An Executive Design Document:

Classes:

GoogleAPI - Class is used to bridge to Google Calendar for functionality. The user provides credentials and can access their Google calendar information, as well as create new events through the program.

ParseData - Used to take a JSON object from the GoogleAPI class and parse it into objects that will be used by GUI. The ParseData class will also be used to create our Event objects and store them in our DoublyLinkedList data structure.

Event - The class object will be used to hold the following items: date, time, and description for a single event in a user's calendar and multiple entries will be grabbed from the GoogleAPI class. The class will contain setters and getters functions to obtain and record specific items in Event objects.

DoublyLinkedList - Used to store Event objects of the Event class, which gets its information from the Parsing class that is formatted JSON data from the GoogleAPI class. Will store our Event objects in Nodes and then be added to the end of the list for more efficient insertion and deletion in our data structure.

Node - Class to provide pointers and references for the DoublyLinkedList class structure, as well as define exactly what information will be held in the DoublyLinkedList(Events and their details).

GUI - Class used to represent data from the user and the GoogleAPI JSON object. Functions of the GUI class will include a comprehensive report of the user's events in a readable format.

TaskReport - Class is used to print the final report of all events that were created manually and by connecting to Google Calendar. The final report will consist of the date, time, description, and any other data that is obtained. The final report will also be listed from higher importance to least important.

Objects:

Node - Node objects are used to simplify the complexity of a data structure so that we efficiently store and organize elements. The node contains three main items the element, previous, and next, which are all essential in creating, inserting, deleting, and organizing our object nodes in DoublyLinkedList.

DoublyLinkedList - DoublyLinkedList is a data structure that is used for faster and more efficient insertion and deletion of nodes, with the only negative aspect being the large usage of memory. Despite this flaw, it's an ideal choice because the elements already being in order in Nodes and simply need to be inserted or deleted in a data structure.

Event - Our created Event object will contain three main items: time, date, and description. These items will be essential in minimizing the amount of data that we obtain from our GoogleAPI class that returns a large string that will be parsed and used to create our Event objects.

GUI - Object of the GUI class that will display events, as well as offer functionality to print out a report of your calendar events with their respective time/date/description, as well as an add-task button(that will not be functional due to scope issues) that was stated in our requirements.

Data formats:

How the data given to us was formatted:
The data of future events from Google Calendar is obtained, with an API call, as a dictionary object containing the dateTime, event type, description, and also some user credentials. Each event is then received individually in a for loop from most recent to finish in a 30-day period. Each event is stored in a list so that we can process each event in our ParseData class for more efficient data processing in case a large number of events needs to be processed.

How we will format data:
- One large string, not a JSON as expected

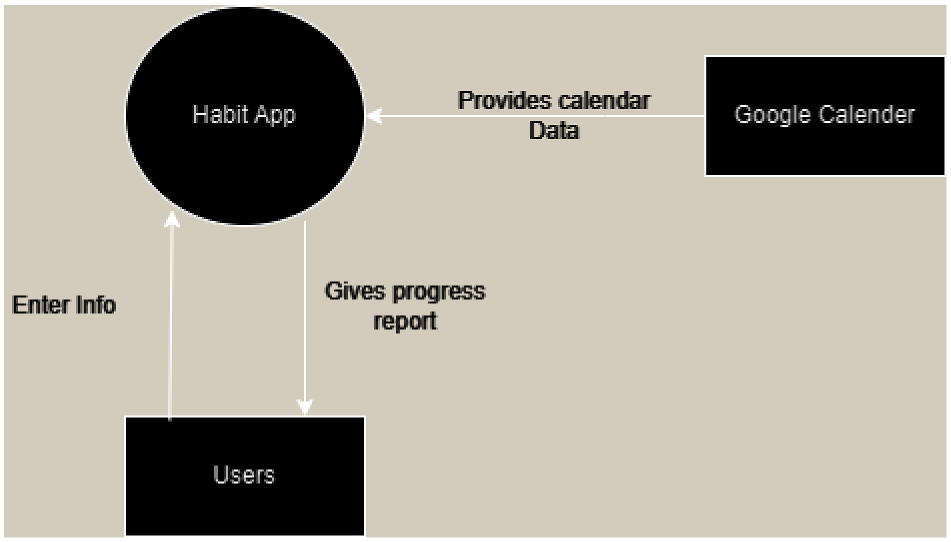
How we will format data in the report:
- # task(s) this month:
    - (List out events and their date)
- # task(s) this week:
    - (List out events and their date)
- # task(s) today:
    - (List out events/date/description)

User Interactions:

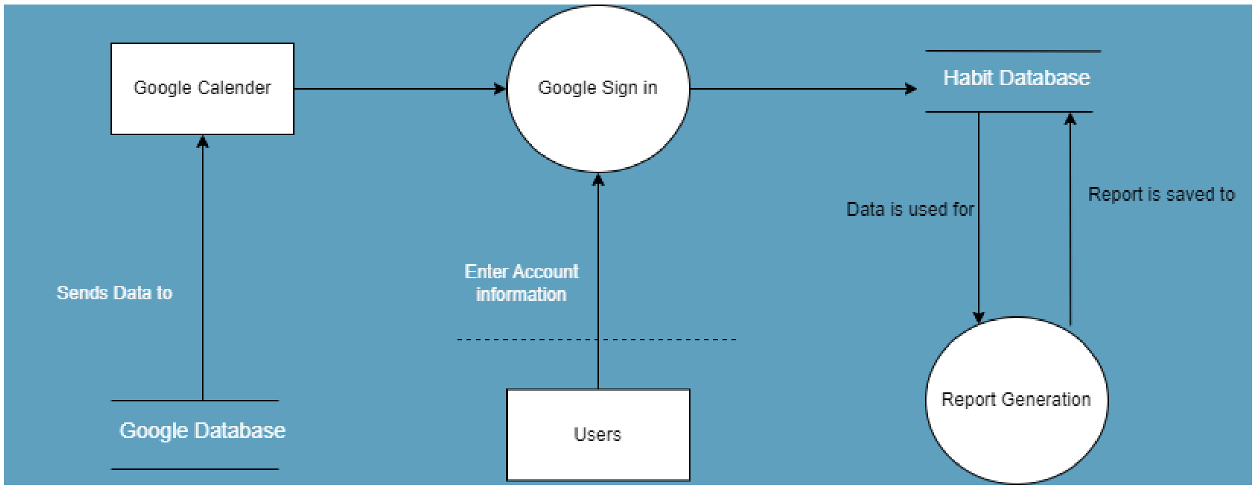
GUI - Main interaction is between the user and the GUI. Providing credentials to connect and access their calendar info, as well as being able to create new events with dates, times, and descriptions.

Diagrams:

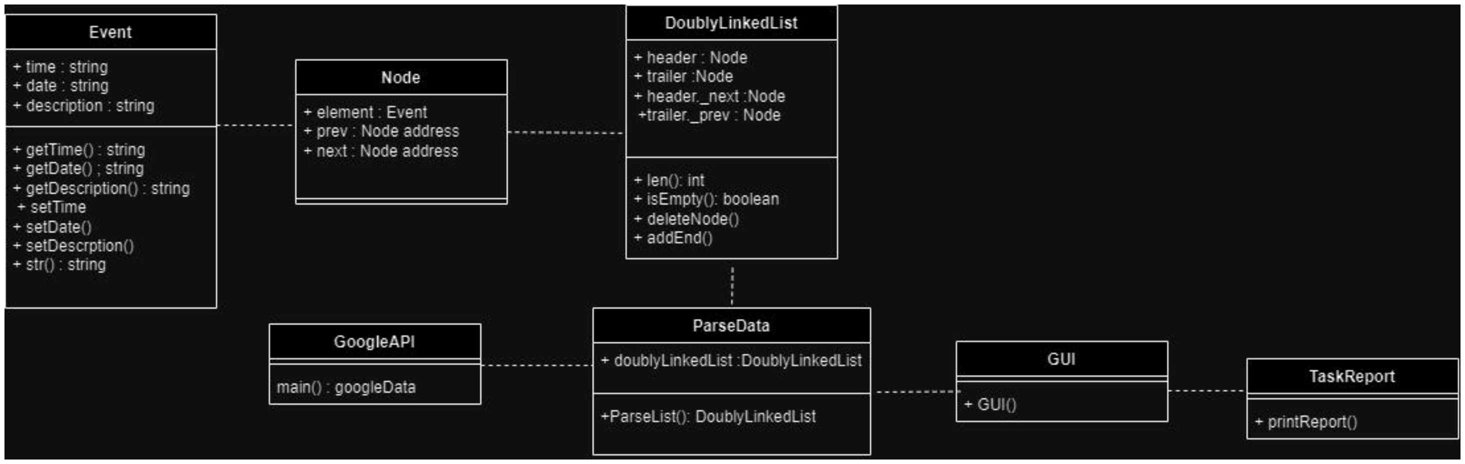
Level 0 DFD:



Level 1 DFD:

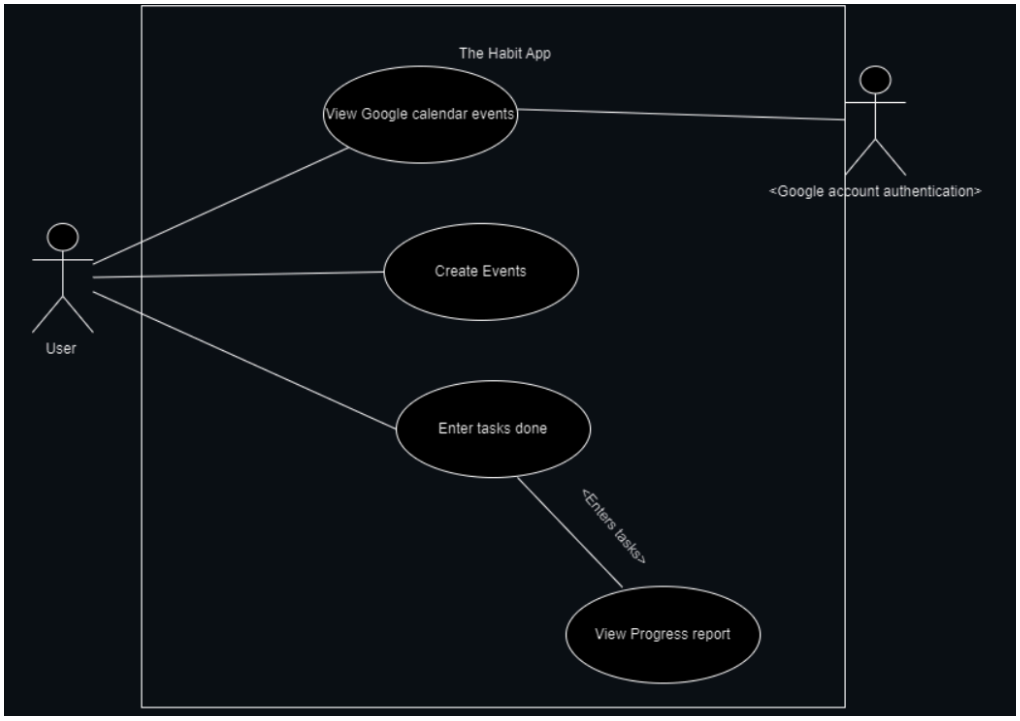


UML:



Additional Diagrams:

UML Use Case:



Security Design:

Limiting features - To restrict users from accessing private data and at the same time make it easier for them to navigate through the Habit app we plan on having the GUI only have 2 options for using the app. One option will be to be able to use Google Calendar to obtain all information in a 30-day period to add to the users tasks. The second option will be to generate a final report of all tasks with the dateTime, title, and description in a week period to a 30-day period.

Error Handling - To keep users' information secure and keep the app running smoothly we plan on implementing error handling in case users run into any problems with credentials or data. This will be done by adding try blocks and if statements in our code in case any errors occur or if an expected problem occurs. This will also ensure that no data is lost while the user is navigating the app and adding events.

Limiting user input - At a point, we plan on having an option where a user will be able to add an event as a feature to make it easier than simply connecting your Google calendar. This will be done by prompting the user with questions such as what would you like to call this event, what time will this need to be done, and would you like to add a description? How we will add security to this by limiting what a user may input to prevent any data breaches or any sort of malicious activity that may cause future problems for the habit app.