**Estimate of Person-Hours:**

* **Jasmine:** 8hrs
* **Thomas:** 25hrs
* **Jennifer:** Not Given
* **Caleb:** 12hrs
* **Edwin:** 5hrs
* **Combined: 50hrs**

We came up with these estimates through utilizing the Agile user story method where Projects 1, 2 and 3 represented our inventory of projects. Individually we each found our baseline for how many hours we thought it would take to complete our project and used our inventory of projects to add to this so that we could have a solid estimate. In addition to this, this method helped us really work out the fine details in what we needed to get done which helped us make our estimate even more accurate to us.

**Actual Account of Hours:**

* **Jasmine:**
  + Nov 2: Meeting to go over project 4 expectations and assign tasks (20min)
  + Nov 3: Meeting to finalize details for project 4/update those that couldn’t come to the first meeting (15min)
  + Nov 7: Messaged teammates for documentation of hours spent, explained what I needed from them and compiled that info in a concise document (2hrs)
  + Nov 8: Researched for deployment plan and maintenance and wrote out both sections (4hrs)
  + Nov 8: Formatting and writing out integration section (1hr)
  + Nov 8: Met with Thomas and messaged with him to discuss documentation, last minute defects and final game plan (20min)
  + Total: 7hrs 55min
* **Thomas:** 
  + Nov 2: Meeting to go over project 4 expectations and assign tasks (20min)
  + Nov 3: Meeting to finalize details for project 4/update those that couldn’t come to the first meeting (15min)
  + Nov 4:Looked into python testing (2hrs)
  + Nov 5: Looked into python testing (2hrs)
  + Nov 6: Looked into python testing and fixed the data structures issue preventing the recipe GUI from working (3hrs)
  + Nov 7: Worked on refactoring the timer GUI (10hrs)
  + Nov 8: Met with Jasmine and messaged with him to discuss documentation, last minute defects and final game plan (20min)
  + Nov 8: Fixed bugs and made/recorded final presentation(14hrs)
  + Total: 31 hrs 55min
* **Jennifer:**
  + Total: 0hrs
* **Caleb:**
  + Nov 1: Planning out ideas to bring to the meeting and research (4hrs)
  + Nov 2: Meeting to go over project 4 expectations and assign tasks (20min)
  + Nov 2: Integrated API in CookBook.py (4hrs 30min)
  + Nov 6: Fixed issue where recipes were not saved to the correct folder (3hrs)
  + Nov 7: Created test\_class.py to test back end functions of Recipe.py & CookBook.py and also removed redundant setter function from Recipe.py (3 hrs 30min)
  + Total: 15hrs 20min
* **Edwin:**
  + Nov 3: Meeting to finalize details for project 4/update those that couldn’t come to the first meeting (15min)
  + Nov 5: Looked up pdf functions and worked through some issues running the project on windows (3hrs)
  + Nov 6: Inputted it into the script and made sure the function worked correctly (3hrs)
  + Total: 6hrs 15min
* **Combined:**
  + **Total: 32hrs 25min**

**Defect Tracking Tool:**

**Defect #1:**

**Date Reported:** 11/4/20

**Who Reported It:** Thomas

**Brief Description of Defect:** When the timer hits zero it wraps around to be 24 hr left on the timer.

**Date Fixed:** 11/7/20

**Who Fixed It**: Thomas

**Brief Description of How it Was Fixed:** Refactored the timer GUI to simplify the way that it was handling the time calculations and then added in a check so that if it would overflow its internal time is set to zero and the timer is then paused. It will pause itself if it Is unpaused without more time being added on to it.

**Defect #2:**

**Date Reported:** 11/4/20

**Who Reported It:** Thomas

**Brief Description of Defect:** The timer should be displayed in the following format hr:min:sec with two integers shown for each: i.e. 00:00:00. However, the timer doesn’t display a zero in front of the number if it is a single digit number.

**Date Fixed:** 11/7/20

**Who Fixed It**: Thomas

**Brief Description of How it Was Fixed:** As the issue came down to how the string was being constructed the fix was relatively simple and only used python’s string formatting. (f"{t\_h:02}:{t\_m:02}:{t\_s:02}")

**Defect #3:**

**Date Reported:** 11/6/20

**Who Reported It:** Caleb

**Brief Description of Defect:** Recipes were not being saved to the correct folder

**Date Fixed:** 11/6/20

**Who Fixed It:** Caleb

**Brief Description of How it Was Fixed:** Changed the directory for saving recipes to the Windows Documents folder to make it more universal so it won’t depend on where the project is saved

**Defect #4:**

**Date Reported:** 11/6/20

**Who Reported It:** Thomas

**Brief Description of Defect:** The recipe class had the ingredients stored in an object that is used to construct part of the recipes GUI. There was a change made so that the three different parts of where then being stored in a string. This caused the recipe GUI to not be created due there no longer being the expected data structure.

**Date Fixed:** 11/7/20

**Who Fixed It**: Thomas

**Brief Description of How it Was Fixed:** Fixed the string issue by reverting the data structure back to what it had been. However when implementing the print button it was found that the pdf generation was just printing out the ingredients with the assumption that it was a formatted string. This issue that the solution had made was fixed by using the data structure to format the string and then use that to print the pdf instead

**Defect #5:**

**Date Reported:** 11/8/20

**Who Reported It:** Thomas

**Brief Description of Defect:** We weren’t allowed to chase the API’s data for more than an hour

**Date Fixed:** 11/8/20

**Who Fixed It:** Thomas

**Brief Description of How it Was Fixed:** Removed the old downloaded ones at launch and created a template so that the users can make their own recipes to save

**Defect #6:**

**Date Reported:** 11/8/20

**Who Reported It:** Thomas

**Brief Description of Defect:** Duplicate named recipes would be displayed twice

**Date Fixed:** 11/8/20

**Who Fixed It:** Thomas

**Brief Description of How it Was Fixed:** Had the application check if the name already exists in the list and if so, the application will not include it

**Defect #7:**

**Date Reported:** 11/8/20

**Who Reported It:** Thomas

**Brief Description of Defect:** When the GUI makes a request to the API it takes too long/won’t refresh the GUI while it is waiting for the response

**Date Fixed:** NA

**Who Fixed It:** NA

**Brief Description of How it Was Fixed:** Not fixed yet, but the solution would be to make that call asyncrences

**Integration of Code:**

In our team's process of creating our fourth and final project, one of the largest assets to our coding was utilizing integration from our previous prototype that was exhibited in our project three. We integrated our code through finding the pieces of our last project that worked well and needed to be built upon to expand and meet the goals of our final project. An example of this is our use of the API function. In our third project we only gave the user the access to a few select recipes to make things easier on us, but in our current project we have an entire API where the user can find theoretically any recipe they can think of. The integration technique that we used in this example and throughout the rest of our project was bottom-up integration. We utilized this because it made the most sense to us when determining how to build upon our previous project and it helps with isolating faults/finding them early on and utilizing the potentially reusable code rather than starting anew. Through utilizing this integration technique we were able to take the code from our project three prototype and shape it into our final finished web application. Along the way we were able to really examine our previous code which helped us find holes in our logic early on and gave us a different perspective than with our previous projects. Instead of a brand new entity, this project was just another building block to a greater goal.

**Deployment Plan:**

Throughout our fourth and final project, our team was able to create a web application that we were proud of and that we felt would be usable in the real world. Because of this and the way this class is designed to tie real world applications into our projects, we developed a deployment plan for getting our web application up and running. The steps that we have already taken in our project four were to first come up with an idea, create an initial design through our discussions as a team along with agreeing on a format (web application or IOS) and language to code in. In addition to this, in our project three we created the prototype for our web application. In this project we further developed our prototype by adding themes, fixing our timer GUI, adding the ability to have more than one timer, allowing the user to search for a recipe with the API and creating the grocery list function. The next steps that we would take to deploy our web application that we designed for our project four would be to clearly identify our testers or better yet, the individuals that we will want to market this app to. In addition to this, some further steps would be to eliminate bugs (which we have already done in our prototype and then final product) and identify our goals based on feedback from testers and our target audience. We could also talk with the company that runs the API we utilize in our app to form a business relationship. Finally, we would deploy the app to the live environment and upgrade it over time as we see fit.

To build on the later steps and clearly define what we would do in our situation, we will be looking at our project deployment through the lense of a young to middle aged adult as our target audience. We would have a young to middle aged adult (ages 20-40) as our target audience because older adults (those over 40) may not be as open to utilizing a web application other than google for finding a recipe and making their grocery lists, and those younger than 20 may not be cooking as often or be skilled enough to want to utilize a specific application for finding recipes. With that in mind, the cost of our web application is seemingly very small in that the only real purchase our team would need to make in developing our project four would be the domain, our API for recipes and the possible costs of marketing. Some of these costs can vary, especially the marketing cost. Registering a new domain generally costs between 10-20 dollars a year but it all depends on different packages one might choose and the registrar that is used. In addition to this, the API we are using for recipes has three levels of costs based on the amount of recipes you can access per day. If we purchased the most expensive package, it would be 79 dollars per month. The marketing cost can vary to a much higher degree. The average cost to market an application is 5,000-15,000 dollars which can include ads for the app, conventions and conferences where we could set up a booth showing off our application and more. Through these variances it is difficult to say how much deploying the application initially would cost, but it could be anywhere between 5,089 dollars and 15,099 dollars.

With these costs and extra steps to consider when it comes to actually deploying our project four application, it seems that we would most likely have to wait until after college to try and deploy our application. Although this is the case, seeing this higher level of what really goes into deploying an application has given our team much to think about in terms of this project and future projects. It also gave us an idea of what our future careers might look like in terms of the timeline and effort that will need to be put in for a project.

**Maintenance Plan:**

After all of our team's efforts to create and plan to deploy our web application from our project four, the next logical step is to look into how much it would cost to maintain our final project for the next year. As with the initial costs, our team definitely had to do a lot of higher level thinking to consider what all went into keeping a web application going. It is not just running it and leaving the web application going for forever. It takes time, effort and money to maintain a web application, even on a small scale like our project four web application.

The main costs that our team was able to come up with in terms of maintaining our project four web application after deploying it were the costs of hiring software developers, the monthly fees to get access to our API for recipes and the annual fees for keeping our domain name. We would need to hire software developers to deal with any technical issues that might arise with our applications, make edits to our web application to improve functioning and expand our web application into something greater than our group could have ever done alone. Because of how invaluable these software developers would be, they would clearly cost a fair amount of money. The average entry level software developer makes anywhere from 47,000-97,000 dollars per year and as the software developers gain more experience and notoriety, their salary can grow to even larger sums of money. We are going to estimate our cost under the assumption that we would be working with entry level software developers. In that case, if we hired three to help us out, we are looking at spending 141,000-291,000 dollars per year on our team members alone. After that, looking into the cost of maintaining our API and web applications domain looks much smaller by only costing us 79 dollars per month and between 10-20 dollars a year respectively. This would give us a total yearly cost of roughly 141,089-291,099 dollars per year to maintain the web application.

With these additional costs in mind, it definitely has shown our team how the costs seem to grow rather than shrink as a company grows. One would think that the initial costs would be the biggest hurdle, but as the project expands and grows, there comes a need for more team members and a company starts to form. These costs did not even go into the benefits that would be provided if the company grew even more and expanded into more projects later on which would both add even more costs. Based on this, one can see that there is much to consider when starting a web application and growing it into something bigger. Although there are a lot of initial costs and even more costs to maintain a web application like the one our team designed in our project four, these costs all add into the expansion and hopefully success of the initial idea, which in the end is worth it.