RÉPUBLIQUE DU CAMEROUN
Paix-Travail-Patrie
****
UNIVERSITE DE YAOUNDE 1
****
FACULTE DES SCIENCES
****
DEPARTEMENT D'INFORMATIQUE
****
NIVEAU 4

REPUBLIC OF CAMEROON
Peace-Work-Fatherland
****
THE UNIVERSITY OF YAOUNDE 1
****
FACUTY OF SCIENCE
****
COMPUTER SCIENCE DEPARTMENT
****
LEVEL 4

DEPARTMENT OF COMPUTER SCIENCE

DEPARTEMENT D'INFORMATIQUE

DEVELOPMENT OF A KNOWLEDGE BASE ONTOLOGY PLATFORM FOR FOOD ITEMS

Supervised by

DR. AZANZI JIOMEKONG

UNIVERSITY OF YAOUNDE I

ACADEMIC YEAR 2020/2021

GROUP ONE (1)

Matricule	*Name*	*Scrum Role*
19W2615	**TEKOH PALMA ACHU**	***Scrum Master***
20U2268	**SEFFI TCHANGA PEGUY**	***Product Owner***
15Q2550	**FONGANG NOUSSI NICOLAS**	***Developer***
17Q2890	**ETALI ETALI MATHIAS JUNIOR**	***Developer***

Table of Contents

I. Introduction

An ontology in computer science is a technical term designating an artifact conceived with a precise aim, namely to allow the modelling of knowledge relating to a domain, real or imaginary. It is in this context that within the framework of the practical work of the EU INF 4188 we were given the task of developing a food ontology. We will proceed in two phases System Analysis and Design, and First of all, we will highlight the functional and non-functional requirements of our system, then we will present its architecture and we will finish by giving the development environment used.

II. Software analysis and Design

1. Functional specifications:

These are the features that are expected by the user from our platform. For our system we have the following functionalities:

- Users should be able to access the ontology application using a web or mobile app interface.
- User should be able to create an account on the application.
- Users is presented with a dashboard to record information and carry out other system functionality.
- Users should be able to fill information about their profile corresponding to their resource
- Nutritionist and Cook can Add Knowledge about a meal into the Ontology.
- Nutritionist and Cook can search and modify Knowledge about an already existing meal in the Ontology.
- Users can run queries on the system and get appropriate response from the User Interface.

- Users can invite other users by email to subscribe to the platform by prescribing their profile (public, health professional, nutritionist, etc.)
- A user can search for a meal and add more knowledge to the meal already put by another user.
- A user can search for all food delicacies commonly eaten in a particular country, Region or City e.g. (Cameroon, Littoral, Edea), (Senegal, Dakar region) etc.
- System administrator can be able to validate account of nutritionist and Cooks.
- System administrator can verify, validate and merge pull request to add or modify information in the ontology.

2. Non-functional specifications:

These are the quality attributes of our system which can be used to judge how good or bad our system is. We have the following as non-functional requirements:

- The system can support 500 thousand users at any given instance.
- Query request should not take more than 5sec to get a reply.
- System should guarantee a 99.5% uptime.
- Transaction to the triple store Database should be Robust and Fault tolerant.
- User interface of our platform should be simple and easy to use.
- Our Ontology system should be highly accessible to many users
- Our food ontology should be design in such a manner that it can evolve easily to reflect the constant changing nature of real life.
- High Maintainability of our system since it is open source can easily be used by others.

3. System Actors:

Actors are people or other systems that interact with our application throughout its life cycle. For our system, we are going to have the following actors

- ❖ **Food Expert**
  - **Nutritionist:** A professional whose specialty is on Food and their nutritional contents and Values. He advises people on matters of food and their effect on their health
  - **Cook:** This is a professional who prepares food for consumption and advices people on various food recipes.
- ❖ **Normal User**
  - Diet Patient: A person suffering from either Malnutrition or Over Nutrition and needs knowledge about various Diets.
  - Meal Planner: A person who has an occasion and is looking on means to satisfy the guest based on aspects such as region where people are from etc.
  - Visitor: Any person who is visiting the Platform
- ❖ **System Administrator:**

The person responsible for managing and administering the entire platform.

4. Activity diagrams

The diagram below represents the workflow of activities required in order to realize a given task. In our project we present two (2) activity diagrams:

- Nutritionist/cook registering & adding knowledge about a Meal to Ontology.
- User searching for a meal on the platform

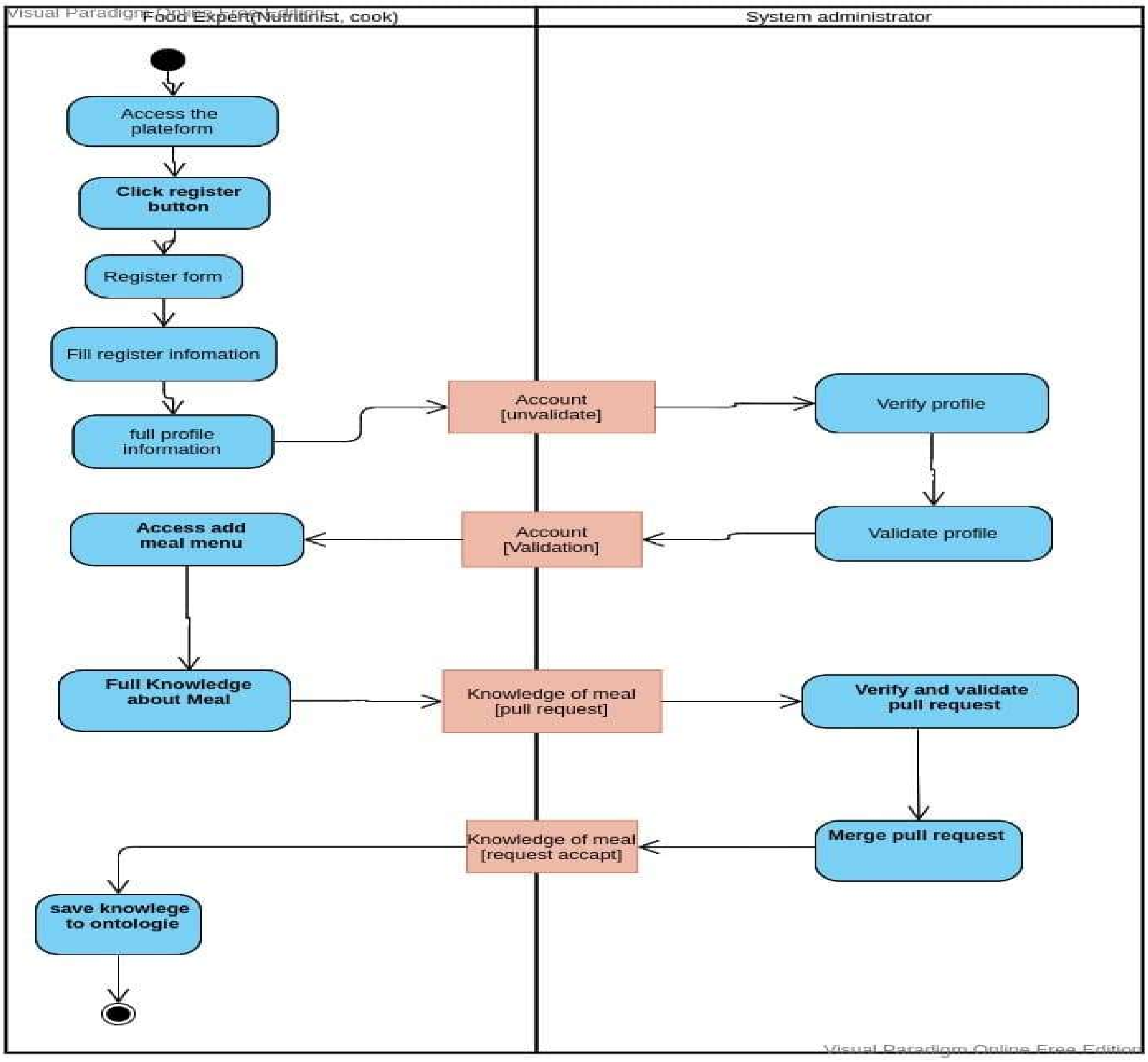

Visual Paradigm Online Free Edition
Food Expert(Nutritionist, cook)
System administrator
Access the plateform
Click register button
Register form
Fill register infomation
full profile information
Account [unvalidate]
Verify profile
Validate profile
Account [Validation]
Access add meal menu
Full Knowledge about Meal
Knowledge of meal [pull request]
Verify and validate pull request
Merge pull request
Knowledge of meal [request accapt]
save knowlege to ontologie
Visual Paradigm Online Free Edition

Figure 1: Activity diagram for Nutritionist/cook registering & adding knowledge about a Meal to Ontology.

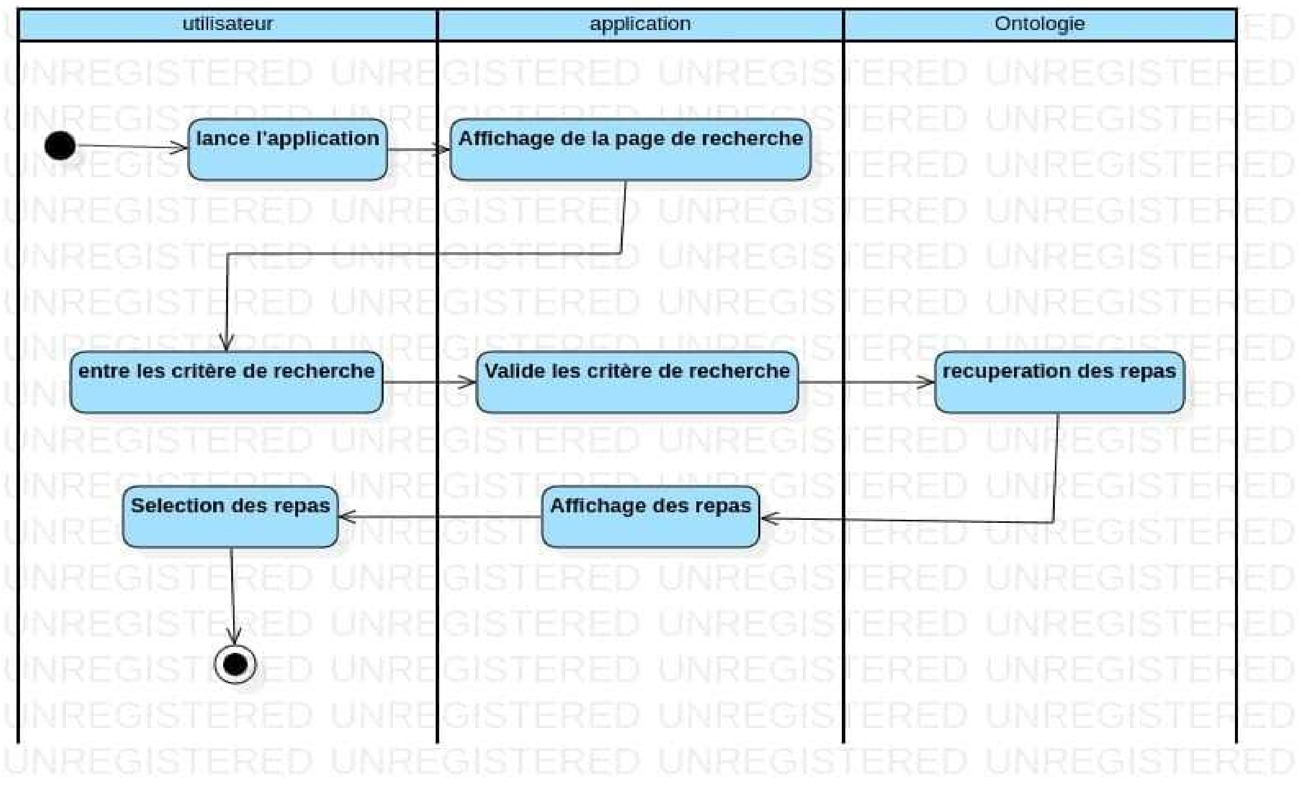


Figure 2:Activity diagram for Searching a Meal

5. Use Cases Classified by Actors

Actor		Use Case
Food Expert	*Nutritionist*	✓ Create account ✓ Sign in ✓ Validate information in his domain. Like what quantity of food will give what nutritional value ✓ Download and export Ontology (in OWL, RDF formats). ✓ Can also import ontology into the platform. ✓ Search for a particular Food

	Cook	✓ Create account ✓ Sign in ✓ Search for a particular Food ✓ Validate information of recipes in the platform
Normal User	*Meal planner*	✓ Get the right type of meal to cook for an occasion to satisfy an audience from a particular country e.g. Senegal ✓ Search for a particular Food ✓ Sign in
	Diet Patient	✓ Get the type of food to eat in order to satisfy calories prescribed by a doctor. ✓ Register his information on his health situation ✓ Enter information about Diet experiences he has had.
	Visitor	✓ Does not need to register ✓ Can download an Ontology ✓ Consult a list of Diets ✓ Search for a particular Food
System Administrator		✓ Sign in. ✓ Validates the account of those who are experts in the domain. ✓ Validate and Invalidate. Information about Food sent on the platform. ✓ Merge pull request of various information entered by experts

6. Description of use cases using the text formalism

We used the following use case diagram to show how users interact with our system.

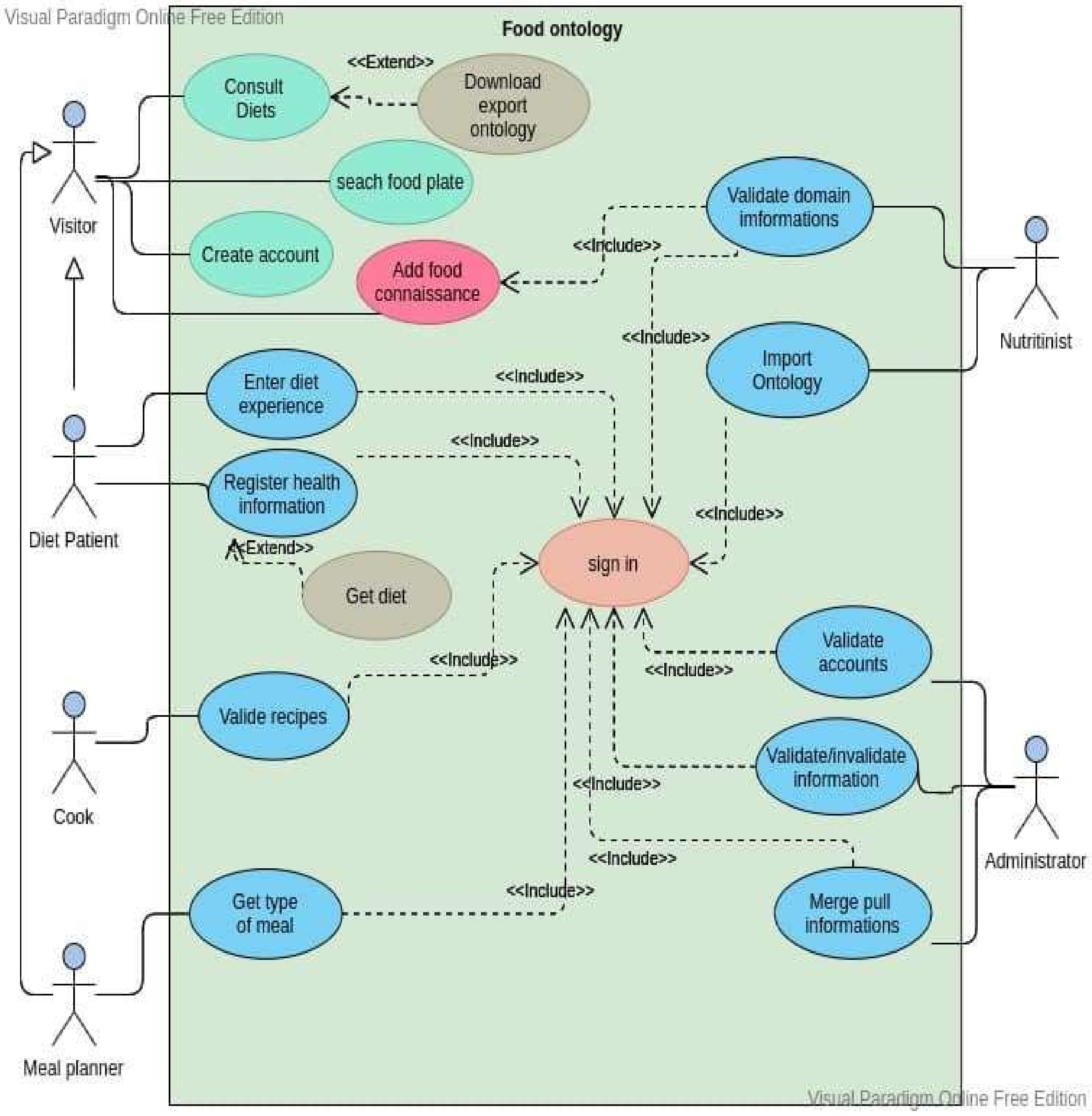


Figure 3 : use case diagram

Textual description of use cases

The following tables describe in detail the different use case scenarios of our system in a textual formalism.

❖ **Textual description of the use case:** sign in

Name:	sign in
Objective:	Permit a user to asses his own space.
Primary Actors:	Administrator
Secondary Actors:	No secondary actors
Preconditions:	have an account in the website
Scenarios:	1. The system ass the actor to enter his login and password. 2. The actor enters the login and the password. 3. The system verifies the parameters 4. The system displays the space corresponding to the actor. 5. The instance of use case end.
Postconditions:	No post conditions
Exception :	If there is an empty field or the user hasn't been found, the system displays an error message.

❖ **Textual description of the use case:** search food plate

Name:	search food plate
Objective:	Permit a visitor the search a food
Primary Actors:	Visitor
Secondary Actors:	No secondary actors
Preconditions:	The visitor has to be on web site.
Scenarios:	1. The visitor enters the food name's in the search barre and clicks on the button search. 2. The system displays the list of food plate which contain the word the visitor had enter. 3. The instance of the use case end.
Postconditions:	No post conditions
Exception:	No exception

❖ **Textual description of the use case:** add knowledge on Food

Name:	add Knowledge on Food
Objective:	Permit a visitor the enter a plate of food with its description.
Primary Actors:	Visitor
Secondary Actors:	Nutritionist
Preconditions:	And nutritionist have to validate it.
Scenarios:	1. A visitor clicks on the button add a new plate 2. The system displays a form. 3. The visitor field the form and clicks on the button save. 4. The system checks if there is no error on form. 5. The system ends the use case.
Postconditions:	An expert will valid.
Exception :	If there is an empty field or the user hasn't been found, the system display an error message.

❖ **Textual description of the use case:** Consult diets

Name:	Consult diets
Objective:	Permit a visitor to consult list of diets available on website.
Primary Actors:	Visitor
Secondary Actors:	No secondary actors
Preconditions:	Consult the website
Scenarios:	1. The visitor clicks on the button view list of diets. 2. The system displays a page with the lists of diets available in the triple store range by alphabetic order. 3. The instance of the use case end.
Postconditions:	User have to be on website.
Exception :	No exception

❖ **Textual description of the use case:** Create account

Name:	Create account
Objective:	Permit a visitor to become a member
Primary Actors:	Visitor
Secondary Actors:	Administrator
Preconditions:	Consult the website

Scenarios:	1. The visitor chooses to create an account. 2. The system displays the form. 3. The visitor field the form. 4. The system verifies enter the information. 5. The system displays the space of the actor. 6. The use instance finish.
Postconditions:	An administrator will valid the account.
Exceptions :	if there is and empty field or a mistake in filed the system displays an error message.

❖ **Textual description of the use case:** Download export ontology

Name:	Download export ontology
Objective:	Permit to a visitor to export and download an ontology.
Primary Actors:	Visitor
Secondary Actors:	No secondary actor
Preconditions:	The visitors have to be on website
Scenarios:	1. The visitor chooses the ontology his want to download, choose the place and the format where his want to save it and chicks on the button export. 2. The system downloads the ontology in a chosen format at the place the visitor has chooses. 3. The user instance end.
Postconditions:	No post conditions
Exception :	If the is not enough place in the user computer the system return and error message.

❖ **Textual description of the use case:** Validate domain information

Name:	Validate domain information
Objective:	Permit a nutritionist to validate a plate of food a visitor had enter
Primary Actors:	Nutritionist
Secondary Actors:	Visitor
Preconditions:	A visitor had entered a food. The nutritionist have to be connect on his account.
Scenarios:	1. The nutritionist clicks on button validate plates of food.

	2. The system displays the list of food to validate. 3. The nutritionist clicks on button good to permit a pate of foot to been chows to visitors or button no good to delete the plate of food. 4. The system ends the instance.
Postconditions:	No post conditions
Exception :	No exception.

❖ **Textual description of the use case:** import ontology

Name:	import ontology
Objective:	Permit a food nutritionist to import an otology in the system
Primary Actors:	Nutritionist
Secondary Actors:	Visitor
Preconditions:	The nutritionist have to be connect on his account.
Scenarios:	1. The nutritionist clicks on button import an ontology. 2. The system displays the page to choose the ontology. 3. The nutritionist clicks on button import. **4.** The system ends the instance.
Postconditions:	No post condition.
Exception :	If the ontology is empty the system displays an error message.

❖ **Textual description of the use case:** Enter diet experience

Name:	Enter diet experience
Objective:	Permit a Diet patient ta narrate his story.
Primary Actors:	Diet patient
Secondary Actors:	Administrator
Preconditions:	Sign in as a diet patient
Scenarios:	1. The diet patient clicks on button Enter an experience. 2. The system displays form with a button save. 3. The diet patient clicks on button save. 4. The system displays the message saved. **5.** The system ends the instance.
Postconditions:	An administrator will validate
Exception :	if there is and empty field or a mistake in filed the system displays an error message.

❖ **Textual description of the use case:** register health information

Name:	register health information
Objective:	Permit a Diet patient to enter his health information.
Primary Actors:	Diet patient
Secondary Actors:	Administrator
Preconditions:	Sign in as a diet patient.
Scenarios:	1. The diet patient clicks on button Enter an experience. 2. The system displays form with a button save. 3. The diet patient clicks on button save. 4. The system displays the message saved. **5.** The system ends the instance.
Postconditions:	An administrator will validate.
Exception :	if there is and empty field or a mistake in filed the system displays an error message.

❖ **Textual description of the use case:** Get diets

Name:	Get diets
Objective:	Permit a Diet patient to Get the type of food to eat in order to satisfy calories prescribed by a doctor.
Primary Actors:	Diet patient
Secondary Actors:	No secondary actors
Preconditions:	Sign in as a diet patient, has enter his health information.
Scenarios:	1. The diet patient clicks on button Get diets. 2. The system displays a diets base of his health information. **3.** The system ends the instance.
Postconditions:	No post conditions
Exception:	No exceptions

❖ **Textual description of the use case:** Validate recipes

Name:	Validate recipes
Objective:	Permit a cook to validate a recipe.
Primary Actors:	Cook
Secondary Actors:	No secondary actors
Preconditions:	The cook have to be connect. A Visitor had to have enter a recipe.
Scenarios:	1. The Cook clicks on button validate recipes.

	2. The system displays the list of food to validate. 3. The Cook clicks on button good to permit a recipe to been chows to visitors or button no good to delete the recipes. **4.** The system ends the instance.
Postconditions:	No post conditions
Exception:	No exceptions.

❖ **Textual description of the use case:** Get type of meal

Name:	Get type of meal
Objective:	Permit a Meal planner to Get the right type of meal to cook for an occasion to satisfy an audience from a particular country
Primary Actors:	Meal planner
Secondary Actors:	No secondary actors
Preconditions:	Meal planner is connected to his account, there is food in triple store
Scenarios:	1. The Meal planner clicks on button type of meal. 2. The Meal planner select a region. 3. The system displays a list of meal type. **4.** The system ends the instance.
Postconditions:	No post conditions
Exception:	If there is no meal type in triple store the system display and error

❖ **Textual description of the use case:** Validate accounts

Name:	Validate accounts
Objective:	Permit an administrator to validate an account
Primary Actors:	Administrator
Secondary Actors:	No secondary actors
Preconditions:	A nutritionist, a cook or a Meal planner had created an account.
Scenarios:	1. The Cook clicks on button validate accounts. 2. The system displays the list of food to validate. 3. The Cook clicks on button good to permit a accounts to been chows to visitors or button no good to delete the accounts. **4.** The system ends the instance.
Postconditions:	No post conditions

Exception :	No exception

❖ **Textual description of the use case:** Validate/Invalidate information

Name:	Validate/Invalidate information
Objective:	Permit an administrator to validate an information
Primary Actors:	Administrator
Secondary Actors:	No secondary actors
Preconditions:	A visitor, a cook or a Meal planner had added an information.
Scenarios:	1. The Cook clicks on button validate information. 2. The system displays the list of food to information. 3. The Cook clicks on button good to permit an information to been chows to visitors or button no good to delete the information. The system ends the instance.
Postconditions:	No post conditions
Exception:	No exceptions

❖ **Textual description of the use case: Merge pull information**

Name:	Merge pull information
Objective:	Save information to the main Ontology
Primary Actors:	Administrator
Secondary Actors:	No secondary actors
Preconditions:	Food Expert should have added new information and send a pull request
Scenarios:	1. A visitor clicks on the button add a new plate 2. The system displays a form. 3. The visitor field the form and clicks on the button save. 4. The system checks if there is no error on form. **5.** The system ends the use case.
Postconditions:	No post conditions
Exception:	

7. Description of use cases using sequence diagrams

To graphically represent the interactions between the actors and the system in order to allow an actor to add knowledge to our ontology or to search for a dish in our system we have used the following sequence diagrams.

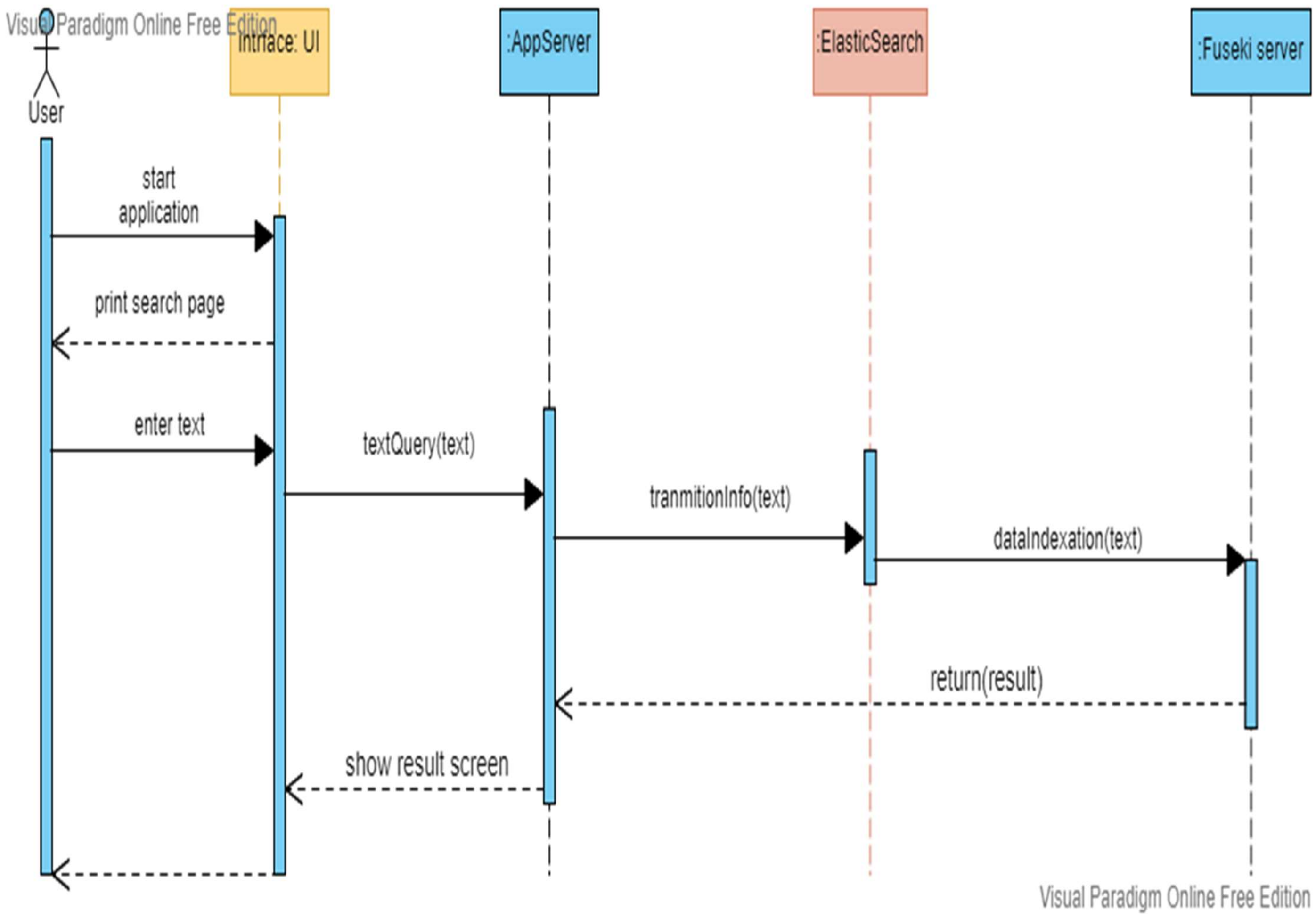
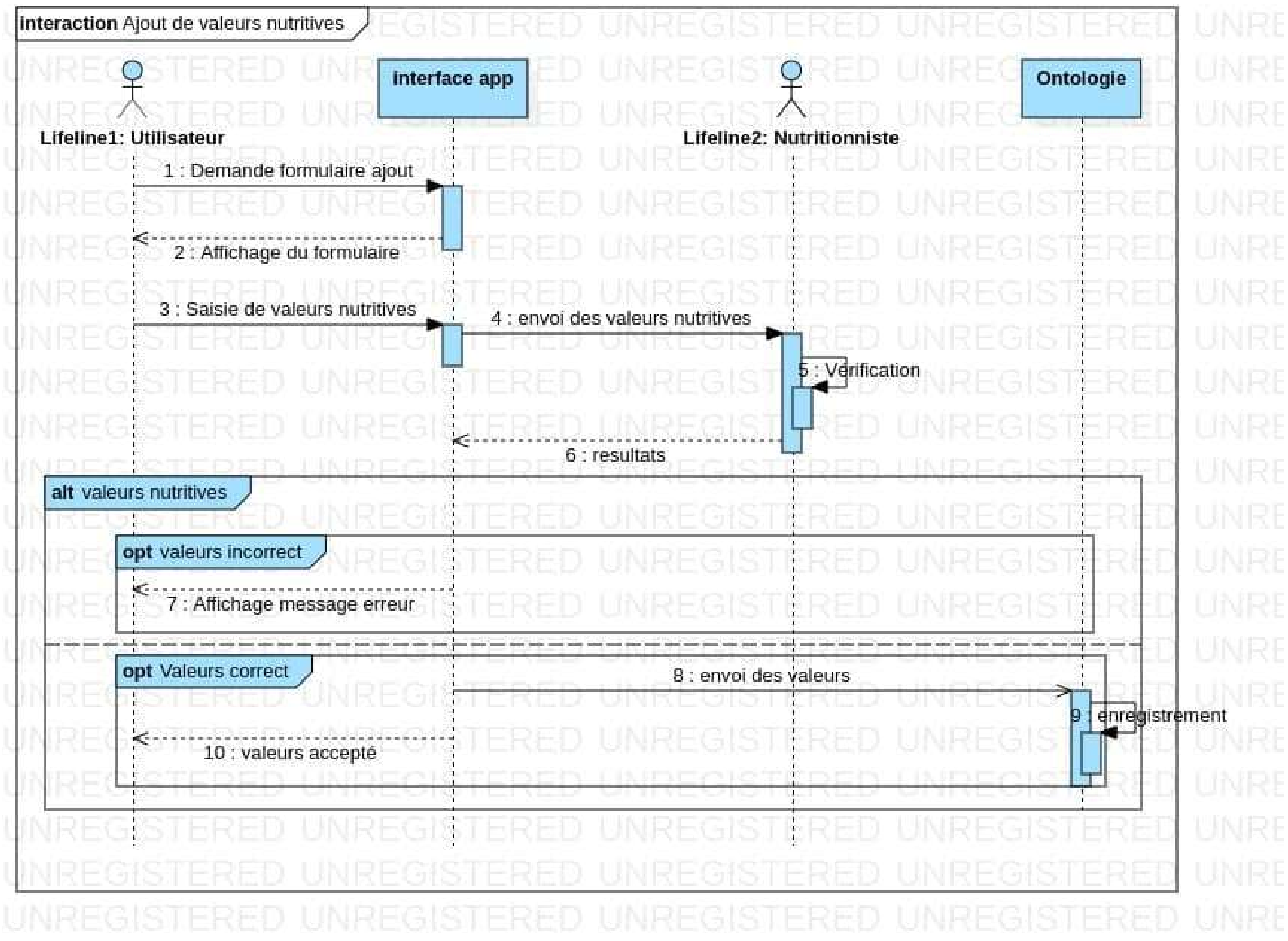


Figure 4: sequence diagram for searching knowledge on a particular Food



interaction Ajout de valeurs nutritives
interface app
Ontologie
Lifeline1: Utilisateur
Lifeline2: Nutritionniste
1 : Demande formulaire ajout
2 : Affichage du formulaire
3 : Saisie de valeurs nutritives
4 : envoi des valeurs nutritives
5 : Verification
6 : resultats
alt valeurs nutritives
opt valeurs incorrect
7 : Affichage message erreur
opt Valeurs correct
8 : envoi des valeurs
9 : enregistrement
10 : valeurs accepté
UNREGISTERED

Figure 5 sequence diagram adds knowledge of food to Ontology

8. State machine diagrams of the actors

This state machine represents the different state of actors in our system, it shows how actors can change from one state actor to another during their interaction with our application

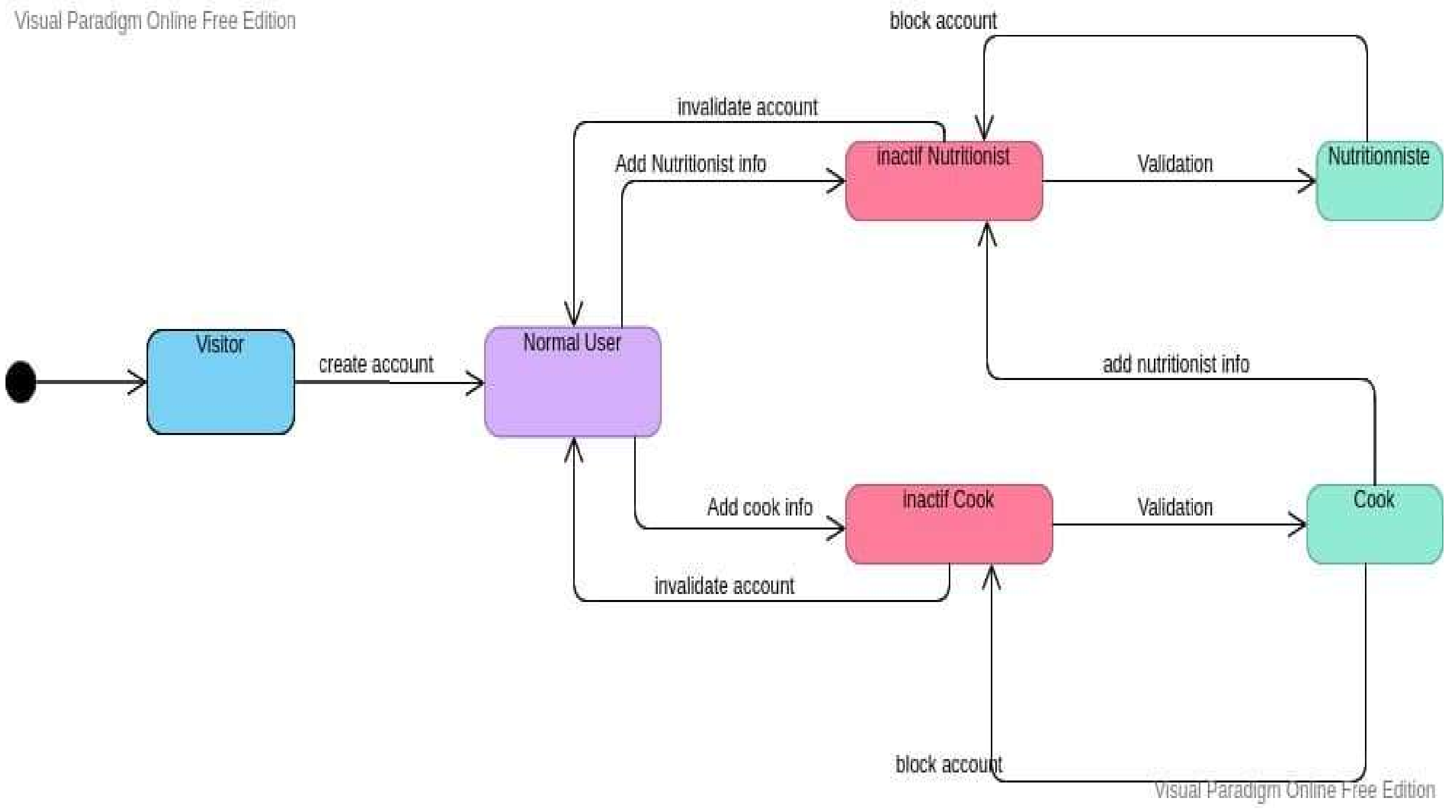


Figure 6: State machine diagram

9. Product Backlog

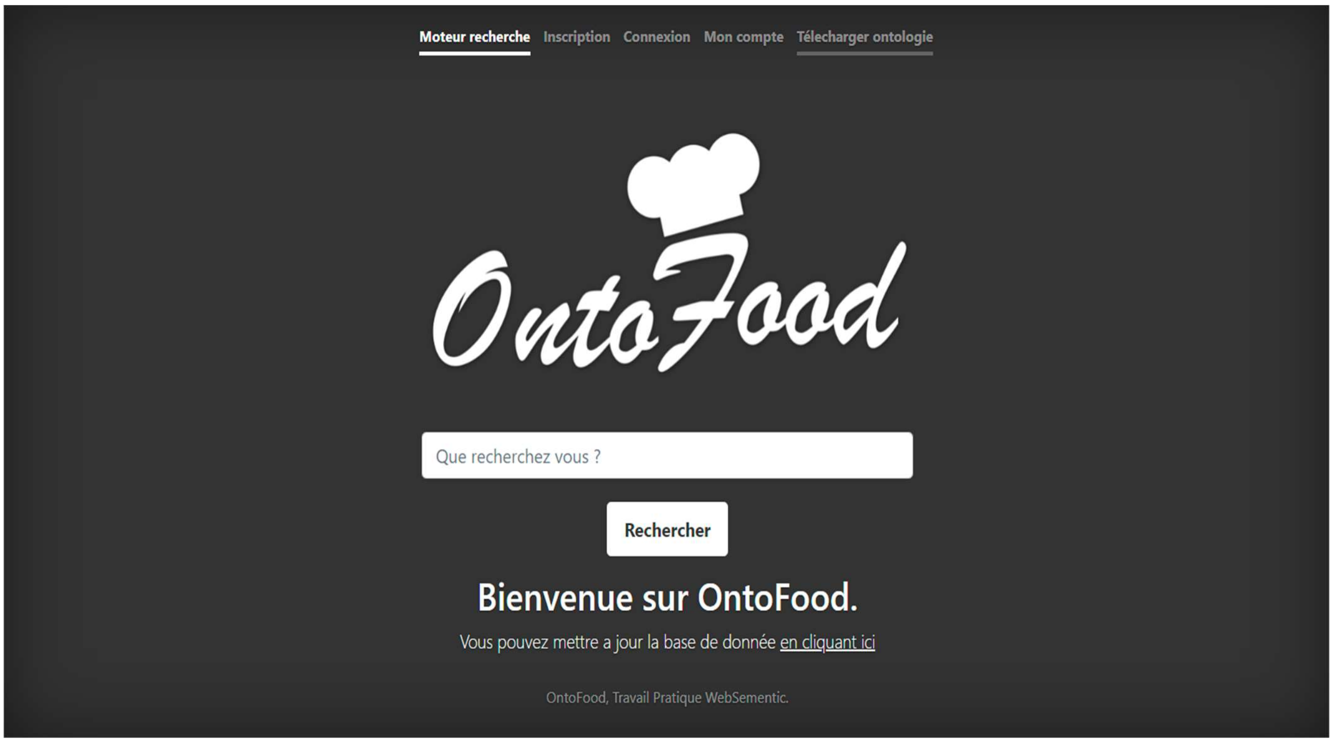
ID	User Story	Acceptance Criteria	Priority	Initial Estimate	Adjustment Factor	Adjustment Estimate
1	***Create user interface*** *As a user of the system, I should be able to access the application using a web interface.*	*When a user opens the platform, they get an interface to interact with*	***1***	***2***		
2	***Create Signup, Login and Logout.*** *As a user of the system, I should be able to create an account on the system*	When a user signup on the system they get access to system functionality	***2***	3		
3	***Register Profile information.*** As a Nutritionist, Cook or Diet Patient, I want to be able to enter my information into my profile in order	When a user has already registered in the system, they can fill in information about their profile	***3***	5		
4	***Invitation by Email.*** *As a user of the system, I want to be able to invite people to subscribe to the platform via email in order to inform others of the existence of such a platform*	When the user is in the system, they can send invitation link to other users who are not yet users of the system via email invite links	***5***	5		
5	***Add, modify and delete knowledge about a meal*** *As a user of the system, I should be able to add, modify and delete knowledge about a meal.*	*An authorized user of the system when logged in is able to add, modify and delete information about a meal*	***8***	5		
6	***Query Execution*** *As a user of the system in order to get vital information about a meal I want to be able to run queries on the system.*	*Any user who accesses the system can ask questions in the form of queries and the system will response to them*	***7***	8		
7	***Search for a meal.*** As a user of the system, in order to get useful information about a meal eaten in a particular area, its	*A user can search for a meal and get information about where it is mostly*	***9***	5		

	calory value, I should be able to search for a meal	*eaten and its nutritional value*				
8	***Modify knowledge of an Existing Ontology*** As a Nutritional expert or User with experience In order to provide more accurate information about a meal, I should be able to modify an existing ontology.	*An authorized user of the system logs in, they can modify knowledge about a food*	***6***	**5**		
9	***Verification and validation of Accounts*** As a system administrator, in order to know if someone is a nutritional expert, I should be able to verify and validate their account.	*An administrator logs into the system, and clicks on the verify account tab, they can see and validate creation of various pending account profiles*	***4***	**5**		
10	***Verification and Validation of Addition and Modifications of Otology.*** As a system administrator in order to prevent everyone from adding false information to the ontology, I want to be able validate all pull request to ontology before it can be saved to the system.	*An administrator logs into the system, and clicks on the verify ontology tab, they can read and validate various pending ontologies*	**10**	**8**		

10. User Interface Design

– Home Page:

This constitutes the first page a user sees when he/she Opens our ontology platform. It constitutes a search bar which can be used to search for information.



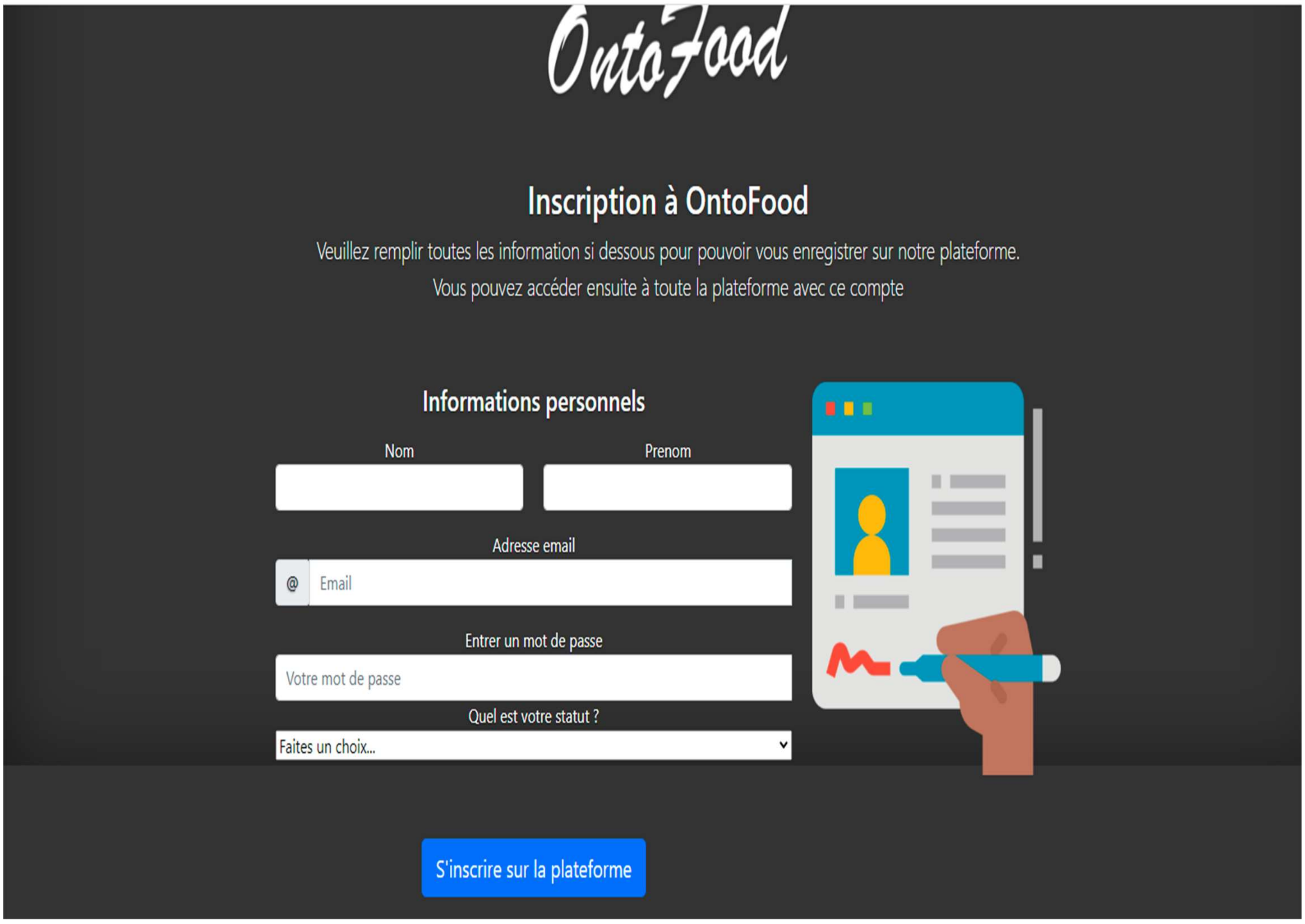
Bienvenue sur OntoFood.

Vous pouvez mettre a jour la base de donnée en cliquant ici

OntoFood, Travail Pratique WebSementic.

– Registration Page:

This is the page a user sees when trying to register on our platform. They are presented with a form where they can fill in their information


OntoFood

Inscription à OntoFood

Veuillez remplir toutes les information si dessous pour pouvoir vous enregistrer sur notre plateforme.
Vous pouvez accéder ensuite à toute la plateforme avec ce compte

Informations personnels

Nom	Prenom

Adresse email

@ Email

Entrer un mot de passe

Votre mot de passe

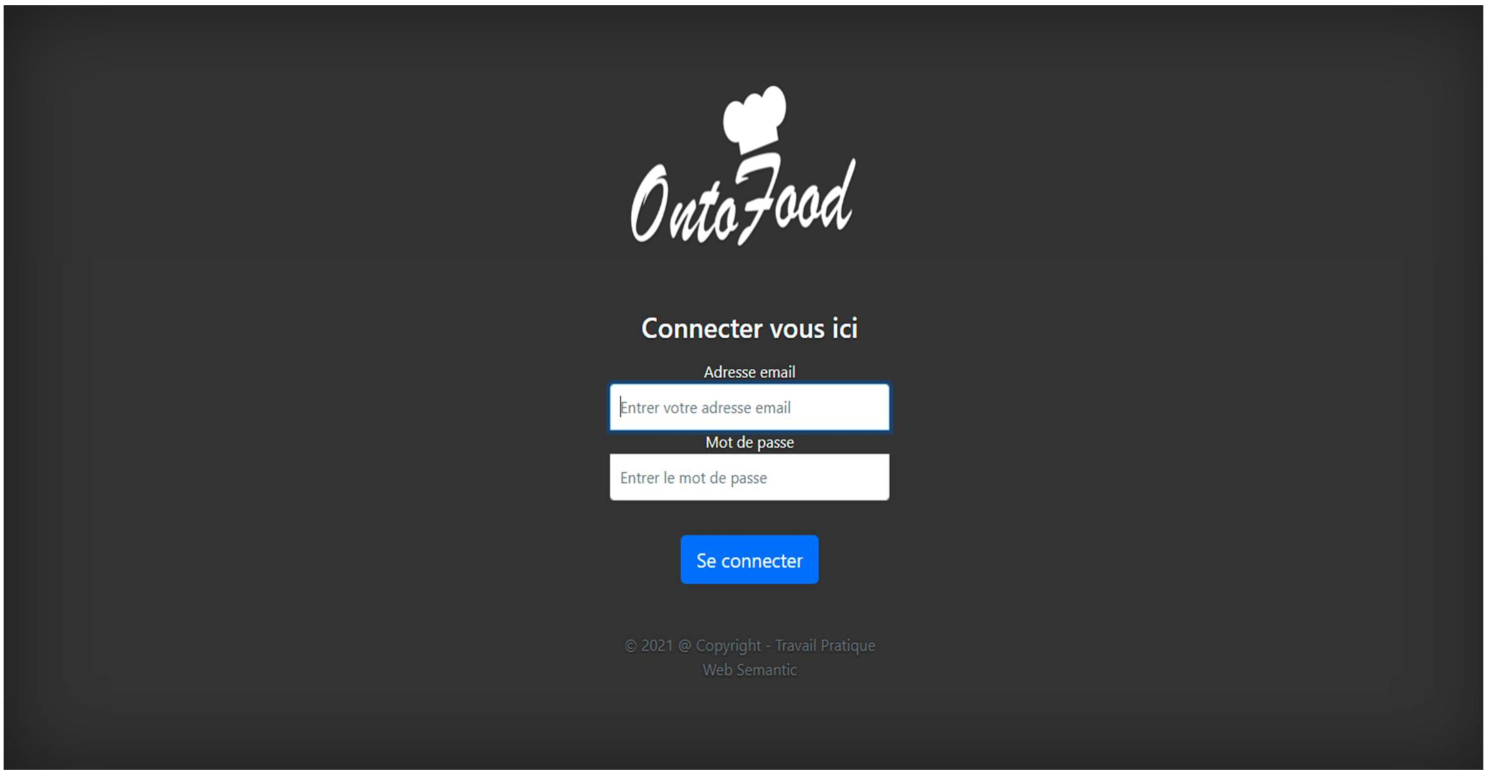
Quel est votre statut ?

Faites un choix...

S'inscrire sur la plateforme

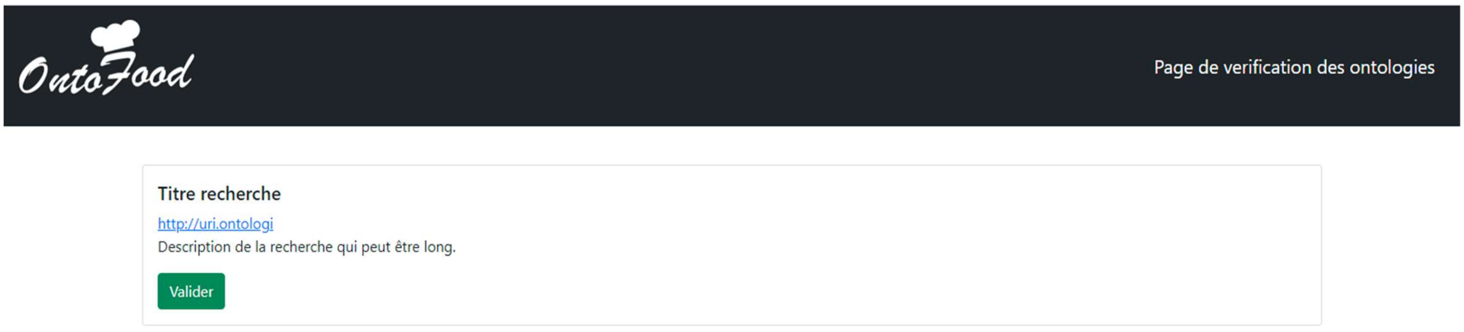
– **<u>Login page:</u>**

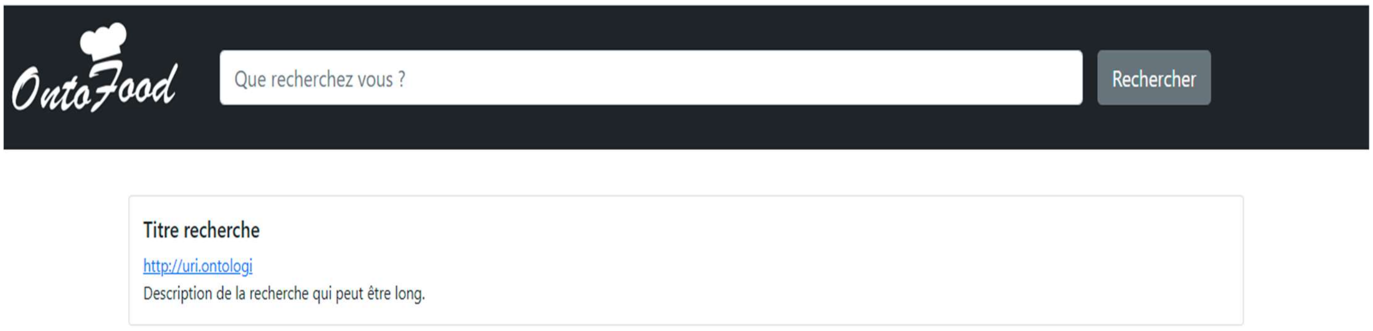
This page is responsible for authenticating users and granting them access to their accounts.



– **<u>Ontology Verification page.</u>**

This is the page used by the administrator to verify ontology send to the platform before it can be added to the main Ontology.


OntoFood
Page de verification des ontologies
Titre recherche
http://uri.ontologi
Description de la recherche qui peut être long.
Valider



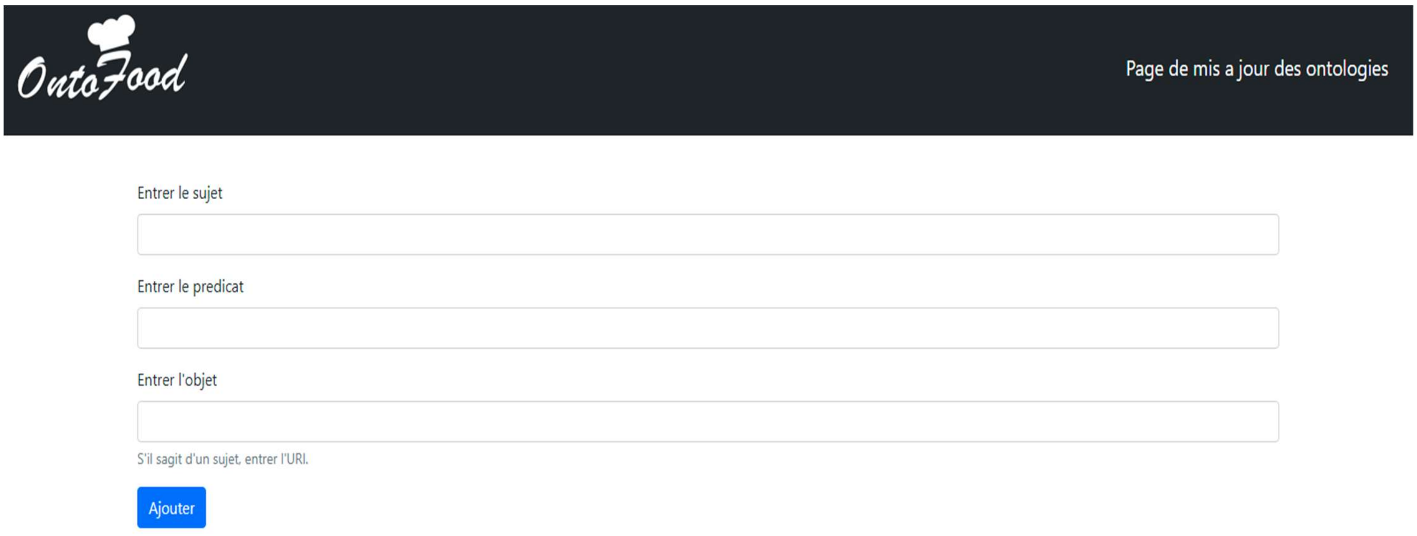
Titre recherche

http://uri.ontologi

Description de la recherche qui peut être long.

- **Add Ontology page:**

This page is used by the Food Expert (Nutritionist or Cook) to add new knowledge about a food to the platform.



Entrer le sujet

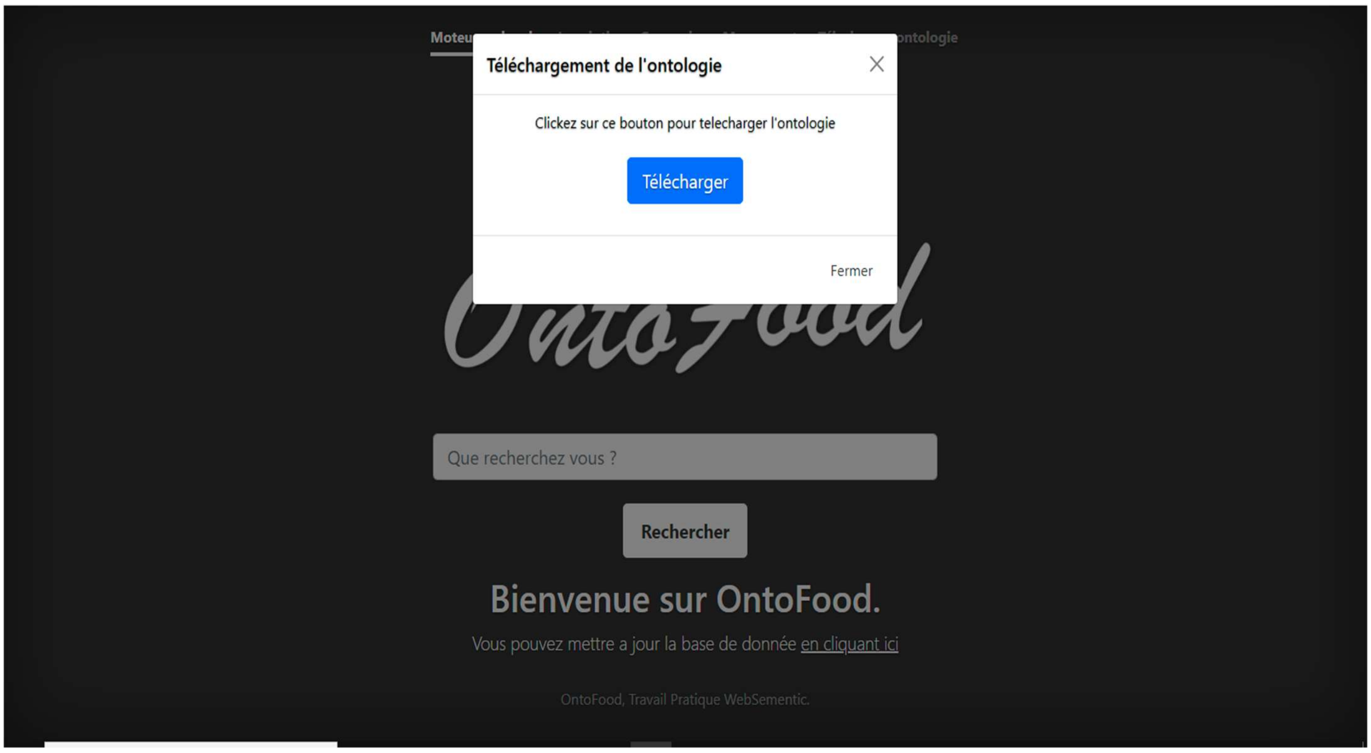
Entrer le predicat

Entrer l'objet

S'il sagit d'un sujet, entrer l'URI.

– **Download Ontology page:**

This page offers the ability for any person interested in our knowledge base to download it


Téléchargement de l'ontologie
Clickez sur ce bouton pour telecharger l'ontologie
Télécharger
Fermer
OntoFood
Que recherchez vous ?
Rechercher

Bienvenue sur OntoFood.

Vous pouvez mettre a jour la base de donnée en cliquant ici

OntoFood, Travail Pratique WebSementic.

11. Sprint Backlog

Release	***Sprint***	***ID of User Stories***	***Period***
Release 1: Early Release	Sprint 1	1,3,2	12th April 2021 – 26th April 2021
	Sprint 2	2,5	27th April 2021 – 11th May 2021
Release 2: Mid Release	Sprint 3	4,6,9	12th May 2021 – 26th May2021
	Sprint 4	7,8	27th May 2021 – 10th June 2021
Release 3: Final Release	Sprint 5	10	11th June 2021 – 20th June 2021

III. Software design

12. DevOps architecture for the system

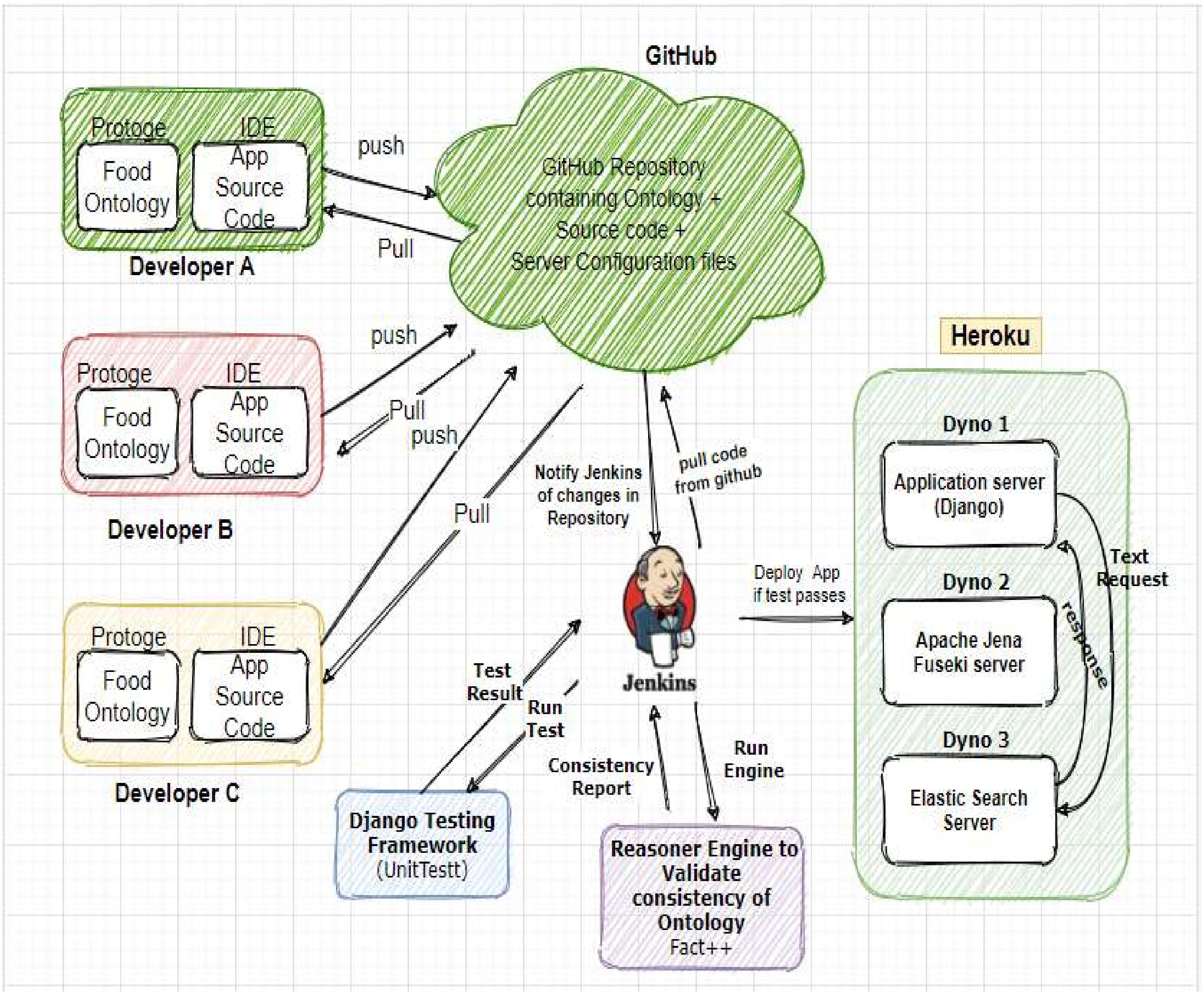


Figure 7:DevOp Architecture of Food Ontology Platform

– Description of Architecture

The Development and Operations (**DevOp**) architecture consist of the following components

I. **Developers**: In the development of our system we have 4 developers constituting the group members who develop the application in Django(python) using Various IDE such as **VScode**, **Sublime** and **Pycharm**. We also develop the food ontology using **Protégé**. After developing a new feature, the make a push to the main code base on GitHub.
II. **GitHub Repository:** GitHub serves as the main code base for our project. It stores all the ontology and source code for the project. It also helps in our DevOp pipeline by serving as a trigger to our Jenkins server whenever a new code is committed.
III. **Jenkins server:** it Serves as a tool for doing Continuous integration that is automatically running test whenever new code is added to the repository and Continuous deployment by automatically sending code that has passed all test to Production server (Heroku) where it is deployed
IV. **Testing server (Unittest and Fact ++ Reasoner Engine):** Our test Server runs two types of Test
    a. **Unittest:** which is pythons default testing framework. It is used to test the source code of our Django application
    b. **Fact ++ Reasoner Engine:** This is an engine that test for the correctness and consistency of an ontology
V. **Heroku:** Serves as our production server where our application will be deployed so the public can access.

13. Architecture of Food Ontology software

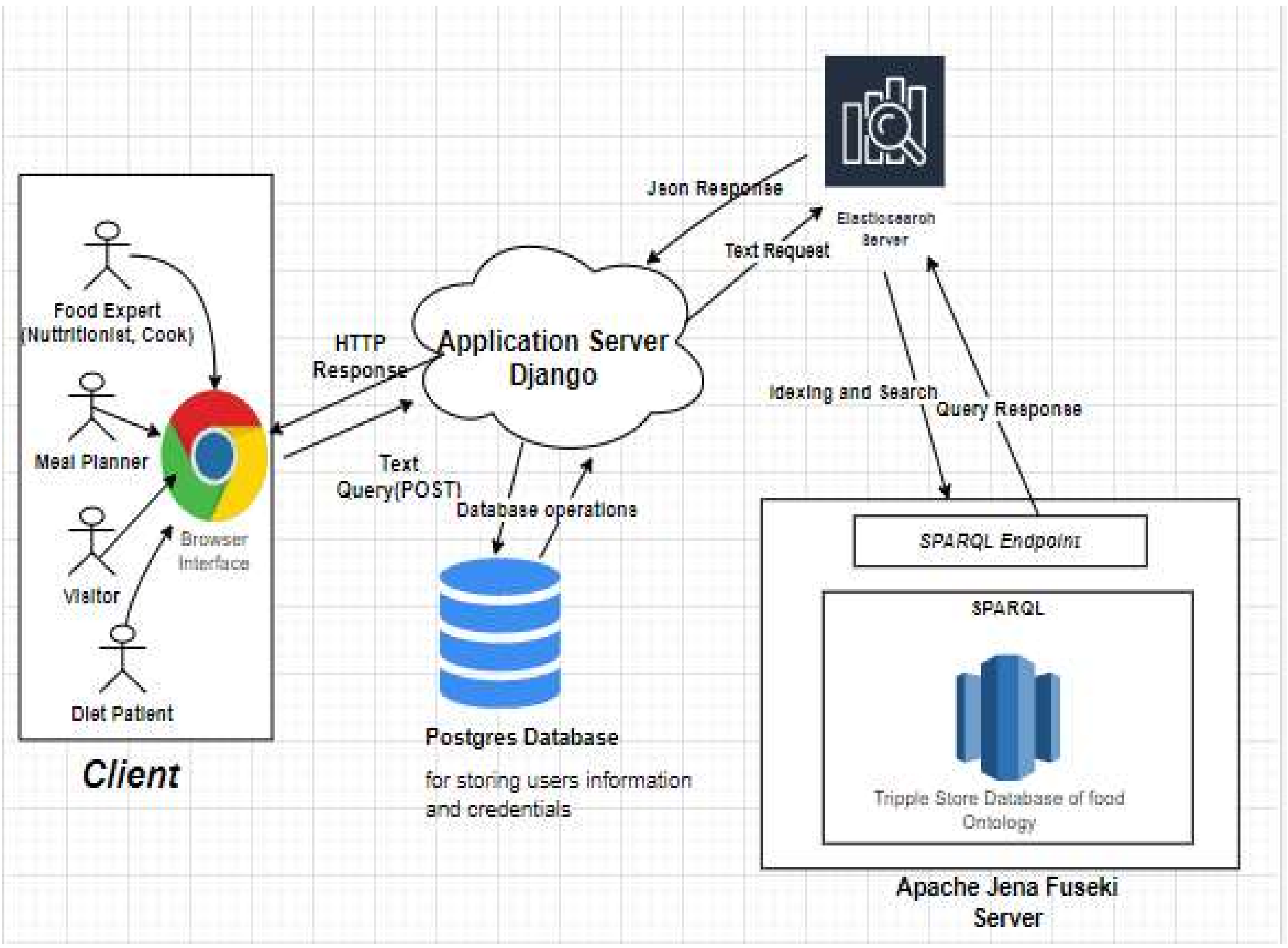


Figure 8: Client-Server Architecture of Food Ontology Platform

– **Description of Architecture**

We use a client server Architecture for our food Ontology Platform which has the following components:

Clients, Food ontolgy App, postgres Database,Apache Jena Fuseki Server and Elastic Search.

It all starts with a client using a browser as an interface to interact with our application. The client sents a text search to our Application, the application forwards this information to our elastic search server which interns uses the SPARQL End Point s Provided by Apache Fuseki server to queary all the information stored in our Ontoloogy.

14. Development Environment

- **Protégé:** is a free open source ontology editor and framework for building intelligent systems. We use it in our project to build our Ontology. And do local consistency checks on the Ontology using Fact ++ Reasoning Engine
- **Django Framework:** it is a high-level python web framework that enables rapid development of secure and maintainable websites. This will help use to construct our Ontology web Application
- **Jena Fuseki Server:** it's a SPARQL server which takes our Ontology file and provides a SPARQL End point for interacting with our Ontology
- **Elastic Search:** Elastic search is a distributed free and open search and Analytics engine for all types of data, including textual, numerical, Geospatial, structured and unstructured. It will help us to do text search and phrase search on our ontology

- **Kibana elastic search Stack:** kibana provides an interface to our elastic search instance and offers the possibility of having a graphical view of our Ontology
- **Postgres Database:** it’s a free and open source database management server which will be used in our system to store user login credentials and other user information such as profile