Sprint 1:
Set up the Git and GitHub account to store and track the enhancements made to the prototype.
Design the basic structure of the application and create a skeleton of the user interface.

Sprint 2:
Implement the file retrieval feature to retrieve file names in ascending order using Java.
Implement the option to add a user-specified file to the application.
Implement the option to delete a user-specified file from the application.
Implement the option to search a user-specified file from the application.
Test and validate the implemented features to ensure they meet the business requirements.

Sprint 3:
Implement the navigation option to close the current execution context and return to the main context.
Implement the option to close the application.
Refine the user interface to improve the user experience.

Flow of the application:

1. When the application is launched, the user is presented with the main menu, which includes the project and developer details.
2. The user can choose from the following options:
    a. Retrieve all file names in ascending order.
    b. Business-level operations:
    c. Add a user-specified file to the application.
    d. Delete a user-specified file from the application.
    e. Search for a user-specified file in the application.
    f. Return to the main menu.
    g. Close the application.
3. If the user selects the option to retrieve all file names, the application displays a list of all file names in ascending order.
4. For business-level operations, the user is prompted to enter the file name or provide other necessary information.
5. If the user chooses to add a file, the application adds the file to the directory and confirms the operation.
6. If the user chooses to delete a file, the application removes the file from the directory and confirms the operation.
7. If the user chooses to search for a file, the application searches for the file in the directory and displays the result if found, or informs the user that the file does not exist.
8. If the user chooses to return to the main menu, the application goes back to the main menu.
9. If the user chooses to close the application, the application exits.
10. Test and validate the implemented features to ensure they meet the business requirements.

Core concepts and algorithms used in this code:

- File Handling: The code performs various file operations such as creating, deleting, and viewing files. It uses the Java File class and related methods to interact with the file system.

- Input/Output (I/O): The code uses classes such as Scanner, FileWriter, and System.out.println for input and output operations. Scanner is used to read user input from the console, FileWriter is used to write content to a file, and System.out.println is used to display messages on the console.

- Control Flow: The code uses control flow statements such as if-else, for-each loop, while loop, and switch-case to control the program's execution based on user choices and conditions.

- Exception Handling: The code handles potential exceptions that may occur during file operations using try-catch blocks. It specifically handles the IOException that can occur when creating a new file.

- Sorting: The code sorts the list of files in ascending order before displaying them using the Arrays.sort() method from the Java Arrays class.

- Arrays: The code uses arrays to store and iterate through the list of files retrieved from the file system.

Writeup for the code:

The program allows users to perform various file operations on a set of files stored in a specific directory.

The fileHandling class serves as the entry point of the program. Upon execution, it displays the project and developer information. It then presents a menu to the user with three options: retrieving all files in ascending order, performing business-level operations, or closing the application.

If the user chooses the option to retrieve all files, the viewAllFiles method of the fileOperations class is called. This method reads the list of files from the specified directory and sorts them in ascending order. It then displays the names of the files to the user.

If the user selects the option for business-level operations, a sub-menu is presented with four choices: adding a new file, deleting a file, searching for a file, or returning to the main menu. Depending on the user's selection, the corresponding method in the fileOperations class is invoked.

The addNewFile method prompts the user to enter the name of the file and its content. It checks if a file with the same name already exists in the directory. If not, it creates a new file with the provided name and writes the content to it.

The deleteFile method asks the user for the name of the file to be deleted. It searches for the file in the directory and, if found, deletes it from the system.

The searchFile method prompts the user to enter the name of the file to search for. It then checks if the file exists in the directory and provides the corresponding message to the user.

Overall, the program provides a simple command-line interface for managing files in a directory. It allows users to view, add, delete, and search for files based on their preferences.