders.html - contains all directories within given directory

Connection B

6. static/js/server.js
    a. creates a frontend ssh terminal end for power users to use

7. Server.js - creates the ssh connection on the backend for the frontend terminal to exchange user input and terminal output across the frontend to the backend
    a. Requires express, websocket, term.js, ssh2
8. Terminal.py - automatically start node server for the terminal

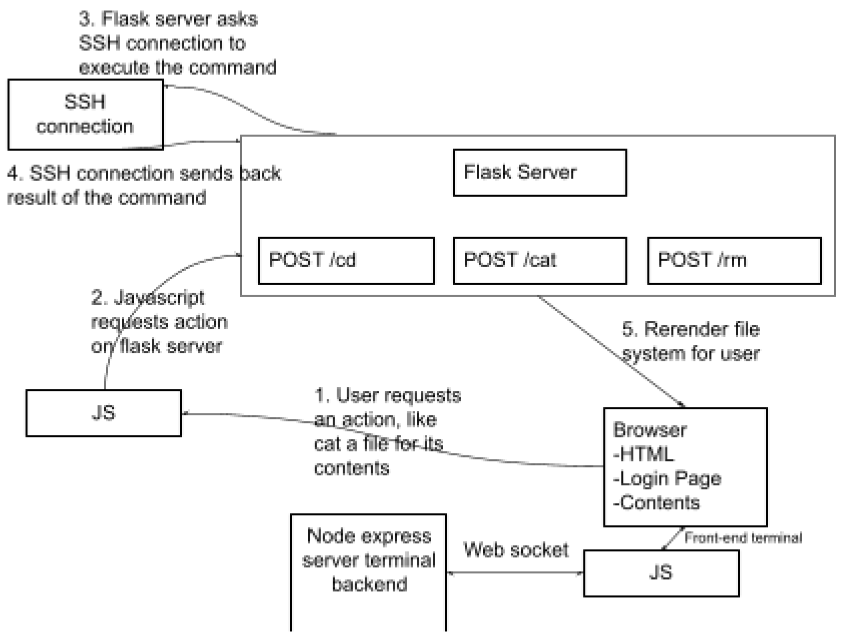
Database Organization

No database required! Storage is managed on the lab machines, and no user information is stored. Downloading files works without a DB because we can get the contents of the file using the terminal, and allow the contents to be downloaded in place without actually having the file in our possession. Reverse for upload -> turn the file into a string of its contents and then write it to a file in place.

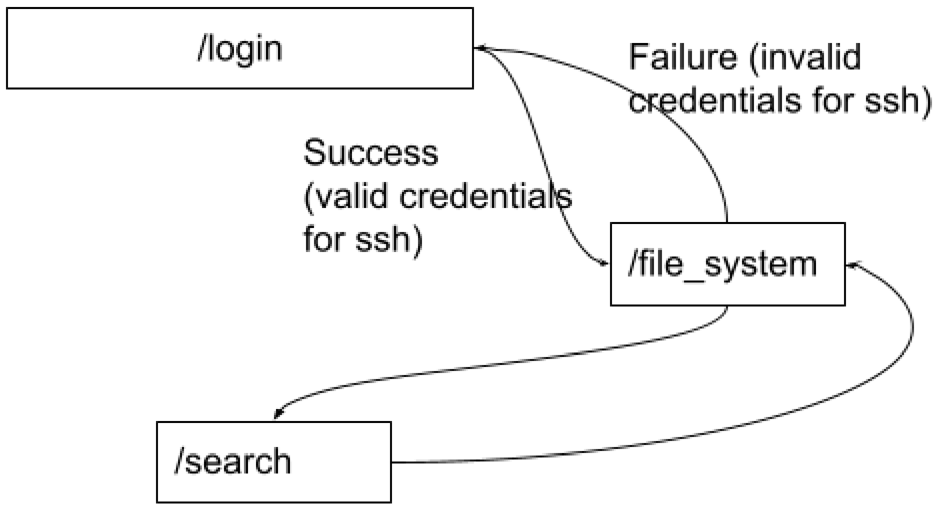
We also do not need to store the login information. When a user logs in, we create an ssh connection with their credentials, and their username is stored as a session cookie to ensure that only that user can interact with that particular ssh connection. No username/password information needs to be verified past that point.

API Section:
No APIs needed

Component Map


3. Flask server asks SSH connection to execute the command
SSH connection
4. SSH connection sends back result of the command
Flask Server
POST /cd
POST /cat
POST /rm
2. Javascript requests action on flask server
5. Rerender file system for user
JS
1. User requests an action, like cat a file for its contents
Browser
-HTML
-Login Page
-Contents
Front-end terminal
Node express server terminal backend
Web socket
JS

Site Map (Front-end)


/login
Failure (invalid credentials for ssh)
Success (valid credentials for ssh)
/file_system
/search

Assignments to Members

Marc:

- Uploading files
- Moving folders
- Renaming folders

Kevin:

- File manipulation
- Directory manipulation

Ian:

- Downloading files
- Droplet management
- File previews

Hui:

- Front-end

MVP:

- [x] ~~Upload files from local computer to server~~
- [x] ~~Download from server to local computer~~
- [x] ~~File manipulation~~
  - [x] ~~Viewing~~
  - [x] ~~Moving~~
  - [x] ~~Renaming~~
  - [x] ~~Deleting~~
  - [x] ~~Creating~~
  - [x] ~~Editing~~
- [x] ~~Directory manipulation~~
  - [x] ~~Renaming~~
  - [x] ~~Deleting~~
  - [x] ~~Creating~~
  - [x] ~~Entering ***~~

Stretch Goals:

- [x] ~~Web based ssh'ed terminal for power users who need more functionality~~
- [x] ~~Search filesystem for a particular file~~
- [x] ~~Code editor when editing files~~
- [x] ~~Previewing different file types~~
  - [x] ~~pdfs~~
  - [x] ~~images~~
- [x] ~~Sharing files with other users~~
  - [ ] Investigated: Not possible without root access
- [ ] Advanced filtering
  - [ ] By extension
  - [ ] By date modified

FEF Features Why and How?

Why:

- Bootstrap is easier to work with & looks cleaner.
- Bootstrap containers improve responsiveness and user experience
- Foundation is difficult to work with

How:

- Bootstrap modals can help with previewing and editing files
- Mobile responsiveness