1. Functional Requirements

1.1 High Priority - in scope

1. The app shall allow for on-line users to navigate a map based on real data that displays a marker at the location of various, living apple-trees. For users this should be an educational experience.
2. The app shall allow response to the user's intentional manipulation of the map. If the user intends to move the map in a given direction, or zoom in or out, the map shall respond as expected. This should allow the user to control this app in the same way as they are accustomed to with other, more common apps.
3. Each marker displayed within the app's map shall be clickable, allowing the user to learn more about the apple. This should allow the user to learn about apple trees they have specifically chosen to learn more about.
4. The system should have a database that contains all information about every apple tree collected by the Boulder Apple Tree Project. This database must be either secure, or have abstraction built into it to obscure the location of certain, private apple trees. This database will be the only means of data storage this app requires.
5. The app should be able to make calls to the database as requested by app usage.
    a. There must be a way to access all apples within a given region for the purpose of populating the map.
    b. There must be a way to access all non-private information about a given apple tree for the purpose of display of specific data to the user.
    c. There must be a method by which this database can be updated to reflect the acquisition of information by the Boulder Apple Tree Project of new trees.
6. The app shall obscure the specific location of certain apple trees that are either considered in need of protection by Boulder Apple Tree Project, or that are located on private property.
7. The app shall be usable in both mobile and desktop environments.
8. The app shall be capable of responding and fulfilling the requests of multiple users at the same time.
9. The app shall be scalable such that if traffic severely increases, database size severely increases, or serviced map area severely increases, the app's code-base itself would require little to no modification.
10. The app shall allow for a search feature in which the user can input a phrase and have relevant apples and information appear on the map.
11. The app shall allow for a filter feature such that filters can be applied and have an effect on the map in a relevant manner.

1.2 Medium Priority - stretch goals

1. The app's database shall have support for uploading or updating of authorized data such that it can be extended or modified when more data is collected about the apples.
2. The app shall allow for apples to be viewed in a 3D space, a model shall be made for some or all apples so that they can be viewed in a 3D interactive manner.
3. The app shall have an interesting application of data science applied to the apple data stored in the database.
4. The app shall allow for different maps based on historical period to appear when the timeline is scrubbed also updating the map to show only the relevant apples.

1.3 Low Priority - out of scope

1. The app shall have cross-platform support, i.e. be a native application for iOS, Android, or any other mobile platform.
2. The app shall have user credentials for log-in or authorization of specific capabilities.
3. The app shall translate web pages into different languages.

2. Non-Functional Requirements

2.1 Reliability

- The system shall be completely operational at least 50% of the time.
- Down time after a failure shall not exceed 48 hours.

2.2 Usability

- A new user should be able to use the system immediately or after reading the help section of the system.

2.3 Performance - aws or servers, code commented, etc

- The system should be able to support 100,000 simultaneous users.

2.4 Security - location of trees!

- Inputting tree data must be transmitted in encrypted form.

2.5 Supportability - documentation for when we leave the project

- The system should be able to accommodate new features without major reengineering.
- The system should be able to port to different operating systems and the various popular web browsers (Google Chrome, Firefox, Safari, etc.)

2.6 Support

- The system shall provide an integrated help section and an additional web page that explains how to navigate the system.

2.7 Purchased Components - Figma, AWS

- AWS hosting will be needed.
- Figma will be needed.

2.8 Interfaces - ex. Data cleaning with python

- The system must interface with the new PostgreSQL / PostGIS database system for tree information.