Software Requirements Document for Pocket Chef

Author: Group 10

Nate Howe, Bryant Baltes, Scott Hood, Tyler Cook

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Change** |
| 0.1 | 09/13/07 | SM | Initial Document |
| 0.2 | 2/27/14 | NH | Filled out first page, some of page 1, and some of page 2, and use cases. |
| 0.3 | 3/1/14 | NH/TC | Both worked on filling out the rest of the document, except or use case. |
| 0.4 | 3/1/14 | TC | Added in two more use cases. |
| 0.5 | 3/2/14 | TC | Added use case diagram and terms/definitions. |

Table of Contents

1 Introduction 3

1.1 Purpose 3

1.2 Scope 3

1.3 Definitions, acronymns, abbreviations 3

1.4 References 3

1.5 Overview 3

2 Overall Description 4

2.1 Product Perspective 4

2.2 Product functions 5

2.3 User characteristics 5

2.4 Constraints 5

2.5 Assumptions and Dependencies 5

3 Specific Requirements 6

3.1 External Interface Requirements 6

3.2 FEATURES 6

3.3 Performance requirements 6

3.4 Design Constraints 6

3.5 Software System Attributes 6

3.6 Other Requirements 6

# Introduction

## Purpose

The purpose of the SRS is to develop and understand customer requirements.

Scope

The scope of the SRS is to focus on details of user-system interactions and detailed requirements.

## Definitions, acronymns, abbreviations

// alphabetical list of terms and their descriptions

// This is part of analysis and you must make sure you describe terms used in this document

|  |  |
| --- | --- |
| Term | Description |
| Admin | A higher power user; moderates the system. |
| Guest | An actor that has not registered with the system. |
| Item/Food | The class/object that represents an ingredient or grocery item. (can of beans, bananas) |
| User | Either the class/object used in code, or referring to the User actor. |

## References

None for now.

## Overview

[OMIT]

# Overall Description

Pocket Chef is a grocery management system. In essence, Pocket Chef will allow users to manage their groceries easily (using a UPC scanner), and find recipes to cook. Based on the groceries a user has, our application will suggest recipes to the user. Users will also be able to search for specific recipes matching selected items in their inventory, or just browse recipes in general. Pocket Chef will also track favorite recipes, your created recipes, grocery lists, and other useful information to provide an all-in-one grocery solution.

## Product Perspective

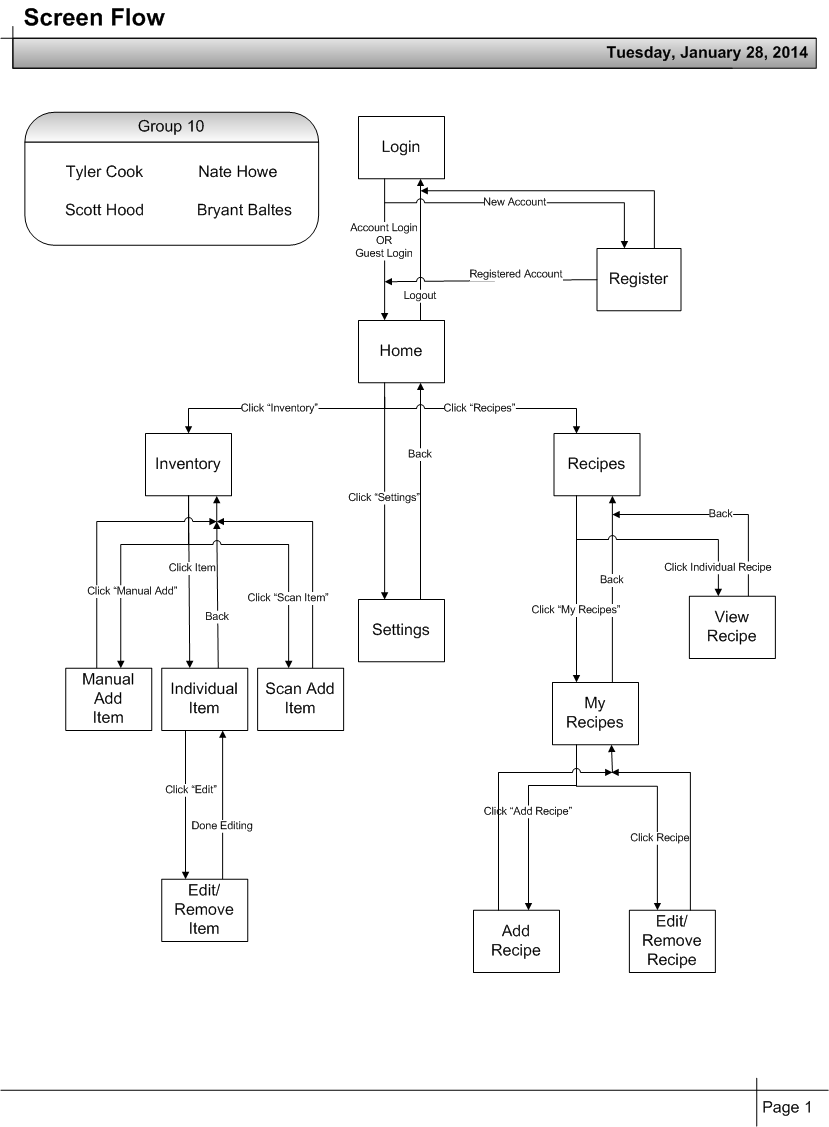
Similar products have been made, but few use UPC decoding, and also lack an enhanced recipe system (most are just a basic recipe browse system). Also, other applications tend to focus more on grocery lists, calorie counting, or creating health/fitness goals, while our application is designed to focus on grocery management.

### Concept of Operations

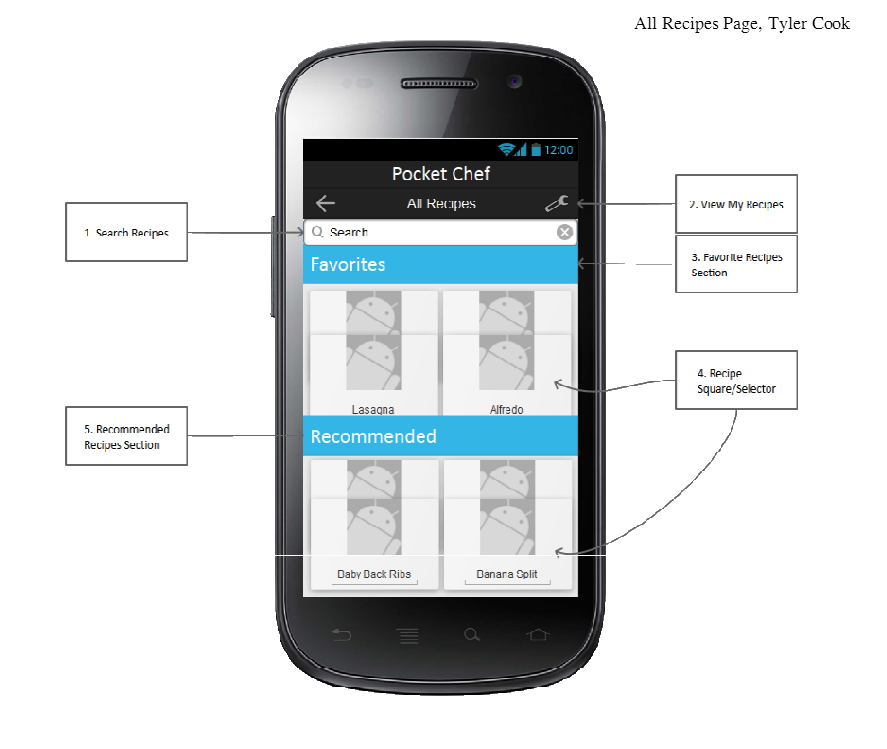
Pocket Chef will operate as an online application on the Android platform. Users will be able to download the application from the Play Store, and launch it from their Android devices. Our libraries include JDBC, ZBar (for scanning UPC’s), and Apache Commons for string manipulation.

Our system will connect to a MySQL database using the JDBC library and stored procedures. The database will contain user information. Upon login, the application retrieves an inventory, recipes, and other information unique to the logged in user. The scanner module uses the ZBar library to retrieve scanned barcodes from the Android camera. JSON is used to query a large, online UPC database to decode the UPC. Searching and suggesting will be based on a series of algorithms yet to be developed.

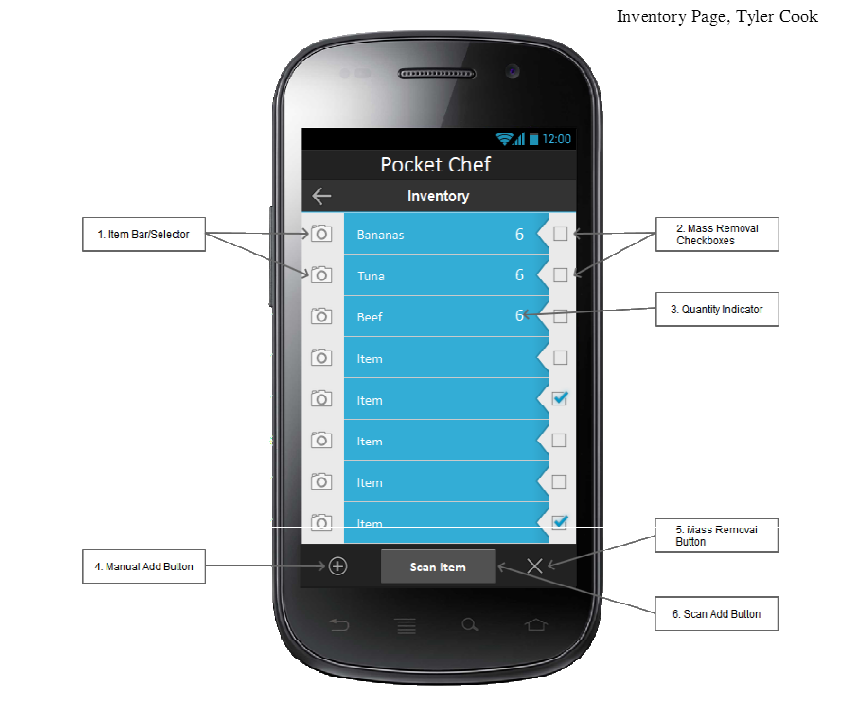
### Major User Interfaces



#### Example Screenshots



### Hardware Interfaces



Touch screen via Android phones and tablets.

### Software Interfaces

// example: CGI-URL or function signatures etc (OMIT for now).

### Communication Interfaces

// example: modem etc (OMIT for now)

### Memory Constraints

// RAM, and other storage constraints (OMIT for now)

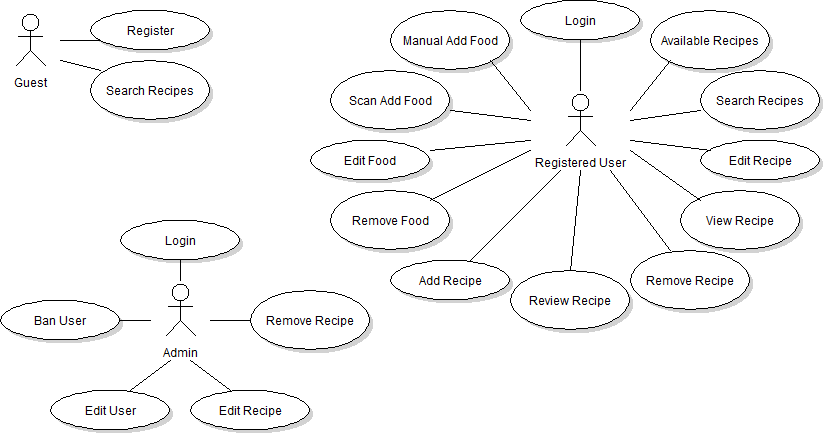
### Operations

// special operations (if any) (OMIT for now)

### Site Adaptation Requirements

//ex: Japanese language etc (OMIT for now)

## Product functions



There will be three actors: two more significant than the third. A registered user will have the most features and use cases. A guest will be able to become a registered user and search for recipes. An admin will act as the police, enforcing proper recipes and dealing with unwanted issues.

### UC-1 Scan Add Food (Nate Howe)

User uses the application to scan a barcode.

Actors: Guest, Registered User

Main Success Scenario:

1. User scans barcode.
2. Barcode is retrieved from item.
3. Barcode is decoded into item information.
4. Result fills text field for the user to edit.

Extensions:

1a. Camera is unavailable

1a1. The barcode is never scanned, and a display notifies the user of the problem.

2a. Internet access offline.

2a1. Barcode cannot be decoded, and the problem is displayed.

3a. Barcode cannot be decoded

3a1. Notify the user that the item could not be found.

2.2.2 UC-2 Manual Add Food (Nate Howe)

User manually enters item information and adds the item to their inventory.

Actors: Guest, Registered User

Main Success Scenario:

1. User enters item information.
2. User adds item.
3. System creates item with given information.
4. Item is added to the user inventory.
5. Item is saved for persistence.

Extensions:

1a. User selects expiration date for the item

1a1. System creates item with this information.

2a. User selects quantity for the item

2a1. System creates item with this information.

3a. User selects to add information with scanner

3a1. Scanner use case

### 2.2.3 UC-3 Remove Item (Bryant Baltes)

A user uses this case to remove a food item from his inventory.

Actors: Guest, Registered User

Pre-Condition: There is an item to remove in the inventory.

Main Success Scenario:

1. User navigates to the item inventory page.
2. User selects item(s) to be removed and presses remove.
3. The user is asked if they are sure they want to remove the item(s).
4. The item is removed from the system and the database.
5. Page refreshes and item is removed from the screen.

Extensions:

1a. Quantity is more than one.

1a1. The user must select how many items of that type he wants to remove.

2a. Internet access offline.

2a1. Item cannot be removed from the database.

3a. Quantity is one

3a1. Item is removed completely from the database.

4a. There is no recipe to remove

4a1. The option will not be available to select remove.

2.2.4 UC-4 Remove Recipe (Bryant Baltes)

User removes a recipe from their “My Recipes”.

Actors: Guest, Registered User

Pre-Condition: There are recipes to be removed.

Main Success Scenario:

1. User navigates to the “My Recipe” page.
2. User selects the recipe to be removed and presses remove.
3. The user is asked if they are sure they want to remove the item(s).
4. The item is removed from the system and the database.
5. Page refreshes and item is removed from the screen.

Extensions:

1a. Internet access offline.

1a1. Item cannot be removed from the database.

2a. There is no recipe to remove

2a1. The option will not be available to select remove.

2.2.5 UC-5 Add Recipe (Scott Hood)

User adds a recipe from their “My Recipes”.

Actors: Guests, Registered Users

Pre-Condition: None.

Main Success Scenario:

1. User Navigates to “My Recipe” page.
2. User selects the “Add Recipe” button on the page.
3. The user is taken to a separate activity where they input, recipe name, description, instructions and ingredients.
4. The user then presses the “save” button.
5. It takes them back to the “My Recipe” page where the page refreshes with their updated list of recipes.

Extensions:

1a. Internet access offline.

1a1. Item cannot be added from the database.

2a. The name of this recipe already exists in your recipes.

2a1. Recipe is not created, system notifies actor.

3a. There a fields missing that need to be added.

3a1. The recipe is not created, system notifies user.

2.2.6 UC-6 Edit Recipe (Scott Hood)

User edits an existing recipe from their “My Recipes”.

Actors: Guests, Registered Users

Pre-Condition: Must have recipes in database.

Main Success Scenario:

1. User Navigates to “My Recipe” page.
2. User selects recipe to be edited and presses edit.
3. The user is taken to an activity where they can change information of the selected recipe.
4. User makes changes, and then clicks save.
5. User is returned to previous page “My Recipe”, recipe changes are saved within the selected recipe.

Extensions:

1a. Internet access Offline

1a1. Recipe cannot be edited, could not retrieve from database.

2a. There is no recipe to edit.

2a1. The option will not be available to select edit.

3a. Text fields were left blank.

3a1. The recipe will not change and the user will be notified of fields that are blank.

2.2.7 UC-7 Login (Tyler Cook)

An actor starts the application and must log in to begin using features.

Primary Actor: Admin, Registered User

Pre-Condition: Application installed?

Main Success: 1) Actor provides username and password.

2) Actor presses login button.

3) System retrieves user info from database and constructs User object.

4) System displays ‘All Recipes’ page to actor.

Extensions: 1a) The username or password field is left blank.

1a1) The system returns an error message box notifying the user.

2a) There is no wifi connection.

2a1) The system returns an error notifying the user.

3a) The credentials are not found in the database.

3a1) The system notifies the user of invalid credentials.

2.2.8 UC-8 Register (Tyler Cook)

An actor with no account must register to use features.

Primary Actor: Guest

Pre-Condition: Application installed?

Main Success: 1) Actor provides username, password, first and last name

2) Actor presses register button

3) System creates a new User and sends the info to the database

4) System informs actor of success, returns to the login page

Extensions: 1a) One or more of the credential fields are blank.

1a1) The User is not created; system returns an error notification to the user.

2a) There is no wifi connection.

2a1) The User is not created; system notifies the actor.

3a) There is already a User with the same credentials.

3a1) The User is not created; system notifies the actor.

## User characteristics

Adult grocery shoppers. Frequency of use will depend on how often the user decides to use the recipe search/suggestion system, and how often they wish to update their food inventory.

## Constraints

Pocket Chef requires an internet connection. The user must be able to operate a mobile device. For scanning, users must by items with barcodes. Some items will likely not be found in the UPC database when scanned.

## Assumptions and Dependencies

Assumes the user has an Android device updated to a somewhat recent API.

# Specific Requirements

// Here you need to put in details (if any). Mark items [None] if you do not have any information.

## External Interface Requirements

### User Interfaces

### Hardware Interfaces

### Software Interfaces

### Communications Interfaces

## FEATURES

### FEATURE-1 ….

#### …

#### …..

## Performance requirements

## Design Constraints

## Software System Attributes

### Reliability

### Availability

### Security

### Maintainability

### Portability

## Other Requirements

// ADD Appendices (if any)

// Regenerate Table of Contents