COMPSCI 326 Group 1 (Online)

Mini Terminal

(https://github.com/VicayoMua/326_Mini_Terminal)

Milestone 5

(https://github.com/VicayoMua/326_Mini_Terminal/milestone/3)

Team Member: **Vicayo Zhang, Aryan Ghosh, and Stella Dey**

4/11/2025

Project Name: Mini Terminal

Problem/Why This Project?

- Large universities, like UMass, usually provide ssh server environments, like EdLab, for students to get familiar with Linux-styled commands. However, most students only use the most basic features on EdLab, so the power assumption and the continuous maintenance cost are not worth it.
- Students, who are not enrolled in a university and have no previous experience in installing and using Linux, usually find it hard to install a Linux distribution on their PCs.

Solution:

- A terminal simulator, which can be accessed on web browsers and run on local devices (phones, tablets, and PCs), can perfectly solve this problem.

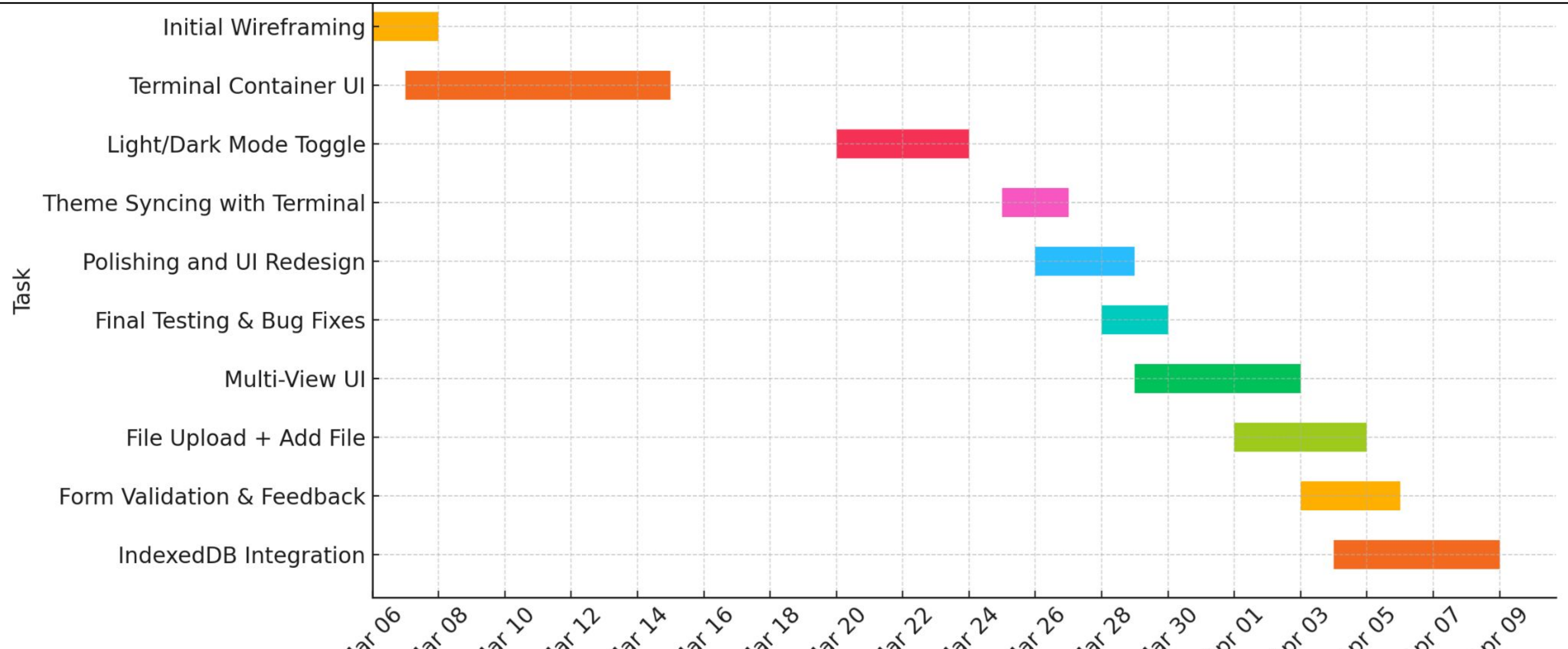
Key Features:

- The terminal simulator will support the most usual Linux commands, like ls, pwd, cd, mkdir, rename, touch, cp, edit. It has file managing abilities.
- Additionally, we can set up additional buttons outside the terminal window to support log-saving, file-system-importing, and file-system-exporting. So, people can easily save and resume their work efficiently.
- Finally, if time allows, the terminal will have supports for Web Assembly.

Team Members

- **Vicayo Zhang – Project Manager & Core Service Developer**
    - **First Issue To Work ON: Basic terminal input and output logics**
    - **Second Issue To Work ON: Log recording logics**
    - **Third Issue To Work ON: file system simulation logics**
    - **Fourth Issue To Work ON: some demo on-terminal applications/commands (hello, help, man, and echo)**
- **Aryan Ghosh – Software Developer & Admin Monitoring**
    - **First Issue To Work ON: Command input screen**
    - **Second Issue To Work ON: User Session data**
    - **Third Issue To Work ON: Project Documentation**
- **Stella Dey – Debugging & Testing Coordinator**
    - **First Issue To Work ON: Customizable Theme - (Light/Dark Mode) #17**
    - **Second Issue To Work ON: File System Structure Data #18**
    - **Third Issue To Work ON: Project Documentation**

Historical Development Timeline



Vicayo Zhang - Assigned Work Summary

In the HTML file:

- **Add a new button for indexDB application (Feature A. Screenshot 1)**
- **Update the button names (Feature A. Screenshot 1)**
- **Research online to find appropriate libraries for terminal simulations (Feature B. Screenshot 2)**
- **Include all the scripts in html file (Feature B. Screenshot 2)**
- **Update code for light-and-dark UI mode (Feature D. Screenshot 4)**

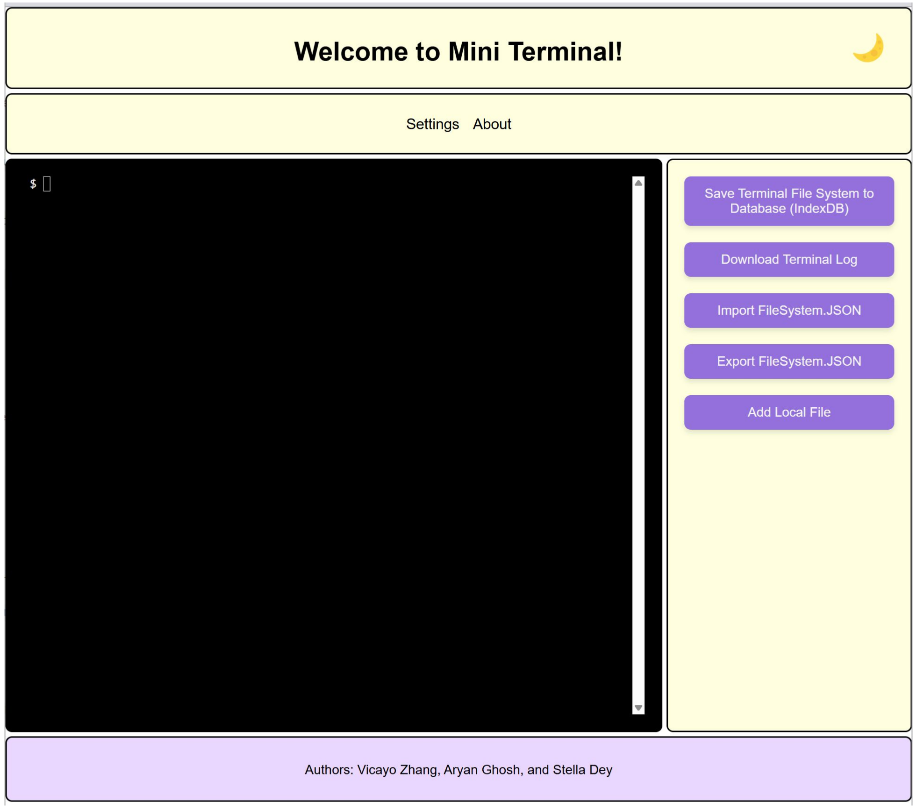
In the JS files:

- **Write JS code for indexDB to store user preferences or temporary data (Feature C. Screenshot 3)**
- **Implement the core service of the terminal with support of xterm.js (Feature E. Screenshot 5)**
    - **File system simulation, keyboard input handler, with elegant data-hiding performant code structure**
- **Link the core service function to the button safely (Feature G. Screenshot 7)**

In the CSS file:

- **Delete the css code for light-and-dark UI mode, and implement the switch in HTML file using JS (Feature D. Screenshot 4)**
- **Adjust the page height (Feature F. Screenshot 6)**

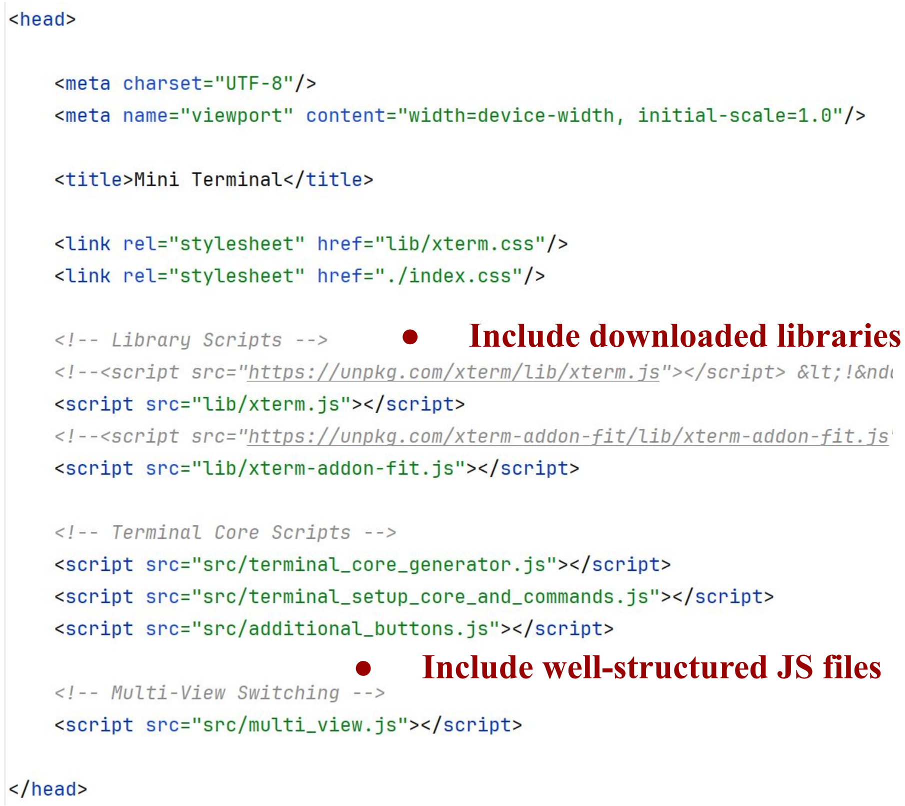
Vicayo Zhang - Screenshot 1 (Code & UI Explanation)


Welcome to Mini Terminal!
Settings About
$
Save Terminal File System to Database (IndexDB)
Download Terminal Log
Import FileSystem.JSON
Export FileSystem.JSON
Add Local File
Authors: Vicayo Zhang, Aryan Ghosh, and Stella Dey

```
<div class="additional-buttons">
    <button
            type="button"
            onclick="button_to_save_terminal_file_system_to_indexDB()">
        Save Terminal File System to Database (IndexDB)
    </button>
    <button
            type="button"
            onclick="alert('Button clicked!')">
        Download Terminal Log
    </button>
    <button
            type="button"
            onclick="alert('Button clicked!')">
        Import FileSystem.JSON
    </button>
    <button
            type="button"
            onclick="alert('Button clicked!')">
        Export FileSystem.JSON
    </button>
    <button
            type="button"
            onclick="alert('Button clicked!')">
        Add Local File
    </button>
```

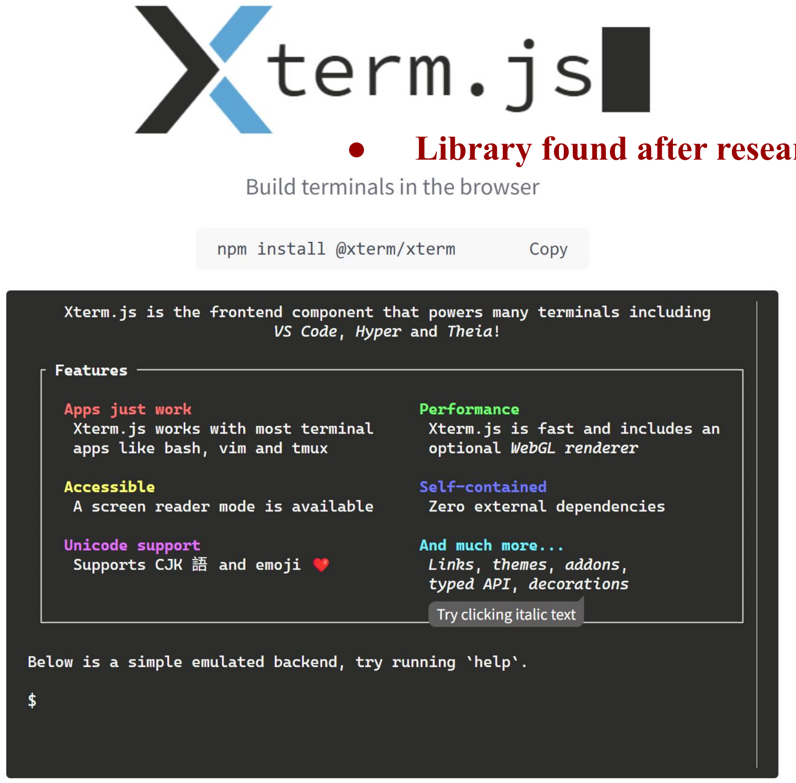
- **Code for buttons**

Vicayo Zhang - Screenshot 2 (Code & UI Explanation)



```
<head>

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

    <title>Mini Terminal</title>

    <link rel="stylesheet" href="lib/xterm.css"/>
    <link rel="stylesheet" href="./index.css"/>

    <!-- Library Scripts -->
    <!--<script src="https://unpkg.com/xterm/lib/xterm.js"></script> &lt;!&nd
    <script src="lib/xterm.js"></script>
    <!--<script src="https://unpkg.com/xterm-addon-fit/lib/xterm-addon-fit.js
    <script src="lib/xterm-addon-fit.js"></script>

    <!-- Terminal Core Scripts -->
    <script src="src/terminal_core_generator.js"></script>
    <script src="src/terminal_setup_core_and_commands.js"></script>
    <script src="src/additional_buttons.js"></script>

    <!-- Multi-View Switching -->
    <script src="src/multi_view.js"></script>

</head>
```

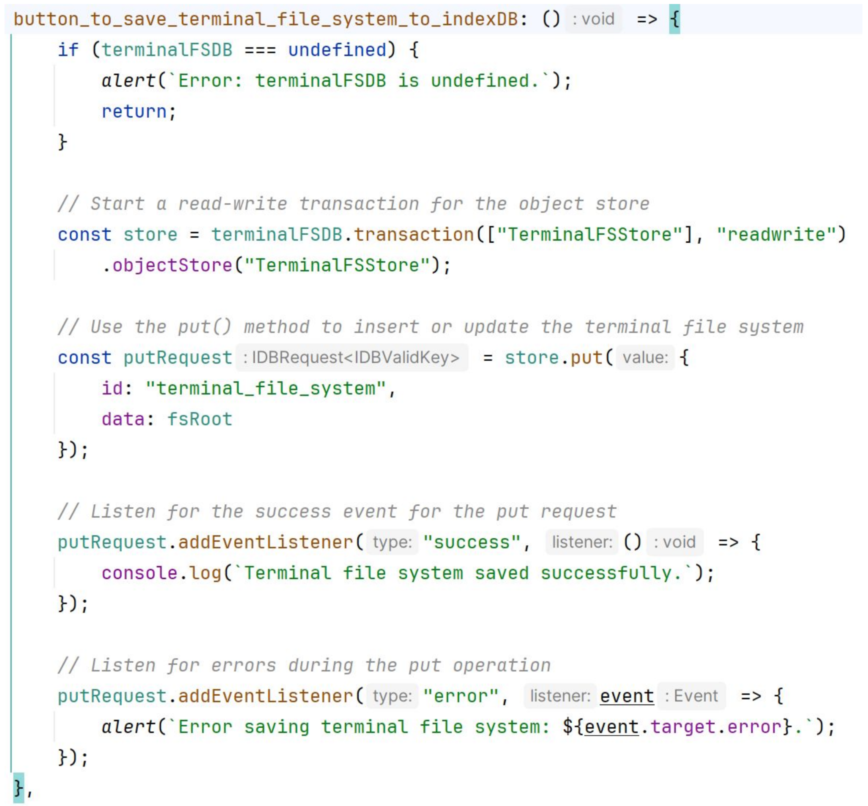
- **Include downloaded libraries**
- **Include well-structured JS files**



```
Xterm.js is the frontend component that powers many terminals including
                    VS Code, Hyper and Theia!

 Features
   Apps just work                          Performance
    Xterm.js works with most terminal       Xterm.js is fast and includes an
    apps like bash, vim and tmux            optional WebGL renderer

   Accessible                              Self-contained
    A screen reader mode is available       Zero external dependencies

   Unicode support                         And much more...
    Supports CJK 語 and emoji ❤️            Links, themes, addons,
                                            typed API, decorations

Below is a simple emulated backend, try running `help`.

$
```

- **Library found after research**

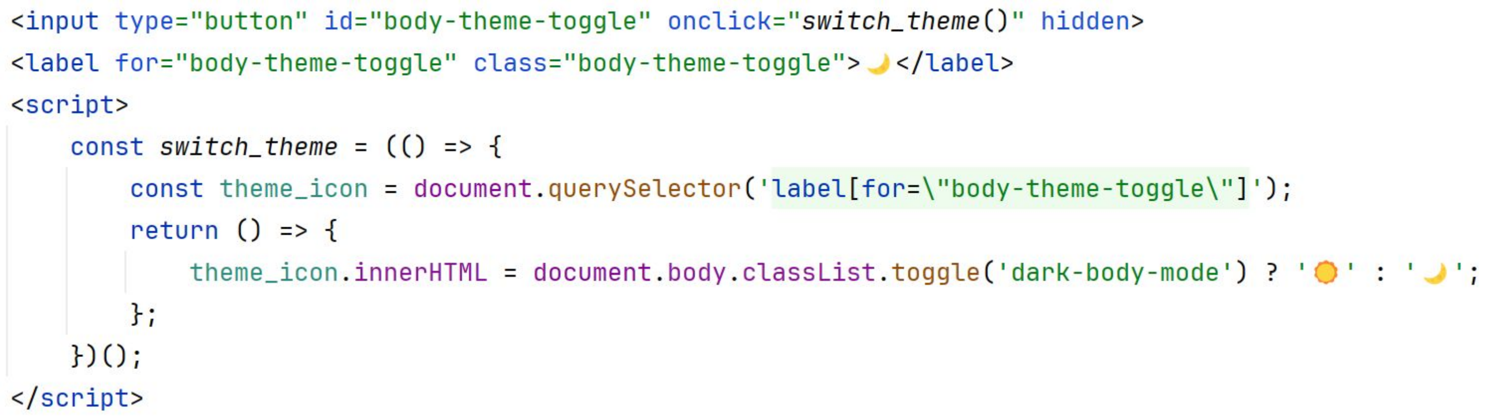
Vicayo Zhang - Screenshot 3 (Code & UI Explanation)



```
button_to_save_terminal_file_system_to_indexDB: () : void  => {
    if (terminalFSDB === undefined) {
        alert(`Error: terminalFSDB is undefined.`);
        return;
    }

    // Start a read-write transaction for the object store
    const store = terminalFSDB.transaction(["TerminalFSStore"], "readwrite")
        .objectStore("TerminalFSStore");

    // Use the put() method to insert or update the terminal file system
    const putRequest : IDBRequest<IDBValidKey>  = store.put( value: {
        id: "terminal_file_system",
        data: fsRoot
    });

    // Listen for the success event for the put request
    putRequest.addEventListener( type: "success",  listener: () : void  => {
        console.log(`Terminal file system saved successfully.`);
    });

    // Listen for errors during the put operation
    putRequest.addEventListener( type: "error",  listener: event : Event  => {
        alert(`Error saving terminal file system: ${event.target.error}.`);
    });
},
```

- **Open the database transaction and store**
- **Send put request**
- **Assign action handling success and error cases.**

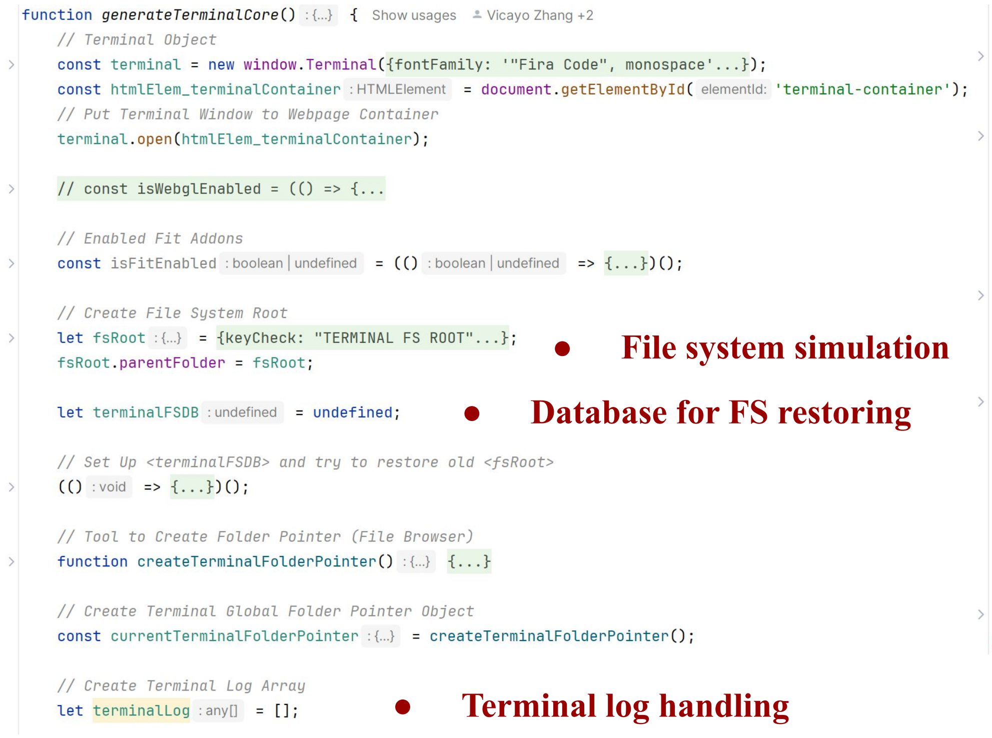
Vicayo Zhang - Screenshot 4 (Code & UI Explanation)



```
<input type="button" id="body-theme-toggle" onclick="switch_theme()" hidden>
<label for="body-theme-toggle" class="body-theme-toggle">🌙</label>
<script>
    const switch_theme = (() => {
        const theme_icon = document.querySelector('label[for=\"body-theme-toggle\"]');
        return () => {
            theme_icon.innerHTML = document.body.classList.toggle('dark-body-mode') ? '🌞' : '🌙';
        };
    })();
</script>
```

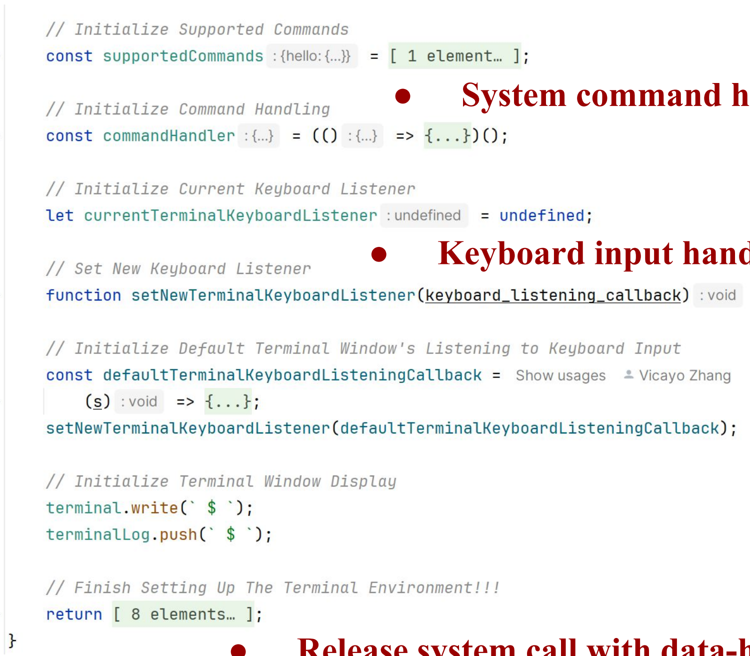
- **Now theme switching are unified using JS code.**

Vicayo Zhang - Screenshot 5 (Code & UI Explanation)



```
function generateTerminalCore() :{...}  {   Show usages   Vicayo Zhang +2
    // Terminal Object
    const terminal = new window.Terminal({fontFamily: '"Fira Code", monospace'...});
    const htmlElem_terminalContainer :HTMLElement  = document.getElementById( elementId: 'terminal-container');
    // Put Terminal Window to Webpage Container
    terminal.open(htmlElem_terminalContainer);

    // const isWebglEnabled = (() => {...

    // Enabled Fit Addons
    const isFitEnabled :boolean|undefined  = (() :boolean|undefined  => {...})();

    // Create File System Root
    let fsRoot :{...}  = {keyCheck: "TERMINAL FS ROOT"...};
    fsRoot.parentFolder = fsRoot;

    let terminalFSDB :undefined  = undefined;

    // Set Up <terminalFSDB> and try to restore old <fsRoot>
    (() :void  => {...})();

    // Tool to Create Folder Pointer (File Browser)
    function createTerminalFolderPointer() :{...}  {...}

    // Create Terminal Global Folder Pointer Object
    const currentTerminalFolderPointer :{...}  = createTerminalFolderPointer();

    // Create Terminal Log Array
    let terminalLog :any[]  = [];
```

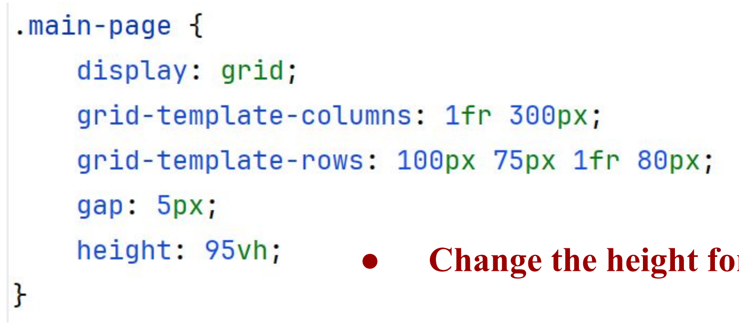
- **File system simulation**
- **Database for FS restoring**
- **Terminal log handling**



```
    // Initialize Supported Commands
    const supportedCommands :{hello:{...}}  = [ 1 element... ];

    // Initialize Command Handling
    const commandHandler :{...}  = (() :{...}  => {...})();

    // Initialize Current Keyboard Listener
    let currentTerminalKeyboardListener :undefined  = undefined;

    // Set New Keyboard Listener
    function setNewTerminalKeyboardListener(keyboard_listening_callback) :void  {...}

    // Initialize Default Terminal Window's Listening to Keyboard Input
    const defaultTerminalKeyboardListeningCallback =   Show usages   Vicayo Zhang
        (s) :void  => {...};
    setNewTerminalKeyboardListener(defaultTerminalKeyboardListeningCallback);

    // Initialize Terminal Window Display
    terminal.write(` $ `);
    terminalLog.push(` $ `);

    // Finish Setting Up The Terminal Environment!!!
    return [ 8 elements... ];
}
```

- **System command handler**
- **Keyboard input handler**
- **Release system call with data-hiding**

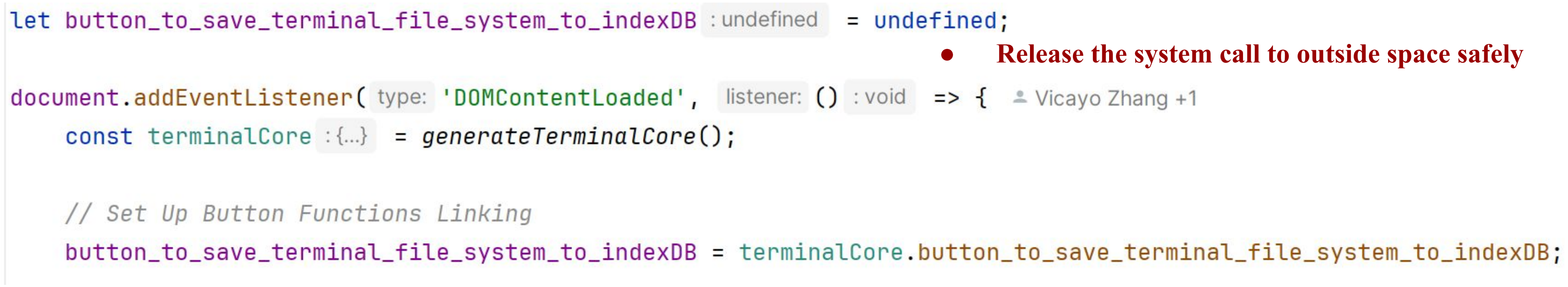
Vicayo Zhang - Screenshot 6 (Code & UI Explanation)



```
.main-page {
    display: grid;
    grid-template-columns: 1fr 300px;
    grid-template-rows: 100px 75px 1fr 80px;
    gap: 5px;
    height: 95vh;
}
```

- **Change the height for better looking**

Vicayo Zhang - Screenshot 7 (Code & UI Explanation)



```
let button_to_save_terminal_file_system_to_indexDB : undefined  = undefined;

document.addEventListener( type: 'DOMContentLoaded',  listener: () : void  => {   Vicayo Zhang +1
    const terminalCore : {...}  = generateTerminalCore();

    // Set Up Button Functions Linking
    button_to_save_terminal_file_system_to_indexDB = terminalCore.button_to_save_terminal_file_system_to_indexDB;
```

- **Release the system call to outside space safely**

Vicayo Zhang - Challenges and Insights

- **As more and more buttons and functions are added in this project, we need to adjust the css stylesheet accordingly.**
- **We need to study the imported libraries (xterm.js) more, to better work with it.**
- **We need to take care of the asynchronous race cases, data hiding, and (storage and runtime) performance issues when implementing the core services.**

Vicayo Zhang - Future Improvements

- **Implementing the javascript functions so that we have many other available commands.**

Aryan Ghosh - Assigned Work Summary

1. **Multi-View User Interface**:
    - Implement JavaScript-based navigation to switch between different views in the application without requiring a full page reload.

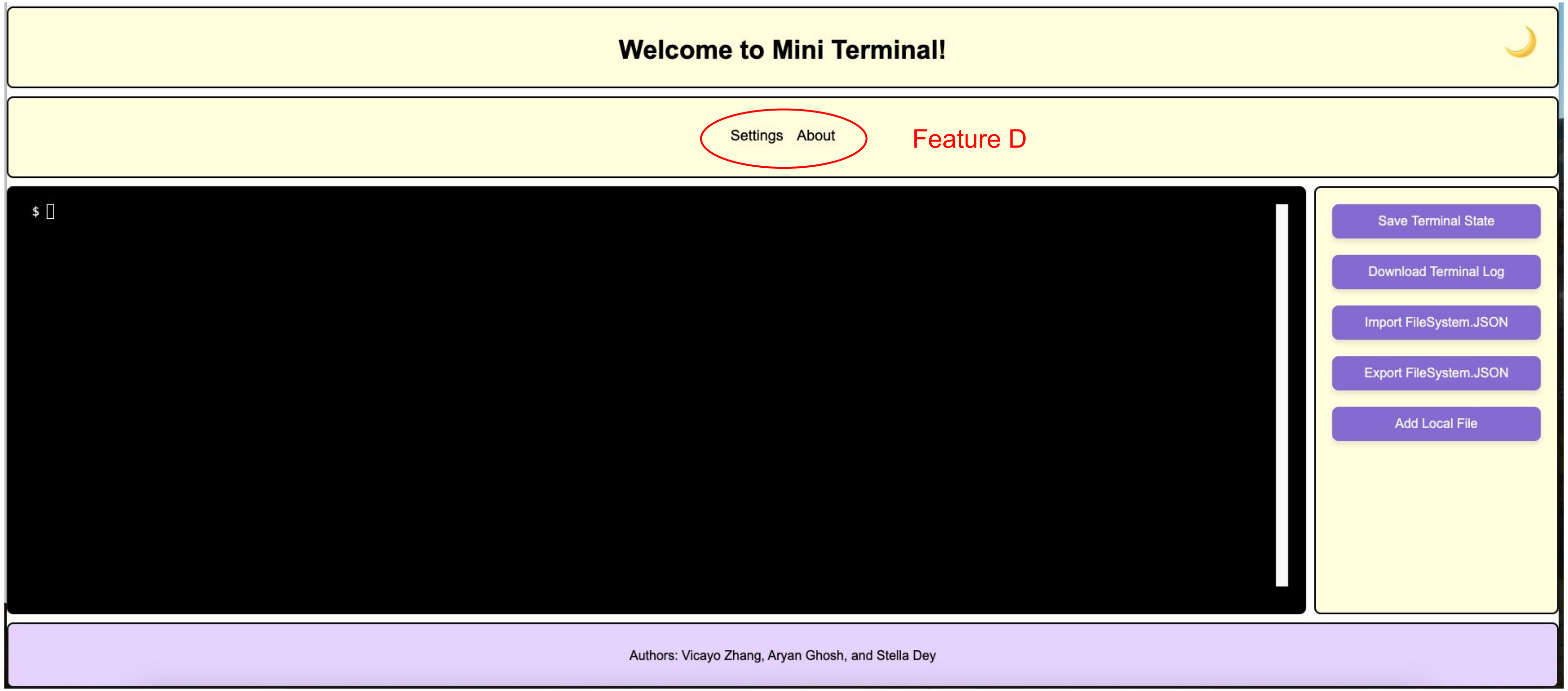
https://github.com/VicayoMua/326_Mini_Terminal/pull/31

2. **Session Management Data Design (Yet to implement)**:
    - Designing how user sessions are managed and stored, enabling users to save their terminal sessions and resume later.
    - Structuring the session attributes including SessionID, UserID, CommandsEntered, and LastAccessed.

3. **Software Development and Admin Monitoring**:
    - Focused on JavaScript coding and ensuring the consistency of documentation provided by the team.
    - Monitoring the implementation of key terminal commands such as pwd, ls, cd, touch, and so forth (yet to implement)

https://github.com/VicayoMua/326_Mini_Terminal/pull/30

Aryan Ghosh - Screenshot

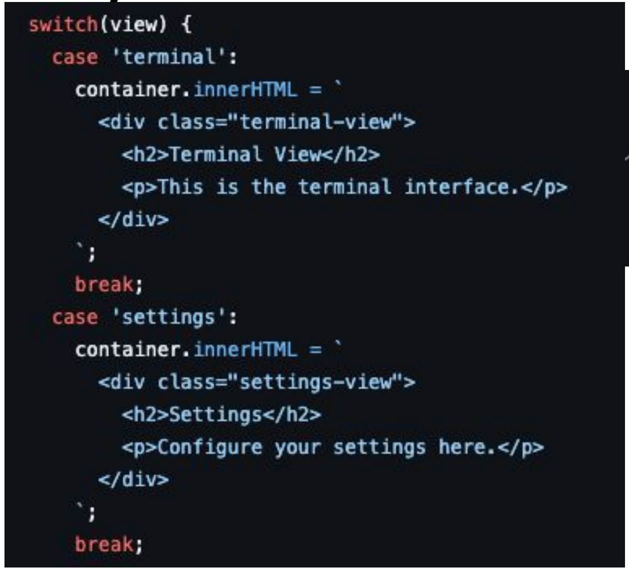


```
$
```

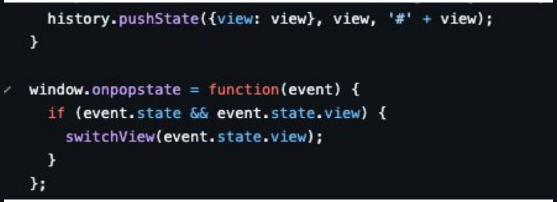
Authors: Vicayo Zhang, Aryan Ghosh, and Stella Dey

- Implemented multi-view options for the user to switch between
- Feature points estimate: 2
- Feature functionality is mostly complete; however, UI design for remaining views is yet to be completed.
- Git branch: Multi-View-User-Interface-Aryan & Multi-View-Aryan

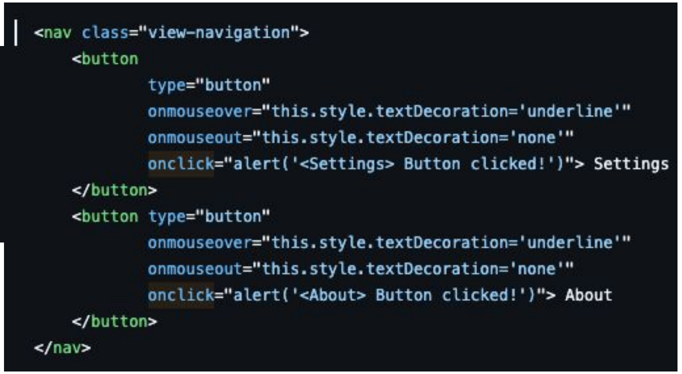
Aryan Ghosh - Code & UI Explanation



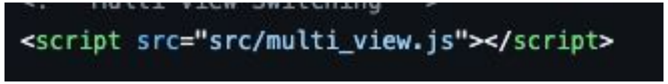
```
switch(view) {
  case 'terminal':
    container.innerHTML = `
      <div class="terminal-view">
        <h2>Terminal View</h2>
        <p>This is the terminal interface.</p>
      </div>
    `;
    break;
  case 'settings':
    container.innerHTML = `
      <div class="settings-view">
        <h2>Settings</h2>
        <p>Configure your settings here.</p>
      </div>
    `;
    break;
```



```
  history.pushState({view: view}, view, '#' + view);
}

window.onpopstate = function(event) {
  if (event.state && event.state.view) {
    switchView(event.state.view);
  }
};
```



```
<nav class="view-navigation">
    <button
            type="button"
            onmouseover="this.style.textDecoration='underline'"
            onmouseout="this.style.textDecoration='none'"
            onclick="alert('<Settings> Button clicked!')"> Settings
    </button>
    <button type="button"
            onmouseover="this.style.textDecoration='underline'"
            onmouseout="this.style.textDecoration='none'"
            onclick="alert('<About> Button clicked!')"> About
    </button>
</nav>
```



```
<script src="src/multi_view.js"></script>
```

switchView(view):

This function is the core of the multi-view logic. It looks for the container element by its ID (view-container). Based on the view parameter (for example, 'terminal', 'settings', or 'about'), it replaces the container's HTML:

- **Terminal view:** Injects the HTML for the terminal interface and provides a comment indicating where additional initialization code for the terminal could be placed.
- **Settings and About views:** Inserts simple HTML that you can later expand.
- **Default/Error case:** Handles unexpected view names.

• **Updating the URL:**
After replacing the content, the code calls history.pushState to update the URL without reloading the page. This lets users bookmark a specific view or use the back/forward browser buttons.

• **Handling Navigation Events:**
The window.onpopstate event handler listens for back/forward actions in the browser. When such an event occurs, it calls switchView() to update the view accordingly.

• **Default Behavior on Load:**
When the page loads, the code checks the current URL hash. If none is found, it defaults to the terminal view. This ensures that users see the appropriate view even if they navigate directly to a URL with a hash.

Aryan Ghosh - Challenges and Insights

- **Transitioning to Client-Side Navigation:**
  Moving away from traditional full-page reloads to a dynamic single-page approach was challenging. Using the History API to update the URL without reloading not only improved user experience but also required us to refactor our navigation to work entirely via JavaScript, ensuring smoother transitions between views.
- **Ensuring DOM Readiness and Seamless Updates:**
  A major obstacle was making sure the dynamic view container was available and that the content could be updated without interfering with the rest of the page. By listening for the window's load event, we guaranteed that our switchView function only executed when all DOM elements were fully available, preventing errors like null references.
- **Handling Browser Navigation:**
  Implementing client-side navigation meant we needed to support the back and forward buttons without reloading the page. Integrating history.pushState along with an onpopstate event handler ensured that users could navigate through the view history as expected, which was key for maintaining a natural user flow.
- **Collaboration and Integration:**
  Working in a team environment highlighted the importance of clear communication and proper version control. Regular pull requests and code reviews ensured that changes, especially those affecting the critical navigation logic, were well-integrated and conflict-free, which would eventually help us develop a better architecture for our application.

Aryan Ghosh - Future Improvements

Terminal Performance Improvements:

- **Issue:** Enhance the terminal's initialization speed and responsiveness by refactoring the terminal setup code.
- **Action:** Refactor the terminal setup code to avoid unnecessary DOM manipulations or redundant API calls during startup.
- **GitHub Issue:** https://github.com/VicayoMua/326_Mini_Terminal/issues/32?issue=VicayoMua%7C326_Mini_Terminal%7C33

UI & Accessibility Enhancements:

- **Issue:** Improve the user interface by adding keyboard navigation and high-contrast modes to increase accessibility.
- **Action:** Update the multi-view UI to support additional interactions and adhere to accessibility best practices.
  GitHub Issue: https://github.com/VicayoMua/326_Mini_Terminal/issues/32?issue=VicayoMua%7C326_Mini_Terminal%7C34

Stella Dey - Assigned Work Summary

Assigned Issues:

• #35: Form Validation and User Feedback
https://github.com/VicayoMua/326_Mini_Terminal/issues/35
• #37: Asynchronous File Handling
https://github.com/VicayoMua/326_Mini_Terminal/issues/37

Tasks Completed:

• Implemented an "Add File" button to dynamically create and inject files into the virtual file system within the terminal interface

• Developed an "Upload File" feature using FileReader to asynchronously read .txt and .json files from the user's local system.

• Validated file names and provided user feedback for empty or duplicate entries to improve the file creation experience.

• Displayed the uploaded file content in a preview panel for enhanced user interaction and confirmation.

Closed PRs:

• #36: Form Validation
https://github.com/VicayoMua/326_Mini_Terminal/pull/36
• #36: Upload Local File
https://github.com/VicayoMua/326_Mini_Terminal/pull/38

Commits Authored:

• 8c1857f: updated html file to add add-local-file window
• 157ef9d: Added js code for add_local_file validation
• a95921c: Added css code for add_local_file validation
• 0116cde: updated new file button and adding file to directory
• 3a9dac8: added uploading local file button and implemented adding it to directory
• 2d46989: minor changes
• 05f83a9: Merge pull request #36

Feature Demonstration

Feature Description:

I implemented file creation and upload functionality within the terminal interface.
1. The “Add File” button allows users to dynamically create and name new files inside the virtual file system.
2. The “Upload File” button asynchronously reads local .txt or .json files using FileReader and integrates them into the virtual file system. These updates are immediately reflected in the terminal output and preview panel.

Feature Type: Basic Feature (2 points)

A. Dynamic Content Updates
D. Form Validation and Feedback
F. Asynchronous Data Handling

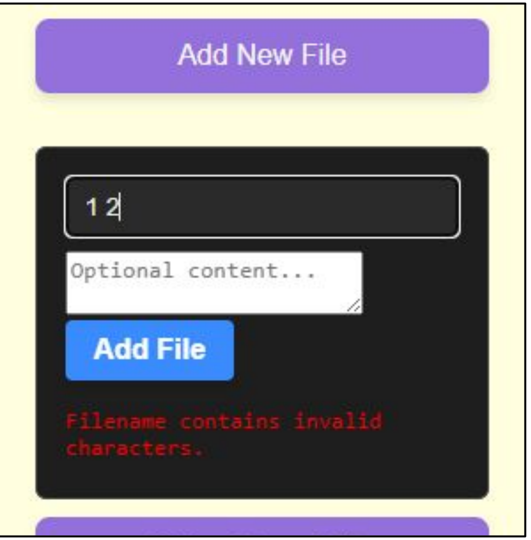
Completion Progress:

Both file creation and upload are fully implemented and functional, including error handling and file content preview.

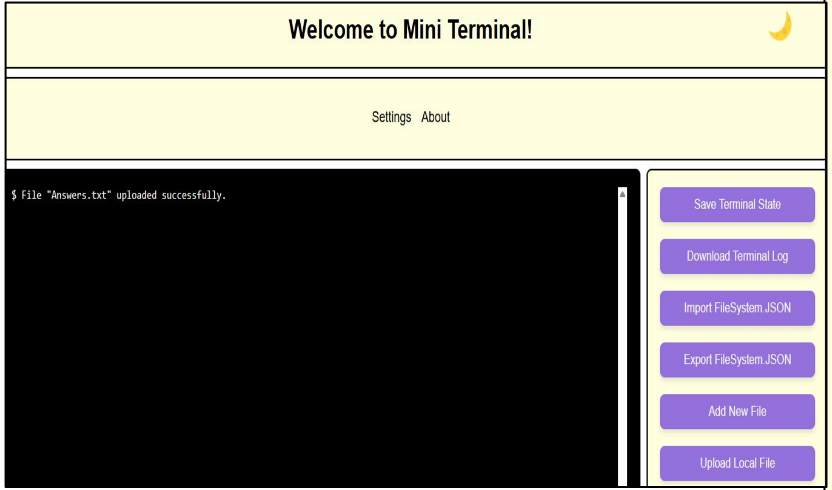
Git Branches:

https://github.com/VicayoMua/326_Mini_Terminal/pull/36
https://github.com/VicayoMua/326_Mini_Terminal/pull/38

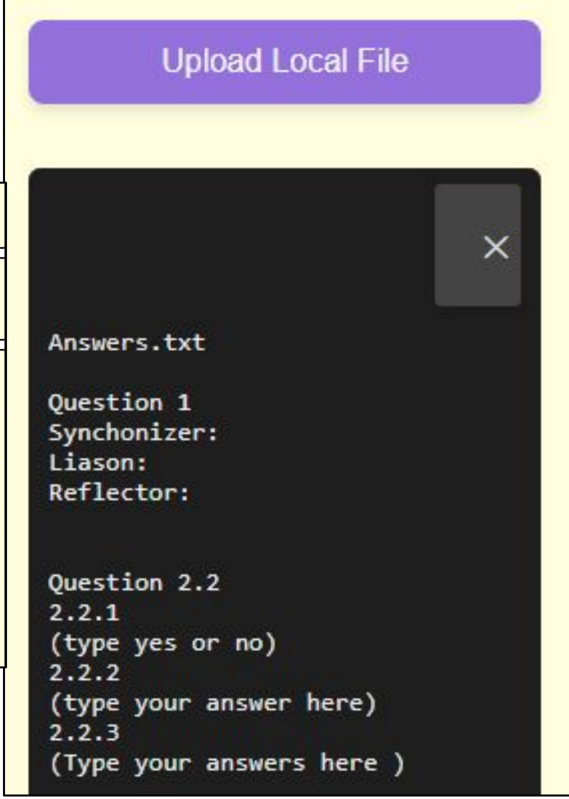
Stella Dey - Screenshots



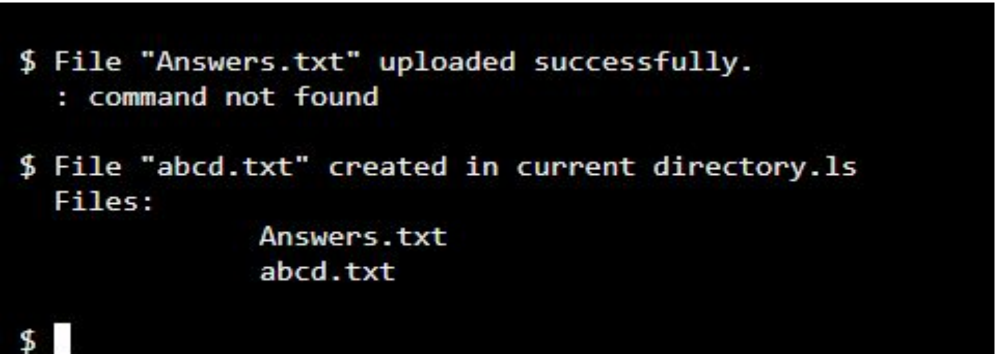
Filename contains invalid characters.



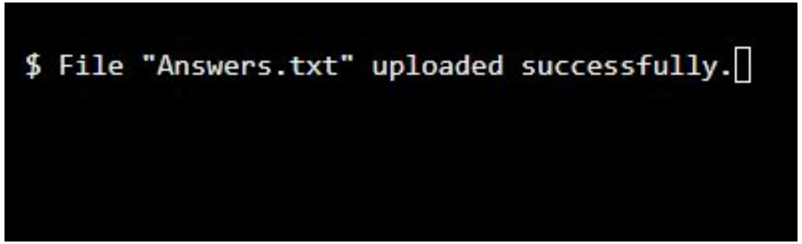
```
$ File "Answers.txt" uploaded successfully.
```



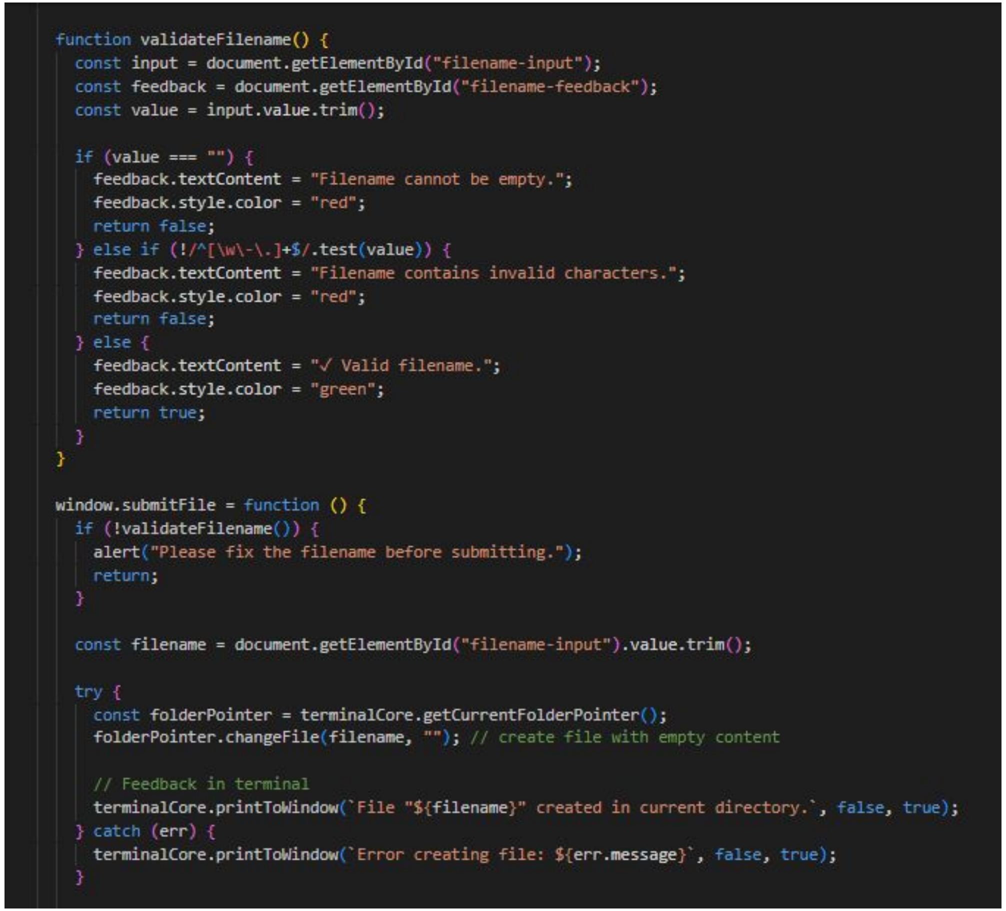
```
Answers.txt

Question 1
Synchonizer:
Liason:
Reflector:


Question 2.2
2.2.1
(type yes or no)
2.2.2
(type your answer here)
2.2.3
(Type your answers here )
```



```
$ File "Answers.txt" uploaded successfully.
  : command not found

$ File "abcd.txt" created in current directory.ls
  Files:
              Answers.txt
              abcd.txt

$ 
```

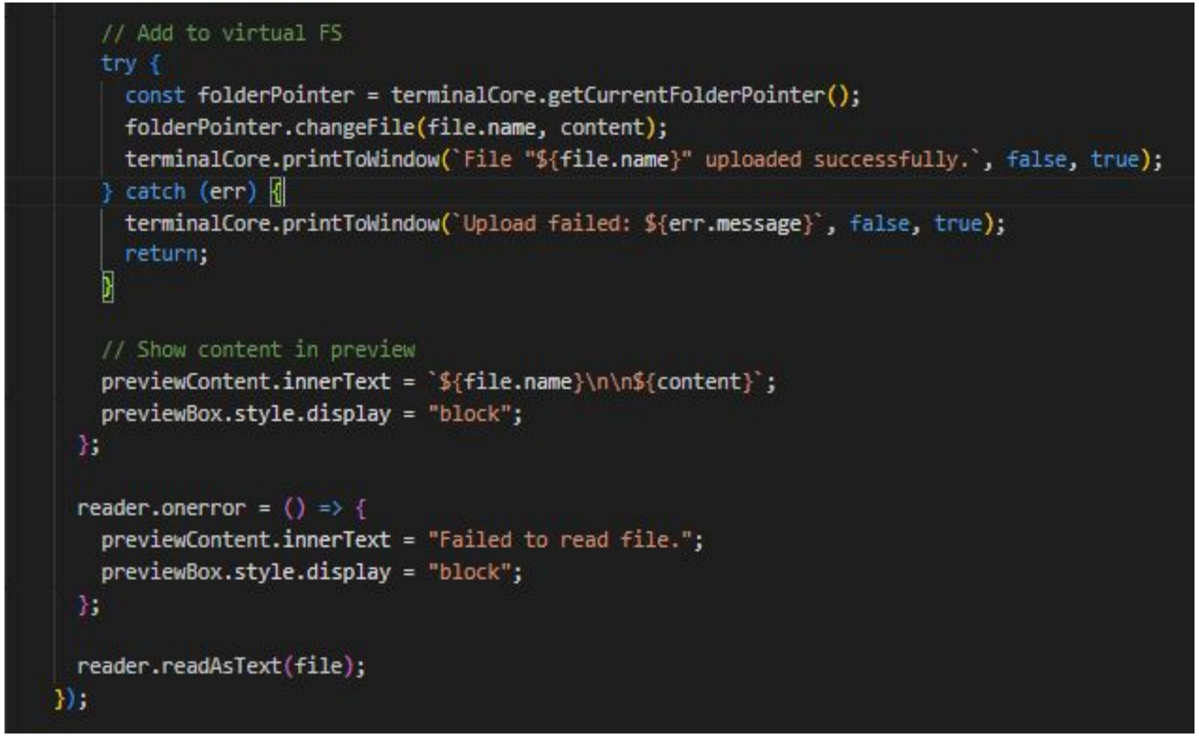


```
$ File "Answers.txt" uploaded successfully.
```



```
function validateFilename() {
  const input = document.getElementById("filename-input");
  const feedback = document.getElementById("filename-feedback");
  const value = input.value.trim();

  if (value === "") {
    feedback.textContent = "Filename cannot be empty.";
    feedback.style.color = "red";
    return false;
  } else if (!/^[\w\-\.]+$/.test(value)) {
    feedback.textContent = "Filename contains invalid characters.";
    feedback.style.color = "red";
    return false;
  } else {
    feedback.textContent = "✓ Valid filename.";
    feedback.style.color = "green";
    return true;
  }
}

window.submitFile = function () {
  if (!validateFilename()) {
    alert("Please fix the filename before submitting.");
    return;
  }

  const filename = document.getElementById("filename-input").value.trim();

  try {
    const folderPointer = terminalCore.getCurrentFolderPointer();
    folderPointer.changeFile(filename, ""); // create file with empty content

    // Feedback in terminal
    terminalCore.printToWindow(`File "${filename}" created in current directory.`, false, true);
  } catch (err) {
    terminalCore.printToWindow(`Error creating file: ${err.message}`, false, true);
  }
```

Stella Dey - Code Explanation



```
    // Add to virtual FS
    try {
      const folderPointer = terminalCore.getCurrentFolderPointer();
      folderPointer.changeFile(file.name, content);
      terminalCore.printToWindow(`File "${file.name}" uploaded successfully.`, false, true);
    } catch (err) {
      terminalCore.printToWindow(`Upload failed: ${err.message}`, false, true);
      return;
    }

    // Show content in preview
    previewContent.innerText = `${file.name}\n\n${content}`;
    previewBox.style.display = "block";
  };

  reader.onerror = () => {
    previewContent.innerText = "Failed to read file.";
    previewBox.style.display = "block";
  };

  reader.readAsText(file);
});
```

The upload code integrates tightly with the app's virtual file system (VFS) by using changeFile() to insert uploaded content directly. This makes the files immediately accessible through terminal commands like ls and cat. Terminal feedback is provided via printToWindow(), and a live preview of the uploaded content is shown in the right panel, connecting both the CLI and GUI layers for a seamless user experience.

Async File Read: Handled using FileReader.onload with error fallback via onerror.
Upload Failures: Used try-catch to catch and display file system errors.
Terminal Integration: Used changeFile() to ensure uploaded files work with terminal commands.
UI Sync: Preview panel updates alongside terminal feedback for consistent UX.

Stella Dey - Challenges and Insights

Challenges & insights

One of the main challenges was handling asynchronous file uploads while integrating them into the virtual file system without disrupting existing terminal functionality. Ensuring seamless feedback across both the terminal and preview panel required careful coordination between UI and CLI layers. I also had to implement form validation and error handling to improve usability and prevent incorrect inputs. I learnt how small details can really affect the user experience.

Collaborative Takeaways

Working in a team taught me a lot about staying in sync with others. Since someone else was working on persistence, I made sure my feature (file creation and upload) could stand on its own without interfering. It was a great reminder of how important it is to keep your code modular and to communicate often so nothing clashes. Overall, I learned how to build my part in a way that fits into the bigger picture.

Stella Dey - Future Improvements

1. Rename and Delete File Functionality - Currently, users can only add or upload files. A natural extension would be to allow them to rename or delete files directly through the interface or terminal commands like rm or mv.. This would improve file system management and align with standard terminal behavior.

Github issue link: https://github.com/VicayoMua/326_Mini_Terminal/issues/32?issue=VicayoMua%7C326_Mini_Terminal%7C39

2. File Type Detection and Icon Preview - Right now, all uploaded files are treated the same. Adding basic file type detection (e.g., text vs JSON vs image) and preview icons in the UI could improve the user experience, especially for large or varied uploads.

Github issue link: https://github.com/VicayoMua/326_Mini_Terminal/issues/32?issue=VicayoMua%7C326_Mini_Terminal%7C40