TABLE OF CONTENTS

[**TASK BREAKDOWN** 3](#_Toc178791107)

[**GUTHUB COMMANDS Soft Tec (fix conflicts)** 4](#_Toc178791108)

[Week2- git version control 5](#_Toc178791109)

[**Week3 – teamwork 5%** 5](#_Toc178791110)

[**Week4 – Gantt Chart** 5](#_Toc178791111)

[**Week5 Use Case Diagram** 5](#_Toc178791112)

[**Week6 – GUI Implementation** 5](#_Toc178791113)

[**Week7 – Matplotlib & wxPython** 6](#_Toc178791114)

[**Week 8 - Pandas and Regular Expressions** 7](#_Toc178791115)

[**Week 9 – Unit Test and Coverage Test** 7](#_Toc178791116)

[**Week 10 – Unit Test and Coverage Test Load DB and Execute Queries save data from db to CSV** 8](#_Toc178791117)

**Required Submission Files:** Please check Canvas

**Tasks Distribution**

1. **GUI**
2. VISUAL DESIGN: All Pages and Its Elements

All **FIVE** features must be integrated into one single desktop app (pop-up windows are allowed).

Otherwise, 5 marks will be deducted

1. Its Screenshot of (**Object Tree** and the **Designer**)

**RUBRIC:**

Object Tree and GUI are fully clear and visible, all components properly labeled and accurate.

1. **EXECUTIVE SUMMARY (md & pdf)**

**RUBRIC: FOR (ALL 5) FEATURES**

1. Clear and concise description of the feature, comprehensive step-by-step instructions, and well-aligned, relevant screenshots for each step
2. **UNIT TESTING AND REPORT (md & pdf)**
3. Focus solely on testing all the self-defined functions related to the five required features
4. NO TESTING GUI
5. First decouple your code and separate the logic from the GUI-related code BEFORE TESTING

**RUBRIC:**

* Unit\_Testing\_Report (UTR) 1 - Test Summary
  + All functions related to the five required features and corresponding test functions are clearly listed.
* UTR 2 - Test Case Details
  + All test cases for the listed functions are correct, containing all seven components (or 5 components if the tested function has no arguments) completely and accurately.
* UTR 3 - Testing Report Summary
  + Clear and complete screenshot showing that all tests have passed with no errors or failures.

1. **COVERAGE TESTING AND REPORT**

* Similar requirement to UNIT TESTING. Focus solely on testing all the self-defined functions related to the five required features
* NO TESTING GUI
* First decouple your code and separate the logic from the GUI-related code BEFORE TESTING
* You should perform statement coverage testing and branch coverage testing. For each type, provide a description and an analysis explaining how you evaluated the coverage.

**RUBRIC:**

* Coverage\_Testing\_Report (CTR) CTR 2.1 Description
  1. Thorough and clear explanation of how test cases were designed to achieve 100% statement coverage, covering all scenarios
* CTR 2.2 Testing Results
  1. Clear and complete screenshot showing 100% statement coverage with all details visible and legible.
* CTR 3.1 Description
  1. Thorough and clear explanation of how test cases were designed to achieve 100% branch coverage, addressing all conditions and edge cases.
* CTR 3.2 Testing Results
  1. Clear and complete screenshot showing 100% branch coverage, with all details visible and legible

1. **PROJECT PLAN (md & pdf)**
2. An updated project plan document to align with your software.
3. Please ensure that any updates to the text are highlighted in red font.
4. Please provide a description in red font for any updates made to the images

**RUBRIC: Updated Project\_Plan**

* Clear, well-organized project plan with all updates accurately highlighted, fully aligned with group activities.

1. **SOFTWARE DESIGN DOCUMENT**
2. An updated software design document to align with your software.
3. Please ensure that any updates to the text are highlighted in red font.
4. Please provide a description in red font for any updates made to the images.

**RUBRIC: Updated Software\_Design\_Document**

* Clear and detailed software design with all updates accurately highlighted, fully aligned with the developed GUI.

1. **GANTT CHART**
2. An updated separate Gantt chart.
3. You should embed the screenshot (as an image) of this in your Project Plan.md
4. **git\_log.txt**
5. 10 marks will be deducted for not providing the git\_log.txt file.
6. Have to use this cmd to produce the git\_log.txt: **git log --oneline --graph --decorate --pretty=format:"%h %ad by [%an] | %s%d" --date=short > git\_log.txt**
7. **All \*.py files and their corresponding \*.txt files**
8. 10 marks will be deducted due to the missing \*.py files, which prevents the app from running.
9. Create a .txt for each .py file by coping and pasting the code for plagiarism review.
10. **Other resources**
11. Such as images/screenshots in png/bmp/jpg.

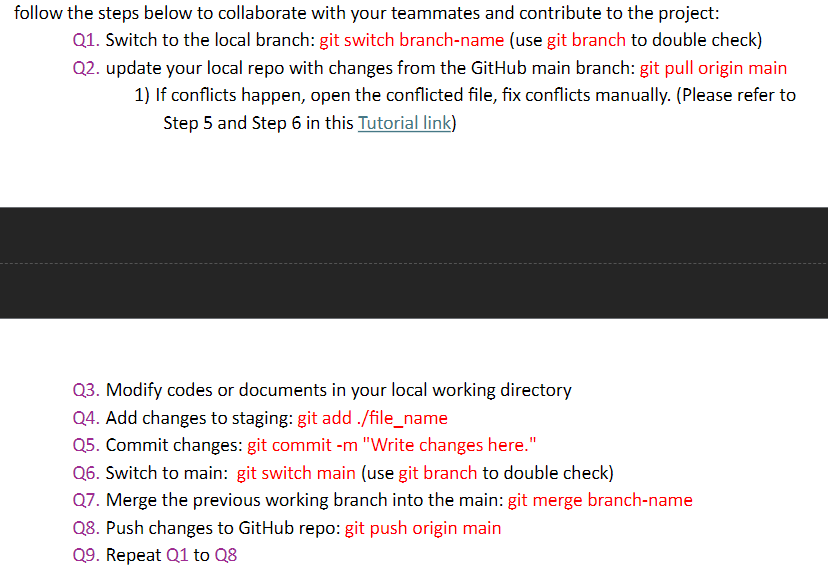
## **TASK BREAKDOWN**

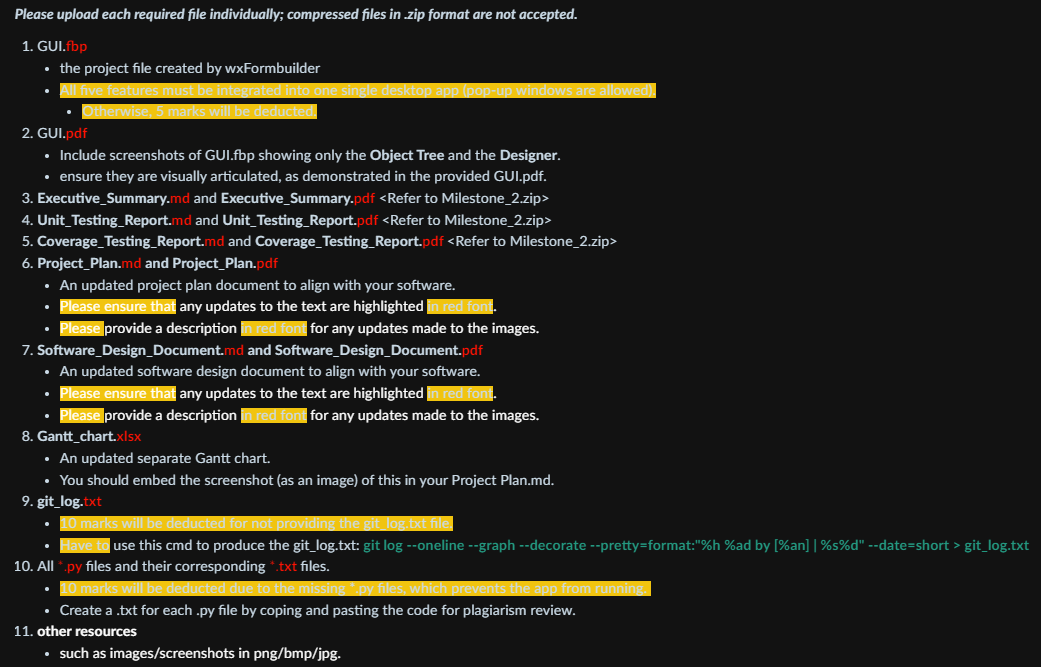
* + - 1. Name: Food Search
      2. Name: Nutrition Breakdown
      3. Coverage Testing
      4. Name: Nutrition Range Filter
      5. Name: Nutrition Level Filter
      6. Executive Summary
      7. Feature 5
      8. Unit Testing
    - PERSON1: GUI
    - PERSON2: Implement Functions
    - PERSON3: Testing

**Together**

* **All create the Gantt Chart based on the Work Breakdown Structure**
* **Git Log**
* **.Py and txt files**

## **GUTHUB COMMANDS Soft Tec (fix** [**conflicts**](https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/addressing-merge-conflicts/resolving-a-merge-conflict-using-the-command-line)**)**





## Week2- git version control

## **Week3 – teamwork 5%**

## **Week4 – Gantt Chart**

**A white background with black text

Description automatically generated**

## **Week5 Use Case Diagram**

## **Week6 – GUI Implementation**

**A screenshot of a computer

Description automatically generated**

## **Week7 – Matplotlib & wxPython**

**A screenshot of a computer

Description automatically generated**

## **Week 8 - Pandas and Regular Expressions**

**A screenshot of a computer

Description automatically generated**

## **Week 9 – Unit Test and Coverage Test**

**A screenshot of a computer program

Description automatically generated**

## **Week 10 – Unit Test and Coverage Test Load DB and Execute Queries save data from db to CSV**

**A screenshot of a computer

Description automatically generated**

**Person A: GUI Design and Integration**

1. **Visual Design**
   * Create a visual design for all five features.
   * Ensure all pages and elements are integrated into the single desktop application.
2. **Screenshots**
   * Capture and annotate screenshots of the Object Tree and the Designer.
   * Ensure components are properly labeled.
3. **Documentation**
   * Update and complete the **Software Design Document**
   * Highlight changes in red font and provide descriptions for any updated images.

**Person B: Data Analysis Functions and Unit Testing**

1. **Implementation of Functions**
   * Implement Functions 1, 2, and 3:
     + search\_food\_by\_its\_name()
     + display\_all\_nutritional\_breakdown()
     + filter\_foods\_by\_nutrient\_range()
2. **Unit Testing and Report**
   * Write unit tests for the implemented functions.
   * Prepare the **Unit Testing Report**:
     + Test Summary, Test Case Details, Testing Report Summary.
   * Ensure all tests have passed and document results.
3. **Coverage Testing and Report**
   * Conduct statement and branch coverage testing for the implemented functions.
   * Prepare the **Coverage Testing Report**:
     + Detailed descriptions and results.

**Person C: Project Management and Documentation**

1. **Project Planning**
   * Update the **Project Plan** document.
   * Highlight updates in red font and describe changes to images.
2. **Gantt Chart**
   * Create and update the Gantt chart to reflect the project timeline.
   * Embed the Gantt chart screenshot in the Project Plan.
3. **Implementation of Functions**
   * Implement Functions 4 and 5:
     + calculate\_nutrient\_level()
     + filter\_foods\_by\_nutrient\_level()
4. **Executive Summary**
   * Write the Executive Summary for all five features, ensuring descriptions and relevant screenshots are clear and concise.

**Collaboration and Coordination**

* **Regular Check-ins**: Schedule regular meetings to discuss progress, share updates, and address any challenges.
* **GitHub Collaboration**: Follow the provided Group\_Project\_Collaboration\_Workflow.pdf to manage contributions and code integration effectively.
* **Code Reviews**: Conduct peer reviews to ensure quality and functionality before final submission.