# PHASE ONE PROJECT

Submitted by

**VIJAY GAVANDE**

**E-mail: vijaygavande915@gmail.com**

****

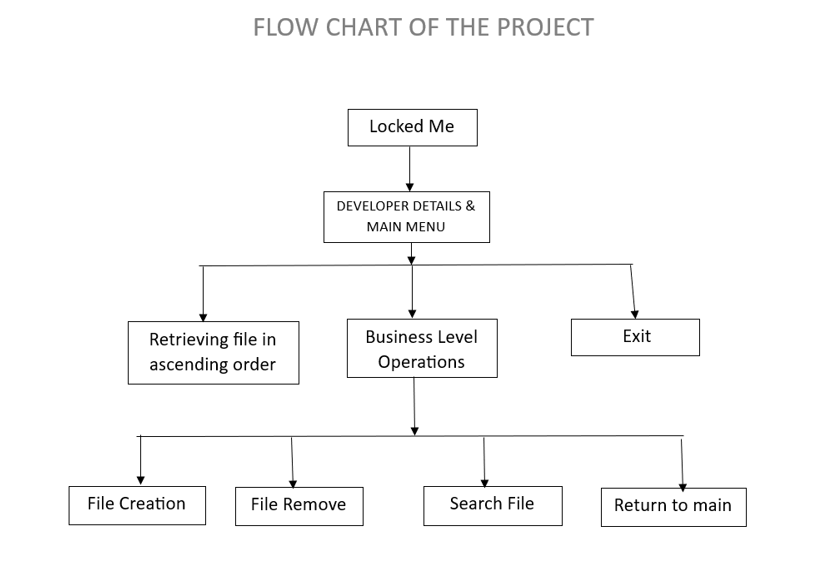
Acknowledgement

THANKFUL TO SIMPLILEARN TEAM TO BUILD MY KNOWLEDGE AND SKILLS FOR COMPLETING THE ASSESSMENT.

Thanks to all.

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Contents** | **Page No.** |
| **1**. | **Flowchart of the ASSESSMENT** | **4** |
| **2.** | **Goal of the Project** | **5** |
| **3.** | **User story** | **6** |
| **4.** | **Sprint** | **7-8** |
| **4.** | **Project objective and background of the statement** | **9** |
| **5.** | **Technology used and requirements** | **10** |
| **6.** | **Features of the project** | **11** |
| **7.** | **Function of the project** | **12** |

**Flowchart of the project**

****

Goal of the project

**THE GOAL OF PROJECT TO EXPLORE PRODUCT OF COMPANY TO EVERYONE WITH LockedMe.com.**

**The goal of the company is to deliver a high-end quality product as early as possible**.

**USER STORY**

Sprint 1: Duration: 1 week

1. Set up the development environment:

* Install Eclipse/IntelliJ IDE for Java development.
* Set up Git and create a GitHub repository for the project.

1. Plan the application architecture and flow:

* Document the flow of the application and prepare a flow chart.
* Identify the core concepts and algorithms required for the project.

1. Implement the welcome screen:

* Create a Java class to display the welcome screen.
* Include the application name and developer details.
* Display user interface details and options for interaction.
* Accept user input to select one of the listed options.

1. Implement the first option:

* Create a Java method to retrieve and display current file names in ascending order.
* Handle cases where the root directory is empty or contains files/folders.

Sprint 2: Duration: 1 week

1. Implement the second option:

* Create a Java method to add a user-specified file to the directory list.
* Handle case sensitivity of file names (consider using case-insensitive comparisons).

1. Implement the file deletion functionality:

* Create a Java method to delete a user-specified file from the directory list.
* Consider case sensitivity for accurate file deletion.
* Return appropriate messages, such as "File not found" (FNF), if necessary.

1. Implement the search functionality:

* Create a Java method to search for a user-specified file in the main directory.
* Utilize case sensitivity for accurate file retrieval.
* Display the result upon successful operation and appropriate messages for unsuccessful searches.

1. Implement navigation and application closure:

* Create a Java method to allow navigation back to the main context.
* Include an option to close the application gracefully.

1. Optimize the source code:

* Utilize appropriate exception handling to handle invalid user input.
* Utilize collections for efficient data management.
* Implement sorting techniques (e.g., using the Collections.sort() method) for improved performance.

Documentation and Finalization: Duration: 1 week

1. Document the step-by-step process:

* Create a specification document using an open-source document or Google Docs.
* Include project and developer details.
* Outline the planned sprints and the tasks achieved in each sprint.
* Provide algorithms and flowcharts of the application.
* Describe the core concepts used in the project.
* Add the GitHub repository link for verification.

1. Finalize the application and repository:

* Complete any remaining tasks, such as code review and testing.
* Push the source code to the GitHub repository.
* Ensure the repository contains proper documentation and algorithms.

1. Submit the final specification document:

* Include the specification document in a presentable format.
* Share the GitHub repository link for verification.
* Conclude by highlighting the enhancements made and defining the Unique Selling Points (USPs) of the application.

Throughout the development process, follow the Scrum framework by regularly reviewing progress, adapting the plan if needed, and delivering incrementally. Remember to focus on delivering a high-quality product by following coding best practices and adhering to the requirements specified.

Please note that this is an outline, and you will need to write the actual code for each functionality. Utilize Java concepts such as classes, methods, loops, conditional statements, file operations, and exception handling to implement the application.

If you encounter

**Project objective:**

As a Full Stack Developer, complete the features of the application by planning the development in terms of sprints and then push the source code to the GitHub repository. As this is a prototyped application, the user interaction will be via a command line.

**Background of the problem statement:**

Company Lockers Pvt. Ltd. hired you as a Full Stack Developer. They aim to digitize their products and chose LockedMe.com as their first project to start with. You’re asked to develop a prototype of the application. The prototype of the application will be then presented to the relevant stakeholders for the budget approval. Your manager has set up a meeting where you’re asked to present the following in the next 15 working days (3 weeks):

* Specification document - Product’s capabilities, appearance, and user interactions
* Number and duration of sprints required
* Setting up Git and GitHub account to store and track your enhancements of the prototype
* Java concepts being used in the project
* Data Structures where sorting and searching techniques are used.
* Generic features and three operations:
* Retrieving the file names in an ascending order
* Business-level operations:
* Option to add a user specified file to the application
* Option to delete a user specified file from the application
* Option to search a user specified file from the application
* Navigation option to close the current execution context and return to the main context
* Option to close the application

**Technology used:**

* Eclipse: An IDE to code for the application
* Java: A programming language to develop the prototype
* Git: To connect and push files from the local system to GitHub
* GitHub: To store the application code and track its versions
* Scrum: An efficient agile framework to deliver the product incrementally
* Search and Sort techniques: Data structures used for the project
* Specification document: Any open-source document or Google Docs

**Requirements:**

* The source code should be pushed to your GitHub repository. You need to document the steps and write the algorithms in it.
* The submission of your GitHub repository link is mandatory. In order to track your task, you need to share the link of the repository. You can add a section in your document.
* Document the step-by-step process starting from sprint planning to the product release.
* Application should not close, exit, or throw an exception if the user specifies an invalid input.
* You need to submit the final specification document which includes:
  + Project and developer details
  + Sprints planned and the tasks achieved in them
  + Algorithms and flowcharts of the application
  + Core concepts used in the project
  + Links to the GitHub repository to verify the project completion
  + Your conclusion on enhancing the application and defining the USPs (Unique Selling Points)

### Features of the Project

* Code to display the welcome screen. It should display:
  + Application name and the developer details
  + The details of the user interface such as options displaying the user interaction information
  + Features to accept the user input to select one of the options listed
* The first option should return the current file names in ascending order. The root directory can be either empty or contain few files or folders in it
* The second option should return the details of the user interface such as options displaying the following:
  + Add a file to the existing directory list
    - You can ignore the case sensitivity of the file names
  + Delete a user specified file from the existing directory list
    - You can add the case sensitivity on the file name in order to ensure that the right file is deleted from the directory list
    - Return a message if FNF (File not found)
  + Search a user specified file from the main directory
    - You can add the case sensitivity on the file name to retrieve the correct file
    - Display the result upon successful operation
    - Display the result upon unsuccessful operation
  + Option to navigate back to the main context
* There should be a third option to close the application

### functionality

Generic features and three operations:

* Retrieving the file names in an ascending order
* Business-level operations:
  + Option to add a user specified file to the application
  + Option to delete a user specified file from the application
  + Option to search a user specified file from the application
  + Navigation option to close the current execution context and return to the main context
* Option to close the application