COMPSCI 326 Group 1 (Online)

Mini Terminal

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

Milestone 4

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

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.

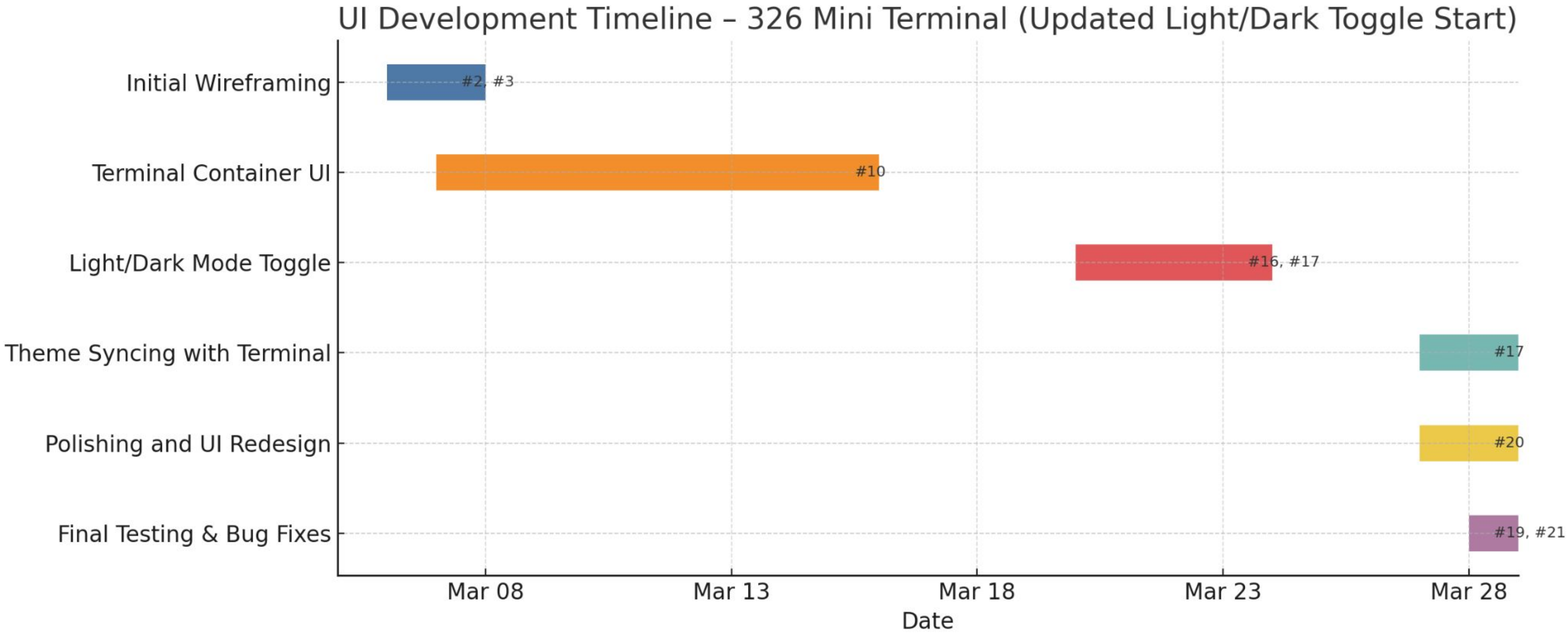
Key Features:

- 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**

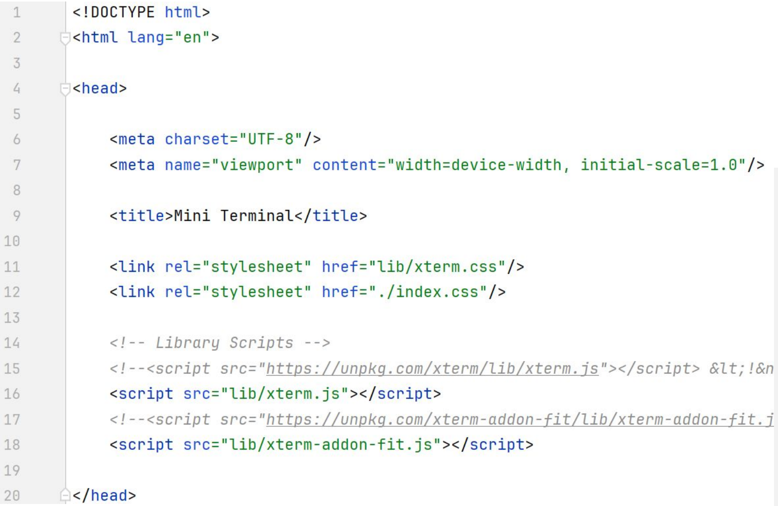
Historical Development Timeline



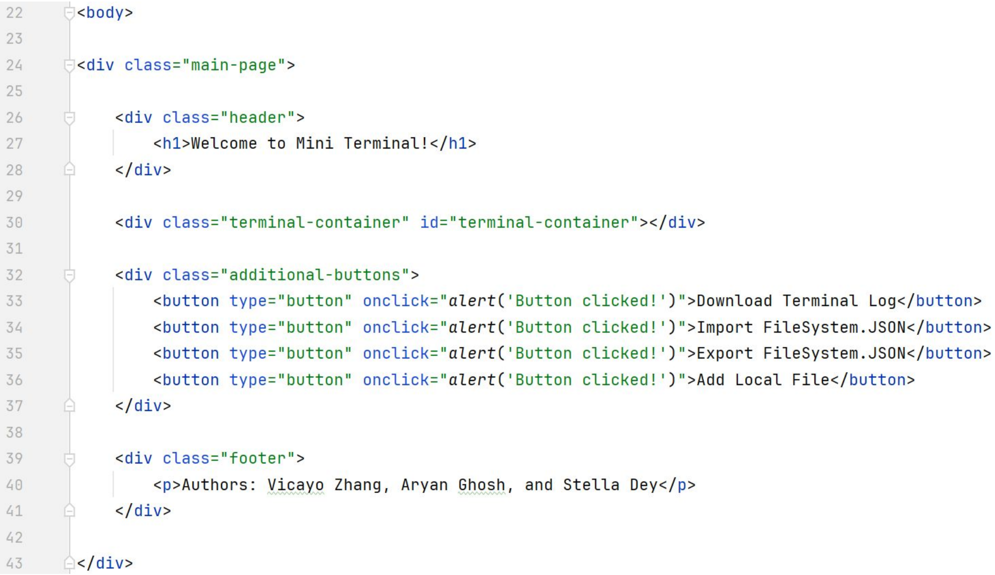
Vicayo Zhang - Assigned Work Summary

- **Developing the user interface for the Command Input Screen where users directly interact with the terminal to execute Linux commands.**
- **Implementing html elements, such as quick-access buttons.**
- **Implementing the basic css stylesheet.**

Vicayo Zhang - Screenshots



```html
<!DOCTYPE html>
<html lang="en">

<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;!&n
    <script src="lib/xterm.js"></script>
    <!--<script src="https://unpkg.com/xterm-addon-fit/lib/xterm-addon-fit.j
    <script src="lib/xterm-addon-fit.js"></script>

</head>
```



```html
<body>

<div class="main-page">

    <div class="header">
        <h1>Welcome to Mini Terminal!</h1>
    </div>

    <div class="terminal-container" id="terminal-container"></div>

    <div class="additional-buttons">
        <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>
    </div>

    <div class="footer">
        <p>Authors: Vicayo Zhang, Aryan Ghosh, and Stella Dey</p>
    </div>

</div>
```

Vicayo Zhang - Code & UI Explanation

In the HTML file:

- **the head part describes the title of the webpage, includes the stylesheets that the webpage needs, and imports the javascript libraries;**
- **the body part introduces the HTML elements: the header bar, the terminal window container, quick-access buttons, and the footer bar.**

In the css file:

- **the styles of the header bar, the terminal window, quick-access buttons, and the footer bar are specified;**
- **the window styles for different screen resolutions are specified.**

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.**

Vicayo Zhang - Future Improvements

- **Implementing the javascript functions so that we can better design the user interface.**

Aryan Ghosh - Assigned Work Summary

1. **UI Design for Command Input Screen**:
   - Developing the user interface for the Command Input Screen where users directly interact with the terminal to execute Linux commands.
   - Implementing features such as an Execute Button, Command History, and additional quick-access buttons for saving logs, importing, and exporting file systems.
   - Implemented cross border compatibility so that the browser windows resize across different browsers.

https://github.com/VicayoMua/326_Mini_Terminal/issues/2

https://github.com/VicayoMua/326_Mini_Terminal/pull/22

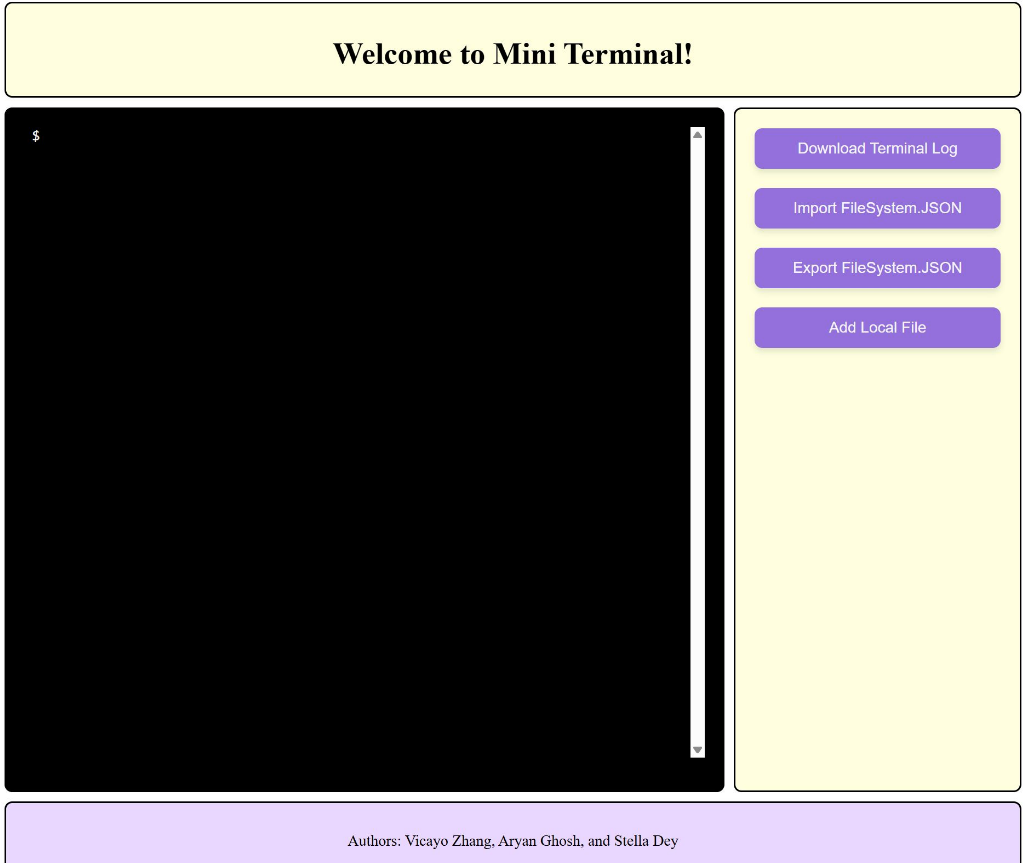
2. **Session Management Data Design (Yet to implement)**:
   - Designing how user sessions are managed and stored, enabling users to save their terminal sessions and resume later.
   - Structuring the session attributes including SessionID, UserID, CommandsEntered, and LastAccessed.

3. **Software Development and Admin Monitoring (Yet to implement javascript code)**:
   - 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.

Aryan Ghosh - Screenshot

Unable to show browser

window resize



Welcome to Mini Terminal!

```
$
```

Authors: Vicayo Zhang, Aryan Ghosh, and Stella Dey

Aryan Ghosh - Code & UI Explanation

Explanation of Code:

- **Grid System**: Utilizes CSS Grid for layout structure, making it flexible and easy to shift between multi-column and single-column layouts based on screen size.
- **Flexbox in Additional Buttons**: Flexbox properties enhance the positioning flexibility within the grid layout, especially for reordering content dynamically.

Challenges Faced and Solutions Implemented:

- **Challenge**: Ensuring the UI remains accessible and functional across devices of varying sizes, especially smaller screens where space is limited.
- **Solution**: Implemented CSS Media Queries to adapt the layout dynamically, shifting to a single-column format on smaller screens to enhance readability and usability. Adjusted padding and font sizes within these breakpoints to ensure interface elements are still easy to interact with.
- **Challenge**: Different browsers rendering the layout and font sizes inconsistently.
- **Solution**: Used box-sizing: border-box to include padding and border in the width and height calculations, ensuring consistency across browsers. Set a common font family (Arial, sans-serif) to prevent font rendering issues.

Impact on UI/UX:

- Improved user experience by making the UI adaptable to any screen size, ensuring that elements are not only visible but also functionally interactive across devices.
- Enhanced accessibility by ensuring text readability and easy navigation, particularly on mobile devices where screen real estate is at a premium.

```
@media (max-width: 768px) {
  .main-page {
    grid-template-columns: 1fr; /* Switch to a single column layout */
  }
  .additional-buttons {
    order: -1; /* Moves the buttons above the terminal container */
  }
}

@media (max-width: 480px) {
  .header, .footer {
    padding: 8px; /* Smaller padding for smaller screens */
  }
  .additional-buttons button {
    padding: 8px 16px; /* Smaller button size for ease of use on mobile */
    font-size: 14px; /* Smaller font size for better readability on small devices */
  }
}

/* Ensuring Cross-Browser Compatibility */
* {
  box-sizing: border-box; /* Includes padding and border in element's total width and
height */
}
html {
  font-family: 'Arial', sans-serif; /* Common font for consistency across different browsers */
}
```

Aryan Ghosh - Challenges and Insights

1. Responsive Design Implementation:

- **Obstacle**: Adapting the UI to function seamlessly across various devices, from large desktop monitors to compact mobile phones.
- **Insight**: Learned the importance of meticulous CSS media queries and the utility of flexible grid and flexbox layouts to ensure responsive design principles are upheld.

2. Cross-Browser Compatibility:

- **Obstacle**: Ensuring that all elements display consistently across different web browsers, which interpret CSS rules differently.
- **Insight**: Gained proficiency in using CSS normalization techniques and understanding browser-specific quirks, enhancing the robustness of web applications.

Lessons Acquired:

Effective Communication: Understanding that regular and clear communication among team members is crucial to synchronize efforts, clarify requirements, and share solutions to common problems.

Emphasis on User-Friendly Design: Realized the importance of always considering the user's perspective and accessibility needs when designing and implementing new features, ensuring the product remains usable and accessible to all users.

Importance of Feedback: Regular feedback, both giving and receiving, was instrumental in refining the project and personal development. Constructive criticism helped improve the quality of the work and foster personal growth.

Project Management Skills: Enhanced project management skills including task delegation, prioritization, and deadline management, which are essential in a collaborative project setting to ensure timely and successful project completion.

Aryan Ghosh - Future Improvements

1. **Mobile Responsiveness:**
   Improvement: Implement additional media queries and test interfaces on various mobile devices to ensure better usability across all screen sizes.
   GitHub Issue: https://github.com/VicayoMua/326_Mini_Terminal/issues/23
2.

Stella Dey - Assigned Work Summary

Assigned Issues:

- #17: Customizable Theme - Light/Dark mode
- #21: UI Code Testing for Milestone #4

Tasks Completed:

- Built toggle UI (sun/moon button, position adjustment)
- Created CSS variable system for theming
- Hooked into checkbox state change to toggle .dark-mode class
- Fixed spacing, styling, and positioning bugs via multiple commits

Closed PRs:

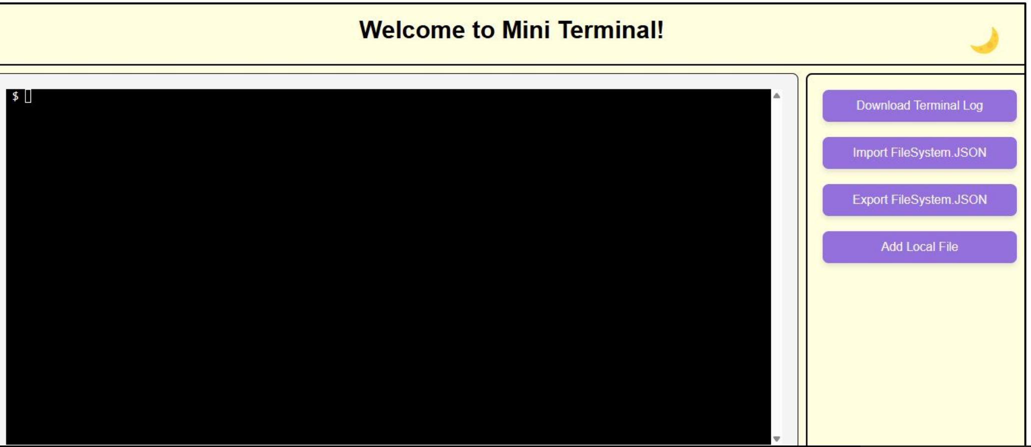
- #24: Light Dark Mode

https://github.com/VicayoMua/326_Mini_Terminal/pull/24

Commits Authored:

- 34a817e: light/dark mode first draft
- 4b0bdc5: toggle button positioning
- 474d75b: final theme fix

Stella Dey - Screenshot (Light/Dark Mode)

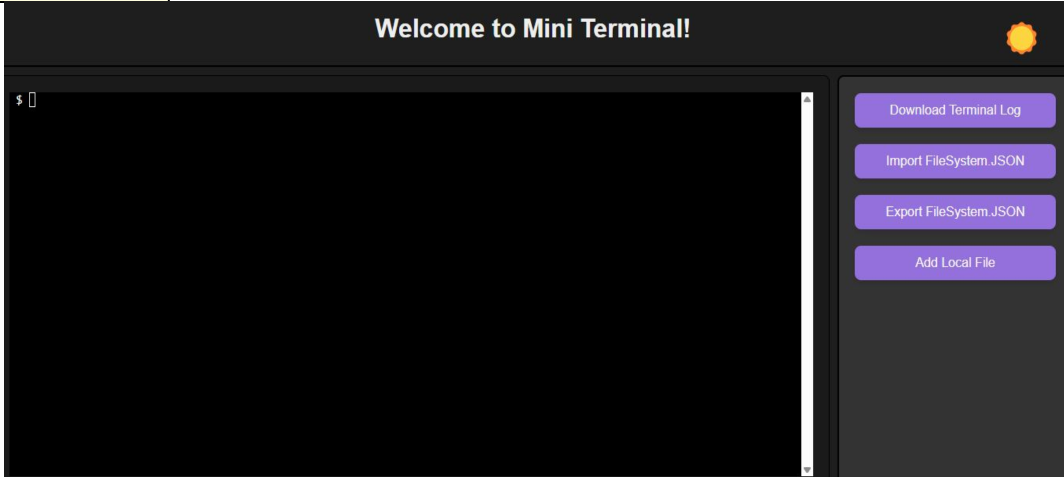


Welcome to Mini Terminal!

```
$ ▯
```

Light Mode

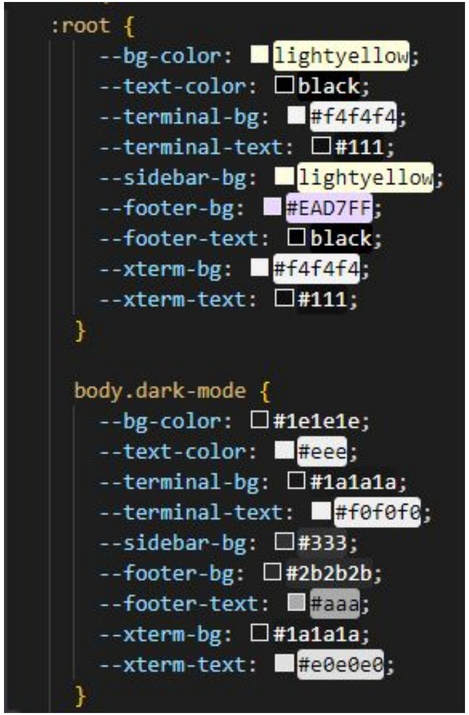
Dark Mode



Welcome to Mini Terminal!

```
$ ▯
```

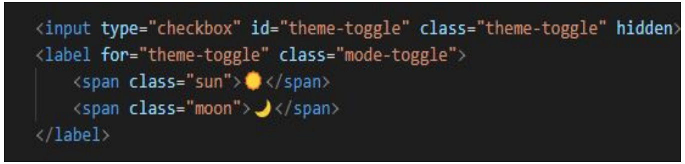
Stella Dey - Code & UI Explanations



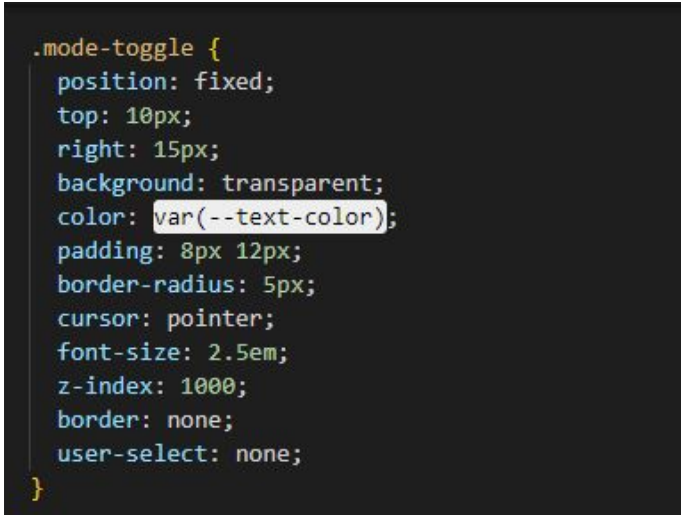
```css
:root {
    --bg-color: lightyellow;
    --text-color: black;
    --terminal-bg: #f4f4f4;
    --terminal-text: #111;
    --sidebar-bg: lightyellow;
    --footer-bg: #EAD7FF;
    --footer-text: black;
    --xterm-bg: #f4f4f4;
    --xterm-text: #111;
  }

  body.dark-mode {
    --bg-color: #1e1e1e;
    --text-color: #eee;
    --terminal-bg: #1a1a1a;
    --terminal-text: #f0f0f0;
    --sidebar-bg: #333;
    --footer-bg: #2b2b2b;
    --footer-text: #aaa;
    --xterm-bg: #1a1a1a;
    --xterm-text: #e0e0e0;
  }
```

The theme toggle integrates into the overall UI by dynamically adding or removing a dark-mode class on the <body> element. All visual components—including the header, terminal container, sidebar, and footer—reference CSS variables (like --bg-color and --text-color) for styling. When the dark-mode class is active, these variables are redefined, allowing the entire interface to switch themes consistently with minimal code duplication.

This approach ensures centralized styling control and keeps the UI modular and easy to maintain.



```html
<input type="checkbox" id="theme-toggle" class="theme-toggle" hidden>
<label for="theme-toggle" class="mode-toggle">
    <span class="sun">🌞</span>
    <span class="moon">🌙</span>
</label>
```



```css
.mode-toggle {
  position: fixed;
  top: 10px;
  right: 15px;
  background: transparent;
  color: var(--text-color);
  padding: 8px 12px;
  border-radius: 5px;
  cursor: pointer;
  font-size: 2.5em;
  z-index: 1000;
  border: none;
  user-select: none;
}
```

Stella Dey - Challenges and Insights

Challenges faced

- Figuring out how to apply theme changes across the whole UI without messing up the layout
- Managing merge conflicts during colab
- Making sure the CSS was neat and reusable

Insights

- Using CSS variables made theme switching way easier and cleaner
- Debugging layout issues taught me to test changes often and stay patient
- Working with a team helped me understand how important communication and clear roles are
- I got more comfortable with Git — especially dealing with conflicts and rollbacks

Stella Dey - Future Improvements

1. Save theme preference across sessions - Store user's theme choice in localStorage so it stays applied after refresh or reopen.

Github issue link:
https://github.com/VicayoMua/326_Mini_Terminal/issues/25

2. Improve accessibility - Ensure contrast ratios are good in both modes and add keyboard shortcuts for toggling theme.

Github issue link:
https://github.com/VicayoMua/326_Mini_Terminal/issues/26