WashOS

# Project idea:

One of the most crucial electronic devices that is an impecable part of any household nowadays is basically a washing machine. We became so reliant on it, and companies started creating innovative designs and adding new features to washing machines everyday to provide customers with a comfortable, efficient, and luxorious experience while washing clothes. For that reason we decided to create WashOS. WashOS is a fully-functional system that simulates the performance of an ideal washing machine with both basic and additional featues. It will give its’ users a complete interative experience of using a washing machine.

# Operating System Type:

Interactive Operating System

# Functionalities and Featues:

1. A Graphical representation of the washing machine is displayed to User
2. User first starts the WashOS
3. User chooses the required wash program
4. User has the ability to create a custom program to be saved and used later on
5. User can adjust wash settings, such as spins, temperatue, wash time, spin speed,intensity of both rinsing, washing, and spinning,
6. User starts the wash and a timer is displayed , WashOS starts operation.
7. User has the ability to interrupt, cancel or pause the wash in operation.
8. Errors would occur and program will not run in cases where the door is not closed, or there is a water leakage in the machine, or no water supply.

\*Functionalities in Yellow will be implemented in Milestone 2

\*Functionalities in Blue will be implemented in Milestone 3

\*Functionalities in Grey will be implemented in Milestone 4

# Work Methodology:

1. Create the required classes (User interface, engine, error handling) (Java programming language)
2. Add the functionalities as functions to the engine class
3. Handle different error scenarios.
4. Test the systems with the functionalities implemented.
5. Create the graphical user interface, and connect it to the engine created
6. Test the system, and start solving errors or adding features.

The Team members will be assigned to different tasks, and the project will be shared on GITHUB where all members will using “git” to push/pull the latest changes to the sytem.

Before starting any implementation, an intensive research on the different features, and implementations of similar systems will be made, to build an accurate picute of how the system should correctly be implemented.

# Description of Implemented Classes:

1. We have three main packages in our operating system: CPU, OS, and Apps
2. CPU : Contains
   1. Main cpu which handles the final execution of the processes( handle to processor), fetching apps from hard disk, and dispatching processes to/from the ram
   2. Ram Memory containing all processes in Ready State
   3. Hard Disk containing all apps used in operating system
3. OS: Contains
   1. Main OS:
      1. containing two main queues (Ready/Blocked) storing PCB of processes in ready and blocked states
      2. According to the program chosen by the user, the MainOS dispatches specific apps responsible for the program to the ready queue from the hard disk (dispatchToReady)
      3. All PCBs in ready queue is ordered according to priority in descending order
      4. A scheduler is always running in a separate thread to determine which processes will be chosen to be executed from the ready queue and calls the cpu for execution.
   2. PCB : containing all relevant information related to processes
   3. Process : represents a single process entity containing PCB, and is abstract, to allow all apps to extend it and implement the run() method differently according to each apps’ functionality.
   4. States: All possible states
4. Apps:
   1. Contains all apps that are representation of the control of various hardware devices and tools.

# Process Flow:

Program is chosen => OS gets all the apps needed for this program into the ready queue by calling cpu = >Cpu accesses the hard disk and return to the OS the required processes/Cpu adds the required processes to ram for faster access afterwards =>OS sets the priorities of these apps and sorts them in descending order => Scheduler chooses which processes will run from the ready queue taking the priorities into consideration and calls the cpu for execution => CPU execute the process and then removes it from the ram.