Task 1

Software requirement specifications

Business requirements (BR)

The company requires a program that summarizes the productivity of each of its departments. The “why” for this program may include being able to track the hours worked by each department to ensure that the HR/finance department(s) correctly account(s) for these hours when preparing pay slips. Additionally, the company may wish to conduct internal analytics and to determine the optimal amount of time worked versus output per department. The company may also wish to determine which employees have spent the most amount of time working in each department in order to better determine candidates for promotions.

The problem the company is trying to solve with this program is the lack of a simple procedure to determine key performance metrics such as average/total hours worked and employee with the most hours logged. If the company is quite extensive in terms of workforce, it would likely have collected a large amount of data on the different employees and departments. Therefore, it may become laborious to sift through the numerous columns and rows of data to determine the various key performance metrics. A program that can take an extensive amount of data and output a summary of the key performance metrics per department would solve this problem.

User requirements (UR)

The end-user should be able to provide a CSV file containing employee names, departments, and daily hours worked for a week to the program. They should then be able to run the program and receive a text file that displays a simple breakdown by department showing average hours worked, total hours worked, and employee with most hours.

Functional requirements (FR)

For the program to work correctly, it will need to read from a CSV file that is placed within the resources folder. The program will need to treat the first row of data as containing the headings for the columns; it should not include this row of data in the calculations that it will perform. Subsequently, the program will need to filter the data by department. It will then need to perform simple calculations, such as summing up the total amount of hours worked per department and the average amount of hours worked by employees, then return these data in a simple text file. It will also need to calculate the most hours worked by a single employee and return the name of that employee in the aforementioned text file.

A run-down of the functional requirements are listed below:

FR-001: The program should be able to read from a .csv file.
FR-002: The program should differentiate the column headings in the .csv file from the data.
FR-003: The program should be able to filter data based on department.
FR-004: The program should be able to sum the total hours worked per department.
FR-005: The program should be able to sum the average hours worked per department.
FR-006: The program should be able to sum the total hours worked per department.
FR-007: The program should be able to calculate the most hours worked by an employee in a given department.
FR-008: The program should output the summarized metrics of interest to a .txt file.

Non-functional requirements (NFR)

The program should be able to quickly output a summary of the key performance metrics for each department in a simple text file. It should be able to do this for CSV files containing large amounts of data entries. Therefore, the program needs to demonstrate good scalability in terms of data input amount. As a starting metric, the program will need to be able to process a .csv file containing 25 data entries. As the organization grows, it will need to handle larger .csv files.
In terms of security, the program should only be accessible to certain individuals within an organization who would need to make use of the program to fulfil their duties e.g., HR/finance staff who are overseeing pay slips. Moreover, the data process handling data will need to be secure so that data remnants are not collected in locations other than the direct output file path.
The program should also be highly reliable; it should only output clear, easy-to-interpret breakdowns per department in a simple text file. Therefore, it will need to be robustly tested for bugs prior to release, and a complete change management procedure will need to be conducted.

The NFRs are summarized below:

NFR-001: The program needs to produce the .txt file quickly.
NFR-002: The program needs to be able to handle larger .csv files for when the organization expands.
NFR-003: The program will need to be accessible by only selected personnel and workstations at the organization.
NFR-005: The data process containing employee data will need to be secure.
NFR-006: Prior to release, a change management procedure will need to be conducted before incorporating it within standard business operations.

Program design language

The program code should follow the logic outlined below:

Ask the user to place a CSV file containing the necessary fields within the resources folder.

```
If the file ends with anything other than .csv
    Request that the user submit a .csv file and not any other file type
else
    Take the CSV file and separate the values by commas
Create a function that transforms each data entry into an array, except for the first index (containing the titles).
For an array of a given department type
    Create a variable named totalhours and set it to an initial value of 0.
    Create a variable named totalemployees and set it to an initial value of 0.
    Create an array named topemployee
    Sum the total hours worked and add them to the totalhours variable and to the topemployee array.
    Increment the totalemployees variable by 1.
    Create a variable named average hours and set it to the result of dividing totalhours by totalemployees.
```

Return a text file that outputs a breakdown of the data by department name, including total hours worked, average hours worked by the employees, and the employee with the most hours worked.