**Last Stop Development Journal**

|  |  |
| --- | --- |
| Wed March 11 | 4:30 PM. Development begins. I’m debating between using the .NET Framework in C# and between using Electron/TypeScript. After a bit of research, I conclude that using C# is the more prudent decision. It feels like this project would benefit from a more robust type system, and Dragon is able to use its full feature set with Windows Forms. Otherwise, I’d have to do RPC with a WinForms stub.  5:30 PM. First check in. Titled the project “Last Stop” because I hope this is the last time I’ll ever try to make an editor. Feeling discouraged, low energy. I’ve been here before and it doesn’t feel good to start from zero again. This really has to be the last time.  Decision paralysis sets in. Not sure where to start, what the project scope should be. The ultimate goal is to be able to make software by voice with no compromises in any language (provided I’ve made voice definitions for it). Many ways to achieve that goal, all different.  Preliminary structures:   * Model: a data object containing all reversible (i.e. voice controlled) information for the application. The model is only modified by deltas. Snapshots can be taken of the model (these are stored OUTSIDE the serialized model). Undo commands amount to re-applying stored deltas to a recently taken snapshot. (This prevents the need for storing inverse deltas, which is really a pain in the butt.) * External: an object containing all information which is not reversible, that is, outside of voice control. Things like changes on the file system, time/date, network access, and so on belong here. * View: The collection of windows which subscribe to the model. * Windows: each window subscribes to a certain view of the model. Changes in the model trigger events sent to the window. contains a single UserControl and subscribes to a certain view of the model. * Content: a character-matrix UserControl with background/color/in-text markup options * Input: a textbox UserControl   5:54 PM. Taking a break.  9:18 PM. Resuming!  Let’s think about how to carve this program up into separable components.  #1: View components. They know nothing about the model or where their data is coming from – they simply display what clients tell them to display and raise events back to clients when stuff happens.  #2. Model. Consists of open buffers and their text, cursors/anchors, etc.  #2: Project/file system monitor. This tracks what files and directories exist under the project root and can raise events (maybe? Or is it polling?) when files have been modified, added, or deleted. Clients can issue save and load commands, but this module need know nothing about why.  #3: Command parser. The front end for lexical analysis and parsing of spoken text takes spoken language definitions (which are guaranteed not to change between snapshots) and spits out a list of model deltas and/or irreversible commands.  #4: Syntax highlighter.  11:19 PM. Decided to use F# when I realized it’s basically a modernized, .NET compatible OCaml. Nice of them to let me know! It’s *good shit though.* Done for the night.  **Daily effort: 3h 25m**  **Total effort: 3h 25m** |
| Thu March 12 | 8:34 AM. It looks like I’m either using .NET Core with VS code (and the nice editor) or .NET Framework with Visual Studio (and QEdit). That’s going to restrict my options either way. I’m thinking it might be smart to just go with [15 minute break] Visual Studio. QEdit isn’t that bad. Break @ 9:00 AM  11:37 AM. Going to investigate mixing framework and core. Okay, it really is impossible. As far as I can tell… I’m going to stick with Framework and use QEdit. Time to start making defs. Break 28 min.  2:45 pm stopping, mostly done with defs |