# Requirements Document

Iram Jehan, Tara Le, Sophia Sklar, Ian Tompkins, Tien VoNguyen

## Summary

* 1. “Implement an app that keeps track of relationships between people and family genealogies or family trees. Family trees allow the user to determine ancestors and descendants of any person. Often family trees are incomplete especially during research to create them (there may be many people in them, some connected to each other and others not yet or ever connected). This app should allow storage and searching of arbitrarily complex sets of people partially connected into one or more family trees. Since children result from partnering between the child’s biological parents, the app includes records of partnership and any offspring associated with that partnering. Children are related to both their biological parents and parents may be in more than one partnership at the same or different times. It is required to trace all children of each person across all partnerships. It is required to trace upwards from any child to all their biological ancestors. It is required to trace downwards from any person to their descendants.”

## Purpose

* 1. The purpose of this application is to provide a way to allow users to document their family trees, see a readily understood depiction of their connections and relationships, as well as potentially discover familial connections with other users.

## Requirements

### Functional

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Req #** | **Function** | **Description** | **Expected Result** | **Priority** |
| **1** | **Data Import** | 1.1: Users will be able to import data into the program with text files.  1.2: Users will be able to update data in the program with text files.  1.1.1: Text files will include member information, which includes full name (last, first), birth date & place, death date & place (if applicable), and relationship to other members (beginning & end dates). | * The data from the imported text file should appear in the program * The new data from the imported text file should appear in the program | 1 |
| **2** | **Manual data addition** | 2.1: Users will be able to manually add members and their data into the program.  2.2: Users will be able to manually remove members and their data from the program.  2.3: Users will be able to manually update member data in the program. | * The manually added data should appear in the program * The manually removed data should disappear from the program * The manual changes should appear in the program | 3 |
| **3** | **Member search** | 3.1 Users will be able to search for members and their information within the entire database or within their own tree by name, place, or birthdate.  3.2: Users will be able to see the corresponding information for the member that they search for: name, birth date & place, relationships to other members & offspring, death date & place (if applicable). | * Any matching member and their data to the search term should be displayed * All the information corresponding to the matching member should be displayed | 2 |
| **4** | **Common relationships** | 4.1: Users will be able to view members that are related to them and other members (including other trees) to determine more distant relationships. | * Users with other members in common should be displayed | 4 |
| **5** | **Display** | 5.1: Users will be able to see family trees displayed depicting all members and their names, relationship connections, and summary of information (birth to death years & current location).  5.1.1: Click on a specific member to see more of their information (birth place, death place, etc.) | * An image of a user’s family tree should be displayed along with their relationships, names, birthdate, and death date. | 5 |
| (6) | Data export | 6.1 Users can export their family tree data as an image file of the tree depicting all members and their relationships or as a text file in the original import format. | * Users should be able to export the image of the family tree as a downloadable file. | 6 |

## Cycle Deliverables

### Cycle One

* + 1. Non-functional
       1. Requirements Document - *complete*
       2. Use Cases Document - *complete*
    2. Functional
       1. Read-in Functionality (req. #1)- *pending*

### Cycle Two

* + 1. Non-functional
       1. Design, decomposition, components
       2. Formal inspection
       3. Peer feedback
    2. Functional
       1. Manual data addition (req. #2)
       2. Member Search (req. #3)

### Cycle Three

* + 1. Non-functional
       1. Debugging
       2. Finalizing
    2. Functional
       1. Common relationships (req. #4)
       2. Display (req. #5)