A. Description of Testing Environment:

1.

Operating System: Windows 10 (Home Chinese Edition) (64-bit), Version 22H2
Browser and Version: Chrome 110.0.5481.178 (Official Build) (64-bit) / Mozilla Firefox (version 120.0.1/64-bit) / Microsoft Edge (version 120.0.2210.61/64-bit)
Computer Architecture: 64-bit
Node.js Version: v20.10.0
typescript@5.3.3

2.

Operating System: mac Monterey 12.7.1
Browser and Version: Safari 16.0 / Chrome 120.0.6099.109/ Mozilla Firefox 10.11.6 / Microsoft Edge 10.13.6
Computer Architecture: 64-bit
Node.js Version: v20.10.0
typescript@5.3.3

B. Testing Methods:

Set Up the Project

1. **npm install**: Install all necessary dependencies.
2. **npm run build**: Obtain an executable index.html file within the dist folder.
3. **npm run dev**: Open the server-side project in VSCODE, then run npm run dev in the terminal to start the server.
4. Open the **index.html** in the **dist** folder in a browser. You'll be able to view the entire mailbox layout and test its functionality within that browser.

To ensure my code works effectively for the majority of users accessing the web page served by the Node.js server, I would conduct the following tests:

-Cross-browser Compatibility Testing: I would test the web page on various browsers such as Chrome, Firefox, Safari, and Edge to ensure compatibility and consistent rendering.
-Operating System Testing: I would verify the functionality of the webpage on different operating systems like Windows, macOS, and Linux to ensure a seamless user experience across platforms.
-Device Testing: I would test the webpage on different devices such as desktops, laptops, tablets, and mobile phones to ensure responsiveness and proper display on various screen sizes.
-Network Environment: I would test the webpage's performance on different network speeds (such as high-speed and slower connections) to ensure it loads efficiently and is accessible under varying network conditions.

C. How AJAX helps my web application:

AJAX, facilitated by modules like Axios, greatly enhances client-side interactions. It elevates responsiveness by enabling selective updates of elements within the UI without the necessity of refreshing the entire page. This lightweight nature of AJAX requests contributes to a smoother user experience, as it avoids the need to reload the entire webpage for each specific action (GET/PUT/POST/DELETE), focusing only on refreshing the necessary components.