Team M
Cycle 1.5
Requirements/Use Cases Document

Functional Requirements:

1. Search for a person via first and last name
    a. Uses: Finding a specific person within the app, if full name is known
    b. Info to specify:
        1. Name (first, last)
    c. Limit: Some names have repeats and if name not known cannot complete this search
    d. Uses: Finding a specific person and the information and connections known about them.
2. Find grandparents of a specific person
    a. Uses: Find the relationship between a known person and their grandparents
    b. Info to specify:
        i. Name of known person
    c. Limit: must know a person in the tree to do this search
    d. Uses: able to build the history of one specific person because as you discover grandparents, the user can continue to search their names up and build a tree from the nodes up to the root.
3. Given a person, determine if they are known to the app
    a. Uses: looking for an ancestor, preventing people from being added twice
    b. Search for:
        i. Name
        ii. Birthday
        iii. City of birth/residence
        iv. All 3 of these conditions are useful because it narrows down the number of people who have the same names, birthdays or cities
    c. User can input the information from (b); return a list of people who apply with this information
4. Add a person to the app
    a. Automatically pass (1) to check if the person is in the app or not
    b. Information to specify:
        i. Name
        ii. Birthday
        iii. City of birth/residence
        iv. Any known family relationships (spouse, parents, children)
    c. Uses: filling out the user's family tree

   d. Limits: the specified information is required to add a person to the app
   e. User can enter in information listed in (b) to generate a new node on the family tree graph with the associated information
5. Collect information on each family
   a. Read in a text file
      i. contents:
      ii. people (name, date of birth, birth place, current residence, etc.)
      iii. people's relationships (spouse, parents, children)
      iv. for each specific person:
   b. Save added information back to the text file
      i. When information is updated on the app, fill in this information on the text file
   c. This will automatically fill in the family tree based on the information in the file
   d. Uses: adding large amounts of data to the app so it does not have to be done manually (by adding each person with (2))
      i. Saves large amounts of information to a text file so it is not lost
   e. User can upload a file to automatically generate or update an existing graph with the information in the file
6. Record the start and end dates of a partnership
   a. Read in the file:
      i. Create the relationships for spouses
      ii. Read in the dates from the partnership section of the file
      iii. First date signifies beginning of relationship, second date is the end
   b. Save added information back to the text file
      i. When information is updated on the app, fill in this information on the text file
7. Record children in a new or existing partnership
   a. User finds a specific partnership in the tree
   b. User adds a child:
      i. Information to specify:
         1. Name
         2. Birthday
         3. City of birth/residence
         4. Any known family relationships (spouse, parents, children)
   c. Save added information back to the text file
      i. When information is updated on the app, fill in this information on the text file
8. For any person, find people of a specified relationship
   **Utilizing the relationship using "mother of"," father of" or "child of" is the least erroneous way to navigate a specific relationship. This is because there can be a

possibility of multiple grandparents and multiple grandchildren within any branch of the overall family tree.

   a. Parent is <u>Mother(Full name) is mother of "c"</u>, <u>Father(Fullname) is father of "c".</u>
   b. Grandparent is searched through "mother of".
   c. Child is <u>Child(Full name) is child of "a"</u>
   d. Grandchild is searched through "child of".
   e. Cousin is searched through "child of".
   f. Partner is a person who does not have "child of" defaults as partner.

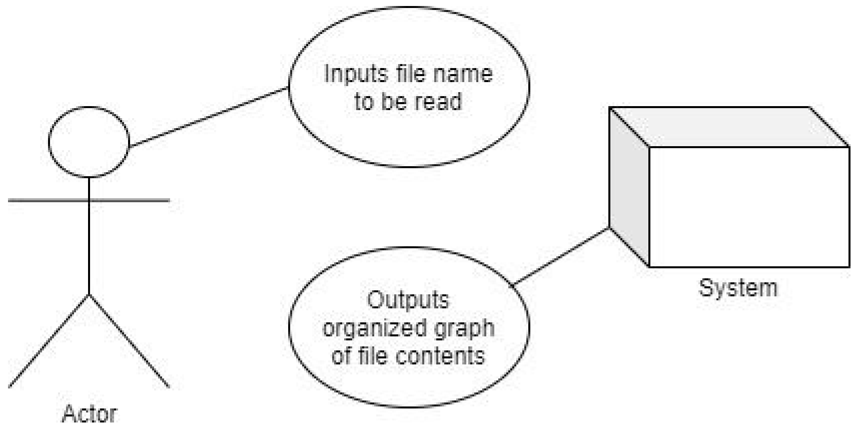
9. Determine if two people are related (they will have a common ancestor somewhere "up" in their family tree)
   a. Create two lists to collect the data of each person as it is scaling towards the root of the family tree
   b. Compare each person added to the list to see if they match
   c. If matching, this is the related ancestor
   d. If root of family tree is reached and no relationship established, no related ancestor exists
   e. Limits: only shows first related ancestor, does not continue to check for other ones

<u>Nonfunctional Requirements:</u>

1. Use Java v11 and IntelliJ for our program development. Save our work in the course GIT repository for your team: Comp330Fall2020TeamM
2. Structure the system so that the user interface is separate from the logic and searching functions.
3. Have a well-structured functional decomposition of the app into separate parts. This decomposition should support separate development of key components by individual programmers.

Use Case Models/Narratives:

Event: Read in file
Actor: Person or computer system
Purpose: To read in a file from which to create a family tree from
Overview: The actor provides the system with a file, the system reads the file and provides the user the graph to the user. If the file is wrongly formatted, the system returns an error to the actor.


Inputs file name to be read
Outputs organized graph of file contents
System
Actor

Use Case Narrative:
User

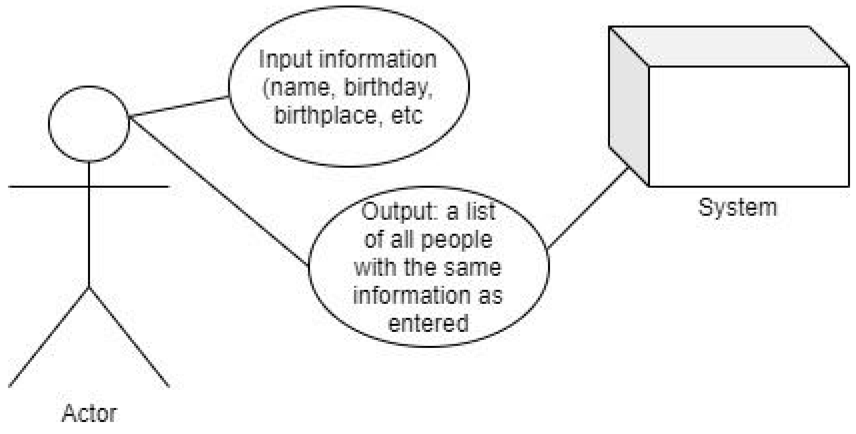
1. The user uploads a file to be read into the app
4. The user receives the output of a graph

System

2. The system reads the file and separates the pieces of information into a graph data structure
3. The system outputs the graph to the user

Alternative Flow:
Line 2: File structure is incorrect and the system cannot process information. Print error. Return to step 1.

Event: Determine if a person is known to the app
Actor: Person
Purpose: A user can check if a person is known in the family tree to the system
Overview: The actor provides the necessary information to search for a person and submits it. The system traverses the tree until it finds the person and returns them along with related info. If no person found the system prints an error.

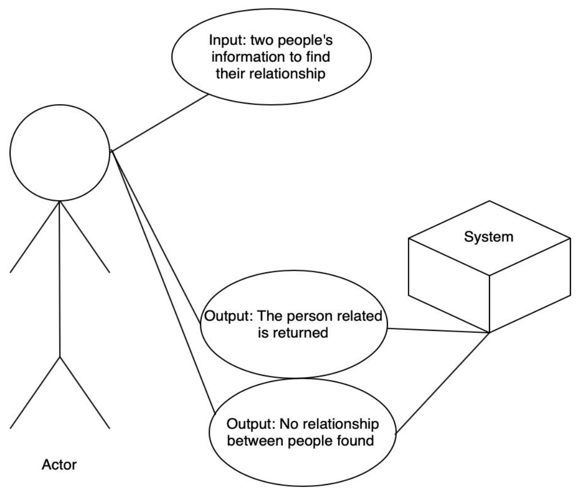

Input information (name, birthday, birthplace, etc
Output: a list of all people with the same information as entered
System
Actor

Use Case Narrative:
User
1. The user is given options for information to be entered
    a. Name (required)
    b. Birthdate
    c. Place of birth
    d. Spouse/Parents/Children
2. The user submits the entered information for the search
5. The system outputs the person along with related information

System
3. The system searches through the tree of people until a person with matching information is found.
4. The system outputs the person and their information and relationships to the user

Alternative Flow:
Line 3: The system searches through the system and does not find a match. Prints a system error. Return to step 2.

Event: Determine if two people are related
Actor: Person
Purpose: Determine if a relationship exists between two people.
Overview: The actor provides the needed info for two people and submits it. The system will traverse the tree to find both people, if one, both or none are found an error is returned. If found, the system will then traverse the nodes related to each until a relationship is found, if no relationship found an error is outputted. If a relationship is found, the system outputs the relationship to the actor.


Input: two people's information to find their relationship
System
Output: The person related is returned
Output: No relationship between people found
Actor

Use Case Narrative:
User:

1. User inputs names of the two people they want to find a relationship for
2. User submits the info into the search
7. The user receives the output displaying the relationship

System:

3. The system will traverse the tree towards the root for each person
4. It will create a list for each person of related ancestors and compare each person added, checking if they are identical
5. If the system finds a person who overlaps for both, this person is outputted to the user as the member both are related to
6. The system outputs the relationship

Alternative Flow:
Line 3: If the tree is traversed and one or both people cannot be located, the system will print an error. Return to step 2.
Line 4: Once the whole tree is traversed and no relationship is found between the people the system will output an error that there is no relationship between the two people. Go to step 7