BCGL: Barcode Grocery List

Software Requirements
Specification (SRS) Document

BCGLSoftwareRequirementsSpecification.doc

Final Draft

February 3, 2021

RAID Inc.

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Rough Draft: Version 1	Bailey Martin, Andrew Moore, Josh Davidson, Kyle Hieb	*BCGLSoftwareRequirementsSpecification* provides details on RAID Inc's proposed Barcode Grocery List (BCGL).	1/27/21
Final Draft	Bailey Martin, Andrew Moore, Josh Davidson, Kyle Hieb	*BCGLSoftwareRequirementsSpecification* provides details on RAID Inc's proposed Barcode Grocery List (BCGL).	2/3/21

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to provide a detailed overview of the development process for our product, named BCGL. It will highlight some of the essential functional and non-functional requirements which will be implemented in the creation of the product. It also serves as a guide and overall illustration of the product design and user interface. It is intended to be read by the project development team as well as any clients, stakeholders, and project supervisors that would have input and feedback in the software design process.

1.2 Description

The product that we will be designing is called BCGL, which stands for Barcode Grocery List. It will allow shoppers to create a shopping list on their phone, mobile device, or computer by scanning the barcode of the item that they intend to buy. Users will also have the ability to look up the SKU number of the product if they are not able to utilize the barcode scanner feature. There will be an option to create more than one shopping list. An example of this would be to create a grocery list and a Christmas list using the same account. Using BCGL account credentials, the user will be able to access their lists from multiple devices. The software will also display general product information for each product that is found.

This product will not inform the user if a product is in stock in a specific store. It will not process payment, and will not allow for curbside pickup directly on the app. This app is solely for personal organization, so that the user knows which items they need to buy.

This product will be designed to be used by everyday consumers. Anyone who goes shopping will find it useful, as it will localize their lists to one place, allowing for easy additions and deletions from the list. Our goal is to make this app a helpful part of the everyday lives of consumers.

1.3 Overview

Moving forward, this document will provide the reader with a further in-depth description of BCGL. It will highlight the requirements necessary for the development of the app, including functional and non-functional requirements, as well as domain and user interface ideas. This document includes a Table of Contents and an Index to provide easy navigation.

1.4 Glossary of Terms

Agile: Software life-cycle model that stresses the importance of customer communication throughout the entire development process.

API: *Application Programming Interface*. An API is used to provide a resource of data and functionality to a software development project. Developers can utilize calls and requests to creatively implement the data that the API contains into their specific project.

BCGL: *Bar Code Grocery List*. This is the product that is being created.

Git/GitHub: Source code management for storing and viewing developed code.

Kanban: Japanese production task scheduling system that often utilizes a Kanban board to organize project tasks.

MVC Framework: *Model View Controller*. This software development approach segments the code for the application's model, view, and controller into three distinct partitions. This helps developers stay organized and allows for universal readability.

RAID Inc.: Company developing the BCGL software.

SKU: *Stock Keeping Unit*. This is the number on a barcode, which is unique to each product.

SLCM: *Software Life Cycle Model*. A term used in the software engineering community to describe the approach utilized in a software development project.

UPC: *Universal Product Code*. This is the number displayed on a barcode, which is unique to each product.

Xamarin: Cross-platform mobile application development infrastructure.

1.5 Future Business Potential

When this project is complete, it will be uploaded to the Apple App Store for use by the general public. We are open to potential contracts in the future, as well as any acquisition offers that might be made.

2 Overall Description

2.1 Product Perspective

BCGL will be an attempt to refine and expand upon similar applications that may already be in existence. It will rely on tools such as the Barcode API (see Section 3.3.3) to gather product information but will otherwise be self-contained and independent of other products. The BCGL software is intended for use by individuals who are seeking an easier way to manage their weekly grocery shopping lists.

2.1.1 System Interfaces

Mobile application: This will likely be the main interface. It will allow users to access their mobile device's camera to scan barcodes. It will also be what users take with them when they go to the store to physically purchase the items on their list. Users will be able to create, edit, and view their list using this interface.

Web application: This will allow users to create, edit, and view their lists, but will likely not include an option to scan the barcode itself. Instead, it will allow for SKU and product look-up via a search feature. It will serve as a companion to the mobile app, with limited functionality.

Internal user information database: This is where the account credentials for each user will be securely stored. It will also include individual shopping lists for each user.

Product database/Barcode API: This is the external database where product information, including barcodes and SKU numbers will be stored and referenced.

2.1.2 Memory Constraints

Due to the nature of this application, estimated memory consumption is negligible, because the application is primarily non-resource intensive.

2.1.3 Operations

Upon first launching the app, the user will be required to grant access to the device's camera, as well as to enable push notification functionality that may be implemented throughout the development of this app. Also, before the user can begin to make their first list, they must create a free account by entering an email address and password. Backup and recovery operations are not required by the user because any changes made will be automatically uploaded to an online database. Users will need a stable Internet connection in order to process changes to the internal database, as well as to access the barcode API.

2.2 Product Functions

BCGL will provide a convenient means of scanning a product's barcode and adding it to a shopping list. The list will then be accessible via an internal database stored on the user's mobile device. This will allow users to edit their lists while shopping. Users will have the option to view additional information for each product that is scanned and placed onto the shopping list.

2.3 Similar Systems

There currently exists a mobile application named Pantry Check. This product allows a person to scan the barcode of the product and add it to their shopping list. It then sends push notifications notifying the user when their food is going to expire. However, the user interface of the system is not user-friendly, and only allows a single grocery list to be made.

This appears to be the only existing application that BCGL would have to compete with, at least on this type of scale. We plan to differentiate ourselves by providing a simpler user interface and by allowing users to create multiple lists. We will also implement better marketing, which will allow a greater number of people to learn about our product.

2.4 User Characteristics

Our intended user is moderately tech savvy. This user demographic owns smartphones, uses several apps as a part of their daily lifestyle, and knows how to access the Internet in addition to their phone's camera. Due to this required level of technical knowledge, our intended user base will likely be members of a younger generation (ages 16-50 years old).

The user's education level is not important, but they likely will be knowledgeable and comfortable with the use of their smartphones. They are regular consumers. They are familiar with shopping and are also accustomed to popular store layouts.

2.5 User Objectives

Users can expect the BCGL product to provide the following features:

- Easy-to-use interface
- Large repository of products in the external database
- Quick functionality and responsiveness
- Built-in help guide and tutorial
- A reliable and innovative platform for storing grocery shopping lists

2.6 Constraints

Our software system will be primarily constrained by the external database of products, accessible through the public Barcode API. The number of products stored in the database, as well as the database's included functionalities, will have a direct effect on our BCGL software.

Additionally, the RAID Inc. team does not have access to a physical Android device. Therefore, all testing of the Android app will have to be completed using Xamarin's device emulators. They should be accurate but are not as reliable as testing the software on a physical device.

Due to the limited time window for the project, RAID Inc. may have to eliminate or modify certain requirements in order to produce a final product by the deadline.

Lastly, the program will likely require an Internet connection in order to be fully functional.

3 Specific Requirements

3.1 Functional Requirements

Priority Scale: Low (1) – Medium (2) – High (3)

1. Low: Items that can be eliminated should the need arise, without adversely affecting the product. These items are not urgent and not as important to the final product.
2. Medium: Items that are desired by the customer and/or users of the system, but that may be postponed until a future release. These items are not urgent and but are important parts of the final product.
3. High: Items that are mission critical and without which the system cannot function in a manner that is satisfactory to the customer. These items are urgently needed and important to the success of the final product.

Functional Requirement 1

- Feature Name: Barcode Scanner
- Feature Description: Users can access their camera from the mobile app in order to scan the barcode on items. Items can then be added to the shopping list via the barcode.
- Priority: High (3)
- Technical Issues:
    - Pre-Conditions: Users will need to have enabled camera access for the BCGL app in order to scan a barcode.
    - Post-Conditions: A stable Internet connection (via Wi-Fi or Cellular Data) will be required to look up the barcode in the system database (Functional Requirement 2).
    - Other Technical Concerns: Lighting and the surrounding environment while scanning a barcode may influence the overall responsiveness and accuracy of the barcode scanner.
- Risks: The user will not be able to easily add items to their shopping list, defeating the primary purpose of the software.
- Dependencies on Other Requirements: Functional Requirement 2

Functional Requirement 2

- Feature Name: Product Barcode Database
- Feature Description: Rather than manually entering product barcodes and information into the database, we will utilize an existing database distributed as a public API. After users scan a product's barcode, the BCGL system will reference this API to find the associated product. This app must be able to access that database and find the desired item almost immediately, in order to add items to the list in a user-friendly manner.
- Priority: High (3)

- Technical Issues:
  - Pre-Conditions: A stable Internet connection (via Wi-Fi or Cellular Data) will be required to access the system's product database.
  - Post-Conditions: N/A
  - Other Technical Concerns: Some local disk storage may need to be available on users' mobile devices to download or cache portions of the database. RAID Inc. will continue to investigate this throughout the development process.
- Risks: Users will have a limited selection of products without the successful implementation of this API and database. This is a key component of our software and will directly influence RAID Inc.'s overall level of success.
- Dependencies on Other Requirements: None

Functional Requirement 3

- Feature Name: Manual Product Search Functionality
- Feature Description: Users will also be given the option to search for items instead of scanning the product's UPC barcode. This will be provided via a search bar integrated within the mobile app. Users will then be able to add products to their shopping list from the search results page.
- Priority: Medium (2)
- Technical Issues:
  - Pre-Conditions: A stable Internet connection (via Wi-Fi or Cellular Data) will be required to access the system's product database.
  - Post-Conditions: N/A
  - Other Technical Concerns: Users may experience difficulty finding their desired item if they do not enter a brand name with the item name, as the search will likely query the database based on the product name and manufacturer.
- Risks: Failure to implement this feature means that users will be unable to manually search for items. If they are using a device without a camera or are having trouble scanning a product's barcode, they will not be able to add the item to their shopping list.
- Dependencies on Other Requirements: Functional Requirement 2

Functional Requirement 4

- Feature Name: Sign-In Capabilities
- Feature Description: Users can sign into their mobile app, web-app, and BCGL account using an email address and password. This allows them to view their current shopping list and any scanned items on a device that is signed in with their credentials to the BCGL system. Their shopping lists will be stored in the cloud on BCGL infrastructure, ensuring availability on any Internet-connected device.
- Priority: Medium (2)
- Technical Issues:
  - Pre-Conditions: A stable Internet connection (via Wi-Fi or Cellular Data) will be required to access the BCGL system. Users will have to ensure that their device and platform meet the minimum operating requirements, as specified in *Section 3.3-System Requirements*.

    - Post-Conditions: The BCGL system will need to compare the user's entered credentials to those stored on the BCGL infrastructure. If they match, the user will be granted access to their account. If the credential sets do not match, the user will be alerted with an "Invalid Credentials" message.
    - Other Technical Concerns: Users will need to have an option to sign out, and have the system automatically sign them out after a period of inactivity.
- Risks: Failure to provide users with BCGL accounts forces them to only being able to create, modify, and view their shopping list on a single device. The shopping list would be saved locally on the device.
- Dependencies on Other Requirements: Functional Requirement 2

Functional Requirement 5
- Feature Name: Additional Product Detail
- Feature Description: Items can be inspected, and the user can view additional details about an item by selecting it. This information may include product pictures, nutritional information, prices at nearby stores, etc. This additional information will be extracted from the product database referenced in Functional Requirement 2.
- Priority: Low (1)
- Technical Issues:
    - Pre-Conditions: A stable Internet connection (via Wi-Fi or Cellular Data) will be required to access the system's product database.
    - Post-Conditions: N/A
    - Other Technical Concerns: Some local disk storage may need to be available on users' mobile devices to download or cache portions of the database. RAID Inc. will continue to investigate this throughout the development process.
- Risks: Users will not be able to view any additional product information from their shopping list. Without the successful implementation of this feature, users will only be able to view the name of a product on their shopping list and will have to utilize other platforms to gather additional information.
- Dependencies on Other Requirements: Functional Requirement 2

Functional Requirement 6
- Feature Name: Companion Web App
- Feature Description: BCGL will offer a companion web application that can be used in tandem with the mobile app. This will allow users to view their lists from any device, including devices of different platforms. They will access their saved grocery lists by authenticating with their BCGL account.
- Priority: Low (1)
- Technical Issues:
    - Pre-Conditions: A stable Internet connection (via Wi-Fi or Cellular Data) will be required to access the BCGL system. Users will have to ensure that their device and platform meet the minimum operating requirements, as specified in *Section 3.3-System Requirements*.
    - Post-Conditions: A stable Internet connection (via Wi-Fi or Cellular Data) will be required to sync changes to the BCGL system. Users will have to

    ensure that their device and platform meet the minimum operating requirements, as specified in *Section 3.3-System Requirements*.
- Other Technical Concerns: Ensuring the web application works in a variety of modern web browsers (Chrome, Edge, Firefox, Safari) on various operating systems (Windows 10, macOS, Linux, iOS, Android).
- Risks: Users will only be able to access, view, and modify their shopping lists from a mobile app on an iOS or Android device.
- Dependencies on Other Requirements: Functional Requirement 4

Functional Requirement 7

- Feature Name: Multiple List Types
- Feature Description: Users will have the ability to make different types of lists, based on the types of items they wish to place on the list. An example of this would be a standard weekly grocery list and a list in preparation for an upcoming holiday dinner.
- Priority: Low (1)
- Technical Issues:
    - Pre-Conditions: Users will need to access the product database, via the barcode scanner or the search bar functionality. Additionally, a stable Internet connection (via Wi-Fi or Cellular Data) will be required to access the BCGL system. Users will have to ensure that their device and platform meet the minimum operating requirements, as specified in *Section 3.3-System Requirements*.
    - Post-Conditions: Changes made to any list will need to be synchronized to the BCGL servers.
    - Other Technical Concerns: Users need to maintain an Internet connection while working in multiple lists to ensure that their changes are accurately saved.
- Risks: Users will only be able to have one list, and therefore can only place products on a single shopping list. Users will have to continually modify their list to ensure it is accurate and up to date.
- Dependencies on Other Requirements: Functional Requirements 1, 2, 3

Functional Requirement 8

- Feature Name: Price-Checking Capabilities
- Feature Description: Barcode Grocery List will be able to display the prices of a specific item at competing retailers. This will be integrated into the "additional product information" view. This feature will help consumers easily find the best price on an item, similar to Google Shopping or other online price-shopping tools.
- Priority: Low (1)
- Technical Issues:
    - Pre-Conditions: Users will need to access the product database, via the barcode scanner or the search bar functionality. Additionally, a stable Internet connection (via Wi-Fi or Cellular Data) will be required to access the BCGL system. Users will have to ensure that their device and platform meet the minimum operating requirements, as specified in *Section 3.3-System Requirements*.

    - Post-Conditions: After the item's prices at multiple retailers are displayed, the user can select the store or retailer from whom they plan to purchase the item. The retailer's name can then be stored in the shopping list underneath the item name, making it easy to identify which items will be purchased from which stores.
    - Other Technical Concerns: Accuracy of price-checking capabilities will need to be further investigated. RAID Inc. will test this during the software development process to ensure this is a good feature to include.
- Risks: Users would seek out other platforms and services that offer price shopping features.
- Dependencies on Other Requirements: Functional Requirement 2 and 5

Functional Requirement 9
- Feature Name: Biometric Authentication
- Feature Description: iOS users would be given the option be able to authenticate and sign into their BCGL accounts using their device's biometrics interfaces (ex: Touch ID and Face ID). This is a convenience feature, as users can use their fingerprint or facial recognition to authenticate into the BCGL mobile app.
- Priority: Low (1)
- Technical Issues:
    - Pre-Conditions: Users will need to have biometrics enabled on their iOS device. Additionally, users will have to sign into the mobile app with their credentials at least once, before the operating system will ask if they would like to utilize Touch ID or Face ID to sign into the app in the future.
    - Post-Conditions: A stable Internet connection (via Wi-Fi or Cellular Data) will be required to authenticate into and access the BCGL system.
    - Other Technical Concerns: A password reset option should be included on the login screen of the mobile app in case users forget their passwords and cannot gain access to their account using their biometrics.
- Risks: Users will have to remember their credentials in order to sign in to the BCGL iOS app. This may also be less secure, as they would be authenticating using an email address and password instead of a fingerprint or facial recognition technology.
- Dependencies on Other Requirements: Functional Requirement 4

3.2 User Interface Requirements

3.2.1 User Interface: Graphical (GUI) or Command-Line (CLI)

There will be a number of features associated with the graphical user interface (GUI). The following bullet points describe the general layout of the application:
- There will be a navigation bar at the bottom of the application that allows for feature navigation within the app. This will likely house the camera, search, and list view features, as well as any other features that might be crucial to the use of the app. However, it is important that this bar be kept clean and easy to navigate, so there should not be too many features placed on it.

- There will be a camera icon on the navigation bar, which will access the camera function that is used to scan barcodes so that products can be identified and added to the shopping lists.
- The navigation bar will also feature a search icon, which leads the user to the search function.
- There will be a login page/screen that displays username/email and password fields upon startup. This will allow users to enter their credentials in order to log in to the system.
- On startup, there will be a screen that includes our company logo, as well as a button that takes users to the page to sign in or create a new account.
- There will be a "Table view" that displays grocery lists. This will be the third function that is featured on the navigation bar.
- The app will have a screen for creating a new account. It will be accessible via the "Create New Account" option located near the username/password fields on the initial startup screen.
- This new account screen will display EULA/terms of use during account creation for the end user to view and accept.

3.2.2 Application Programming Interface (API)

There will not be an API for this application. However, we will be utilizing an API to integrate a publicly available barcode and product database. This idea will be elaborated upon in the System Requirements section.

3.2.3 Diagnostics (Error Reporting and Usage Logs)

Users will be instantly made aware of credential errors when logging in, and also when searching for product numbers using the search feature of the application. Also, if the barcode is not found within the database API, an error message will be sent to the user, as well as to the developers. The development team will be alerted via email and then are responsible to manually add that product into the database.

3.3 System Requirements

3.3.1 Hardware Interfaces

- Camera access (mobile device app)
- Internet connection
  - Wireless (Wi-Fi) on mobile devices and some computers
  - Wired Ethernet on desktop computers
- Biometrics (mobile app)
  - Touch ID and Face ID (iOS only)
- Supported Devices:
  - iOS Devices (iOS 12.0 and higher)
  - Android Devices (Android Oreo 8.0 or higher)
  - Windows/Mac/Linux

    - Chrome: Version 87.0.4280 or higher
    - Firefox: Version 68 or higher
    - Edge: Version 83 or higher
    - Safari: Version 12.1.2 or higher
- Peripherals
    - Mouse and keyboard for PC users utilizing the companion web application

3.3.2 Communications Interfaces

- The user will need an Internet connection for the application to communicate with the BCGL infrastructure. The app will communicate with the barcode API to look up items. In addition, the app will communicate to the backend database where the user's shopping list data is stored. This communications relationship allows for the development of the web app, as it will pull the user's data from the backend user shopping list database.

3.3.3 Software Interfaces

- DBMS for Barcodes – Barcode Lookup API
    - Reference to Documentation: Barcode Lookup on Rapid API (https://rapidapi.com/barcodelookup/api/barcode-lookup)

3.4 Domain Requirements/Constraints

3.4.1 User Information Security

All information related to user accounts including usernames and passwords will be handled in a secure manner and shall remain protected.

3.4.2 User Consent

Users will be presented with a EULA/Privacy Policy that they must agree to in order to create an account for use with RAID Inc. software.

3.5 Non-Functional Requirements

3.5.1 Reliability

Software will be reliable, and function fully as intended if there is a provided Internet connection. Otherwise, the software with work with limited functionality. This would include but is not limited to read-only access to existing shopping lists on a mobile device. Therefore, an Internet connection is required for modification of a list. Software will run properly, as long as the web database servers are operational.

3.5.2 Availability

The application will be available to anyone who can access a smartphone, or Internet-capable device, and is able to be connected to the Internet.

3.5.3 Security

Since the software will interact with and contain user input data, security is a high concern. The following practices will be used, and privacy notices will be given.

- Camera authorization
- Hidden passwords
- Data integrity checks
- Updated services for database and API to evade viruses and potential threats
- Database injection protection

3.5.4 Maintainability

This program will be developed using an MVC framework for flexibility and for future development in mind, as future developers should easily understand the code design.

3.5.5 Scalability

The application will be able to run on a variety of platforms, including:

- Mobile
  - iOS
  - Android
- Desktop Operating Systems (accessible via a web browser listed below)
  - Windows
  - Mac
  - Linux
- Modern Web Browsers (*See Section 3.3.1 for minimum version requirements*)
  - Chrome
  - Firefox
  - Safari

3.5.6 Usability

The application will be easy to use and navigate so that people of all ages (though specifically ages 16 to 50 years old) may use it. The application will also have a fast response time when performing actions such as editing shopping lists or scanning product barcodes.

3.6 Logical Database Requirements

3.6.1 Product Information

Product information and associated barcodes will be stored and accessed via a database. This database will also likely contain images and descriptions for each product.

3.6.2 User Account Information and Credentials

Users will need to create an account in order to use the application. Email addresses and/or usernames and passwords will securely be stored in a RAID Inc. database. This will be referenced during authentication to verify user credentials.

3.6.3 User Created Data

Within the application, users are able to create lists of data. These lists can be modified, as items can be added to or removed from the lists. Shopping lists created by users will be stored in a database, so that they may access the lists from any device when signed into the application with their personal credentials.

4 Software Life Cycle Model

4.1 Choice of Software Life Cycle Model

We plan on using the Kanban Agile Method as our Software Life Cycle Model. However, instead of conducting a daily stand-up, we will perform a stand-up at every meeting. We will have weekly stand-ups as our worst-case scenario. However, they will likely take place more frequently, as we will meet as often as necessary. We will also utilize a portion of the Design-to-Schedule Software Life Cycle Model, as we will prioritize our requirements and deliver a final product by the due date of the project (April 26, 2021).

4.2 Justification for Choice of Model

The reason for choosing Kanban is because of the board design and its associated features. The primary reason for wanting to utilize a Kanban board is that it helps to keep teams organized. We will be able to look at it and easily identify which tasks need to be completed, and in what order. The prioritization of tasks based on their column placement on the board makes it visually helpful. This allows us to immediately determine the overall status of the project. Also, being able to manually "pull" items through the Kanban development process will help us visually follow the entire software engineering process. A standard Kanban board features three columns. The columns are traditionally named To Do, In Progress, and Completed. This will help our group visually stay grounded and focused from an organizational perspective. Furthermore, the board possesses a work-in-progress limitation on the number of cards or tasks that we can have in a column at any time. This limit will help our team from becoming overwhelmed and eliminates potential bottlenecks. This work-in-progress limit forces us to successfully complete assigned tasks before moving on and attempting other tasks.

We chose the agile approach because it focuses mainly on customer and user experience, instead of the developer experience. We are not designing this product for the developers' satisfaction; we are designing it for the client and project stakeholders. The agile method reflects this philosophy. Also, agile projects are reactive, rather than predictive. This is important because we cannot predict the changes that will be made to the external barcode database in the future. Therefore, the program must be dynamic and react, or adapt, to those changes as they are made.

5 Index