soBAR

SFWRENG 2XB3

Software Engineering Practice and Experience: Binding Theory to Practice

Department of Computing and Software

McMaster University

Design Specifications Document

Lab Section 03: Group 04

Version 1.0

12 April 2020

William Lee, *leew33@mcmaster.ca*

Waseef Nayeem, *nayeemw@mcmaster.ca*

Justina Srebrnjak, *srebrnjj@mcmaster.ca*

Michelle Domagala-Tang, *domagalm@mcmaster.ca*

Yasmine Jolly, *jollyy@mcmaster.com*

Revision History

Date	Version	Description	Author
04/06/2020	1.0	Initial Revision	William Lee, Waseef Nayeem, Justina Srebrnjak, Michelle Domagala-Tang, Yasmine Jolly
04/09/2020	1.0	Added UML Diagrams	William Lee, Waseef Nayeem, Justina Srebrnjak, Michelle Domagala-Tang, Yasmine Jolly
04/12/2020	1.0	Completed Design Document Specification	Yasmine Jolly

By virtue of submitting this document we electronically sign and date that the work being submitted by all the individuals in the group is their exclusive work as a group and we consent to make available the application developed through SE-2XB3 project, the reports, presentations, and assignments (not including my name and student number) for future teaching purposes.

William Lee	William Lee
Waseef Nayeem	[illegible]
Justina Srebrnjak	[illegible]
Michelle Domagala-Tang	[illegible]
Yasmine Jolly	[illegible]

Contribution

Name	Role	Contributions	Comments
Will	Backend developer	- Graphing algorithms - Shortest Hamiltonian path - Sample output - Assisted with geolocation and map display for front end	None
Waseef	Backend developer	- Communications system - Sorted database - Made model and controller of backend	None
Yasmine	Communications specialist	- Completed the SDS - Worked on all documents and presentations	None
Michelle	Frontend developer	- Contributed to SDS - Developed frontend	None
Justina	Frontend developer	- Contributed to SRS - Developed frontend	None

Executive Summary

The soBar mobile application is an application built for the iOS and Android operating systems built as a guide for the users convenience as to get them to various bars efficiently while ensuring their preferences are kept in mind. The objective of this application is to map out an efficient route of a user specified number of bars all while keeping the user selected preferences in mind. The motivation behind this application was the lack of applications on the market that are targeted towards this problem. It is very difficult for everyone from tourists to permanent residents to find the most time efficient route between various locations. Besides the problem of mapping out an efficient route, there is the additional problem of catering towards certain preferences such as only wanting to go to bars that have food. The soBar application takes both of these problems and sorts through a database for the user.

Table Of Contents

Detailed Description of Components

1.1 Back-end Classes and Modules

Each class is broken down with a description including an explanation of why we have decomposed the application into the given classes.

1.1.1 Business Class Module

Description:
The Business class is a basic class that holds the information about a given business like a bar. It stores its name, location information, its review star value and position given in longitude and latitude. This module will be used to create a shortest path and to pinpoint its location on a Google map. This class is a basic building block that creates the business object that all of our other modules will need and because there is so much information associated with one bar, it was ideal to make an object that would hold all this information.

Uses : N/A

Routine name	In	Out	Exceptions
New Business	String, $\mathbb{R}$, $\mathbb{R}$		None
lon		$\mathbb{R}$	None
lat		$\mathbb{R}$	None
name		String	None
address		String	None
city		String	None
state		String	None
postalcode		String	None
categories		String	None
stars		$\mathbb{R}$	None

State Variables
Lon : ℝ
Lat : ℝ
Name : String
Stars : ℝ
Categories : String
postalCode : String
State : String
City address : String

1.1.2 Shortest Hamiltonian Path Module

Description:
A hamiltonian path is a way of connecting all the vertices in a graph without using the same vertex twice (one path, no branches). This module helps find the shortest one by generating all the found hamiltonian paths then finding the shortest out of those. This is used to generate the path the user will be taking to get from bar to bar which we need according to our project specifications.

Uses: Business, Graph, Edge

Routine name	In	Out	Exceptions
new SHP	Business, Graph		None
distTo	Business, Business	ℝ	None
findShortest	Seq of Seq of Business	Seq of Business	None
permutate	Seq of Business	Seq of Seq of Business	None
validPermutation	Seq of Business, Graph	Boolean	None

State Variables
Paths : Seq of Seq of Business
G : Graph
Marked: Map <Business to a Boolean>

1.1.3 Edge Module

Description: The edge module creates an edge object for the Graph class to use where both of the two ends is one bar. Each edge must be assigned a weight and in the case of our project, it is the distance between each of the bars.

Uses: Bar

Routine name	In	Out	Exceptions
New Edge	Business, Business, $\mathbb{R}$		None
weight		$\mathbb{R}$	None
start		Business	None
end		Business	None
compareTo	Edge	$\mathbb{Z}$	None

State Variables
V : Business
W : Business
Weight : $\mathbb{R}$

1.1.4 Graph Module

Description: The graph module is a recreation of the graph data type that consists of vertices and edges. Said graph is used as a recreation of a map with the vertices as locations and the edge weights being the distance between them.

Uses: Bar, Edge

Routine name	In	Out	Exceptions
addVertex	Business		None
addEdge	Business, Business, $\mathbb{R}$		None

getBars		Seq of Business	None
adj		Map <Business to a seq of Edge>	None
getV		ℤ	None
getE		ℤ	None

State Variables
adj : Map <Business to a seq of Edge>
E : ℤ
V : ℤ

1.1.4 Business Controller Module

Description: The Business controller module is used to return the necessary information back to the front end to display. After the user puts in the starting location and potentially, additional preferences, this module will filter based on said specifications and return a sequence.

Uses: Business, Graph

Routine name	In	Out	Exceptions
New BusinessController	BusinessRepository		
all		Seq of Business	
getNearby	String, ℤ	Seq of Business	InvalidParamFormatException
getNearbyWithPrefs	String, ℤ, Seq of String	Seq of Business	InvalidParamFormatException
getPath	String, Seq of Long	Seq of Long	InvalidParamFormatException
one	Long	Business	BusinessNotFoundException

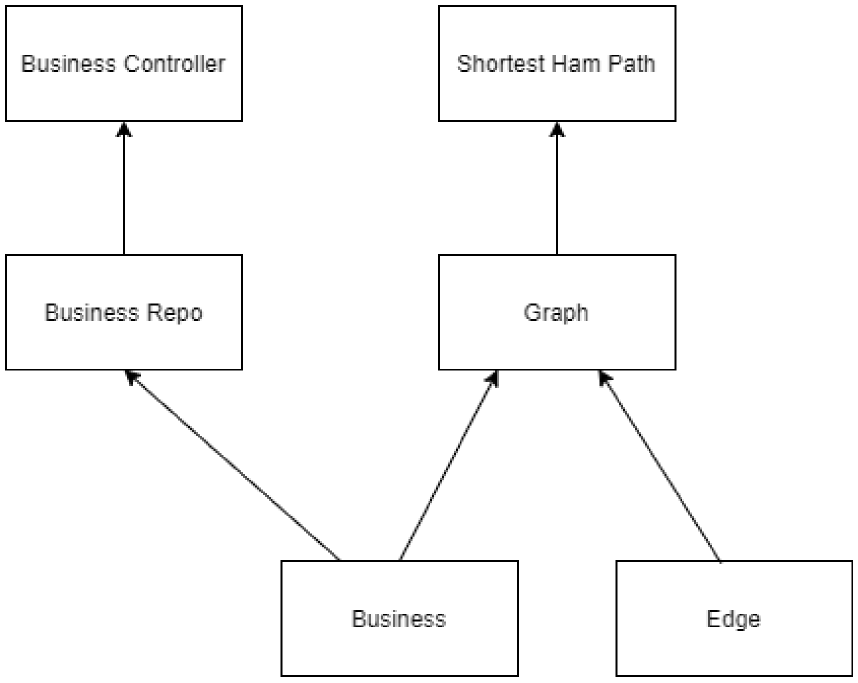
State Variables
Repository : BusinessRepository

1.2 Breakdown of Classes and Modules

The design splits up the application into modules based on back-end and front-end development. The modules were chosen to simplify implementation of the code. For example, a graph module and business module was implemented to allow for the separation and easy access of information. Likewise, the front-end is composed of separate modules depending on what screen is displayed. The relationship between these modules and classes can be shown in the UML (Unified Modeling Language) diagrams below.

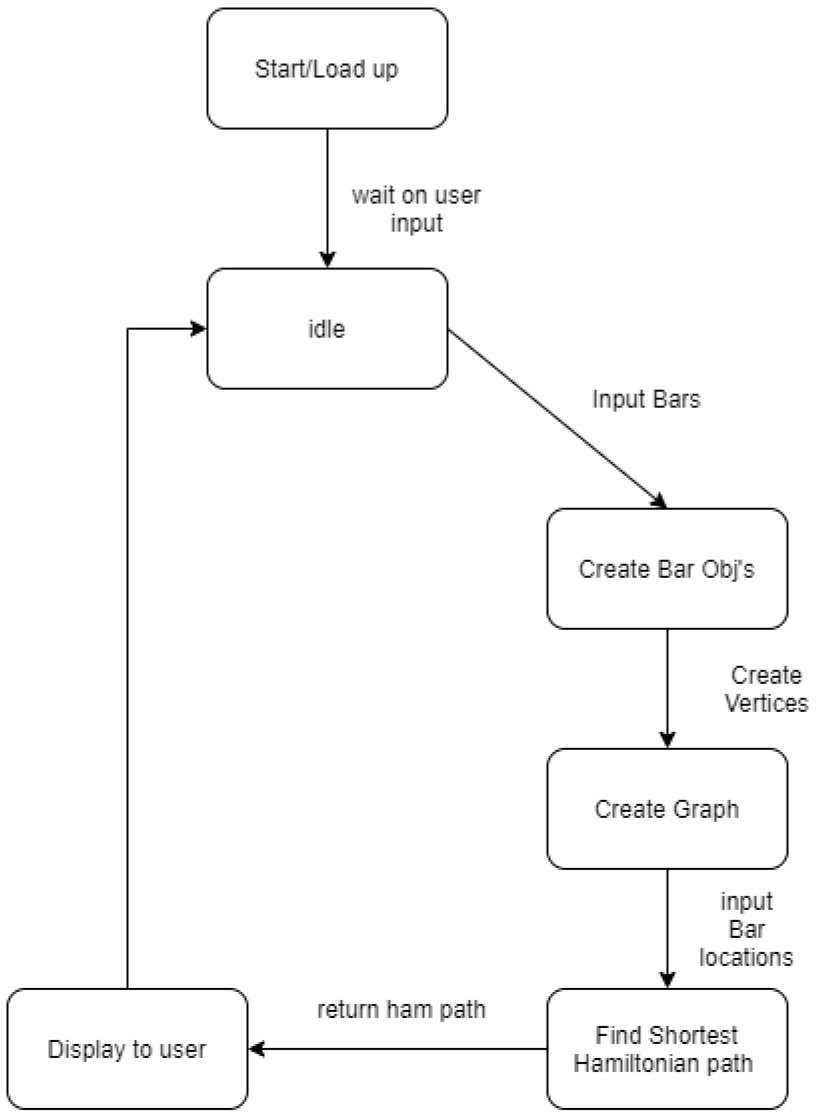
Below is the UML of how we generate the shortest Hamiltonian path and how the program builds up from various modules. The arrows represent the uses relation between components i.e. “Graph” uses “Business” and “Edge”. “Edge” and “Business” are key components that are inherited by all other modules. A graph consists of vertices and edges and that is what “Business” and “Edge” are respectively. When running the shortest Hamiltonian graph, we know in theory it must have a graph to run on and is therefore provided the graph class. Business controller on the other hand uses the business repository as business repository is the class that extends the JpaRepository, but for the business type which is needed when the business controller is iterating over the sequence of business type objects.

Figure 1. A UML Diagram for generating the shortest hamiltonian path



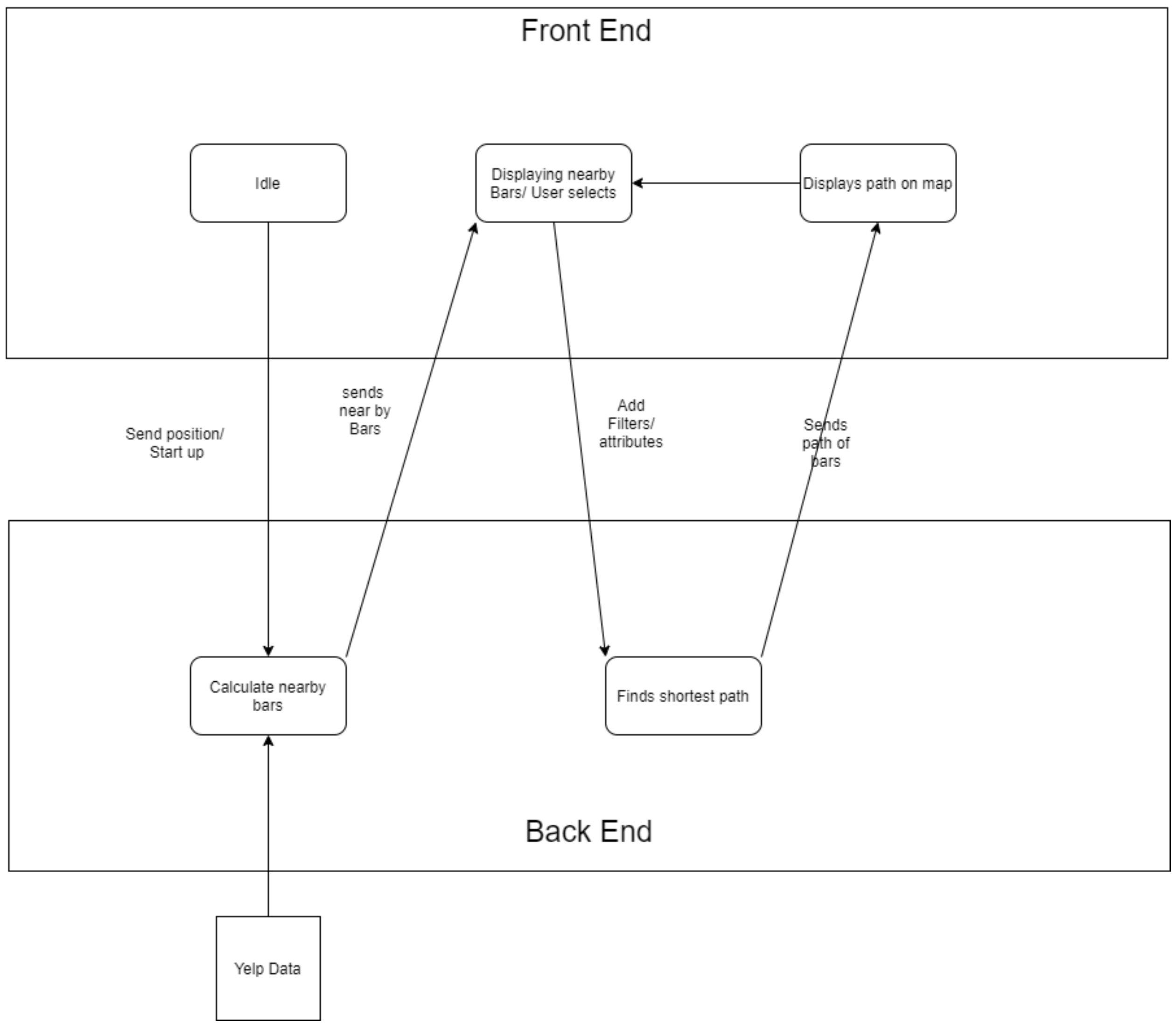
The second UML state machine diagram shows how the class variables are maintained by the methods of the class.

Figure 2. A UML State Machine Diagram for soBar application variables and methods



Below is a UML state machine of how we pass around information between the front-end and back-end.

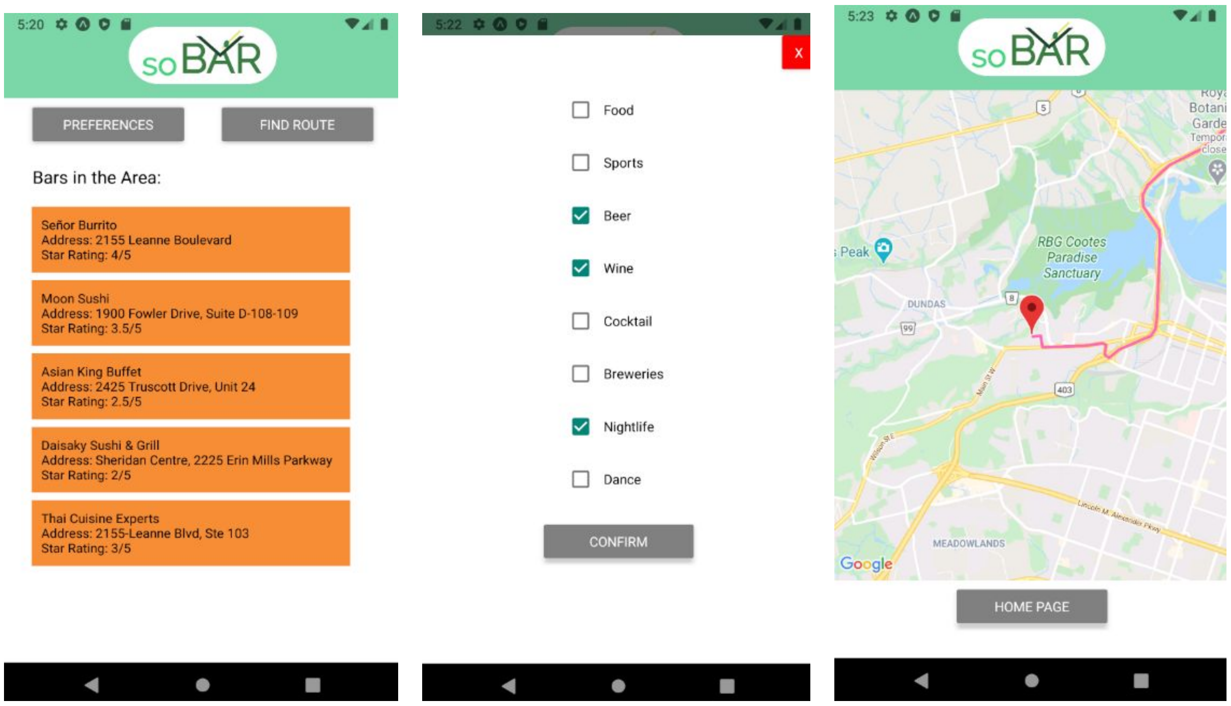
Figure 3. A UML State Machine Diagram for information passing



1.3 Front-end Classes and Modules

This section includes a description of the interface for each class and describes the semantics for the Front-end development of soBar.

Figure 4. Images of Home screen with Bars, Preferences Modal displayed using Android Studio, and Application MapScreen



Bars in the Area:

Señor Burrito
Address: 2155 Leanne Boulevard
Star Rating: 4/5

Moon Sushi
Address: 1900 Fowler Drive, Suite D-108-109
Star Rating: 3.5/5

Asian King Buffet
Address: 2425 Truscott Drive, Unit 24
Star Rating: 2.5/5

Daisaky Sushi & Grill
Address: Sheridan Centre, 2225 Erin Mills Parkway
Star Rating: 2/5

Thai Cuisine Experts
Address: 2155 Leanne Blvd, Ste 103
Star Rating: 3/5

- ☐ Food
- ☐ Sports
- ☑ Beer
- ☑ Wine
- ☐ Cocktail
- ☐ Breweries
- ☑ Nightlife
- ☐ Dance

1.3.1 Home Screen Module

Description: This class uses React Native Components to display the home screen of the application. The screen provides a view of nearby bars and allows navigation to other major screens.

Uses: Preferences, global, HTTP_Bars

Attributes and Operations:

Name	**Type**	**Description**
Preferences Button	TypeButton	Navigates user to the preferences modal
Find Route Button	TypeButton	Navigates user to the Map Screen
Bars FlexList	FlexList	A FlexList displaying a scrollable view of bars near the user's location
Header	Header	Displays the application image on top of app screen
modalVisible	Boolean	When true will turn the buttons will give an appearance of being clicked

Routines:

Routine Name	**In**	**Out**	**Exceptions**
cancelPrefHandler		Boolean	None

State Variables
modalVisible: $\mathbb{B}$

1.3.2 Map Module

Description: This module displays the Map Screen and the optimal route between nearby bars. It includes a header and navigation button to route back to the home screen.

Uses: global

Attributes and Operations:

Name	**Type**	**Description**
Home Page	Type Button	Will return user to homepage
markerLocation	array	Array consisting of marker's longitude and latitude
MapView	Interface mapping system	Displays map of route for end users

Routines:

Routine Name	In	Out	Exceptions
MapScreen	props		None
changeMarkerLocation	Double, Double	MapScreen	None

State Variables
markerLocation: List

1.3.3 Preferences Module

Description: Defines the modal that displays the list of preferences for the user to select. The module also makes http GET requests to send preferences to the backend to be processed and return the new list of bars.

Uses: global

Attributes and Operations:

Name	Type	Description
pref	array	Stores a list of user selected preferences
Preference Checklist	Checkbox	Allows user to select which preferences they desire
Screen Modal	Modal	Displays the preferences screen as a React modal
Confirm	Button	Saves selected preferences and returns to HomeScreen
Cancel	Button	Clears selected preferences and returns to HomeScreen

Routines:

Routine Name	In	Out	Exceptions
preferences	props		None
addPrefHandler	Boolean		None
cancelGoalHandler	Boolean		None
breweriesSelec	Boolean		None
nightlifeSelec	Boolean		None
danceSelec	Boolean		None
foodSelec	Boolean		None
sportsSelec	Boolean		None
beerSelec	Boolean		None
wineSelec	Boolean		None
cocktailSelec	Boolean		None

State Variables
preferencesSelected: List

1.3.4 globalStyles Module

Description: Defines global styles used in the soBar Application.

Uses: None

Attributes and Operations:

Name	Type	Description
globalStyles	StyleSheet	Defines global styles for application
header	Styles	Style description for header
logo	Styles	Style description for the logo

buttonContainer	Styles	Style description for button container
button	Styles	Style description for buttons
titleText	Styles	Style description for title text
paragraph	Styles	Style description for paragraph
cancel	Styles	Style description for cancel button
mapContainer	Styles	Style description for map container
map	Styles	Style description for the map
prefContainer	Styles	Style description for the preferences container
barsContainer	Styles	Style description for the bars container
screen	Styles	Style description for screen
checkboxContainer	Styles	Style description for checkbox container
item	Styles	Style description for the items display
bar	Styles	Style description for the bars
subinfo	Styles	Style description for the subtitle

Routines: None

State Variables: None

1.3.5 App Module

Description: This class is called to display the application screen depending on the state of HomeScreen and MapScreen variables.

Uses: Home, Map, globalStyles

Attributes and Operations:

Name	Type	Description
Content	Map	Will display a blank map
mapShow	Boolean	If true then display the map otherwise display home screen
pref	Array	Is an array for other modules to add preferences to
route	Array	Is an array for other modules to add to the final route

Routines:

Routine Name	In	Out	Exceptions
App		Content	None
changeMap	Boolean		None

State Variables: None

1.3.6 HTTP_Bars Module

Description: This module is used as a filereader, it takes in the data of each bar from the website and parses it so that we can later make objects of type business. The information is exported to other modules to use.

Uses: BarsDisplay

Attributes and Operations:

Name	Type	Description
BarsHTTP	Class	Parses website data for bar info and stores name, address, and rating
BarsDisplay	BarsDisplay	Displays the bar information

Routines:

Routine Name	In	Out	Exceptions
fetch	String, String		

State Variables
Website: String
data: List

1.3.7 BarsDisplay Module

Description: This module creates the view of the pop-up to select which bars the end user would like to add to their route and mainly entails of Flatlist to display this. An example of this pop-up can be seen in the second picture of Figure 4.

Uses: global

Attributes and Operations:

Name	Type	Description
setSelected	FlatList	Changes the colour of the rectangle as if selected
onSelect	FlatList	Changes the state of the bar that is associated with said rectangle to add to the route
Item	FlatList	The display for the bar takes in the associated strings and outputs the information for the bar

Routines:

Routine Name	In	Out	Exceptions
BarDisplay	Boolean	Flatlist	None

State Variables: None

1.4 Internal Review

The final version of soBar is consistent with the Software Requirement Specification document and project proposal in terms of the implementation and content.

Our project proposal explained that the application would be designed to target three main purposes all while being compatible with iOS and Android operating systems. We marketed our main functions as “searching and listing bars and other similar establishments”, “allowing filtering based on user defined criteria” and “displaying geographical navigation information” which were all accounted for. The purpose of the project was to make an application that did not have similar functionality to anything else on the market while also using and implementing a graphing algorithm to filter through a database of over 100 thousand units written in Java. All of this has been done and can be seen within our Java files and are explained earlier within the document.

Our design focused on targeting particular software qualities to ensure a practical design. The main focus behind our variety of classes was the principle of modularity which focuses on having low coupling and high cohesion. We tried to make sure that our modules in the frontend and backend both did not depend on too many other modules and tried to build from the ground up. We first built our simplest modules like “Business”, “Edge” seeing that it did not depend on any other module and then gradually built other modules that would use classes like business. Essentiality is also a property that we tried to follow where we tried to make our code as particular as possible to ensure that the other team members would not have too much difficulty understanding what the other team members were coding. You can see this in modules such as globalStyles where there was a different stylesheet for every different type of object that would be outputted on our interface.

If the soBar app was to release another version with improvements and additional content we would strive for a navigational feature that actively uses your location and can guide you to the bars. Right now our app can tell you the route to take, but it does not have active navigational capabilities.