<Kitchen Buddy>

<Nathaniel A. Barnett>

<University of Missouri – Kansas City>

**Revision History**

|  |  |  |
| --- | --- | --- |
| **Last User to make Changes** | **Date** | **Summary of Changes** |
| N. Barnett | 9/10/17 | Initial Documentation |
|  |  |  |

[This section is optional. Revision history may also be recorded as part of version control.]

Table of Contents

[Vision Statement 4](#_Toc492837064)

[Requirements 5](#_Toc492837065)

[Step #1: Identify Categories of Users 5](#_Toc492837066)

[Step #2: Create Actor-Goal List 5](#_Toc492837067)

[Step #3: Identify User Stories 5](#_Toc492837068)

[Product Backlog 6](#_Toc492837069)

[[Step #4: Write Use Cases] 6](#_Toc492837070)

[Sprint #1 8](#_Toc492837071)

[Review 8](#_Toc492837072)

[Retrospective 8](#_Toc492837073)

[Design 10](#_Toc492837074)

[Coding Standards 11](#_Toc492837075)

# **Vision Statement**

* The Kitchen Buddy application will be able to take input from a user for a given quantity related to cooking, and convert it to any of several compatible units. For this application, quantities that will be accepted from a user are volume, weight, and temperature. Main functionality is the ability to switch between varying measurements, from ½ a Tablespoon to the corresponding number of teaspoons, and to be able to take a measurement and scale it down by a user defined factor, cups scaled back to a quarter portion. This application will not account for the exact material, for instance milk, water, or flour, when selecting a measurement and converting it to the corresponding measurement. Kitchen Buddy will allow for the functionality of taking a dry seasoning and converting from weight (oz.) to the corresponding number of a more common measurement, such as teaspoon. Application will maintain a simple and user-friendly interface that allows for quick access to the desired conversions.

# **Requirements**

## Step #1: Identify Categories of Users

**Cooks/People following recipes –** Users with enough mobile application knowledge to maneuver a logically laid out mobile interface.

## Step #2: Create Actor-Goal List

|  |  |
| --- | --- |
| **Actor** | **Goal** |
| All Users | Find pertinent starting and ending conversion measurements |
|  | Browse through available measurements |
| Cook working with volume units | Convert measurements |
|  | Scale conversions, if desired |
| Cook working with temp. | Find pertinent starting and ending temperature scales |
|  | Convert temperatures |
|  | List Recommended internal temperatures for common meats |
| Cook working with weight | Find pertinent starting and ending measurements |
|  | Convert measurements |
| **\*NOTE** | Allow for conversions between weight of dry seasonings to volume |

## Step #3: Identify User Stories

|  |
| --- |
| **Allow user to easily navigate App interface**  Estimated effort: 15 Story Points  When entering the app, users need to be able to navigate to the pertinent conversion factors. |

|  |
| --- |
| **Allow user to convert between measurements of units**  Estimated Effort: 8 Story Points  A user must be able to select the starting and ending measurements that they need. |

|  |
| --- |
| **List recommended internal temperatures of common meats**  Estimated Effort: 4 Story Points  For users working with temperatures, the recommended internal temperatures for common meats and cuts should be available. |

|  |
| --- |
| **Allow for conversions between dry seasonings (weight) and volume**  Estimated Effort: 6 Story Points  For users working with volume or weight, that need to go to the other unit when working with dry seasonings. |

### **Product Backlog**

The collection of stories makes up your project’s product backlog:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Story ID** | **Story** | **Story Points** | **Priority** | **Status** |
| S1 | Allow user to easily navigate app interface | 15 | 1 | COMPLETED |
| S2 | Allow for user to convert between measurements of the compatible units | 8 | 2 |  |
| S3 | List recommended internal temperatures | 4 | 4 |  |
| S4 | Allow for conversions between dry seasonings (weight) and volume | 6 | 3 |  |
| S5 | Store previously calculated temperatures for reference | 8 | 5 |  |
|  |  |  |  |  |
|  | STRETCH GOALS🡪 |  |  |  |
| S6 | Allow User to store Shopping list | 6 | 6 |  |
| S7 | Allow User to store Recipes | 6 | 7 |  |

## [Step #4: Write Use Cases]

[Writing use cases this semester is option. Sometimes it’s helpful to define more detailed project requirements up front.]

|  |  |
| --- | --- |
| **Title:** | App Interface |
| **Use case ID:** | UC001 |
| **Actor:** | All Users |
| **Description:** | This use case describes the system interactions needed to switch between the three unit bases used for converting measurements. |
| **Basic Flow:**   1. This use case begins when a user launches the app. 2. The system boots up, listing the functionality of the app, and allows user to immediately select starting and ending measurements, for units of volume and weight (this functionality is covered in a different story – Story ID# S2). System has menu with options that allow for Temperature Conversions as well as Recommended Internal Temperatures. 3. The user elects to select either of these options. 4. The system takes user to corresponding page, and allows for the given functionality. (Temperature Conversions and Recommended Internal Temperatures are functions covered in different stories – Story Id# S2 & S3) 5. Once under one of the secondary screens, the system will populate an “Up” arrow in the toolbar to allow the user to proceed to the previous screen. 6. The user elects to exit application, application terminates. | |
| **Alternate Flows:**  3a. If user selects Temperature Conversions.   1. Then a new screen will launch that allows for temperature conversion functionality.   3b. If user selects Recommended Internal Temperatures.   1. Then a new screen will launch that lists the recommended internal temperatures for common meats and their cuts. | |
| **Exceptions:**   1. If user selects incompatible starting and ending measurements/units to be converted, then the User will be warned and prompted to change one of the fields. | |
| **Open issues:** | |

# **Sprint #1**

Before you can plan a sprint you need a prioritized list of user stories (the product backlog). Factors to consider when prioritizing stories include [Cohn, Agile Estimating and Planning]:

1. Business value of feature
2. Cost of implementation (story points or ideal days)
3. Amount and value of new knowledge gained by developing the feature
4. Risks resolved as a consequence of implementing the feature

Sprint Backlog

|  |  |  |  |
| --- | --- | --- | --- |
| **Story ID** | **Story / Task** | **Estimated**  **Hours** | **Actual**  **Hours** |
| S1 | Design UI / research programming methods involved | 5 | 10 |
|  | Implement UI – Multiple activities and shifting between them | 4 | 6 |
|  | Add “up” and “about” buttons to toolbar | 3 | <1 |
|  | Build components for taking in user input of multiple units and measurements to be converted | 4 | - |
|  | Write automated unit tests to validate user input domain | 2 | - |
|  |  |  |  |

## Review

For this sprint, I implemented the main activity, as well as an about activity and secondary menu activity. For this sprint, I was somewhat overwhelmed by the task before me. I have never worked in the android environment or with Android Studio, and so I decided to allocate time in my sprint plan to learn this environment and hopefully make it easier to develop a plan for how to build my app. The research did greatly help me to be able to grasp what tools were at my disposal within the android environment, and I believe that the research allowed me to accomplish what I did. In short, I have the main activity for conversions of weight and volume, and the two secondary activities for temperature conversions and app info. Snapshots of app functionality are included in github repo, under folder marked “Snapshots of Functionality.”

## Retrospective

What went well? What could have gone better? What lessons did you learn? What do you plan to do differently on the next iteration?

For this iteration, I planned to complete one story; this story was not even fully completed. I spent the first week researching and playing around in android studio with small projects and “thirty minute program” guides that I found online. I have not worked in the android environment prior to this project, and I was overwhelmed by trying to design a good application base without even knowing what was at my disposal. At the start of the second week of the sprint, I was still struggling with how to open a second activity, and spoke with a classmate that helped me see where I was faltering. After speaking with this student, I was able to fix the lab #2 which originally crashed when it tried to open the second activity. By the end of the second week of the sprint, I actually began working on the project. I understood the android environment better by this point, but did not have the time remaining in the sprint to complete the tasks that I had planned at the beginning. In short, it was poor time management on my part and my lack of initial knowledge in the android environment that hindered completing this story.

**Project velocity: 0**

[Q. Why is project velocity 16 in this example? A. You planned to complete two stories but only S3 was completed. S3 was worth 16 story points, so your velocity is 16. Note, this is a simple example. You should plan for 3 or more stories each iteration.]

# **Sprint #2**

Before you can plan a sprint you need a prioritized list of user stories (the product backlog). Factors to consider when prioritizing stories include [Cohn, Agile Estimating and Planning]:

1. Business value of feature
2. Cost of implementation (story points or ideal days)
3. Amount and value of new knowledge gained by developing the feature
4. Risks resolved as a consequence of implementing the feature

Sprint Backlog

|  |  |  |  |
| --- | --- | --- | --- |
| **Story ID** | **Story / Task** | **Estimated**  **Hours** | **Actual**  **Hours** |
| S1 | Refine UI color palate/ clip art used for buttons and App | 2 | 1 |
|  | Populate Temperature Menu with conversion features | 3 | 2 |
|  | Populate About Menu with info about app and usage instructions | 3 | 2 |
| S2 | Build conversion functions for various units | 4 | 4 – NOT COMPLETE |
|  | Build User input functionality to store and convert input | 5 | - |
|  | Write automated unit tests to validate user input domain | 2 | - |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Review

* For this sprint I began with completing the story that was not fully completed in the previous sprint. I refined the UI and altered the color palate used across the app, as seen in the below figures. I added some slight implementation changes to the main conversion activity, specifically the use of spinners for starting and ending units and a large “convert” button to trigger the conversions, as seen below in figures 1 and 2. As I was working on the conversion functions for going between the different units, I found a flaw in my implementation and use of multiple dropdown style spinners. This took me several hours to resolve in a manner that I believe will function properly. In the sprint backlog, under “Story 2 – task 1,” I have marked 4 hours for the time spent trying to resolve the issue with my use of spinners. After the time spent in troubleshooting and correcting the spinner implementation, I decided to move back to Story 1 and try to accomplish the rest of that whole story before moving on. I then proceeded to populate the temperature conversion activity with the appropriate dropdown spinners and data fields to input and display conversions, shown in figure 4. Finally, I populated the “about” activity with instructions for how to use the Kitchen Buddy app, as well as developer name, shown in figure 3.

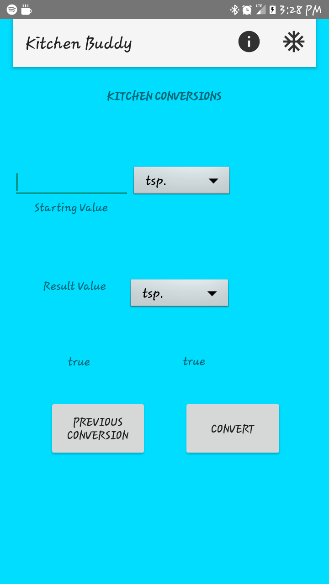
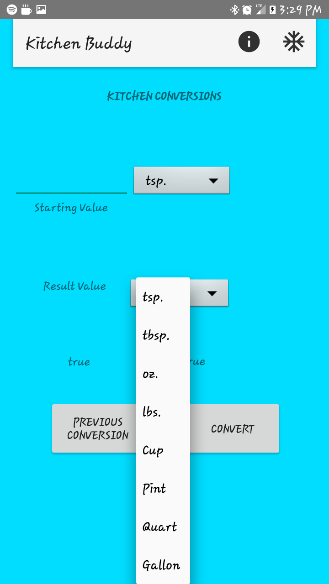
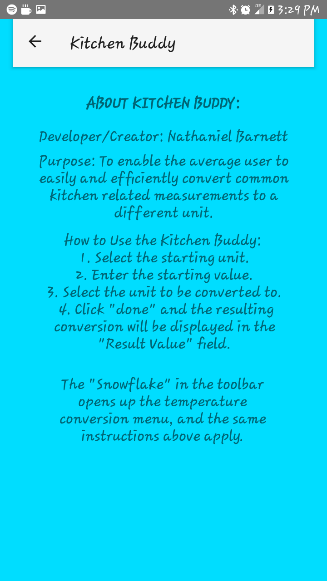
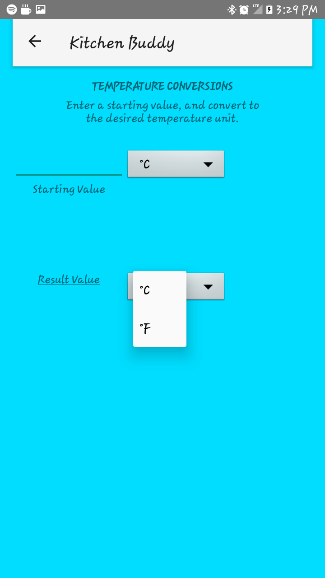
   

Fig. 1 Fig. 2 Fig. 3 Fig. 4

## Retrospective

What went well? What could have gone better? What lessons did you learn? What do you plan to do differently on the next iteration?

* For this sprint, I felt like I had a little better understanding of how all the various resource files, XML files, and java files relate to each other. I had planned to complete 2 stories, and only one was completed. The second story was uncompleted for a few reasons, however, mainly due to error on my part. I began to jump around on the tasks that were in my sprint backlog before individual stories were completed. This led to time being spent in ways that were simply inefficient. As mentioned in the review, I ran into issues with my implementation and use of multiple dropdown spinners to select various data fields, and this in turn led to several hours being spent troubleshooting. After resolving the spinner implementation, I returned to the remaining tasks associated with Story #1, and completed it. Despite the hours spent on the spinner implementation and beginning to code the conversion functions, this work did not add to the velocity of this sprint. Going forward, I intend to stick as close to the sprint backlog as possible, and try to use that structure as a means of organizing my efforts rather than guidelines for where to start. I also plan to research and get a better grasp on various android elements such as spinners before trying to implement them without a clear understanding of how they are structured and implemented.

**Project velocity: 15**

[Q. Why is project velocity 16 in this example? A. You planned to complete two stories but only S3 was completed. S3 was worth 16 story points, so your velocity is 16. Note, this is a simple example. You should plan for 3 or more stories each iteration.]

# **Sprint #3**

Before you can plan a sprint you need a prioritized list of user stories (the product backlog). Factors to consider when prioritizing stories include [Cohn, Agile Estimating and Planning]:

1. Business value of feature
2. Cost of implementation (story points or ideal days)
3. Amount and value of new knowledge gained by developing the feature
4. Risks resolved as a consequence of implementing the feature

Sprint Backlog

|  |  |  |  |
| --- | --- | --- | --- |
| **Story ID** | **Story / Task** | **Estimated**  **Hours** | **Actual**  **Hours** |
| S2 | Build conversion functions for volume conversions | 6 | 8 |
|  | Build User input functionality to store and convert input | 2 | 2 |
|  | Build Conversion functions for temperature conversions | 4 | 3 |
| S3 | Internal temperatures for common meats (pork, beef, chicken) | 2 |  |
|  |  |  |  |
|  |  |  |  |

## Review

## Retrospective

What went well? What could have gone better? What lessons did you learn? What do you plan to do differently on the next iteration?

**Project velocity:**

[Q. Why is project velocity 16 in this example? A. You planned to complete two stories but only S3 was completed. S3 was worth 16 story points, so your velocity is 16. Note, this is a simple example. You should plan for 3 or more stories each iteration.]

# **Design**

[What is the overall structure of the solution? What are the major modules of code? What are the dynamics of communication between these modules? The most common way of depicting this information is with static and dynamic models augmented with short narrative descriptions of design.]

Modules of Code:

1. Interface
2. Unit conversion
   1. Temperature conversion – will be implemented as a separate activity.
   2. Weight and Volume conversion – implemented as a single activity with the user able to select the starting units and ending units.
3. Recommended Internal Temperatures

Recommended Internal Temperatures: A list of data to be displayed.

Unit Conversions: Methods involved in converting measurements of weight or volume.

UI: Layout of application and organization of activities.

# **Coding Standards**

* Function names will begin with a lower-case letter, and the following “words” will begin with a capitalized letter. Example: addMore()
* Private variables will begin with the prefix “m”, and then the following “words” will be capitalized. Example: private int mBallCount
* Use standard brace style
* Use spaces for indentation
* Use Javadoc Standard Comments