RUPE - Rate UP Eateries

Analysis Model

Submitted to:

Asst. Prof. Ma. Rowena C. Solamo
Faculty Member
Department of Computer Science
College of Engineering
University of the Philippines, Diliman

Submitted by:
Dylan Bayona
Annysia Dupaya
Makki Villaluz

In partial fulfillment of Academic Requirements
for the course
CS 191 Software Engineering I
of the
1st Semester, AY2019-2020

Unique Reference:

The documents are stored in the following link:
https://github.com/chinadupaya/RUPE/tree/master/02-Requirements%20Engineering/Project%20Deliverables
referenced with '1 - RUPE - Analysis Model.pdf'.

Purpose:

This document's purpose is to describe our system's analysis model, which helps identify the information and functions needed in the RUPE system. The analysis model is derived from our Use Case Model and Use Case Specifications, through the technique of Use Case Analysis. This document also includes our system's behavioral model, which outlines and describes the possible scenarios for each of our use cases.

Audience:

The audience of this document is for the developers of the software to use as a guide as well as any reader that wishes to know a more detailed look of the elements needed in the software. It also includes any person involved in assessing our work for Workshop 7 in the CS 191 - Software Engineering I course. The audience for this document can also include anyone who may want to view our Analysis Model, as well as the Boundary, Control, and Entity classes.

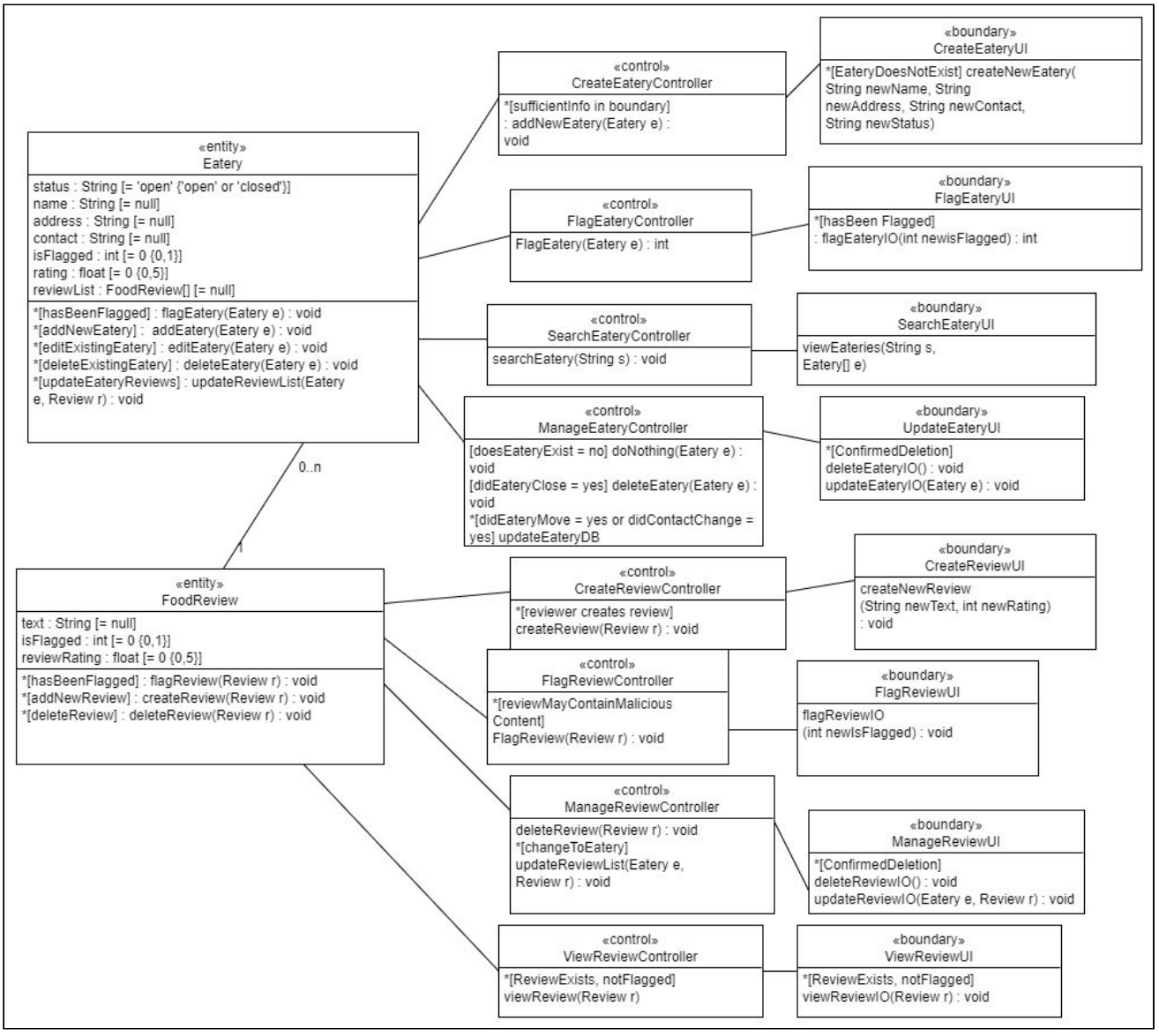
Revision Control:

Revision Date	*Person Responsible*	*Version Number*	*Modification*
09/26/19	Annysia Dupaya	1.0	Initial Document; Version number should match the one found in the footer.
09/27/19	Annysia Dupaya	1.1	Created the Analysis Model.
09/29/19	Annysia Dupaya	1.2	Added Use Case 1-3 and their respective scenarios
09/29/19	Dylan Bayona	1.3	Added Use Case 4, its description, and its relevant scenarios, and added descriptions for Use Cases 5-7
10/01/19	Dylan Bayona	1.4	Added scenarios for Use Cases 5-7
10/02/19	Makki Villaluz	1.5	Added the descriptions for the boundary, control, and entity classes
10/03/19	Dylan Bayona	1.6	Added links in Unique Reference, added text to Document Purpose and Audience Sections
10/03/19	Annysia Dupaya	1.7	Edited the Analysis Model.
10/03/19	Makki Villaluz	1.8	Added diagrams for Use Cases 3,8
10/04/19	Annysia Dupaya	1.9	Updated Boundary, Control, and Entity Classes
10/04/19	Dylan Bayona	1.10	Edited the Analysis Model and edited scenario 3 of Use Case 7.0
10/04/19	Makki Villaluz	1.11	Updated scenarios for Use Cases 3,8

***System Name*:** RUPE - Rate UP Eateries

***Description*:** The system is centered on providing a review system for eateries found in UP Diliman. Eatery reviewers can rate and review the food of these eateries based on a number of categories which include, but are not limited to, taste, cleanliness, and presentation. They can also search for eateries, view eateries, view food reviews made by other reviewers, flag inappropriate reviews, and flag eateries which are closed. Administrators can view food reviews and eateries, and they can also manage food reviews and eatery records. In the case of an inappropriate food review or a flagged eatery, they can modify or delete these records accordingly.

***Analysis Model*:**



Boundary Classes:

Class Name	**Description**
CreateEateryUI	This is the interface of the reviewers to the system when they add a new eatery that does not currently exist in the application. *Responsibilities*: *[EateryDoesNotExist]] createNewEatery(String newName, String newAddress, String newContact, String newStatus) : void
FlagEateryUI	This is the interface of the reviewers to the system when they find an eatery that does not exist in real life. *Responsibilities*: *[hasBeenFlagged] flagEateryIO(int newIsFlagged) : void
SearchEateryUI	This is the interface of the reviewers to the system when they want to look for an eatery in the application. *Responsibilities*: viewEateries(String s, Eatery[] e) : void
UpdateEateryUI	This is the interface of the administrators to the system when an eatery has been flagged and needs to be deleted or updated. *Responsibilities*: *[ConfirmedDeletion] deleteEateryIO() : void updateEateryIO(Eatery e) : void
CreateReviewUI	This is the interface of the reviewers to the system when they want to add a review for a given eatery. *Responsibilities*: createNewReview(Eatery e, Review r) : void
FlagReviewUI	This is the interface of the reviewers to the system when they deem a food review to be malicious. *Responsibilities*: public void flagReviewIO(int newIsFlagged)
ManageReviewUI	This is the interface of the administrators to the system when a food review has been flagged and needs to be deleted. *Responsibilities*: *[ConfirmedDeletion] deleteReviewIO() : void updateReviewList(Eatery e, Review r) : void

ViewReviewUI	This is the interface of the reviewers to the system when they want to look at a food review. *Responsibilities*: *[ReviewExists, notFlagged] viewReviewIO(Review r) : void

Control Classes:

Class Name	**Description**
CreateEateryController	This is the control that handles the creation of a new eatery. *Responsibilities*: *[sufficientInfoInBoundary] addNewEatery(Eatery e) : void
FlagEateryController	This is the control that flags an eatery. *Responsibilities*: FlagEatery(Eatery e) : int
SearchEateryController	This is the control that searches for an eatery in the application. *Responsibilities*: searchEatery(String s) : void
ManageEateryController	This is the control that deletes or updates an eatery. *Responsibilities*: *[doesEateryExist = no] doNothing(Eatery e) : void *[didEateryClose = yes] deleteEatery(Eatery e) deleteEatery(Eatery e) *[didEateryMove = yes OR didContactChange = yes]updateEateryDB(Eatery e) : void
CreateReviewController	This is the control that handles the creation of a food review. *Responsibilities*: *[reviewerCreatesReview] createReview(Review r) : void
FlagReviewController	This is the control that flags a malicious food review. *Responsibilities*: *[reviewMayContainMaliciousContent] FlagReview(Review r) : void
ManageReviewController	This is the control that deletes a food review. *Responsibilities*: deleteReview(Review r) : void

	*[changeToEatery] updateReviewList(Eatery e, Review r) : void
ViewReviewController	This is the control that shows a food review. *Responsibilities*: *[ReviewExists, notFlagged] viewReview(Review r)

Entity Classes:

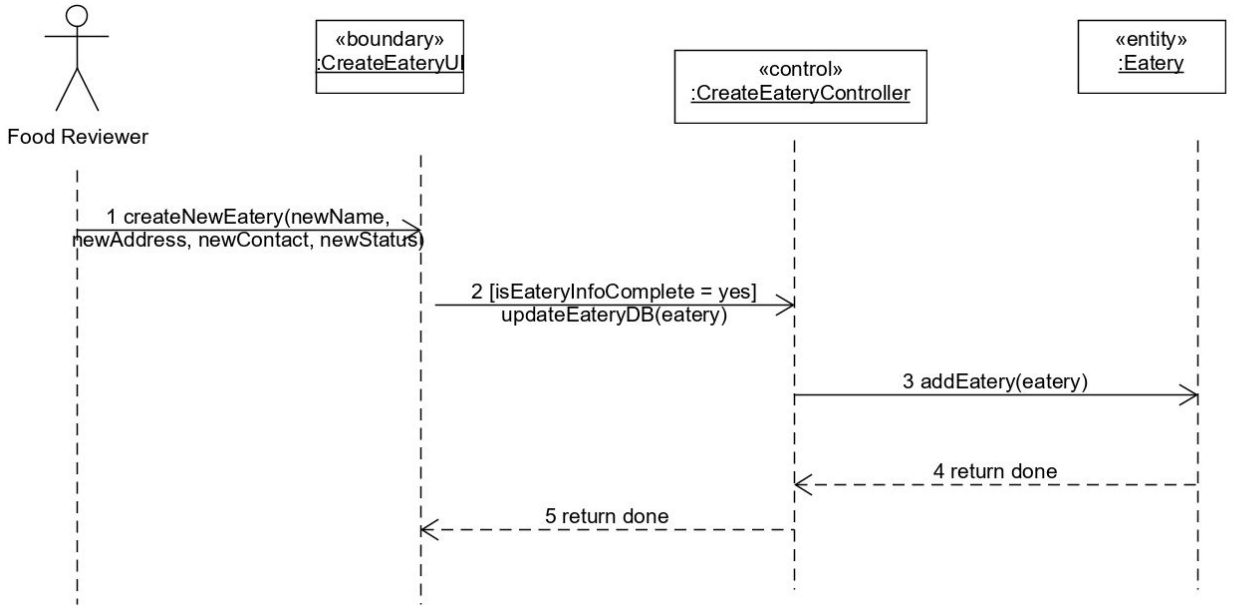
Class Name	**Description**
Eatery	The entity class Eatery contains all of the data about an eatery *Attributes*: status : String [= 'open' {'open' or 'closed'}] name : String [= null] address : String [= null] contact : String [= null] isFlagged : int [= 0 {0,1}] rating : float [= 0 {0,5}] reviewList : FoodReview[] [= null] -- *[hasBeenFlagged] : flagEatery(Eatery e) : void *[addNewEatery] : addEatery(Eatery e) : void *[editExistingEatery] : editEatery(Eatery e) : void *[deleteExistingEatery] : deleteEatery(Eatery e) : void *[updateEateryReviews] : updateReviewList(Eatery e, Review r) : void
FoodReview	The entity class FoodReview contains all of the data about a food review *Attributes*: text : String [= null] isFlagged : int [= 0 {0,1}] reviewRating : float [= 0 {0,5}] -- *[hasBeenFlagged] : flagReview(Review r) : void *[addNewReview] : createReview(Review r) : void *[deleteReview] : deleteReview(Review r) : void

Behavioral Model:

***Use-Case Name*:** 1.0 Create Eatery

Description: If an eatery is yet to exist in the application, an eatery reviewer can choose to create an eatery. They will have to input the name of the eatery (if known), its location, contact details (if known) and a review for the eatery to be added. In case that an eatery contains inappropriate information, a user can flag it as mentioned in Use-Case 5.0.

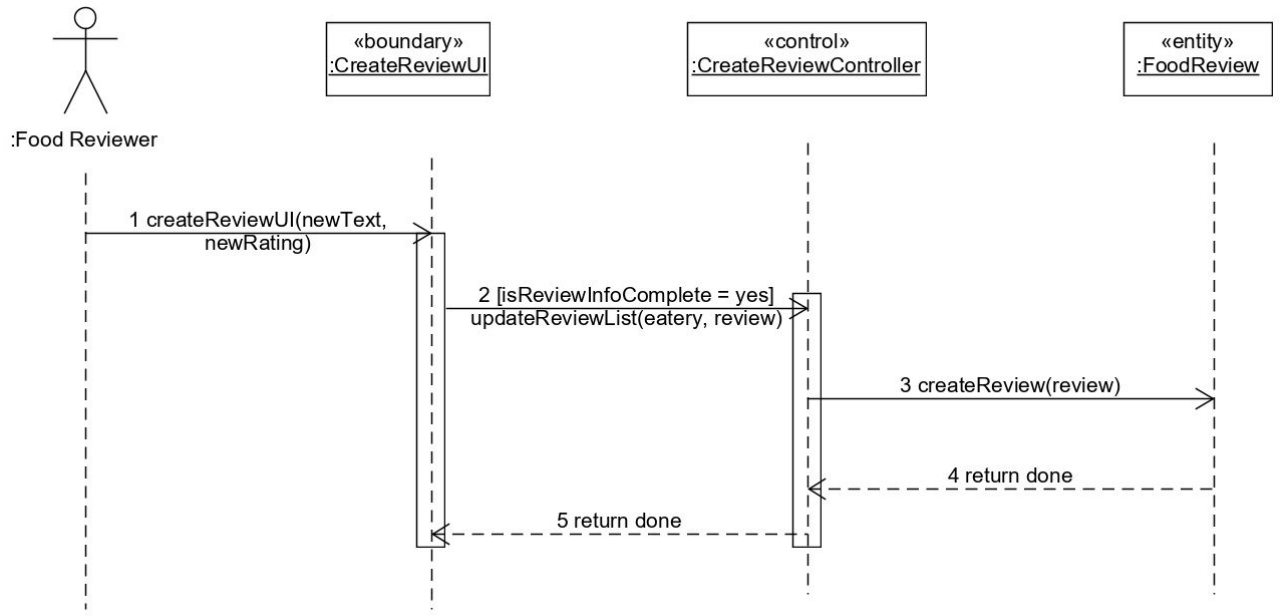
Scenario 1: Add a non-existent eatery to the database.



***Use-Case Name*:** 2.0 Create Review

Description: Food reviews are the main feature of the application. Eater reviewers will be able to input a food review on the eatery of their choice. Reviewers will be able to rate several aspects of the eatery and input what food they had.

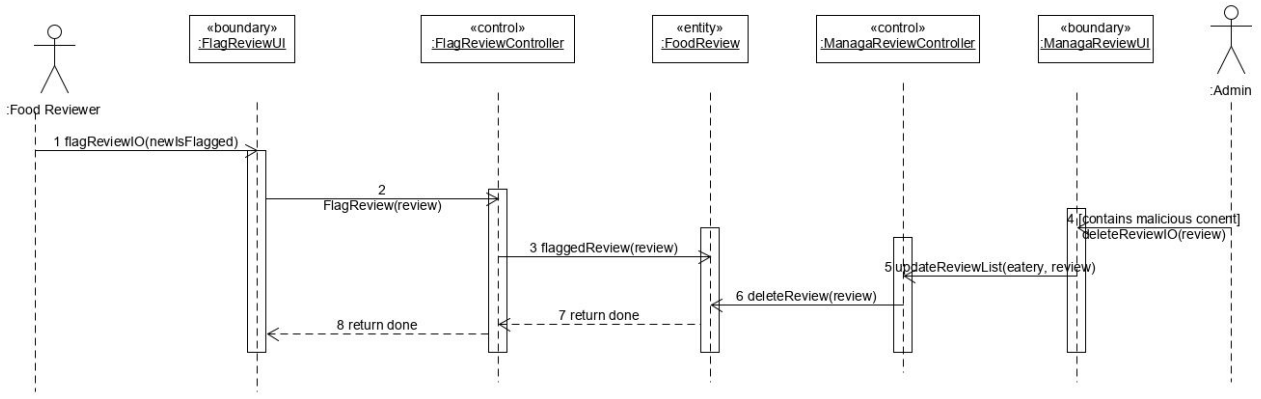
Scenario 1: Food reviewers can review an eatery



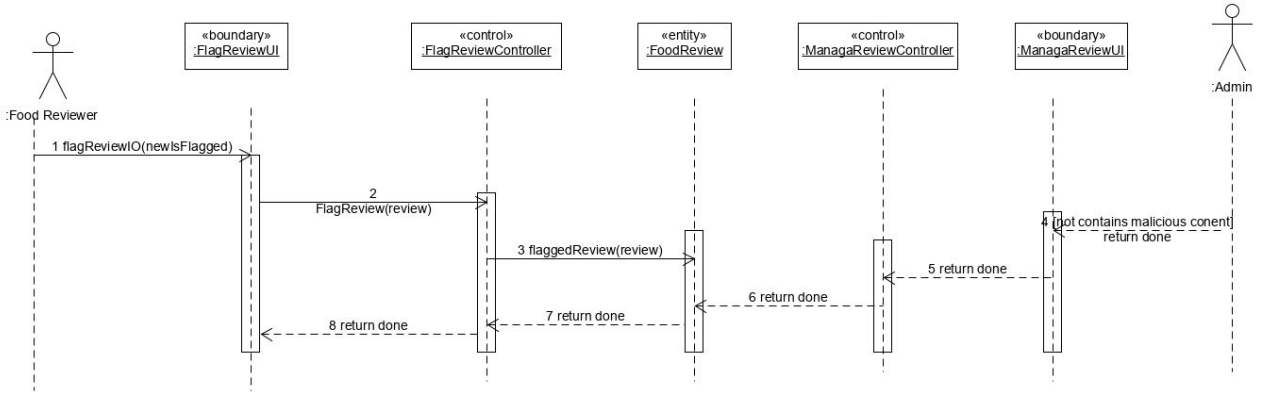
***Use-Case Name*:** 3.0 Flag food review

Description: If a reviewer sees a food review that can be deemed malicious or spam, they can flag it and an admin will be notified. This content can contain text that include but are not limited to: having been made to manipulate the ratings or contain extremely vulgar language. Flagged content that has not yet been reviewed will have its content hidden until a reviewer wishes to see it.

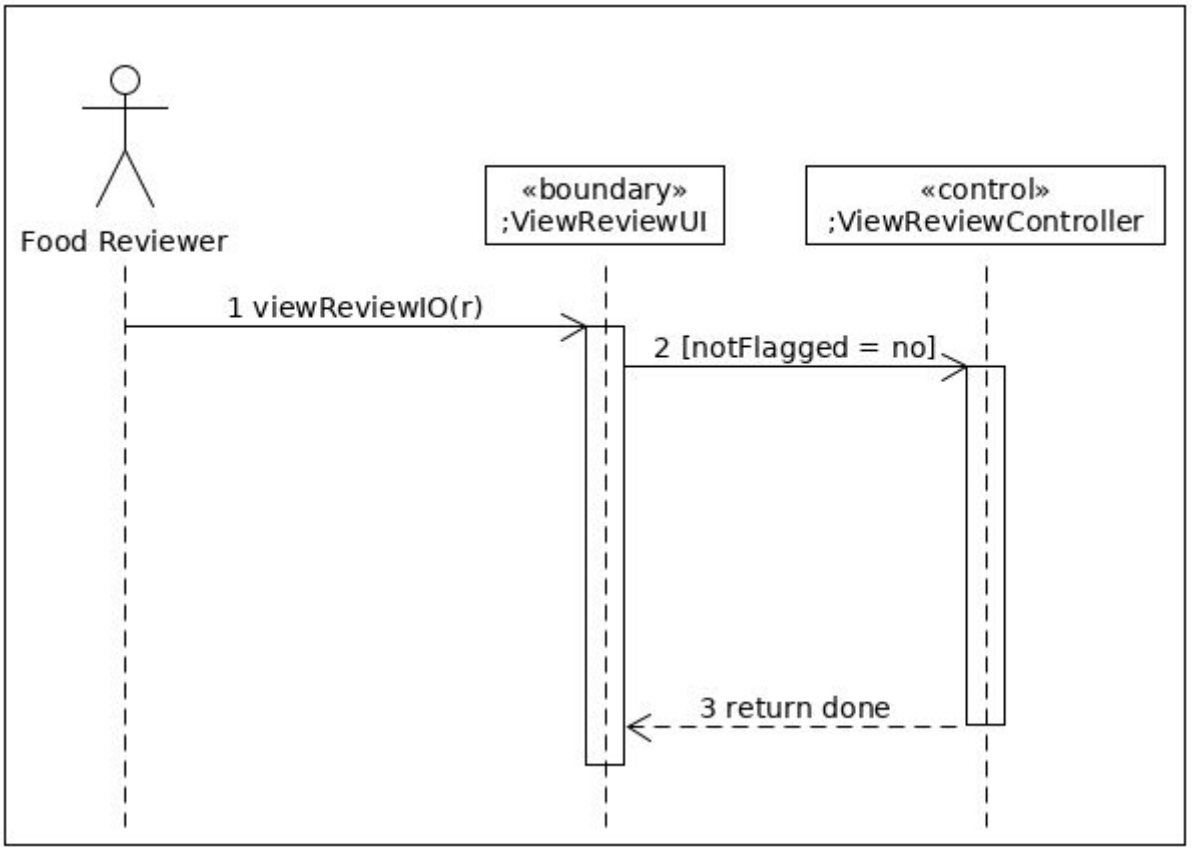
Scenario 1: Food reviewer flags a review and it is deemed inappropriate by Admin.



Scenario 2: Food reviewer flags a review and is not deemed inappropriate by Admin.



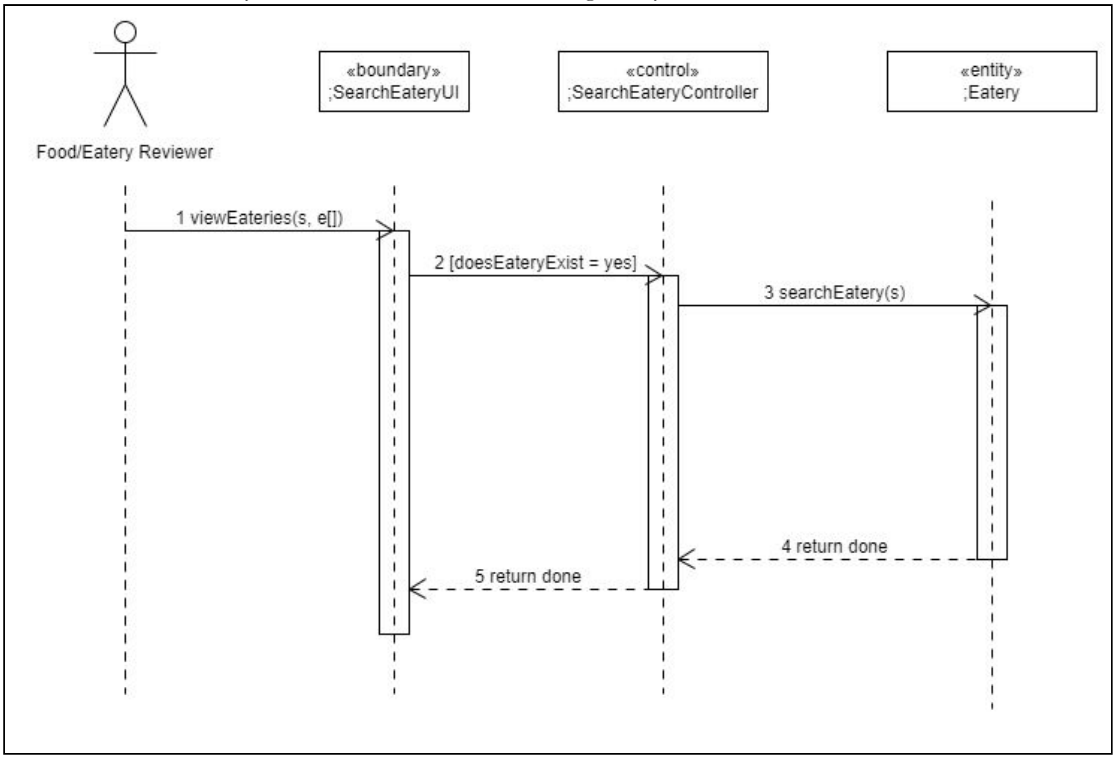
Scenario 3: Already flagged review is attempted to be flagged



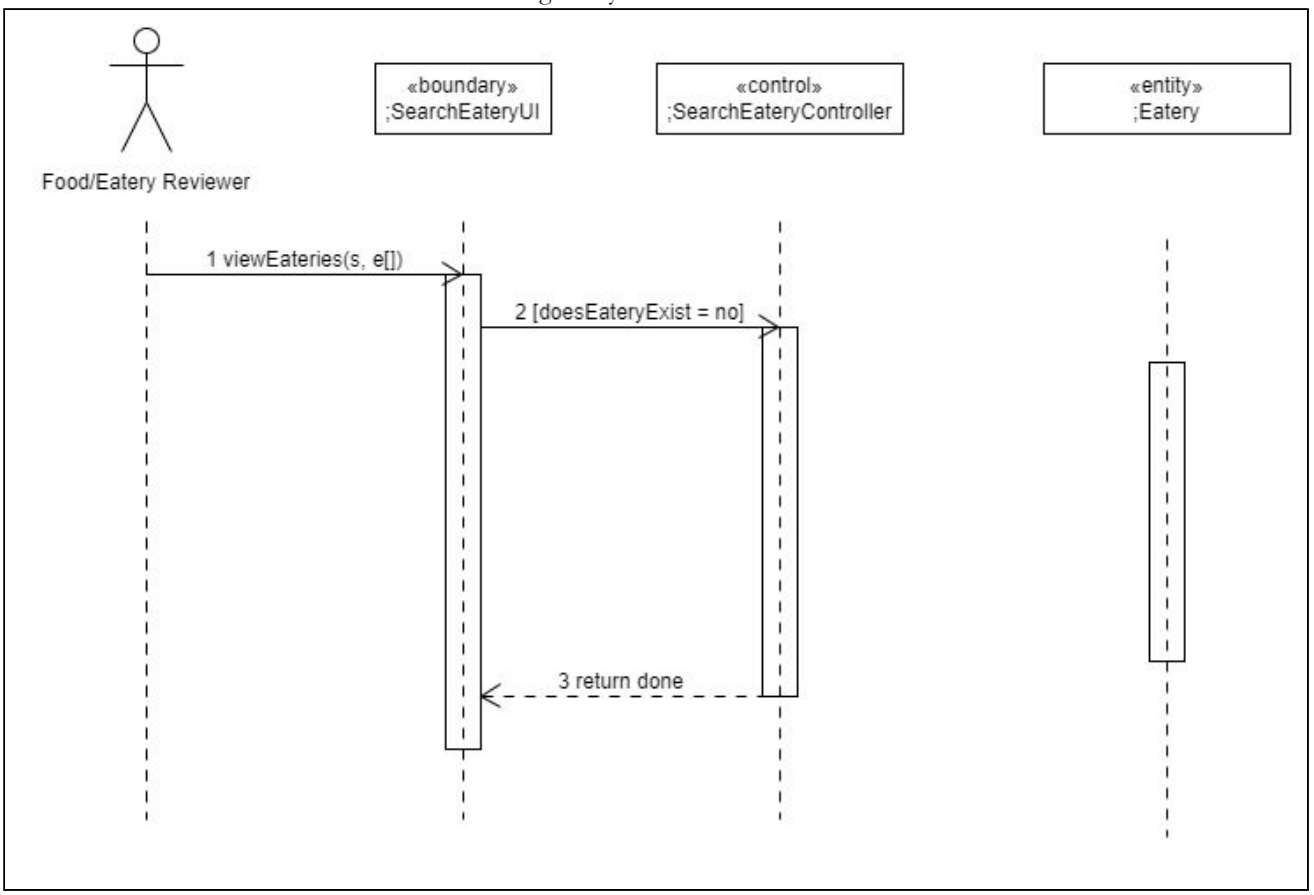
***Use-Case Name*:** 4.0 Search

Description: Food/eatery reviewers can search for a particular eatery. They can do this by inputting the name of the eatery that they want to view. The system would then return a list of eatery names containing the reviewer's queried text.

Scenario 1: Food /eatery reviewer searches for an existing eatery



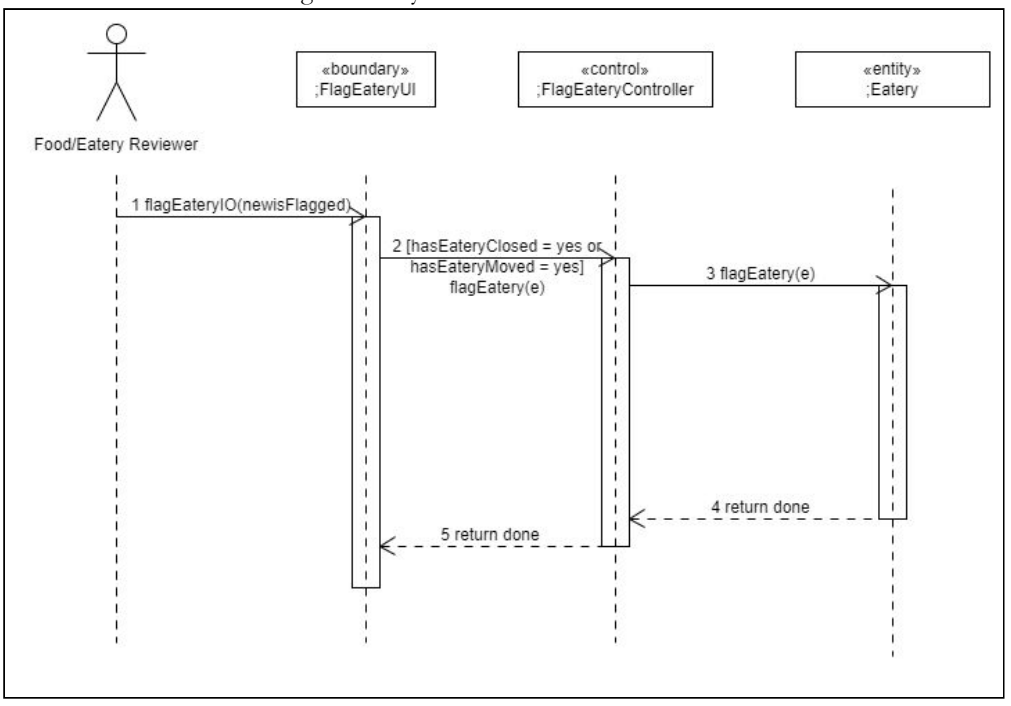
Scenario 2: Food reviewer searches for a non-existing eatery



***Use-Case Name*:** 5.0 Flag eatery

Description: Food/eatery reviewers can flag eateries in our application. This is used to signify that the flagged eatery does not exist anymore, meaning that it has either closed or moved to a different location. If too many reviewers flag a certain eatery, then the administrator can remove it, detailed in Use-Case 7.0.

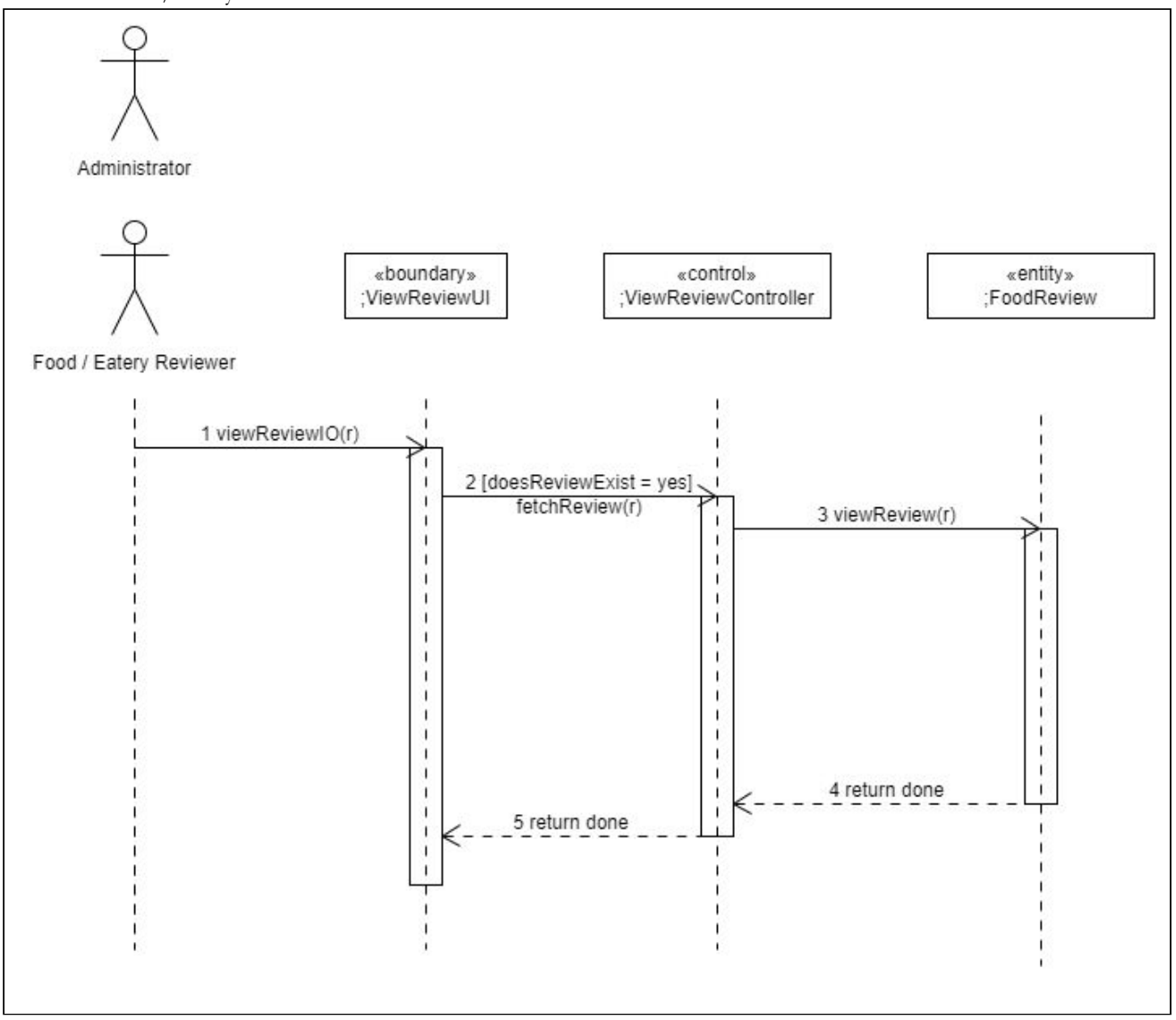
Scenario 1: Food reviewer flags an eatery that has closed or moved locations



***Use-Case Name*:** 6.0 View food review

Description: Food reviewers and administrators can view food reviews made by other food reviewers. The former can utilize this use case to decide which eatery to dine at. The latter can use this to view flagged food reviews and manage them, described in Use-Case 8.0.

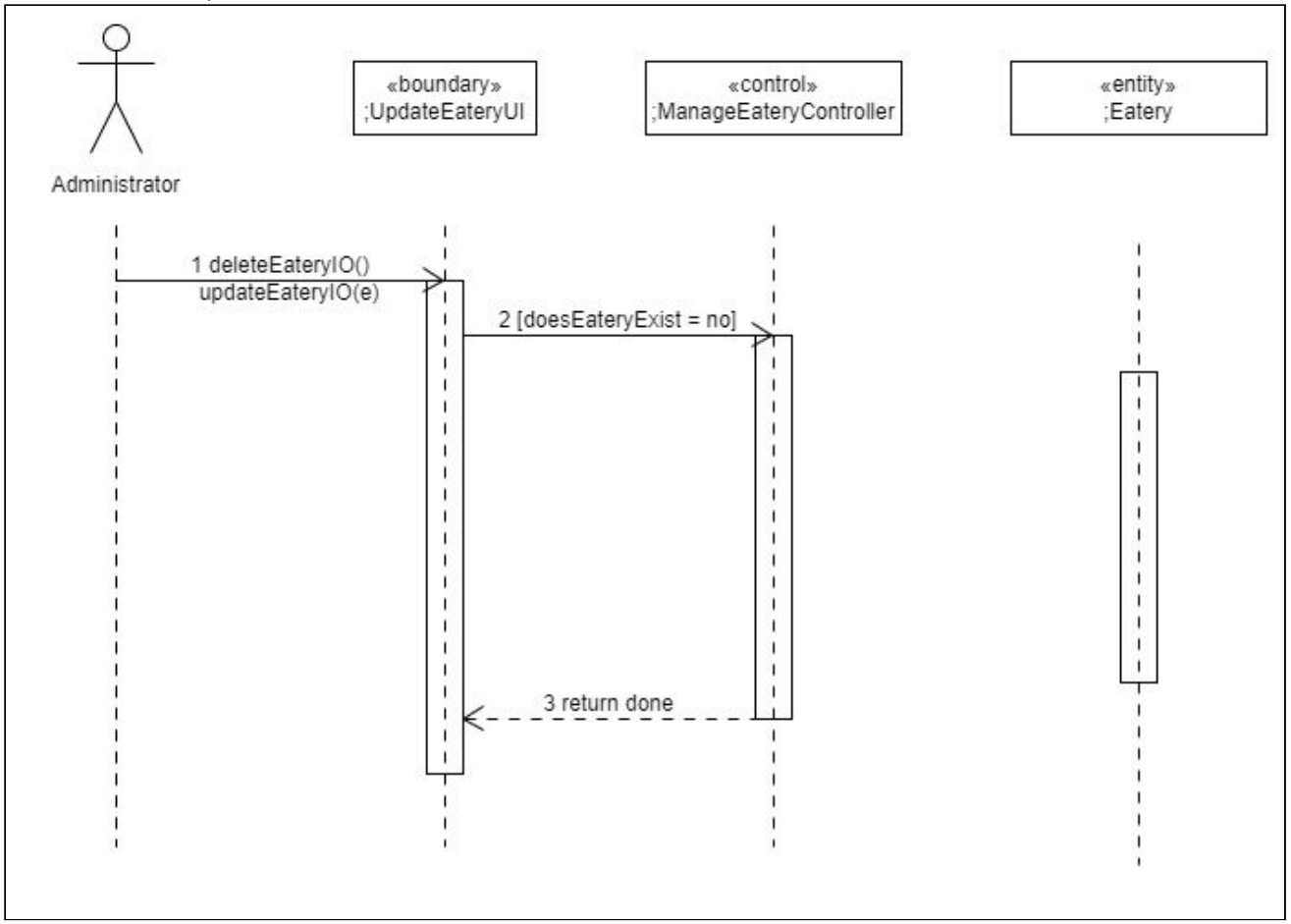
Scenario 1: Food / eatery reviewer or administrator views a food review



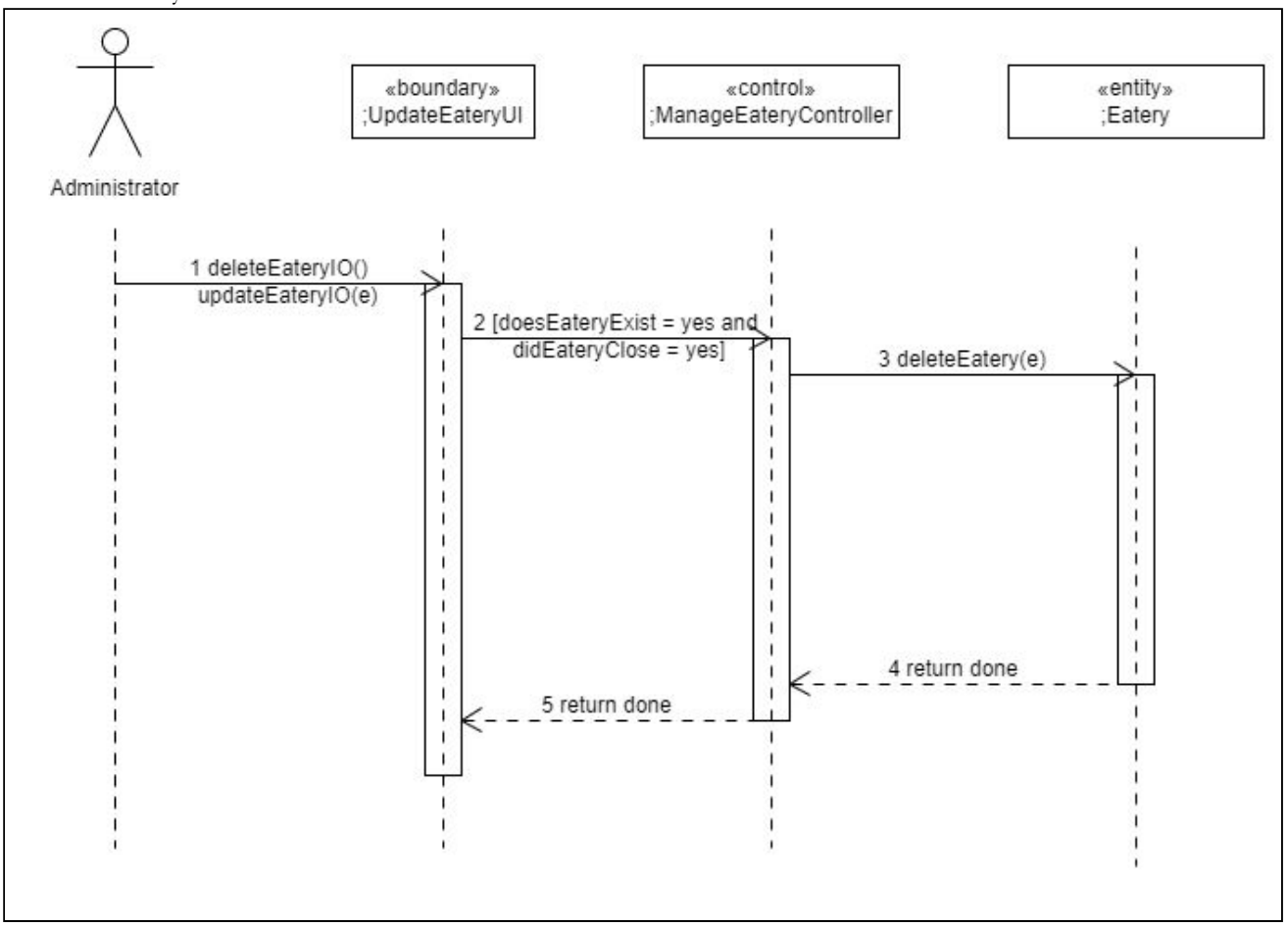
***Use-Case Name*:** 7.0 Manage eatery record

Description: The administrators are the only ones who can manipulate the eatery records. If they see that a record has been flagged, they can modify it to change its values. For example, if an eatery has closed, then the administrator can delete said eatery; if an eatery has moved locations or changed contact details, then the administrator can edit the record accordingly. They can also delete the record if it does not contain any useful or relevant information.

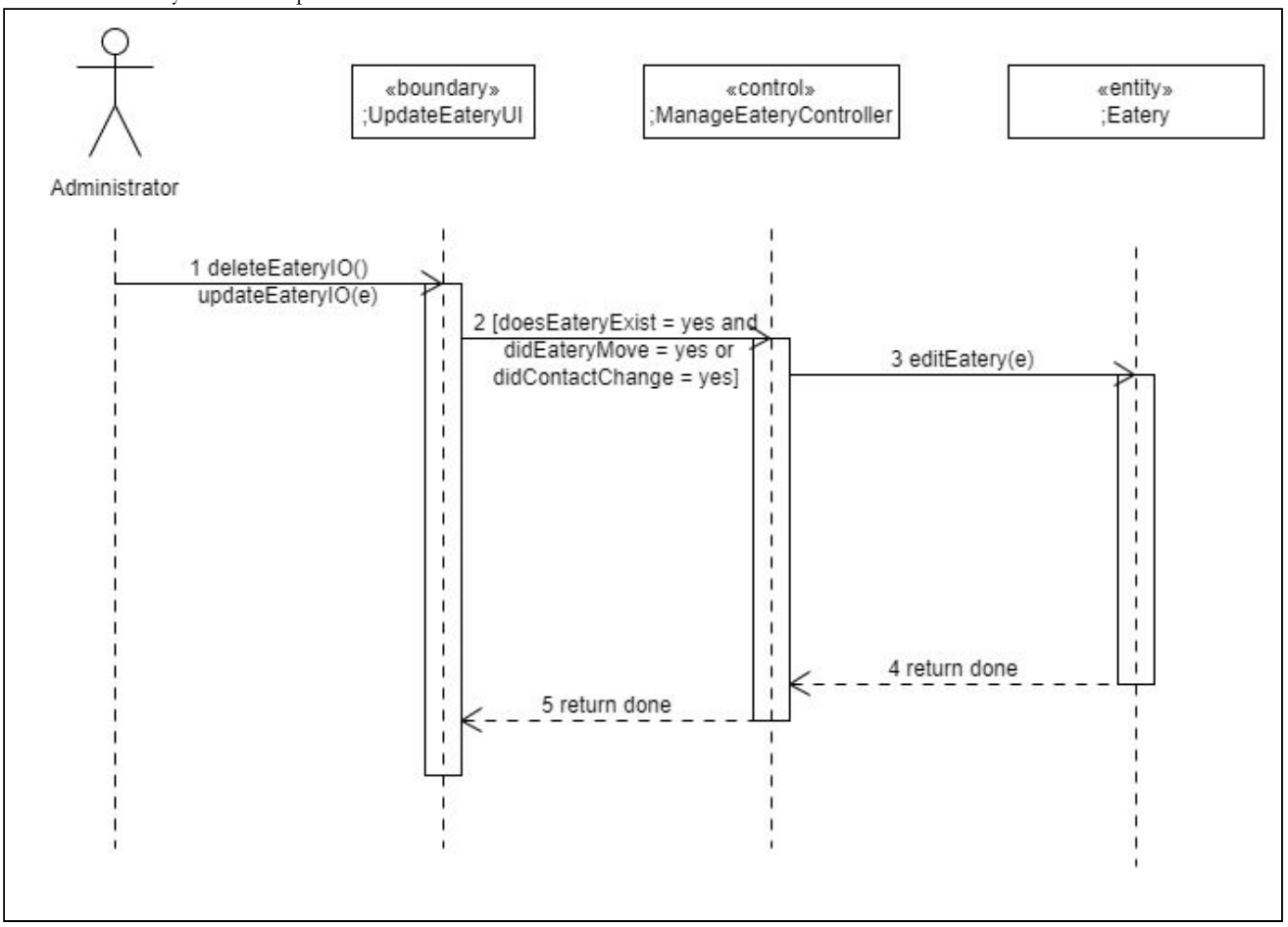
Scenario 1: No eatery records exist



Scenario 2: Eatery record is removed



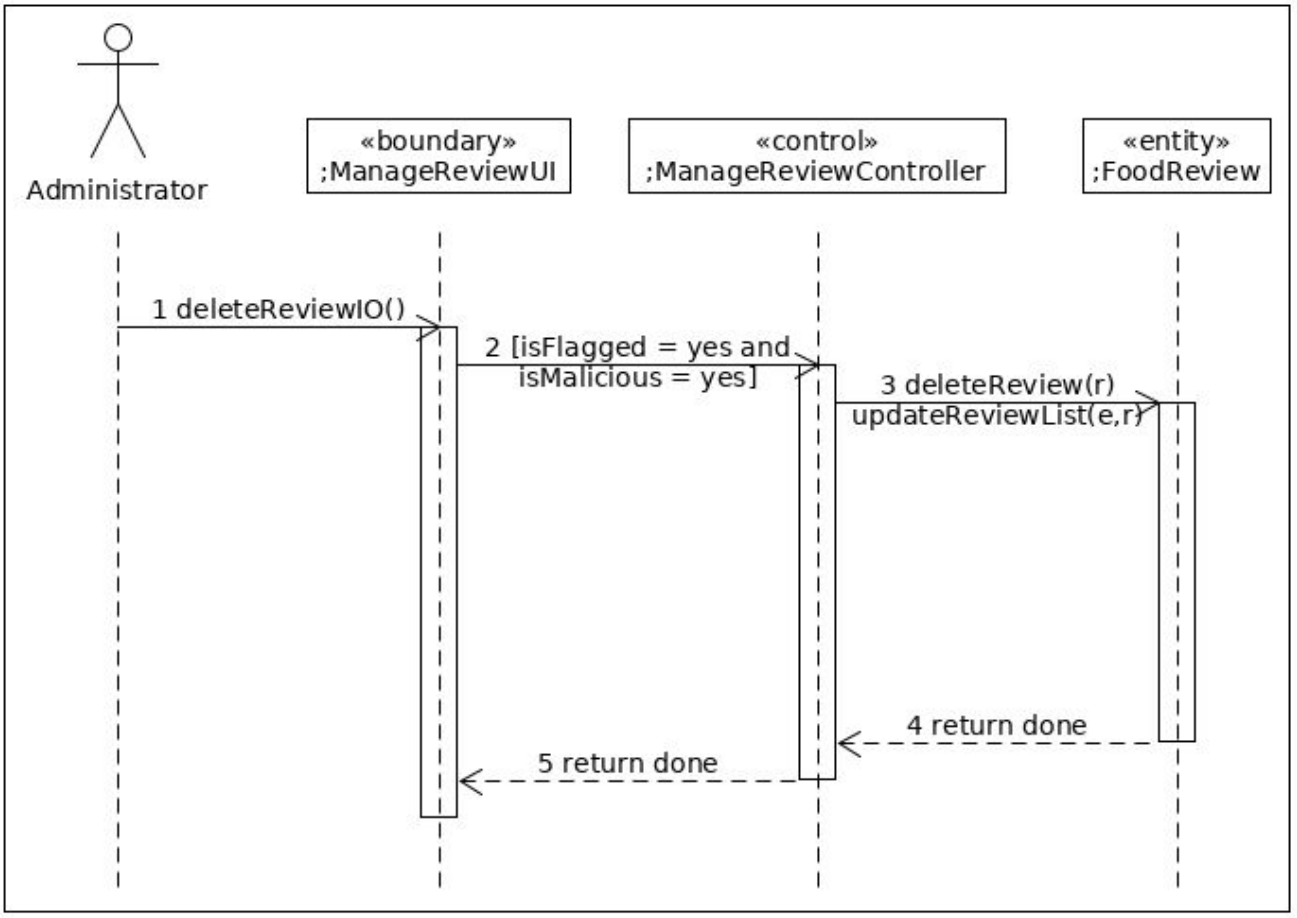
Scenario 3: Eatery record is updated



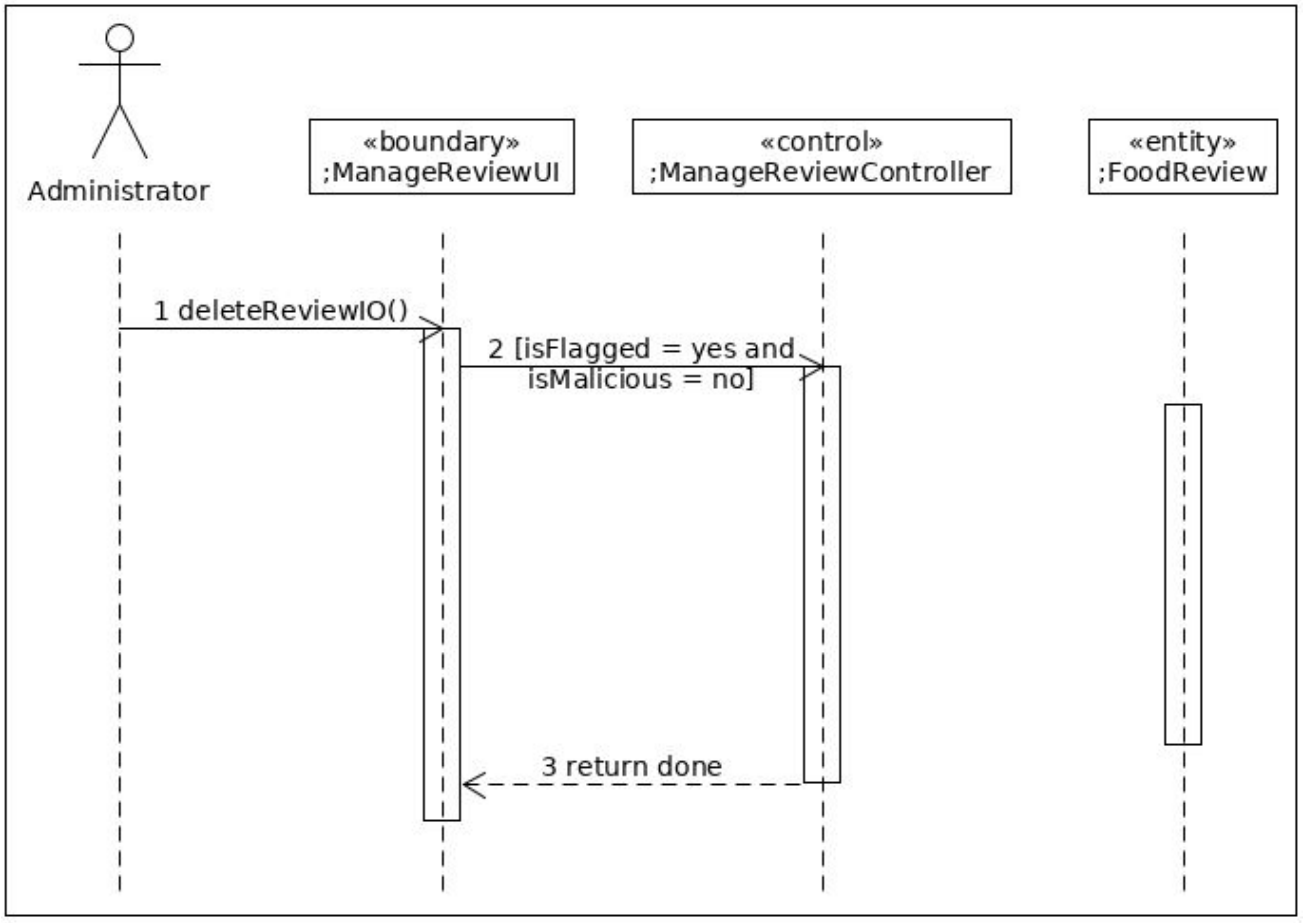
***Use-Case Name*:** 8.0 Manage Food Review

Description: The administrators are the only ones who can manipulate the food reviews. If they see that a review has been flagged, they can delete the review. This will remove reviews that can be considered malicious or spam.

Scenario 1: A food review is flagged and deleted.



Scenario 2: A food review is flagged but remains in the eatery.



Scenario 3 A food review is flagged, remains in the eatery but is flagged more times.