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:

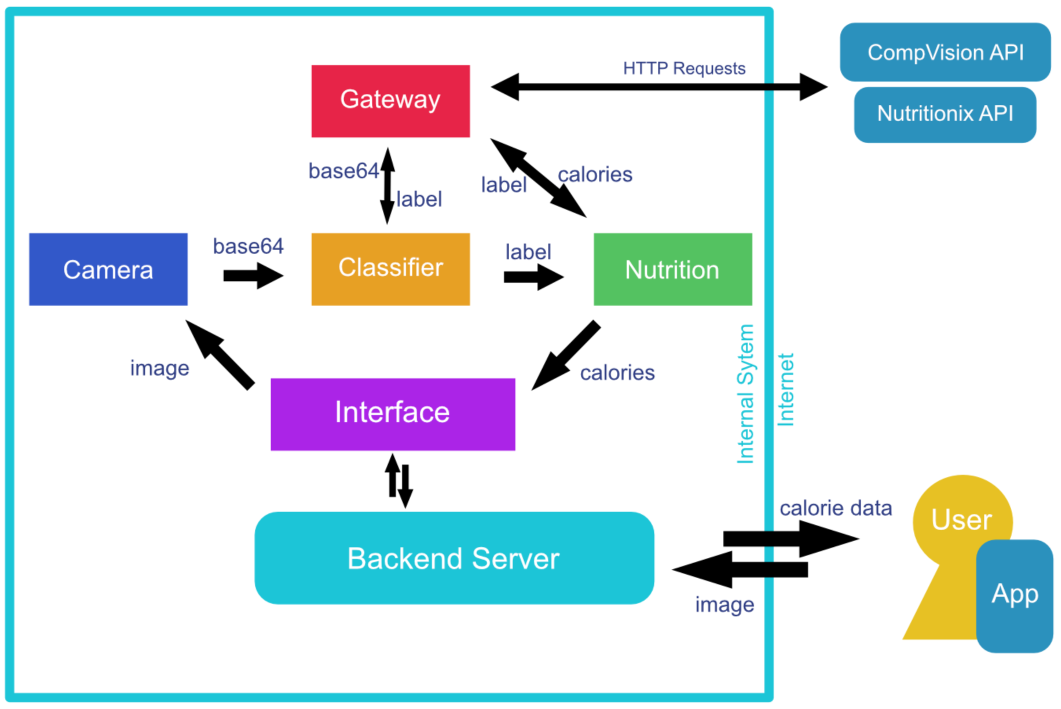
- User uploads photo of food they want tested
- Photo is processed by a computer vision model
- Classification **text** is produced and used for nutrition request
- A query is made to a nutrition API that returns calories

Agents Outline:

- Camera Agent - Process image
- Classifier Agent - Vision API
- Nutrition Agent - Nutritionix API
- Interface Agent - Connect to backend server
- Gateway Agent - handle HTTP requests

What We Will Be Using:

- JADE - Agent Development Environment
- Restful API's
- Gaia - Agent Methodology
- Swing GUI

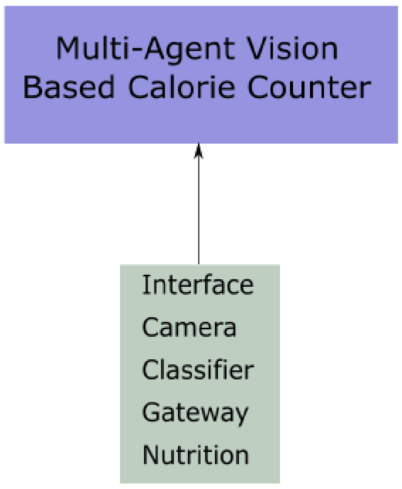


System Requirements:

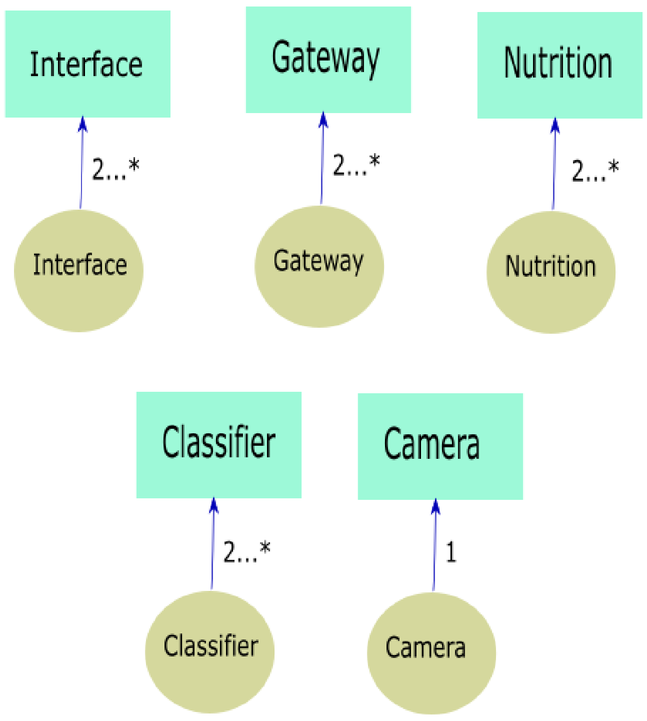
1. Frontend app will have a user-friendly GUI.
2. User would take a **photo of the food item** and request feedback on calories.
3. App must be capable of RESTful HTTP API requests
4. Backend server must be capable of receiving, handling and responding to API requests from the app
5. Upon receiving request, 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 and write from Backend Server
Responsibilities	-Liveness: RequestService = (RequestService.SERVICE) - Safety:

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: ImageConversionService = (ImageConversionService.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. https://vision.googleapis.com/v1/images:annotate
Permissions	Connect to Camera, Gateway and Nutrition Agents
Responsibilities	-Liveness: ImageClassifierService = (ImageClassifierService.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 GET method. https://trackapi.nutritionix.com/v2/natural/nutrients
Permissions	Connect to Classifier, Gateway and Interface Agents
Responsibilities	-Liveness: CalorieCountService = (CalorieCountService.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.

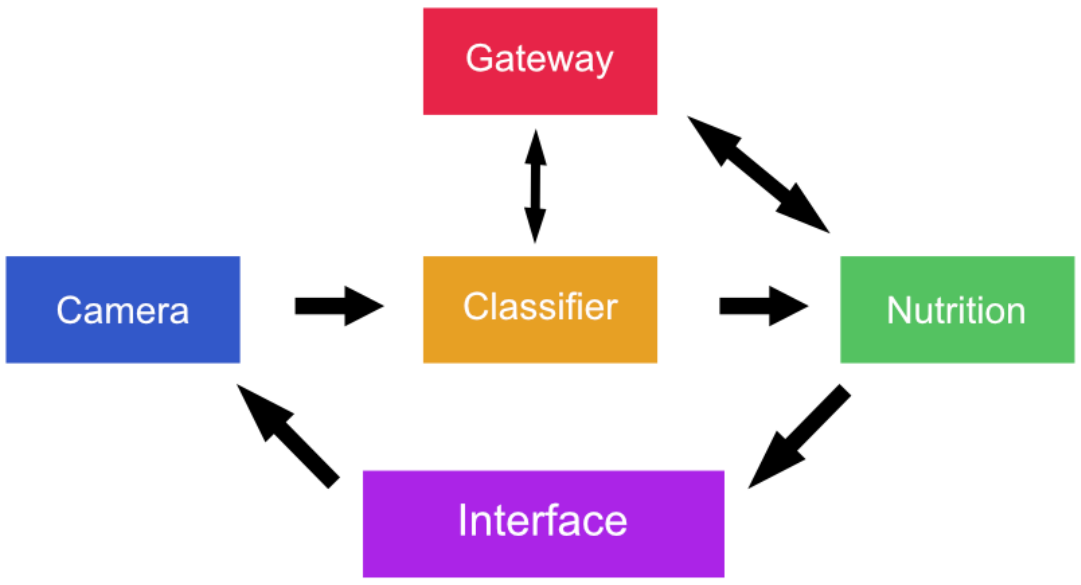
Interaction Model:

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:

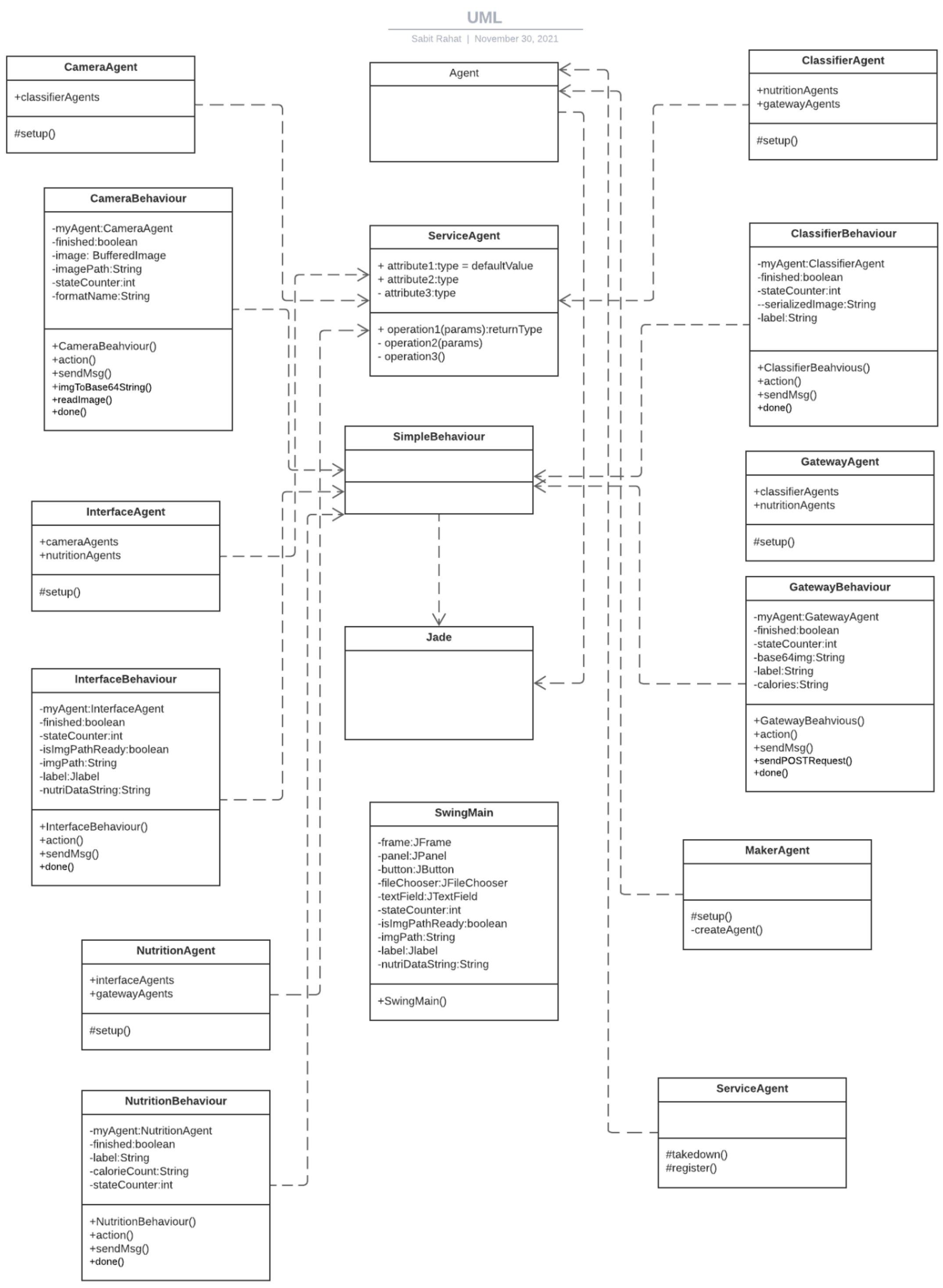
Service	Inputs	Outputs	Preconditions	Postconditions
Interface	Image file	Nutrition data	User sends photo to server via the app	Image file fully uploaded and path is available
Camera	Image path	Base64 image	Path is valid	Conversion is complete
Classifier	Base64 image	Label text	Sends the base64 input to the Gateway agent.	Returned text label is JSON and contains food name
Nutrition	Label text	Nutrition data	Label text is string and contains food name	Nutrition data is sent back to user via Interface and server response
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:



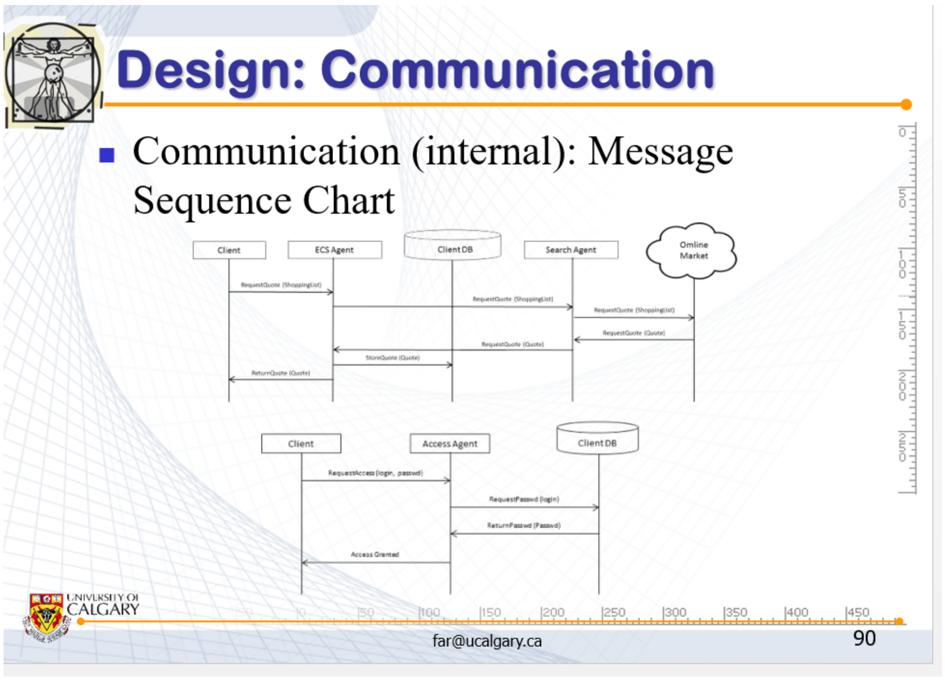
Part 2: Detailed Development Document

1. Detailed class diagram:



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

   Similar to below:



Design: Communication

- Communication (internal): Message Sequence Chart

UNIVERSITY OF CALGARY

far@ucalgary.ca

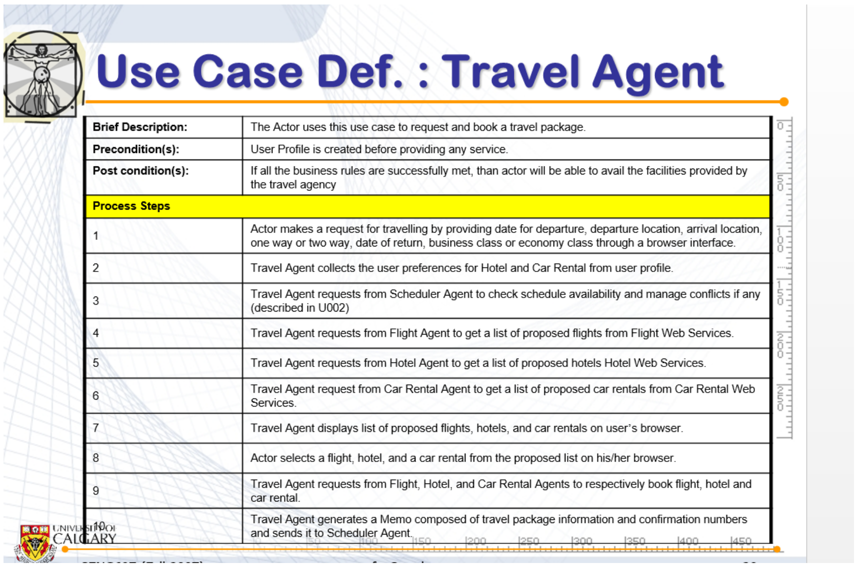
90

3. Use Case Definition (for all agents):

- Use cases for Interface Agent:
- Use cases for Camera Agent:
- Use cases for Classifier Agent:
- Use cases for Nutrition Agent:
- Use cases for Gateway Agent:

Use cases example:

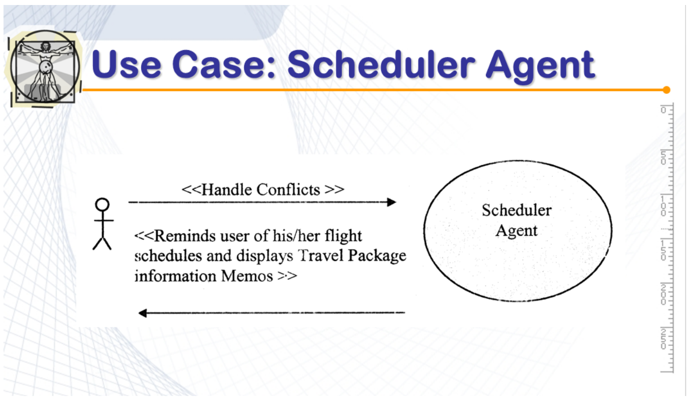
Detailed Use case definition:



Use Case Def. : Travel Agent

Brief Description:	The Actor uses this use case to request and book a travel package.
Precondition(s):	User Profile is created before providing any service.
Post condition(s):	If all the business rules are successfully met, than actor will be able to avail the facilities provided by the travel agency
Process Steps	
1	Actor makes a request for travelling by providing date for departure, departure location, arrival location, one way or two way, date of return, business class or economy class through a browser interface.
2	Travel Agent collects the user preferences for Hotel and Car Rental from user profile.
3	Travel Agent requests from Scheduler Agent to check schedule availability and manage conflicts if any (described in U002)
4	Travel Agent requests from Flight Agent to get a list of proposed flights from Flight Web Services.
5	Travel Agent requests from Hotel Agent to get a list of proposed hotels Hotel Web Services.
6	Travel Agent request from Car Rental Agent to get a list of proposed car rentals from Car Rental Web Services.
7	Travel Agent displays list of proposed flights, hotels, and car rentals on user's browser.
8	Actor selects a flight, hotel, and a car rental from the proposed list on his/her browser.
9	Travel Agent requests from Flight, Hotel, and Car Rental Agents to respectively book flight, hotel and car rental.
10	Travel Agent generates a Memo composed of travel package information and confirmation numbers and sends it to Scheduler Agent.

UNIVERSITY OF CALGARY



Use Case: Scheduler Agent

<<Handle Conflicts >>
<<Reminds user of his/her flight schedules and displays Travel Package information Memos >>
Scheduler Agent

4. Data Specification (for DB, so maybe not needed)