REPORT:

The app overall worked and effectively transitioned between the defined states. Each listen state checked if the utterance of the user was in the grammar. If it was, it was stored in the relevant object variable until the next "click" event (with the exception of the agree/disagree objects, whose values were reset before transitioning to the *GetAnswer2* state, s. "exit: assign({ agree: null, disagree: null })"). If the utterance value was not in the grammar, the speak state question was repeated until an utterance was recognized. To reduce the complexity of the states' transition and the number of states, the speak state utterances "You just said […] and it is : it is not in the grammar" and "I can't hear you" of the original "dm.ts" code were removed. The *LastCheck* state included all object values that were recognized, to check with the user if they were recognized correctly, and if the answer (in *GetAnswer2)* was negative (context.disagree!=null), it simply returned to the first prompt state (="Who are you meeting with?") and repeated all the transitions from there. The app had some difficulty recognizing the values of the grammar: if the utterance is not pronounced with the American accent or there is noise in the background, the system fails to recognize it, therefore sometimes the user has to repeat their utterances multiple times.

The app in that form was very simplistic and constitutes a dialogue system in its most basic\initial form, having thus a lot of restrictions and not being very handy in real life situations. For example, if the user's utterance consists of more words than simply the grammar values, the utterance is not recognized. A potential improvement for this would be by modifying the Get[Object] functions, as well as the guards in the RECOGNISED event of the listening states. Instead, I just added more strings for times and days for the recognition to be a bit more flexible [the manual addition was time consuming and not very efficient; using regular expressions would have made the code more concise].

Another limitation was that if the answer in *GetAnswer2* was negative, the system was returning to the *Who* state, so the user had to answer the questions all over again from the beginning. For this reason, I implemented the *WhatsWrong* state after *GetAnswer2*, which checks explicitly the recognized utterances that were gathered in the previous states, so that only the question about the falsely recognized information should be repeated. I added the listening/accepting *GetChange* state, along with defining an object *change* with the values "person", "day" and "time" in the grammar. One restriction that remains is that only one value change can be performed before re-entering the *LastCheck* state (i.e., the user cannot change the values of e.g. *person* and *time* at once, but has to change one of them, then give a negative answer to the *GetAnswer2* state, and then change the other value).

Two other improvements that I thought of but didn't implement were: 1) Having only one *GetAnswer* state for all the speak states that require a yes/no answer, by adding some extra guards to make the proper transitions [after adding the *WhatsWrong* and *GetChange* states, having only one *GetAnswer* state would be too complicated with regards to properly managing the transitions to the next state].

2) Providing feedback to the user for their answers by relevant speak states (e.g. "I can't hear you" if no utterance is recognized, "Meeting with [person]/ on [day]/ at [time]: Check!") [would be more natural and fun in terms of conversation, but doesn't add much practical/functional use].