KinTree Developer Guide

LAST UPDATED MAY 9, 2025

Table of Contents

Overview

KinTree is an application that allows users to connect with family members, document their familial relationships, and build a dynamic family tree. The aim of this project is to create a socially engaging platform for users to maintain and foster family relations.

KinTree is developed as a part of the Software Engineering Senior Project course sequence at Mississippi State University. The idea for the project was originated by Dr. Charan Gudla. This document serves to educate future team members or stakeholders of the project about the application's functionalities and technical makeup.

Project History

The KinTree project began in Fall of 2024 under Dr. Charan Gudla. At the time, Dr. Gudla taught SE Sr. Project I (CSE 3213) and served as the originator of the project idea and the team's client. The project team consisted of Owen Adams, Destiny Milsap, Kennedi James, and Jade Thompson. During this semester, the application's architecture was designed, and interactive prototypes were made using Figma. Prototypes were made for both desktop and mobile designs.

The project continued in Spring of 2025 with the same development team and client. SE Sr. Project II (CSE 3223) was then taught by Mr. Josh Crowson. Several core functionalities were implemented during this semester.

Technologies

KinTree is built using React, a component-based web framework that utilizes JavaScript. React was used to compose the frontend of the project. Our client code is organized into components and pages, and each page is styled individually. The frontend utilizes a router that manages component rendering based on path.

The project's backend utilizes a Node.js server and API, developed using JavaScript. The API consists of several models, controllers, and routes. Knex is used to support the API and database interfacing. The project uses a MySQL database composed of several tables for storage and retrieval.

As of this document's publication, KinTree is not publicly hosted as an application. Local configuration instructions are available within the project's public repository.

Technical Organization

Frontend

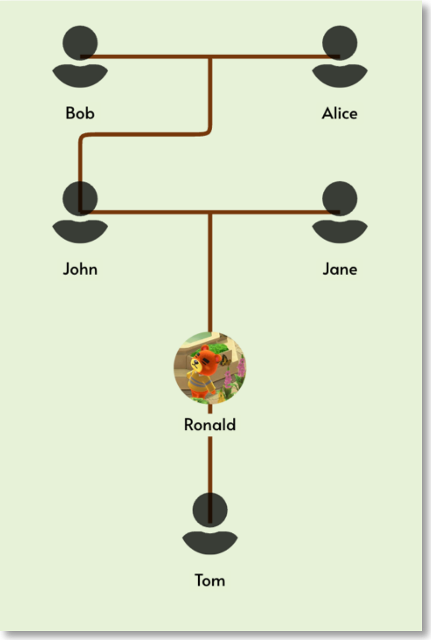
The frontend of the application is located within the client directory within the main project directory. Within this folder, the main code of the application is within the src folder. This folder is organized into pages (within the pages folder) and components (within the components folder). Pages often contain components, which are often smaller in size and scope than a full page, but pages themselves can also be viewed as components that are rendered by React. The src folder also contains an assets folder, which stores images used in the app, and a fonts folder.

Some standalone files of note include index.js, which manages the app's routing. Protected routes are used within the app; the main pages of the app are not reachable unless a user is logged in. Login status is determined by values in local storage in a user's browser, which the CurrentUserProvider.js file writes to and views. Note that local storage can be edited by end users and that this approach is not a secure way to manage authorization in a production-level application.

Major Implemented Features

Family Tree Management

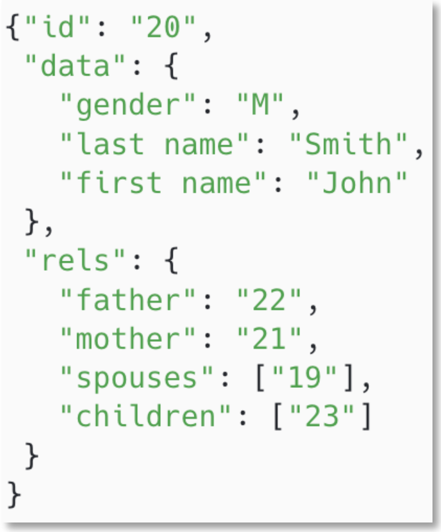
All registered users have a base tree, or personal family tree. By default, this tree only includes a node for the user themselves. As family members are added to the tree, connections are visualized as nodes and straight lines. Family members are not automatically added to this tree; users must define direct family connections for a family member. For instance, in the example tree, Tom might be added to the tree if the user defines Ronald as Tom's parent.



Family trees are dynamically generated using data about the members and their relationships. This data must follow a specific format and organization for the family-chart NPM package to properly process it. Tree data is formatted as an array of JSON objects, with each object containing the ID, information, and relationships for each tree member. Tree members can have multiple spouses and children.

What is the ID?

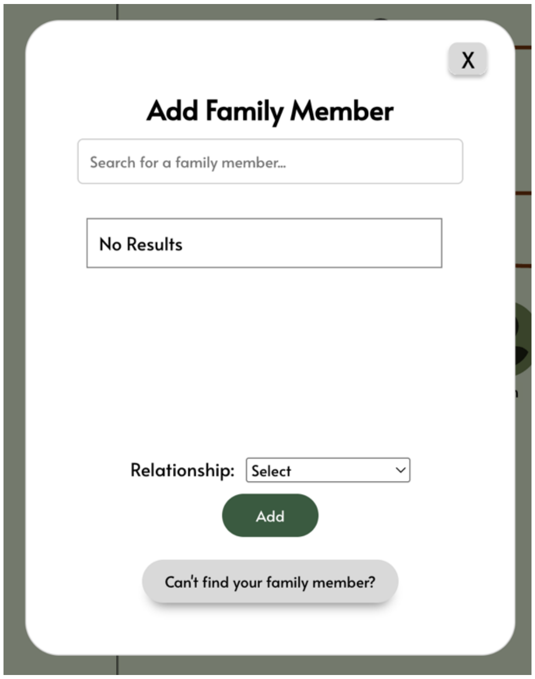
In the application's current state, the ID within the tree object is known as the person's Family Member ID. This is the ID that is used to reference that family member in most cases, including within the tree object, on the account page, and in the relationships table. Family Member IDs are unique in the app's database, but one person might have multiple FMIDs associated with them. This is because FMIDs are specific to a person's instance within a user's group of family members. So, if a person is a family member to multiple users, they will technically possess multiple FMIDs for those separate instances. When a unique ID is needed for a person, user ID can be used if that user has a registered account with KinTree.



```
{"id": "20",
 "data": {
   "gender": "M",
   "last name": "Smith",
   "first name": "John"
 },
 "rels": {
   "father": "22",
   "mother": "21",
   "spouses": ["19"],
   "children": ["23"]
 }
}
```

Family Member Management

Users can add family members in a couple of ways. Added family members do not have to have registered accounts within KinTree. To add a family member that is actively registered with KinTree, users can search for existing users and then manually indicate a relationship type. This relationship type is separate from the relationship type that is defined for family tree integration. The initial relationship defined here is not validated by the application. It merely is a title that the user gives the person, that is only displayed for the user themselves when they view the person's profile page.



To add a family member that is not registered within KinTree, users can click the "Can't find your family member?" button and then enter details about the family member manually. This will create a listing for a family member that is not linked with a KinTree account. This family member can still be added to the user’s family tree.

Sharing Trees

Users can share copies of their trees with other users for viewing. Sharing can only be conducted with family members, but they do not have to have registered KinTree accounts. If the family member does have a KinTree account, an email address will automatically populate in the field, but this email address can be edited to another one if the user desires. Users can also share their tree with non-registered family members, but they must input an email address to send the tree to. Currently, shared trees are only viewable from the Shared Trees tab in the navigation bar. In the future, it would be ideal for family members to receive emails when a tree has been shared with them, so that non-registered users can access them as well.

Known Issues

Gender

Building a family tree often requires defined genders for the connected members. This is largely because some relationship types are gender specific. For example, the tree visualization data uses “mother” and “father” to assign parent roles, not just “parent”. If a user tries to assign someone as their parent, and the family member’s gender is not defined, the application cannot infer whether that family member should be assigned as a mother or as a father, so the connection doesn’t complete. Some relationships will still generate correctly without defined genders, such as “spouse”.

In future developments, KinTree should establish a way to collect, store, and utilize user gender. This could be most simply implemented within the registration page by adding a gender field. Gender should also be addressed when users add non-registered family members; users should be prompted to give the gender of the family member they are adding.

Mobile Navigation Bar

Currently, the app’s navigation bar is set to disappear for small screen sizes. Ideally, a hamburger icon will be used to allow for an expandable navigation bar so that users can easily navigate the app’s pages. This transitory approach has not yet been implemented in KinTree.