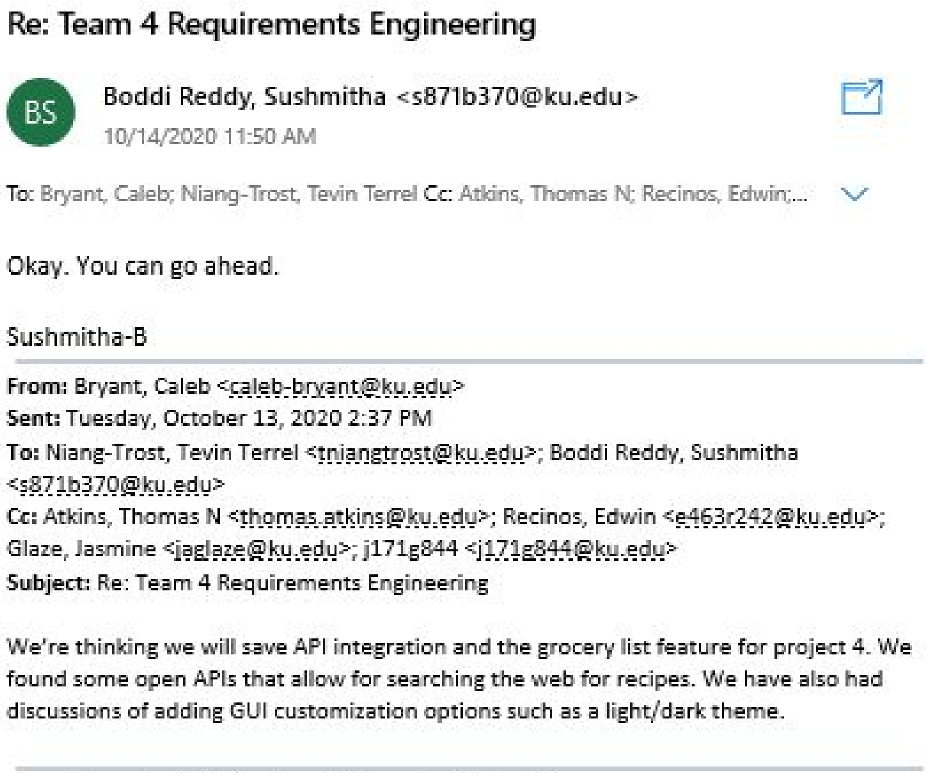
GTA Approval:

Potion of email chain with our GTAs signing off our requirements and more clear features of our prototype:



Re: Team 4 Requirements Engineering

Boddi Reddy, Sushmitha <s871b370@ku.edu>
10/14/2020 11:50 AM

To: Bryant, Caleb; Niang-Trost, Tevin Terrel Cc: Atkins, Thomas N; Recinos, Edwin;...

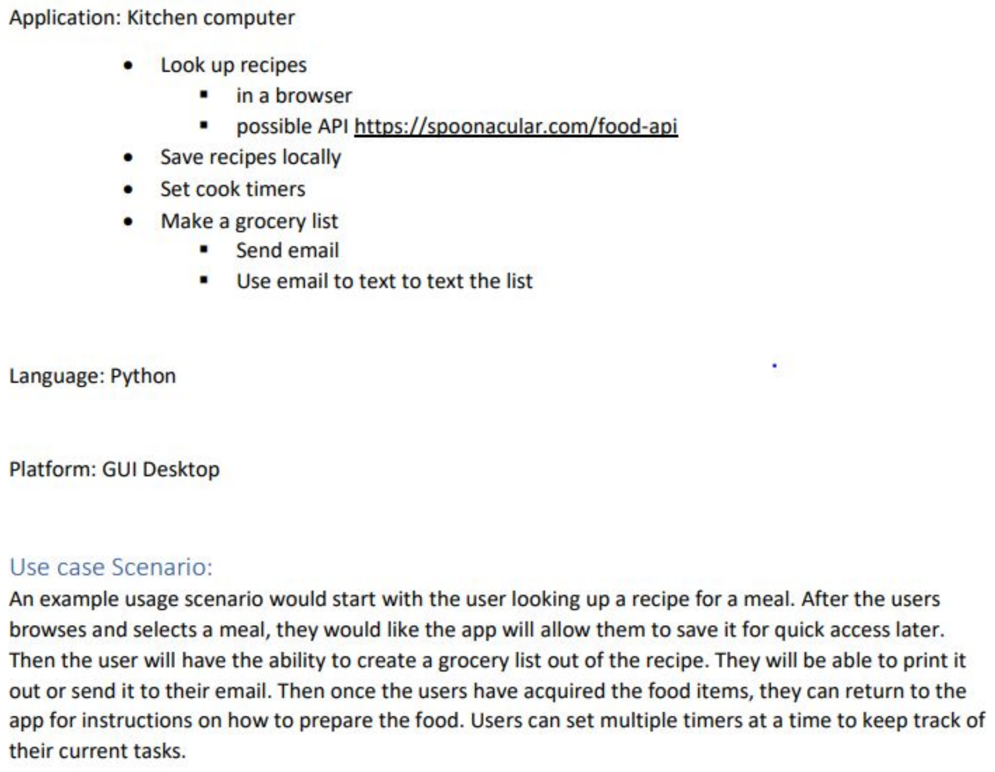
Okay. You can go ahead.

Sushmitha-B

From: Bryant, Caleb <caleb-bryant@ku.edu>
Sent: Tuesday, October 13, 2020 2:37 PM
To: Niang-Trost, Tevin Terrel <tniangtrost@ku.edu>; Boddi Reddy, Sushmitha <s871b370@ku.edu>
Cc: Atkins, Thomas N <thomas.atkins@ku.edu>; Recinos, Edwin <e463r242@ku.edu>; Glaze, Jasmine <jaglaze@ku.edu>; j171g844 <j171g844@ku.edu>
Subject: Re: Team 4 Requirements Engineering

We're thinking we will save API integration and the grocery list feature for project 4. We found some open APIs that allow for searching the web for recipes. We have also had discussions of adding GUI customization options such as a light/dark theme.

Attachment describing features of our prototype and final project:



Application: Kitchen computer

- Look up recipes
  - in a browser
  - possible API https://spoonacular.com/food-api
- Save recipes locally
- Set cook timers
- Make a grocery list
  - Send email
  - Use email to text to text the list

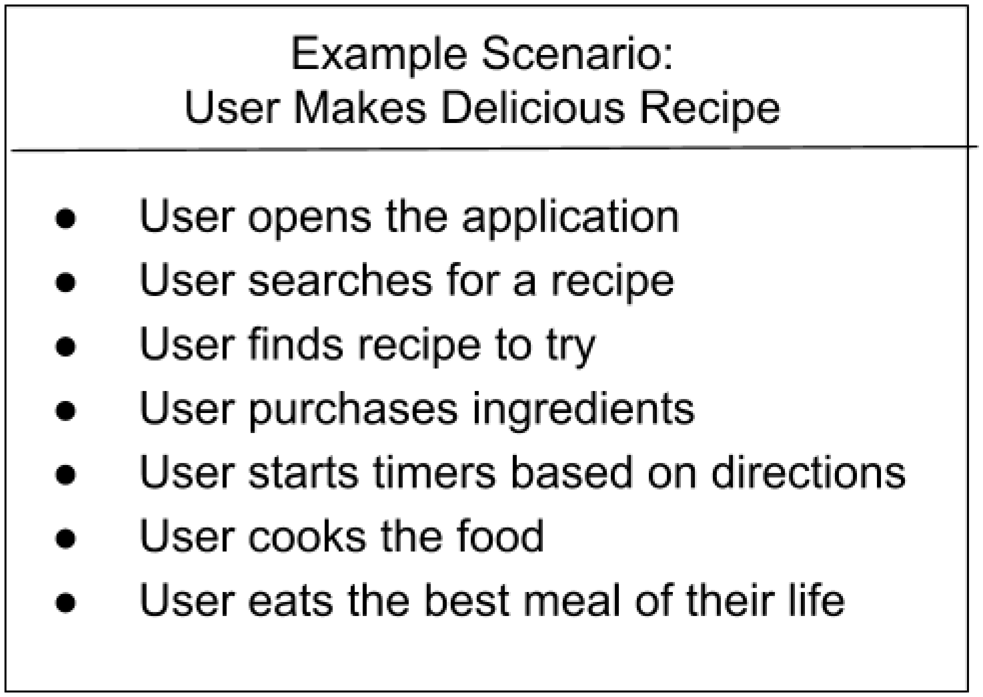
Language: Python

Platform: GUI Desktop

Use case Scenario:
An example usage scenario would start with the user looking up a recipe for a meal. After the users browses and selects a meal, they would like the app will allow them to save it for quick access later. Then the user will have the ability to create a grocery list out of the recipe. They will be able to print it out or send it to their email. Then once the users have acquired the food items, they can return to the app for instructions on how to prepare the food. Users can set multiple timers at a time to keep track of their current tasks.

Artifact:

Usage scenario:



Example Scenario:
User Makes Delicious Recipe

- User opens the application
- User searches for a recipe
- User finds recipe to try
- User purchases ingredients
- User starts timers based on directions
- User cooks the food
- User eats the best meal of their life

Estimate of Person-Hours:

- **Jasmine:** 5-6 hours
- **Thomas:** 7-11hrs
- **Jennifer:** 18hrs
- **Caleb:** 10hrs
- **Edwin:** 16-20rs
- **Combined:** 56-65hrs

We came up with these estimates through utilizing the Agile user story method where Projects 1 and 2 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:**
  - Oct 12: Brainstormed ideas for the layout of the project/what our design choices were for the project to start documentation (1hr)
  - Oct 19: Researched design processes to get background information and started a rough draft document of notes to share with the team (2hr)
  - Oct 19: Met with team to discuss findings and set some expectations for what information I need from them going forward (30min)
  - Oct 20: Created a rough draft for the paragraphs about design paradigm choice and our estimate of person-hours (30min)
  - Oct 22: Met with team to get last bits of info for the estimate of person-hours (from people who hadn't started yet) and got started tracking people that had started working on their project (20min meeting, 40min typing up/getting info)
  - Oct 23: Finalized design paradigm choice, estimate of person-hours sections and started the software architecture and design pattern sections (1hr)
  - Oct 24: Start UML diagram and continue working on design pattern section (1hr)
  - Oct 25: Finished Software Architecture section, UML diagram and design pattern section (3hr)
  - Total: 10hrs
- **Thomas:**
  - Oct 15: Went over the design of the project and assigned tasks (1hr)
  - Oct 21: Finalized what GUI to use (1hr)
  - Oct 22: Huddle meeting to see where everyone was at (30min)
  - Oct 23: Worked on reading documentation for pysimpleGUI (2hrs)
  - Oct 24-Oct 25: Made the MainGUI, RecipeGUI, and remade the Timer part in the recipeGUI file (14hrs)
  - Oct 24: Meeting to finish up project (1hr + 15min)
  - Oct 25: Got the project completely working (2hrs)
  - Total: 21hrs 45min
- **Jennifer:**
  - Oct 15: Went over the design of the project and assigned tasks (1hr)
  - Oct 19: Researching APIs, diagrams etc. (3hrs)
  - Oct 20: Researching APIs, diagrams etc. (3hrs)
  - Oct 21: Worked on code (2hrs)
  - Oct 22: Huddle meeting to see where everyone was at (30min)
  - Oct 22: Worked on code (2hrs)
  - Total: 11hrs 30min
- **Caleb:**
  - Oct 15: Went over the design of the project and assigned tasks (1hr)

- Oct 21: Planning and research (4hrs)
- Oct 22: Huddle meeting to see where everyone was at (30min)
- Oct 23: First draft of Recipe.py (6hrs)
- Oct 24: Meeting to finish up project (1hr + 15min)
- Oct 24: Finished Recipe.py and fixed CookBook.py to work with Recipe.py (4hrs)
- Total: 12hrs 45min

- **Edwin:**
    - Oct 15: Went over the design of the project and assigned tasks (1hr)
    - Oct 16: Researched how to create events, buttons, widgets and text inside windows and how to handle inputs/outputs, async clock functions and handle if the user wants to exit by googling info and watching youtube videos (2hrs)
    - Oct 17: Researched how to create events, buttons, widgets and text inside windows and how to handle inputs/outputs, async clock functions and handle if the user wants to exit by googling info and watching youtube videos (2hrs)
    - Oct 19:Researched how to create events, buttons, widgets and text inside windows and how to handle inputs/outputs, async clock functions and handle if the user wants to exit by googling info and watching youtube videos (2hrs)
    - Oct 20: Researched more of the above and started coding/manipulating the GUI further (5hrs)
    - Oct 21: Coded buttons (1hr)
    - Oct 22: Huddle meeting to see where everyone was at (30min)
    - Oct 23: Added more buttons and tried to integrate MainGUI.py with timergui.py (2hrs)
    - Oct 24: Continued to add more buttons and troubleshoot (2hrs)
    - Oct 24 Meeting to finish up project (1hr + 15min)
    - Total: 18hrs 45min
- **Combined:**
    - **Total: 74hrs 45min**

Design Paradigm Choice:

The design paradigm that we chose for our prototype for project three was object oriented design. The main steps of object oriented design are to create class diagrams, determine the format of attributes and to assign methods to relevant classes. Because a design paradigm is meant to guide the way that developers view the problem, we thought that this design paradigm would be the best to use for our project because as a team it made the most sense in how we usually work out problems. In our past projects, we have typically liked to make a diagram to work out the problem and solve the big picture issues and then determine the smaller details (attributes and methods) that make up the key elements needed to accomplish our goals.

We also decided on this design paradigm because the structure of the prototype for our project seemed to lean into this format. This seemed to be the case when we realized that our prototype has an overarching class (the graphical user interface or GUI) and many elements within the GUI such as the recipes it will be searching for and the kitchen timers that can be played with. The GUI we are designing will be our object oriented design, while the recipes and cooking timers will be the objects since they make up that larger structure. Overall, using this design paradigm seemed to work out nicely in giving us a baseline and structure to work off of and made our usual design process more streamlined and clear cut.

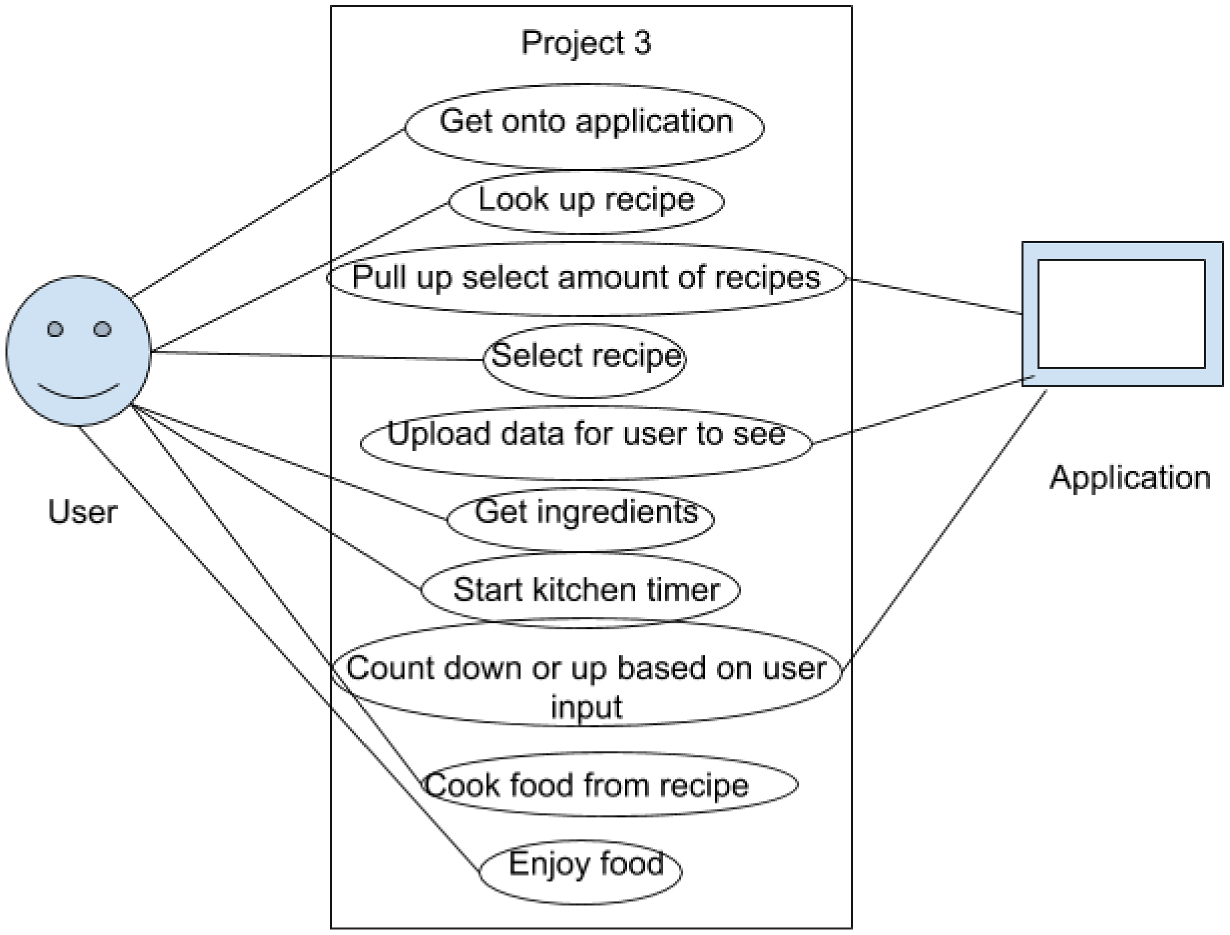
<u>**Software Architecture:**</u>

The software architecture that we chose for our prototype for project three was 3Tier. The key defining factor of a 3Tier software architecture is that it is composed of three tiers typically running as independent modules, most often on separate platforms such as: presentation, logic (a.k.a., application, business logic, or logic tier) and data. This relates to our project in that we have multiple tiers that all interconnect. The basics of our prototype are that we have an overarching class (the GUI) and many elements within the GUI such as the recipes it will be searching for and the kitchen timers that can be played with.

To get more into detail on how this will work, there will be two GUIs that we utilize (the MainGUI and the RecipeGUI). The MainGUI will be the first tier, while the RecipeGUI will be the second tier and the third tier will be the cookbook. The MainGUI will serve as the front end of the project and it will load up the cookbook by creating a cookbook object. It also spawns the RecipeGUI after the user selects which recipe that they want. It does this by passing in the RecipeGUI and one of the recipes from the cookbook. The next and final layer is the cookbook which is currently used to load in yaml files from the disk that serves as the local storage. By utilizing the 3Tier software architecture we were able to easily configure our work and make our thought process and code more clear than it has been in our past projects. This also translates into setting our team up for success in regards to our next project because everything is more organized and has a greater purpose.

UML Modeling Diagram:

Use-Case diagram:



Design Pattern:

The software design pattern that we utilized for our project three prototype was the creational design pattern: prototype. A prototype is meant to specify the kinds of objects to create using a prototypical instance, and create new objects from the 'skeleton' of an existing object, thus boosting performance and keeping memory footprints to a minimum. Throughout our project three we have quite literally been creating a prototype for our project four so the decision to use the prototype design pattern was obvious.

While working on our project three we have created a skeleton for our project four. Currently the basis of our project three is multiple GUIs that create a three tiered architecture where the user can search for a recipe from a select few that we uploaded and also utilize kitchen timers. This is the simplest version that we could figure out to make as that base level for our final project. Later on with our project four we will add in an entire API where the user can search through many more recipes and save the recipe they select locally, in addition to this, we will add in a more complicated feature where the user can create a grocery list based on that recipe and then email or text that list to someone or themselves. Overall, through creating this prototype and utilizing this design pattern we feel that we are setting ourselves up for success and thus boosting our overall performance in this project and our next one.