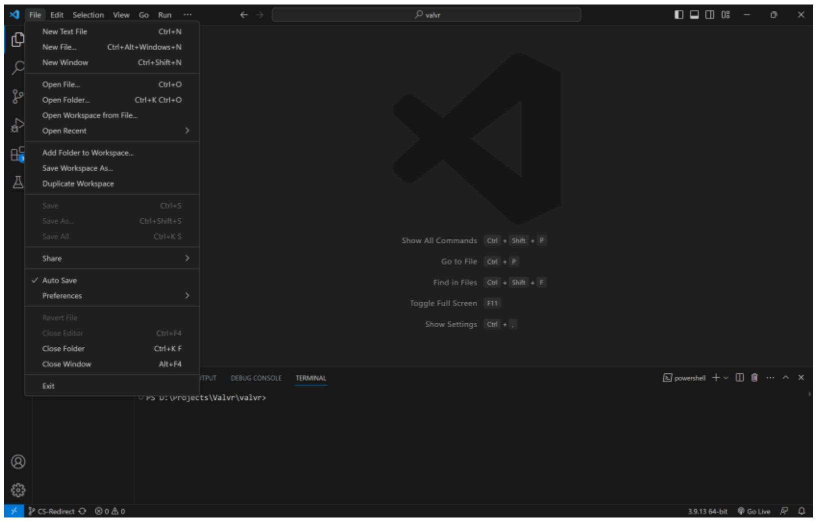
Surface Bag

The Microsoft Surface bag serves as an everyday necessity for professionals and students alike. This paper delves into the usability of this bag, examining its design through principles like affordances, signifiers, mappings, and feedback. Beyond merely holding a Surface device, the bag offers padded compartments and Velcro straps for added security. It's also versatile, designed to be carried by hand, over the shoulder, or inside a larger backpack. Design elements like an orange handle or soft felt lining act as signifiers. They subtly guide user interaction, the orange handle draws attention for carrying, while the felt indicates the safe zone for the Surface device. The bag's compartmental design shows effective mapping. Each zipper clearly correlates with a specific compartment, making navigation straightforward and minimizing cognitive load for the user. The bag provides unspoken yet effective feedback. Resistance when lifting the handle or the sound of zippers and Velcro straps give users intuitive cues about the bag's state and the security of its contents. The underlying conceptual model defines the Surface bag as a portable, protective, and convenient sanctuary for your device. It's not just a storage solution; it's designed with both the device and user in mind. The Surface bag's usability is a result of well-considered design elements that harmonize functionality and user experience. Each feature, from affordances to feedback, is thoughtfully planned to make the bag not just functional, but also exceptionally user-friendly.

Visual Studio Code



```
PS D:\Projects\Valvr\valvr>
```

Visual Studio Code (VS Code) is an Integrated Development Environment (IDE). This paper aims to dissect the usability of VS Code through design principles like affordances, signifiers, mappings, feedback, and its underlying conceptual model. The 'File' tab in VS Code contains a lot of essential affordances for the IDE. From basic actions like creating a new text file to duplicating an entire workspace, there are a lot of affordances, so I won’t be covering all of them. Some notable affordances include File Manipulation: Options like 'Save a file', 'Open a file', and 'Revert a file' cater to the basic needs of a programmer. Workspace Management: 'Open a Workspace from a file', 'Add folder to workspace', and 'Save workspace' serve the more complex project management tasks. Version Control and Collaboration: 'Open the Share tab' provides affordances for collaborative coding and versioning. VS Code has many signifiers that guide user actions. Lines are used to differentiate between different windows, aiding in navigation and focus. Different background colors signify active versus inactive areas. Symbols are used for a lot of the buttons: Search, Debugging, Extensions, Files, Settings, and many other key functionalities are represented by varying symbols. The categorization of functions under various tabs like 'Files', 'Edit', 'View', 'Go', and 'Run' illustrates effective mapping. These categories guide the user to where they can find specific functionalities. For example, anything related to file manipulation is found under the 'Files' tab, while editing tools are neatly packed under the 'Edit' tab. VS Code offers a lot of feedback such as: Tabs and buttons are highlighted when selected or hovered over, giving immediate visual cues. Cursors change to signify different actions, like turning into a hand icon when hovering over certain clickable elements. Most importantly, actionable feedback is prompt; clicking 'New Text File' immediately opens a new text file, confirming the success of the action. VS Code operates on the conceptual model of being an IDE designed to facilitate not just coding but the entire software development process. It aims to be a workspace where a programmer can code, debug, manage versions, collaborate, and even customize their working environment.