Smart Prompt

Blake Patterson
Temple University
College of Science & Technology
Philadelphia, USA
tui02345@temple.edu

***Abstract*—Smart Prompt is an Alexa Skill aimed at helping those with memory issues keep better track of their daily/weekly tasks that they may otherwise forget to complete. It also keeps a record of those tasks and their completion times to better inform the individual and/or caregivers of how often the individual completes their tasks as well as how long it takes them.**

I. Introduction

A very large percentage of people over the age of 65 have age related memory ailments such as dementia. Not only are there no effective treatments for these impairments, but the number is not going down either. Due to this, many older adults are finding themselves unable to complete their basic tasks such as exercising, taking medicine, walking the dog, etc. However, it is often not the case that they are physically and/or mentally unable to complete these tasks, it is simply a matter of forgetting about them. Thus, all these individuals need is someone or something to remind them to complete these tasks, and having someone there to remind them all day long is often not realistic, so it is only natural to look towards technology as a solution.

One major problem with this, however, is that the vast majority of the people with memory impairments are not tech savvy. Thus, making some fancy mobile app with a bunch of bells & whistles is more of a hinderance for them than a solution due to the learning curve. There is, however, one piece of technology that does not require a learning curve while still providing the functionality we need, and that would be Amazon Alexa. Alexa not only has a reminder framework that would allow us to build a solution to this problem, but can be interacted with solely through voice commands allowing for a non-tech savvy person to have a completely natural conversation with it. Not only that, but it is fairly common for any given household in America to have an Alexa enabled device of some sort in their home, allowing for easier adoption.

With this in mind, we build an Amazon Alexa Skill called Smart Prompt to solve the problem many older Americans are facing by reminding users to complete their daily tasks. In addition, it has two unique functionalities to set it apart from standard reminders: personalization and the recording of tasks. Personalization (i.e., referring to the user by name & playing custom audio upon completing a task) makes Smart Prompt more enticing to use as well as encourages the user to complete their tasks. Recording of tasks allows users & caregivers of users to be fully aware of whether or not the user completed their important tasks (i.e., whether or not the user took their morning medicine).

For the remainder of this paper we will examine existing solutions to this problem, examine our solution & how it works, and finally compare our solution to what already exists.

II. Related Work

related work

III. Design

Smart Prompt, at its core, serves a fairly straightforward purpose: it allows users to create and manage weekly reminders. To give more detail, a user is able to create a reminder to go off on a certain day of the week at a certain time, and that reminder will go off at that day & time each week (until the user removes that reminder). Then, when a reminder goes off, the user is notified of the message that they asked the reminder to give, and they can then tell Smart Prompt once they have completed their task in order to mark it as complete. Finally Smart Prompt will send the user's smart phone a push notification alongside each reminder, so that they can see their reminders even if they are not at home.

Here is a list of features in Smart Prompt (More details about the features not yet implemented will be provided in the Discussion section).

Feature	Status
Grant Permissions via Verbal Approval	Complete
Create Reminder	Complete
Refer to User by Name	Complete
Store User Data in Database	Complete
Store Reminder Data in Database	Complete
Recording a Reminder as Complete	Complete
Playing an Audio File Upon Task Completion	Complete
Managing Reminders (i.e., CRUD)	Complete
Manually Snooze Reminders	Complete
Auto Snooze Reminders	In-Progress

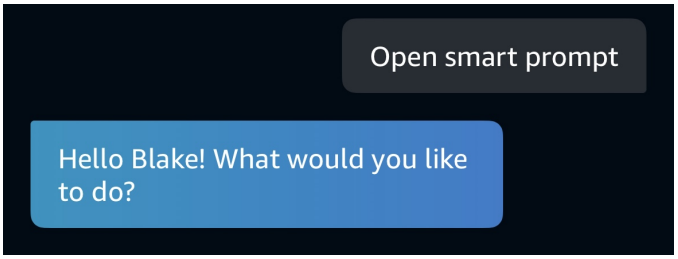
At a technical level, Smart Prompt consists of two parts: Amazon Alexa as a front end & Google Firebase Realtime Database as a backend. Let us examine those two parts and how they contribute to this purpose.

A. *Frontend*

The frontend, as explained before, is built using Amazon Alexa. In its simplest form an Alexa can be explained as a set of intents, i.e., actions for the users to perform with the skill, and the skill can do any number of things when an intent fires. There are also some outliers such as event & request handlers, which can be viewed in a very similar light, with the major difference being that they are not usually fired from the user saying something. There are many more details than that, but let us go through the main intents (as well as some request & event handlers) of Smart Prompt and what they do.

1) Launch Request: This is simply what occurs when the skill launches, i.e., immediately after the user says, "Alexa, open Smart prompt". There is not much to speak of here, although this is where the necessary permissions are requested and granted the first time a user uses Smart Prompt. More specifically, Smart Prompt requires explicit permission from the user in order to create reminders for them as well as use their first name. Once the user has granted these permissions, the skill will simply ask them what they would like to do. Once the permissions are granted, every proceeding launch request will skip straight to asking the user what they would like to do.

Here is what the launch request looks like:

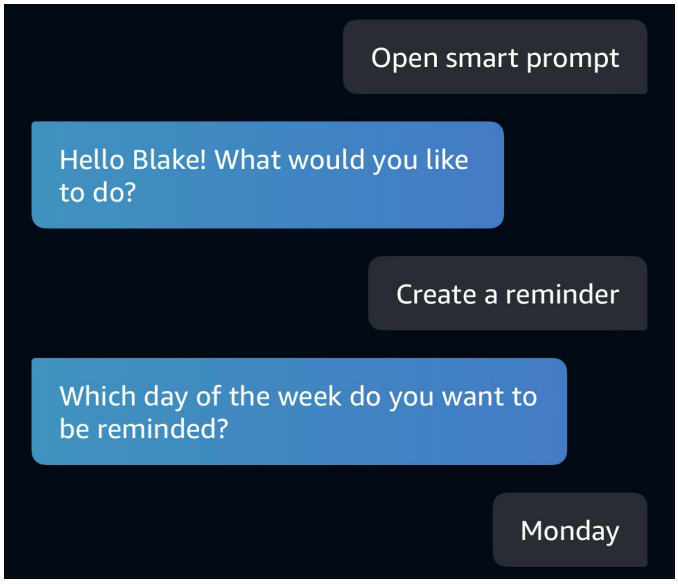


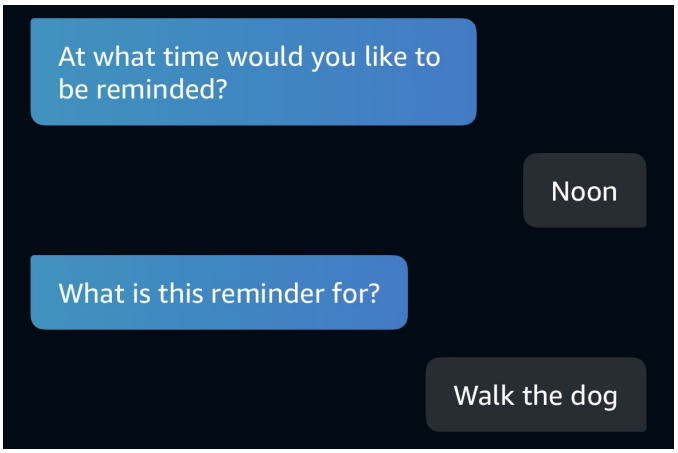
2) Create Reminder Intent: The Create Reminder Intent, as one would assume, allows the user to create a reminder. Ideally reminders would be set up early on (possibly even by a caregiver or an expert prior to the user ever interacting with the skill), so a user should not have to use this intent often, although it is there for whenever they need to quickly set up a new reminder.

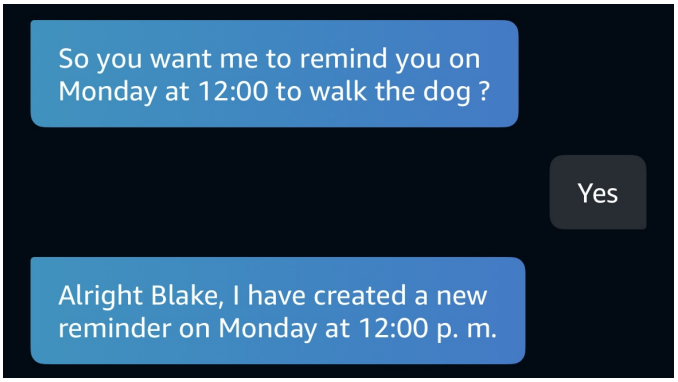
The Create Reminder Intent takes in three required slots, i.e., small bits of information provided by the user for processing. The three slots are the day of the week the reminder should take place on (i.e., Sunday, Monday), the time of day the reminder should go off (i.e., 9am, noon, 3pm), and finally the message the reminder should give (i.e., "walk the dog", "take medicine"). The intent can collect these three pieces of information in multiple different ways. The user could start by providing no information other than that they want to create a reminder, and the skill will prompt them for each slot individually. They can also provide the day, the time, or the day and time up front, in order to skip the tedious process of Alexa asking them for that. Finally, they can also just provide the reminder message. Whatever slots they do not provide up front will simply be collected via additional prompts given by the skill. As one final check once the slots are gathered, the skill will read them back to the user and ask if they are correct, giving the user one final chance to cancel in case Alexa interpreted something incorrectly. This varied way of creating a reminder allows for a much more natural feeling interaction with the skill.

After collecting all of the slots and confirming they are correct with the user, the skill will go ahead and do two main things: create the reminder using the Alexa Reminders API & register the skill in the database. The database will be discussed further in the backend portion of this section, but there is one important thing to note about how the reminders are created with the API. When the reminder is created, it is actually creating multiple reminders at once, one for the initial reminder and one for each snooze that should take place if the user does not record it as complete. This is done due to technical limitations of not being able to update the one reminder's time automatically at certain points of the project's execution cycle. Everything else about the creation of the reminder(s) is fairly straight forward, all of the information collected along with some other boiler plate information is send through an API request to Alexa's system and the reminder is created. If for whatever reason this fails at any point (considering it is an API request as well as a database operation), the user is notified of the failed creation and asked to try again.

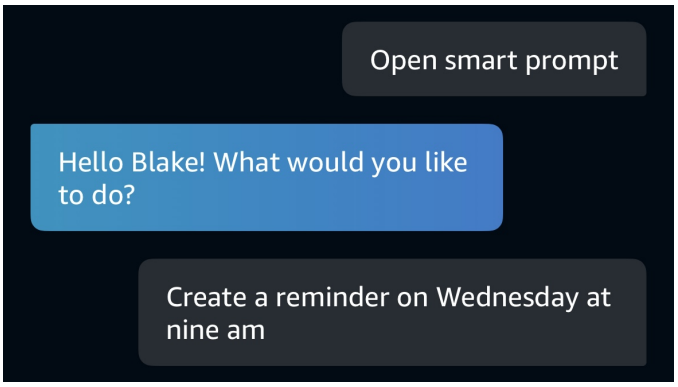
Here is one example of how to create a reminder:

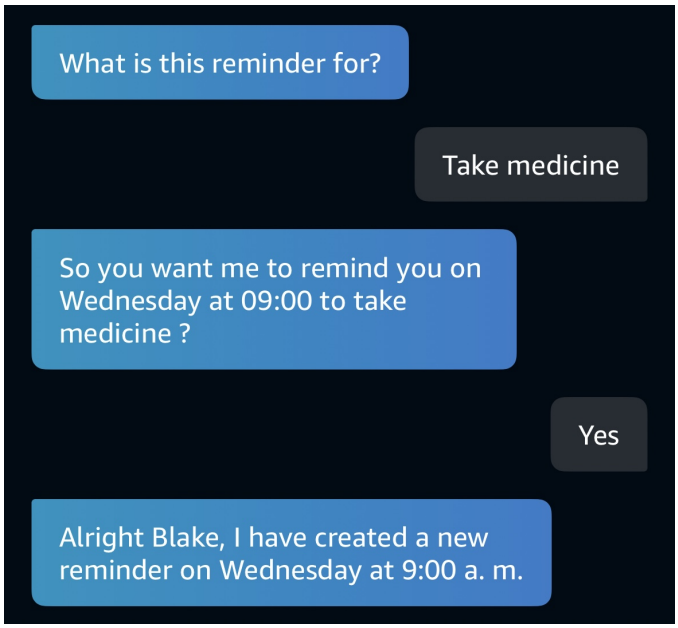






Here is another example where the user provides more detail from the very beginning:





3) Completed Reminder Intent: The Completed Reminder Intent is slightly less straight forward, although still what it sounds like: it allows the user to record a reminder as complete. There is no required information here, the user simply needs to say something along the lines of ”I completed my task”. Once the user says this and the intent fires, the skill performs some database operations. First it checks what reminder is pending (i.e., the last reminder that went off). Then it performs some read & write operations in order to record the task as complete as well as the time of completion. Finally, it clears the pending reminder and checks if the reminder completed has an audio file saved. If it does, it plays the audio file. Otherwise it simply notifies the user that the reminder has been recorded as complete.

4) Reminder Started Event: This event handler, as it sounds, occurs whenever a reminder starts, i.e., right as it goes off. Obviously there is no user interaction here (the actual noise of the reminder going off as well as the user asking for it to stop is handled solely by Alexa’s built in framework), but there are some database operations that occur here. First off, the skill uses the reminder ID to search the database for the reminder data. A new ”iteration” of the reminder is stored for each week that it occurs, so if it is the first time that reminder is going off that week, a new iteration will be created and the start time of the reminder will be recorded. Otherwise, the current iteration will simply be found and the number of snoozes for that iteration will be incremented. Finally, this reminder will be entered as the previous reminder in the database, for the purposes required by the Completed Reminder Intent.

B. Backend

The Firebase Realtime Database, being a NoSQL database, is structured more like a large JSON object than anything else. At the most global level there is a users list. This list is indexed by the user’s Alexa ID, such that when the skill needs to access

the database it simply uses the user id that it has stored itself. Within each user object then exists a reminders list. Similarly to the users array, the reminders list is indexed using the Alexa ID of each reminder, again for the same reason as the users ares stored using their Alexa ID. Additionally, there exists a previous reminder object stored at the user level in order for the skill to know the pending reminder for each individual user. Moving on, each reminder object contains numerous fields. such as the reminder day, time, & message. Each reminder also includes an optional audio file field, which contains the name of an audio file stored in an Amazon S3 bucket (if this field is included, the audio file named will be played when a reminder is recorded as complete). The last field in each reminder is the iterations list, with each iteration representing the reminder data for any given week. Each iteration stores the start time (the exact time the reminder first went off), the completion time (if it was completed), whether not it was completed, and the number of times it went off (i.e., number of snoozes that went off).

IV. Evaluation

In this section we will examine other projects and commercial solutions related to Smart Prompt, and see how Smart Prompt compares to them. Surprisingly, there are no notable & commercially available solutions that are directly comparable to Smart Prompt, so we will focus on existing Amazon reminders, other general reminder/task management solutions, and other dementia related health solutions (not specifically task management).

A. Alexa Reminders

What becomes immediately evident upon researching the topic of Alexa reminders, and even more so specifically for users with memory ailments, is that the reminder system built into every Alexa device remains the king of this category. It is, admittedly, a well fleshed out, useful system that would fulfill most peoples wishes when making a reminder. However, it does not provide everything we are looking for in our reminder system.

Let us examine the main features more closely. Alexa allows users to create reminders for any sort of time on any sort of interval. They can create daily, weekly, monthly, or any sort of recurring reminder. They can also create relative reminders, i.e., to be reminded of something an hour from now. They can even create location based reminders, such that they are reminded of something upon arriving at a certain location. Additionally, once the reminder is created, they can do almost anything with it. They can change the time, the message, the day, the details of the recurrence, and pretty much every other detail you can think of (all of this can be edited in the app or via voice controls). Being reminded of what reminders you may have is straight forward as well. The user can ask, "what reminders do I have today?", or, "when is my next reminder?". When the reminder goes off, the user can even ask for it to be snoozed for any amount of time, and if it is a recurring reminder, it will only alter the one instance and not the whole series. Essentially, users can create any type of reminder, and viewing, updating, and deleting those reminders is straight forward and thoroughly implemented.

For the vast majority of users, this is more than what they need or even want. However, for our use case, this is lacking in some departments. First of all, considering our users will have memory ailments, it would not be easy for them to remember if they completed a reminder or not. And when they did, it would still be hard for them to remember exactly when (in case they had snoozed it or took a long time to complete it). These details are often important for our potential user base. For example, one reminder might be related to taking a very important medication, and it would be helpful for the user & potential caregiver to be able to view whether or not the medication was taken. Another example would be if the user is undergoing physical therapy. Considering it might be hard for the user to remember when they performed their necessary exercises, it would be beneficial for the user's doctor to be able to concretely view a record of completion. For these reasons, a tracking system of some sort is a basic requirement for our solution, and Alexa is entirely lacking in that department. Alexa merely sets of a reminder at a given time and forgets about it from there on out. It does not care in the slightest about what happens after the reminder goes off or what the user does in response to the reminder.

Another, somewhat simpler issue is that Alexa completely lacks personalization. For a younger person that is more tech savvy, they are probably happy with a more blunt, to the point Alexa conversation. However, our target user base is much older and did not grow up with technology like Alexa. Thus, to make using our solution feel less like talking to a robot and more like a personal experience, it is a requirement that it contains personalization. Not only would personalization make the transition into using the solution as well as the general experience better, but it would provide more incentive into using the solution and completing your reminders. For example, if the solution were able to refer to the user by name and play a custom clip of a family member greeting them upon completing a reminder, then the user would most definitely be more likely to complete a reminder and continue to use our solution than if they simply received nothing but monotone instructions from a device.

In short, although Alexa has a well built reminder system already, it is lacking some core requirements we are looking for in our solution.

B. Other Reminder Solutions

Firstly, let us examine some third party custom Alexa skills related to reminders & task management. Commercially, there is not much. When you search for reminders under Alexa Skills or go to the Productivity section, most of what comes up are very niche, specific, simple skills. For example, the skill that came up the most was "Stand Up Reminder", a skill built solely to remind you to stand up every 45 minutes. Beyond that the most notable skills involve reminding users to drink water, when their friends' birthdays are, and when to

do yoga. Notably, however, there are a small number of skills attempting to provide a general, non-niche reminder system. Although, none of these general reminder skills expand upon the base functionality of Alexa reminders in any way. In fact, no skills came up that even matched Alexa's base reminder functionality, they were all even simpler with fewer features.

Moving away from Alexa for a little, there are countless reminder & task management systems out there.

C. Other Memory Ailment Related Solutions

put anything related to solving a problem for memory ailments here

D. Extras

put noteworthy things here, i.e., the very similar project that never came to be and the articles about using base Alexa for dementia

V. Discussion

Discuss what couldn't be finished, difficulties, next steps, etc. (specifically go into detail about how to implement the auto snoozing)

VI. Conclusion

Conclusion

VII. Appendix

provide usage & testing instructions (write out high level testing procedures, i.e., how did I test), i.e., what's on gh. Use screenshots

References

[1] reference