Addis Ababa Institute of Technology

Department of IT/SW Eng.

Yet[Lebla]

(version 1.0)

" Find the best food with cheaper price!"

Software Design Specification

Team Members

1. Tibebeselasie Mehari
2. Robel Ephraim
3. Natnael Sisay
4. Yosef Worku

Supervisor Name Mr. Natnael Argaw

Table of Contents

List of figures

Acronyms

SRS	**S**oftware **R**equirements **S**pecification
AAiT	**A**ddis **A**baba **I**nstitute of **T**echnology
FR	**F**unctional **R**equirement
QA	**Q**uality **A**ttribute
UC	**U**se **C**ase
OOP	**O**bject **O**riented **P**rogramming
API	**A**pplication **P**rogramming **I**nterface
JS	**J**ava**S**cript
HTML	**H**yper **T**ext **M**arkup **L**anguage
CSS	**C**ascading **S**tyle**s**heet
JSON	**J**ava**S**cript **O**bject **N**otation
OS	**O**perating **S**ystem
UI	**U**ser **I**nterface
URL	**U**niform **R**esource **L**ocator
HTTP	**H**yper **T**ext **T**ransfer **P**rotocol
MEN	**M**ongoDB **E**xpress **N**ode

1. Introduction

1.1 Purpose

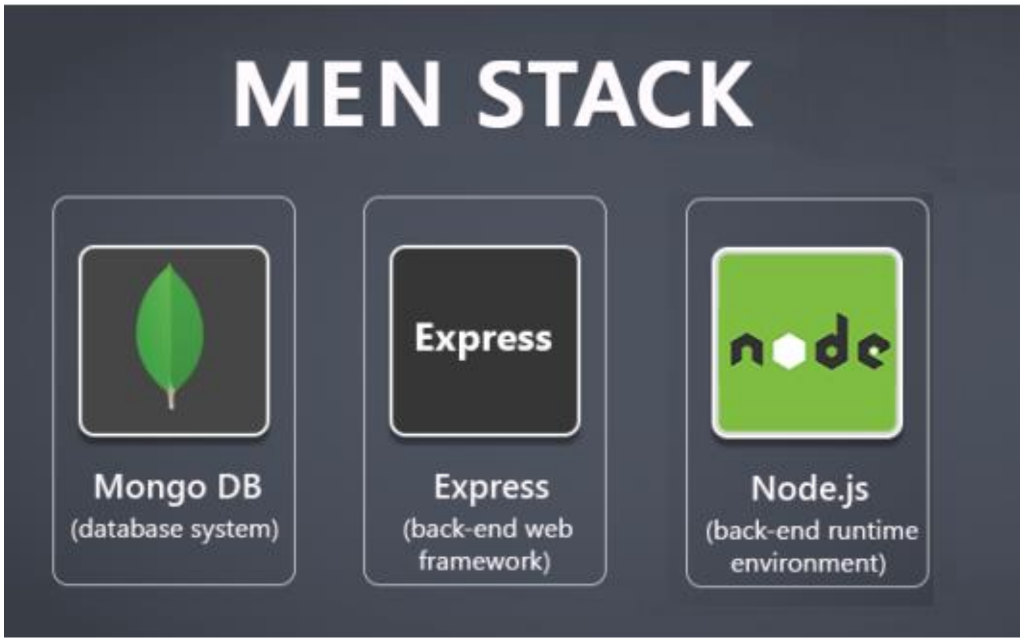
The purpose of this System Design document is to translate the business requirements and business processes into a technical design that will be used to develop the application.

1.2 General Overview

Websites are one of the most effective ways to present and display information to multitude of audiences. In previous times a website was constructed with raw HTML document and served by simple HTTP request. The revolution that started by the introduction of CSS and JavaScript made the web application technology become more and more convenient in reaching multiple audiences. In this period, developing server side of a web application required developers to be keen in one of many programming languages, which are PHP, Java, C#. Else, JavaScript is the primary scripting language to make reactive websites. Hence, in this project the team aims to develop a web application by primarily using the JavaScript technology.

1.3 Development Methods & Contingencies

In the development of this application, JavaScript is the consistent language we will be using throughout. We will be using three JavaScript oriented technologies. MongoDB is the database, Express is backend web framework and Node.js is the back-end runtime environment. The reason why the stack is selected because developers will have improved view of greater picture by understanding the different areas and how they fit together. Frameworks and libraries are no longer limited in supporting user interfaces but also can be combined with other layers of applications. However, wrong architecture, low-level design and unbalance foundation in initial construction can dramatically affect the development process.



1.3.1 Node.js

Node.js is a software platform allowing developers to create web servers and build web applications on it. Meanwhile PHP needs to run separate web server program such as Apache or Internet Information Services, Node.js server can be configured when building an application. However, it is not limited in web projects. More phone apps and desktop applications are built on top of Node.js. Node.js provides many utilities' packages in order to create a powerful web server that runs JavaScript on the server side, for example, asynchronous I/O, single-threaded or multi-threaded, sockets, and HTTP connections. When building web applications using other server-side programming languages, such as Java, PHP, and .NET, the server creates a new thread for each new connection. Assumed that there is a popular website receiving more than two million requests daily, developers probably need more servers or even to invest in more hardware. This approach works well as long as the server has enough hardware infrastructure to provide services to the customers. When the number of clients exceed the server's ability, it starts to slow down and the customers have to wait. The single threaded, non-blocking I/O changes how a request is handled by the server. Instead of creating a new thread for each connection, the web server receives client requests and places them in a queue, which is known as Event Queue. The Event Loop picks up one client request from the queue and starts to process it. If the request does not include any blocking I/O operation, the server processes everything. An available thread from the thread collections will be picked up and assigned to the client request. That thread has then processed the blocking I/O, processed the request and created a response to send back to the client. Node servers can support tens of thousands of simultaneous connections. It does not allocate one thread per connection model, but uses a model process per connection, creating only the memory that is required for each connection.

1.3.2 Express

Even though Node.js is a great choice to make a web server, it cannot provide nothing more than a runtime environment with many built-in modules. The default Node API is not able to handle complex applications as route management. A web framework is needed to do the heavy work. This is where Express comes into play, express is a mature and flexible web framework to build web applications on top of Node. By default, the Express framework uses the Pug engine for supporting templates. That is the reason why it is the best choice to render HTML on the server side for large-scale application. It still provides a lot of routine features to deal with including SPA, the RESTful API, and the MEAN stack.

1.3.3 MongoDB

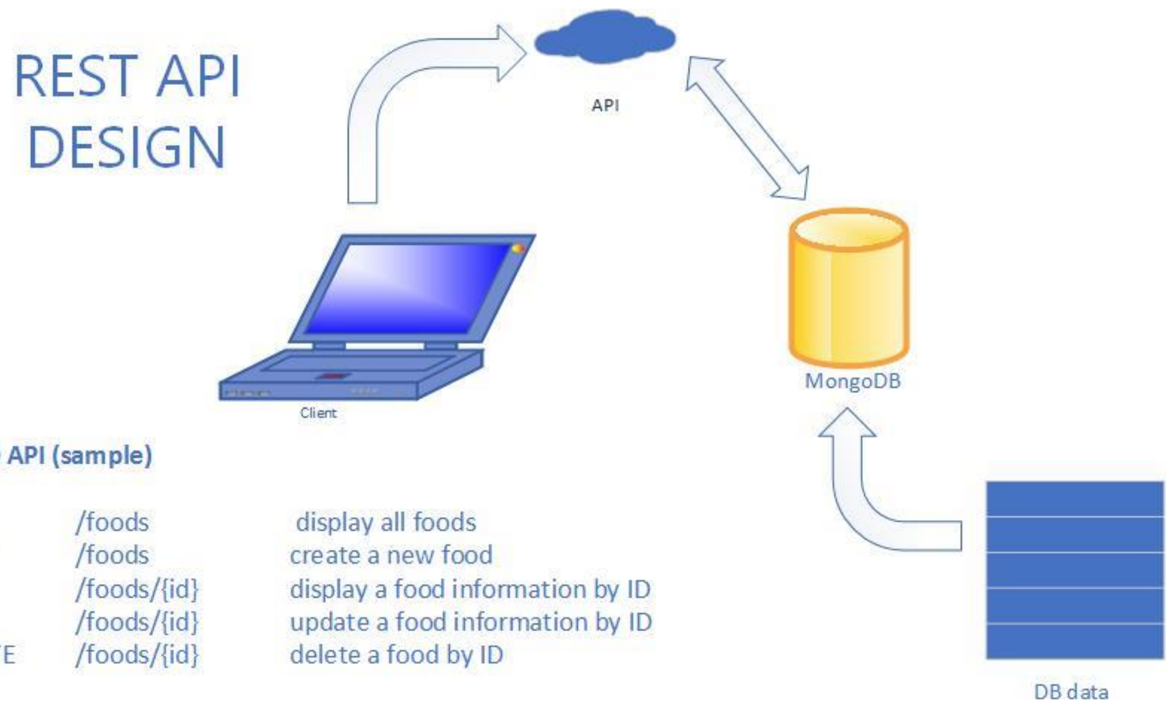
MongoDB is one of the most popular open source NoSQL databases written by C++. MongoDB is document-oriented, and it stores data in the key-value format, using binary JSON (BSON). It is agile, scalable and high performance.

However, MongoDB is schema less and it is not easy to integrate and use with systems. For this reason and others, we prefer to use **Mongoose** Module. Mongoose provides a straightforward, schema-based solution to model an application data unlike other mongo related modules. Mongoose also allows other neat and best features like *populating* (which can be considered as a substitution to the foreign Key relations in Traditional Database Management Systems (RDBMS)

2. System Design Model

2.1 Subsystem decomposition

The following diagram depicts the subcomponent representation and the communication that takes place between each subcomponent.

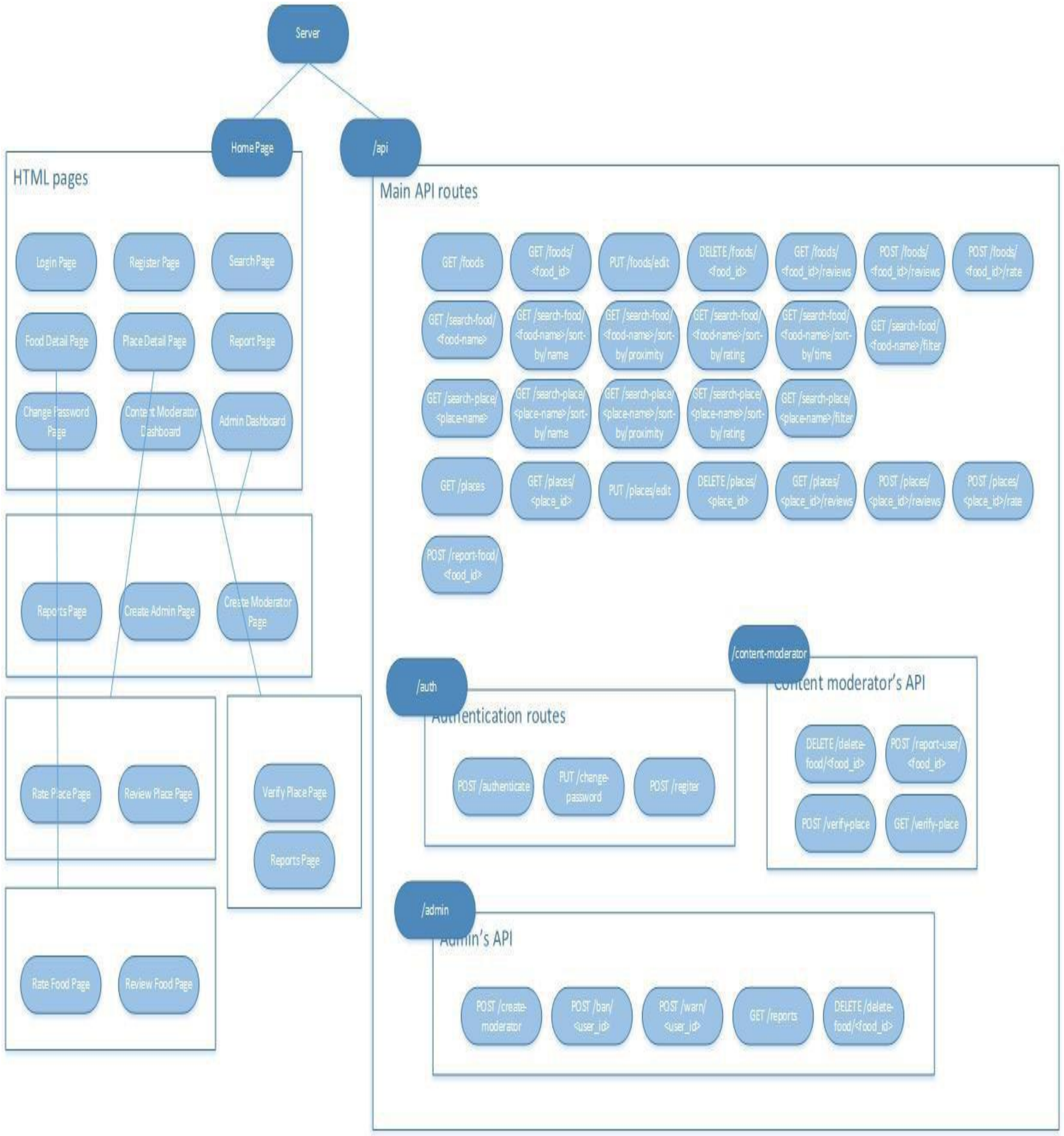


FOOD API (sample)

GET	/foods	display all foods
POST	/foods	create a new food
GET	/foods/{id}	display a food information by ID
PUT	/foods/{id}	update a food information by ID
DELETE	/foods/{id}	delete a food by ID

2.2 Sitemap (API Routes)

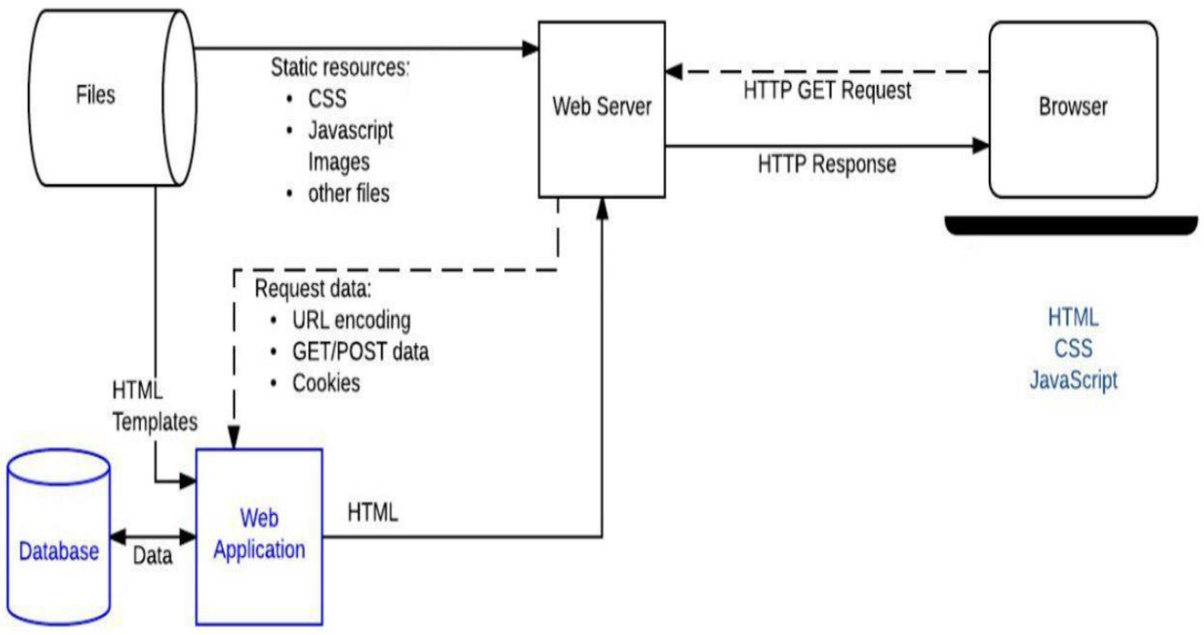
This is how the overall site structure (route points) of the site looks like

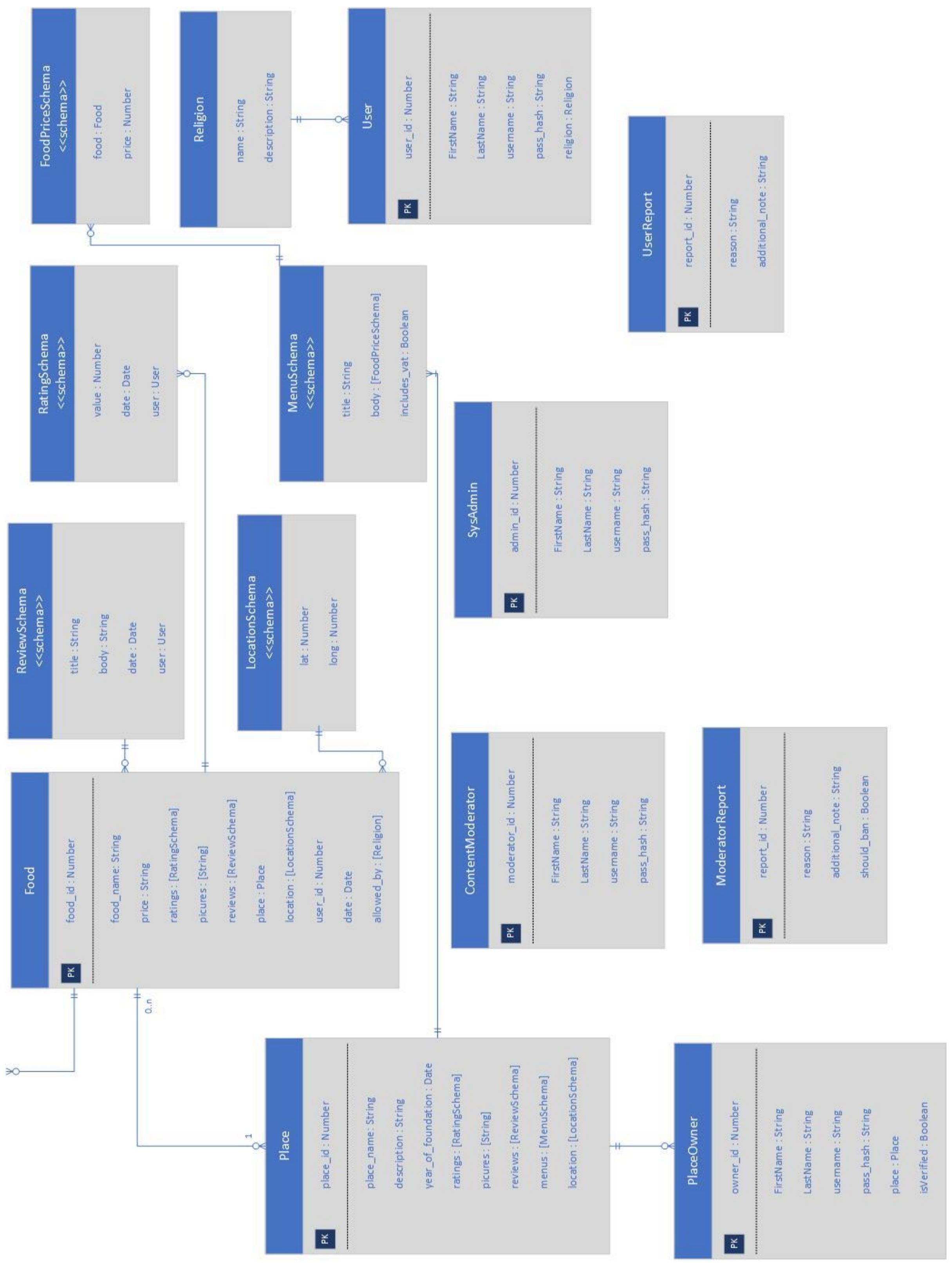

Server
Home Page
/api
HTML pages
Login Page
Register Page
Search Page
Food Detail Page
Place Detail Page
Report Page
Change Password Page
Content Moderator Dashboard
Admin Dashboard
Reports Page
Create Admin Page
Create Moderator Page
Rate Place Page
Review Place Page
Verify Place Page
Reports Page
Rate Food Page
Review Food Page
Main API routes
GET /foods
GET /foods/<food_id>
PUT /foods/edit
DELETE /foods/<food_id>
GET /foods/<food_id>/reviews
POST /foods/<food_id>/reviews
POST /foods/<food_id>/rate
GET /search-food/<food-name>
GET /search-food/<food-name>/sort-by/name
GET /search-food/<food-name>/sort-by/proximity
GET /search-food/<food-name>/sort-by/rating
GET /search-food/<food-name>/sort-by/time
GET /search-food/<food-name>/filter
GET /search-place/<place-name>
GET /search-place/<place-name>/sort-by/name
GET /search-place/<place-name>/sort-by/proximity
GET /search-place/<place-name>/sort-by/rating
GET /search-place/<place-name>/filter
GET /places
GET /places/<place_id>
PUT /places/edit
DELETE /places/<place_id>
GET /places/<place_id>/reviews
POST /places/<place_id>/reviews
POST /places/<place_id>/rate
POST /report-food/<food_id>
/auth
Authentication routes
POST /authenticate
PUT /change-password
POST /register
/content-moderator
Content moderator's API
DELETE /delete-food/<food_id>
POST /report-user/<food_id>
POST /verify-place
GET /verify-place
/admin
Admin's API
POST /create-moderator
POST /ban/<user_id>
POST /warn/<user_id>
GET /reports
DELETE /delete-food/<food_id>

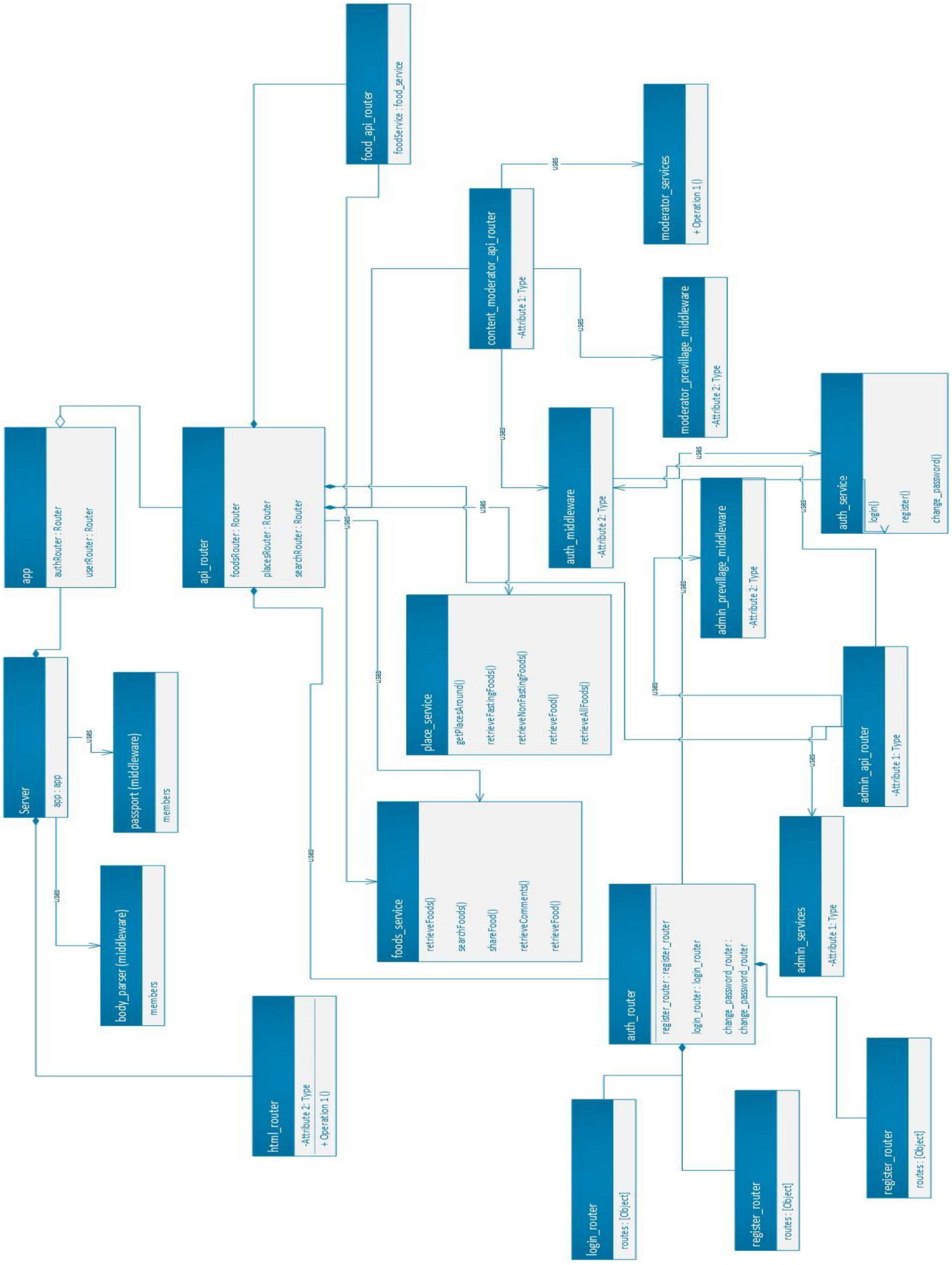
2.3 Application Architecture

REST stands for Representation State Transfer. It is a stateless connection between multiple clients and the server. It is modern client-server architecture using the HTTP protocol to communicate and JSON formatting the data into some meaningful representation. In our system, REST API is used to create stateless interfaces for POST (create), PUT (update), GET (retrieve) and DELETE activities (we chose to use the semantic methods instead of just using POST for every communication).

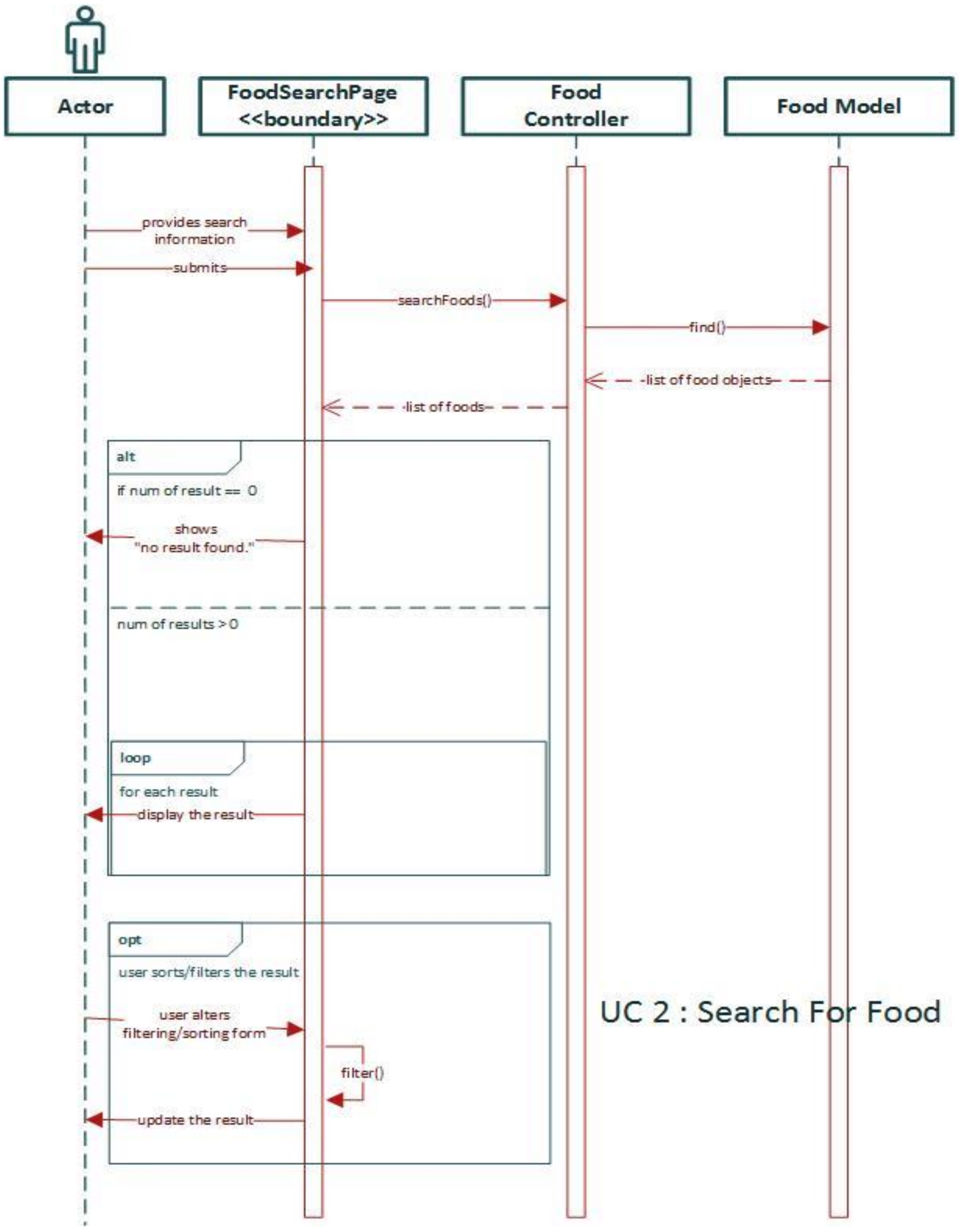
The graph below shows the basic workflow of a REST API.

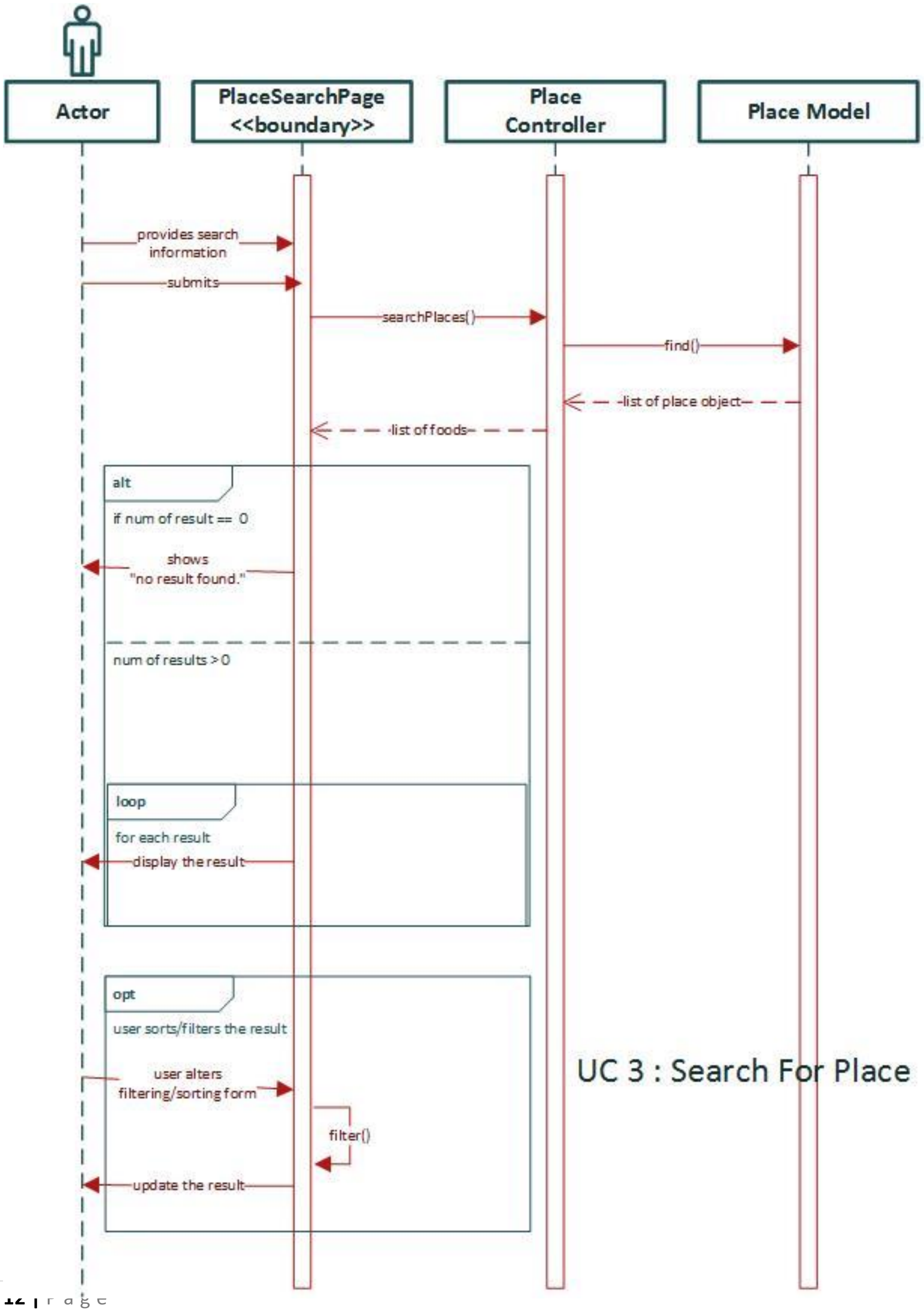



FoodPriceSchema
<<schema>>
food : Food
price : Number
Religion
name : String
description : String
User
PK
user_id : Number
FirstName : String
LastName : String
username : String
pass_hash : String
religion : Religion
RatingSchema
<<schema>>
value : Number
date : Date
user : User
MenuSchema
<<schema>>
title : String
body : [FoodPriceSchema]
includes_vat : Boolean
UserReport
PK
report_id : Number
reason : String
additional_note : String
ReviewSchema
<<schema>>
title : String
body : String
date : Date
user : User
LocationSchema
<<schema>>
lat : Number
long : Number
SysAdmin
PK
admin_id : Number
FirstName : String
LastName : String
username : String
pass_hash : String
Food
PK
food_id : Number
food_name: String
price : String
ratings : [RatingSchema]
picures : [String]
reviews : [ReviewSchema]
place : Place
location : [LocationSchema]
user_id : Number
date : Date
allowed_by : [Religion]
ContentModerator
PK
moderator_id : Number
FirstName : String
LastName : String
username : String
pass_hash : String
ModeratorReport
PK
report_id : Number
reason : String
additional_note : String
should_ban : Boolean
0..n
1
Place
PK
place_id : Number
place_name: String
description : String
year_of_foundation : Date
ratings : [RatingSchema]
picures : [String]
reviews : [ReviewSchema]
menus : [MenuSchema]
location : [LocationSchema]
PlaceOwner
PK
owner_id : Number
FirstName : String
LastName : String
username : String
pass_hash : String
place : Place
isVerified : Boolean

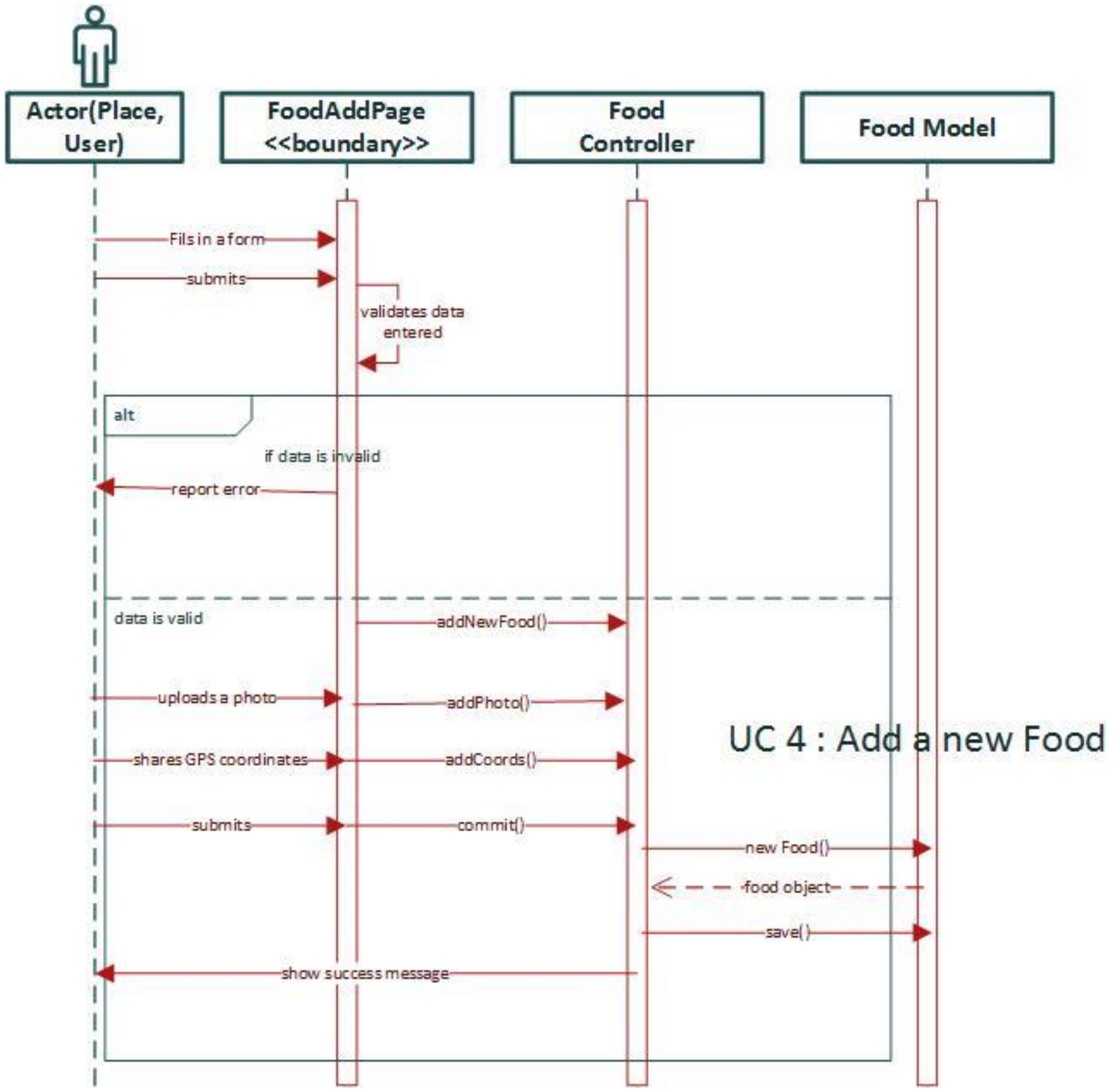

app
authRouter : Router
userRouter : Router
api_router
foodsRouter : Router
placesRouter : Router
searchRouter : Router
food_api_router
foodService : food_service
content_moderator_api_router
-Attribute 1: Type
moderator_services
+ Operation 1 ()
moderator_previllage_middleware
-Attribute 2: Type
auth_middleware
-Attribute 2: Type
auth_service
login()
register()
change_password()
admin_previllage_middleware
-Attribute 2: Type
place_service
getPlacesAround()
retrieveFastingFoods()
retrieveNonFastingFoods()
retrieveFood()
retrieveAllFoods()
foods_service
retrieveFoods()
searchFoods()
shareFood()
retrieveComments()
retrieveFood()
admin_api_router
-Attribute 1: Type
admin_services
-Attribute 1: Type
Server
app : app
passport (middleware)
members
body_parser (middleware)
members
html_router
-Attribute 2: Type
+ Operation 1 ()
auth_router
register_router : register_router
login_router : login_router
change_password_router : change_password_router
login_router
routes : [Object]
register_router
routes : [Object]
register_router
routes : [Object]
uses

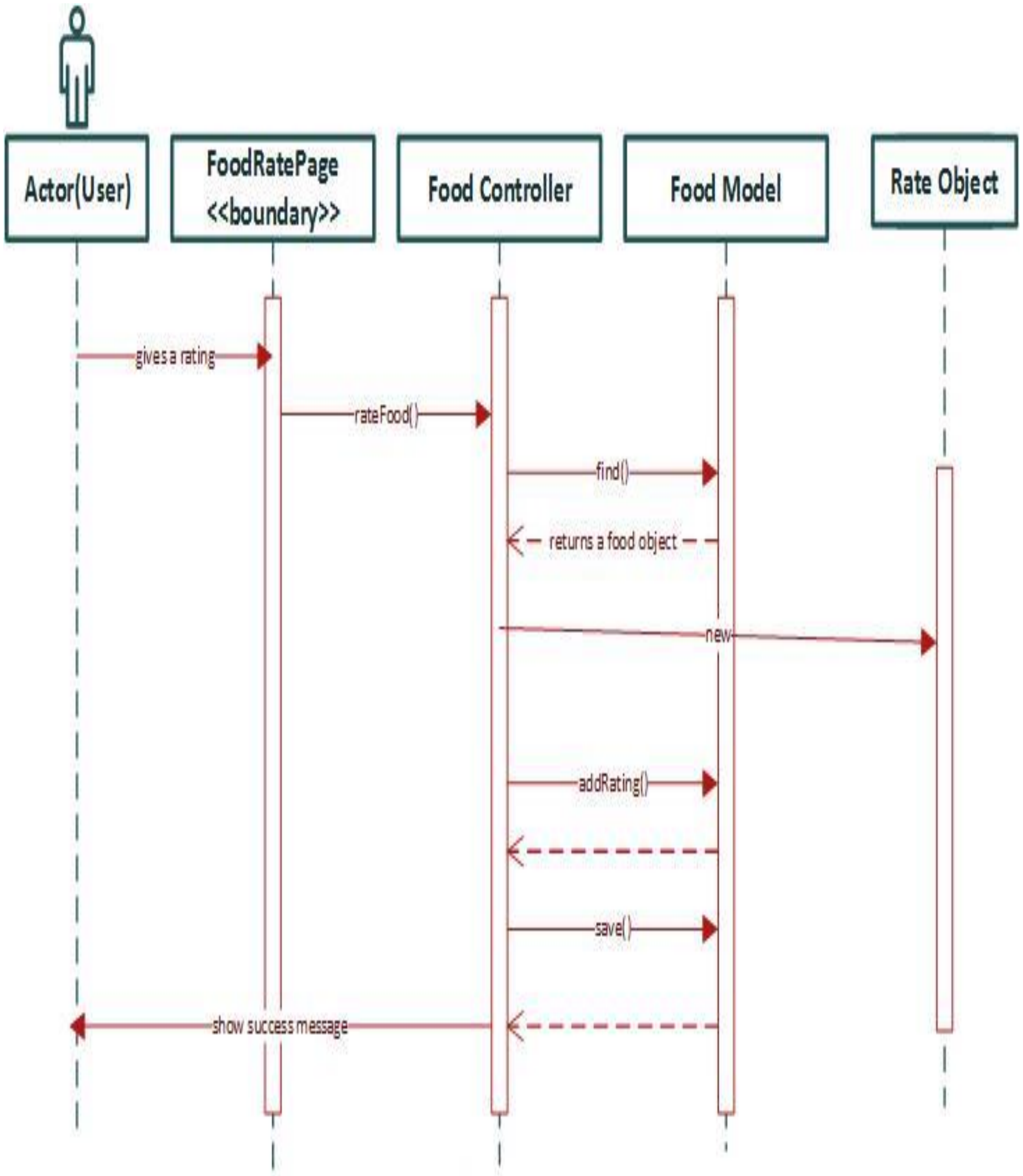
3. Sequence Diagrams



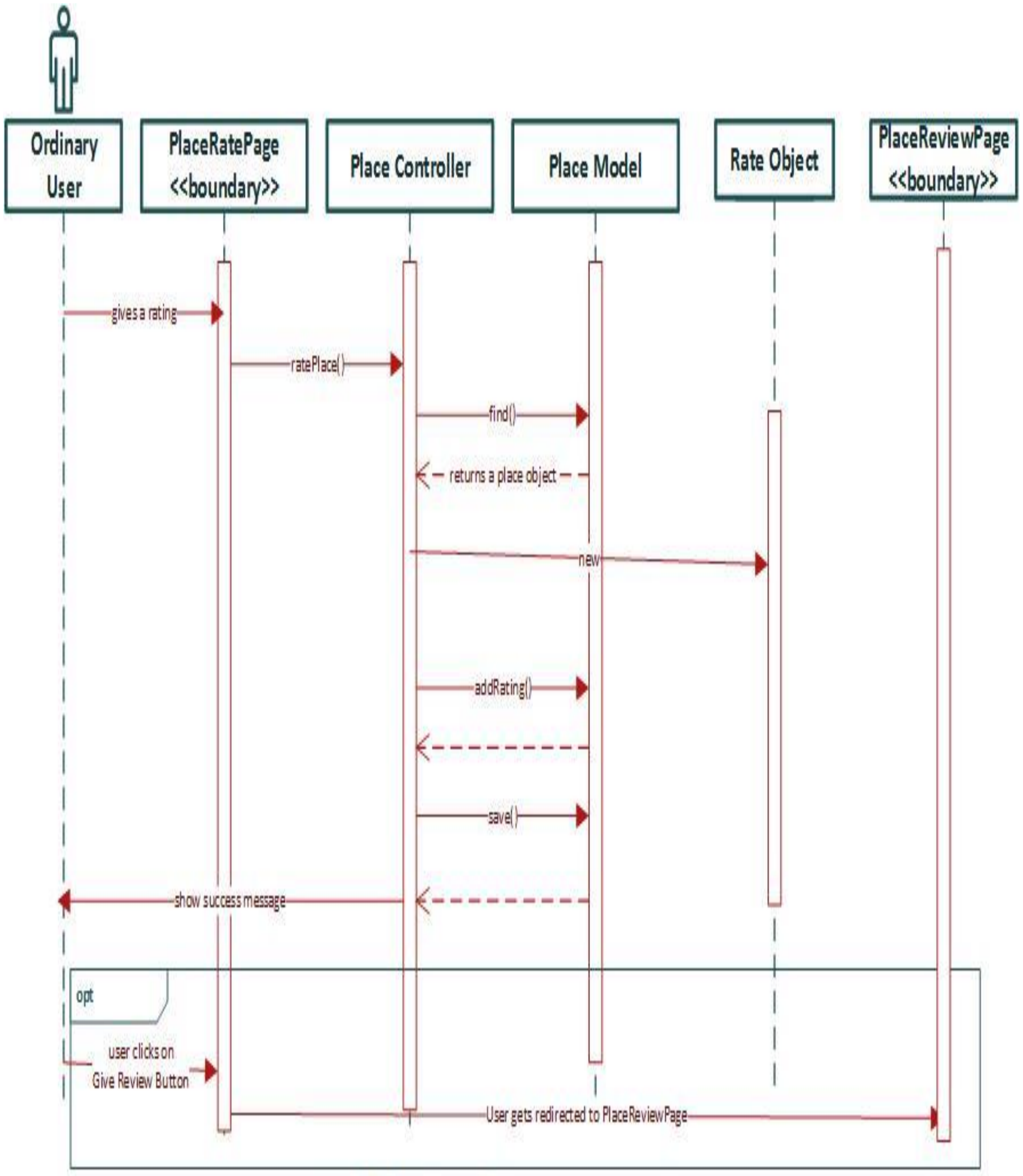


UC 3 : Search For Place

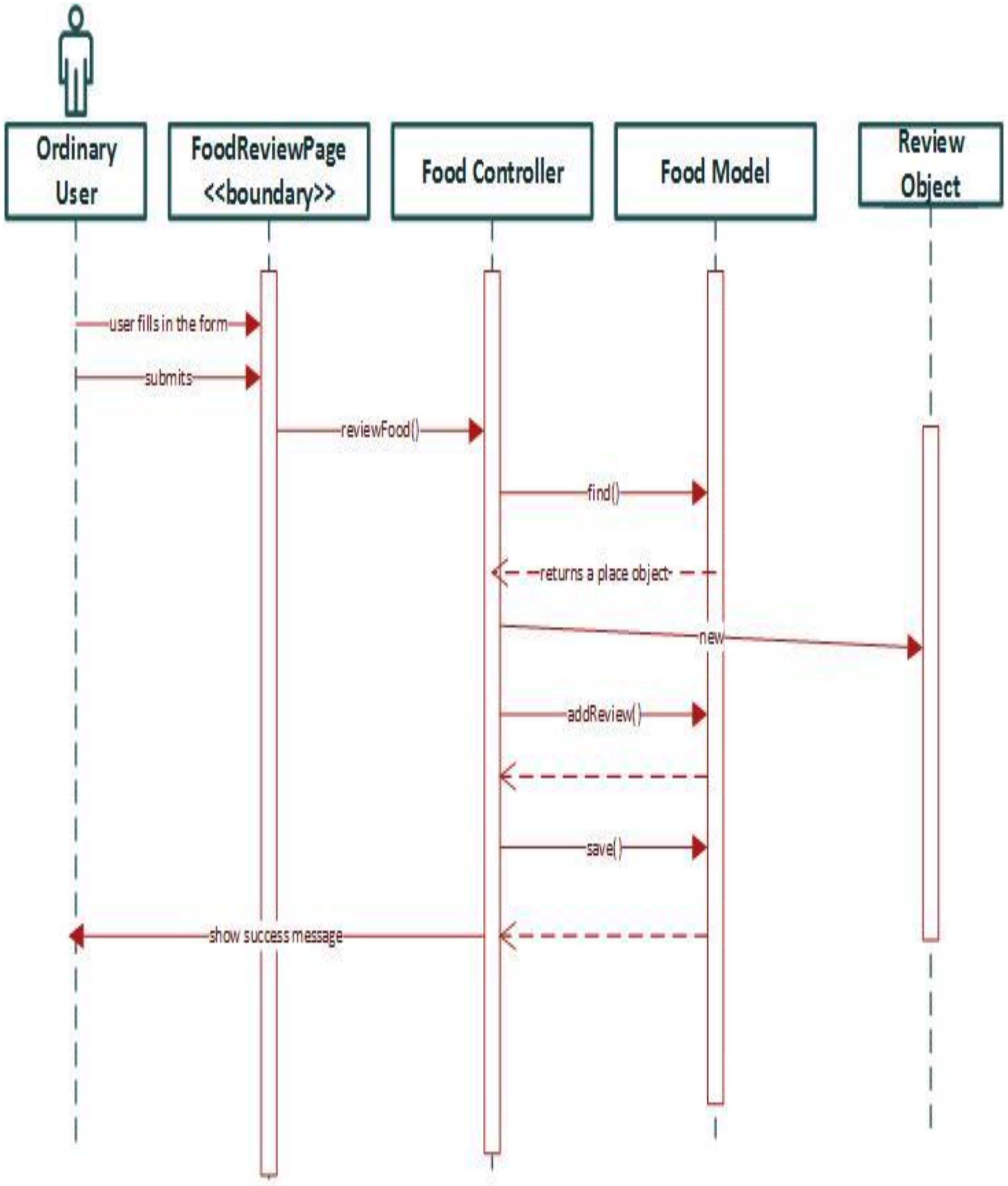

Actor(Place, User)
FoodAddPage <<boundary>>
Food Controller
Food Model
Fils in a form
submits
validates data entered
alt
if data is invalid
report error
data is valid
addNewFood()
uploads a photo
addPhoto()
shares GPS coordinates
addCoords()
submits
commit()
new Food()
food object
save()
show success message
UC 4 : Add a new Food



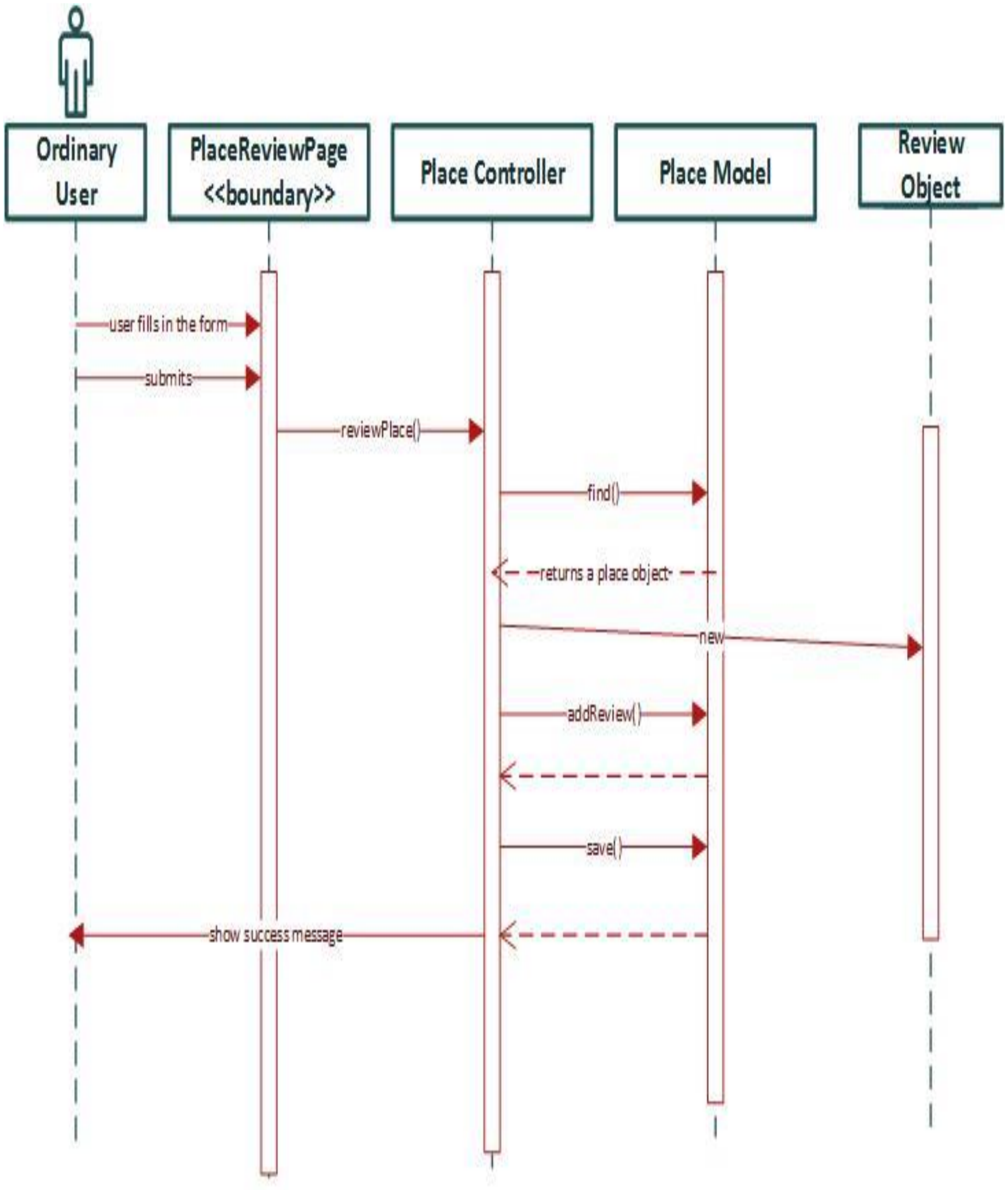
UC 5 : Rate a Food



UC 6 : Rate a Place



UC 7 : Give Food Review



UC 8 : Give Place Review

Websites

http://www.wikipedia.org/

http://www.appcrawlwer.com/web-apps/

Bibliography

Ian Sommerville Software Engineering

Prat, R. 2015. 7 Good Reasons to use MEAN Stack in your next web project. Available: http://www.journaldev.com/7462/node-js-architecture-single-threaded-event-loop. Accessed 8 May 2017.

Raj, J. 2014. An Introduction to the MEAN Stack. Available: https://www.sitepoint.com/introductionmean-stack. Accessed 10 May 2017.

Rouse, M. 2015. Implementation Definition. Available: http://searchcrm.techtarget.com/definition/implementation. Accessed 10 May 2017.