**BUG ANALYSIS**

1. [Feature request] Add option to always open inaccessible files

The feature request highlights a usability concern in Notepad++ where, at startup, the application prompts the user with a popup for inaccessible files that were open in the previous session. This can become a repetitive and unnecessary interruption, especially for users who consistently choose to reload all files regardless of their current accessibility status. The requester notes that while the current "Ask" behavior may be suitable for some users, it introduces the risk of mistakenly clicking "No" during a routine startup, particularly when fatigued or distracted. To streamline the user experience, the suggestion is to introduce a configurable option in Notepad++ settings that allows users to define their preferred behavior when such files are encountered.

2. [BUG] case shift functions drop last character selected

The reported bug in Notepad++ v8.7.7 describes an issue with the **case shift functions** (such as "UPPERCASE" or "lowercase") where, when applied to a selected portion of text, the **last character in the selection is dropped** after the operation. This behavior occurs regardless of whether the selection includes characters, spaces, or control characters like line feeds (LF) or carriage returns (CR). The expected behavior is for the transformation to preserve the full selected range, modifying all characters without truncating any part of the selection.The bug is easy to reproduce by simply selecting a portion of text and applying a case shift function—upon doing so, the final character of the selected area does not appear in the transformed result. This suggests an off-by-one error or mishandling of the selection range in the implementation of the case transformation logic. The issue is particularly problematic for users working with precise text formatting or code editing, where even a single dropped character can cause syntax errors or data loss.

3. On update: can't cancel update when unsaved files are open: canceling save dialog in a loop

This bug report highlights a user experience issue in Notepad++ v8.6.8 related to the **update process when unsaved files are open**. When initiating an update while there are unsaved files, Notepad++ prompts the user to save or discard changes. However, if the user chooses to **cancel** the save dialog — presumably to **abort the update altogether** — the dialog simply **reappears in a loop**, and the update cannot be canceled cleanly. Even if the user cancels the background update progress dialog, the save prompt keeps reappearing, effectively **trapping the user in a loop** until they either save or discard changes. This behavior prevents a graceful exit from the update operation and can be particularly frustrating if the user opened the update unintentionally or wants to postpone it. The **expected behavior** would be that canceling the save dialog (especially after already canceling the update process) should completely **abort the update flow** and return the user to their unsaved session without forcing a decision to save or discard. Alternatively, Notepad++ could implement a mechanism to **temporarily preserve unsaved files** and restore them after the update completes. The bug likely stems from improper state handling between the update progress and save dialog logic, where cancel actions aren't being properly propagated or honored.

4. [BUG] Incorrect file position

This bug report details an issue in Notepad++ v8.7.5 where the **file position displayed in the status bar is incorrect when the file contains accented characters**, even when using single-byte encodings like ISO-8859-1 or Windows-1252. Two nearly identical PDF files were compared — one with accented characters (éèçà) and one with plain ASCII characters (aaaa). Despite the fact that both versions should be byte-for-byte identical aside from those four characters, Notepad++ reports a different position when the cursor is placed at the same logical point (e.g., before the line containing xref). Specifically, the presence of accented characters causes the cursor position to report **a higher byte offset** than it actually is, which is misleading since these accented characters are **stored as single bytes in ISO-8859-1**. This incorrect position reporting was further confirmed by manually creating a test file with four accented characters and observing that the position showed as 9 instead of the expected 5, indicating that Notepad++ is miscalculating character length — possibly treating accented characters as multi-byte or wide characters internally. This behavior points to an inconsistency between the **file’s actual encoding (byte-oriented)** and the **rendering engine’s internal Unicode representation**, which then affects the accuracy of the byte offset shown in the UI.This bug has significant implications when working with binary or structured text files like PDFs, where **precise byte positions are critical**, and users expect the position display to reflect the real offset in the file. The issue seems rooted in how Notepad++ translates character positions to byte offsets, particularly for ANSI encodings with extended characters.

5. [BUG] double click on url issue

This bug report identifies a usability issue in Notepad++ where **double-clicking on a URL within code** (e.g., url = '<http://www.google.com/>') incorrectly includes surrounding punctuation such as single quotes (') or double quotes ("). As a result, when the URL is automatically launched in the default browser, it includes the quote character at the end — leading to malformed URLs like <http://www.google.com/>', which often cause the browser to return a **"404 Not Found"** or **"Forbidden"** error. The core problem is that Notepad++'s **word selection logic** during a double-click does not differentiate between URL content and surrounding syntax in source code. Ideally, the application should **only select the actual URL string**, excluding any enclosing punctuation such as ', ", or even trailing characters like commas or semicolons, which are common in code assignments. The **expected behavior** is that double-clicking on a URL should intelligently identify and select just the URL portion — <http://www.google.com/> — without any trailing or leading non-URL characters, ensuring that when it's opened in the browser, the navigation is successful and accurate. A fix for this would likely involve updating the logic that handles double-click selection and link detection, making it more context-aware in code environments.This would significantly improve user experience, especially for developers frequently testing or referencing URLs within scripts, configuration files, or logs.