To implement the File Manager System take-home exercise, I first read and analyzed the requirements to understand the minimum viable features I needed to accomplish. Once I had a clear understanding of those requirements, I was able to come up with a priority road map, allowing me to tackle the most important feature first.

Second, I downloaded and recreated the API environment. Once I was able to complete this step I dedicated some time to get familiar with the API documentation and tried the different routes with Postman. This step was of the utmost importance since it assured me that the requests I tested in Postman were working correctly (or eventually, throwing an error).

Once I was certain that all the requests I needed to complete the minimum viable features were correct, I dedicated some time to sketching the wireframe of the data I wanted to display and how to make that data as easy and pleasant to read as possible. In this sense, I decided that I should add a Current Directory component to inform the user's current navigation. In the future, I would like to add other options so that users can choose how they want the data to be displayed (eg. as a list, displaying icon depending on the type, etc…).

Third, I started implementing the front end. For this exercise, I decided to use React. Although there are many other frameworks I chose React due to having some experience with it and its main functionalities, which allowed me to implement an app rapidly, which was one of the goals I set. Moreover, the fact that it is a common industry framework as well as having many useful libraries (eg. Bootstrap) reinforced my choice.

Later, I began working on every requirement independently, from easiest to hardest. The GET requirement was the first I tackled since it only required making an API call and then displaying the results. Next, I implemented the CREATE requirement. For this task, I also needed to get the user's input so I had to use a small form to get all the information required by the API. Then I worked on the DELETE requirement and, finally, on the UPDATE requirement.

Fourth, I implemented the Current Directory component. To achieve this functionality, I had to keep track of the directories that were visited by the user. To keep track of this data I had to use a list and update its state when a new directory was visited (which required pushing the directory to the list) or when the user went back to the previous directory (which required popping the list's last element the directory to the list). Switching among directories also required to re-render the appropriate data. This functionality was also extended to other actions, such as CREATE/DELETE/UPDATE.

Once, I completed all these functionalities I dedicated some time to styling my application. Bootstrap allowed me to apply some minimalistic styles rapidly. Additionally, it allows third-party font import to improve the APP aesthetic.

Fifth, I dedicated some time to performing manual testing to check that all the app functionalities were behaving as expected. Next, I focus on improving the flow of the app. Up to this point, the CRUD operations were working correctly, however, there were some hard-coded values that I had to adjust to make the frontend behave smoothly. To achieve this I needed to add the useEffect() hook to get data from the API as soon as the application was displayed. This task required me some time to achieve, as well, as to test it thoroughly.

Next, I focused on refactoring or cleaning code to enhance its readability and adhere to the DRY principle. I'm aware that some code can still be refactored, however, to meet the deadline I set for this exercise, I prioritize achieving all the required functionality first.

Moving forward, I’m looking forward to trying to implement the extra credit features. The first task I will address is displaying and editing file tags since I believe it would be a meaningful feature for users, and it shouldn’t be hard to implement.

Another very important feature I would include is search, but it would require significantly more effort, both in designing the user input, as it includes 5 categories, and in displaying the results.

Finally, I’m aware that I should dedicate some time to dealing with error handling whether to users providing incorrect inputs or API failure, as well as, displaying these errors to the user in such a way that they can understand what they need to do to get the desired outcome. Lastly, I would like to consider the possibility of adding an authentication process, since most of APIs require a key to be able to interact with them.