Group B
Jawad Aziz Khan - 30139146
Ahmad El Masri - 30115844
Sabit Ibn Ali Khan - 30139150

SENG 696: Agent-based Software Engineering - Fall 2021
NutriVision
Project Overview

Vision-based Calorie Counter:
- User uploads **photo of food** they want tested
- Photo is processed by a computer vision model
- Classification **text** is produced and used for nutrition request
- A query is made to a nutrition API that returns **calories**

Agents Outline:
- Camera Agent - Process image
- Classifier Agent - Vision API (mock)
- Nutrition Agent - Nutritionix API
- Interface Agent - Connect to backend server
- Gateway Agent - handle HTTP requests
- Maker Agent - Creates the Above Agents
- Services Agent - Yellow Pages Parent Class Agent

What We Will Be Using:
- JADE - Agent Development Environment
- Restful API's
- Gaia - Agent Methodology
- Swing GUI
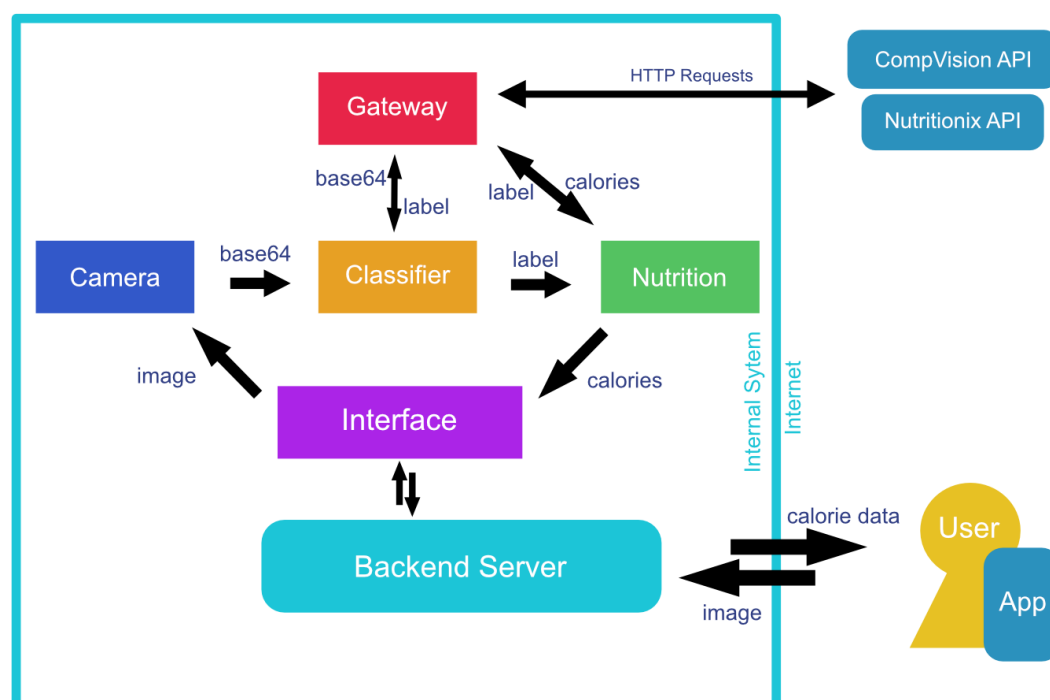
System Requirements:

1. Frontend app will have a user-friendly GUI.
2. User would take a **photo of the food item** and request feedback on calories.
3. App must be capable of RESTful HTTP API requests
4. Backend server must be capable of receiving, handling and responding to API requests from the app
5. Upon receiving request, server will hand off processing to agents
6. System can store the user's daily caloric intake in a DB (in-app or Mongo)
7. System agents can expect inputs from multiple users which are placed in a queue or taken by available same-role agents.
8. System shall utilize different web API's asynchronously to process the different inputs at each stage of the processing.
9. Registered agents can request from each other the available information in real-time.
10. Agents should be able to publish and search for services when needed.

Analysis:

Roles Model:
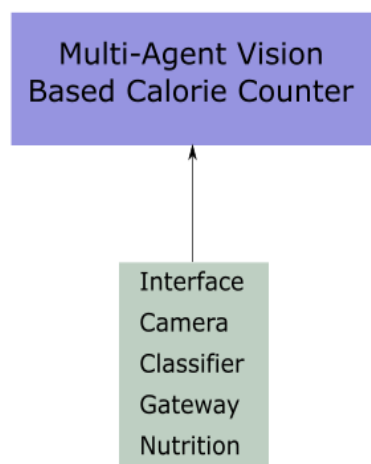
Agents Model:
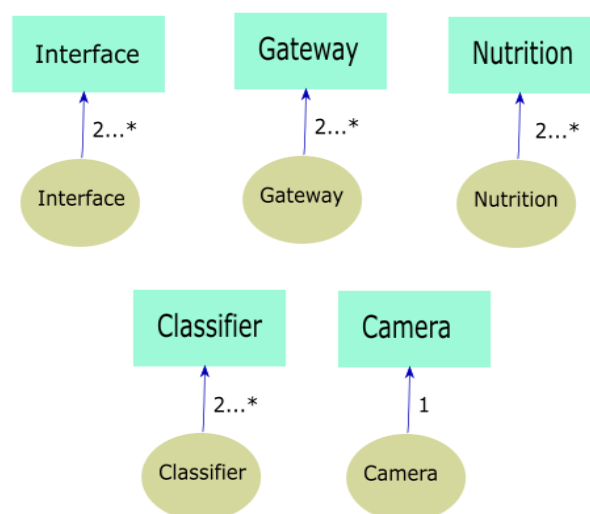
Role Schemas:

Role Schema	Interface
Description	To provide the user input as an image to the camera agent, and send + receive a calorie count request.
Protocols and Activities	- Transfer image file - Receive nutrition data
Permissions	Read Input from GUI and Change GUI text
Responsibilities	-Liveness: InterfaceService= (InterfaceService.SERVICE) - Safety: FileTypeFilter (jpg, jpeg only)

Role Schema	Camera
Description	To convert the image to Base-64 and send a request to the classifier.
Protocols and Activities	Utilize java.util.Base64 package for conversion.
Permissions	Image File Access.
Responsibilities	-Liveness: CameraService = (CameraService.SERVICE) - Safety: Image file not too large to reduce cost of processing.

Role Schema	Classifier
Description	To receive the base-64 image and classify it as a food group in order to make nutrition calculations
Protocols and Activities	Request label from SmartLens web API, as a POST method. **The SmartLens API have recently shutdown**
Permissions	Connect to Camera, Gateway and Nutrition Agents
Responsibilities	-Liveness: ClassifierService= (ClassifierService.SERVICE) - Safety: Secure connection with SmartLens API.

Role Schema	Nutrition
Description	To receive the label text from classifier, request nutrition data from Nutritionix web API and send results back to Interface agent
Protocols and Activities	Query the Nutritionix API, as a POST method. https://trackapi.nutritionix.com/v2/natural/nutrients
Permissions	Connect to Classifier, Gateway and Interface Agents
Responsibilities	-Liveness: NutritionService= (NutritionService.SERVICE) -Safety: Secure connection with Nutritionix API.

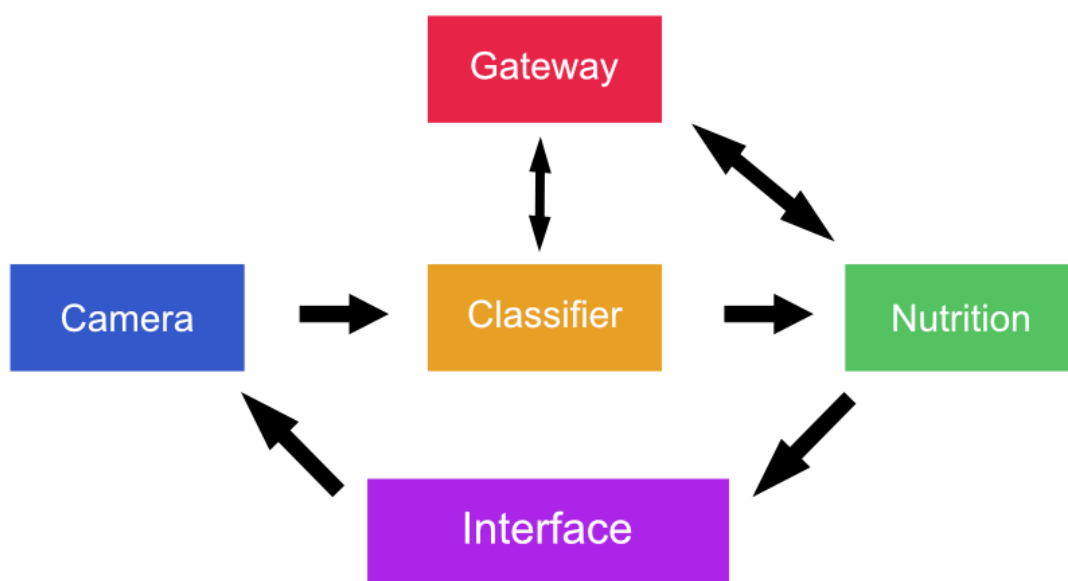| Role Schema	Gateway
Description	Sends HTTP requests to Nutritionx API and CompVision API, and returns the label to the Classifier, and calorie count to the Nutrition agent.
Protocols and Activities	Query the Nutritionix API, as a GET method. https://trackapi.nutritionix.com/v2/natural/nutrients Request label from CompVision API, as a POST method.
Permissions	HTTP Internet Access. Connect to Classifier and Nutrition Agents.
Responsibilities	-Liveness: GatewayService = (GatewayService .SERVICE) -Safety: Secure connection with Nutritionix and CompVision API's.

Interaction Model:

Protocol	Interface	Camera	Classifier	Nutrition	Gateway
Purpose/ Parameters	Provides a camera agent with user input as an image. Sends and receives calorie count request	Converts the image to a base-64 and sends image to classifier	Receives base-64 image, classifies image and sends the classification to nutrition counter	Receives the label from Classifier, requests data from vision API, and sends nutrition calculation back to interface	Acts as a middleman for Classifier and Nutrition agents when making HTTP requests
Initiator(s)	User and backend server	Interface request	Camera	Classifier	Classifier Nutrition
Receiver(s)	Camera agent	Classifier	Nutrition	Interface	Classifier Nutrition
Processing	Image data transfer to Camera agent.	Base-64 Image data transmitted to the Classifier agent.	Sends classification text to the Nutrition Agent.	Returns the result of the calorie request back to the Interface Agent.	Returns JSON data from API calls to agents that requested it

Services Model:

Service	Inputs	Outputs	Preconditions	Postconditions
Interface	Image file	Nutrition data	User picks photo	Image file selected and path is available
Camera	Image path	Base64 image	Path is valid	Conversion is complete
Classifier	Base64 image	Label text	Sends the base64 input to the Gateway agent.	Returned text label is a String and contains food name
Nutrition	Label text	Nutrition data	Label text is string and contains food name	Nutrition data is sent back to the user via Interface and GUI.
Gateway	Base64 image Label	-Label text -Calories Result	Base64 is a string and the agent can make an API call to get Label. Label text is a string and can be used to make an API call	Takes the API output and sends it to the Classifier or the Nutrition agent

Acquaintances Model:

Part 2: Detailed Development Document

1. Detailed class diagram:

CameraAgent

+classifierAgents

#setup()

Agent

ClassifierAgent

+nutritionAgents
+gatewayAgents

#setup()

CameraBehaviour

-myAgent:CameraAgent
-finished:boolean
-image: BufferedImage
-imagePath:String
-stateCounter:int
-formatName:String

+CameraBeahviour()
+action()
+sendMsg()
+imgToBase64String()
+readImage()
+done()
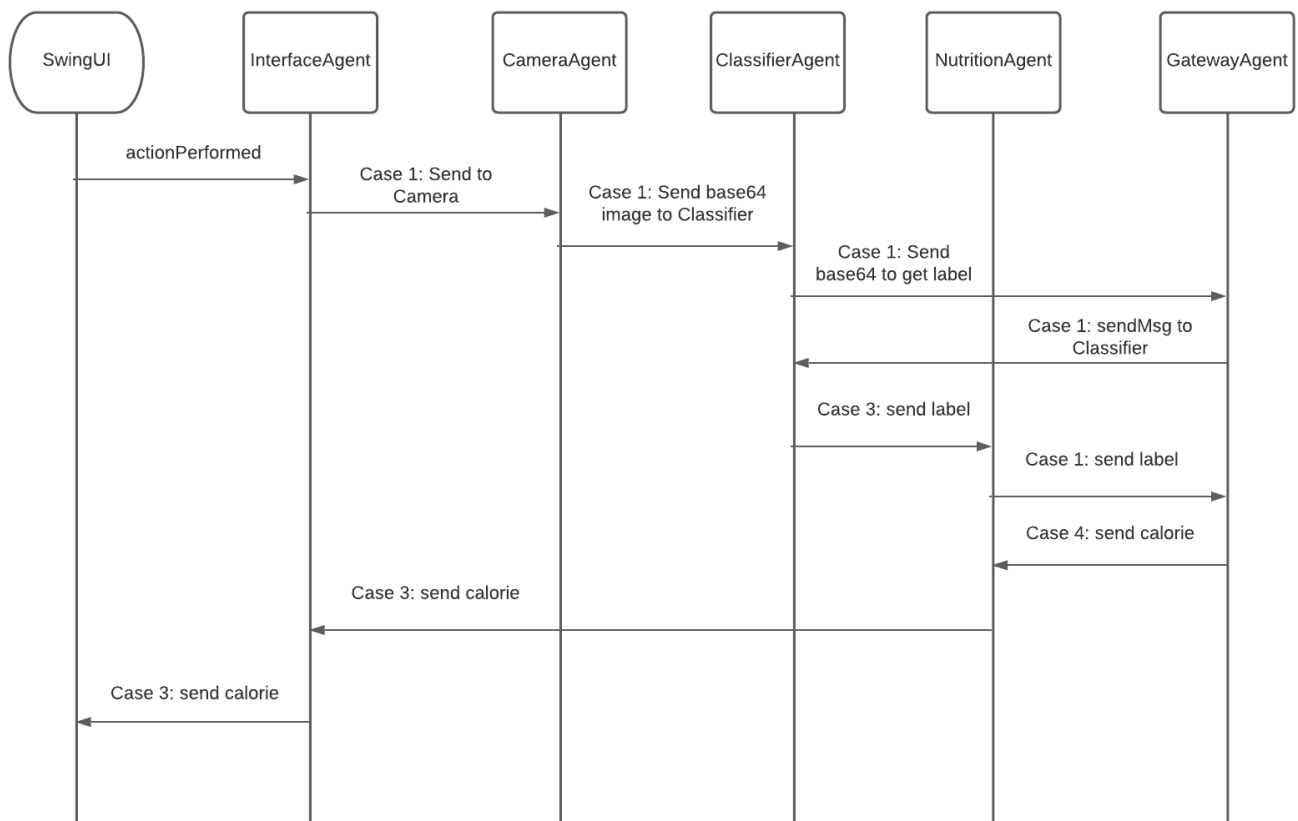

ServiceAgent

+ attribute1:type = defaultValue
+ attribute2:type
- attribute3:type

+ operation1(params):returnType
- operation2(params)
- operation3()

ClassifierBehaviour

-myAgent:ClassifierAgent
-finished:boolean
-stateCounter:int
--serializedImage:String
-label:String

+ClassifierBeahvious()
+action()
+sendMsg()
+done()

SimpleBehaviour

GatewayAgent

+classifierAgents
+nutritionAgents

#setup()

InterfaceAgent

+cameraAgents
+nutritionAgents

#setup()

Jade

GatewayBehaviour

-myAgent:GatewayAgent
-finished:boolean
-stateCounter:int
-base64img:String
-label:String
-calories:String

+GatewayBeahvious()
+action()
+sendMsg()
+sendPOSTRequest()
+done()

InterfaceBehaviour

-myAgent:InterfaceAgent
-finished:boolean
-stateCounter:int
-isImgPathReady:boolean
-imgPath:String
-label:Jlabel
-nutriDataString:String

+InterfaceBehaviour()
+action()
+sendMsg()
+done()

SwingMain

-frame:JFrame
-panel:JPanel
-button:JButton
-fileChooser:JFileChooser
-textField:JTextField
-stateCounter:int
-isImgPathReady:boolean
-imgPath:String
-label:Jlabel
-nutriDataString:String

+SwingMain()

MakerAgent

#setup()
-createAgent()

NutritionAgent

+interfaceAgents
+gatewayAgents

#setup()

ServiceAgent

#takedown()
#register()

NutritionBehaviour

-myAgent:NutritionAgent
-finished:boolean
-label:String
-calorieCount:String
-stateCounter:int

+NutritionBehaviour()
+action()
+sendMsg()
+done()

2. Message sequence chart (i.e. interactions and protocols between agents):

3. Use Case Definition (for all agents):

- Use cases for Interface Agent:

Brief Description	handles the interface and GUI of the overall system
Precondition	GUI has been loaded
Postcondition	Results are showing
Process Steps:	
1	Sends the filepath to the Camera Agent
2	Receives Calorie from Nutrition Agent and sends it to GUI

- Use cases for Camera Agent:

Brief Description	handle image data and forwards to Classifier
Precondition	Image file path is valid
Postcondition	Image converted to base64 forwarded
Process Steps:	
1	Listen for message from Interface
2	Read image from path and convert to base64

- Use cases for Classifier Agent:

Brief Description	handle Image Label data and forward to Nutrition
Precondition	Sends the base64 input to the Gateway agent.
Postcondition	Returned text label is a String and contains food name
Process Steps:	
1	Listen for image base64 from CameraAgent.
2	Send GatewayAgent the base64 to get a label.
3	Wait for a label from Gateway.
4	Send NutritionAgent the label.

- Use cases for Nutrition Agent:

Brief Description	handle Nutrition data (calories) and send to Interface
Precondition	Label text is string and contains food name
Postcondition	Nutrition data is sent back to the user via Interface and GUI.
Process Steps:	
1	Listen for label input from Classifier.
2	Send Gateway the Label.
3	Wait for calorie count from Gateway.
4	Send interface the calories.

- Use cases for Gateway Agent:

Brief Description	make HTTP API requests outside of the Agent network
Precondition	image and label are strings and can make API calls
Postcondition	Send back label and calories to agents
Process Steps:	
1	Listen for message from Classifier
2	Process label (mocked) and reply to Classifier with label
3	Listen for message from Nutrition
4	Process calorie data from label (via API call)
5	Reply to Nutrition with calorie data