FiTracks
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:
Aguilana, Trina B.
Manguiat, Glenn Karlo D.
Villanueva, Ian N.

In partial fulfillment of Academic Requirements
for the course
CS 191 Software Engineering I
of the
1st Semester, AY 2018-2019

Unique Reference:

The documents are stored in the Project Repository Link: https://bit.ly/2NymdWX referenced with Group 4 - FiTracks - Analysis Model.pdf.

Purpose:

This document is provided to model the use cases and its scenarios for the health and fitness record management system through Analysis Model and Behavior Models. In the analysis model, each classes are provided with descriptions for readability for the audience.

Audience:

This document serves its purpose for Asst. Prof. Rowena Solamo and her students analyzing the use cases and its scenarios of the health and fitness record management system.
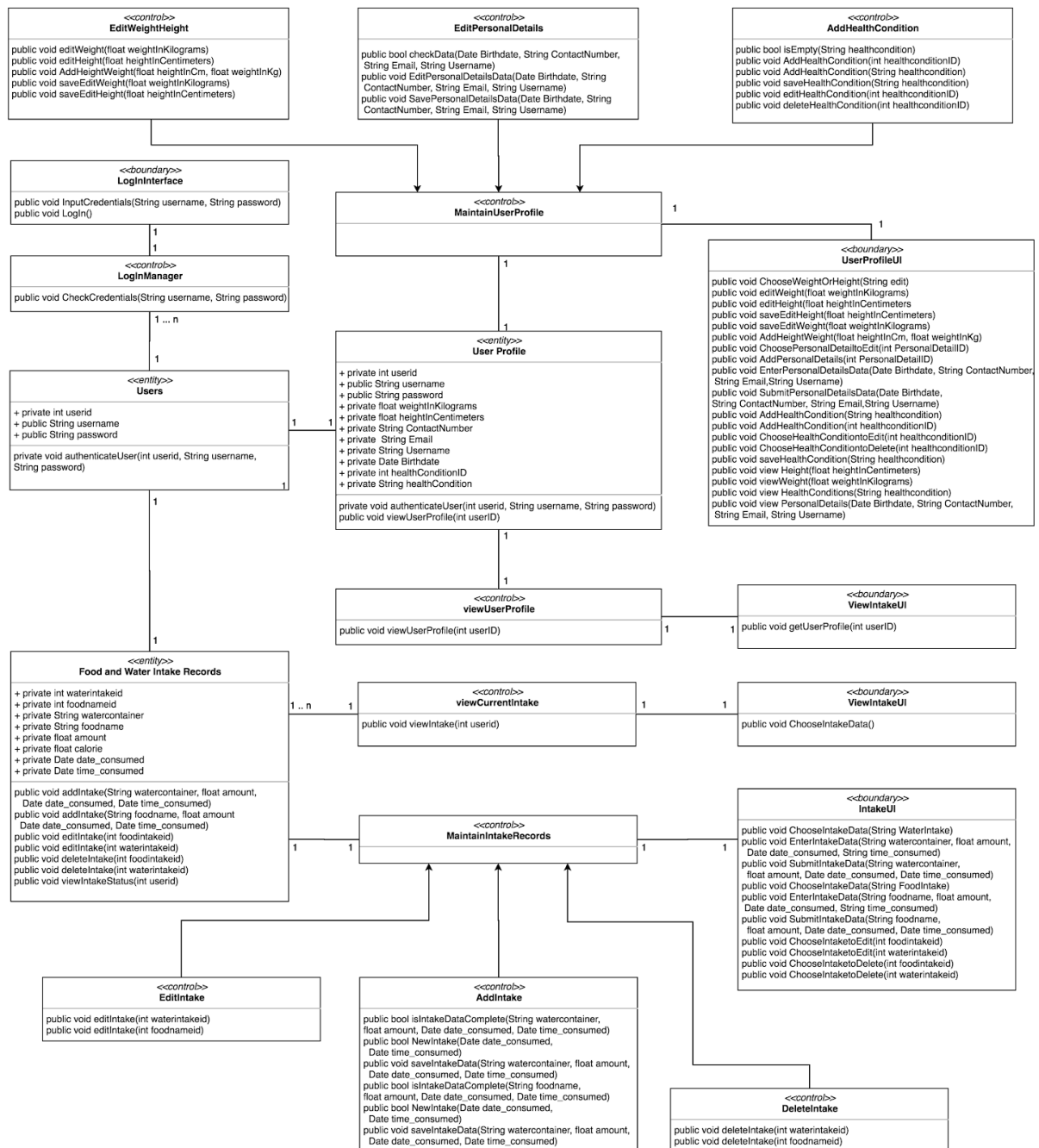
Revision Control:

Revision Date	*Person Responsible*	*Version Number*	*Modification*
10/01/18	Ian N. Villanueva	1.0	Initial Document; Added document purpose and audience, added system name and its description.
10/03/18	Glenn Karlo D. Manguiat	1.1	Added boundary classes for Use Cases 1.0-1.3, 7.0; Added control classes for Use Cases 1.0-1.3, 7.0; Added entity classes for Use Cases 1.0-1.3, 7.0; Added behavior models for Use Case 1.0-1.3; Edited document purpose and audience.
10/04/18	Ian N. Villanueva	1.2	Added boundary classes for Use Cases 2.0-2.3; Added control classes for Use Cases 2.0-2.3; Added entity classes for Use Cases 2.0-2.3; Added behavior models for Use Case 2.0-2.3;
10/04/18	Trina B. Aguilana	1.3	Added boundary classes for Use Cases 4.0, 6.0; Added control classes for Use Cases 4.0, 6.0; Added behavior models for Use Cases 4.0, 6.0
10/26/18	Glenn Karlo D. Manguiat	1.4	Revised boundary class viewCurrentIntake, controller class viewCurrentIntake, and entity class Food and Water Intake Records

System Name: FiTracks - Health and Fitness Record Management System

Description: In our health and fitness record management system, a user maintains a personal food and water intake tracker upon log-in. User establishes a profile which contains his/her personal details, current height, current weight, and current health condition, if with complications or none. The establish profile may be viewed and updated at any time possible. User can input meal intake and water intake, which may be updated and deleted upon addition. Lastly, the current meal and water intake status can be viewed by the user in different time schemes.

Analysis Model:

<<control>> EditWeightHeight
- public void editWeight(float weightInKilograms)
- public void editHeight(float heightInCentimeters)
- public void AddHeightWeight(float heightInCm, float weightInKg)
- public void saveEditWeight(float weightInKilograms)
- public void saveEditHeight(float heightInCentimeters)

<<control>> EditPersonalDetails
- public bool checkData(Date Birthdate, String ContactNumber, String Email, String Username)
- public void EditPersonalDetailsData(Date Birthdate, String ContactNumber, String Email, String Username)
- public void SavePersonalDetailsData(Date Birthdate, String ContactNumber, String Email, String Username)

<<control>> AddHealthCondition
- public bool isEmpty(String healthcondition)
- public void AddHealthCondition(int healthconditionID)
- public void AddHealthCondition(String healthcondition)
- public void saveHealthCondition(String healthcondition)
- public void editHealthCondition(int healthconditionID)
- public void deleteHealthCondition(int healthconditionID)

<<boundary>> LogInInterface
- public void InputCredentials(String username, String password)
- public void LogIn()

<<control>> MaintainUserProfile

<<boundary>> UserProfileUI
- public void ChooseWeightOrHeight(String edit)
- public void editWeight(float weightInKilograms)
- public void editHeight(float heightInCentimeters)
- public void saveEditHeight(float heightInCentimeters)
- public void saveEditWeight(float weightInKilograms)
- public void AddHeightWeight(float heightInCm, float weightInKg)
- public void ChoosePersonalDetailtoEdit(int PersonalDetailID)
- public void AddPersonalDetails(int PersonalDetailID)
- public void EnterPersonalDetailsData(Date Birthdate, String ContactNumber, String Email,String Username)
- public void SubmitPersonalDetailsData(Date Birthdate, String ContactNumber, String Email,String Username)
- public void AddHealthCondition(String healthcondition)
- public void AddHealthCondition(int healthconditionID)
- public void ChooseHealthConditiontoEdit(int healthconditionID)
- public void ChooseHealthConditiontoDelete(int healthconditionID)
- public void saveHealthCondition(String healthcondition)
- public void view Height(float heightInCentimeters)
- public void viewWeight(float weightInKilograms)
- public void view HealthConditions(String healthcondition)
- public void view PersonalDetails(Date Birthdate, String ContactNumber, String Email, String Username)

<<control>> LogInManager
- public void CheckCredentials(String username, String password)

<<entity>> Users
- + private int userid
- + public String username
- + public String password
- private void authenticateUser(int userid, String username, String password)

<<entity>> User Profile
- + private int userid
- + public String username
- + public String password
- + private float weightInKilograms
- + private float heightInCentimeters
- + private String ContactNumber
- + private String Email
- + private String Username
- + private Date Birthdate
- + private int healthConditionID
- + private String healthCondition
- private void authenticateUser(int userid, String username, String password)
- public void viewUserProfile(int userID)

<<control>> viewUserProfile
- public void viewUserProfile(int userID)

<<boundary>> ViewIntakeUI
- public void getUserProfile(int userID)

<<entity>> Food and Water Intake Records
- + private int waterintakeid
- + private int foodnameid
- + private String watercontainer
- + private String foodname
- + private float amount
- + private float calorie
- + private Date date_consumed
- + private Date time_consumed
- public void addIntake(String watercontainer, float amount, Date date_consumed, Date time_consumed)
- public void addIntake(String foodname, float amount, Date date_consumed, Date time_consumed)
- public void editIntake(int foodintakeid)
- public void editIntake(int waterintakeid)
- public void deleteIntake(int foodintakeid)
- public void deleteIntake(int waterintakeid)
- public void viewIntakeStatus(int userid)

<<control>> viewCurrentIntake
- public void viewIntake(int userid)

<<boundary>> ViewIntakeUI
- public void ChooseIntakeData()

<<control>> MaintainIntakeRecords

<<boundary>> IntakeUI
- public void ChooseIntakeData(String WaterIntake)
- public void EnterIntakeData(String watercontainer, float amount, Date date_consumed, String time_consumed)
- public void SubmitIntakeData(String watercontainer, float amount, Date date_consumed, Date time_consumed)
- public void ChooseIntakeData(String FoodIntake)
- public void EnterIntakeData(String foodname, float amount, Date date_consumed, String time_consumed)
- public void SubmitIntakeData(String foodname, float amount, Date date_consumed, Date time_consumed)
- public void ChooseIntaketoEdit(int foodintakeid)
- public void ChooseIntaketoEdit(int waterintakeid)
- public void ChooseIntaketoDelete(int foodintakeid)
- public void ChooseIntaketoDelete(int waterintakeid)

<<control>> EditIntake
- public void editIntake(int waterintakeid)
- public void editIntake(int foodnameid)

<<control>> AddIntake
- public bool isIntakeDataComplete(String watercontainer, float amount, Date date_consumed, Date time_consumed)
- public bool NewIntake(Date date_consumed, Date time_consumed)
- public void saveIntakeData(String watercontainer, float amount, Date date_consumed, Date time_consumed)
- public bool isIntakeDataComplete(String foodname, float amount, Date date_consumed, Date time_consumed)
- public bool NewIntake(Date date_consumed, Date time_consumed)
- public void saveIntakeData(String watercontainer, float amount, Date date_consumed, Date time_consumed)

<<control>> DeleteIntake
- public void deleteIntake(int waterintakeid)
- public void deleteIntake(int foodnameid)

Boundary Classes:

Class Name	Description
IntakeUI	This is the interface of user whenever she or he wants to maintain the food and water intakes. *Responsibilities*: public void ChooseIntakeData(String WaterIntake) public void EnterIntakeData(String watercontainer, float amount, Date date_consumed, String time_consumed) public void SubmitIntakeData(String watercontainer, float amount, Date date_consumed, Date time_consumed) public void ChooseIntakeData(String FoodIntake) public void EnterIntakeData(String foodname, float amount, Date date_consumed, String time_consumed) public void SubmitIntakeData(String foodname, float amount, Date date_consumed, Date time_consumed) public void ChooseIntaketoEdit(int foodintakeid) public void ChooseIntaketoEdit(int waterintakeid) public void ChooseIntaketoDelete(int foodintakeid) public void ChooseIntaketoDelete(int waterintakeid)
LogInInterface	This is the interface for the user log-in. *Responsibilities*: public void InputCredentials(String username, String password) public void LogIn()
UserProfileUI	This is the interface of user whenever they want to view or edit their respective user profiles *Responsibilities*: public void ChooseWeightOrHeight(String edit) public void editWeight(float weightInKilograms) public void editHeight(float heightInCentimeters public void saveEditHeight(float heightInCentimeters) public void saveEditWeight(float weightInKilograms) public void AddHeightWeight(float heightInCm, float weightInKg) public void ChoosePersonalDetailtoEdit(int PersonalDetailID)

	public void AddPersonalDetails(int PersonalDetailID)
	public void EnterPersonalDetailsData(Date Birthdate, String ContactNumber, String Email,String Username)
	public void SubmitPersonalDetailsData(Date Birthdate, String ContactNumber, String Email,String Username)
	public void AddHealthCondition(String healthcondition)
	public void AddHealthCondition(int healthconditionID)
	public void ChooseHealthConditiontoEdit(int healthconditionID)
	public void ChooseHealthConditiontoDelete(int healthconditionID)
	public void saveHealthCondition(String healthcondition)
	public void view Height(float heightInCentimeters)
	public void viewWeight(float weightInKilograms)
	public void view HealthConditions(String healthcondition)
	public void view PersonalDetails(Date Birthdate, String ContactNumber, String Email, String Username)
ViewUserProfileUI	This is the interface for when the user wants to view his or her profile. _Responsibilities:_ public void getUserProfile(int userID)
viewCurrentIntake	This is the interface for when the user wants to view a food or water intake. _Responsibilities:_ public void ChooseIntakeData( )

Control Classes:

Class Name	Description
MaintainIntakeRecords	This is the control that maintain food and water intakes record. It is considered an abstract class.
AddIntake	This is the control that adds a food or water intake to the system. It extends MaintainIntakeRecords. *Responsibilities:* public bool isIntakeDataComplete(String watercontainer, float amount, Date date_consumed, Date time_consumed) public bool NewIntake(Date date_consumed, Date time_consumed) public void saveIntakeData(String watercontainer, float amount, Date date_consumed, Date time_consumed) public bool isIntakeDataComplete(String foodname, float amount, Date date_consumed, Date time_consumed) public bool NewIntake(Date date_consumed, Date time_consumed) public void saveIntakeData(String foodnamer, float amount, Date date_consumed, Date time_consumed)
EditIntake	This is the control that edits a food or water intake to the system. It extends MaintainIntakeRecords. *Responsibilities:* public void editIntake(int waterintakeid) public void editIntake(int foodnameid)
DeleteIntake	This is the control that deletes a food or water intake to the system. It extends MaintainIntakeRecords. *Responsibilities:* public void deleteIntake(int waterintakeid) public void deleteIntake(int foodnameid)
ViewIntake	This is the control that lets the user view his/her current food or water intake *Reponsibilities:* public void viewIntake(int userid)
MaintainUserProfile	This is the control that maintains user profiles. It is considered an abstract class. public void CheckCredentials(String username, String password)
EditWeightHeight	This is the control that edits user weight/height. It extends MaintainUserProfile *Responsibilities:* public void editWeight(float weightInKilograms)

	public void editHeight(float heightInCentimeters) public void AddHeightWeight(float heightInCm, float weightInKg) public void saveEditWeight(float weightInKilograms) public void saveEditHeight(float heightInCentimeters)
EditPersonalDetails	This is the control that edits user personal details It extends MaintainUserProfile. *Responsibilities:* public bool checkData(Date Birthdate, String ContactNumber, String Email,String Username) public void EditPersonalDetailsData(Date Birthdate, String ContactNumber, String Email, String Username) public void SavePersonalDetailsData(Date Birthdate, String ContactNumber, String Email, String Username)
AddHealthCondition	This is the control that adds user health conditions. It extends MaintainUserProfile. *Responsibilities:* public bool isEmpty(String healthcondition) public void AddHealthCondition(int healthconditionID) public void AddHealthCondition(String healthcondition) public void saveHealthCondition(String healthcondition) public void editHealthCondition(int healthconditionID) public void deleteHealthCondition(int healthconditionID)
viewUserProfile	This is the control that lets the user view his/her user profile details. *Responsibilities:* public void viewUserProfile(int userID)
viewCurrentIntake	This is the control that lets the user view a food or water intake. *Responsibilities:* public void viewIntake(int waterintakeid) public void viewIntake(int foodintakeid)
LogInManager	This is the control that checks the credentials of the user log-in whether it is authenticated or not. *Responsibilities:* public void CheckCredentials(String username, String password)

Entity Classes:

Class Name	Description
Food and Water Intake Records	This is the entity class Food and Water Intake Records, which contains the data about the food and water intakes added by the user. *Attributes*: private int waterintakeid private int foodnameid private String watercontainer private String foodname private float amount private float calorie private Date date_consumed private Date time_consumed
Users	This is the entity class Users, which contains all the information about the users in the health and fitness management record. *Attributes*: private int userid public String username private String password
User Profile	This is the entity class User Profile, which contains the user's profile information. *Attributes*: private int userID private float weightInKilograms private float heightInCentimeters private String ContactNumber private String Email private String Username private Date Birthdate private int healthConditionID private String healthCondition

Behavioral Model:

Use-Case Name: Add Meal or Water Intake

Description:

The user will be able to add a food or a water intake, and with that user describes when they ate the food, or drank the water. User must be able to add the quantity of the food/water intake.

Scenario 1: User adds a food intake.
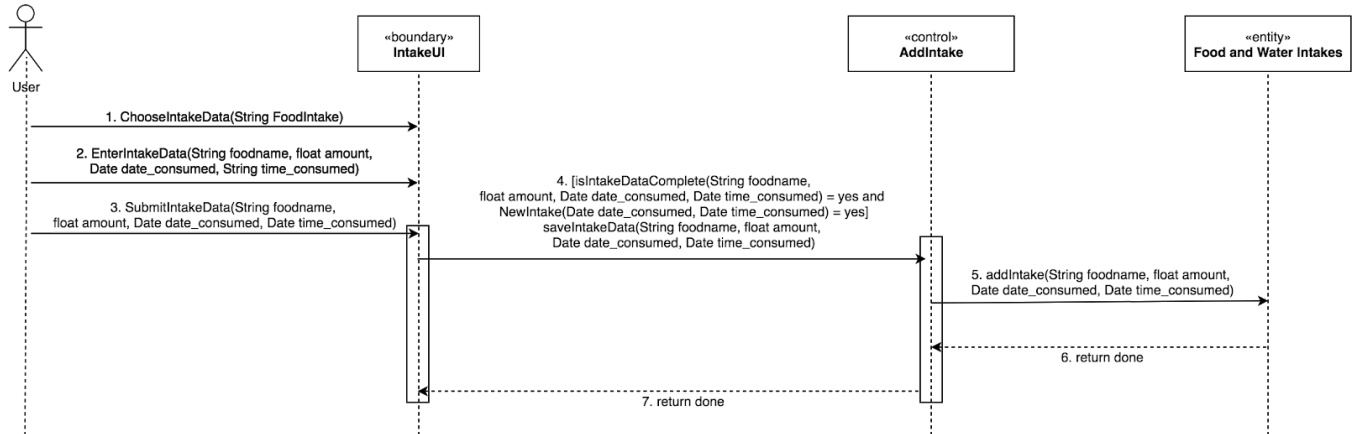
Scenario 2: User adds a water intake.
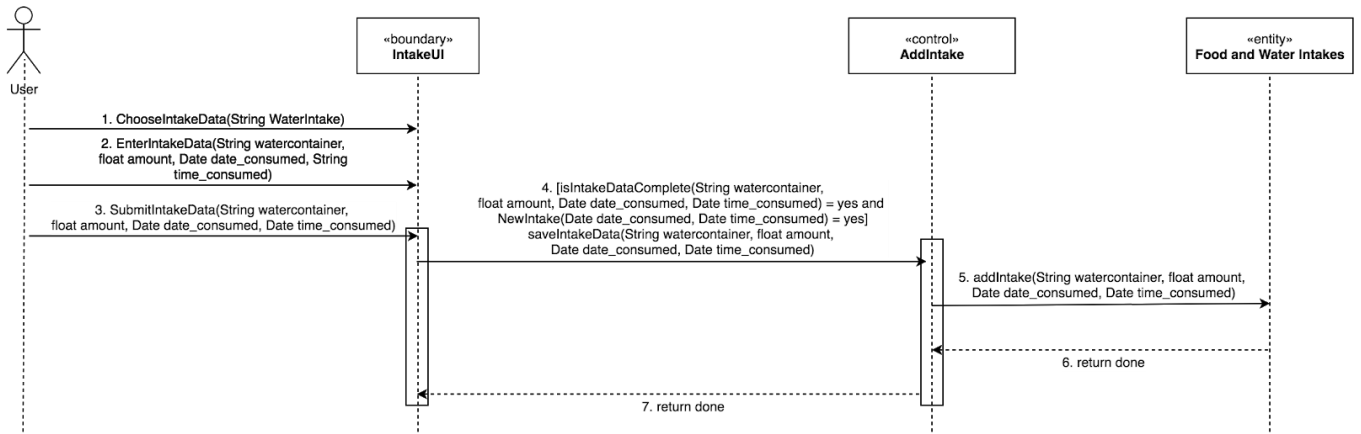
***Use-Case Name*:**   Edit Food or Water Intake

Description:

After the user is done adding a food or a water intake, the user may edit their input. User may also edit their previous intake/s in the records.

Scenario 1: User edits a food intake.
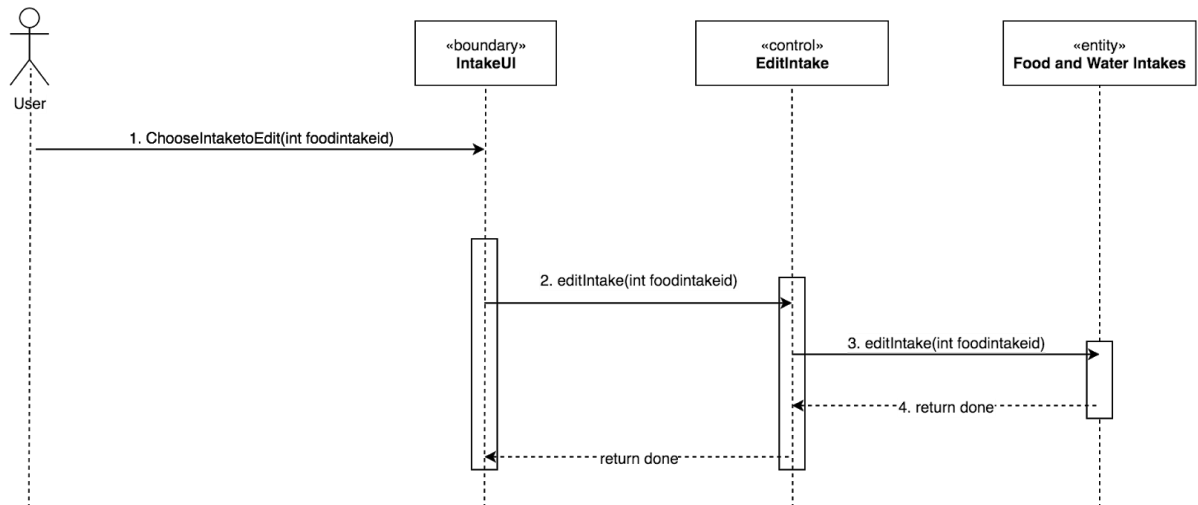
Scenario 2: User edits a water intake.
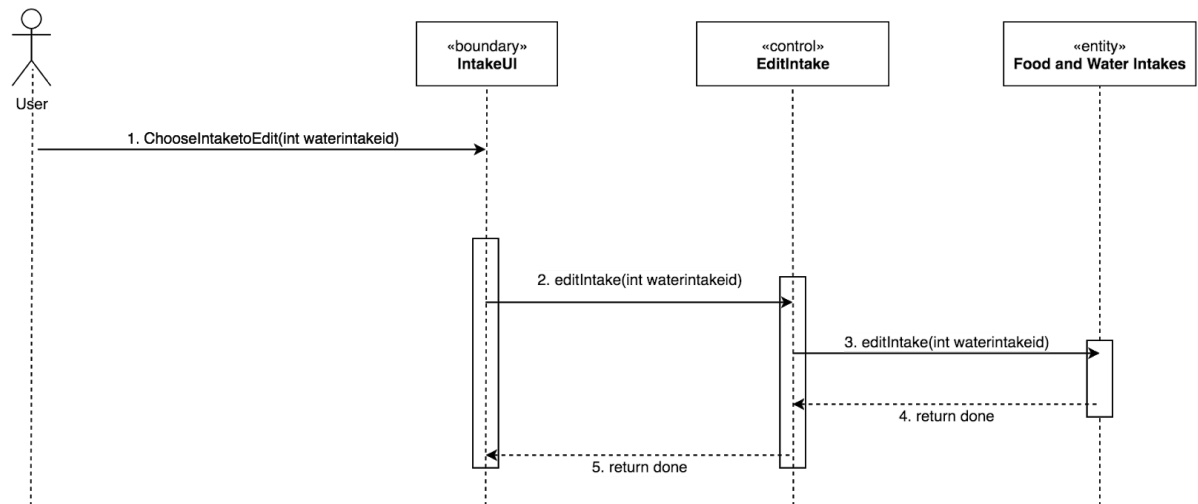
Use-Case Name: Delete Food or Water Intake

Description:

When an intake isn't needed anymore or accidentally added, all attributes associated with intake may be removed or deleted. User may clean up or declutter their records using this function.

Scenario 1: User deletes a food intake.
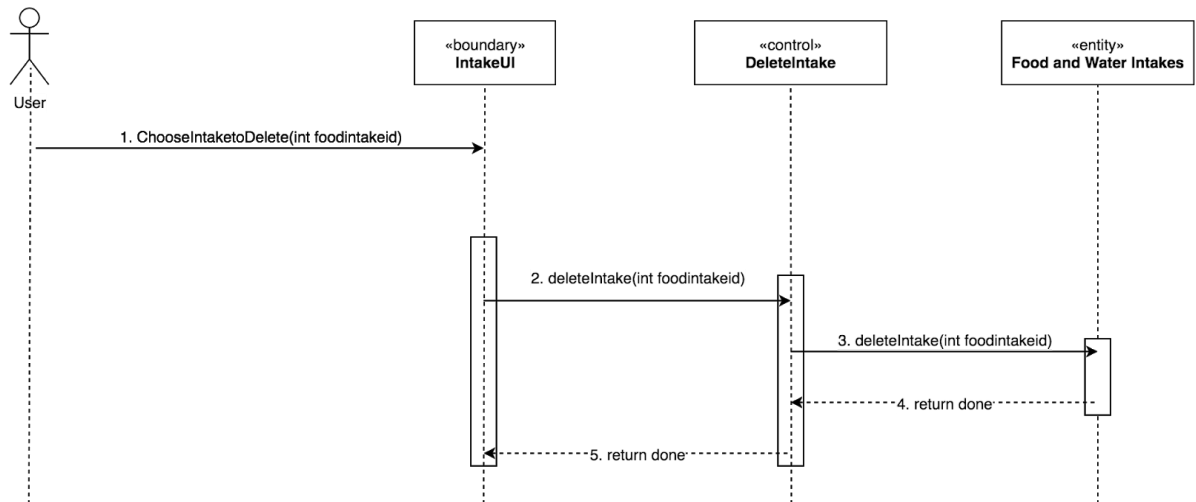
Scenario 2: User deletes a water intake.
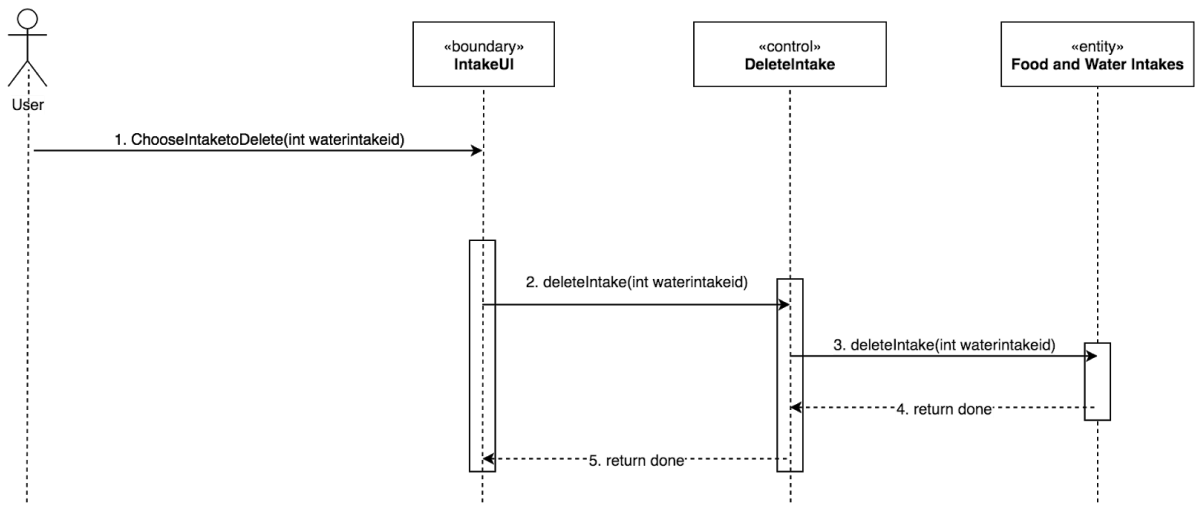
Use-Case Name: Edit Weight or Height

Description:

From the user's given profile, this function can edit his/her height. User can also edit his/her previous weight, because in the management of the system, user tracks his/her growth in terms of height, or gain/loss of user's weight.

Scenario 1: User edits weight/height successfully
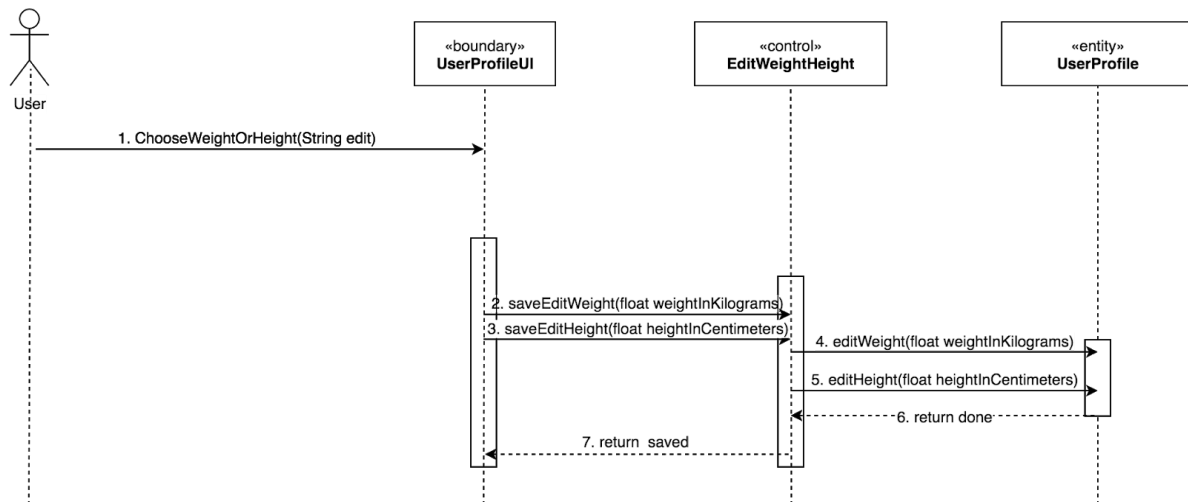
Scenario 2: User edits weight/height unsuccessfully
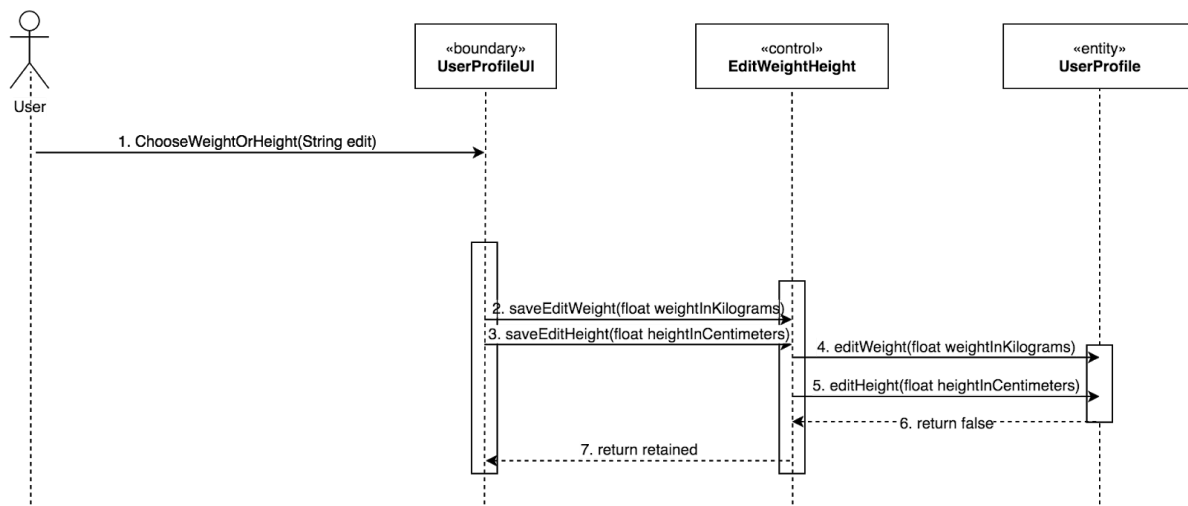
Scenario 3: First time user edits weight/height successfully
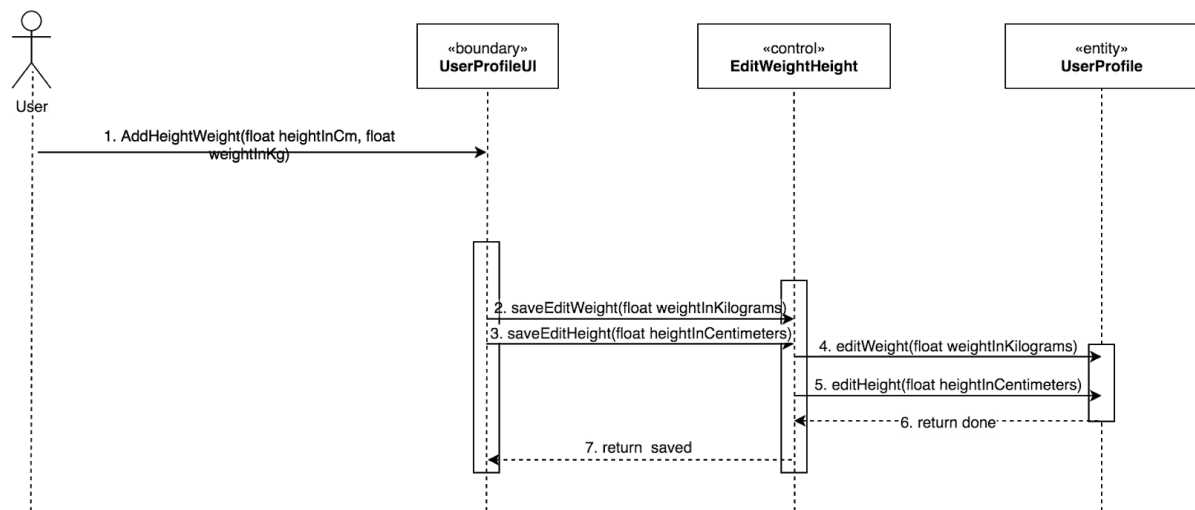
Use-Case Name: Edit Personal Details

Description: Using this function, user may edit the other personal details besides height, weight and other health conditions.

Scenario 1: User edits personal details successfully
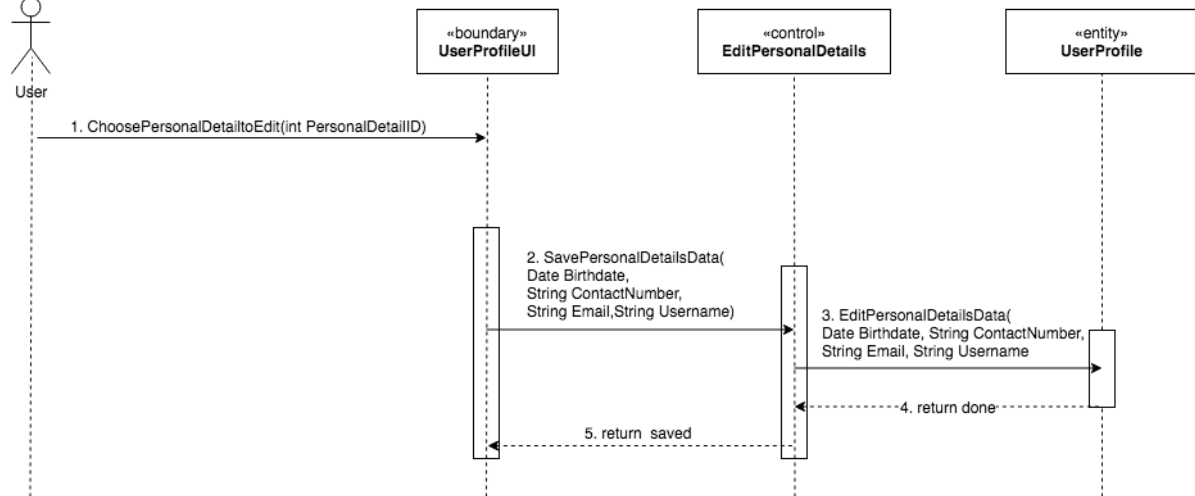
Scenario 2: User edits personal details unsuccessfully
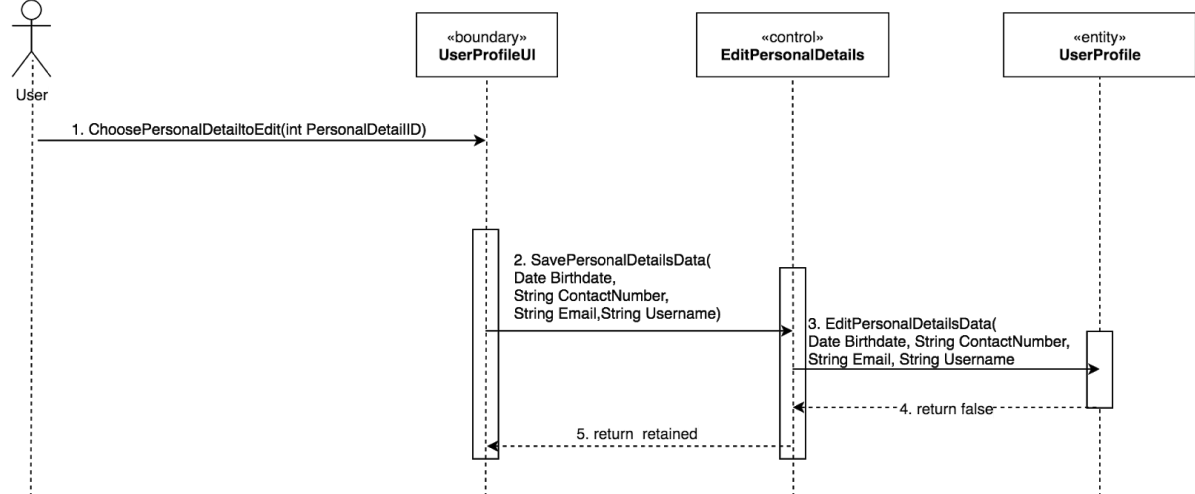
Scenario 3: First time user edits personal details successfully
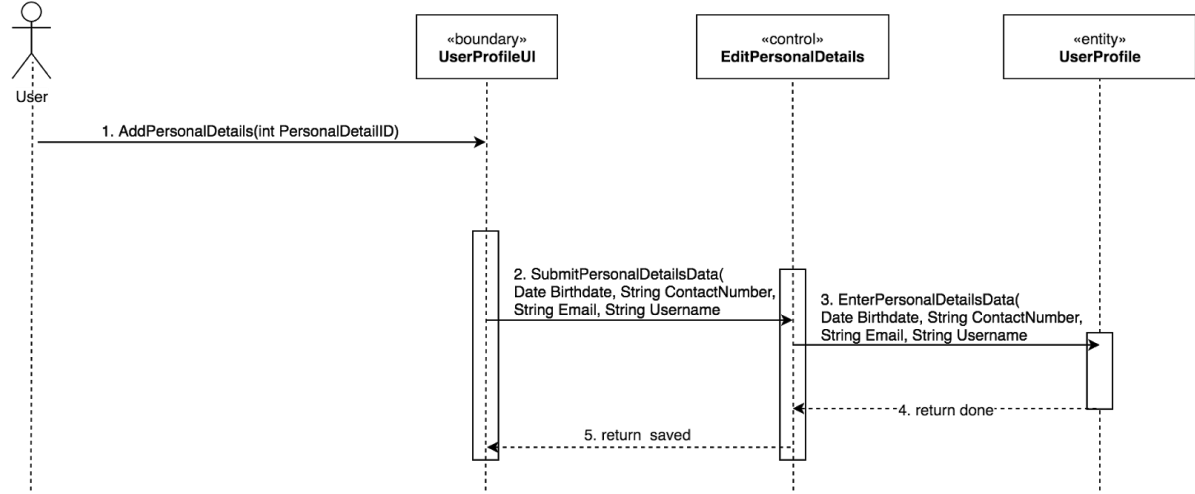
Use-Case Name: Add Health Conditions

Description:

From the user's given profile, this function can edit his/her health conditions. This function helps the user established his/her health profile. Health conditions may be left blank.

Scenario 1: User adds health condition/s successfully
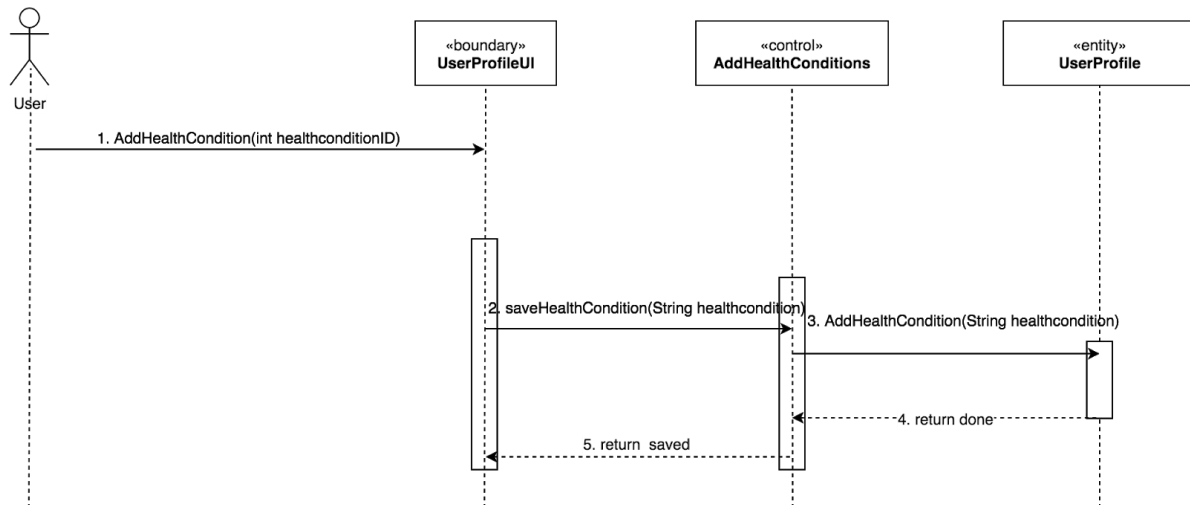
Scenario 2: User edits health condition/s successfully
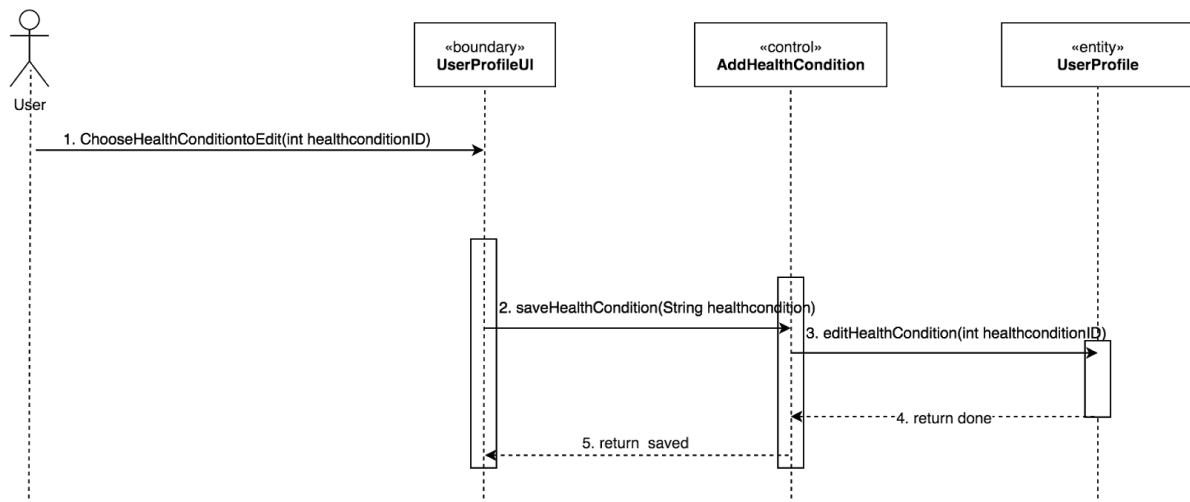
Scenario 3: User edits/deletes health condition/s unsuccessfully
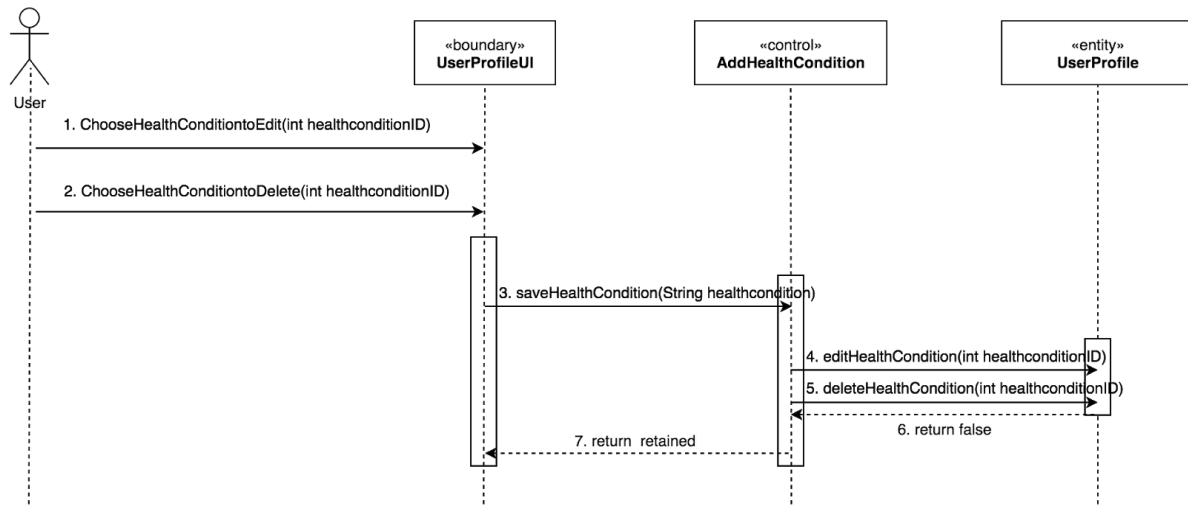
Scenario 4: User deletes health condition/s successfully
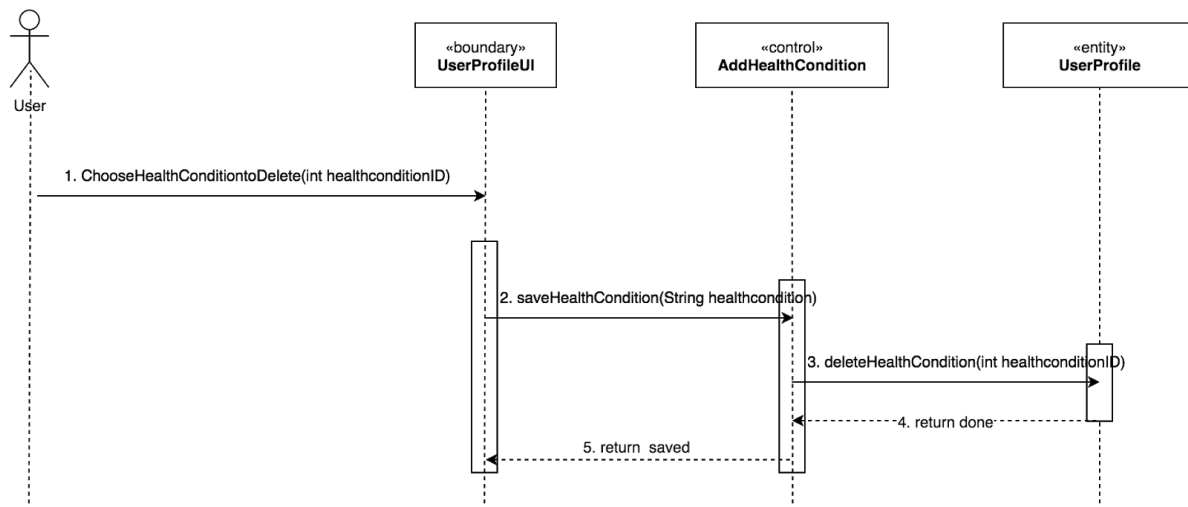
Scenario 5: First time user leaves health conditions blank
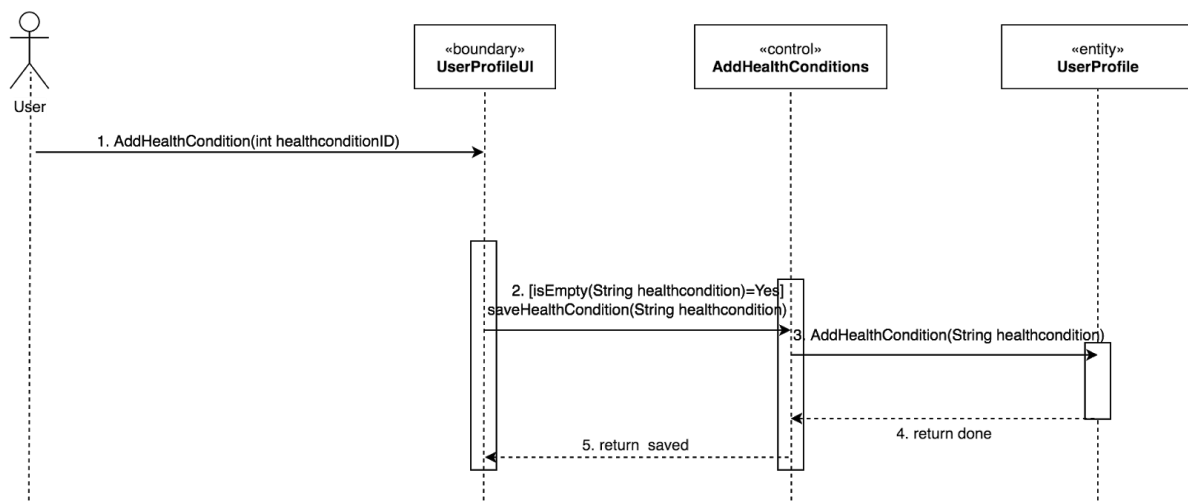
Scenario 6: First time user adds/edits health condition/s
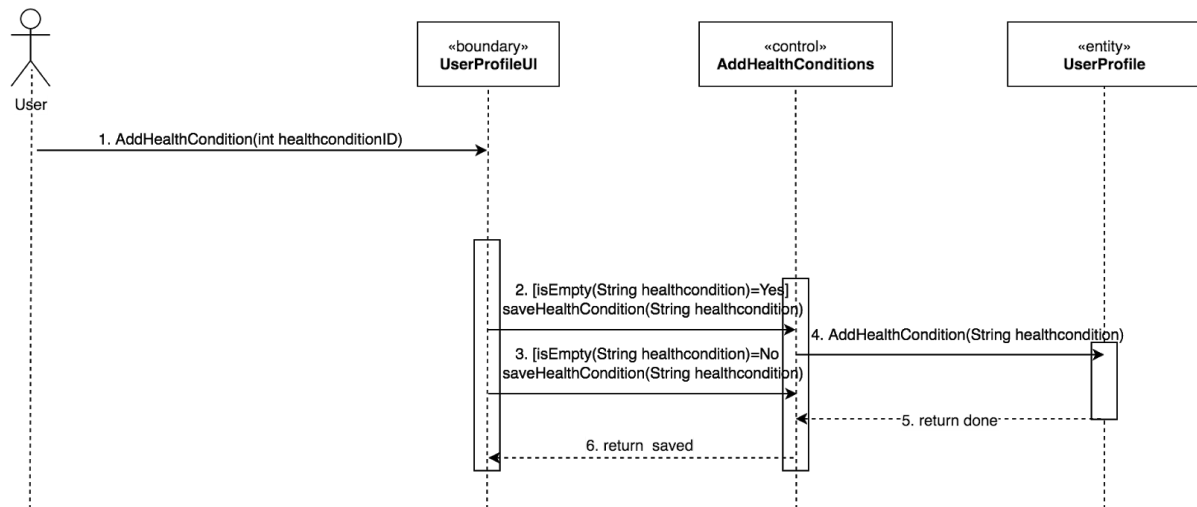
Use-Case Name: View Current Intake

Description:

The user can view his/her own user profile where he/she can see his/her current meal/water intake
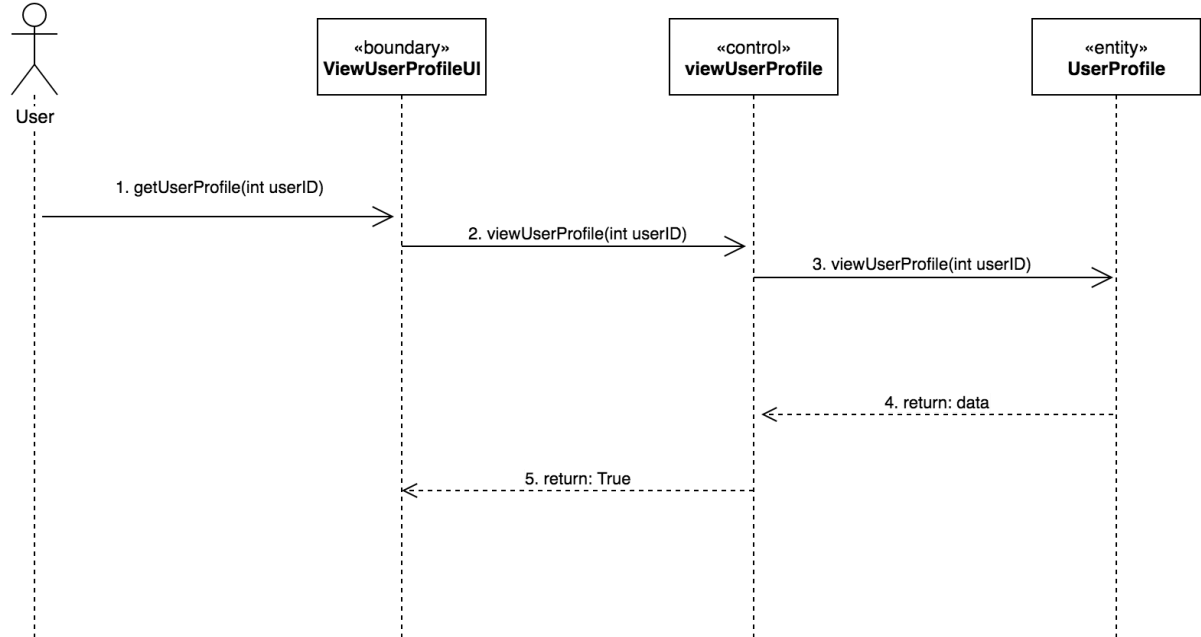
Scenario 1: User is able to view all his food or water intake
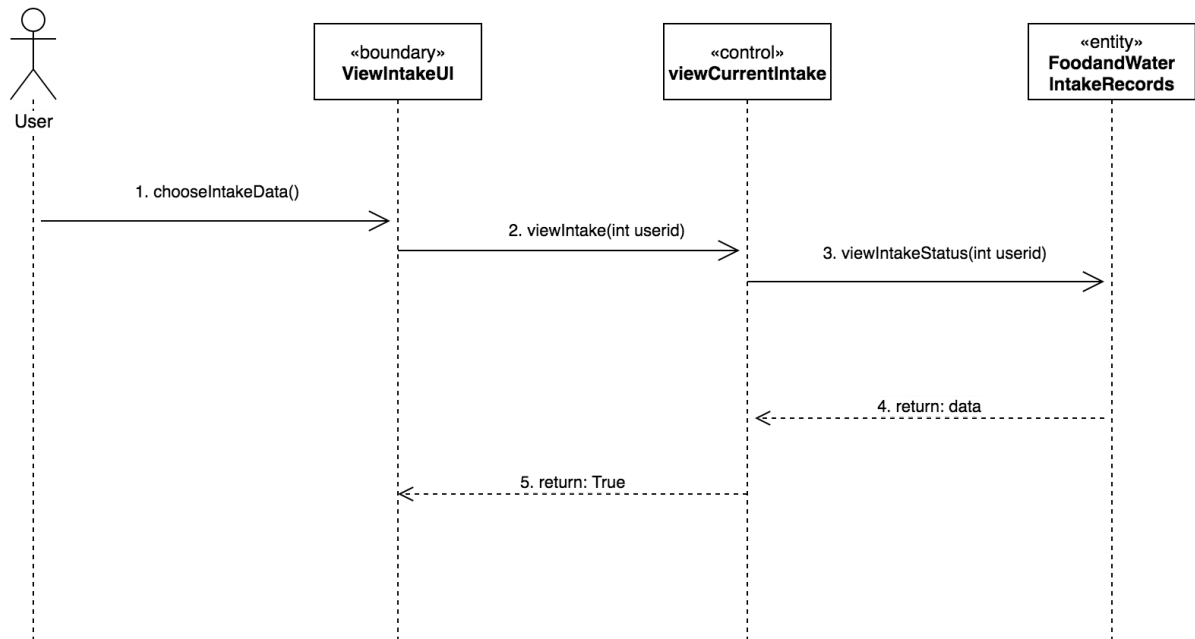
Use-Case Name: View User Profile

Description:

The user can view his/her own user profile where he/she can see his/her current height, weight, health conditions, and other personal details.

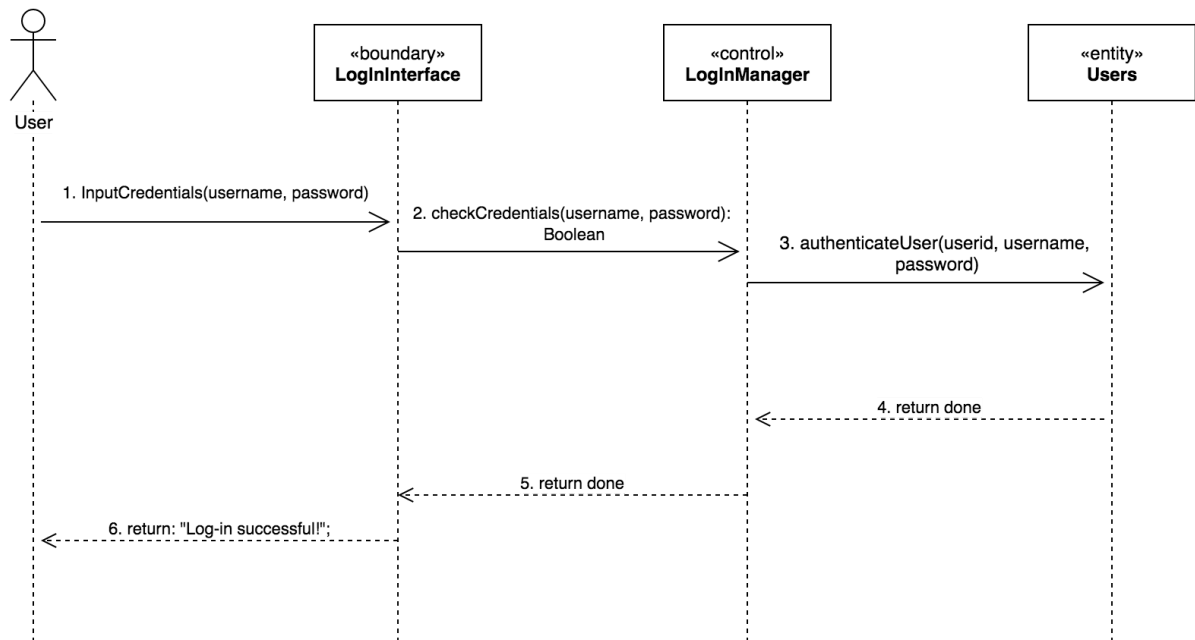Scenario 1: User is able to view his/her user profile.
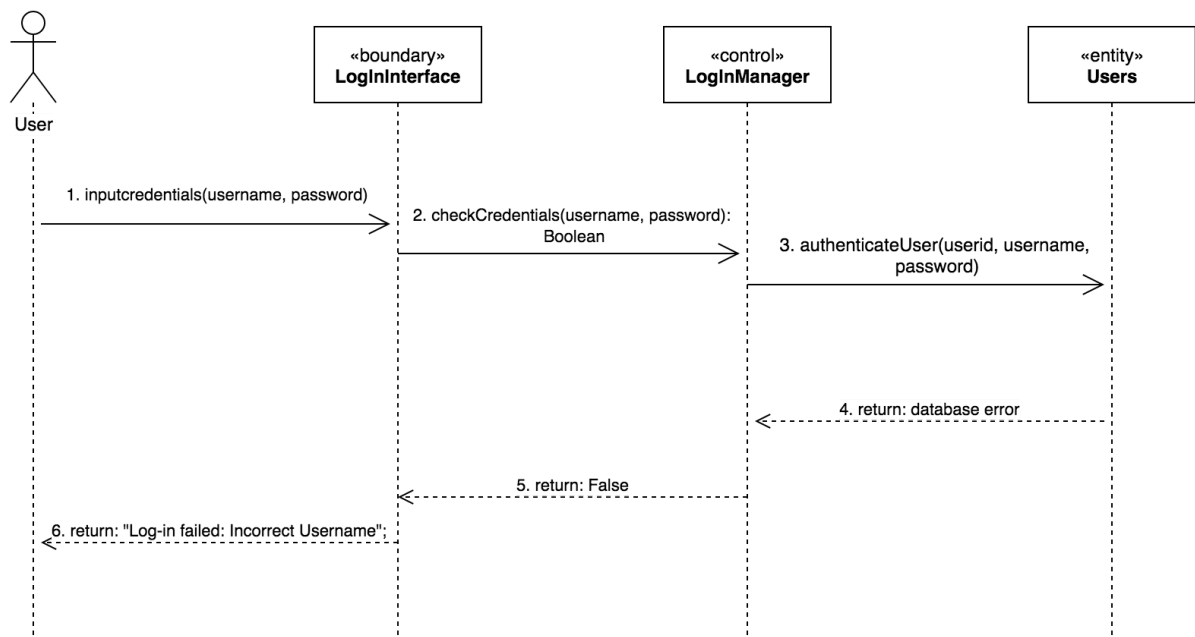

Use-Case Name: Log-in

Description:

The user must create his/her own account and login using his/her own account in order to start viewing his health and fitness records and maintain it accordingly.

Scenario 1: User logs his or her username and password successfully.

Scenario 2: User logs username and password, but the entered username is invalid.

Scenario 3: User logs username and password, but the entered password is invalid.