Deliverable #2 – High-Level Architectural Design Document

Chen, Arthur Campbell, Christopher Endrizzi, Johnny Coovert, Mitchell
Gill, Surinder Dhadda, Terin

March 7, 2016

Contents

1 Introduction **3**
1.1 Purpose . . . 3
1.2 System Description . . . 3
1.3 Overview . . . 3

2 Use Case Diagram **4**

3 Analysis Class Diagram **6**

4 Architectural Design **6**
4.1 System Architecture . . . 6
4.2 Subsystems . . . 6

5 Class Responsibility Collaboration (CRC) Cards **6**

A Division of Labour **8**

List of Figures

1 Use Case Diagram for the BEER'D Application . . . 4

List of Tables

1 Contributions and Signatures of Team Members . . . 8

1 Introduction

The following section provides a brief overview of the entire document.

1.1 Purpose

The purpose of this document is to lay out the high level architectural design of the "BEER'D" application. It will first give a description of the system and a general overview of what it is for, how it is expected to be used, and why it is being developed. It also contains information about the variety of use cases for the application, an analysis class diagram, a breakdown of the intended architecture design, and finally a class responsibility collaboration breakdown. This document is intended primarily for the developers of the application, the professor, and the teaching assistants.

1.2 System Description

The "BEER'D" system is a mobile application that aims to solve the question: "What beer is this?" This application is primarily being developed as a project for the third year Software Architecture class (course code SE 3A04) taught at McMaster University. A team of 6 students will design, develop, and create the application.

The "BEER'D" application will take specific inputs from a user. Based on these inputs, varying "experts" will attempt to analyse and come up with their best prediction (based on data provided by publicly available API's) as to which beer the inputs may be identifying. The application will return and display a list of possible answers in a forum. Within this forum, users will also be able to share their answers on popular social media networks or find local stores which sell the beers referred to in the answers - based on their current location in an map.

1.3 Overview

The rest of the document is split up into four main sections:

- The first section, Use Case Diagram, will contain each use case associated with the application.
- The second section, Analysis Class Diagram, will contain the analysis class diagram for the application based upon the use case diagram.
- The third section, Architectural Design, will provide an overview of the overall architectural design for the application. It will first identify and provide reasoning for the chosen software architecture. Then, it explain the division of the system into subsystems and describe each subsystem.
- The fourth and final section, Class Responsibility Collaboration, will contain the "CRC Cards" of the application.

2 Use Case Diagram

The following section provides a use case diagram for the application.

Figure 1: Use Case Diagram for the BEER'D Application

Home Page The Home Page use case will generally be the one encountered the most, due to the nature of mobile applications. When a user starts the application, they will be directed to the home page, where they will begin interacting with the application.

Request General Information When a customer is curious and wishes to obtain general information about a beer, such as the different types, tastes, colors, etc. they can open this page which will display such information.

Input Information The Input Information use case is included in the Home Page use case. The user will be asked to select several different inputs for the experts to analyse.

Review Previous Searches When the user wishes to review their previous searches (i.e. the results to their inputs from searches they had in the past) they will be able to view them.

Delete Search The user deletes a a search in their search history.

Re-Search The user searches again from a specific search in their history, using the same input data saved in the search. This use case will use the Access Information Controller to perform the search.

Access Information Controller This use case is an abstract use case. It is used by many other use cases. It uses the Input Information use case. It coordinates the use cases of taking input, consulting experts to analyse the input, accessing the database, and displaying the result to the forum.

Consult Experts When the user has input their information for a desired search, the Access Information Controller will use this use case to analyse the inputs. This is where the experts will predict which beer the user's inputs are describing. It will use up to 3 experts to come up with the prediction. It will then return this information back to the controller.

Expert 1/2/3 Analysis These use cases will analyse the user's input that is specific to them, respectively. It will then come up with their best prediction(s) based on the input.

Forum When the search has been completed and the application is ready to display it's predictions, it will display them in this use case. The Forum use case uses the results from the Access Information Controller use case.

Share on Social Media When the user wants to share their results on social media, they log in (or sync their social media accounts, encrypted by the application) and post them.

Open Nearby Locations When the application displays it's predicted results, it will display a map based on the user's current location and mark nearby beer retailers which sell the beers listed in the results.

Edit Expert This use case includes one of the primary functional requirements. The developer may deem it necessary to swap, add, or remove an existing expert. This use case extends Consult Experts since any modifications to the experts must be considered when consulting experts. In other words, the application should be using the must recently existing experts.

Update Beer Database When a new beer comes to the market, the developer may need to update the database to include information about this beer for the experts to be able to analyse and include in their predictions. This extends the Access Information Controller because any additions (or modifications in general) must be accessible to the controller.

3 Analysis Class Diagram

This section should provide an analysis class diagram for your application.

4 Architectural Design

This section should provide an overview of the overall architectural design of your application. You overall architecture should show the division of the system into subsystems with high cohesion and low coupling.

4.1 System Architecture

a) Identify and explain the overall architecture of your system

b) Be sure to clearly state the name of the architecture

c) Provide the reasoning and justification of the choice

d) Provide a structural architecture diagram showing the relationship among the subsystems (if appropriate)

4.2 Subsystems

a) Provide a brief description of each subsystem. Be sure to document its purpose and relationship to other subsystems.

5 Class Responsibility Collaboration (CRC) Cards

Class Name: Experts	
Responsibility:	**Collaborators:**
Knows user input	
Handles beer prediction output	
Knows AccessInfoController	
Knows beer information	AccessinfoController

Class Name: ForumPage	
Responsibility:	**Collaborators:**
Knows user current location	
Knows AccessInfoController	
Knows LCBO and Beerstore locations	NearLocIdentifier
Knows beers chosen by experts	AccessInfoController
Displays Map	
Handles click event for "Facebook", "Twitter", and "Instagram" buttons	Social/MediaShare

Class Name: SocialMediaShare	
Responsibility:	**Collaborators:**
Knows user message input	
Checks message word limit	
Knows image input	
Knows AccessInfoController	
Knows media account information	AccessInfoController

Class Name: NearLocIdentifier	
Responsibility:	**Collaborators:**
Knows user location	
Knows Beer Store and LCBO locations	AccessInfoController

Class Name: InputCheck	
Responsibility:	**Collaborators:**
Receive input from homepage	Homepage
Receive input from ReSearch	ReSearch
Ensure proper input format	InputException
Analyze input for keywords	
Sort keywords into array	
Return array of keywords	AccessInfoController

Class Name: InputException	
Responsibility:	**Collaborators:**
Receive input error	InputCheck
Display error	

Class Name: AccessInfoController	
Responsibility:	**Collaborators:**
Access keyword array	InputCheck
Search API information	APIWrapper
Request API Updates	UpdateAPI
Return keywords to experts	Experts
Receive expert information	Experts
Reference expert information with API	APIWrapper, Experts
Select up to three possible beverage options, placed into an array	
Send Array of options to ForumPage	ForumPage

Class Name: APIWrapper	
Responsibility:	**Collaborators:**
Receive access request	AccessInfoController
Receive update	UpdateAPI
Return API request information	AccessInfoController

Class Name: UpdateWrapper	
Responsibility:	**Collaborators:**
Receive update information	AccessInfoController
Access update information	
Sends update	

A Division of Labour

Team Member	Student Number	Contribution	Signature
Arthur Chen	1306616	CRC Cards: Experts, ForumPage, SocialMediaShare, NearLocIdentifier	
Christopher Campbell	1143732	CRC Cards: InputCheck, InputException, AccessInfoController, APIWrapper, UpdateWrapper	
Johnny Endrizzi	1310603		
Mitchell Coovert	1306701		
Surinder Gill	1308896		
Terin Dhadda	1312555	Title, TOC, Introduction, Use Case Diagram	

Table 1: Contributions and Signatures of Team Members