Dynamic Calendar Report

Logan Parker and Rylan Cox

COSC 329

Project Outline:

The purpose of this project was to create and use a Dynamic Bayesian Network (DBN), alongside a calendar API, in order to send a user notifications (emails and pop-ups) based on their preferences and data within their calendar. This was executed using Matlab for our DBN, which received inputs via a Python script reading data from a user's Google Calendar.

The aforementioned Python script reads the amount of events in the upcoming week, their importance, and how far away they are. It then inputs these values, along with the preset value of how frequently a user checks their calendar, into the Matlab DBN. Using the data passed from Python, our DBN generates a list of actions created across 7 time-steps, with each timestep modeled as the passing of a single day. This list of actions is then returned to Python, in order to attach the specific reminders to events. The Python script also uses this list of actions to determine the amount of email and pop-up reminders that are sent, as well as when they are sent. This is due in part to the Google Calendar API only allowing for a maximum of five notifications per event. However, in order to avoid potentially annoying users, notifications were limited to a maximum of one email and three pop-up notifications per event.

In its current implementation, this project queries the Google Calendar API once per week, checking events within a seven day period from when it is run. If this project were to be implemented with the goal of being widely used, it would be more optimal to query the calendar seven days before each event, in order to give an action for each day leading up to the event.

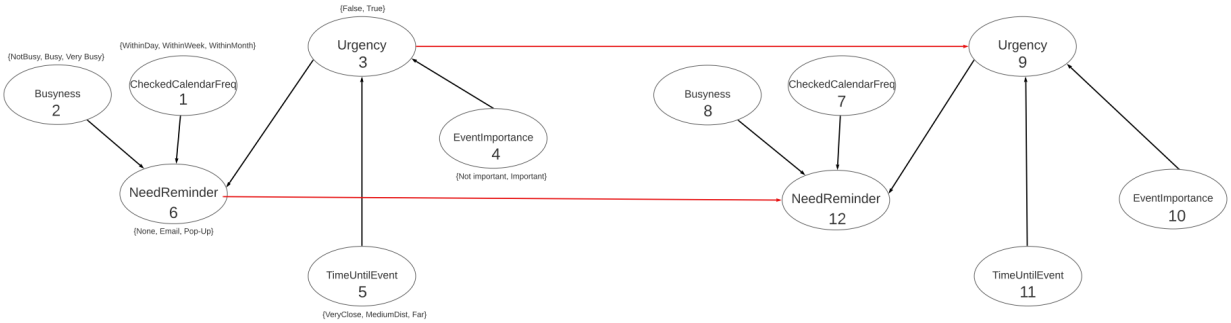
At the time of writing, there are no known bugs within the code, and the actions output by the DBN are reasonable given the evidence that is entered. Querying the Google Calendar API, as well as feeding data back into it is successful. Although unlikely, if a user has approximately 300 or more events within their calendar in a given week, the user may run into an issue with the API throttling requests. This is because there is a maximum of 600 requests per minute allowed to the API. The maximum requests per minute that was observed during testing was 12.5 according to the developer console, and as such, users are not expected to run into many problems in that regard. In addition, given the handcrafted nature of our CPTs, there is some uncertainty regarding the values. Although the system outputs reasonable actions, there is room for improvement within the CPTs, as some have over 160 values. Due to the size of these CPTs, defining perfect values for each combination is near impossible, and the current values entered provide good results, however, with more time, better results could likely be achieved.

DBN And CPTs:

This following section goes over the Dynamic Bayesian Network (DBN), Constructed Probability Tables (CPTs), and testing of the DBN made in Matlab.

The DBN Model:

Figure 1. The constructed DBN model



As seen in **figure 1.** The DBN has been modeled to contain 2 hidden nodes and 4 observation nodes. The definition for the nodes are as follows.

Observation Nodes:

- The **CheckCalendarFrequency (CCR)** node represents how often the user checks their calendar. This is a constant value that the user inputs, containing 3 possibilities: *daily, weekly,* and *monthly*. This is a value set by the user before running the program.
- The **Busyness** node represents how busy the user is. It has 3 possibilities, *not busy, busy, and very busy*. Busyness is defined if a user has between 5 and 10 events in a week. Anything below 5 is defined as not being busy, and anything above 10 is defined as being very busy. These values are read from the Google calendar API.
- The **EventImportance** node represents if an event is important or not. It has 2 possible values: *not important,* and *important*. These values are set and read by changing the color of an event in the calendar. Event's coloured red are considered to be important events.
- The **TimeUntilEvent** node represents the time until an event occurs. It has 3 possibilities: *very close, medium distance,* and *far away*. A very close event is within 2 days from current day, a medium distance event is between 3 and 5 days away, and a far away event is between 6 and 7 days away. This is read from the Google calendar API.

Hidden Nodes:

- The **Urgency** node represents whether or not an event is urgent. This node takes the input from the EventImportance and TimeUntilEvent nodes and outputs 2 possibilities: *not urgent,* and *urgent*. These values are then fed into the NeedReminder node.

- The **NeedReminder** node represents what kind of reminder the user needs, if they need one. This node reads the outputs from the Urgency, Busyness and CCR nodes and outputs three possibilities: *no notification, email notification,* and *pop-up notification*. This node is also used for inference because it changes based on it's previous values.

The CPTs:

The accompanying file, *FinalCPTs.xlsx* contains all of the CPTs and their values. They are each labeled with a CPT number and the below explanations will be referencing these numbers. For the larger CPTs there will not be a large description of each value, but rather a summary of the logic used to set them.

Observation CPTs:

- **CPT 1.** Is the CPT relating to the Important Node. This CPT helps determine whether an event is important or not. This node is evenly distributed because there is a 50% chance an event is important.
- **CPT 2.** Is the CPT relating to the Time Until Event Node. This CPT helps determine the distance until an event. This is evenly distributed, as the time until an event is unknown until read from the calendar API.
- **CPT 5.** Is the CPT relating to the Busyness node. This CPT helps determine whether a user is not busy, busy, or very busy. This assumes users are busy 60% of the time, and then are either not busy, or very busy 20% of the time.
- **CPT 6.** Is the CPT relating to whether a user checks their calendar daily, weekly, or monthly. It is assumed that 60% of users check their calendar daily, 30% check weekly, and 10% check monthly.

Hidden CPTs:

- **CPT 3.** Is the CPT relating to the Urgency of an event. This takes the input of both the Importance Node and the Time Until Event Node and outputs whether or not the event is important. For this CPT there is a significantly higher chance of an event being urgent when the event is important. There is also an increased chance of the event being urgent the closer the event is to the current day.
- **CPT 7.** Is the CPT relating to the user needing a reminder. This CPT takes three inputs, whether or not the event is urgent, the busyness of the user, and the frequency the user checks their calendar. The CPT then outputs whether the user needs an email, a pop-up notification, or no notification at all. When the user does not check their calendar frequently it heavily leans toward notifying them. When the user is less busy it values emails over pop-ups. Lastly when the event is urgent it heavily leans towards notifying the user in general.

Hidden Inferenced CPTs:

- **CPT 4.** Is the CPT for an event's urgency that takes account for what the previous urgency value was. When an event was previously urgent the CPT strongly leans towards the event remaining to be urgent.
- **CPT 8.** Is the CPT for determining if a user needs a reminder that takes account for the previous value of need reminder. When the user was previously not notified not much

has changed so the values stay the same as they were in the original CPT. When the user was previously emailed the CPT leans towards either not notifying them again or sending a pop-up, because the system does not want to send multiple emails to a user (the Python code for using the calendar API actually limits the amount of email sent). Lastly, when a pop-up was previously sent it leans heavily to either not notifying the user again, or sending another pop-up. The program does not want to send an email if a pop-up was previously sent because it is past the point of wanting to send email.

Testing the model:

A large portion of the project was creating, designing, and simulating the DBN. Matlab and bnt-master engine were used to simulate the DBN and read the outputs. The following is a list of cases displaying how the DBN was tested along with explanations for the outputs. The beginning of each output says to email the user because this is the base state, this value is ignored in Python when reminders are set. Each timestep for inference is a single day.

Figure 2. User checks daily and is not busy. The event is unimportant, and far away.

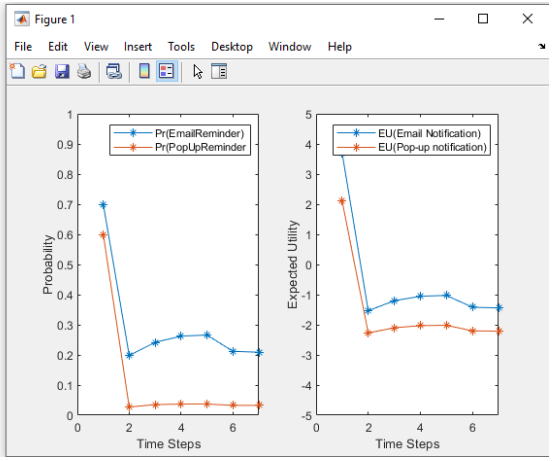
```
>> ActionList = sim_reminder(mk_reminders, 1, 3, 1, 1)

ActionList =

  1×8 string array

    "email"    "none"    "none"    "none"    "none"    "none"    "none"    "none"

>>
```



In **Figure 2.** The user does not need any form of reminder because they already check their calendar daily and the specific event in question has no urgency.

Figure 3. User checks weekly and is not busy. The event is unimportant and far away.

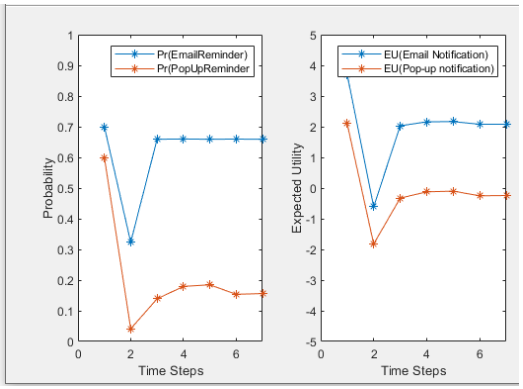
```
>> ActionList = sim_reminder(mk_reminders, 1, 3, 2, 1)

ActionList =

  1×8 string array

    "email"    "none"    "email"    "email"    "email"    "email"    "email"    "email"

>>
```



In **Figure 3.** The DBN initially does not want to notify the user, but for the rest of the simulation it wants to send an email. This works with our program because we only send the user one email and it tells the program on which day to send the email, which is 2 days after the day we run the program.

Figure 4. User checks monthly and is not busy. The event is unimportant and far away.

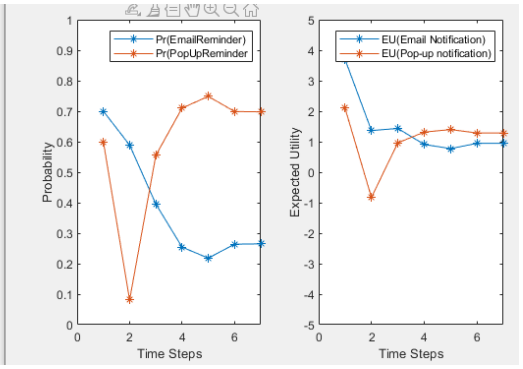
```
>> ActionList = sim_reminder(mk_reminders, 1, 3, 3, 1)

ActionList =

  1×8 string array

    "email"    "email"    "email"    "popup"    "popup"    "popup"    "popup"    "popup"

>>
```



In **Figure 4.** The program says to email the user and then send several pop-ups. The DBN leans heavily towards notifying users that only check their calendar once a month so these outputs make sense even if the event is not urgent we still want to remind them.

Figure 5. User checks daily and is busy. The event is important and very close.

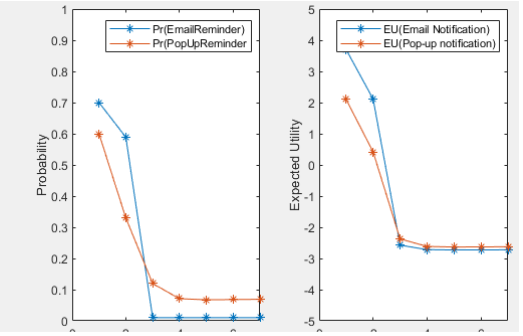
```
>> ActionList = sim_reminder(mk_reminders, 2, 1, 1, 3)

ActionList =

  1×8 string array

    "email"    "email"    "none"    "none"    "none"    "none"    "none"    "none"

>>
```



In **Figure 5.** When the event is clearly urgent and the user is busy the program only sends a single email because the user checks their calendar daily. It does not want to send them lots of notifications when they are already checking daily.

Figure 6. User checks weekly and is busy. The event is important and very close.

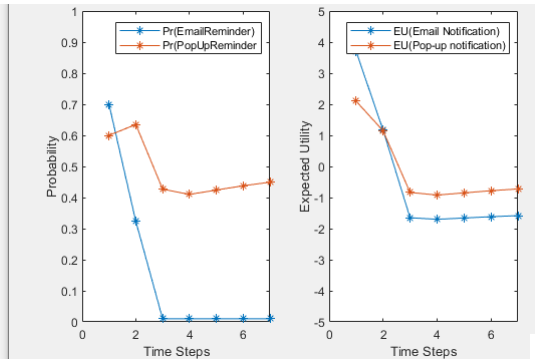
```
>> ActionList = sim_reminder(mk_reminders, 2, 1, 2, 3)

ActionList =

  1×8 string array

    "email"    "email"    "none"    "none"    "none"    "none"    "none"    "none"

>>
```



In **Figure 6.** The DBN says to send an email as soon as possible but then nothing after that. This is one of the weakest outputs. It was very difficult to fix this output without having the other outputs get drastically worse. Although, since only one email is ever sent, this still works compared to **Figure 3.** when there is a similar user input. They both only send one email but when the event is more important we send an email earlier.

Figure 7. User checks monthly and is busy. The event is important and very close.

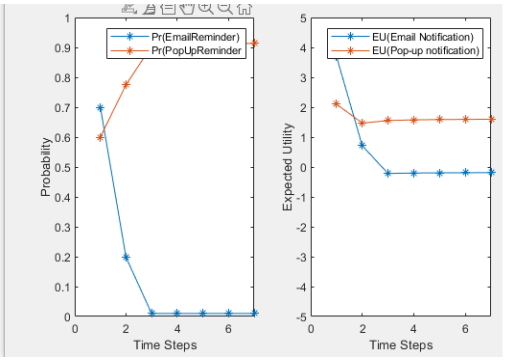
```
>> ActionList = sim_reminder(mk_reminders, 2, 1, 3, 3)

ActionList =

  1×8 string array

    "email"    "popup"    "popup"    "popup"    "popup"    "popup"    "popup"    "popup"

>>
```



In **Figure 7.** The user only checks monthly and this event has maximum importance, therefore the DBN outputs pop-up notifications each time which is the desired output.

Summary:

Defining these CPTs was incredibly difficult due to their large size. Due to their size, perfecting these CPTs is nearly impossible. Unfortunately this means that some of the outputs are not exactly as envisioned, however, they are still within reason..

Calendar Testing:

Below are the 9 test cases along with explanations of the inputs and outputs. Examples for each level of busyness are also included.

Figure 8. An example of a non busy calendar.

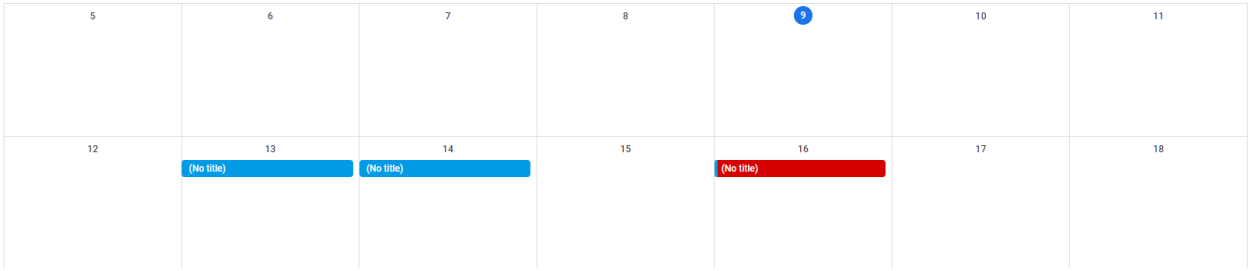


Figure 9. An example of a busy calendar.

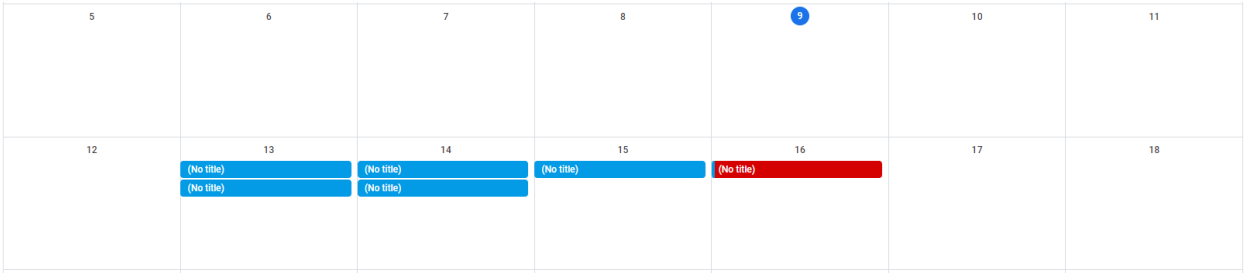


Figure 10. An example of a very busy calendar.

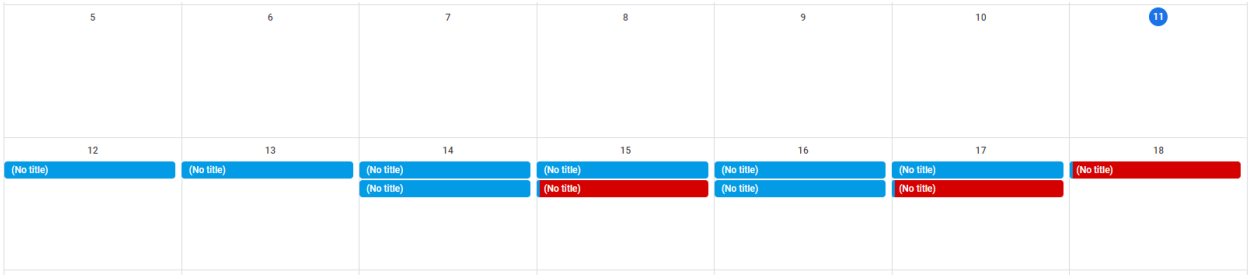
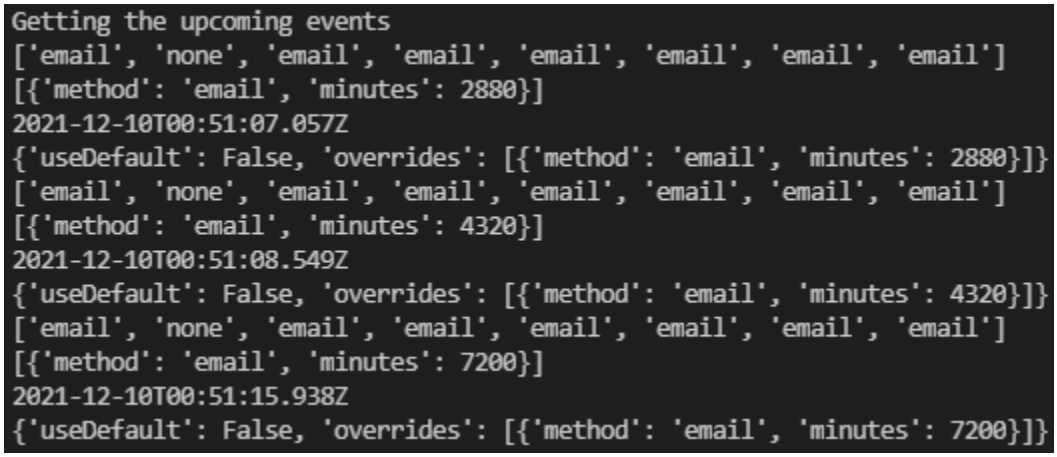


Figure 11. Calendar output for a user who checks their calendar daily and has a non busy calendar.

```
Getting the upcoming events
['email', 'none', 'none', 'none', 'none', 'none', 'none', 'none']
[]
['email', 'none', 'none', 'none', 'none', 'none', 'none', 'none']
[]
['email', 'none', 'none', 'none', 'none', 'none', 'none', 'none']
[]
```

In **Figure 11.** The system has decided to not notify the user which is the desired output. The user checks their calendar daily and they do not have a busy week meaning they should not be notified.

Figure 12. Calendar output for a user who checks their calendar weekly and has a non busy calendar.



In **Figure 12.** Here, the DBN has decided to send emails for each of the events which is the desired output. The first day no notification is sent. Since the program limits notifications to a single email this result just tells the program what day to send the email.

Figure 13. Calendar output for a user who checks their calendar monthly and has a non busy calendar.

In **Figure 13.** Each event gets a different set of actions. It's an email and a set of pop-ups. This is the desired output because the user only checks their calendar once a month so the program notifies them about their events.

Figure 14. Calendar output for a user who checks their calendar daily and has a busy calendar.

In **Figure 14.** The user only receives a single email for each event. Since they check their calendar daily this is the prefered output. Sending them an email with one reminder is what is desired in this case.

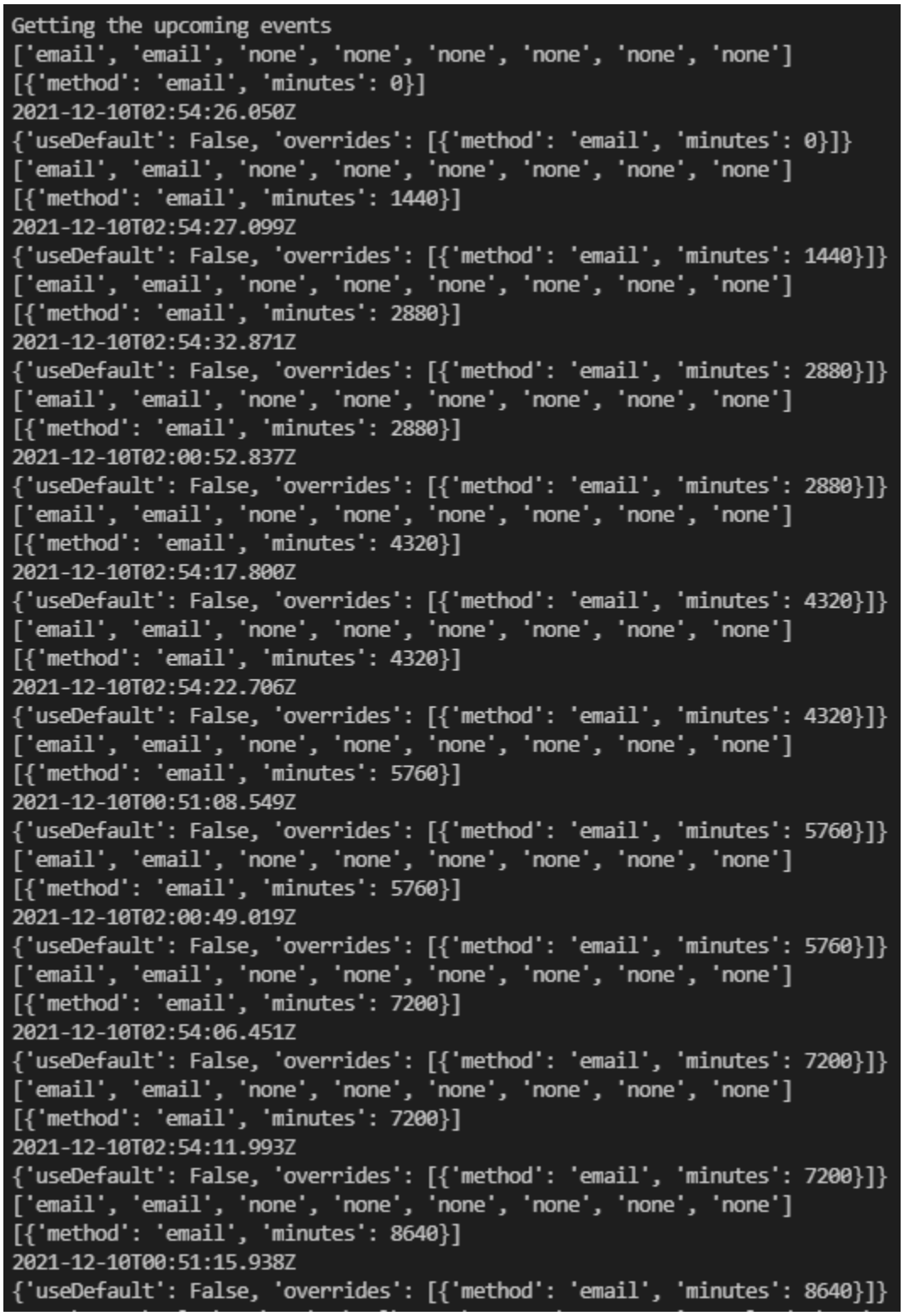
Figure 15. Calendar output for a user who checks their calendar weekly and has a busy calendar.

In **Figure 15.** This response is a bit extreme which is explained by the issue in the DBN section. The desired output for the user checking weekly and being busy would most likely be a single email and 1 maybe 2 pop-ups. Unfortunately it can be seen above that the output for each event is the maximum output.

Figure 16. Calendar output for a user who checks their calendar monthly and has a busy calendar.

Figure 16. Has a similar output to **Figure 15.** which is not ideal. Although, the output in **Figure 16.** is appropriate for this case. Since the user checks their calendar monthly and is busy, an email and maximum pop-ups is the desired output.

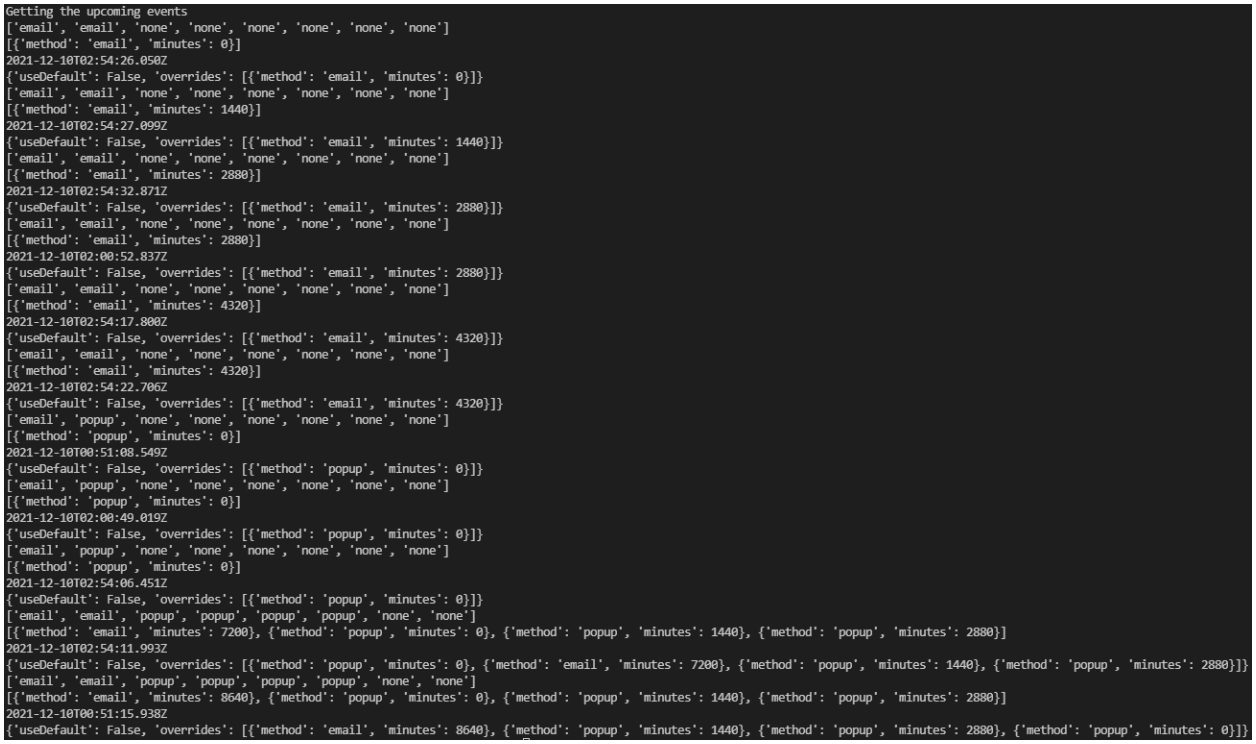
Figure 17. Calendar output for a user who checks their calendar daily and has a very busy calendar.



```
Getting the upcoming events
['email', 'email', 'none', 'none', 'none', 'none', 'none', 'none']
[{'method': 'email', 'minutes': 0}]
2021-12-10T02:54:26.050Z
{'useDefault': False, 'overrides': [{'method': 'email', 'minutes': 0}]}
['email', 'email', 'none', 'none', 'none', 'none', 'none', 'none']
[{'method': 'email', 'minutes': 1440}]
2021-12-10T02:54:27.099Z
{'useDefault': False, 'overrides': [{'method': 'email', 'minutes': 1440}]}
['email', 'email', 'none', 'none', 'none', 'none', 'none', 'none']
[{'method': 'email', 'minutes': 2880}]
2021-12-10T02:54:32.871Z
{'useDefault': False, 'overrides': [{'method': 'email', 'minutes': 2880}]}
['email', 'email', 'none', 'none', 'none', 'none', 'none', 'none']
[{'method': 'email', 'minutes': 2880}]
2021-12-10T02:00:52.837Z
{'useDefault': False, 'overrides': [{'method': 'email', 'minutes': 2880}]}
['email', 'email', 'none', 'none', 'none', 'none', 'none', 'none']
[{'method': 'email', 'minutes': 4320}]
2021-12-10T02:54:17.800Z
{'useDefault': False, 'overrides': [{'method': 'email', 'minutes': 4320}]}
['email', 'email', 'none', 'none', 'none', 'none', 'none', 'none']
[{'method': 'email', 'minutes': 4320}]
2021-12-10T02:54:22.706Z
{'useDefault': False, 'overrides': [{'method': 'email', 'minutes': 4320}]}
['email', 'email', 'none', 'none', 'none', 'none', 'none', 'none']
[{'method': 'email', 'minutes': 5760}]
2021-12-10T00:51:08.549Z
{'useDefault': False, 'overrides': [{'method': 'email', 'minutes': 5760}]}
['email', 'email', 'none', 'none', 'none', 'none', 'none', 'none']
[{'method': 'email', 'minutes': 5760}]
2021-12-10T02:00:49.019Z
{'useDefault': False, 'overrides': [{'method': 'email', 'minutes': 5760}]}
['email', 'email', 'none', 'none', 'none', 'none', 'none', 'none']
[{'method': 'email', 'minutes': 7200}]
2021-12-10T02:54:06.451Z
{'useDefault': False, 'overrides': [{'method': 'email', 'minutes': 7200}]}
['email', 'email', 'none', 'none', 'none', 'none', 'none', 'none']
[{'method': 'email', 'minutes': 7200}]
2021-12-10T02:54:11.993Z
{'useDefault': False, 'overrides': [{'method': 'email', 'minutes': 7200}]}
['email', 'email', 'none', 'none', 'none', 'none', 'none', 'none']
[{'method': 'email', 'minutes': 8640}]
2021-12-10T00:51:15.938Z
{'useDefault': False, 'overrides': [{'method': 'email', 'minutes': 8640}]}
```

In **Figure 17.** The user only receives a single email because they check their calendar daily. Despite being very busy since they check daily the program does not want to give the user a pop-up

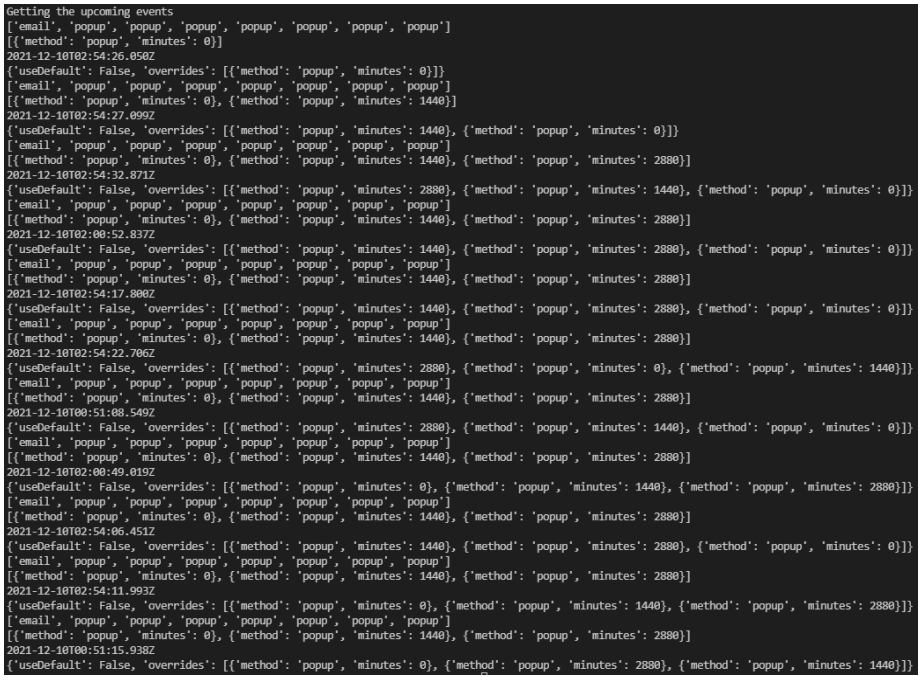
Figure 18. Calendar output for a user who checks their calendar weekly and has a very busy calendar.



```
Getting the upcoming events
['email', 'email', 'none', 'none', 'none', 'none', 'none', 'none']
[{'method': 'email', 'minutes': 0}]
2021-12-10T02:54:26.050Z
{'useDefault': False, 'overrides': [{'method': 'email', 'minutes': 0}]}
['email', 'email', 'none', 'none', 'none', 'none', 'none', 'none']
[{'method': 'email', 'minutes': 1440}]
2021-12-10T02:54:27.099Z
{'useDefault': False, 'overrides': [{'method': 'email', 'minutes': 1440}]}
['email', 'email', 'none', 'none', 'none', 'none', 'none', 'none']
[{'method': 'email', 'minutes': 2880}]
2021-12-10T02:54:32.871Z
{'useDefault': False, 'overrides': [{'method': 'email', 'minutes': 2880}]}
['email', 'email', 'none', 'none', 'none', 'none', 'none', 'none']
[{'method': 'email', 'minutes': 2880}]
2021-12-10T02:00:52.837Z
{'useDefault': False, 'overrides': [{'method': 'email', 'minutes': 2880}]}
['email', 'email', 'none', 'none', 'none', 'none', 'none', 'none']
[{'method': 'email', 'minutes': 4320}]
2021-12-10T02:54:17.800Z
{'useDefault': False, 'overrides': [{'method': 'email', 'minutes': 4320}]}
['email', 'email', 'none', 'none', 'none', 'none', 'none', 'none']
[{'method': 'email', 'minutes': 4320}]
2021-12-10T02:54:22.706Z
{'useDefault': False, 'overrides': [{'method': 'email', 'minutes': 4320}]}
['email', 'popup', 'none', 'none', 'none', 'none', 'none', 'none']
[{'method': 'popup', 'minutes': 0}]
2021-12-10T00:51:08.549Z
{'useDefault': False, 'overrides': [{'method': 'popup', 'minutes': 0}]}
['email', 'popup', 'none', 'none', 'none', 'none', 'none', 'none']
[{'method': 'popup', 'minutes': 0}]
2021-12-10T02:00:49.019Z
{'useDefault': False, 'overrides': [{'method': 'popup', 'minutes': 0}]}
['email', 'popup', 'none', 'none', 'none', 'none', 'none', 'none']
[{'method': 'popup', 'minutes': 0}]
2021-12-10T02:54:06.451Z
{'useDefault': False, 'overrides': [{'method': 'popup', 'minutes': 0}]}
['email', 'email', 'popup', 'popup', 'popup', 'popup', 'none', 'none']
[{'method': 'email', 'minutes': 7200}, {'method': 'popup', 'minutes': 0}, {'method': 'popup', 'minutes': 1440}, {'method': 'popup', 'minutes': 2880}]
2021-12-10T02:54:11.993Z
{'useDefault': False, 'overrides': [{'method': 'popup', 'minutes': 0}, {'method': 'email', 'minutes': 7200}, {'method': 'popup', 'minutes': 1440}, {'method': 'popup', 'minutes': 2880}]}
['email', 'email', 'popup', 'popup', 'popup', 'popup', 'none', 'none']
[{'method': 'email', 'minutes': 8640}, {'method': 'popup', 'minutes': 0}, {'method': 'popup', 'minutes': 1440}, {'method': 'popup', 'minutes': 2880}]
2021-12-10T00:51:15.938Z
{'useDefault': False, 'overrides': [{'method': 'email', 'minutes': 8640}, {'method': 'popup', 'minutes': 1440}, {'method': 'popup', 'minutes': 2880}, {'method': 'popup', 'minutes': 0}]}
```

Figure 18. Is the most varied output. The user receives a single email, then a single pop-up, and finally an email with max pop-ups. The variation is a welcomed output although jumping to maximum output is not desired.

Figure 19. Calendar output for a user who checks their calendar monthly and has a very busy calendar.



```
Getting the upcoming events
['email', 'popup', 'popup', 'popup', 'popup', 'popup', 'popup', 'popup']
[{'method': 'popup', 'minutes': 0}]
2021-12-10T02:54:26.050Z
{'useDefault': False, 'overrides': [{'method': 'popup', 'minutes': 0}]}
['email', 'popup', 'popup', 'popup', 'popup', 'popup', 'popup', 'popup']
[{'method': 'popup', 'minutes': 0}, {'method': 'popup', 'minutes': 1440}]
2021-12-10T02:54:27.099Z
{'useDefault': False, 'overrides': [{'method': 'popup', 'minutes': 1440}, {'method': 'popup', 'minutes': 0}]}
['email', 'popup', 'popup', 'popup', 'popup', 'popup', 'popup', 'popup']
[{'method': 'popup', 'minutes': 0}, {'method': 'popup', 'minutes': 1440}, {'method': 'popup', 'minutes': 2880}]
2021-12-10T02:54:32.871Z
{'useDefault': False, 'overrides': [{'method': 'popup', 'minutes': 2880}, {'method': 'popup', 'minutes': 1440}, {'method': 'popup', 'minutes': 0}]}
['email', 'popup', 'popup', 'popup', 'popup', 'popup', 'popup', 'popup']
[{'method': 'popup', 'minutes': 0}, {'method': 'popup', 'minutes': 1440}, {'method': 'popup', 'minutes': 2880}]
2021-12-10T02:00:52.837Z
{'useDefault': False, 'overrides': [{'method': 'popup', 'minutes': 1440}, {'method': 'popup', 'minutes': 2880}, {'method': 'popup', 'minutes': 0}]}
['email', 'popup', 'popup', 'popup', 'popup', 'popup', 'popup', 'popup']
[{'method': 'popup', 'minutes': 0}, {'method': 'popup', 'minutes': 1440}, {'method': 'popup', 'minutes': 2880}]
2021-12-10T02:54:17.800Z
{'useDefault': False, 'overrides': [{'method': 'popup', 'minutes': 1440}, {'method': 'popup', 'minutes': 2880}, {'method': 'popup', 'minutes': 0}]}
['email', 'popup', 'popup', 'popup', 'popup', 'popup', 'popup', 'popup']
[{'method': 'popup', 'minutes': 0}, {'method': 'popup', 'minutes': 1440}, {'method': 'popup', 'minutes': 2880}]
2021-12-10T02:54:22.706Z
{'useDefault': False, 'overrides': [{'method': 'popup', 'minutes': 2880}, {'method': 'popup', 'minutes': 0}, {'method': 'popup', 'minutes': 1440}]}
['email', 'popup', 'popup', 'popup', 'popup', 'popup', 'popup', 'popup']
[{'method': 'popup', 'minutes': 0}, {'method': 'popup', 'minutes': 1440}, {'method': 'popup', 'minutes': 2880}]
2021-12-10T00:51:08.549Z
{'useDefault': False, 'overrides': [{'method': 'popup', 'minutes': 2880}, {'method': 'popup', 'minutes': 1440}, {'method': 'popup', 'minutes': 0}]}
['email', 'popup', 'popup', 'popup', 'popup', 'popup', 'popup', 'popup']
[{'method': 'popup', 'minutes': 0}, {'method': 'popup', 'minutes': 1440}, {'method': 'popup', 'minutes': 2880}]
2021-12-10T02:00:49.019Z
{'useDefault': False, 'overrides': [{'method': 'popup', 'minutes': 0}, {'method': 'popup', 'minutes': 1440}, {'method': 'popup', 'minutes': 2880}]}
['email', 'popup', 'popup', 'popup', 'popup', 'popup', 'popup', 'popup']
[{'method': 'popup', 'minutes': 0}, {'method': 'popup', 'minutes': 1440}, {'method': 'popup', 'minutes': 2880}]
2021-12-10T02:54:06.451Z
{'useDefault': False, 'overrides': [{'method': 'popup', 'minutes': 1440}, {'method': 'popup', 'minutes': 2880}, {'method': 'popup', 'minutes': 0}]}
['email', 'popup', 'popup', 'popup', 'popup', 'popup', 'popup', 'popup']
[{'method': 'popup', 'minutes': 0}, {'method': 'popup', 'minutes': 1440}, {'method': 'popup', 'minutes': 2880}]
2021-12-10T02:54:11.993Z
{'useDefault': False, 'overrides': [{'method': 'popup', 'minutes': 0}, {'method': 'popup', 'minutes': 1440}, {'method': 'popup', 'minutes': 2880}]}
['email', 'popup', 'popup', 'popup', 'popup', 'popup', 'popup', 'popup']
[{'method': 'popup', 'minutes': 0}, {'method': 'popup', 'minutes': 1440}, {'method': 'popup', 'minutes': 2880}]
2021-12-10T00:51:15.938Z
{'useDefault': False, 'overrides': [{'method': 'popup', 'minutes': 0}, {'method': 'popup', 'minutes': 2880}, {'method': 'popup', 'minutes': 1440}]}
```

Figure 19. Gives every event an action list of only pop-ups. This is the desired output because this is the extreme case where the user only checks their calendar once a month and they are very busy. Unfortunately due to the limitations on the number of reminders a user can be given the full response is not stronger than if they were only busy.

Conclusion:

Relative to the scope of the project, all features have been successfully implemented. Little to no bugs are present within the project, although further refinement could be made with regard to the CPTs.