Team 30 – Final Product

OS Operating System: *Payroll System*

The aim of our project is to create a batch operating system that fulfils the role of a customized Payroll system. It's designed to run per user/call and perform multiple jobs that in the end accumulate to provide the financial statement due to each employee along with the accompanied transactions and updates the log file with any changes caused by the transactions. It has 2 main resources:- memory and input file.

- **OS Type:** Batch Processing

Our system requires real time input from the user only when he requests to pay an installment of his/her existent loan.

- **OS Components:**
    a. **Input**
        i. Employees record file from database (log text file)
    b. **Code**
        i. Module to allow loan installment payment (updates log file)
        ii. Module to add working hours (updates log file)
        iii. Module to print out financial slips
    c. **Output**
        - In case of financial statement request, 2-line report in the console
        - Updated version of log file
        - Throughout its run time, the system will output status update to help users track the function of the system and offer an insight on the operations performed and in which order

- **OS Methodology:**

1) *Module to pay loan installment*

This process/class/module allows employee to pay off part or all of his loan. All employee has to do is enter the amount they want to pay and the system checks if its sufficient or not (must be within the loan due or else error message will appear) and if so, it 'hypothetically' withdraws the amount from his/her bank account and updates the log file to show the updated amount that employee now owes.

2) *Module to log in working hours*

   This process/class/module allows employee to log in the amount of hours worked on that particular day. It 'hypothetically' retrieves that data from the computer itself (how many hours he/she been logged on – in reality, done using a randomly generated number between 2 and 8) and updates the log file to show the updated number of hours that employee has worked this month.

3) *Module to print out financial statement*

   This process/class/module allows employee to retrieve the appropriate information (employee name, amount to be paid (loan), hours worked, and cumulative salary so far). It requires not input from the system or the user, it fetches the data from the log file.

- **First Code Implementation:**

Our code is divided into 14 classes: -

1. Processes --> this package contains the Process class that initiates any program to the NEW state, ready to be activated by the instantiation of a specific program/application. The process class is abstract with attributes (process ID , memory Address, state, ….etc) and contains an abstract run() method that each application that will implement the class must have it customized to the specified use of the app/process. It also contains an Enum for the various process states (ready, new, …etc)

   Customized Processes are: LoanPayment, LogWorkingHours, FinancialStatement

2. Memory --> this package contains a memory class that's responsible for holding the array simulating the RAM memory of a real-life OS. We have made it of a fixed size to ensure that it always for resource allocation and since real life memory is not infinite. This class also manages the memory and informs CPU of available space needed to store new processes or be used by processes that are already being executed. There is also a customized memory segment class that we created to be able to store all info related to a single memory segment in one structure (ex. the process it contains, its size, whether it's occupied or not, …etc).
3. OS --> This contains 2 classes that are the scheduler and acts as the main class of the whole program as well as the Console which is our main output resource. The Console is created to be able to allow all processes to access it with the permission of a semaphore and be able to output all details about the inner execution of the system. The Scheduler is mainly where the whole OS is. Instances of all other classes are made in it and it

handles the dynamic of the problem: deciding which process to run when and how the whole system operates.

4. GUI --> This package contains all classes responsible for the GUI that is displayed in the last stage of the execution of the code. It portraits the inside of the system: the ready queue, the CPU utilization and the progress of each process at runtime.

Notes:

- Since this is a payroll system, we will not be implementing any priority schemes or preemption since all jobs have to be fully executed with no interruptions.