CSCI 3100 Software Engineering

Assignment 2

Jonathan Lam Pui Hong (1155093597)

Question 1

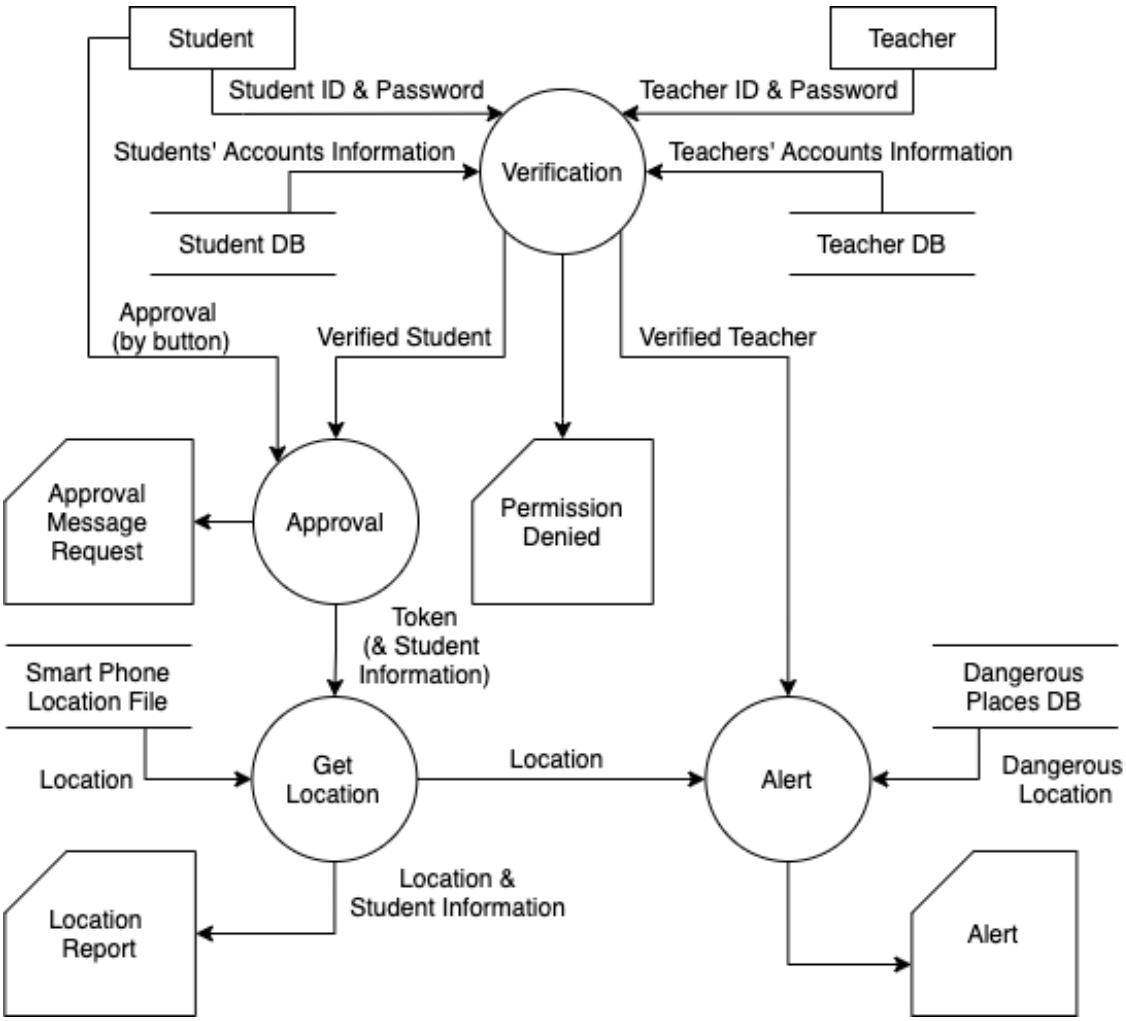

Fig.1 A DFD for Safe-Bernard. Left part mostly concerns that of "Student[s]" while the right of "Teacher[s]". Verification/Verified is interchangeable with log-in/logged-in. "Verification", albeit presented as one process, only requires either "Student" and "student DB", or "Teacher" and "Teacher DB", which in turn heads to either "Approval" or "Alert" respectively. Since it is not the concern of DFD, it is merged into one process. "Alert" is only fired through the app in which the teacher has to log in and verified.

Question 2

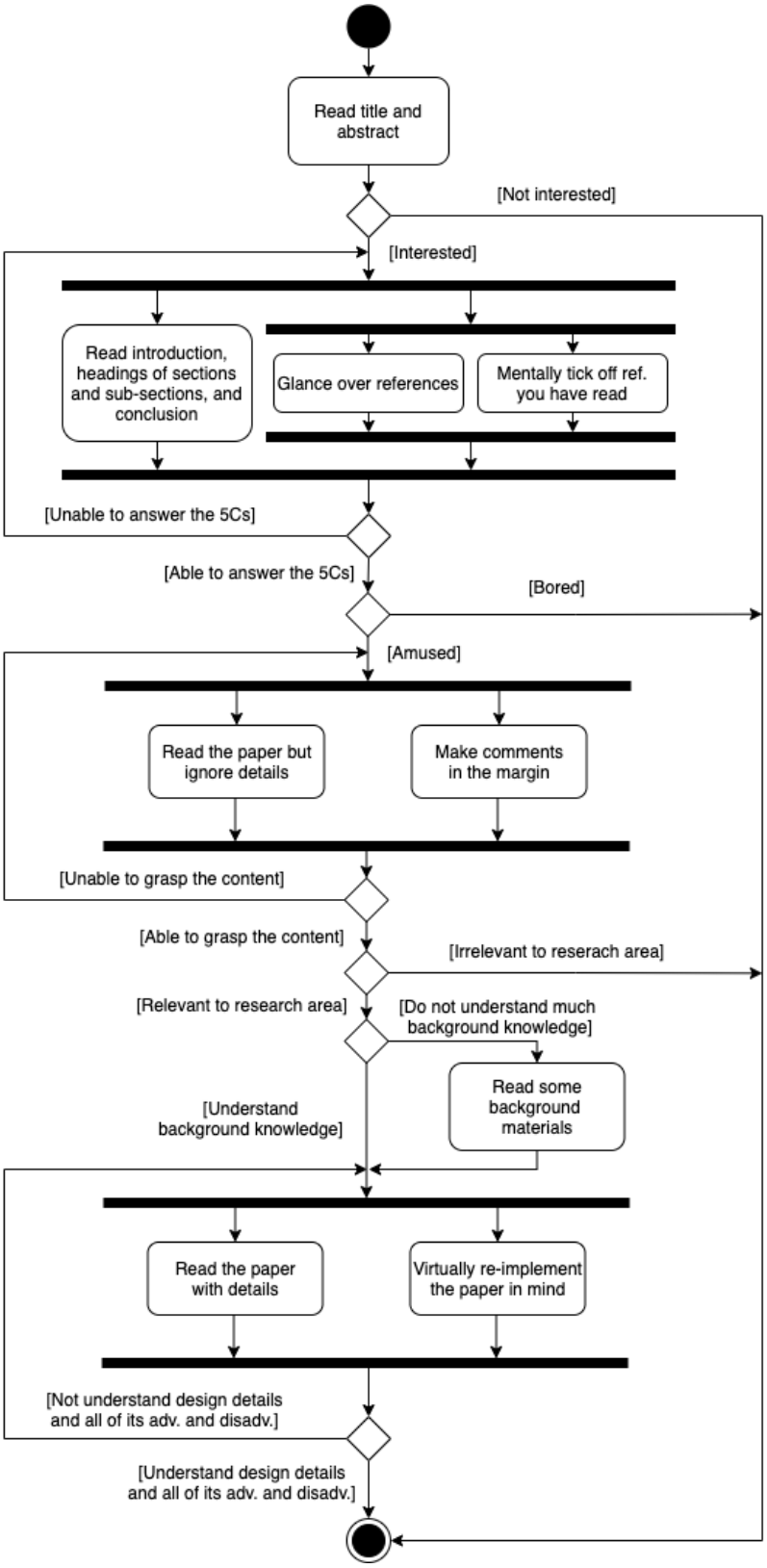

Fig.2 An UML activity diagram for the process of paper reading. Upon each "pass" that does not fulfil the objective of that stage, the process is assumed to be repeated until the objective is fulfilled.

Question 3

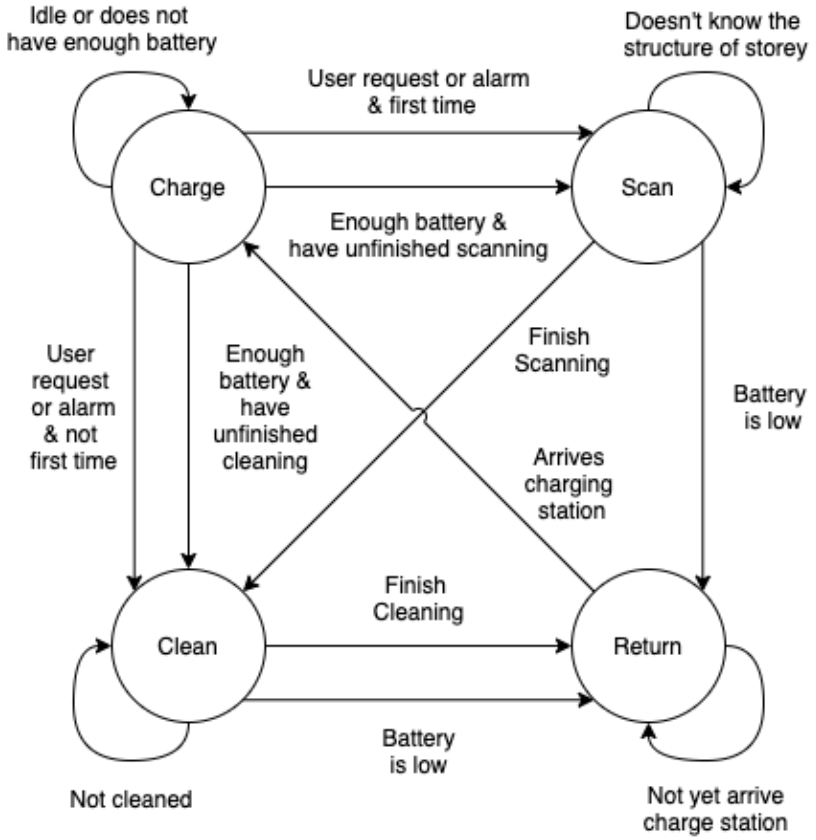

Fig.3 An FSM denoting the cleaning robot that Mr. Bernard wants. The state where the robot is idle (waiting for invocation) and does not need charging (i.e., full battery) is omitted and considered as charging. An "or" operation in the loop back to itself in "Charge" means it will either charge to full when idle or charge to enough when have unfinished work.

b. Advantages: The robot is either staying at base or moving, where the latter is split into working (scanning and cleaning) and returning. These kinds of simple mechanism with a few finite states are where FSM shines as it provides a fairly simple and clear way to define the actions, such abstraction needed to better facilitate the implementation of the system.

Disadvantages: FSM, a diagram mainly to show when to change states, may oversimplify the problem since states are fixed and rigid. In actual running of the robot, it may encounter situations that the current state does not anticipate thus cannot move from state to state. For example, when working, the robot may stuck on edges or places; the failure of cleaning the scanned place of its entirety – or even to move – means it could not get to "Return" and hence "Charge" even if the battery is low and slowly dies out. There are also state oscillation problem if terms like "enough" and "low" are not so clearly defined. If "enough" and "low" means, say, 25% and 10%, it may go back and forth between charging and working with no optimization method provided in a simple FSM.

Question 4

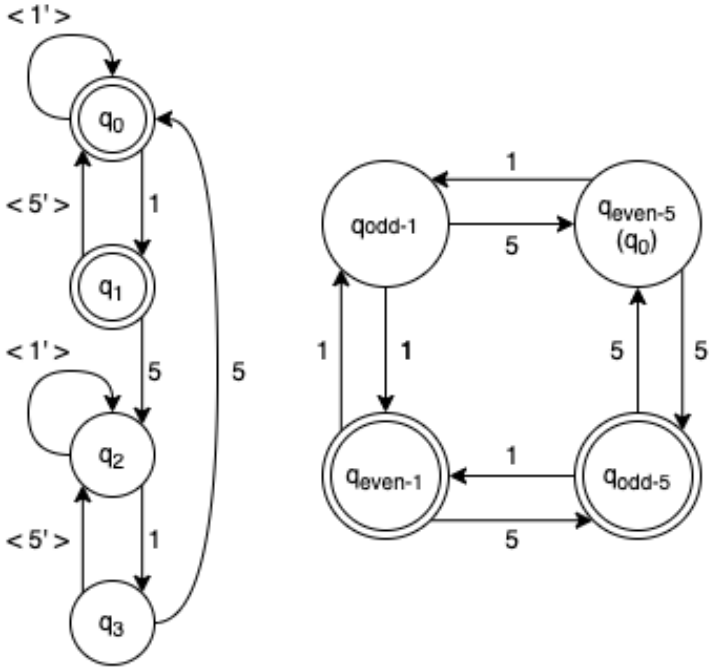

Fig.4 FSMs denoting problem in part a (left) and part b (right), both with q_0 as initial state and double-circle as final states. In part a, <1'> means any alphabet that is not 1, i.e., {2, 3, 4, 5, 6, 7}; <5'> means any alphabet that is not 5, i.e., {1, 2, 3, 4, 6, 7}.

a. Parity of '15's is even in q_0 and q_1, and odd in q_2 and q_3. Any successive input of 1 and 5 moves the states from q_0 via q_1 to q_2, or from q_2 via q_3 to q_0. This acts as a counter of the number of '15's. If the input after 1 (i.e., in state q_1 and q_3) is interrupted by any inputs other than 5, it will loop back to the previous state (either q_0 or q_2), awaiting for another 1 to be inputted.

b. States are labelled with q followed by <parity of number of digits> dash <ending digit>, for example, $q_{even\text{-}1}$ means there is an even number of digits in the sequence that ends with 1. Any inputs will shift the state between odd and even states, heading to respective state depending what the input is, until the condition is right.