WebOS Terminal

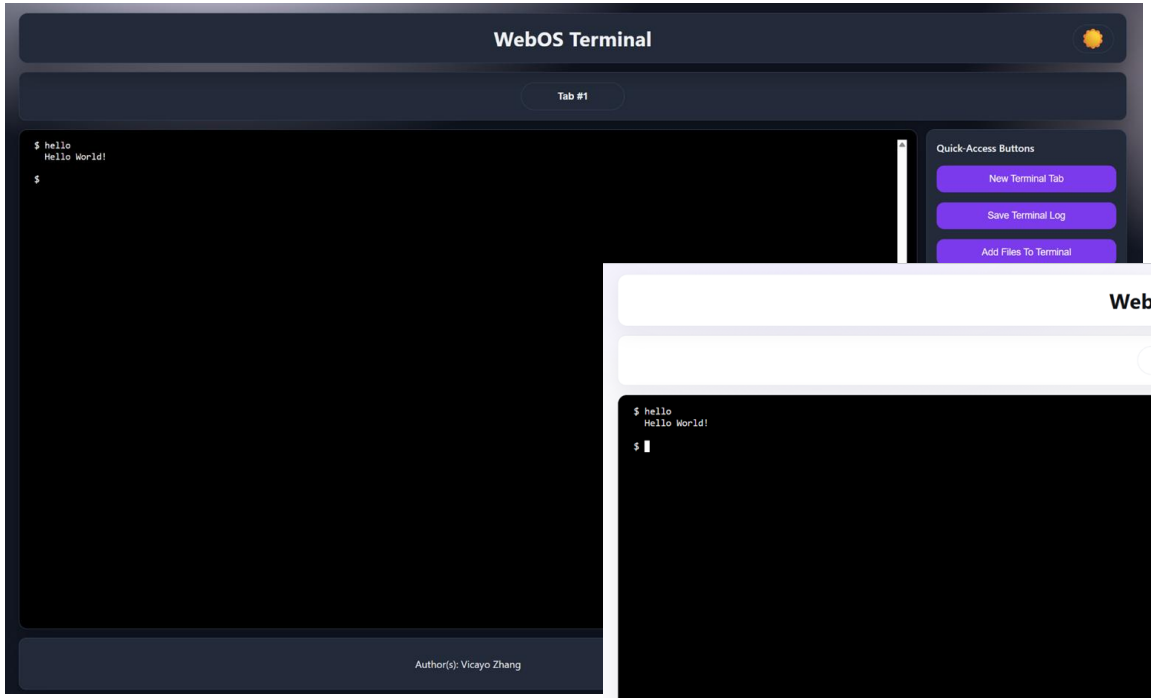
Vicayo Zhang

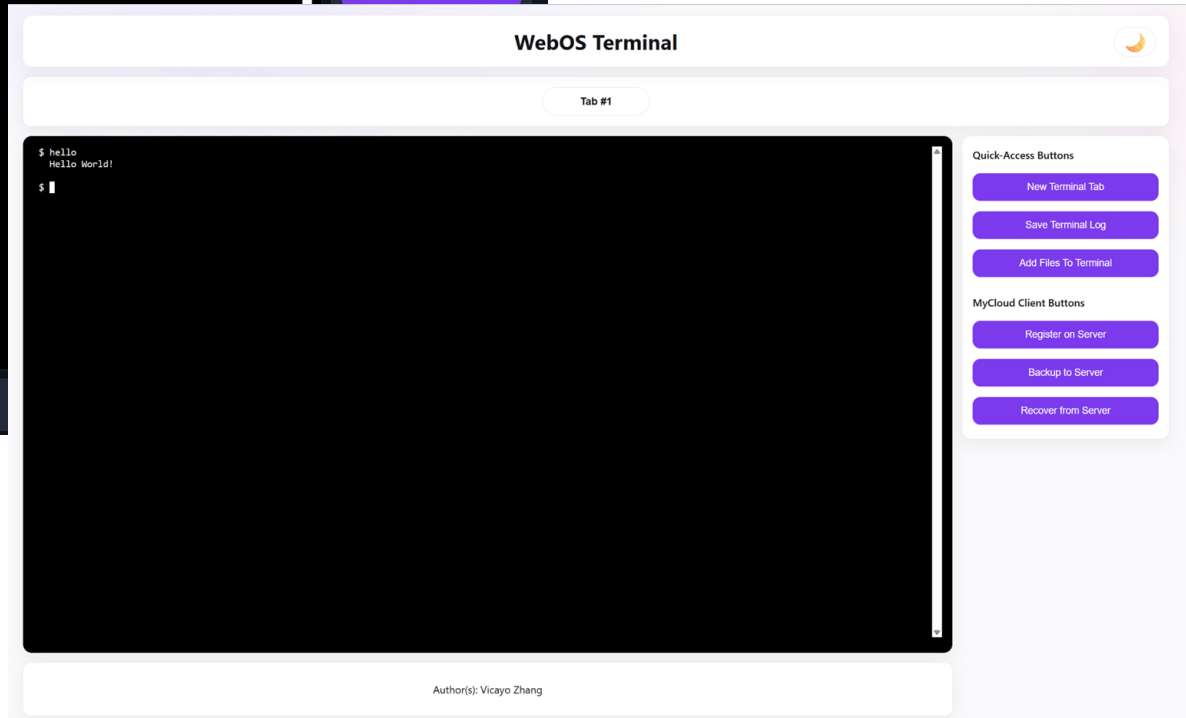
https://github.com/VicayoMua/WebOS_Terminal

Client-Side

```
<head>

    <meta charset="UTF-8"/>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>

    <title>WebOS Terminal</title>

    <!-- Style Sheets -->
    <link rel="stylesheet" href="stylesheets/index.css"/>
    <link rel="stylesheet" href="stylesheets/buttons.css"/>
    <link rel="stylesheet" href="stylesheets/xterm.css"/>
    <link rel="stylesheet" href="stylesheets/ace_editor_popup.css"/>
    <link rel="stylesheet" href="stylesheets/media_player_popup.css"/>
    <link rel="stylesheet" href="stylesheets/alert_popup.css"/>
    <link rel="stylesheet" href="stylesheets/mycloud_popup.css"/>

    <!-- Terminal Core Scripts -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/ace/1.4.12/ace.js"></script>
    <script type="module" src="terminal_setup.js"></script>

</head>
```

```html
<div class="main-page">

    <div class="header">
        <h1>WebOS Terminal</h1>
        <button type="button" id="button-to-switch-theme">🌙</button>
    </div>

    <nav class="view-navigation" id="view-navigation">
        <!--
                    <button type="button" style="font-weight: bold;">{ Tab #1 }</button>
        -->
    </nav>

    <div class="terminal-window-frames" id="terminal-window-frames">
        <!--
                    <div class="terminal-window-frame current-tab">
                        <div dir="ltr" class="terminal xterm xterm-dom-renderer-owner-1"> ... </div>
                        <div class="ace-editor-window"> ... </div>
                    </div>
        -->
    </div>

    <div class="additional-buttons">

        <h2 class="quick-access-buttons-title"> Quick-Access Buttons </h2>
        <button class="button primary-button" type="button" id="button-to-open-new-terminal-tab">New Terminal Tab</button>
        <button class="button primary-button" type="button" id="button-to_save-terminal-log">Save Terminal Log</button>
        <button class="button primary-button" type="button" id="button-to-add-files-to-terminal">Add Files To Terminal</button>

        <h2 class="mycloud-buttons-title"> MyCloud Client Buttons </h2>
        <button class="button primary-button" type="button" id="button-to-register-on-mycloud-server">Register on Server</button>
        <button class="button primary-button" type="button" id="button-to-backup-to-mycloud-server">Backup to Server</button>
        <button class="button primary-button" type="button" id="button-to-recover-from-mycloud-server">Recover from Server</button>

    </div>

    <div class="footer">
        <p>Author(s): Vicayo Zhang</p>
    </div>

</div>
```

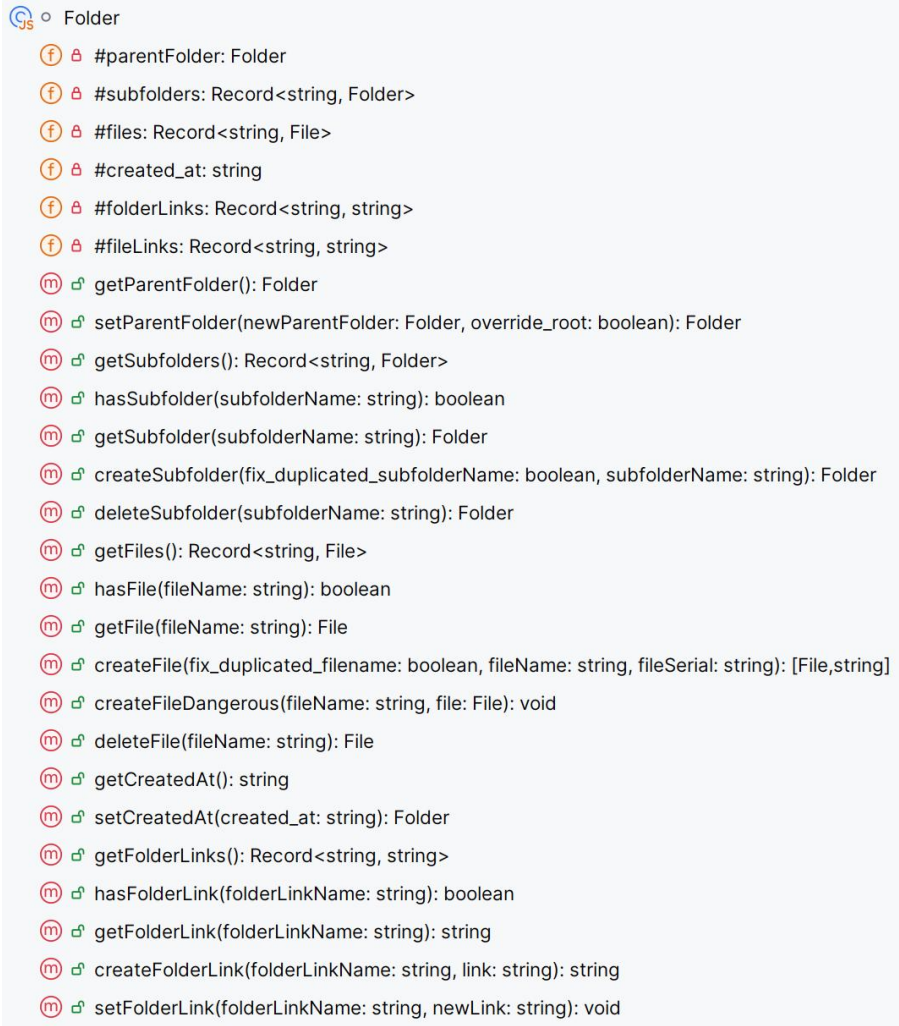

WebOS Terminal
Tab #1
$ hello
Hello World!
$
Quick-Access Buttons
New Terminal Tab
Save Terminal Log
Add Files To Terminal
Author(s): Vicayo Zhang


WebOS Terminal
Tab #1
$ hello
Hello World!
$
Quick-Access Buttons
New Terminal Tab
Save Terminal Log
Add Files To Terminal
MyCloud Client Buttons
Register on Server
Backup to Server
Recover from Server
Author(s): Vicayo Zhang

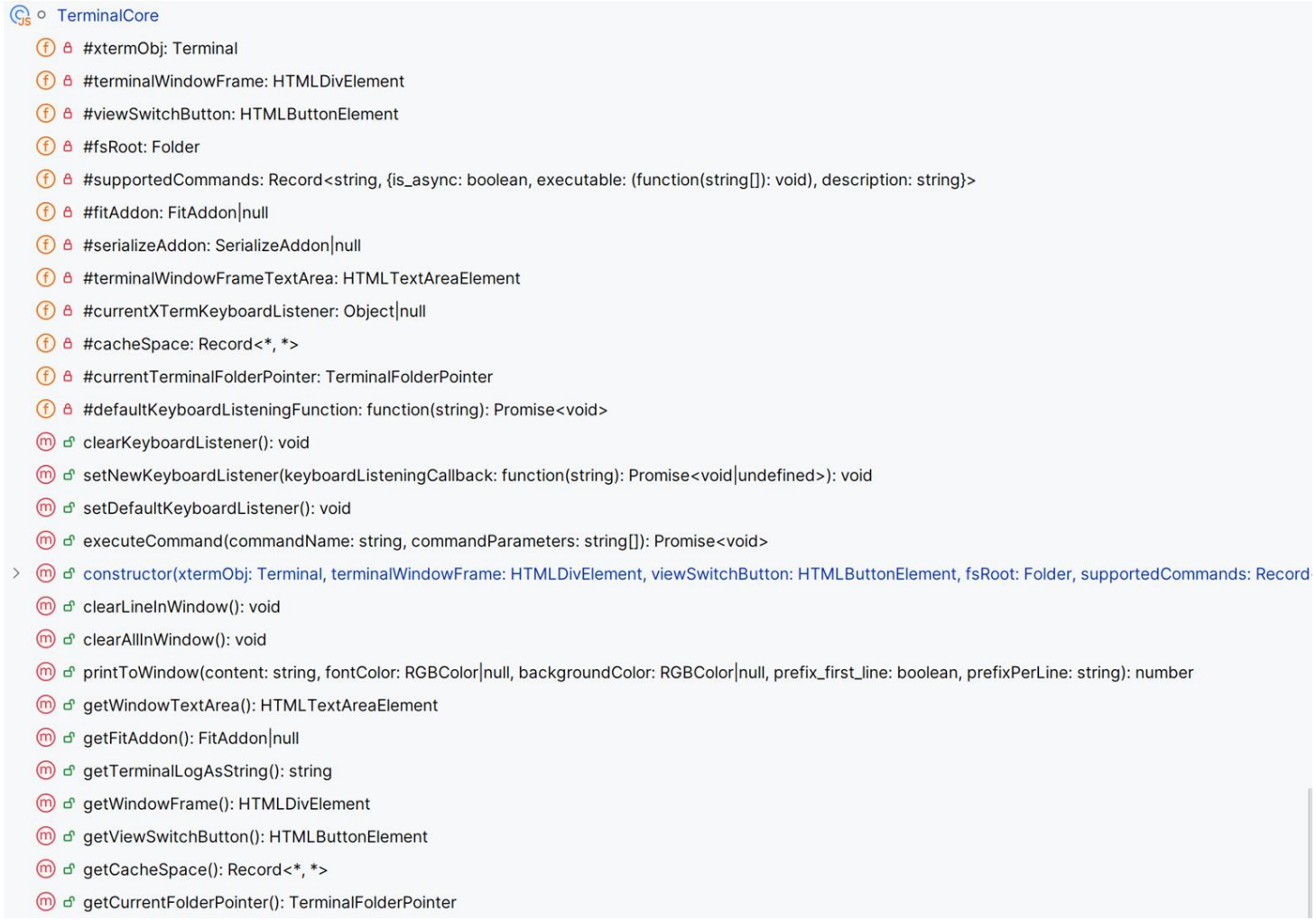
```
legalFileSystemKeyNameRegExp = /^(?!\.{1,2}$)[^\/\0\b\r]{1,1024}$/,
legalFileSerialRegExp = /^(?:ROOT|[A-Za-z_][A-Za-z0-9_]{127,4096})$/,
/**
 * This function extracts the directory-path and key-name from <path>.
 * @param {string} path
 * @returns {[string, string]} [fileFolderPath, fileName]
 * @throws {TypeError}
 * */
extractDirAndKeyName = (path) => {
    if (typeof path !== 'string')
        throw new TypeError('path must be a string');
    const index = path.lastIndexOf('/');
    if (index === -1) // <path> is a single filename
        return ['.', path];
    if (index === 0) // file is in the ROOT
        return ['/', path.slice(1)];
    //     [   parent directory path, immediate key name   ]
    return [path.substring(0, index), path.substring(index + 1)];
};
```

```
/**
 * This structure represents the space, which serial numbers are picked from.
 * Methods may throw Errors due to illegal inputs.
 * */
class SerialLake {  Show usages   vicayo +1 *
    /** @type {Set<string>} */
    #serialSet : Set<string> ;
    /** @type {function():string} */
    #serialGenerator;

    /**
     * @param {function():string} fileSerialGenerator
     * @throws {TypeError}
     * */
    constructor(fileSerialGenerator) {...}

    /**
     * This method recovers <serialSet> with <fileSerials>, overwriting all previously-saved information.
     * @param {string[]} fileSerials
     * @returns {SerialLake}
     * @throws {TypeError}
     * */
    recover(fileSerials : string[] ) : SerialLake  {...}

    /**
     * This method generates a new _unique_ serial number.
     * @returns {string}
     * @throws {Error}
     * */
    generateNext() : string  {...}
}
```


File
#fileSerial: string
#content: ArrayBuffer
#created_at: string
#updated_at: string
constructor(fileSerial: string, content: ArrayBuffer, created_at: string|undefined, updated_at: string|undefined)
duplicate(fileSerial: string): File
getSerial(): string
getContent(): ArrayBuffer
getCreatedAt(): number
getUpdatedAt(): number
setContent(newContent: ArrayBuffer, do_copy: boolean): File
extractContentsAsZip(serialLake: SerialLake): Promise<Folder>


Folder
#parentFolder: Folder
#subfolders: Record<string, Folder>
#files: Record<string, File>
#created_at: string
#folderLinks: Record<string, string>
#fileLinks: Record<string, string>
getParentFolder(): Folder
setParentFolder(newParentFolder: Folder, override_root: boolean): Folder
getSubfolders(): Record<string, Folder>
hasSubfolder(subfolderName: string): boolean
getSubfolder(subfolderName: string): Folder
createSubfolder(fix_duplicated_subfolderName: boolean, subfolderName: string): Folder
deleteSubfolder(subfolderName: string): Folder
getFiles(): Record<string, File>
hasFile(fileName: string): boolean
getFile(fileName: string): File
createFile(fix_duplicated_filename: boolean, fileName: string, fileSerial: string): [File,string]
createFileDangerous(fileName: string, file: File): void
deleteFile(fileName: string): File
getCreatedAt(): string
setCreatedAt(created_at: string): Folder
getFolderLinks(): Record<string, string>
hasFolderLink(folderLinkName: string): boolean
getFolderLink(folderLinkName: string): string
createFolderLink(folderLinkName: string, link: string): string
setFolderLink(folderLinkName: string, newLink: string): void
deleteFolderLink(folderLinkName: string): string
getFileLinks(): Record<string, string>
hasFileLink(fileLinkName: string): boolean
getFileLink(fileLinkName: string): string
createFileLink(fileLinkName: string, link: string): string
setFileLink(fileLinkName: string, newLink: string): void
deleteFileLink(fileLinkName: string): string
clear(): Folder
constructor(is_root: boolean, parentFolder: Folder|undefined)
isRoot(): boolean
getContentListAsString(): string
getFilesAsList(): File[]
getRecordsJSON(): string
getZipBlobPromise(): Promise<Blob>


TerminalFolderPointer
#fsRoot: Folder
#currentFolder: Folder
#currentFolderTrackingStack: string[]
constructor(fsRoot: Folder, currentFolder: Folder, currentFullPathStack: string[])
duplicate(): TerminalFolderPointer
recover(another: TerminalFolderPointer, deep_copy_stack: boolean): TerminalFolderPointer
getCurrentFolder(): Folder
getFullPath(): string
gotoRoot(): Folder
gotoParentFolder(): Folder
#gotoSubfolder(subfolderName: string, include_link: boolean): Folder
gotoPath(path: string, include_link: boolean): Folder
createPath(path: string, goto_new_folder: boolean): Folder
movePath(type: "file"|"directory", oldPath: string, newPath: string, include_link: boolean): TerminalFolderPointer
copyPath(type: "file"|"directory", oldPath: string, newPath: string, serialLake: SerialLake, include_link: boolean): TerminalFolderPointer
deletePath(type: "file"|"directory", path: string): TerminalFolderPointer

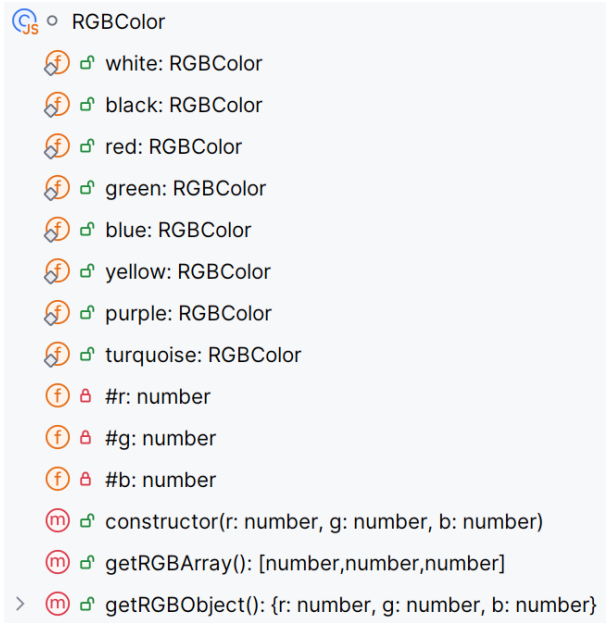

TerminalCore
#xtermObj: Terminal
#terminalWindowFrame: HTMLDivElement
#viewSwitchButton: HTMLButtonElement
#fsRoot: Folder
#supportedCommands: Record<string, {is_async: boolean, executable: (function(string[]): void), description: string}>
#fitAddon: FitAddon|null
#serializeAddon: SerializeAddon|null
#terminalWindowFrameTextArea: HTMLTextAreaElement
#currentXTermKeyboardListener: Object|null
#cacheSpace: Record<*, *>
#currentTerminalFolderPointer: TerminalFolderPointer
#defaultKeyboardListeningFunction: function(string): Promise<void>
clearKeyboardListener(): void
setNewKeyboardListener(keyboardListeningCallback: function(string): Promise<void|undefined>): void
setDefaultKeyboardListener(): void
executeCommand(commandName: string, commandParameters: string[]): Promise<void>
constructor(xtermObj: Terminal, terminalWindowFrame: HTMLDivElement, viewSwitchButton: HTMLButtonElement, fsRoot: Folder, supportedCommands: Record
clearLineInWindow(): void
clearAllInWindow(): void
printToWindow(content: string, fontColor: RGBColor|null, backgroundColor: RGBColor|null, prefix_first_line: boolean, prefixPerLine: string): number
getWindowTextArea(): HTMLTextAreaElement
getFitAddon(): FitAddon|null
getTerminalLogAsString(): string
getWindowFrame(): HTMLDivElement
getViewSwitchButton(): HTMLButtonElement
getCacheSpace(): Record<*, *>
getCurrentFolderPointer(): TerminalFolderPointer

```
this.#defaultKeyboardListeningFunction = async (keyboardInput : string ) : Promise<void>  => {
    switch (keyboardInput) {
        // case '\x1bOP': { // F1 (ignored)...
        case '\x1b[A': { // Up Arrow
            if (previousCommands.length <= 0)
                break;
            if (indexPrevComm === -1 || indexPrevComm === 0) {
                indexPrevComm = previousCommands.length - 1;
            } else {
                indexPrevComm--;
            }
            const
                [commandName, commandParameters] = previousCommands[indexPrevComm],
                command = commandParameters.reduce((acc, str) : string  => `${acc} ${str.indexOf(' ') === -1 ? str : `'${str}'`}`, commandName);
            this.clearLineInWindow();
            this.printToWindow( content: ` $ ${command}`);
            bufferReset();
            bufferAddString(command);
            break;
        }
        case '\x1b[B': { // Down Arrow
            if (previousCommands.length <= 0)
                break;
            if (indexPrevComm === -1 || indexPrevComm === previousCommands.length - 1) {
                indexPrevComm = 0;
            } else {
                indexPrevComm++;
            }
            const
                [commandName, commandParameters] = previousCommands[indexPrevComm],
                command = commandParameters.reduce((acc, str) : string  => `${acc} ${str.indexOf(' ') === -1 ? str : `'${str}'`}`, commandName);
            this.clearLineInWindow();
            this.printToWindow( content: ` $ ${command}`);
            bufferReset();
            bufferAddString(command);
            break;
        }
```

```
        case '\u007F': { // Backspace
            if (bufferRemoveChar()) { // if the char is successfully removed from the buffer
                this.printToWindow( content: '\b \b');
            }
            break;
        }
        case '\r': { // Enter
            if (buffer.length <= 0) break;
            const [commandName : string , commandParameters : string[] ] = bufferAnalyzeCommandNameParameters();
            bufferReset();
            previousCommands.push([commandName, commandParameters]);
            indexPrevComm = -1;
            await this.executeCommand(commandName, commandParameters);
            break;
        }
        case '\u000C': { // Ctrl+L
            this.clearAllInWindow();
            this.printToWindow( content: ` $ ${bufferToString()}`);
            break;
        }
        case '\u0016': { // Ctrl+V
            const clipText : string  = await navigator.clipboard.readText();
            const safeKeyboardInput : string  = clipText.replace( searchValue: /[^ -~]/g,  replaceValue: '');
            this.printToWindow(safeKeyboardInput);
            bufferAddString(safeKeyboardInput);
            break;
        }
        default: {
            // delete every character outside the printable ASCII range
            const safeKeyboardInput : string  = keyboardInput.replace( searchValue: /[^ -~]/g,  replaceValue: '');
            this.printToWindow(safeKeyboardInput);
            bufferAddString(safeKeyboardInput);
        }
    }
};
```

```
/**
 * @param {string} commandName
 * @param {string[]} commandParameters
 * @returns {Promise<void>}
 * */
async executeCommand(commandName : string , commandParameters : string[] ) : Promise<void>  {  Show usages  vicayo
    if (typeof commandName !== 'string')
        throw new TypeError('CommandName must be a string.');
    if (!Array.isArray(commandParameters) || !commandParameters.every((str : string ) => typeof str === 'string' && str.length > 0))
        throw new TypeError('CommandParameters must be an array of non-empty strings.');
    this.clearKeyboardListener();
    // exit of kernel mode
    this.printToWindow( content: '\n');
    const commandObject : {...}  = this.#supportedCommands[commandName];
    if (commandObject === undefined) {
        this.printToWindow( content: `Command "${commandName}" is not found.`, RGBColor.red);
    } else {
        try {
            if (commandObject.is_async === true) { // must check "=== true"
                await commandObject.executable(commandParameters);
            } else {
                commandObject.executable(commandParameters);
            }
        } catch (error) {
            this.printToWindow( content: `Command "${commandName}" failed due to uncaught errors. <-- ${error}`, RGBColor.red);
        }
    }
    this.printToWindow( content: '\n\n $ ',  fontColor: null,  backgroundColor: null,  prefix_first_line: false,  prefixPerLine: '');
    // enter of kernel mode
    this.setDefaultKeyboardListener();
}
```

```
printToWindow( Show usages  vicayo *
    content : string ,
    fontColor : RGBColor | null  = null,
    backgroundColor : RGBColor | null  = null,
    prefix_first_line : boolean  = false,
    prefixPerLine : string  = '    '
) : number  {
    if (typeof content !== 'string' || content.length === 0)
        throw new TypeError('content must be a non-empty string.');
    // if (content.indexOf('\r') !== -1)
    //     throw new TypeError(`content must not include '\\r'.`);
    if (typeof prefix_first_line !== 'boolean')
        throw new TypeError(`prefix_first_line must be a boolean.`);
    if (typeof prefixPerLine !== 'string')
        throw new TypeError('prefixPerLine must be a string.');
    if (prefixPerLine.indexOf('\b') !== -1 || prefixPerLine.indexOf('\r') !== -1 || prefixPerLine.indexOf('\n') !== -1)
        throw new TypeError(`prefixPerLine must not include '\\b', '\\r', or '\\n'.`);
    // prefix the content string
    content = content.replaceAll( searchValue: '\n',  replaceValue: `\n\r${prefixPerLine}`);
    if (prefix_first_line) content = prefixPerLine + content;
    // read font color
    if (fontColor !== null) {
        if (!(fontColor instanceof RGBColor))
            throw new TypeError(`fontColor must be an RGBColor or null.`);
        // add fontColor as a presetting
        const [fr : number , fg : number , fb : number ] = fontColor.getRGBArray();           // reset font color
        content = `\x1b[38;2;${fr};${fg};${fb}m` + content;
    }
    // read background color
    if (backgroundColor !== null) {
        if (!(backgroundColor instanceof RGBColor))
            throw new TypeError(`backgroundColor must be an RGBColor or null.`);
        // add backgroundColor as a presetting
        const [br : number , bg : number , bb : number ] = backgroundColor.getRGBArray();     // reset background color
        content = `\x1b[48;2;${br};${bg};${bb}m` + content;
    }
    // reset the styles
    content = content + `\x1b[0m`;                                                          // reset to default
    // write to window object
    this.#xtermObj.write(content);
    return content.length;
}
```



```
/**
 * This function parses <object> to <FormData>
 * @param {Object} object
 * @returns {FormData}
 * @throws {TypeError}
 * */
formData = (object : Object ) : FormData  => {...},
/**
 * @param {string} ipp
 * @param {string} userKey
 * @returns {Promise<void>}
 * @throws {TypeError | Error}
 * */
registerUserKeyToMyCloud = async (ipp : string , userKey : string ) : Promise<void>  => {...},
/**
 * @param {string} ipp
 * @param {string} userKey
 * @returns {Promise<void>}
 * @throws {TypeError | Error}
 * */
verifyMyCloudSetup = async (ipp : string , userKey : string ) : Promise<void>  => {...},
/**
 * @param {string} ipp
 * @param {string} userKey
 * @param {Folder} fsRoot
 * @returns {Promise<void>}
 * @throws {TypeError | Error}
 * */
backupFSToMyCloud = async (ipp : string , userKey : string , fsRoot : Folder ) : Promise<void>  => {...},
/**
 * @param {string} ipp
 * @param {string} userKey
 * @param {Folder} fsRoot
 * @param {SerialLake} serialLake
 * @returns {Promise<void>}
 * @throws {TypeError | Error}
 * */
recoverFSFromMyCloud = async (ipp : string , userKey : string , fsRoot : Folder , serialLake : SerialLake ) : Promise<void>  => {...};
```

```
_supportedCommands_['hello'] = {
    is_async: false,
    executable: (_ : string[] ) : void  => {
        currentTerminalCore.printToWindow( content: `Hello World!`);
    },
    description: `Say 'Hello World!'`
};
```

```
/**
 * This function pops up an alert window.
 * If primaryButtonText.length === 0, then the button will not appear!
 * @param {HTMLElement} parentHTMLElement
 * @param {string} alertTitle
 * @param {string} alertMessage
 * @param {boolean} display_primary_button
 * @param {boolean} display_secondary_button
 * @param {string} primaryButtonText
 * @param {(function():void) | null} callbackPrimaryButton
 * @param {string} secondaryButtonText
 * @param {(function():void) | null} callbackSecondaryButton
 * @returns {{primary: function():void, secondary: function():void}}
 * @throws {TypeError}
 * */
popupAlert = (
    parentHTMLElement,
    alertTitle, alertMessage,
    display_primary_button = true, display_secondary_button = false,
    primaryButtonText = 'Confirm', callbackPrimaryButton = null,
    secondaryButtonText = 'Cancel', callbackSecondaryButton = null
) => {...},
```

```
_supportedCommands_['edit'] = {
    is_async: true,
    executable: async (parameters : string[] ) : Promise<void>  => {
        if (parameters.length === 1) {
            try {
                const
                    [fileDir : string , fileName : string ] = extractDirAndKeyName(parameters[0]),
                    file : File  = currentTerminalCore.getCurrentFolderPointer() TerminalFolderPointer
                        .duplicate()
                        .gotoPath(fileDir) Folder
                        .getFile(fileName),
                    fileContent : ArrayBuffer  = file.getContent();
                await new Promise( executor: (resolve) : void  => {
                    popupFileEditor(
                        currentTerminalCore.getWindowFrame(),
                        fileName,
                        utf8Decoder.decode(fileContent),
                        callbackToSaveFile: (newFileContent : string ) : void  => { // save
                            file.setContent(utf8Encoder.encode(newFileContent).buffer,  do_copy: false);
                            currentTerminalCore.printToWindow( content: ` --> Updated a text file.`, RGBColor.yellow);
                            resolve( value: undefined);
                            currentTerminalCore.getWindowTextArea().focus();
                        },
                        callbackToCancelEdit: () : void  => { // cancel
                            currentTerminalCore.printToWindow( content: ` --> Canceled updates on a text file.`, RGBColor.yellow);
                            resolve( value: undefined);
                            currentTerminalCore.getWindowTextArea().focus();
                        }
                    );
                    currentTerminalCore.printToWindow( content: ` --> Opened a text editor.\n`, RGBColor.green);
                });
            } catch (error) {
                currentTerminalCore.printToWindow( content: `${error}`, RGBColor.red);
            }
            return;
        }
        currentTerminalCore.printToWindow( content: `Wrong grammar!\nUsage: edit [file_path]`, RGBColor.red);
    },
    description: 'Edit an existing file.\n' +
        'Usage: edit [file_path]'
};
```

```
/**
 * This function pops up the editor window for the <edit> command.
 * @param {HTMLDivElement} terminalWindowFrame
 * @param {string} fileName
 * @param {string} orginalFileContent
 * @param {function(newFileContent: string): void} callbackToSaveFile
 * @param {function():void} callbackToCancelEdit
 * @returns {void}
 * @throws {TypeError}
 * */
popupFileEditor = (
    terminalWindowFrame,
    fileName, orginalFileContent,
    callbackToSaveFile, callbackToCancelEdit
) => {...},
```

WebOS Terminal

Tab #1

Quick-Access Buttons

New Terminal Tab

Save Terminal Log

Add Files To Terminal

MyCloud Client Buttons

Register on Server

Backup to Server

Recover from Server

Author(s): Vicayo Zhang

```
videoTypes = {
    '.mp4': 'video/mp4',
    '.m4v': 'video/mp4',
    '.mov': 'video/quicktime',
    '.webm': 'video/webm',
    '.mkv': 'video/x-matroska',
    '.ogv': 'video/ogg',
    '.ogg': 'video/ogg',
    '.hevc': 'video/hevc'
},
audioTypes = {
    '.mp3': 'audio/mpeg',
    '.aac': 'audio/aac',
    '.m4a': 'audio/mp4',
    '.oga': 'audio/ogg',
    '.opus': 'audio/ogg',
    '.wav': 'audio/wav',
    '.flac': 'audio/flac',
    '.aiff': 'audio/aiff',
    '.aif': 'audio/aiff'
},
pictureTypes = {
    '.jpg': 'image/jpeg',
    '.jpeg': 'image/jpeg',
    '.png': 'image/png',
    '.gif': 'image/gif',
    '.webp': 'image/webp',
    '.avif': 'image/avif',
    '.svg': 'image/svg+xml',
    '.bmp': 'image/bmp',
    '.ico': 'image/x-icon',
    '.tiff': 'image/tiff',
    '.tif': 'image/tiff'
},
documentTypes = {
    '.pdf': 'application/pdf',
    '.html': 'text/html',
    '.htm': 'text/html',
    '.xml': 'text/xml',
},
```

```
/**
 * This function pops up the media player window for the <omedia> command.
 * @param {HTMLDivElement} terminalWindowFrame
 * @param {string} mediaFileName
 * @param {ArrayBuffer} mediaFileContent
 * @param {function():void} callbackToCloseMediaPlayer
 * @returns {void}
 * @throws {TypeError | Error}
 * */
popupMediaPlayer = (
    terminalWindowFrame,
    mediaFileName,
    mediaFileContent,
    callbackToCloseMediaPlayer
) => {...};
```

```
_supportedCommands_['media'] = {
    is_async: true,
    executable: async (parameters : string[] ) : Promise<void>  => {
        if (parameters.length === 1) {
            try {
                const
                    mediaFilePath : string  = parameters[0],
                    [mediaFileFolderPath : string , mediaFileName : string ] = extractDirAndKeyName(mediaFilePath),
                    mediaFileContent : ArrayBuffer  = currentTerminalCore.getCurrentFolderPointer().duplicate() TerminalFolderPointer
                        .gotoPath(mediaFileFolderPath) Folder
                        .getFile(mediaFileName) File
                        .getContent();
                await new Promise( executor: (resolve) : void  => {
                    popupMediaPlayer(
                        currentTerminalCore.getWindowFrame(),
                        mediaFileName,
                        mediaFileContent,
                        callbackToCloseMediaPlayer: () : void  => {
                            resolve( value: undefined);
                            currentTerminalCore.getWindowTextArea().focus();
                        }
                    );
                    currentTerminalCore.printToWindow( content: ` --> Opened a media player.`, RGBColor.green);
                });
            } catch (error) {
                currentTerminalCore.printToWindow( content: `${error}`, RGBColor.red);
            }
            return;
        }
        currentTerminalCore.printToWindow(
            content: 'Wrong grammar!\n' +
            'Usage: media [media_file_path]',
            RGBColor.red
        );
    },
    description: 'Open a media file in a separate window.\n' +
        'Supported video format:   .mp4/.m4v/.mov/.webm/.mkv/.ogv/.ogg/.hevc\n' +
        '          audio format:   .mp3/.aac/.m4a/.oga/.opus/.wav/.flac/.aiff/.aif\n' +
        '          picture format: .jpg/.jpeg/.png/.gif/.webp/.avif/.svg/.bmp/.ico/.tiff/.tif\n' +
        'Usage: media [media_file_path]'
}
```

WebOS Terminal

Tab #1

$ help

Playing Media File: pic.jpg

× Close

Quick-Access Buttons

New Terminal Tab

Save Terminal Log

Add Files To Terminal

MyCloud Client Buttons

Register on Server

Backup to Server

Recover from Server

Author(s): Vicayo Zhang

```
$ help
  This terminal supports:
       hello, help, man, echo, touch, mkdir,
       ls, cd, pwd, mv, cp, rm,
       edit, printf, download, wget, zip, media.

  For more details, please use the command 'man [command_name]'.

$ man man
   --> A detailed manual of the terminal simulator.
       Usage: man [command_name]

$ man mkdir
   --> Make a new directory.
       Usage: mkdir [folder_path]

$ man cp
   --> Copy an existing file or directory.
       Usage: cp -f [original_file_path] [destination_file_path]
              cp -d [original_directory_path] [destination_directory_path]

$ man mv
   --> mv an existing file or directory.
       Usage: mv -f [original_file_path] [destination_file_path]
              mv -d [original_directory_path] [destination_directory_path]

$ man edit
   --> Edit an existing file.
       Usage: edit [file_path]

$ man touch
   --> Make a new file in the current directory.
       Usage: touch [file_path]

$ man download
   --> Download a single file or a directory (as .zip file) from the terminal file system to your local machine.
       Usage: download -f [file_path]
              download -d [directory_path]

$
```

```
$ man cd
   --> Goto the given folder.
       Usage: cd [folder_path]

$ pwd
  /

$ mkdir fd1
   --> Created a directory, OR the directory may already exist.

$ mkdir fd2
   --> Created a directory, OR the directory may already exist.

$ ls
  Folders:
              fd1
              fd2

$ cd fd1
   --> Went to a directory.

$ pwd
  /fd1

$ ls
  No file or folder existing here...

$
```

```
$ ls
  No file or folder existing here...

$ touch 1.txt
   --> Created a file.

$ printf 1.txt
  <EMPTY FILE>

$ edit 1.txt
   --> Opened a text editor.
   --> Updated a text file.

$ printf 1.txt
  THIS IS A TEXT FILE.

$ download -f 1.txt
   --> Downloaded a file.

$ ls
  Files:
              1.txt

$
```

```
$ ls
  No file or folder existing here...

$ mkdir fd1
   --> Created a directory, OR the directory may already exist.

$ cd fd
  Error: Path fd not found. <-- Error: Folder fd not found.

$ touch /fd/1.txt
   --> Created a file.

$ ls
  Folders:
              fd1
              fd

$ ls /fd
  Files:
              1.txt

$ cp -d /fd /fd_copied
   --> Copied a directory.

$ ls
  Folders:
              fd1
              fd
              fd_copied

$ ls /fd_copied
  Files:
              1.txt

$
```

```
$ ls
  Folders:
              fd1
              fd
              fd_copied

$ rm -d /fd
   --> Removed a directory.

$ ls
  Folders:
              fd1
              fd_copied

$ ls /fd1
  No file or folder existing here...

$ ls fd_copied
  Files:
              1.txt

$ rm -f /fd_copied/1.txt
   --> Removed a file.

$ ls fd_copied
  No file or folder existing here...

$
```

```
$ ls
  No file or folder existing here...

$ man wget
   --> Download a single file from a URL to a path.
       Usage: wget [URL] [file_path]

$ wget https://www.shutterstock.com/image-photo/awesome-pic-natureza-600nw-2408133899.jpg 1.png
   --> Downloaded a file.

$ ls
  Files:
              1.png

$ man zip
   --> Create a zip file from a folder path.
       Usage: zip [zip_file_path] [folder_path]

$ zip img.zip /
   --> Created a zip file.

$ ls
  Files:
              1.png
              img.zip

$
```

```
const
    /** @type {number} */
    MAX_TAB_COUNT : number  = 40,
    /** @type {HTMLButtonElement} */
    button_to_switch_theme : HTMLButtonElement  = document.getElementById( elementId: 'button-to-switch-theme'),
    /** @type {HTMLDivElement} */
    div_terminal_window_frames : HTMLDivElement  = document.getElementById( elementId: 'terminal-window-frames'),
    /** @type {HTMLElement} */
    nav_view_navigation : HTMLElement  = document.getElementById( elementId: 'view-navigation'),
    /** @type {HTMLButtonElement} */
    button_to_open_new_terminal_tab : HTMLButtonElement  = document.getElementById( elementId: 'button-to-open-new-terminal-tab'),
    /** @type {HTMLButtonElement} */
    button_to_save_terminal_log : HTMLButtonElement  = document.getElementById( elementId: 'button-to_save-terminal-log'),
    /** @type {HTMLButtonElement} */
    button_to_add_files_to_terminal : HTMLButtonElement  = document.getElementById( elementId: 'button-to-add-files-to-terminal'),
    /** @type {HTMLButtonElement} */
    button_to_register_on_mycloud_server : HTMLButtonElement  = document.getElementById( elementId: 'button-to-register-on-mycloud-server'),
    /** @type {HTMLButtonElement} */
    button_to_backup_to_mycloud_server : HTMLButtonElement  = document.getElementById( elementId: 'button-to-backup-to-mycloud-server'),
    /** @type {HTMLButtonElement} */
    button_to_recover_from_mycloud_server : HTMLButtonElement  = document.getElementById( elementId: 'button-to-recover-from-mycloud-server');

button_to_switch_theme.addEventListener( type: 'click',  listener: () : void  => {...});

if (localStorage.getItem( key: 'mini-terminal-dark-body-mode-enabled') === 'true') {...}

button_to_open_new_terminal_tab.addEventListener( type: 'click',  listener: () : void  => {...});
button_to_open_new_terminal_tab.click(); // automatically open Tab #1

button_to_save_terminal_log.addEventListener( type: 'click',  listener: () : void  => {...});

button_to_add_files_to_terminal.addEventListener( type: 'click',  listener: () : void  => {...});

if (localStorage.getItem( key: 'mini-terminal-mycloud-ipp') === null)
    localStorage.setItem('mini-terminal-mycloud-ipp', '127.0.0.1:80');
if (localStorage.getItem( key: 'mini-terminal-mycloud-user-key') === null)
    localStorage.setItem('mini-terminal-mycloud-user-key', '');

button_to_register_on_mycloud_server.addEventListener( type: 'click',  listener: () : void  => {...});

button_to_backup_to_mycloud_server.addEventListener( type: 'click',  listener: () : void  => {...});

button_to_recover_from_mycloud_server.addEventListener( type: 'click',  listener: () : void  => {...});
```

WebOS Terminal

Tab #1 | Tab #2 | Tab #3 | Tab #4 | Tab #5 | Tab #6 | Tab #7

$

Quick-Access Buttons

- New Terminal Tab
- Save Terminal Log
- Add Files To Terminal

MyCloud Client Buttons

- Register on Server
- Backup to Server
- Recover from Server

Author(s): Vicayo Zhang

WebOS Terminal

terminal_log @ 2025-12-05T16_06_19.910Z.txt
183 B • Done

Full download history

Tab #1

```
$ hello
  Hello World!

$ ls
  No file or folder existing here...

$ man man
   --> A detailed manual of the terminal simulator.
       Usage: man [command_name]

$
```

Quick-Access Buttons

New Terminal Tab

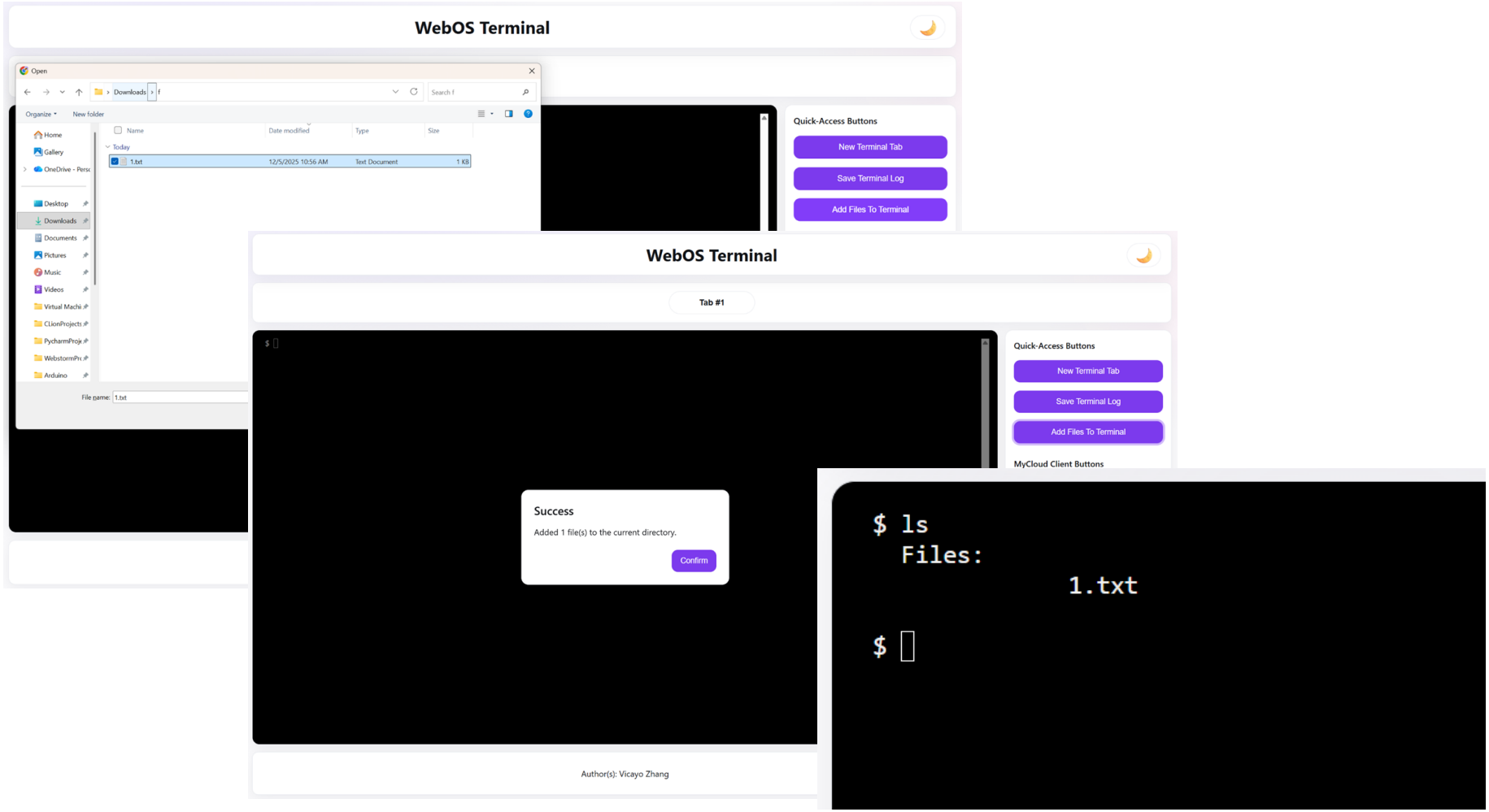
Save Terminal Log

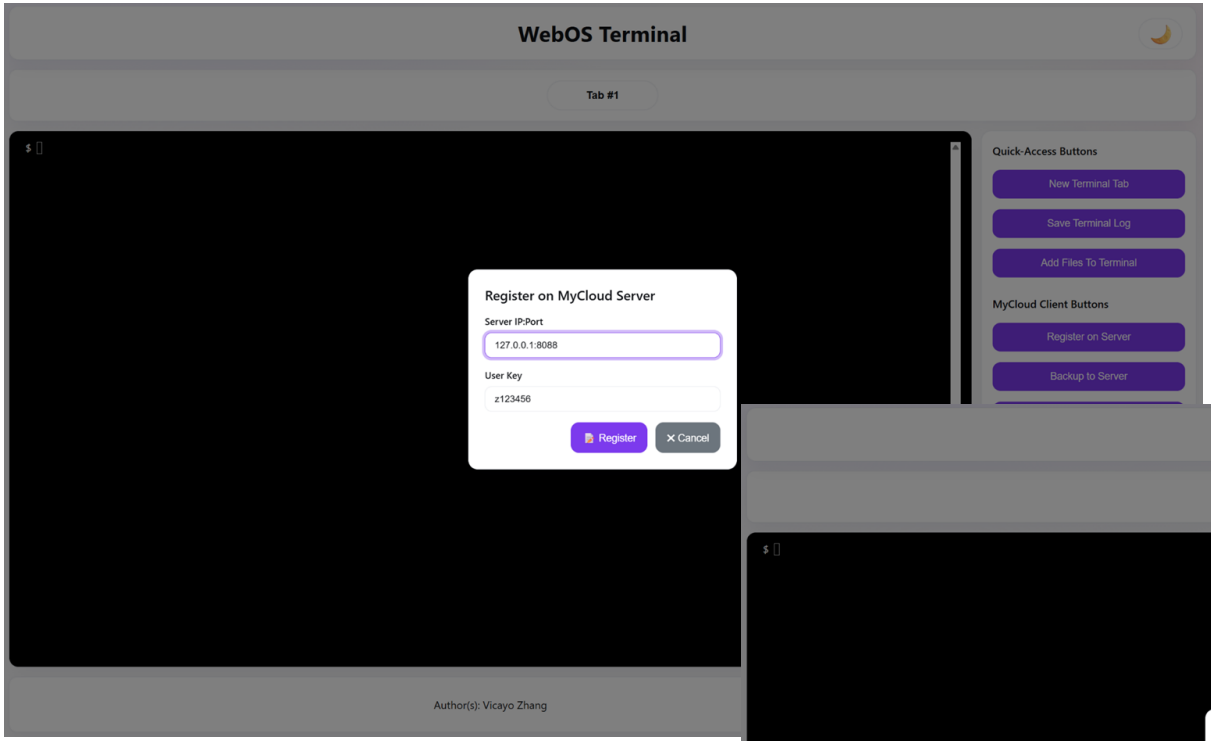
terminal_log @ 2025-12-05T16_06_

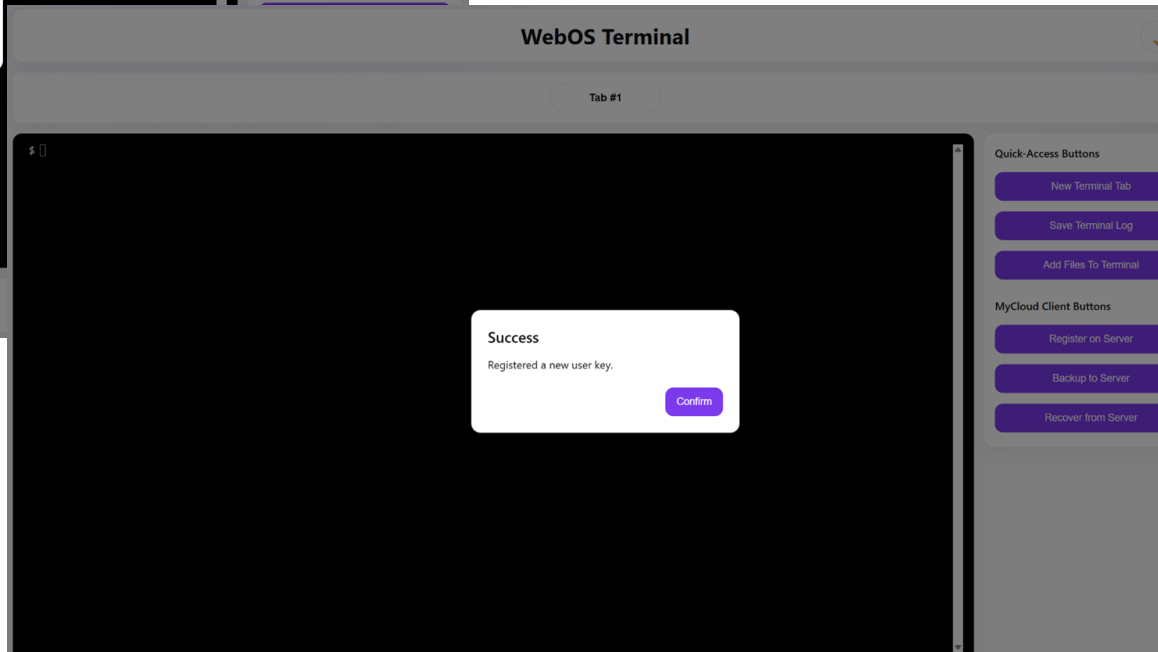
File Edit View H1

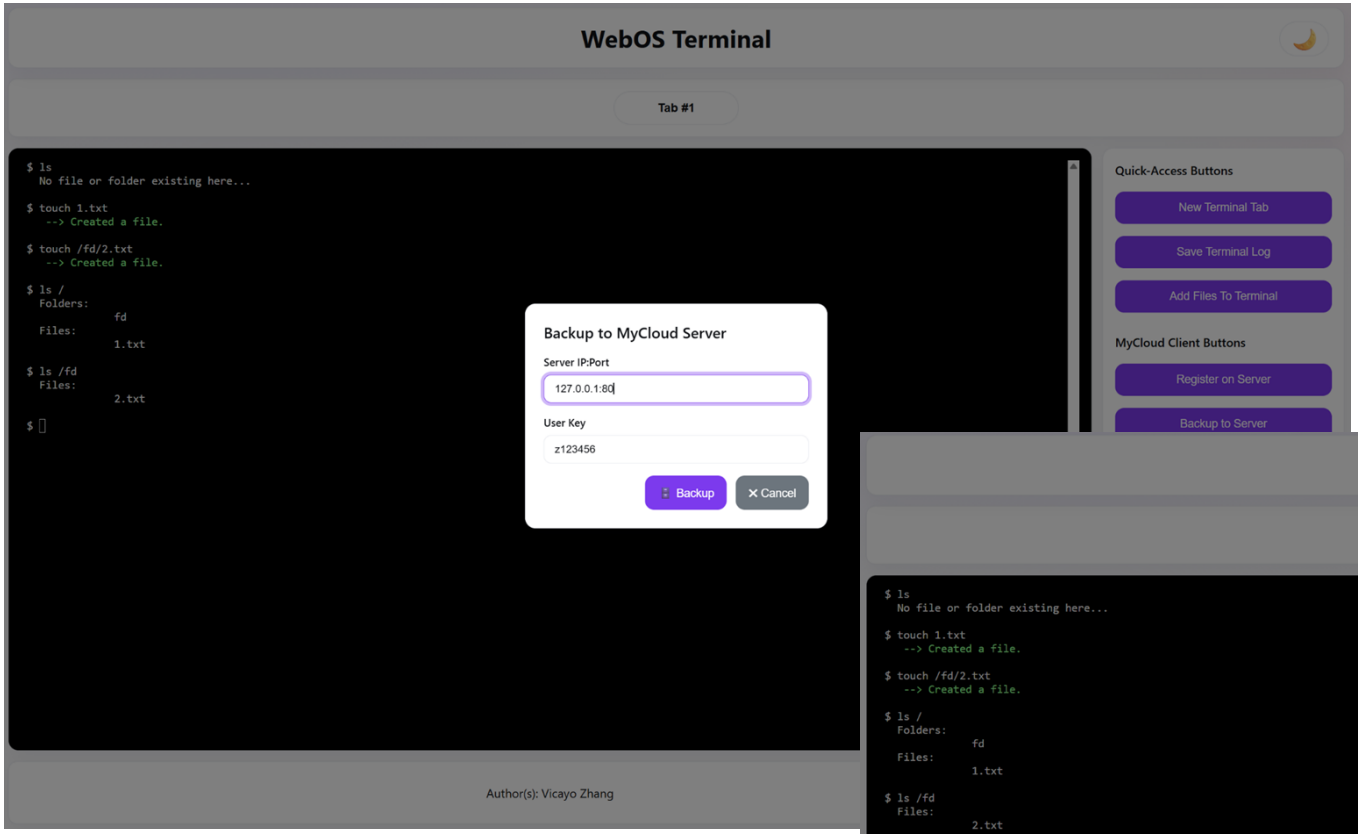
```
$ hello
  Hello World!

$ ls
  No file or folder existing here...

$ man man
   --> A detailed manual of the terminal simulator.
       Usage: man [command_name]

$
```

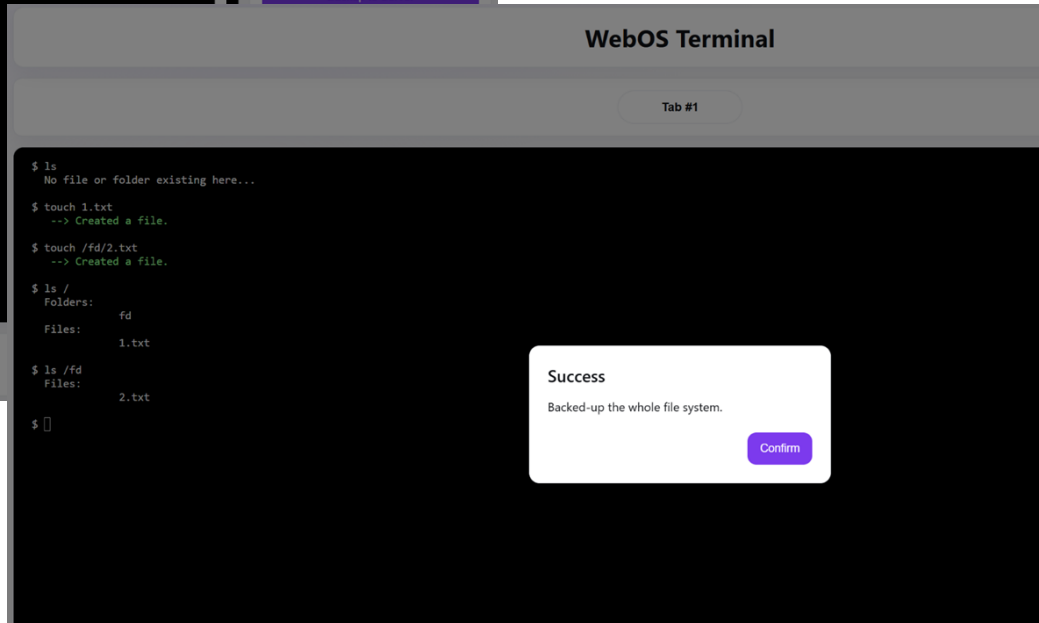
Ln 1, Col 1 | 173 characters | Plain text | 100% | Windows (CRLF) | UTF-8


WebOS Terminal
Open
Downloads
Search f
Organize
New folder
Home
Gallery
OneDrive - Perso
Desktop
Downloads
Documents
Pictures
Music
Videos
Virtual Machi
CLionProjects
PycharmProje
WebstormPro
Arduino
Name
Date modified
Type
Size
Today
1.txt
12/5/2025 10:56 AM
Text Document
1 KB
File name: 1.txt
Quick-Access Buttons
New Terminal Tab
Save Terminal Log
Add Files To Terminal
WebOS Terminal
Tab #1
$
Quick-Access Buttons
New Terminal Tab
Save Terminal Log
Add Files To Terminal
MyCloud Client Buttons
Success
Added 1 file(s) to the current directory.
Confirm
$ ls
Files:
1.txt
$
Author(s): Vicayo Zhang


WebOS Terminal
Tab #1
$
Quick-Access Buttons
New Terminal Tab
Save Terminal Log
Add Files To Terminal
MyCloud Client Buttons
Register on Server
Backup to Server
Register on MyCloud Server
Server IP:Port
127.0.0.1:8088
User Key
z123456
Register
Cancel
Author(s): Vicayo Zhang


WebOS Terminal
Tab #1
$
Quick-Access Buttons
New Terminal Tab
Save Terminal Log
Add Files To Terminal
MyCloud Client Buttons
Register on Server
Backup to Server
Recover from Server
Success
Registered a new user key.
Confirm


WebOS Terminal
Tab #1
$ ls
No file or folder existing here...
$ touch 1.txt
--> Created a file.
$ touch /fd/2.txt
--> Created a file.
$ ls /
Folders:
fd
Files:
1.txt
$ ls /fd
Files:
2.txt
$
Quick-Access Buttons
New Terminal Tab
Save Terminal Log
Add Files To Terminal
MyCloud Client Buttons
Register on Server
Backup to Server
Backup to MyCloud Server
Server IP:Port
127.0.0.1:80
User Key
z123456
Backup
Cancel
Author(s): Vicayo Zhang


WebOS Terminal
Tab #1
$ ls
No file or folder existing here...
$ touch 1.txt
--> Created a file.
$ touch /fd/2.txt
--> Created a file.
$ ls /
Folders:
fd
Files:
1.txt
$ ls /fd
Files:
2.txt
$
Success
Backed-up the whole file system.
Confirm

WebOS Terminal

Tab #1

```
$ ls
  No file or folder existing here...

$ touch 1.txt
   --> Created a file.

$ touch /fd/2.txt
   --> Created a file.

$ ls /
  Folders:
            fd
  Files:
            1.txt

$ ls /fd
  Files:
            2.txt

$
```

Recover from MyCloud Server

Server IP:Port

127.0.0.1:80

User Key

z123456

Recover | Cancel

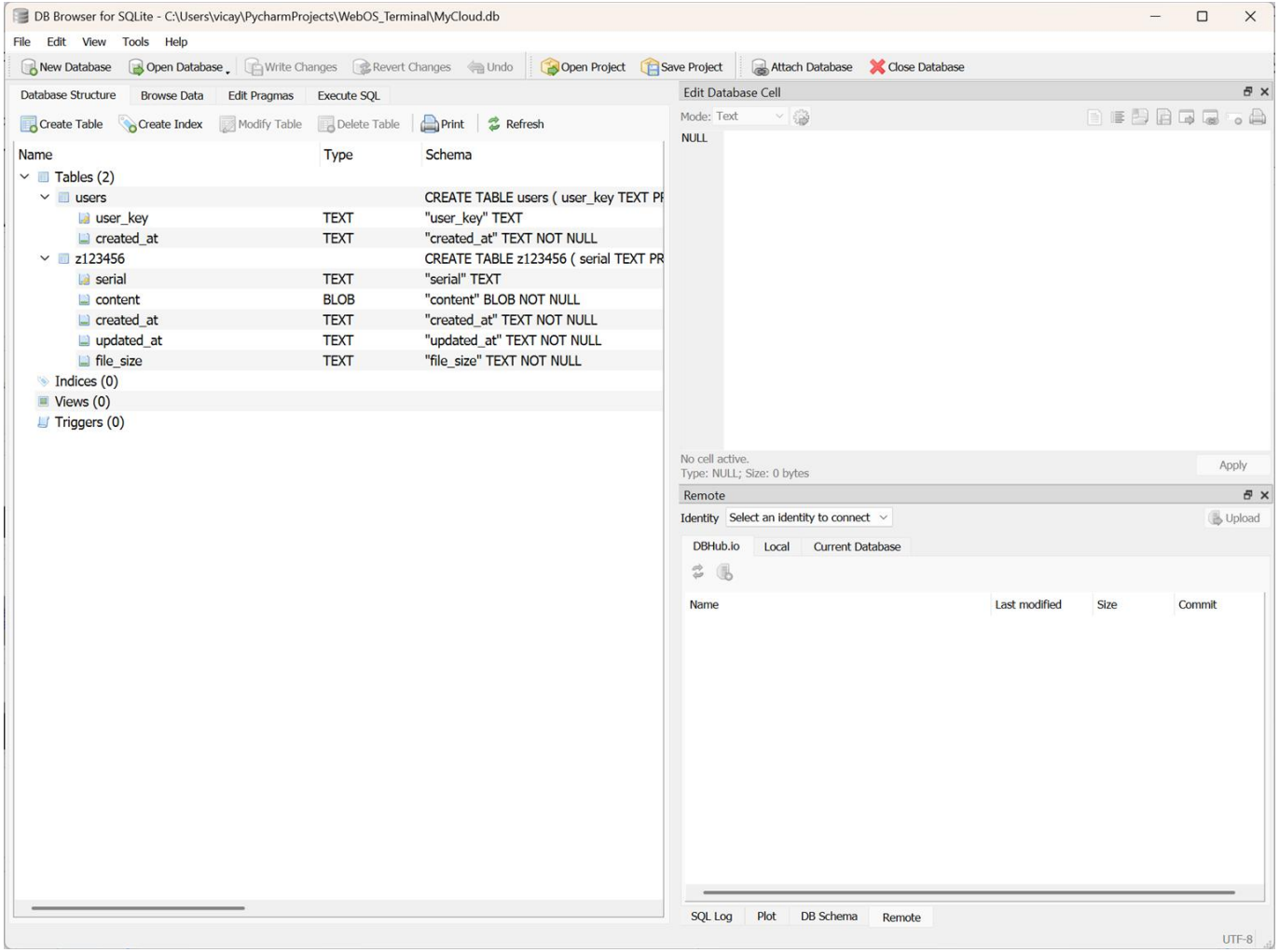
Quick-Access Buttons

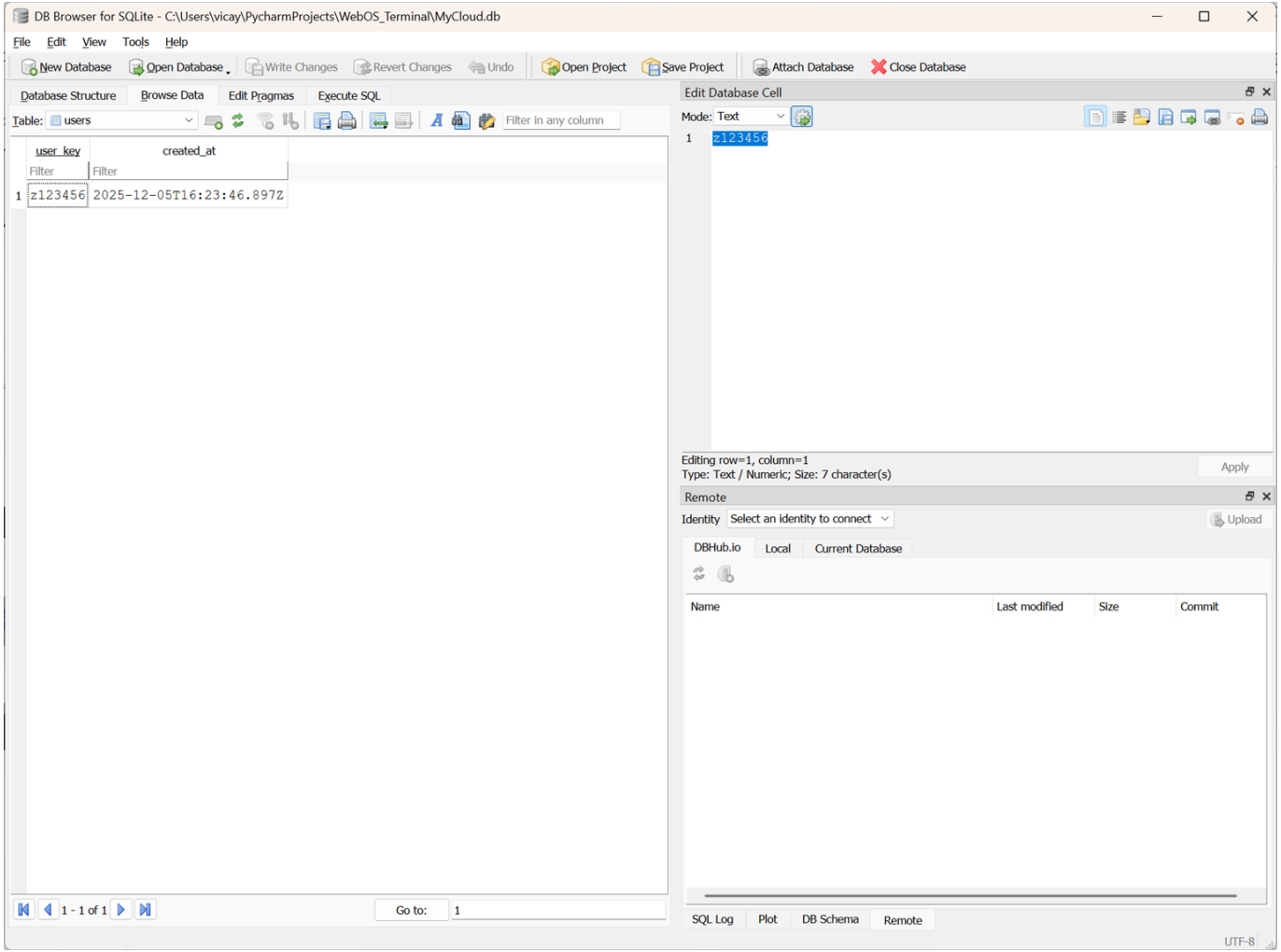
- New Terminal Tab
- Save Terminal Log
- Add Files To Terminal

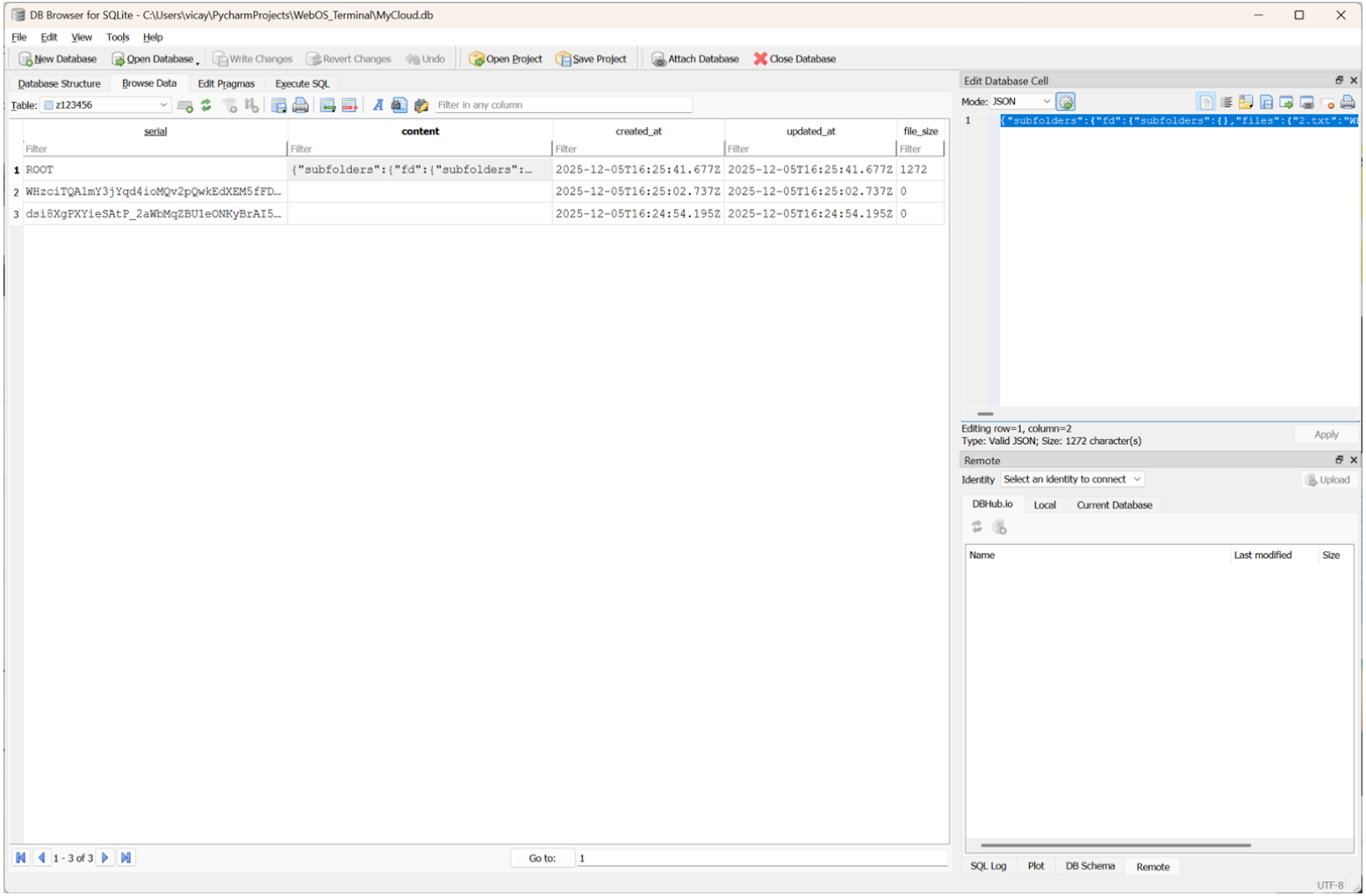
MyCloud Client Buttons

- Register on Server
- Backup to Server
- Recover from Server

Author(s): Vicayo Zhang


DB Browser for SQLite - C:\Users\vicay\PycharmProjects\WebOS_Terminal\MyCloud.db
File Edit View Tools Help
New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Database
Database Structure Browse Data Edit Pragmas Execute SQL
Create Table Create Index Modify Table Delete Table Print Refresh
Name Type Schema
Tables (2)
users CREATE TABLE users (user_key TEXT PR
user_key TEXT "user_key" TEXT
created_at TEXT "created_at" TEXT NOT NULL
z123456 CREATE TABLE z123456 (serial TEXT PR
serial TEXT "serial" TEXT
content BLOB "content" BLOB NOT NULL
created_at TEXT "created_at" TEXT NOT NULL
updated_at TEXT "updated_at" TEXT NOT NULL
file_size TEXT "file_size" TEXT NOT NULL
Indices (0)
Views (0)
Triggers (0)
Edit Database Cell
Mode: Text
NULL
No cell active.
Type: NULL; Size: 0 bytes
Apply
Remote
Identity Select an identity to connect
Upload
DBHub.io Local Current Database
Name Last modified Size Commit
SQL Log Plot DB Schema Remote
UTF-8


DB Browser for SQLite - C:\Users\vicay\PycharmProjects\WebOS_Terminal\MyCloud.db
File Edit View Tools Help
New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Database
Database Structure Browse Data Edit Pragmas Execute SQL
Table: users
Filter in any column
user_key created_at
Filter Filter
1 z123456 2025-12-05T16:23:46.897Z
1 - 1 of 1
Go to: 1
Edit Database Cell
Mode: Text
1 z123456
Editing row=1, column=1
Type: Text / Numeric; Size: 7 character(s)
Apply
Remote
Identity Select an identity to connect
Upload
DBHub.io Local Current Database
Name Last modified Size Commit
SQL Log Plot DB Schema Remote
UTF-8


DB Browser for SQLite - C:\Users\vicay\PycharmProjects\WebOS_Terminal\MyCloud.db
File Edit View Tools Help
New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database Close Database
Database Structure Browse Data Edit Pragmas Execute SQL
Table: z123456
Filter in any column
serial content created_at updated_at file_size
Filter Filter Filter Filter Filter
1 ROOT {"subfolders":{"fd":{"subfolders":… 2025-12-05T16:25:41.677Z 2025-12-05T16:25:41.677Z 1272
2 WHzciTQAlmY3jYqd4ioMQv2pQwkEdXEM5fFD… 2025-12-05T16:25:02.737Z 2025-12-05T16:25:02.737Z 0
3 dsi8XgPXYieSAtP_2aWbMqZBU1eONKyBrAI5… 2025-12-05T16:24:54.195Z 2025-12-05T16:24:54.195Z 0
1 - 3 of 3
Go to: 1
Edit Database Cell
Mode: JSON
1 {"subfolders":{"fd":{"subfolders":{},"files":{"2.txt":"w
Editing row=1, column=2
Type: Valid JSON; Size: 1272 character(s)
Apply
Remote
Identity Select an identity to connect
Upload
DBHub.io Local Current Database
Name Last modified Size
SQL Log Plot DB Schema Remote
UTF-8

Server-Side

```
// ---------------------------------------- DB ----------------------------------------
import path from 'path';
import {fileURLToPath} from 'url';
import {dirname} from 'path';
import sqlite3Module from 'sqlite3';
import {randomUUID} from 'crypto';
import {hrtime} from 'process';
import fs from 'fs';
import multer from 'multer';
import express from 'express';
import cors from 'cors';

const
    __filename : string  = fileURLToPath(import.meta.url),
    __dirname : string  = dirname(__filename),
    sqlite3 : sqlite3  = sqlite3Module.verbose(),
    database : Database  = new sqlite3.Database(
        path.join(__dirname, 'MyCloud.db'),
        mode: sqlite3.OPEN_READWRITE | sqlite3.OPEN_CREATE,
        callback: (error : Error | null ) : void  => {
            if (error) console.error(`Failed to open the database: ${error.message}`);
        }
    );

/*
* Initialize the database
* */
database.serialize( callback: () : void  => {  vicayo +1
    database.run( sql: 'PRAGMA foreign_keys = ON;');
    database.run( sql: 'PRAGMA journal_mode = WAL;');
    database.run(
        sql: 'CREATE TABLE IF NOT EXISTS users (' +
        '    user_key      TEXT PRIMARY KEY,                                  /* unique user key (string) */' +
        '    created_at    TEXT NOT NULL' +
        ') WITHOUT ROWID;                                                     /* good when PRIMARY KEY is not an integer */',
        callback: (createError : Error | null ) : void  => {
            if (createError) {
                console.error('Failed to read/create the users table');
                return;
            }
            console.log('Successfully located users table.');
        }
    );
});
```

```
// ------------------------------------ MULTER ------------------------------------

class TempFileNameLake {  Show usages  vicayo
    #lastTime : bigint  = 0n;
    #idx : bigint  = 0n;

    generateNext() : string  {  Show usages  vicayo
        const currentTime : bigint  = hrtime.bigint() / 1_000_000n; // monotonic ms since process start
        if (currentTime === this.#lastTime) {
            this.#idx++;
        } else {
            this.#lastTime = currentTime;
            this.#idx = 0n;
        }
        return `${process.pid}-${currentTime}-${this.#idx}-${randomUUID()}`;
    }
}

const
    UPLOAD_DIR : string  = path.join(__dirname, 'uploads'),
    tempFileNameLake : TempFileNameLake  = new TempFileNameLake();

if (!fs.existsSync(UPLOAD_DIR))
    fs.mkdirSync(UPLOAD_DIR,  options: {recursive: true});

const multerUpload : Multer  = multer( options: {
    storage: multer.diskStorage( opts: {
        // where to save
        destination: (req, file, cb) : void  => {
            cb(null, UPLOAD_DIR);
        },
        // what to name the file
        filename: (req, file, cb) : void  => {
            cb(null, tempFileNameLake.generateNext());
        }
    })
});
```

```
// ------------------------------------ Server App ------------------------------------

const
    app : Express  = express(),
    legalUserKeyRegExp : RegExp  = /^[A-Za-z_][A-Za-z0-9_]{5,1048576}$/,
    legalFileSerialRegExp : RegExp  = /^(?:ROOT|[A-Za-z_][A-Za-z0-9_]{127,4096})$/,
    getISOTimeString = () : string  => new Date().toISOString(),
    utf8Decoder : TextDecoder  = new TextDecoder( encoding: 'utf-8'),
    utf8Encoder = new TextEncoder();
// MAX_TEMP_FILE_SIZE = 1024 * 1024 * 1024 * 1024, // 1T.

app.use(express.static(path.join(__dirname, 'webpage')));

// Basic Webpage Server
{
    app.get('/', (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) : void  => {
        res.sendFile(path.join(__dirname, 'webpage', 'index.html'));
    });
}

app.use(express.json());
app.use(cors());
app.use((error : Request<RouteParameters<...>, any, any, ParsedQs, Record<...>> , req : Response<any, Record<...>> , res : NextFunction , next) : any | undefined  => {
    if (error && error.type === 'entity.too.large') {
        return res.status(413).json({error: 'Payload Too Large (getRecordsJSON body limit).'});
    }
    if (error instanceof SyntaxError && 'body' in error) {
        return res.status(400).json({error: 'Invalid getRecordsJSON body.'});
    }
    next(error);
});
```

```
const
    HOST : string  = '0.0.0.0',
    PORT : number  = 80,
    server : Server  = app.listen(PORT, HOST, backlog: () : void  => {
        console.log(`Server listening on http://${HOST}:${PORT}.`);
    }),
    shutdownServer = (signal) : void  => {
        console.log(`\n${signal} received: shutting down...`);
        server.close( callback: (serverCloseError : Error | undefined ) : void  => {
            if (serverCloseError) {
                console.error(`Failed to close the server. <-- ${serverCloseError}`);
                process.exit( code: 1);
                return;
            }
            console.log('The server is closed.');
            database.close( callback: (databaseCloseError : Error | null ) : void  => {
                if (databaseCloseError) {
                    console.error(`Failed to close the database. <-- ${databaseCloseError}`);
                    process.exit( code: 1);
                    return;
                }
                console.log('The database is closed.');
                process.exit( code: 0);
            });
        });
    };

process.on( eventName: 'SIGINT', listener: () : void  => shutdownServer( signal: 'SIGINT'));  vicayo
process.on( eventName: 'SIGTERM', listener: () : void  => shutdownServer( signal: 'SIGTERM'));  vicayo
```

```
/**
 * This POST request
 *      (1) Registers a <user_key> in the <users> table
 *      (2) Creates a <user_key> table
 *
 * req.body:
 *      user_key: string
 *
 * res.body:
 *      error               when failure
 * */
app.post( path: '/mycloud/users/register/', multerUpload.none(), (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<any, Record<...>> ) : any | undefined  => {
    if (req.body === undefined) {
        return res.status( code: 400).json( body: {
            error: `Body not found.`
        });
    }
    const
        /** @type {{user_key: string}} */
        {user_key: user_key} = req.body;
    if (typeof user_key !== 'string' || !legalUserKeyRegExp.test(user_key)) {
        return res.status( code: 400).json( body: {
            error: `"${user_key}" must be a valid user_key. Please Use [A-Za-z_][A-Za-z0-9_]{1024,1048576}.`
        });
    }
    // language=SQL format=false
    database.run(
        sql: 'INSERT INTO users (user_key, created_at) VALUES (?, ?);',
        params: [user_key, getISOTimeString()],
        callback: (insertError : Error | null ) : any | undefined  => {...}
    );
});
```

```
/**
 * This POST request
 *      Determines whether <user_key> is in the <users> table
 *
 * req.body:
 *      user_key: string
 *
 * res.body:
 *      error               when failure
 *      result=true/false   when success
 * */
app.post( path: '/mycloud/users/verify/', multerUpload.none(), (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<any, Record<...>> ) : any | undefined  => {
    if (req.body === undefined) {
        return res.status( code: 400).json( body: {
            error: `Body not found.`
        });
    }
    const
        /** @type {{user_key: string}} */
        {user_key: user_key} = req.body;
    if (typeof user_key !== 'string' || !legalUserKeyRegExp.test(user_key)) {
        return res.status( code: 400).json( body: {
            error: `"${user_key}" must be a valid user_key. Please Use [A-Za-z_][A-Za-z0-9_]{1024,1048576}.`
        });
    }
    database.get(
        sql: 'SELECT ' +
        '    EXISTS(SELECT 1 FROM users WHERE user_key = ?) AS user_present,' +
        "    EXISTS(SELECT 1 FROM main.sqlite_schema WHERE type='table' AND name = ?) AS table_present;",
        params: [user_key, user_key],
        callback: (selectError : Error | null , selectRow) : any  => {...}
    );
});
```

```
/**
 * This POST request
 *      Create/Update <serial, content> pair to the <user_key> table (updating <updated_at>)
 *
 * req:
 *      user_key: string      ---> req.body.user_key
 *      serial: string        ---> req.body.serial
 *      content               ---> req.file
 *      (file_size)           <--- req.file
 *      created_at: string    ---> req.body.created_at
 *      updated_at: string    ---> req.body.updated_at
 *
 * res.body:
 *      error              when failure
 * */
app.post( path: '/mycloud/files/backup/', multerUpload.single( name: 'content'), async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<any, Record<...>> ) : Promise<...> => {
    if (req.body === undefined) {...}
    if (req.file === undefined) {...}
    const
        /** @type {{user_key: string, serial: string, created_at: string, updated_at: string}} */
        {user_key: user_key, serial: serial, created_at: created_at, updated_at: updated_at} = req.body;
    if (typeof user_key !== 'string' || !legalUserKeyRegExp.test(user_key)) {...}
    if (typeof serial !== 'string' || !legalFileSerialRegExp.test(serial)) {...}
    if (typeof created_at !== 'string' || created_at.length === 0) {...}
    if (typeof updated_at !== 'string') {...}
    const
        /** @type {{encoding: string, mimetype: string, path: string, size: number}} */
        {encoding: fileEncoding, mimetype: fileType, path: filePath, size: fileSize} = req.file;
    if (typeof fileEncoding !== 'string' || typeof fileType !== 'string' || typeof fileSize !== 'number') {...}
    if (typeof filePath !== 'string') {...}
    const fileContent : NonSharedBuffer  = await fs.promises.readFile(filePath).catch(_ => null);
    if (fileContent === null) {...}
    // language=SQL format=false
    database.run(
        sql: `INSERT INTO ${user_key} (serial, content, created_at, updated_at, file_size) VALUES (?, ?, ?, ?, ?)` +
        'ON CONFLICT(serial) DO UPDATE SET' +
        '    content    = excluded.content,' +
        '    updated_at = excluded.updated_at,' +
        '    file_size  = excluded.file_size',
        params: [serial, fileContent, created_at, updated_at, `${fileSize}`],
        callback: async (upsertError : Error | null ) : Promise<any> => {...}
    );
});
```

```
/**
 * This POST request
 *      Gets <content> by <serial> in the <user_key> table (getting <created_at> and <updated_at>)
 *
 * req.body:
 *      user_key: string
 *      serial: string
 *
 * res.header:
 *      error              when failure
 *      serial             when success     (for the convenience of client-side implementations)
 *      created_at         when success
 *      updated_at         when success
 *      file_size          when success
 *
 * res.arrayBuffer:
 *      content            when success
 * */
app.post( path: '/mycloud/files/recover/', multerUpload.none(), (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<any, Record<...>> ) : any | undefined  => {
    if (req.body === undefined) {
        return res.status( code: 400).json( body: {
            error: `Body not found.`
        });
    }
    const
        /** @type {{user_key: string, serial: string}} */
        {user_key: user_key, serial: serial} = req.body;
    if (typeof user_key !== 'string' || !legalUserKeyRegExp.test(user_key)) {
        return res.status( code: 400).json( body: {
            error: `"${user_key}" must be a valid user_key. Please Use [A-Za-z_][A-Za-z0-9_]{1024,1048576}.`
        });
    }
    if (typeof serial !== 'string' || !legalFileSerialRegExp.test(serial)) {
        return res.status( code: 400).json( body: {
            error: `"${user_key}" must be a valid user_key ([A-Za-z_][A-Za-z0-9_]{128,4096}). Please check the client implementation.`
        });
    }
    // language=SQL format=false
    database.get(
        sql: `SELECT content, created_at, updated_at, file_size FROM ${user_key} WHERE serial = ? LIMIT 1;`,
        params: [serial],
        callback: (selectError : Error | null , selectRow) : any  => {...}
    );
});
```

```
PS C:\Users\vicay\PycharmProjects\WebOS_Terminal> npm start

> MyCloudServer@1.0.0 start
> node create_mycloud_server.mjs

Server listening on http://0.0.0.0:80.
Successfully located users table.
 --> Registered z123456 in the user table.
 --> Created a file table called z123456.
 <-- User "z12345.." made a new backup.
 --> User "z12345.." requested a previous backup.
```

```
/**
 * This POST request
 *      compiles .js/.c/.cpp/.py file to .wasm file/////
 *
 * req:
 *      content             ---> req.file
 *      (file_size)         <--- req.file
 *
 * res.body:
 *      error               when failure
 *
 * res.arrayBuffer:
 *      content             when success
 * */
app.post( path: '/mycloud/compile/', multerUpload.single( name: 'content'), async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<any, Record<...>> ) : Promise<any>  => {

});
```

Thank You!