Module Interface Specification for SmartServe

Team 21, StoneCap Solutions
Max Turek *turekm*
Ryan Were *werer*
Sam Nusselder *nusselds*
Peter Minbashian *minbashp*
David Bednar *bednad*1

April 6, 2023

1 Revision History

Version	Date	Developer(s)	Change(s)
1.0	01/18/23	Max Turek Ryan Were Sam Nusselder Peter Minbashian David Bednar	Initial Draft

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at SRS

Contents

3 Introduction

The following document details the Module Interface Specifications for Smart Serve, an autonomous drink creation machine.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at Smart-Serve.

4 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	**Level 2**
Hardware-Hiding Module	Python Hardware
Behaviour-Hiding Module	Send Order Volume Tracker Drink Ready
Software Decision Module	Menu Page Admin Ingredient Input Order History Header Login Page

Table 1: Module Hierarchy

5 MIS of Drink Ready

5.1 Module

Drink Ready

5.2 Uses

Used to determine whether a not a drink has been completed. This is used for the system logic as it helps the system decide if a new drink is ready to be made.

5.3 Syntax

5.3.1 Exported Constants

Name	finished
Type	Boolean

5.3.2 Exported Access Programs

Name	In	Out	Exceptions
drinkDone	sentDone	finished	errorInMaking

5.4 Semantics

5.4.1 State Variables

sendDone: *Boolean*

5.4.2 Environment Variables

Cup Ingredients

5.4.3 Assumptions

N/A

5.4.4 Access Routine Semantics

sendDone():

- output:
  If cupfill()
  Then send 'True' Boolean
- exception: **If** error found in making drink
  Then send 'False' Boolean

drinkDone():

- output:
  If sentDone()
  Then notifyUser()

- exception: If error found in making drink

5.4.5 Local Functions

notifyUser(): Displays pop up window to user that their drink is complete

cupfill(): Outputs True Boolean if cup if full, using sensor data, or output False if not full

6 MIS of Volume Tracker Module

6.1 Module

Volume Tracker

6.2 Uses

Used to keep track of the amount of ingredients available within the system to ensure appropriate drinks are being offered

6.3 Syntax

6.3.1 Exported Constants

Name	ingredientAmounts
Type	JSON

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
trackVol	ingredientAmounts	N/A	negativeIngredients

6.4 Semantics

6.4.1 State Variables

currentVolumes: *JSON*
ingredientAmounts: *JSON*
drinkOrdered: *Boolean*

6.4.2 Environment Variables

Ingredients

6.4.3 Assumptions

Owner of machine appropriately sets the location of each of their ingredients

6.4.4 Access Routine Semantics

changeVolumeAmounts():

- output:
  If drinkOrdered
  Then Subtract Volume Amounts drinkOrdered

- exception:
  If ingredient volume goes is in the negative

6.4.5 Local Functions

N/A

7 MIS of Send Order Module

7.1 Module

Send Order

7.2 Uses

Used to send data to the hardware to signal a drink must be made and the ingredients for said drinks

7.3 Syntax

7.3.1 Exported Constants

Name	drinkIngredients
Type	JSON

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
sendOrder	drinkName	drinkIngredients	drinkDNE

7.4 Semantics

7.4.1 State Variables

drinkName: *String*
drinkList: *JSON*

7.4.2 Environment Variables

N/A

7.4.3 Assumptions

N/A

7.4.4 Access Routine Semantics

sendOrder():

- output:
  If getIngredients(drinkName) is not NULL
  Then sendDrinkIngredients(ingredients)

- exception: **If getIngredients(drinkName) is NULL**

7.4.5 Local Functions

getIngredients(drinkName): Gets ingredients of corresponding drink name from JSON library of all drink names and respective ingredients. Returns Null if drink does not exist.

sendDrinkIngredients(ingredients): Send JSON object containing all appropraite ingredients and their measurements

8 MIS of Python Hardware

8.1 Module

Python Hardware

8.2 Uses

This module serves to link the web application and the hardware with each other. The module gets called with parameters that are already hard coded in the web application.

8.3 Syntax

8.3.1 Exported Constants

Name	cocktailCreated
Type	boolean

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
makeDrink	ingredientList ingredientLocations	Boolean	incorrectOrder noOrder

8.4 Semantics

8.4.1 State Variables

Drink: *Drink*
cocktailCreated: *cocktailCreated*

8.4.2 Environment Variables

Cup
Ingredients

8.4.3 Assumptions

Owner of machine appropriately sets the location of each of their ingredients

8.4.4 Access Routine Semantics

makeCocktail():

- output:
  If Drink order is sent
  and If cuppresent()
  then pourDrink()
  then rotate()
- exception: If Ingredients are not available **or** If Cups not available

8.4.5 Local Functions

cupfill(): Outputs True Boolean if cup if full, using sensor data, or output False if not full

cuppresent(): Outputs Boolean True if the cup is present, using the sensor, or outputs False

pourDrink(): Called to activate GPIO pins to activate pump

rotate(): Called after the drink is made to rotate the "lazy susan" to fill the next cup

9 MIS of Login Page Module

9.1 Module

Login Page

9.2 Uses

The module is used to register or sign in regular and administrative users. Access to the header module is granted upon successful registration or sign in.

9.3 Syntax

9.3.1 Exported Constants

loginDB:
for each row entry:

Name	username	password
Type	string	string

9.3.2 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
signIn	username, password	N/A	doesUExist, doesUPMatch, isChLimit
register	username, password	N/A	!doesUExist,isChLimit

9.4 Semantics

9.4.1 State Variables

Name Type Range
Username string 15 characters
Password string 20 characters

9.4.2 Environment Variables

The web application is displayed on a screen using HTML, CSS, and reactJS

9.4.3 Assumptions

N/A

9.4.4 Access Routine Semantics

signIn (username, password):

- transition:
  if (doesUnameExist(username)
  and doesUnamePassMatch(username, password)
  and isCharLimit(username, password))
  then
  Goto header module

- exception:
  else
  Goto Login Page Module

Register(username, password):

- transition:
  if (not doesUnameExist(username)
  and isCharLimit(username, password))
  then
  Goto header module

- exception:
  else
  Goto Login Page Module

9.4.5 Local Functions

doesUExist(username):

- transition:
  Output: True if username in all loginDB.username

doesUPMatch(username, password):

- Output: True if loginDB(username) == password

isChLimit(username, password):

- Output: True if username.length $\leq$ to 15 and password.length $\leq$ 20

10 MIS of Header Module

10.1 Module

Header

10.2 Uses

The module is used to navigate to either the menu page, the admin ingredient input, or the order history module

10.3 Syntax

10.3.1 Exported Constants

adminDB:
for each row entry:

Name	adminUName
Type	string

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
goToMenu	N/A	N/A	N/A
goToAdmin	username	N/A	isAdmin
goToHistory	N/A	N/A	N/A

10.4 Semantics

10.4.1 State Variables

N/A

10.4.2 Environment Variables

The web application is displayed on a screen using HTML, CSS, and reactJS

10.4.3 Assumptions

N/A

10.4.4 Access Routine Semantics

goToMenu ():

- transition: Goto menu page module

goToAdmin (username):

- transition:
  if isAdmin(username)
  then
  Goto admin ingredient input module

- exception:
  else
  Goto Header Module

goToHistory():

- transition: Goto order history module

10.4.5 Local Functions

isAdmin(username):

- transition:
  Output: True if username in all adminDB.adminUName

11 MIS of Menu Page Module

11.1 Module

Menu Page

11.2 Uses

The module is used to order available drinks.

11.3 Syntax

11.3.1 Exported Constants

drinkDB:
for each row entry:

Name	name	image	category	ingredient
Type	string	png/jpg	string	list(string)

drinkHistoryDB:
for each row entry:

Name	drinkName
Type	string

11.3.2 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
orderDrink	drinkName	N/A	isDrinkAvailable

11.4 Semantics

11.4.1 State Variables

Name Type Range
drinkName string 30 characters

11.4.2 Environment Variables

The web application is displayed on a screen using HTML, CSS, and reactJS

11.4.3 Assumptions

N/A

11.4.4 Access Routine Semantics

orderDrink(drinkName):

- transition:
  if isDrinkAvailable(drinkName)
  then
  Send "send order module":
  (drinkDB(drinkName).ingredients and ingDB)
  and
  drinkHistoryDB.append(drinkName)
  and
  export drinkHistoryDB to order history module
- transition: Goto menu page module

11.4.5 Local Functions

isDrinkAvailable(drinkName):

- transition:
  Output: True if
  all ingredients in drinkDB(drinkName).ingredients
  in ingDB.ing

12 MIS of Admin Ingredient Input

12.1 Module

Admin Ingredient Input

12.2 Uses

The module is used by an admin user to input ingredients and corresponding dispenser locations

12.3 Syntax

12.3.1 Exported Constants

ingDB:
for each row entry:

Name	ing	dispenser
Type	string	int

12.3.2 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
addIng	ingName, dispNum	N/A	N/A

12.4 Semantics

12.4.1 State Variables

Name Type Range
ingName string 30 characters
dispNum int -

12.4.2 Environment Variables

The web application is displayed on a screen using HTML, CSS, and reactJS

12.4.3 Assumptions

N/A

12.4.4 Access Routine Semantics

addIng(ingName, dispNum):

- transition: ingDB(ingName) = dispNum
  and
  export ingDB to main page module

12.4.5 Local Functions

N/A

13 MIS of Order History Module

13.1 Module

Order History

13.2 Uses

The module is used to store a list of the names of made drinks

13.3 Syntax

13.3.1 Exported Constants

N/A

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
displayList	N/A	drinkList	N/A

13.4 Semantics

13.4.1 State Variables

Name Type Range
N/A

13.4.2 Environment Variables

The web application is displayed on a screen using HTML, CSS, and reactJS

13.4.3 Assumptions

N/A

13.4.4 Access Routine Semantics

displayList():

- otput: import drinkHistoryDB from ”main menu module”

13.4.5 Local Functions

N/A

14 Appendix