Feature requests:

The objective here is to give some insights into the way I go about considering whether and to what extent to implement feature requests, as I am sure many requesters feel frustrated at my unwillingness to tackle what they may see as easy changes. I hope this goes some way to explaining why I am often reluctant.

Overview:

My strategy for adding a feature (when I have time available) is to consider:
a) the level of demand - the higher the demand the more consideration it gets;
b) whether it represents a new feature or a convenience - priority given to new features unless a convenience is very easy to incorporate;
c) ease of modification - the easier the more likely to be incorporated;
d) the potential for disrupting stable aspects of the program - the more likely this is the more wary I am of tackling it; and
e) the degree of complexity it adds for the user - new users often find the current interface quite overwhelming, so something that adds complexity for the user should be offset by substantial benefits to be considered.

Of these a), b) and e) are relatively easy to assess, but (c) and (d) are very much harder. Even after starting a modification unforeseen effects, sometimes in what would appear to be completely unrelated parts of the code, often emerge that are quite difficult and time consuming to deal with. Several of these cropped up with the timetable event skipping functions and with the random delays and track element failures. This is the reason I am reluctant to announce in advance that I am working on a new feature. It's quite possible that unwanted effects can emerge late in development that raise the difficulty level immensely, sometimes to the point of scuppering the whole process.

Detail:

There are several reasons not to implement a request:

a) If the request is impossible without a major restructuring of the program (e.g. 3-way points [switches] or crossovers) then it will be rejected. The program is what it is, I don't have the willpower/energy/stamina/drive/time to start all over again!

b) Other requests are subject to an informal effort/benefit analysis where the benefit must justify the effort required, bearing in mind that the effort is always understimated initially as so many unforeseen and often complex factors emerge during coding. Effort assessment also includes the potential for introducing bugs and the extent of testing required to find them - this often requires the development of special functions to carry out the tests. Benefits should do something that either can't be done without changes or are very difficult or time consuming to do by other means, and are genuinely useful in enhancing enjoyment and/or assisting people when using the program. Benefits that are 'nice to have' without other redeeming features are done only if the effort needed and risk of creating bugs is low.

c) What appears to be a relatively simple change from a user's point of view, and is simple to access during use of the program, is often surprisingly complex to implement. For example being able to change a service description in a timetable seemed like a simple function addition but required significant changes to session file saving and loading, several other timetable functions - especially for repeating services, train splitting, and log file creation.

Most requests are added to a wishlist kept in a file called DevHistory.txt at https://github.com/AlbertBall/railway-dot-exe/blob/Post_v2.20.0_Dev/DevHistory.txt. The latest request number is 109, though 32 have been done and there are some duplicates. When a change is to be made in a particular area I check the list to see if there are other requests in that area, as it's often easier to make two or more changes at the same time than separately.

Modifying code is a risky and often messy and tedious business, several things can cause problems:

a) Other parts of the program not directly involved with the new feature are detrimentally affected - this almost always happens. For example in allowing non-station named elements (blue squares) to be placed on points, crossovers and diagonals, several of the train splitting functions had to be redesigned and rewritten from scratch, and extensive new routines developed to test them effectively. This wasn't foreseen initially but added to the worload considerably and was required because trains could split when on much more complex elements than before.

b) Modifications are mostly done using 'patches', which are pieces of code that change the behaviour of a function or functions without having to redesign and rewrite the original function(s). These are very prone to causing bugs that can affect what seem to be unrelated aspects of the program, requiring extensive and time consuming testing procedures to try to find and correct them. What is most feared is that the new code goes out containing unrecognised bugs that are found by users - very embarrassing! That has happened several times, for example in allowing routes to be truncated from the front as well as from the back a user discovered that when truncating a route that led to buffers or a continuation the signal in a linked rear route didn't change to red as it should have done. It was caused by a bug in the patch that didn't affect route truncation but caused the signal setting routine to work incorrectly - i.e. a routine that wasn't directly involved in route truncation.

c) It's possible to come to a dead stop in making a change. Fortunately this has only happened once so far. It was when I was trying to implement a multiplayer feature. I intended that players, each controlling adjacent areas of a railway could work together to signal trains between the areas. I had most of the handshaking routines done but during testing found that for no obvious reason connections kept dropping. The documentation wasn't as good as it could have been and I could find no help from the web, so progress stalled. I was working in very unfamilar territory with internet functions and reluctantly had to give up. Perhaps someone much more skilled than me in this area will pick this up in the future as it would add considerably to playability. Documentation is at https://github.com/AlbertBall/railway-dot-exe/tree/Post_v2.20.0_Dev/Multiplayer%20Development%20Docs. Much of the code remains in the program but is not accessed.

****