University of Colorado
Department of Computer Science
CSCI 4308

Project Definition Document (PDD)

Eden Crashers
(Senior Design Team)

Monday 16th September, 2019

Approvals

Role	Name	Affiliation	Signature	Date
Customer	Amy Dunbar-Wallis	Boulder Apple Tree Project		
Course Coordinator	Alan Paradise	CU/CSCI		

Project Customers

Name: Amy Dunbar-Wallis
Email: amy.dunbarwallis@colorado.edu
Phone: (720) 299-4532

Team Members

Name: Jason Lubrano Email: Jason.Lubrano@colorado.edu Phone: (225) 623-8965	Name: Cassidy Carpenter Email: caca8883@colorado.edu Phone: (303) 495-9245
Name: Thomas Alder Email: Thomas.Alder@colorado.edu Phone: (720) 220-9172	Name: Chakrya Ros Email: Chakrya.Ros@colorado.edu Phone: (713) 885-6424
Name: Erika Bailon Email: Erika.Bailon@colorado.edu Phone: (720) 789-4270	Name: Dennis Nguyen Email: Dennis.Nguyen@colorado.edu Phone: (303) 856-5422
Name: Sky Ruff Email: loho1010@colorado.edu Phone: (970) 222-1385	Name: Javier Ramirez Email: Javier.Ramirez@colorado.edu Phone: (719) 493-0820

1. Problem or Need

Colorado is home to a wide variety of apples. Boulder Apple Tree Project (BATP) is currently working on sharing a rich history with a vast knowledge of apples to people all around Colorado. By partnering with the Boulder Apple Tree Project, our senior design team will work on a interactive user interface to share the data that is now owned by the Boulder Apple Tree Project. Together, we can help promote appreciation of apple trees and their fruit, as well as the preservation of these trees, their genetic lineage and their connection to the history of Boulder. Products are needed to share knowledge in a meaningful, engaging, and enjoyable way.

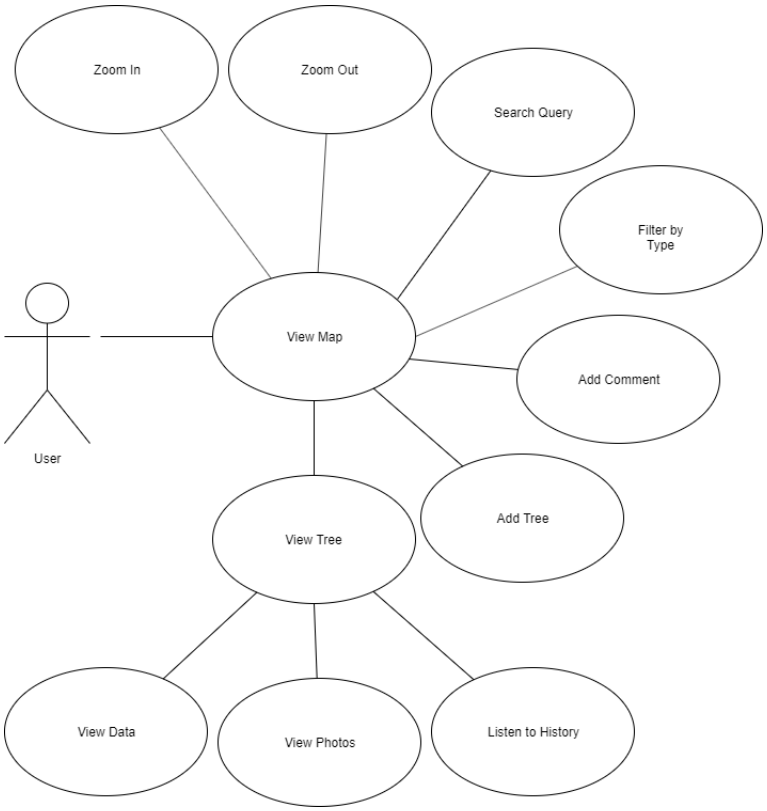

Figure 1. Use Case for user viewing map

2. Previous Work

The passionate members of the Boulder Apple Tree Project (BATP) have collected a wealth of information, including historical maps, ledgers, and tree data. They are currently making an effort to expand the knowledge base of apple trees within and around the Boulder County area through tagging trees, gathering missing data of the variety of trees and their fruits, and gaining insights into the past through oral and written histories. With this data, the BATP created simple maps representing the locations of apple trees in the past and present.

3. Specific Objectives

After meeting with Amy Dunbar-Wallis, a project MVP and scope was agreed on. For a minimum viable product, the team will build:

1. **An interactive map as the principal product.**
2. **Information about apple trees displayed on the map.**
3. **The map should be engaging and interactive.**
4. **Allows for expansion of the apple tree database.**
5. **The app should be scalable to be deployed online and in museums.**

6. **The web page and app should be compatible with mobile displays.**

7. **Locations of apple trees should not be exact in order not to compromise the tree's safety and to respect the privacy of people whose property in which a tree maybe growing.**

Although future developers will expand this web page and application in the future, the scope does not contain:

1. **Developing a mobile application for iOS, Android, or any other mobile application platform.**

2. **User credentials for log-in or authorization specific capabilities.**

4. Critical Project Elements

4.1. Database

This app will need a functional database that can contain all of the apple information that the BATP has collected. This database must be in third-normal form and sorted in away that allows for the API to access its information without any troubles.

4.2. Back-end/API

The app will need a functional API and back-end service, which can be accessed by API calls to PUT, POST, and GET data. The back-end must be able to handle numerous requests of this nature and be stable such that outages are none to extremely minimal. In the event an outage occurs, the back-end must be easily re-instantiated without too much overhead and loading time.

4.3. Front-end

The app will need a functional front-end which interacts with the back-end to present data in an educational, interactive map-based format. The front-end must follow the model-view-controller framework and be scalable such that it can handle large, increasing amounts of data about the trees and their fruit.

5. Functional Requirements

A functional requirement is statements of services the system should provide: how the system should react to particular inputs, and how the system should behave in any particular situation. For example, when a user clicks the interest point icon, information shows up. The handling for the interaction goes as such: On click, leaflet informs Django that a click action has happened. With that information, Django notifies the database for what information to grab with a key. Based on the primary key of the database, information such as historical data, genetic type, and size of the apple are fetched and sent back up to Django. Django handles that information to the AngularJS, which makes calls to the leaflet framework. Leaflet handles displaying the information to the user within the interactive app.

In order for information to be displayed about the trees and their fruit, we first need to have a PostgreSQL paired with PostGIS relational database hosted on AWS RDB. PostgreSQL is a free, open source object-relational database which stores data in tables as rows and columns. Each row of data has a primary key which will be used to access, modify, or delete a row of data. Base PostgreSQL enables for the apple and tree data of simple types such as integers, floats and strings. In order to extend the database to contain geospatial data beneficial for mapping applications, PostGIS will be also be installed. Its additions of functions, operators, and types will allow faster retrieval of data designed for mapping and provides easy export as GeoJSON files. To get the data to the back-end, the database will be queried by Django.

Django will serve as our back-end hosted on an AWS EC2 instance. Driven by Python, Django is powerful for database driven web applications allowing for easy mapping to our database layout through its object-relational mapper. Django comes with a database-abstraction API which makes the requests for data to our database simple without the need for an extra module or plug-in installation. This makes interactions between our back-end and our database extremely simple. In addition, a Python API will be built designed to handle CSV and Excel spreadsheet data as provided by the BATP. Once data is ready for the front-end, Django will pass the data to AngularJS for rendering.

For rendering the data and interactive map on the client, AngularJS will be used. Without Angular, our front-end would be basic static HTML/CSS. Essentially, no interact-ability would be available as when the data

changed in the back-end there would be no way to see that data change unless a page refresh occurred. This JavaScript framework enables dynamic web pages so that when a user clicks on a point, filters out trees, or has general interaction with the web application the page will dynamically change in response to that action. The general structure of the Angular system will be Model-View-Controller. In order for communication between the Angular front-end and Django, the Django REST (Representational state transfer) Framework (DRE) will serve as our API service between the two ends. What this means is that the front-end can request from the back-end data that is relevant for a user and receive that data from Django once the database request has been served. Ideally our architecture will be RESTful so that our web application is lightweight, fast, and simple. Angular on its own will not be able to give the interactive map our project requires, but since it is extensible the JavaScript library of leaflet will also be used.

Designed for mobile-friendly interactive maps, leaflet fits our needs perfectly. It is usable, small in size, and performs well in addition to being open source. It too can be extended so that if there is a feature that we wish to implement but leaflet does not have a possible plug-in can be available or one could be developed. Using the data received with Angular from Django via the DRE API, leaflet will be able to visualize that data on a map which can be interacted with by the user. Should the user have some action with the leaflet map that requires a dynamic change, Angular would be immediately able to request the data with the DRE API to Django which then grabs the data from database with the Django database-abstraction API in which it is handed back through Django to the Angular front-end and ultimately displayed on the map. The functional body diagram in figure 2 shows a simple system architecture that was described previously.

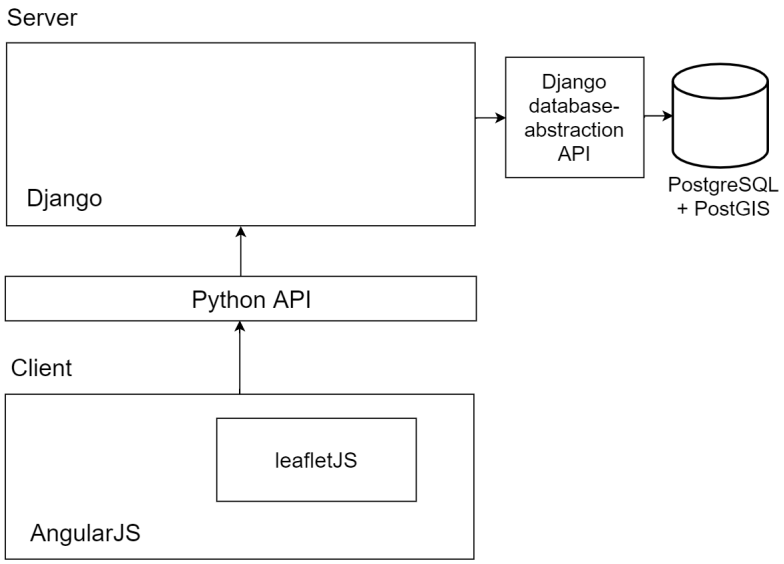

Figure 2. FBD - needs to be updated w/ Django instead of NodeJS/Express

6. Team Skills and Interests

Our team has a wide variety of skills and interest that will be useful when working on this project. Our talents and skills combined spread over the entire development stack. Some of us are back-end developers; on the other hand, others are front-end developers; whereas others are full-stack developers.

Critical Project Elements	Team member(s) and associated skills/interests
Front-end	Erika, Thomas, Chakrya, Jason, Cassidy
Back-end	Chakrya, Javier, Sky, Jason, Cassidy, Dennis, Erika
AngularJS	Sky, Jason, Erika, Chakrya
Django	Erika, Javier
PostgreSQL	Sky, Cassidy, Dennis, Jason, Chakrya
Python	Sky, Jason, Cassidy, Erika, Chakrya, Dennis, Javier, Thomas
AWS	Sky, Thomas, Jason
HTML/CSS/javascript	Sky, Jason, Cassidy, Erika, Chakrya, Dennis, Javier, Thomas

7. Resources

https://www.overleaf.com/project/5d895a1847aad0000159fda4 **Most of the software and frameworks we plan to use are open source; however, there will be some financing needed. The project requires AWS credits and financing to run our server. Additionally, the project needs data and documentation as already collected by BATP.**

Critical Project Elements	Resource/Source
financing	BATP, CU EBIO Department
data and documentation	BATP

References

[1] **Jackson, Jelliffe. "Project Definition Document (PDD)", University of Colorado–Boulder, Retrieved August 29, 2019, from https://canvas.colorado.edu/**