SEP Online

Requirements Specification and Analysis

v.1.0

12 March 2018

Team Members

Oğuzhan Ulusoy
Ayşenur Bayram
Tahir Tekizbaş
Didar Turanlı
Dilara Turanlı

Prepared for
SE302 Software Engineering

IŞIK UNIVERSITY
COMPUTER SCIENCE AND ENGINEERING

Table of Contents

REQUIREMENTS ANALYSIS DOCUMENT [1]

1. Introduction

1.1. Purpose of the System

SEP Online project is a social media network. The main goal of SEP Online is to display items as daily, weekly and periodically, and the other is to provide to meet with consumers and local producer.

1.2. Scope of the System

As we have mentioned at Purpose of the System section, SEP Online is social media network and has two main targets. Displaying items as daily, weekly and periodically, and satisfying to meet with consumers and local producer are main targets.

There are two actors except admin. First actor is normal user, second actor is author. Author is a normal user, also have an authority to write blog. They are approved by admin at registration process.

Local markets that are called local producers, are shown by region on map. It is completely free and user friendly. Every actor can comment in to posts and suggest an item that is not added to the system. Searching, filtering and sharing are other features of actors. Domain is to be determined on based healthy life and foods. Each content is free and the achieve was prepared by general information.

Admin has many authorities on system. Admin can add, change or delete local market and item. Admin also can add or delete local. Admin can display all members (*both normal users and authors*). They can be deleted from the system by admin. Author is approved or rejected by admin. Admin can display the suggestion for items that are come from any member. Admin can remove any blog and comment from the system.

1.3. Objectives and Success Criteria of the Project

The success of the system depends on providing the given main set of arguments:

- To provide reliable, efficient, lossless data.
- Well association between platform and database design.
- The general design of system in order to have fast, efficient system.
- The demo of system should ensure good success rate.
- The system has implementations understandable, clear and efficient.
- The system should be used by the people who is related to domain.
- The system should keep protected and guarantee every user’s information and their data.

SEP Online

1.4. Definitions, Acronyms, and Abbreviations

- View is a visual representation of a model.
- Domain is a name and address of web-site.
- GUI is graphical user interface.
- DB is short version of database term.
- Author is an actor on SEP Online, who is a normal user that is approved by author.
- Normal user is an actor on SEP Online.
- HTTP is a protocol for secure communication over a computer network which is widely used on the internet.
- Model is a schematic description of a system that accounts for its known or inferred properties.

1.5. Overview

- Rest of the RAD contains non-functional (*includes usability, reliability, performance, supportability, implementation, interface, operational, packaging, and legal requirements*) and functional requirements (*includes high-level functionality of the system*).
- System models are given. Scenarios are in side of system models section. Scenarios are to tell us the details of functional requirements. Use case models, object model, dynamic model and user interface view (*mockup*) are parts of system models section.

2. Current System

There are some websites that are related to local producers. These websites are to provide shopping for consumer. The web sites mentioned are TazeDirekt.com[2], TazeMasa.com[3] and MemlekettenGelsin.com[4]. There are some websites that are related to healthy life including blog-posts. One of the web sites mentioned is vitaminrehberi.org[5]. There is no web-site which has goals like the we proposed system.

3. Proposed System

We are going to try to develop a social network that is completely free. The web site is user friendly, and useful for everyone. Local markets are shown by region on map to members. Items (*both vegetables and fruits*) are shown with all details for 100-grams. Blog-posts are shown for members. We will try to develop a search algorithm based on Google that is to prepare proper result for selecting items.

3.1. Overview

The system will be including three actors. These are named normal user, author and admin. Author is a normal user who has authority to write blog. Functional Requirements section contains what any actor can be done.

SEP Online

3.2. Functional Requirements

Functional requirements for normal user are given below.

- Normal user shall be able to register.
- Normal user shall be able to login.
- Normal user shall be able to comment.
- Normal user shall be able to suggest new item.
- Normal user shall be able to display item as filtering.
- Normal user shall be able to display local market as filtering.
- Normal user shall be able to search.
- Normal user shall be able to edit the profile.

Functional requirements for author are given below.

- Author shall be able to register.
- Author shall be able to login.
- Author shall be able to comment.
- Author shall be able to suggest new item.
- Author shall be able to display item as filtering.
- Author shall be able to display local market as filtering.
- Author shall be able to search.
- Author shall be able to edit the profile.
- Author shall be able to write blog.

Functional requirements for admin are given below.

- Admin shall be able to login.
- Admin shall be able to create an admin.
- Admin shall be able to comment.
- Admin shall be able to display item as filtering.
- Admin shall be able to display local market as filtering.
- Admin shall be able to search.
- Admin shall be able to edit the profile.
- Admin shall be able to add, change or delete item.
- Admin shall be able to add, change or delete local market.
- Admin shall be able to delete member (both normal user and author).
- Admin shall be able to delete comment or post.
- Admin shall be able to display suggestion from members.

Any visitor that is not actor on the system, can do given:

- To display item as filtering.
- To display local market as filtering.
- To search.

SEP Online

3.3. Nonfunctional Requirements

Usability

The SEP Online should have three different navigation menus for all actor. Normal user should use standard navigation menu. Author should have a navigation menu that is including standard navigation menu and “write blog” link. Admin should have a navigation menu that is including standard navigation menu and “admin panel” link. Each page should contain same header, footer, head, content blocks. Almost every page (both static and dynamic) should have same body page. Profile page, register, login is basic-level membership. The content should be shown as quite simple. There should no guide for users, because SEP Online should be created easier for users.

Reliability

The SEP Online should be provided to access for all members by secure. Unregistered visitor should be able to do some stuff. Yet, registered visitor, which are author and normal user, should be able to do some stuff more different than unregistered visitor. Logging to the web-site should be provided with unique e-mails and password that are appropriate for password criteria. The passwords should be hit as hashed password in database. The membership information should be private and should not be shared with anyone. Session and PHP Data Object are used to provide reliability.

Performance

The SEP Online should be responsive in different devices. SEP Online should be shown in mobile phones, tablet and computers. SEP Online should be running in more than one computer simultaneously, and access should be guaranteed. The web-site is going to be dynamic content, so there should not be complicated queries in back end to not decrease performance.

Supportability

The system should be managed by admin. The system should be supported on different browser and be independent by hardware. In FTP side, there are going to be some content components (*i.e. header, head, script, footer, and so on*) to maintain the system easily, rapidly and properly.

Implementation

There are no constraints on the hardware platform. There are no constraints imposed by the maintenance team. There are no constraints imposed by the testing team. The design methodology is MVC (model-view-controller). We will check input over controller, our model is satisfied object relation model over SQL. PHP language is used in this system. HTML5, CSS and Bootstrap (*briefly*) should be used to implement.

SEP Online

Interface

The system should not interact with any existing system. The system should be able to use by a user. The user should be connected to the network to use the features of the system.

Packaging

Admin should install the system. Also, the system is a web site so the site should uploaded on the server. There should no time constraints on the installation.

Legal

The SEP Online does not use any license or licensed stuff. It is a social media network.

3.4. System Models

Scenarios

Scenario Name: <u>Add New Item</u>
Participating Actor Instances: Oğuzhan: System Admin
Flow of Events:

1. Oğuzhan is a person who defined by the system administrator at the beginning of the project. So, he has own panel to manage the system. Now, the new item should be added in to the system. Therefore, he can login to system as an admin.
2. Oğuzhan goes to admin panel, then admin functions will be open in the incoming page. He clicks to “Add New Item” linked text-label.
3. “Add New Item” page has own attributes: name, protein, fat, carbohydrate, mineral, vitamin, reference time, (*briefly*) short description and photo. Oğuzhan adds “Broccoli” as a new item and types its attributes are name: “Broccoli”, protein: “4.95g”, fat: “0.3g”, carbohydrate: “3.77g”, mineral: “559.5mg”, vitamin: “256mg”, reference time: “winter”, short description: “eat for your health” and photo: “…uploads/brokoli.jpg”.
4. If there is a mistake, it can be edited before saved. (It can be changed later.)
5. Oğuzhan clicks to the “Save" button to add the item to the list.

Scenario Name: <u>Delete Item</u>
Participating Actor Instances: Oğuzhan: System Admin
Flow of Events:

1. Oğuzhan is a person who defined by the system administrator at the beginning of the project. So, he has own panel to manage the system. Now the item should be deleted from the system. Therefore, he can login to system as an admin.
2. Oğuzhan goes to admin panel, then admin functions are appeared in here. He clicks to “Delete Item” linked text-label.
3. The “Delete Item” page is opening, then a list will be here. The list is list of the existing item on the system. They are shown as a line for an item.

4. Each line has a ratio button. (Also, each line has "See details" linked text-label to see own page.)
5. Oğuzhan wants to remove an item from list. Thus, he selects an item that wants to remove, and clicks to check box. The item is going to be selected.
6. Oğuzhan clicks to "Save" button to finish the operation.

Scenario Name: Delete Admin
Participating Actor Instances: Oğuzhan: System Admin
Flow of Events:

1. Oğuzhan has to remove an admin on the system. He logs in to the system as admin. Then, he goes to the admin page who is private a page.
2. He clicks to "Delete Admin" linked text-label. After a while, that page will be appeared. It has own attributes. They are admin identification number, admin name and surname, admin e-mail, admin phone.
3. Each instance has a check-box at beginning of the line.
4. Oğuzhan selects an admin that will be removed from the system, then clicks to the check-box.
5. Oğuzhan clicks to "Save" button to finish the operation.

Scenario Name: Create Admin
Participating Actor Instances: Oğuzhan: System Admin
Flow of Events:

1. Another admin is required to operate the system. Oğuzhan is an also admin. He logs in to the system as admin. Then, he is redirected to admin panel where there are functions of the admin to manage the system.
2. He clicks to "Create Admin" linked text-label.
3. Incoming page has own attributes. These are first name, last name, e-mail, password, phone and profile picture. He types information to the related fields. First name: "Zeynep Selen", last name: "Görkem", e-mail: "zeynepselengorkem@gmail.com", password: "19051995", phone: "05373119592", profile picture: "..uploads/pp/zsg_100.jpeg".
4. If there is no mistake, (if so, it can be changed momentarily), he clicks to "Save" button to finish the operation.

Scenario Name: Change Profile Picture
Participating Actor Instances: Oğuzhan: System Admin, Author, Normal User
Flow of Events:

1. Oğuzhan is a person who defined a system administrator at the beginning of the project. His task (or role) is to manage the system, and also he has own profile. He wants to change his profile picture. Then, he logs in to the system. Then, he goes to "My Profile" linked text-label on the header section.

2. Incoming page has own attributes. These are first name, last name, e-mail, password, phone and profile picture. Each attribute has "Change" button. "Oğuzhan" for first name, "Ulusoy" for last name, "ulusoyoguzhan@gmail.com" for e-mail, "19051995" for password, "05373119592" for phone and "…uploads/pp/ou.jpeg" for profile picture.
3. He clicks to "Change" button of the profile picture field. Then, a new window is going to be opened. He should select a new profile picture from here. Therefore, he chooses a profile picture.
4. Then, he clicks to "Save" button to finish the operation.

Scenario Name: Add Location
Participating Actor Instances: Tahir: System Admin
Flow of Events:

1. Tahir is a person who defined by the system administrator at the beginning of the project. So, he has own panel to manage the system. Now, the location should be added in to the system. Therefore, he can login to system as an admin.
2. Tahir goes to admin panel, then admin functions will be open in the incoming page. He clicks to "Add Location" linked text-label.
3. When "Add Location" page is loaded, there is a form in here with an attribute. That is name field.
4. Tahir adds a location as "Istanbul" for name.
5. Then, he clicks to "Save" button to finish the operation.

Scenario Name: Delete Location
Participating Actor Instances: Tahir: System Admin
Flow of Events:

1. Tahir is a person who defined by the system administrator at the beginning of the project. So, he has own panel to manage the system. Now, the location should be deleted from the system. Therefore, he can login to system as an admin.
2. Tahir goes to admin panel, then admin functions will be open in the incoming page. He clicks to "Delete Location" linked text-label.
3. When "Delete Location" page is loaded, there are locations in here with ratio button.
4. Tahir selects a location to remove as "Istanbul" for name.
5. Then, he clicks to "Save" button to finish the operation.

Scenario Name: Change Location
Participating Actor Instances: Tahir: System Admin
Flow of Events:

1. Tahir is a person who defined by the system administrator at the beginning of the project. So, he has own panel to manage the system. Now, the location should be changed in the system. Therefore, he can login to system as an admin.

2. Tahir goes to admin panel, then admin functions will be open in the incoming page. He clicks to “Change Location” linked text-label.
3. When “Change Location” page is loaded, there are locations as text-fields.
4. Tahir clicks to any location whatever he should change the location. He clicks a field as “Istanbul” for name, then he cleans the field and types “Giresun” for name.
5. Then, he clicks to “Save” button to finish the operation.

Scenario Name: Add Local Market
Participating Actor Instances: Tahir: System Admin
Flow of Events:

1. Tahir is a person who defined by the system administrator at the beginning of the project. So, he has own panel to manage the system. Now, the local market should be added in to the system. Therefore, he can login to system as an admin.
2. Tahir goes to admin panel, then admin functions will be open in the incoming page. He clicks to “Add Local Market” linked text-label.
3. When “Add Local Market” page is loaded, there is a form in here with some attributes which are name, phone, mail, address, picture, description and location (actually it is location identification number.
4. Tahir types some variables – “Musu” for name, “05373119592” for phone, “musutarim@gmail.com” for mail, “Giresun, Görele, Şahinyuva” for address, “…uploads/localMarkets/1.jpg” for picture, “Taze finduk” for description and “Giresun” for location (ID: 28) - to add local market.
5. Then, he clicks to “Save” button to finish the operation.

Scenario Name: Delete Local Market
Participating Actor Instances: Tahir: System Admin
Flow of Events:

1. Tahir is a person who defined by the system administrator at the beginning of the project. So, he has own panel to manage the system. Now, the local market should be deleted from the system. Therefore, he can login to system as an admin.
2. Tahir goes to admin panel, then admin functions will be open in the incoming page. He clicks to “Delete Local Market” linked text-label.
3. When “Delete Local Market” page is loaded, there is a list in here with some attributes and ratio button which are name, phone, mail, address, picture, description and location (actually it is location identification number.
4. The list contains some variables. He selects an instance’s ratio button from the list – “Musu” for name, “05373119592” for phone, “musutarim@gmail.com” for mail, “Giresun, Görele, Şahinyuva” for address, “…uploads/localMarkets/1.jpg” for picture, “Taze finduk” for description and “Giresun” for location (ID: 28).
5. Then, he clicks to “Save” button to finish the removing operation.

Scenario Name: Display Local Market
Participating Actor Instances: Unregistered, Author, Tahir: Normal User, System Administrator
Flow of Events:

1. Tahir is a member whose role normal user. Therefore, he can login to system through as his role in system [*Scenario: Login*].
2. Tahir goes to "Local Market" page over menu navigation bar.
3. There are going to be shown local markets. Tahir selects and instance, then goes to inside of it.
4. "Musu" for name, "…uploads/localMarkets/1.jpg" for picture, "Taze finduk" for description.
5. When inside of its is loaded in new page, there are going to be another details in here which are name, phone, e-mail address, address, photo and description. "Musu" for name, "05373119592" for phone, "musutarim@gmail.com" for mail, "Giresun, Görele, Şahinyuva" for address, "…uploads/localMarkets/1.jpg" for picture, "Taze finduk" for description

Scenario Name: Change Local Market
Participating Actor Instances: Tahir: System Admin
Flow of Events:

1. Tahir is a person who defined by the system administrator at the beginning of the project. So, he has own panel to manage the system. Now, the local market should be changed in the system. Therefore, he can login to system as an admin.
2. Tahir goes to admin panel, then admin functions will be open in the incoming page. He clicks to "Change Local Market" linked text-label.
3. When "Change Local Market" page is loaded, there is a list in here with some attributes and text-fields which are name, phone, mail, address, picture, description and location.
4. The list contains some variables. He selects an instance to change an instance from the list – "Musu" for name, "05373119592" for phone, "musutarim@gmail.com" for mail, "Giresun, Görele, Şahinyuva" for address, "…uploads/localMarkets/1.jpg" for picture, "Taze finduk" for description and "Giresun" for location (ID: 28).
5. He can change text-fields of he selected instance. He types new phone number to phone attribute. "05385607279" for phone (updated phone number), instead of "05373119592" for phone.
6. Then, he clicks to "Save" button to finish the operation.

Scenario Name: <u>Suggest-an-Item</u>
Participating Actor Instances: Ayşenur: Author, Normal User, Oğuzhan: System Admin
Flow of Events:

1. Ayşenur is a member who was approved by Oğuzhan. She can log in to the system with her authentication information (*i.e. e-mail address and password*) over login panel. She has a standard menu with "Write Blog" linked text-label.
2. Ayşenur goes to "Suggest Item" page on menu-navigation bar from above the header section.
3. When "Suggest Item" page is loaded, there is an input form with some attributes, which are name, protein, fat, carbohydrate, mineral, vitamin, reference time, (*briefly*) short description and photo. Ayşenur adds "Broccoli" as a new item and types its attributes are name: "Broccoli", protein: "4.95g", fat: "0.3g", carbohydrate: "3.77g", mineral: "559.5mg", vitamin: "256mg", reference time: "winter", short description: "eat for your health" and photo: "…uploads/brokoli.jpg".
4. After she has typed some variables to related input text-fields, clicks to "Save" button to finish the operation and waits for approval from Oğuzhan. This is validation process.

Scenario Name: <u>Add Comment</u>
Participating Actor Instances: Ayşenur: Normal User, Author, System Administrator
Flow of Events:

1. Ayşenur is a member who is normal user. She can log in to the system with her authentication information (*i.e. e-mail address and password*) over login panel. She has a standard menu.
2. Ayşenur goes to "Blogs" page on menu-navigation bar from above the header section.
3. Then, she clicks to any blog-post title. It is linked text-label. "Hello World!" for title of blog. Each title has a url that is through unique identification number of blog.
4. Thus, she goes to clicked blog's page. There is a small input form. It is for adding new comment. Each form has two attributes which are comment and commentor. They are text-fields. "The first comment of this blog-post" for comment, "Ayşenur" for commentor.
5. Then, she clicks to "Save" button to finish the operation.

Scenario Name: <u>Delete Comment</u>
Participating Actor Instances: Oğuzhan: System Administrator
Flow of Events:

1. Oğuzhan is a person who was defined as admin at beginning of the project. He can log in to the system with his authentication information (*i.e. e-mail address and password*) over login panel. He has a standard menu with "Admin Panel" linked text-label.

2. Oğuzhan goes to “Admin Panel” page on menu-navigation bar from above the header section. When the “Admin Panel” page is loaded, there are functions which are managed by Admin.
3. Then, he clicks to “Delete Comment” linked text-label. When the “Delete Comment” page is loaded, there is a list that contains some attributes that are ID of comment, commenter name and comment text. Each row has a ratio button at the beginning of the line.
4. Oğuzhan selects an instance to delete from the system. “28” for ID of comment, “Ozi” for commenter and “Hello, this is the first comment of the this blog post!” for comment.
5. After he selected an instance, he clicks to “Save” button to finish the operation.

Scenario Name: Add Blog
Participating Actor Instances: Ayşenur: Author
Flow of Events:

1. Ayşenur is a person who was approved by system administrator at after registration. She can log in to the system with her authentication information (*i.e. e-mail address and password*) over login panel. She has a standard menu with “Write Blog” linked text-label.
2. Ayşenur goes to “Write Blog” page on menu-navigation bar from above the header section. When the “Write Blog” page is loaded, there are input text-fields, which are title and text.
3. Then, she types some variables on to the relevant fields. “Hello World!” for title, “Hello guys, this is my first blog post” for text.
4. After she has typed some variables, he clicks to the “Save” button to finish the operation.

Note: Add Blog is name of function in back-end, Write blog is name of the function’s in the menu navigation bar.

Scenario Name: Delete Blog
Participating Actor Instances: Ayşenur: System Administrator
Flow of Events:

1. Ayşenur is a person who was defined as admin at the beginning of the project. She can log in to the system with her authentication information (*i.e. e-mail address and password*) over login panel. She has a standard menu with “Admin Panel” linked text-label.
2. Ayşenur goes to “Admin Panel” page on menu-navigation bar from above the header section. When the “Admin Panel” page is loaded, there is a list of functions which are managed by admin.
3. Ayşenur has to delete a blog post, so she clicks to “Delete Blog” linked text-label.
4. When the “Delete Blog” page is loaded, there is a table that includes instances with ID of blog, title of blog, text of blog. Each row has a ratio button at beginning of the

line. “34” for ID of blog, “Hello World” for title of blog, “Hello guys, this is my first blog post” for text of blog.

5. She selects an instance with ratio button, whatever she wants to delete. Then, she clicks to “Save” button to finish the operation.

Scenario Name: Search
Participating Actor Instances: Ayşenur: Author, Normal User, System Administrator
Flow of Events:

1. Ayşenur is a person who was approved as an author by system administrator. She can log in to the system with her authentication information (*i.e. e-mail address and password*) over login panel.
2. Ayşenur goes to “Search” linked text-label. There are three components in here which are search bar, search and clean buttons.
3. This structure is based on Google search. It returns results from Google searching results. This is done with a iFrame code.

Scenario Name: Change Member Details
Participating Actor Instances: Dilara: Normal User, Author
Flow of Events:

1. Dilara is a member who is a normal user. She can log in to the system with her authentication information (*i.e. e-mail address and password*) over login panel.
2. When Dilara is redirected to main page, she clicks to “My Profile” linked text-label.
3. There are some components which relevant input text-fields. These are first name, last name, phone, e-mail address. Also, there are another two components which are “change my password” and “change my profile picture” linked text-labels. They are another scenarios, they will be mentioned in other scenarios. In addition to these, she can display her unique identification number which is generated by the back-end system at registration phase.
4. Dilara can see her membership information. “Dilara” for first name, “Turanlı” for last name, “05385207279” for phone, “dilara.turanli@hotmail.com” for e-mail address. She wants to clicks any input text-field to typing new information. (i.e. She wants to change her e-mail address. She types new e-mail address information to relevant field. “dilara.turanli@hotmail.com” for old e-mail address, “dilara.turanli@gmail.com” for new e-mail address.
5. Then, she clicks to “Save” button to finish the operation.

Scenario Name: Change Member Password
Participating Actor Instances: Dilara: Normal User, Author
Flow of Events:

1. Dilara is a member who is a normal user. She can log in to the system with her authentication information (*i.e. e-mail address and password*) over login panel.

2. When Dilara is redirected to main page, she clicks to "My Profile" linked text-label. There are some components which relevant input text-fields. There are some components about membership. Also, there are another two components which are "change my password" and "change my profile picture" linked text-labels. She clicks to "Change My Password" linked text-label. Then, she goes to the page.
3. There are two components, which are input password-fields. They are encrypted to not display to anyone.
4. She types new password and re-password.
5. "******" for password, "******" for re-password.
6. Then, she clicks to "Save" button to finish the operation.

Scenario Name: Delete Member
Participating Actor Instances: Oğuzhan: System Administrator
Flow of Events:

1. Oğuzhan is a person who was defined as admin at beginning of the project. He can log in to the system with his authentication information (*i.e. e-mail address and password*) over login panel. He has a standard menu with "Admin Panel" linked text-label.
2. Oğuzhan goes to "Admin Panel" page on menu-navigation bar from above the header section. When the "Admin Panel" page is loaded, there are functions which are managed by Admin.
3. Then, he clicks to "Delete Member" linked text-label. When the "Delete Member" page is loaded, there is a list that contains some attributes that are ID of member, first name, last name, e-mail address, phone. Each row has a ratio button at the beginning of the line.
4. Oğuzhan selects an instance to delete from the system. "73" for ID of member, "Oğuzhan" for first name, "Ulusoy" for last name, "ulusoyoguzhan@gmail.com" for e-mail address, "05373119592" for phone.
5. After he selected an instance, he clicks to "Save" button to finish the operation.

Scenario Name: Register-for-Author
Participating Actor Instances: Didar: Unregistered person (aka visitor)
Flow of Events:

1. Didar is a person who is not defined as member (both author or normal user). Therefore, she does not have authentication information (*i.e. e-mail address and password*), so she cannot log in to the system over login panel. As we have mentioned earlier, author is a member who was approved by system administrator. Thus, author has authentication to write blog posts.
2. Didar clicks to "Register" linked text-label at main page. Then, she goes to the page.
3. When she has gone to the "Register" page, there is a form that runs with POST HTTP message. The form contains some components which are first name, last name, phone, e-mail address, password and profile picture.
4. Didar types her information to get authentication from the system.

5. She enters “Didar” for first name, “Turanlı” for last name, “05385207279” for phone, “didar.turanli@outlook.com” for e-mail address, “******” for password and “...Documents/avatar.jpeg” for profile picture.
6. Then, there is ratio button. It says “Are you going to be an author?”. The answers are “Yes” and “No”. Didar selects “Yes” option to become an author on the system.
7. Finally, she clicks to “Save” button to finish the operation.

Scenario Name: Register-for-Normal User
Participating Actor Instances: Didar: Unregistered person (aka visitor)
Flow of Events:

1. Didar is a person who is not defined as member (both author or normal user). Therefore, she does not have authentication information (*i.e. e-mail address and password*), so she cannot log in to the system over login panel. As we have mentioned earlier, author is a member who was approved by system administrator. Thus, author has authentication to write blog posts.
2. Didar clicks to “Register” linked text-label at main page. Then, she goes to the page.
3. When she has gone to the “Register” page, there is a form that runs with POST HTTP message. The form contains some components which are first name, last name, phone, e-mail address, password and profile picture.
4. Didar types her information to get authentication from the system.
5. She enters “Didar” for first name, “Turanlı” for last name, “05385207279” for phone, “didar.turanli@outlook.com” for e-mail address, “******” for password and “...Documents/avatar.jpeg” for profile picture.
6. Then, there is ratio button. It says “Are you going to be an author?”. The answers are “Yes” and “No”. Didar selects “No” option to become an author on the system.
   Finally, she clicks to “Save” button to finish the operation.

Scenario Name: Login
Participating Actor Instances: Didar: Normal User, Author
Flow of Events:

1. Didar is a person who is normal user. Therefore, she can be authenticated on the system. She can log in to the system with authentication information (*i.e. e-mail address and password*).
2. Didar clicks to “Login” linked text-label at main page. Then, she goes to the page.
   When she has gone to the “Login” page, there is a login form that runs with POST HTTP message. The form contains e-mail address and password components.
3. She types her authentication information. “didar.turanli@aol.com” for e-mail address and “******” for password.
4. Then, she clicks “Let me go inside” button. There is also clean button to refresh the form.

Scenario Name: Admin Login
Participating Actor Instances: Oğuzhan: System Administrator
Flow of Events:

1. Oğuzhan is a person who was defined as system administrator at the beginning of the project. Therefore, he can be authenticated on the system as admin. He can log in to the system with authentication information (*i.e. e-mail address and password*).
2. "Admin Login" page is different than membership login. The difference is about to reach it. Entrance of admin panel is not in header section or visible part of the web-site. Admin should go to "manage.domain.extend" and enter his authentication information. The artifact is like membership login panel. Oğuzhan types his authentication information. "ulusoyoguzhan@gmail.com" for e-mail address and "*******" for password. Then, he clicks "Let me go inside" button. There is also clean button to refresh the form.

Scenario Name: Logout
Participating Actor Instances: Didar: Author, Normal User, System Administrator
Flow of Events:

1. Didar is a member whose role is author. Therefore, she can be authenticated on the system through her role. She can log in to the system with authentication information (*i.e. e-mail address and password*).
2. She logs out from the system. She clicks to the "Logout" linked text-label in header section.
3. Finally, she can be logged out.

Scenario Name: Approve Author
Participating Actor Instances: Didar: Author, Oğuzhan: System Administrator
Flow of Events:

1. Didar is a member that wants to become an author, therefore she has to select "Yes" option for author question at registration phase. After she has typed variables on to the relevant fields and has selected option, clicks to "Save" button to finish the operation [*Scenario: Register-for-author*]. When she has clicked, back end of the system runs, gets information and a query begins to run.
2. While execution is completing, a notification (*or warning*) goes to relevant page in Admin Panel.
3. Oğuzhan is a person who was defined as system administrator at beginning of the project. He can log in to the system with his authentication information over admin panel [*Scenario: Admin Login*].
4. Then, he clicks to "Admin Panel" linked text-label on menu navigation bar.
5. When the page is loaded, there are administrative functions in here. He selects "Approve Author" linked text-label.

6. When the page is loaded, there is a list that contains people who waiting for approval. Each instance has some components which are first name, last name, phone, e-mail address. Also, each instance selection menu. The list contains “0” and “2” options.
7. 0 is for normal user membership. 2 is for approved membership that is called author.
8. Oğuzhan selects an instance. “Didar” for first name, “Turanlı” for last name, “didar.turanli@aol.com” for e-mail address, “053834567890” for phone number. Then, he observes her e-mail address, does it belong to a company or search on internet. If so, he selects “2” and clicks to “Save” button to finish the operation.

Use case scenarios

Use-Case Name: Add New Item
Participating Actor Instances: System Admin
Flow of Events:

1. The system administrator logs in to the system over login panel.
2. The method that is in back-end, checks the administrator. The method searches the person in Admin table in database, if it exists. If so, redirects to home page, as Admin.
3. Admin has a standard navigation menu with Admin Panel linked-text label. Admin goes to admin panel.
4. System opens the Admin Panel, there is going to be functions who are managed by Admin.
5. Admin clicks to the “Add New Item”.
6. System opens the “Add New Item” page, it has own form with attributes: name, protein, fat, carbohydrate, mineral, vitamin, reference time, (*briefly*) short description and photo.
7. Admin sees attributes of Add New Item page. Then, he types to some variable to the input-fields. After this entering operation, he clicks to the “Save” button to finish the operation.
8. System checks input fields. Controllers in the back-end runs with these variables. If there is no problem (i.e. null input point), the related method is called by controller. Then, the calling method runs query to add the variable to the related table in the database. The parse operation runs, and finishes. Statement is finished, and variables adds to the table in database.

Entry Conditions:
- ➢ Admin must be logged in to the system.

Exit Conditions:
- ➢ Admin can add new item successfully.

Exceptional Cases:
- ➢ If the person who try to access to system over admin panel, is not included on Admin table in database, the system can detect any invalid input.
- ➢ When admin adds new item to system, the system can detect any invalid input.

Use-Case Name: Delete Item

SEP Online

Participating Actor Instances: System Admin
Flow of Events:

1. The system administrator logs in to the system over login panel.
2. The method that is in back-end, checks the administrator. The method searches the person in Admin table in database, if it exists. If so, redirects to home page, as Admin.
3. Admin has a standard navigation menu with Admin Panel linked-text label. Admin goes to admin panel.
4. System opens the Admin Panel, there is going to be functions who are managed by Admin.
5. Admin clicks to the "Delete Item".
6. System opens the "Delete Item" page, it has own list with attributes: name, protein, fat, carbohydrate, mineral, vitamin, reference time, (*briefly*) short description and photo. Each instance (is called row) has a ratio button.
7. Admin selects and clicks to the ratio button, whatever he wants to remove. Then, he clicks to the "Save" button to finish the operation.
8. The relevant controller in back-end checks input (coming from ratio button that is selected), if there is no problem, it calls related method in back-end. The calling method begins to run a query for removing selected instance from related table in the database. It executes the statement and is finished the operation.

Entry Conditions:
- ➢ Admin must be logged in to the system.

Exit Conditions:
- ➢ Admin can delete item successfully.

Exceptional Cases:
- ➢ If the person who try to access to system over admin panel, is not included on Admin table in database, the system can detect any invalid input.
- ➢ When admin deletes item from system, the system can detect any invalid input. (*i.e. not clicked to ratio button.)*

Use-Case Name: <u>Delete Admin</u>
Participating Actor Instances: System Admin
Flow of Events:

1. The system administrator logs in to the system over login panel.
2. The method that is in back-end, checks the administrator. The method searches the person in Admin table in database, if it exists. If so, redirects to home page, as Admin.
3. Admin has a standard navigation menu with Admin Panel linked-text label. Admin goes to admin panel.
4. System opens the Admin Panel, there is going to be functions who are managed by Admin.
5. Admin clicks to the "Delete Admin".
6. System opens the "Delete Admin" page, it has own list with attributes: first name, last name, e-mail, phone. Each instance (is called row) has a ratio button.

7. Admin selects and clicks to the ratio button, whatever he wants to remove. Then, he clicks to the “Save” button to finish the operation.
8. The relevant controller in back-end checks input (coming from ratio button that is selected), if there is no problem, it calls related method in back-end. The calling method begins to run a query for removing selected instance from related table in the database. In executes the statement and is finished the operation.

Entry Conditions:

➢ Admin must be logged in to the system.

Exit Conditions:

➢ Admin can delete admin successfully.

Exceptional Cases:

➢ If the person who try to access to system over admin panel, is not included on Admin table in database, the system can detect any invalid input.

➢ When admin deletes admin from system, the system can detect any invalid input.

Use-Case Name: <u>Create Admin</u>
Participating Actor Instances: System Admin
Flow of Events:

1. The system administrator logs in to the system over login panel.
2. The method that is in back-end, checks the administrator. The method searches the person in Admin table in database, if it exists. If so, redirects to home page, as Admin.
3. Admin has a standard navigation menu with Admin Panel linked-text label. Admin goes to admin panel.
4. System opens the Admin Panel, there is going to be functions who are managed by Admin.
5. Admin clicks to “Create Admin” linked text-label.
6. System opens the “Create Admin” page, it has own form with attributes: first name, last name, e-mail, password, phone and profile picture.
7. Admin types some information input-field (*the password can be changed later*). Then, he clicks to the “Save” button to finish the operation.
8. The relevant controller in back-end checks input (*coming from admin registration form*), if there is no problem, it calls related method in back-end. The calling method begins to run a query for creating admin. Therefore, added information to input fields adds to the related table in database. If there is a problem, the error message is appeared.

Entry Conditions:

➢ Admin must be logged in to the system.

Exit Conditions:

➢ Admin can create admin instance successfully.

Exceptional Cases:

➢ If the person who try to access to system over admin panel, is not included on Admin table in database, the system can detect any invalid input.

SEP Online

- When admin adds new item to system, the system can detect any invalid input.

Use-Case Name: Change Profile Picture
Participating Actor Instances: System Admin, Author, Normal User
Flow of Events:

1. Anyone is a person who was defined a member (i.e. administrator, author or normal user), logs in to the system over login panel (*it is standard for everyone*).
2. The method that is in back-end, checks the member. The method searches the person in Admin or Member tables in database, if it exists. If so, redirects to home page through role in database. *(By the way, the navigation menu differs according to member role: i.e. admi*n, *author or normal user).*
3. The person who has logged on to the system, clicks to “My Profile” button in header section of the web-site. Then, goes to the “My Profile” page.
4. System opens the “My Profile” page, it has own input field as a select file bar.
5. The person who has logged on to the system, selects a new profile picture over input file field. Then, she clicks to the “Save” button to finish the operation.
6. The relevant controller in back-end checks the input (*coming from change profile picture page*), if there is no problem, it calls related method in back-end. The calling method begins to run a query for changing the profile picture with new photo file. Therefore, changed picture updates to the related table in database. If there is a problem, the error message is appeared.

Entry Conditions:
- Admin, author or normal user must be logged in to the system.

Exit Conditions:
- The person can change profile picture successfully.

Exceptional Cases:
- The person is not found on relevant tables in database.
- The database might be full capacity.
- When the person changes his profile picture , the system can detect any invalid input. (*i.e. not jpeg*)

Use-Case Name: Add Location
Participating Actor Instances: System Admin
Flow of Events:

1. The system administrator logs in to the system over login panel.
2. The method that is in back-end, checks the administrator. The method searches the person in Admin table in database, if it exists. If so, redirects to home page, as Admin.
3. Admin has a standard navigation menu with Admin Panel linked-text label. Admin goes to admin panel.
4. System opens the Admin Panel, there is going to be functions who are managed by Admin.
5. Admin clicks to “Add Location” linked text-label.

6. System opens the “Add Location” page, it has own form with single attribute that is name. It is just text-input-field.
7. Admin types a variable for input field of attribute.
8. The relevant controller in back-end checks the input (*coming from add location page*), if there is no problem, it calls related method in back-end. The calling method begins to run a query for adding location with entered attribute. Therefore, typed location adds to the related table (*Location*) in database. If there is a problem, the error message is appeared.

Entry Conditions:

- Admin must be logged in to the system.

Exit Conditions:

- Admin can add location successfully.

Exceptional Cases:

- If the person who try to access to system over admin panel, is not included on Admin table in database, the system can detect any invalid input.
- When admin adds location to system, the system can detect any invalid input.

Use-Case Name: Delete Location
Participating Actor Instances: System Admin
Flow of Events:

1. The system administrator logs in to the system over login panel.
2. The method that is in back-end, checks the administrator. The method searches the person in Admin table in database, if it exists. If so, redirects to home page, as Admin.
3. Admin has a standard navigation menu with Admin Panel linked-text label. Admin goes to admin panel.
4. System opens the Admin Panel, there is going to be functions who are managed by Admin.
5. Admin clicks to “Delete Location” linked text-label.
6. System opens the “Delete Location” page, there is a list with single attribute that is name. Each instance has a ratio button at beginning of the line.
7. Admin selects a location to remove, then clicks to “Save” button to finish the operation.
8. The relevant controller in back-end checks the input (*coming from delete location page*), if there is no problem, it calls related method in back-end. The calling method begins to run a query for deleting location with entered attribute. Therefore, selected location is deleted from the related table (*Location*) in database. If there is a problem, the error message is appeared.

Use-Case Name: Change Location
Participating Actor Instances: System Admin
Flow of Events:

1. The system administrator logs in to the system over login panel.

2. The method that is in back-end, checks the administrator. The method searches the person in Admin table in database, if it exists. If so, redirects to home page, as Admin.
3. Admin has a standard navigation menu with Admin Panel linked-text label. Admin goes to admin panel.
4. System opens the Admin Panel, there is going to be functions who are managed by Admin.
5. Admin clicks to "Change Location" page, there is a list with one attribute in input text-field. Admin changes the one. Then, he clicks to "Save" button to finish the operation.
6. The relevant controller in back-end checks the input (*coming from change location page*), if there is no problem, it calls related method in back-end. The calling method begins to run a query for deleting location with entered attribute. Therefore, selected location is updated on the related table (*Location*) in database. If there is a problem, the error message is appeared.

Entry Conditions:

- ➢ Admin must be logged in to the system.

Exit Conditions:

- ➢ Admin can change location successfully.

Exceptional Cases:

- ➢ If the person who try to access to system over admin panel, is not included on Admin table in database, the system can detect any invalid input.
- ➢ When admin changes location on the system, the system can detect any invalid input.

Use-Case Name: <u>Add Local Market</u>
Participating Actor Instances: System Admin
Flow of Events:

1. The system administrator logs in to the system over login panel.
2. The method that is in back-end, checks the administrator. The method searches the person in Admin table in database, if it exists. If so, redirects to home page, as Admin.
3. Admin has a standard navigation menu with Admin Panel linked-text label. Admin goes to admin panel.
4. System opens the Admin Panel, there is going to be functions who are managed by Admin.
5. Admin clicks to "Add Local Market" page, it has own input form with some attributes in input text-fields. These are orderly name, phone, mail, address, picture, description and location (*actually it is location identification number*). Admin adds some variables through the attribute's names. Then, he clicks to "Save" button to finish the operation.
6. The relevant controller in back-end checks the input (*coming from add local market page*), if there is no problem, it calls related method in back-end. The calling method begins to run a query for deleting location with entered attribute. Therefore, typed

information adds to the related table (*Local Market*) in database. If there is a problem, the error message is appeared.

Entry Conditions:

- Admin must be logged in to the system.

Exit Conditions:

- Admin can add new local market successfully.

Exceptional Cases:

- If the person who try to access to system over admin panel, is not included on Admin table in database, the system can detect any invalid input.
- When admin adds new local market to system, the system can detect any invalid input.

Use-Case Name: Delete Local Market
Participating Actor Instances: System Admin
Flow of Events:

1. The system administrator logs in to the system over login panel.
2. The method that is in back-end, checks the administrator. The method searches the person in Admin table in database, if it exists. If so, redirects to home page, as Admin.
3. Admin has a standard navigation menu with Admin Panel linked-text label. Admin goes to admin panel.
4. System opens the Admin Panel, there is going to be functions who are managed by Admin.
5. Admin clicks to "Delete Local Market" page, there is a list that contains local markets with some attributes in input text-fields. These are orderly name, phone, mail, address. Each instance has ratio button at the beginning of the instance. Admin selects a ratio button through what should be deleted. Then, he clicks to "Save" button to finish the operation.
6. The relevant controller in back-end checks the input (*coming from delete local market page*), if there is no problem, it calls related method in back-end. The calling method begins to run a query for deleting location with entered attribute. Therefore, selected ratio button's ID number (*unique identification number*) is deleted from the related table (*Local Market*) in database. If there is a problem, the error message is appeared.

Entry Conditions:

- Admin must be logged in to the system.

Exit Conditions:

- Admin can delete local market successfully.

Exceptional Cases:

- If the person who try to access to system over admin panel, is not included on Admin table in database, the system can detect any invalid input.

Use-Case Name: Change Local Market
Participating Actor Instances: System Admin
Flow of Events:

1. The system administrator logs in to the system over login panel.
2. The method that is in back-end, checks the administrator. The method searches the person in Admin table in database, if it exists. If so, redirects to home page, as Admin.
3. Admin has a standard navigation menu with Admin Panel linked-text label. Admin goes to admin panel.
4. System opens the Admin Panel, there is going to be functions who are managed by Admin.
5. Admin clicks to "Change Local Market" page, there is a list includes some attributes, which are orderly name, phone, mail, address, description, in input text-field. Admin changes the one. Then, he clicks to "Save" button to finish the operation.
6. The relevant controller in back-end checks the input (*coming from change local market page*), if there is no problem, it calls related method in back-end. The calling method begins to run a query for deleting location with entered attribute. Therefore, updated input-field's ID number (*unique identification number*) is changed on the related table (*Local Market*) in database. If there is a problem, the error message is appeared.

Entry Conditions:
- ➢ Admin must be logged in to the system.

Exit Conditions:
- ➢ Admin can change local market successfully.

Exceptional Cases:
- ➢ If the person who try to access to system over admin panel, is not included on Admin table in database, the system can detect any invalid input.
- ➢ When admin changes local market on the system, the system can detect any invalid input.

Use-Case Name: Suggest-an-Item
Participating Actor Instances: Author, Normal User, System Admin
Flow of Events:

1. Any member (*both author and normal user*) logs in to the system with her authentication information (*i.e. e-mail address and password*) over login panel.
2. The method that is in back-end, checks the member. The method searches the person in Member table in database, if it exists. If so, redirects to home page through role in database. It changes menu-navigation bar.
3. She has a standard menu with "Write Blog" linked text-label. Then, clicks to "Suggest Item" linked text-label on menu-navigation bar.

4. When "Suggest Item" page is loaded, there is an input form with some attributes, which are name, protein, fat, carbohydrate, mineral, vitamin, reference time, (*briefly*) short description and photo.
5. She adds some variables for each input text-field. Then, clicks to "Save" button to finish the operation. Yet, the item cannot be added to list directly. It must be approved by system administrator.
6. The relevant controller in back-end checks the input (*coming from suggest-an-item page*), if there is no problem, it calls related method in back-end. The calling method begins to run a query for deleting location with entered attribute. Therefore, the typed information adds with ID (*with unique identification number*) to the related table (*Suggestion*) in database. If there is a problem, the error message is appeared. When the item adds to the database, it has default role number, that is zero. Admin can display the "list of waiting for approval" on own Admin Panel. Each instance has two buttons. These are "accept" and "reject". When Admin clicks to "accept" button, role of the selected item is changed by another function and adds to Item table in database over another query. When the execution ended, the suggestion (aka new item) is shown on Items tab. Otherwise (when Admin clicks to "reject" button, role of the selected item is changed by another function and is deleted from Suggestion table in database over another query.

Entry Conditions:
- ➢ Admin, author, normal user must be logged in to the system.

Exit Conditions:
- ➢ The person can suggest new item successfully.

Exceptional Cases:
- ➢ If the person who try to access to system, is not included on relevant tables in database, the system can detect any invalid input.
- ➢ When the person suggests new item to system, the system can detect any invalid input.

Use-Case Name: <u>Add Comment</u>
Participating Actor Instances: Normal User, Author, System Administrator
Flow of Events:

1. Any member (*both author and normal user*) logs in to the system with her authentication information (*i.e. e-mail address and password*) over login panel.
2. The method that is in back-end, checks the member. The method searches the person in Member table in database, if it exists. If so, redirects to home page through role in database. It changes menu-navigation bar.
3. The member who is anybody whose role is not important, clicks to "Blog" tab on menu bar.
4. System opens the request.
5. The member who is anybody whose role is not important clicks to any blog title.
6. System opens the request. The title, or details of the post, is created through unique identification number of blog. It is a web-based dynamic rule. The page that contains details of blog, is created that ID. The relevant method runs with this ID number and

prepares a query. The condition is ID number. The query that was prepared previous, executes and returns a result as dictionary. The view pattern of web-site was prepared block structure. Then, it satisfies the related information on relevant fields.

7. The member who is anybody whose role is not important, sees comment form (input form, runs with get method) above the blog. There are two components and "Save" button to finish the operation. She types her comment and clicks to button.
8. The relevant controller in back-end checks the input (*coming from comment block*), if there is no problem, it calls related method in back-end. The calling method begins to run a query for adding a comment with entered attribute (*that is comment*), commenter name (*comes from session control*) and blog-post ID (*comes from session control*). Therefore, the typed comment adds with ID (*with unique identification number*) to the related table (*Comment*) in database. If there is a problem, the error message is appeared.

Entry Conditions:
➢ Admin, author, normal user must be logged in to the system.
Exit Conditions:
➢ The person can add comment successfully.
Exceptional Cases:
➢ If the person who try to access to system, is not included on relevant tables in database, the system can detect any invalid input.
➢ When admin adds new item to system, the system can detect any invalid input.

Use-Case Name: Delete Comment
Participating Actor Instances: System Administrator
Flow of Events:

1. The system administrator logs in to the system over login panel.
2. The method that is in back-end, checks the administrator. The method searches the person in Admin table in database, if it exists. If so, redirects to home page, as Admin.
3. Admin has a standard navigation menu with Admin Panel linked-text label. Admin goes to admin panel.
4. System opens the Admin Panel, there is going to be functions who are managed by Admin.
5. Then, he clicks to "Delete Comment" linked text-label. When the "Delete Comment" page is loaded, there is a list that contains some attributes that are ID of comment, commenter name and comment text. Each row has a ratio button at the beginning of the line. Each ratio button runs with ID of comment. Then, he selects an instance to delete, and clicks to "Save" button.
6. The relevant controller in back-end checks the input (*coming from delete comment page*), if there is no problem, it calls related method in back-end. The calling method begins to run a query for deleting the selected comment with coming attribute (*that is ratio button has unique identification number*. Therefore, the selected comment is

deleted from related table (*Comment*) in database. If there is a problem, the error message is appeared.

Entry Conditions:
- ➢ Admin must be logged in to the system.

Exit Conditions:
- ➢ Admin can delete comment successfully.

Exceptional Cases:
- ➢ If the person who try to access to system over admin panel, is not included on Admin table in database, the system can detect any invalid input.
- ➢ When admin deletes comment from the system, the system can detect any invalid input.

Use-Case Name: Add Blog
Participating Actor Instances: Author
Flow of Events:

1. Author is a person who was approved by system administrator at after registration. She logs on to the system with her authentication information (*i.e. e-mail address and password*) over login panel. She has a standard menu with "Write Blog" linked text-label.
2. The method that is in back-end, checks the administrator. The method searches the person in Member table in database, if it exists. If so, redirects to home page, as Admin.
3. Author has a standard navigation menu with Write Blog linked-text label. Author goes to "Write Blog" page.
4. System opens "Write Blog" page, there are two components. First is title, it is input-field and second is text, it is input text-area. Author types some variables to relevant field, then clicks to "Save" button to finish the operation.
5. The relevant controller in back-end checks the input (*coming from add blog page*), if there is no problem, it calls related method in back-end. The calling method begins to run a query for adding blog with coming Therefore, the added information is added on the related table (*Blog*) in database. If there is a problem, the error message is appeared. By the way, same query needs three another requirement. These are ID of blog, ID of member and publish date (*it automatically renders over database.)*

Note: Add Blog is name of function in back-end, Write blog is name of the function's in the menu navigation bar.

Entry Conditions:
- ➢ Author must be logged in to the system.

Exit Conditions:
- ➢ Admin can add new blog successfully.

Exceptional Cases:

- If the person who try to access to system, is not included on Member table in database, the system can detect any invalid input.
- When author adds new blog to system, the system can detect any invalid input.

Use-Case Name: <u>Delete Blog</u>
Participating Actor Instances: System Administrator
Flow of Events:

1. The system administrator logs in to the system over login panel.
2. The method that is in back-end, checks the administrator. The method searches the person in Admin table in database, if it exists. If so, redirects to home page, as Admin.
3. Admin has a standard navigation menu with Admin Panel linked-text label. Admin goes to admin panel.
4. System opens the Admin Panel, there is going to be functions who are managed by Admin.
5. Admin clicks to "Delete Blog" page, there is a list that contains ID of blog, title of blog. Each instance has a ratio button at the beginning of the row. Then, he selects an instance, whatever has to be deleted.
6. The relevant controller in back-end checks the input (*coming from delete blog page*), if there is no problem, it calls related method in back-end. The calling method begins to run a query for deleting the selected blog with coming attribute (*that is ratio button has unique identification number)*. Therefore, the selected blog is deleted from related table (*Blog*) in database. If there is a problem, the error message is appeared.

Entry Conditions:
- Admin must be logged in to the system.

Exit Conditions:
- Admin can deletes blog successfully.

Exceptional Cases:
- If the person who try to access to system over admin panel, is not included on Admin table in database, the system can detect any invalid input.

Use-Case Name: <u>Change Member Details</u>
Participating Actor Instances: Normal User, Author
Flow of Events:

1. Normal is a member who was defined through role in the database. She can log in to the system with her authentication information (*i.e. e-mail address and password*) over login panel.
2. The method that is in back-end, checks the member. The method searches the person in Member table in database, if it exists. If so, redirects to home page, as normal user.

3. When Dilara is redirected to main page, she clicks to “My Profile” linked text-label.
4. System opens “My Profile” page through member’s unique identification number.
5. There are some components which relevant input text-fields. These are first name, last name, phone, e-mail address. Also, there are another two components which are “change my password” and “change my profile picture” linked text-labels. They are other scenarios and use-case scenarios, they will be mentioned in other scenarios. In addition to these, she can display her unique identification number which is generated by the back-end system at registration phase. She can see her membership information. Her e-mail address should be changed. She types new e-mail address information to relevant field. New e-mail address, instead of old e-mail address. Then, she clicks to “Save” button to finish the operation.
6. The relevant controller in back-end checks the input (*coming from my profile page*), if there is no problem, it calls related method in back-end. The calling method begins to run a query for changing profile with coming attributes. Therefore, the changed information is added on the related table (*Member*) in database. If there is a problem, the error message is appeared.

Entry Conditions:
- Normal user, author must be logged in to the system.

Exit Conditions:
- The person can change profile details successfully.

Exceptional Cases:
- If the person who try to access to system, is not included on Member table in database, the system can detect any invalid input.
- When the person changes profile details, the system can detect any invalid input.

Scenario Name: Change Member Password
Participating Actor Instances: Normal User, Author
Flow of Events:

1. Normal user is a member. She can log in to the system with her authentication information (*i.e. e-mail address and password*) over login panel.
2. The method that is in back-end, checks the member. The method searches the person in Member table in database, if it exists. If so, redirects to home page, as normal user.
3. When the member is redirected to main page, she clicks to “My Profile” linked text-label.
4. System opens “My Profile” page.
5. When “My Profile” page is loaded, there are some components which relevant input text-fields. There are some components about membership. Also, there are another two components which are “change my password” and “change my profile picture” linked text-labels. She clicks to “Change My Password” linked text-label.
6. System opens “Change My Password” page.
7. When “Change My Password” page is loaded, there are two components, which are input password-fields. They are encrypted to not display to anyone. She types new password and re-password. Then, she clicks to “Save” button to finish the operation.

8. The relevant controller in back-end checks the input (*coming from change my password page*), if there is no problem, it calls related method in back-end. The calling method begins to run a query for updating password with coming attributes. Therefore, the changed information is added on the related table (*Member*) in database. If there is a problem, the error message is appeared.

Entry Conditions:
- ➢ Author, normal user must be logged in to the system.

Exit Conditions:
- ➢ The person can change password successfully.

Exceptional Cases:
- ➢ If the person who try to access to system, is not included on Member table in database, the system can detect any invalid input.
- ➢ When the person changes password, the system can detect any invalid input.

Use-Case Name: <u>Delete Member</u>
Participating Actor Instances: System Admin
Flow of Events:

1. The system administrator logs in to the system over login panel.
2. The method that is in back-end, checks the administrator. The method searches the person in Admin table in database, if it exists. If so, redirects to home page, as Admin.
3. Admin has a standard navigation menu with Admin Panel linked-text label. Admin goes to admin panel.
4. System opens the Admin Panel, there is going to be functions who are managed by Admin.
5. Admin clicks to the “Delete Member”.
6. System opens the “Delete Member” page, it has own list with attributes: ID of member, first name, last name, e-mail address, phone. Each instance (*is called row*) has a ratio button.
7. Admin selects and clicks to the ratio button, whatever he wants to remove. Then, he clicks to the “Save” button to finish the operation.
8. The relevant controller in back-end checks input (*coming from ratio button that is selected*), if there is no problem, it calls related method in back-end. The calling method begins to run a query for removing selected instance from related table in the database. It executes the statement and is finished the operation.

Entry Conditions:
- ➢ Admin must be logged in to the system.

Exit Conditions:
- ➢ Admin can delete member successfully.

Exceptional Cases:
- ➢ If the person who try to access to system over admin panel, is not included on Admin table in database, the system can detect any invalid input.

Use-Case Name: Register-for-Author
Participating Actor Instances: Unregistered person (*aka visitor*), System Administrator
Flow of Events:

1. Unregistered (*that is called visitor*) is a person who is not defined as member (*both author or normal user*). Therefore, she does not have authentication information (*i.e. e-mail address and password*), so she cannot log in to the system over login panel. As we have mentioned earlier, author is a member who was approved by system administrator. Thus, author has authentication to write blog posts. Visitor clicks to "Register" linked text-label at main page.
2. System opens "Register" page.
3. When she has gone to the "Register" page, there is a form that runs with GET HTTP message. The form contains some components which are first name, last name, phone, e-mail address, password and profile picture. She types her information to get authentication from the system. In addition to these, there is ratio button that operates the request to become author. Then, she selects "Yes" option. Finally, she clicks to "Save" button to finish the operation.
4. Ratio button has text-label about what it does. The answers are "Yes" and "No". "Yes" option sends an input to relevant method as 1. Otherwise, it is going to be 0 as input. The relevant controller in back-end checks input (*coming from register page*), if there is no problem, it calls related method in back-end. The calling method begins to run a query for making registration instance on to the related table in the database. It executes the statement and is finished the operation. The person has to wait to take approval from system. Otherwise, it is just a member that is normal user.

Entry Conditions:
➢ There must be no logged account.
Exit Conditions:
➢ The person can register successfully.
Exceptional Cases:
➢ When the person adds information to register, the system can detect any invalid input.
Use-Case Name: Register-for-Normal User
Participating Actor Instances: Unregistered person (*aka visitor*), System Administrator
Flow of Events:

1. Unregistered (*that is called visitor*) is a person who is not defined as member (*both author or normal user*). Therefore, she does not have authentication information (*i.e. e-mail address and password*), so she cannot log in to the system over login panel. As we have mentioned earlier, author is a member who was approved by system administrator. Thus, author has authentication to write blog posts. Visitor clicks to "Register" linked text-label at main page.
2. System opens "Register" page.
3. When she has gone to the "Register" page, there is a form that runs with GET HTTP message. The form contains some components which are first name, last name,

phone, e-mail address, password and profile picture. She types her information to get authentication from the system. In addition to these, there is ratio button that operates the request to become author. Then, she selects “No” option. Finally, she clicks to “Save” button to finish the operation.

4. Ratio button has text-label about what it does. The answers are “Yes” and “No”. “No" option sends an input to relevant method as 0. If she has selected, it would have been 1 as input. The relevant controller in back-end checks input (*coming from register page*), if there is no problem, it calls related method in back-end. The calling method begins to run a query for making registration instance on to the related table in the database. It executes the statement and is finished the operation.

Entry Conditions:
- There must be no logged account.

Exit Conditions:
- The person can register successfully.

Exceptional Cases:
- When the person adds information to register, the system can detect any invalid input.

Use-Case Name: <u>Login</u>
Participating Actor Instances: Normal User, Author
Flow of Events:

1. People can be authenticated on the system. She can log in to the system with authentication information (*i.e. e-mail address and password*). Any member (*both normal user and author*) clicks to “Login” linked text-label at main page. Then, she goes to the page.
2. System opens “Login” page.
3. When she has gone to the “Login” page, there is a login form that runs with POST HTTP message. The form contains e-mail address and password components. She types her authentication information. Then, she clicks “Let me go inside” button. There is also clean button to refresh the form.
4. When the button is clicked, the relevant controller in back-end checks input (*coming from login page*), if there is no problem, it calls related method in back-end. The calling method begins to run a query for searching the person on the related table (*Member*) in the database, if it exits. If so, she is redirected on to main page as logged on the system. Otherwise, the error message is shown as warning. In addition to these scenario, “Clean” button refresh the form.

Entry Conditions:
- There must be no logged account.

Exit Conditions:
- The person can login successfully.

Exceptional Cases:
- When the person try to login, the system can detect any invalid input.

Use-Case Name: Admin Login
Participating Actor Instances: System Administrator
Flow of Events:

1. Admin should go to "manage.domain.extend" over web-browser and enter his authentication information. The artifact is like membership login panel. Admin types his authentication information. Then, she clicks "Let me go inside" button. There is also clean button to refresh the form.
2. When the button is clicked, the relevant controller in back-end checks input (*coming from admin login page*), if there is no problem, it calls related method in back-end. The calling method begins to run a query for searching the person on the related table (*Admin*) in the database, if it exits. If so, she is redirected on to main page as logged on the system. Otherwise, the error message is shown as warning. In addition to these scenario, "Clean" button refresh the form.

Entry Conditions:
➢ There must be no logged account.
Exit Conditions:
➢ The person can register successfully.
Exceptional Cases:
➢ When the person try to login, the system can detect any invalid input.

Use-Case Name: Logout
Participating Actor Instances: Author, Normal User, System Administrator
Flow of Events:

1. All people can log in to the system. Therefore, she can be authenticated on the system as admin. She can log in to the system with authentication information (*i.e. e-mail address and password*). She logs out from the system. She clicks to the "Logout" linked text-label in header section.
2. Finally, she can be logged out.

Use-Case Name: Approve Author
Participating Actor Instances: Author, System Administrator
Flow of Events:

1. Anyone is a member that wants to become an author, therefore she has to select "Yes" option for author question at registration phase. After she has typed variables on to the relevant fields and has selected option, clicks to "Save" button to finish the operation [*Scenario: Register-for-author*]. When she has clicked, back end of the system runs, gets information and a query begins to run [*Use-Case: Register-for-author*].
2. While execution is completing, a notification (*or warning*) goes to relevant page in Admin Panel.

3. System admin goes to own panel with his authentication information over admin panel [*Scenario: Admin Login*]. Then, he clicks to "Admin Panel" linked text-label on menu navigation bar.
4. System opens "Admin Panel" page.
5. When the page is loaded, there are administrative functions in here. He selects "Approve Author" linked text-label.
6. System opens "Approve Author" page.
7. When the page is loaded, there is a list that contains people who waiting for approval. Each instance has some components which are first name, last name, phone, e-mail address. Also, each instance selection menu. The list contains "0" and "2" options.
8. 0 is for normal user membership. 2 is for approved membership that is called author. Admin selects and instance, then clicks to "Save" button to finish the operation.
9. The relevant controller in back-end checks input (*coming from approve-author page*), if there is no problem, it calls related method in back-end. The calling method begins to run a query for approving instance on to the related table in the database. Actually, it just updates its role value from 1 to 2. It executes the statement and is finished the operation.

Entry Conditions:
- ➢ Admin must be logged in to the system.

Exit Conditions:
- ➢ Admin can approve author successfully.

Use-Case Name: Display Local Market
Participating Actor Instances: Unregistered, Author, Normal User, System Administrator
Flow of Events:

1. If the person who wants to display details of local market, is member, she can log in to the system with her authentication information [*Use-Case Scenario: Login*]. After she logged, goes to "Local Market" page over menu navigation bar.
2. When "Local Market" page is loaded, local markets are going to be shown with photo and description. Also, URLs run with its own unique identification number.
3. Person selects an instance to go to the inside.
4. "Local Market Details" page is prepared with unique identification number. Session structure is used in back end.
5. After the content is prepared, there are going to be all attributes of local market instance. These are name, address, phone, e-mail address, description and photo.

Entry Conditions:
- ➢ No entry condition.

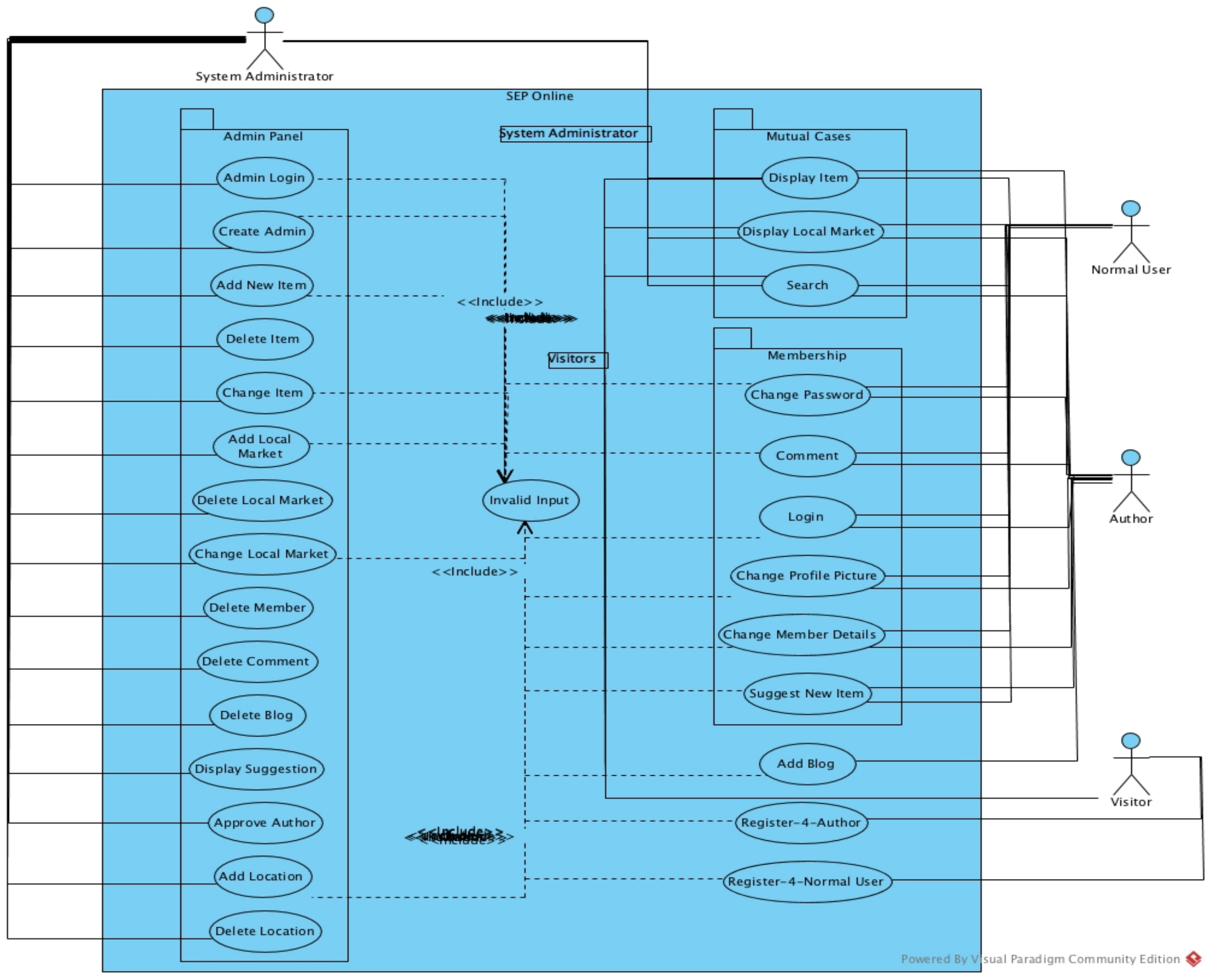
Exit Conditions:
- ➢ She can display local market successfully.

Exceptional Cases:
- ➢ If the person who wants to display details of local market is member on the system, the system can detect any invalid input.

SEP Online

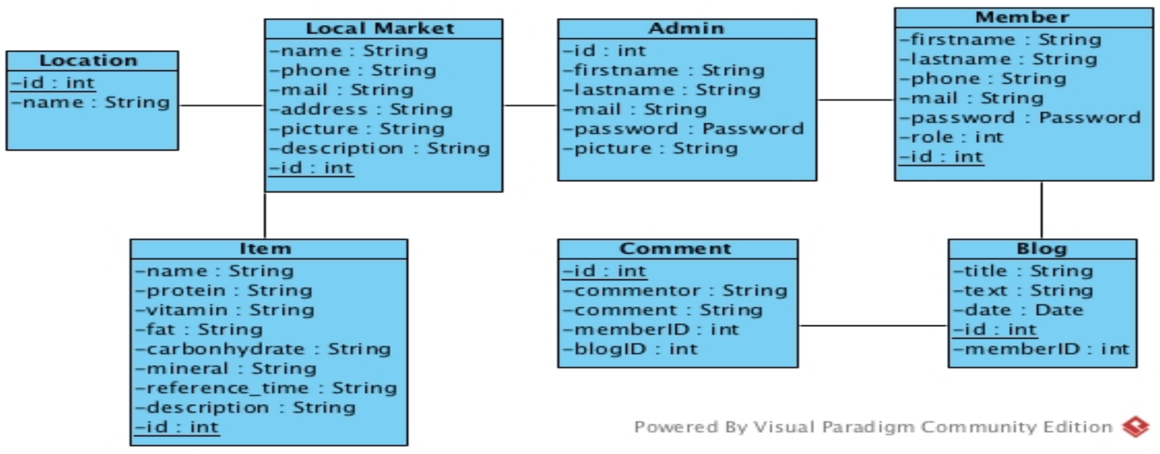
Use case model

A use case is a generalization of a number of scenarios. The use case model as follows, and it is added to project folder.



Object model

We are going to be implement the project by using PHP. It is not objective oriented programming language, so we cannot generate objects by input. However, we have prepared UML diagram to represent the association tables.

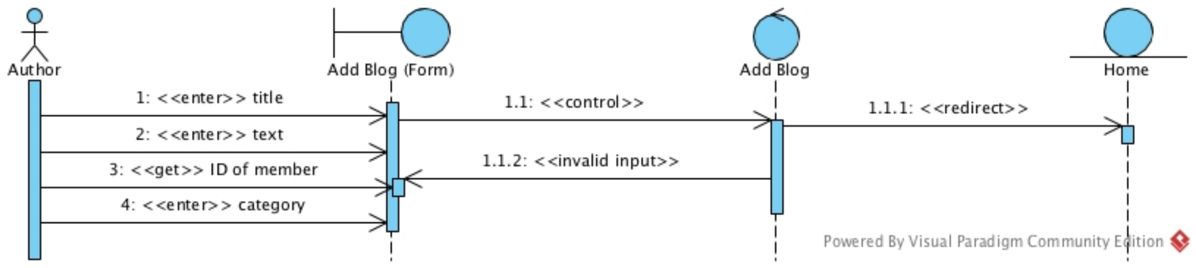


SEP Online

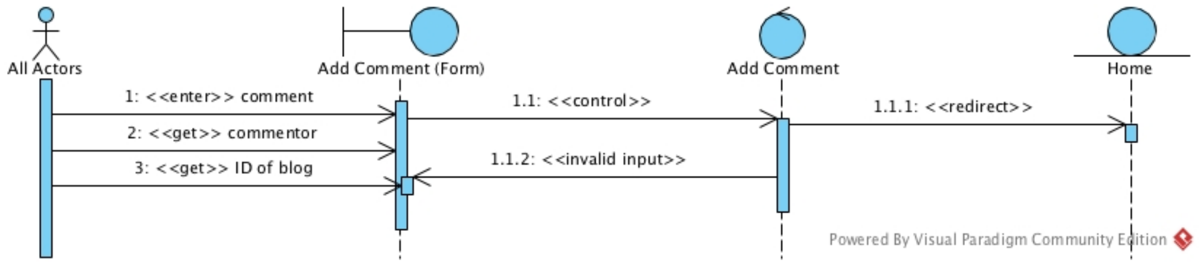
Dynamic model

We assume that actors have logged in to the system in some cases.

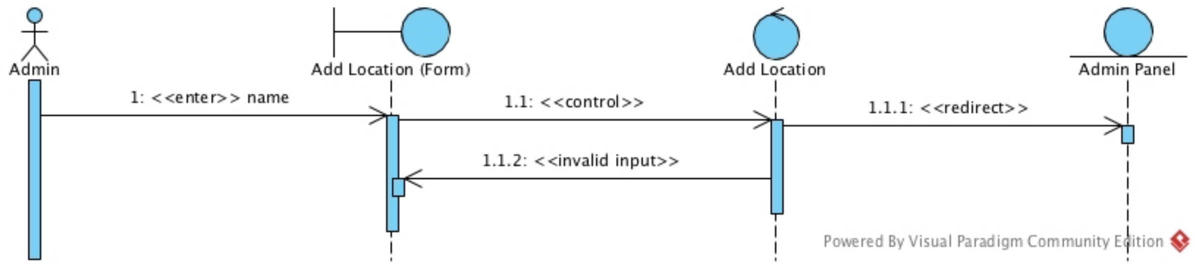
➢ Add Blog:



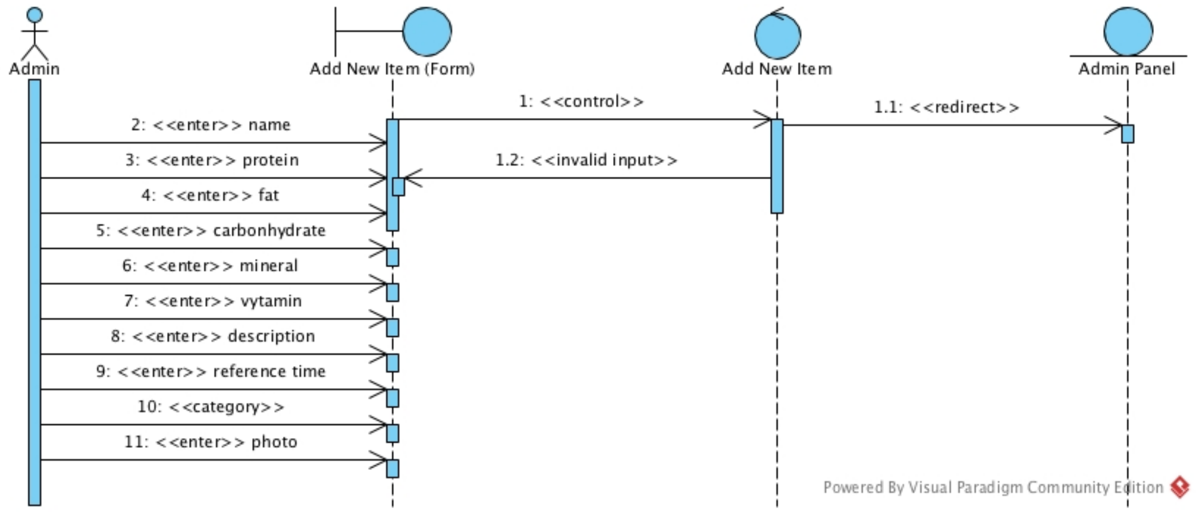
➢ Add Comment:



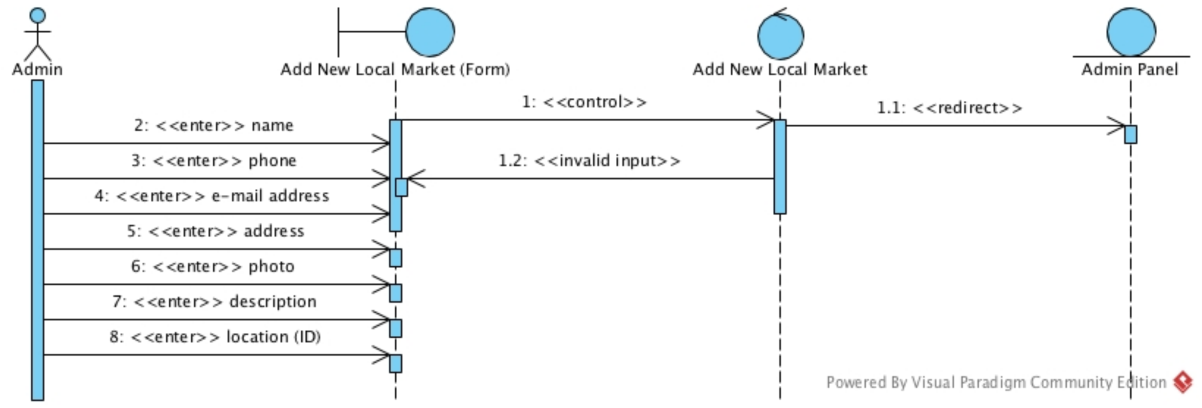
➢ Add Location:



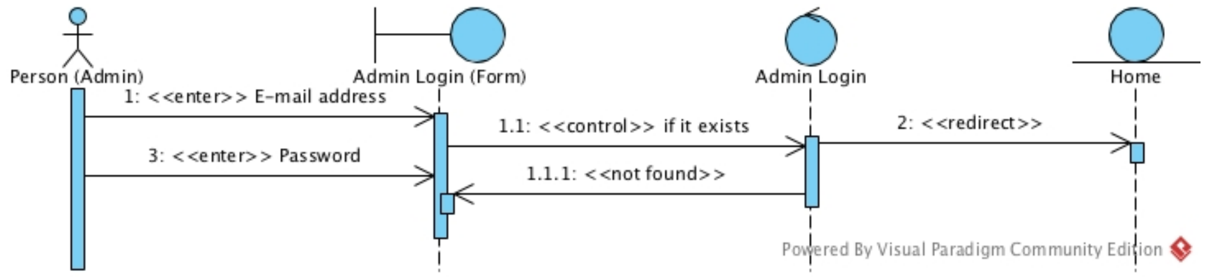
➢ Add New Item:



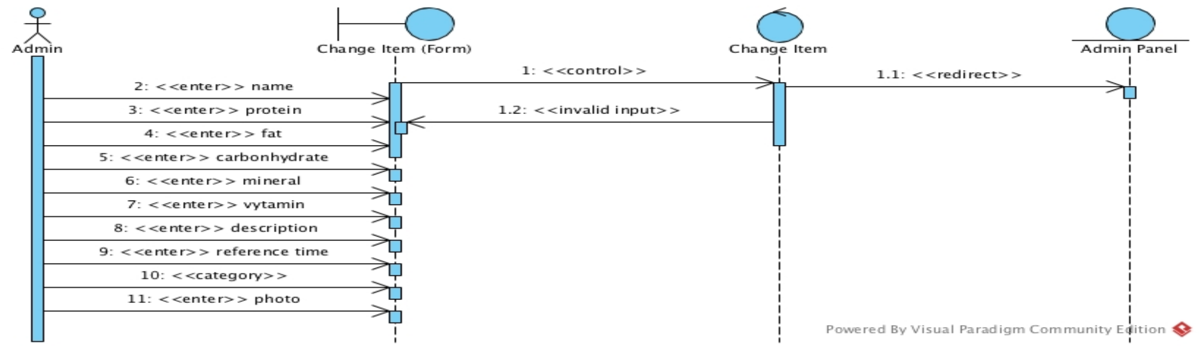
- Add New Local Market:



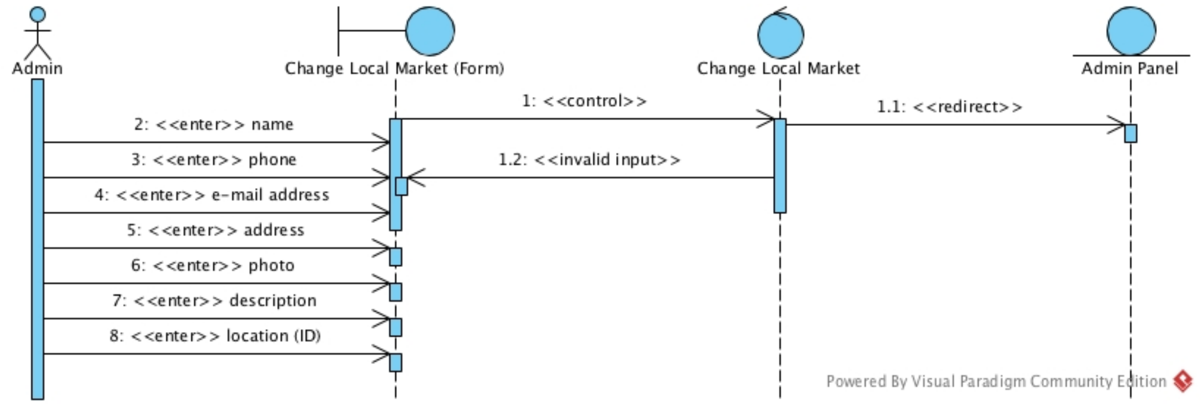
- Admin Login:



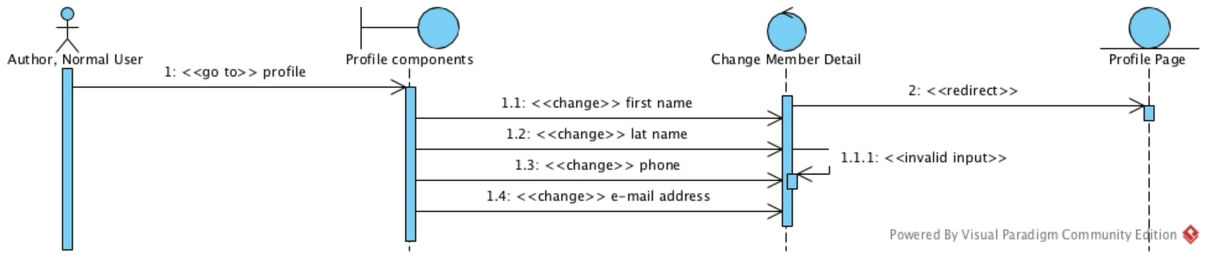
- Change Item:



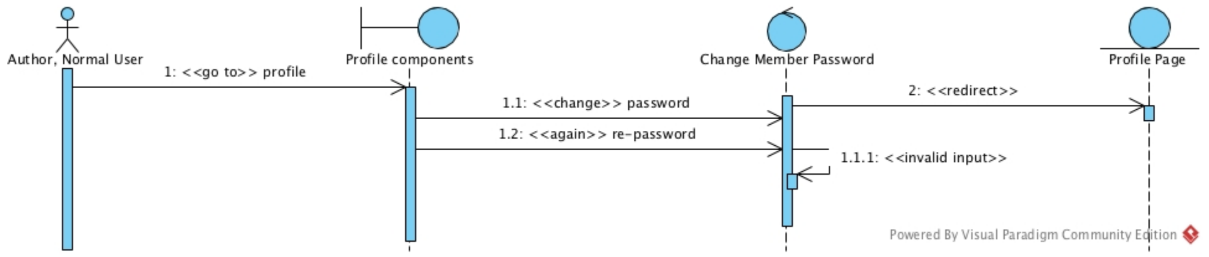
- Change Local Market:



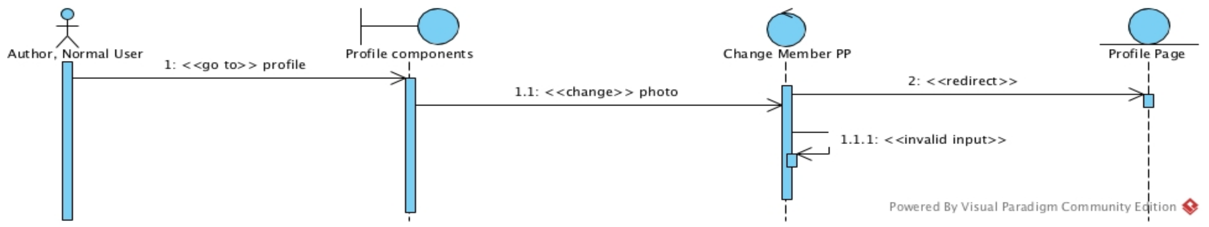
- Change Member Detail:



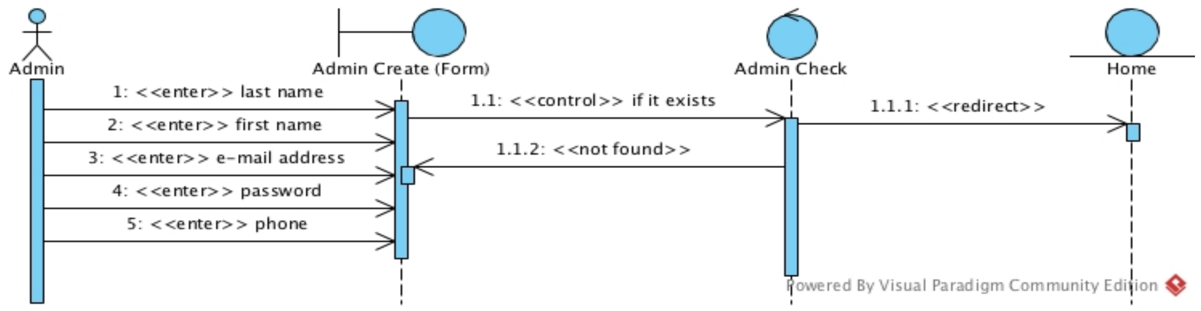
- Change Member Password:



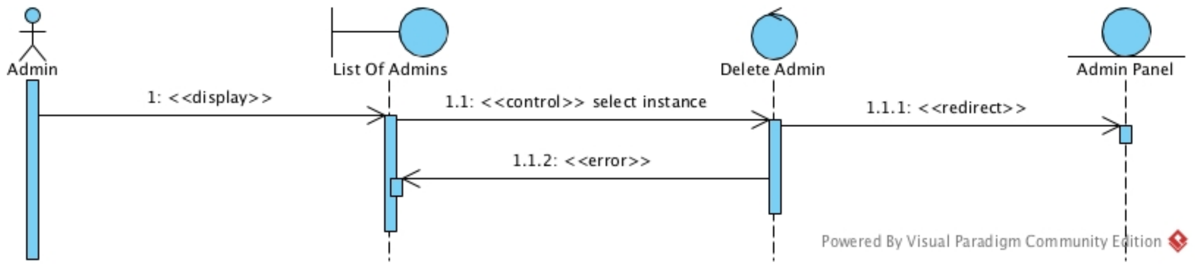
- Change Member Profile Picture:



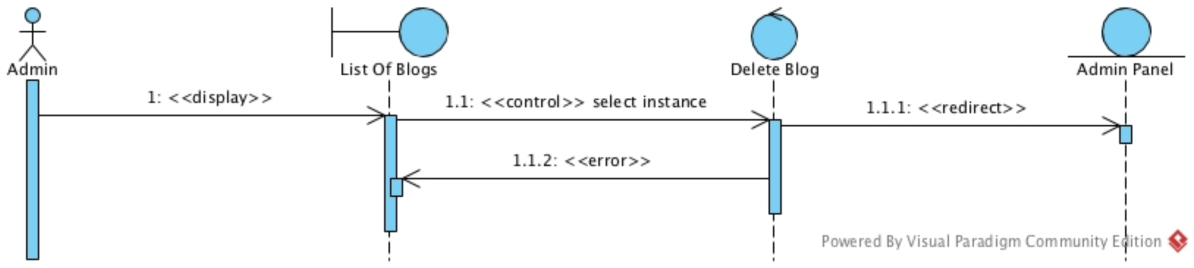
- Create Admin:



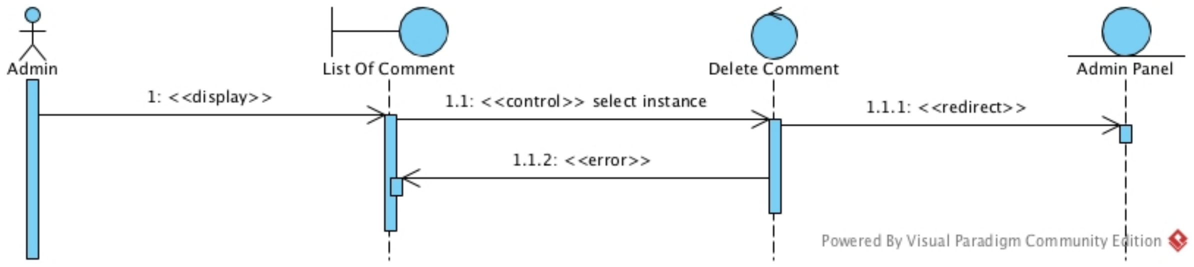
- Delete Admin:



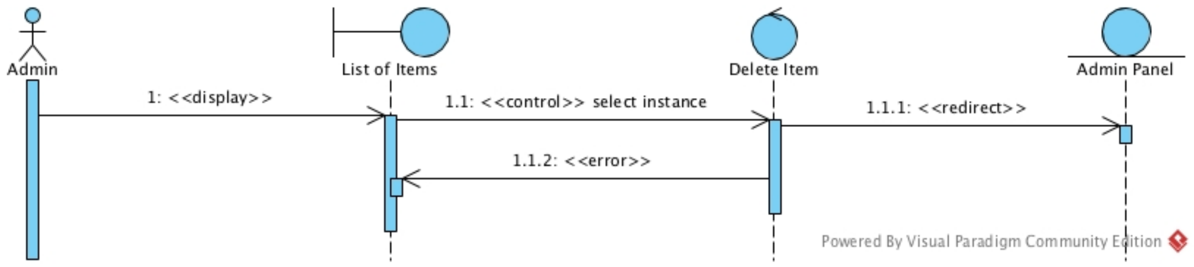
- Delete Blog:



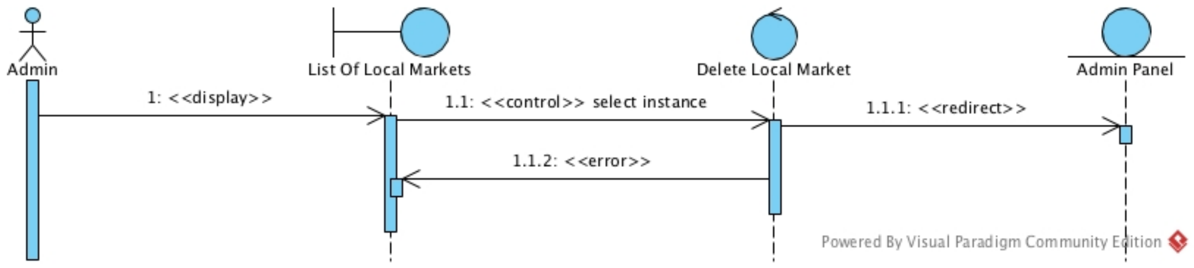
- Delete Comment:



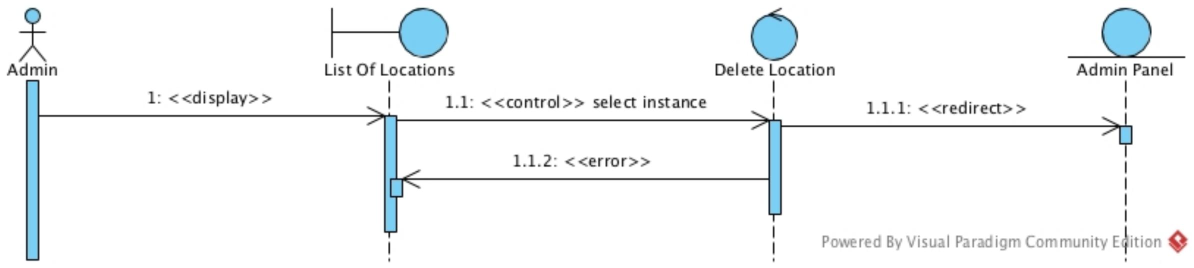
- Delete Item:



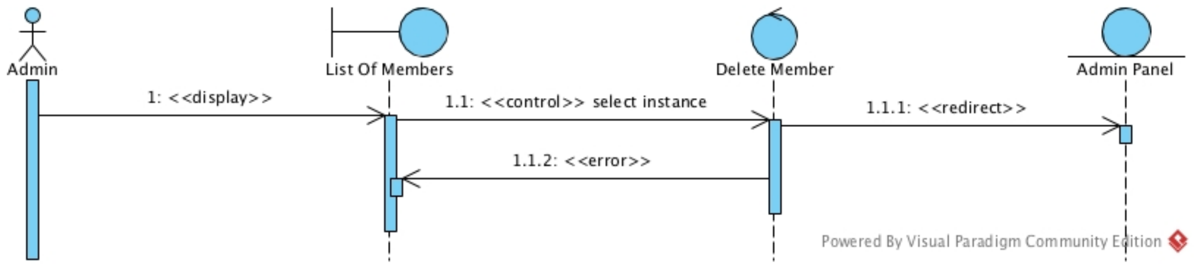
- Delete Local Market:



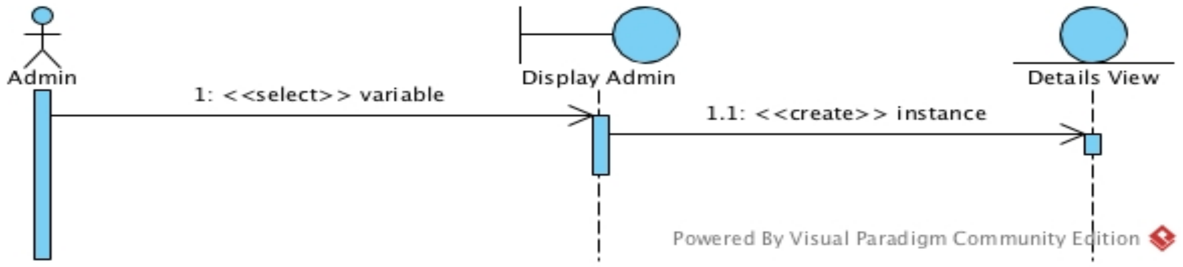
- Delete Location:



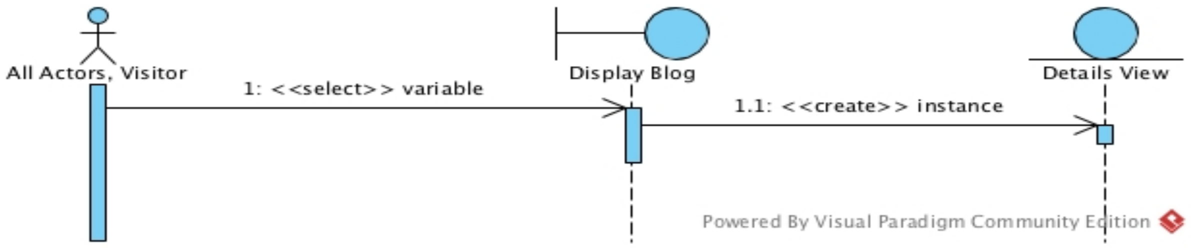
- Delete Member:



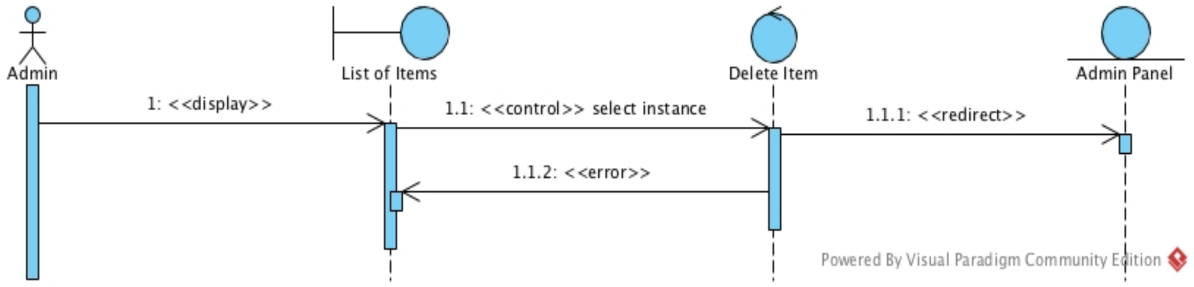
- Display Admin:



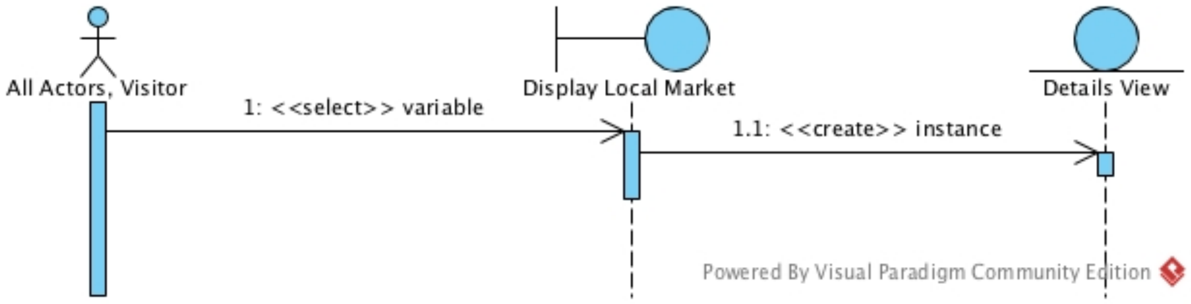
- Display Blog:



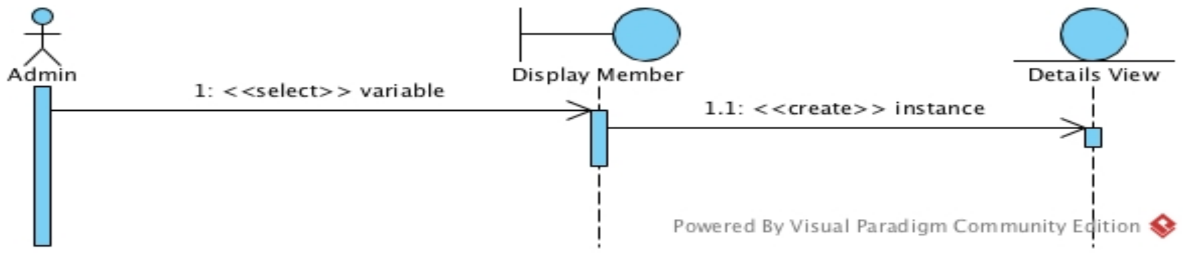
- Display Item:



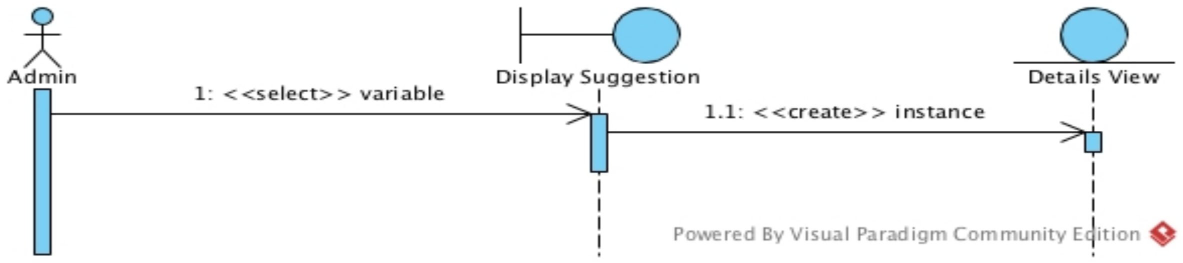
- Display Local Market:



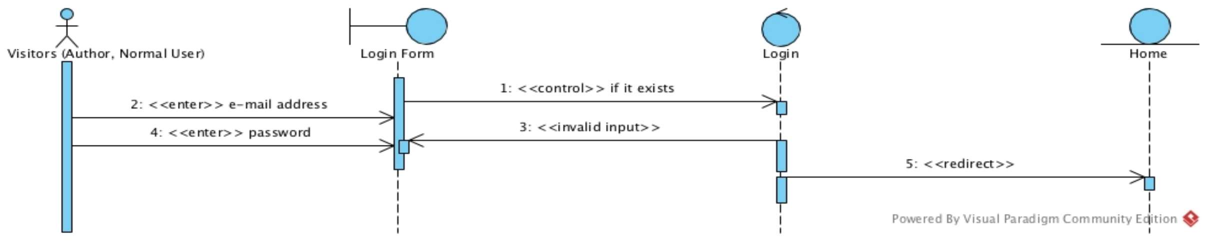
- Display Member:



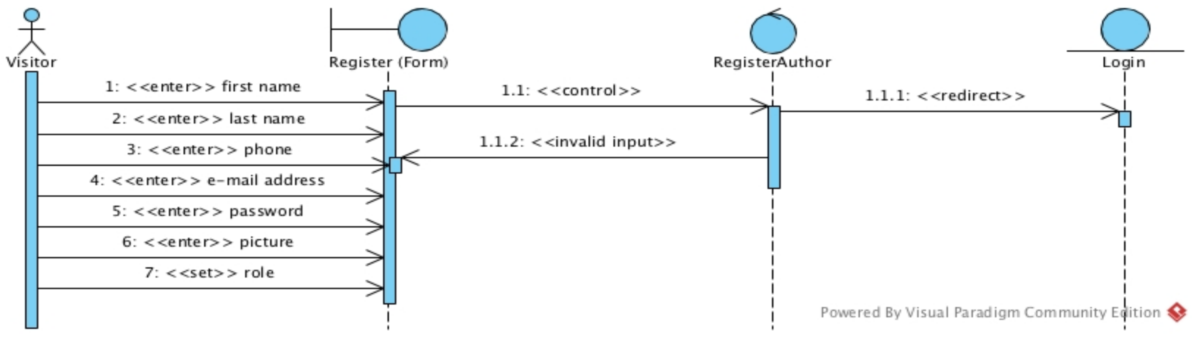
- Display Suggestion:



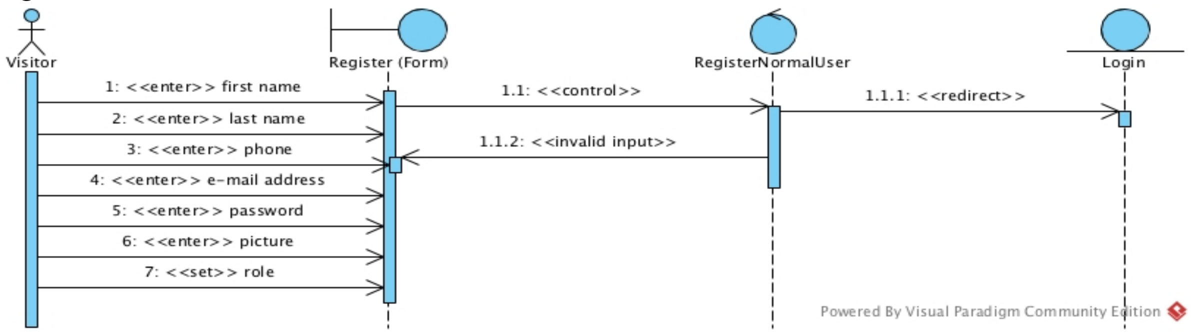
- Login:


Visitors (Author, Normal User)
Login Form
Login
Home
1: <<control>> if it exists
2: <<enter>> e-mail address
3: <<invalid input>>
4: <<enter>> password
5: <<redirect>>
Powered By Visual Paradigm Community Edition

- Registration Author:

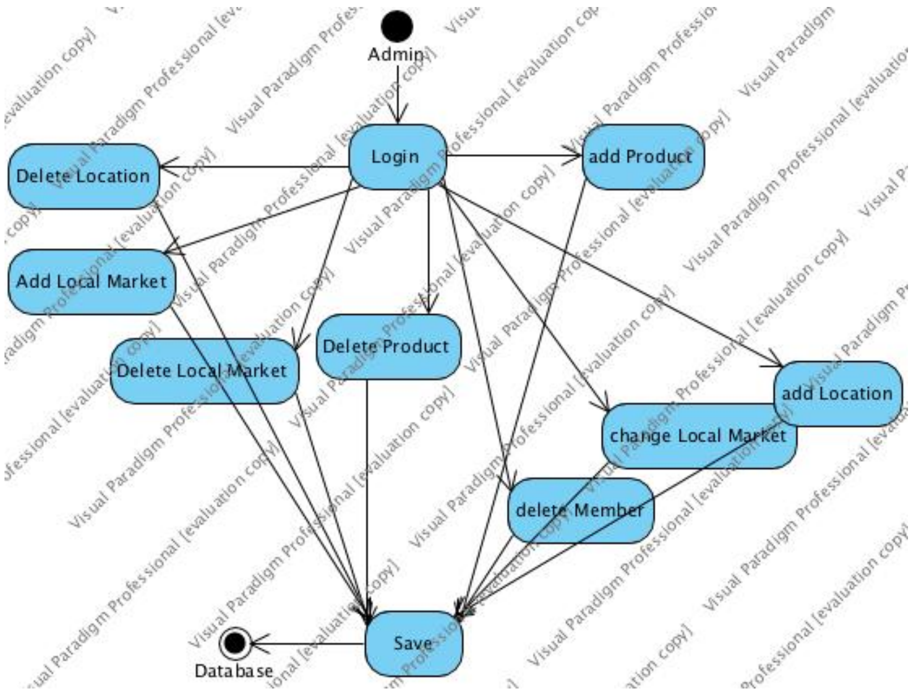


- Registration Normal User:

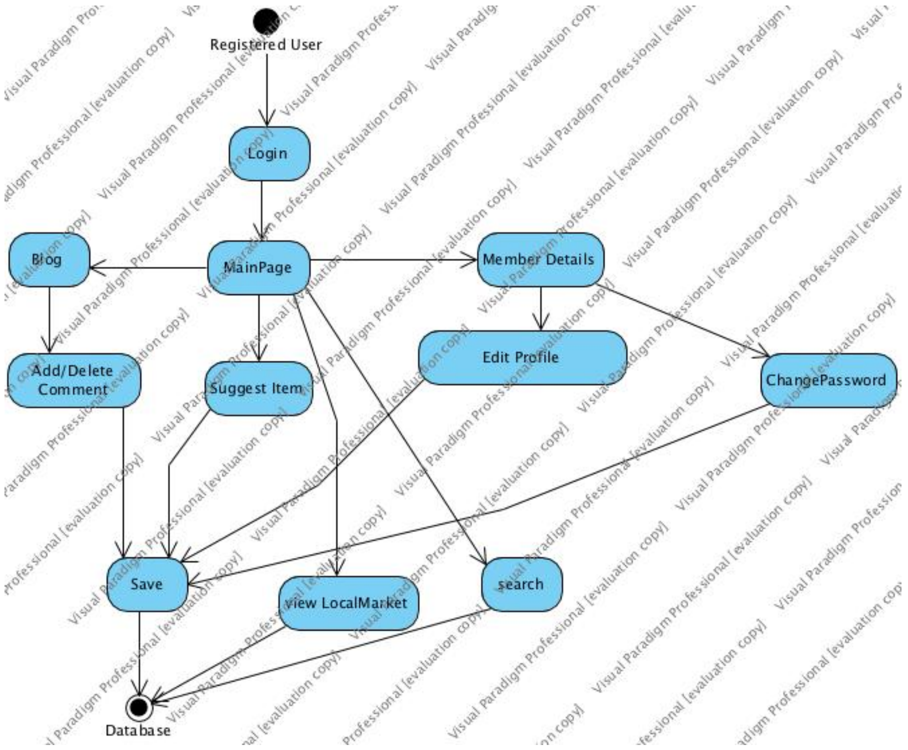


Activity Diagrams

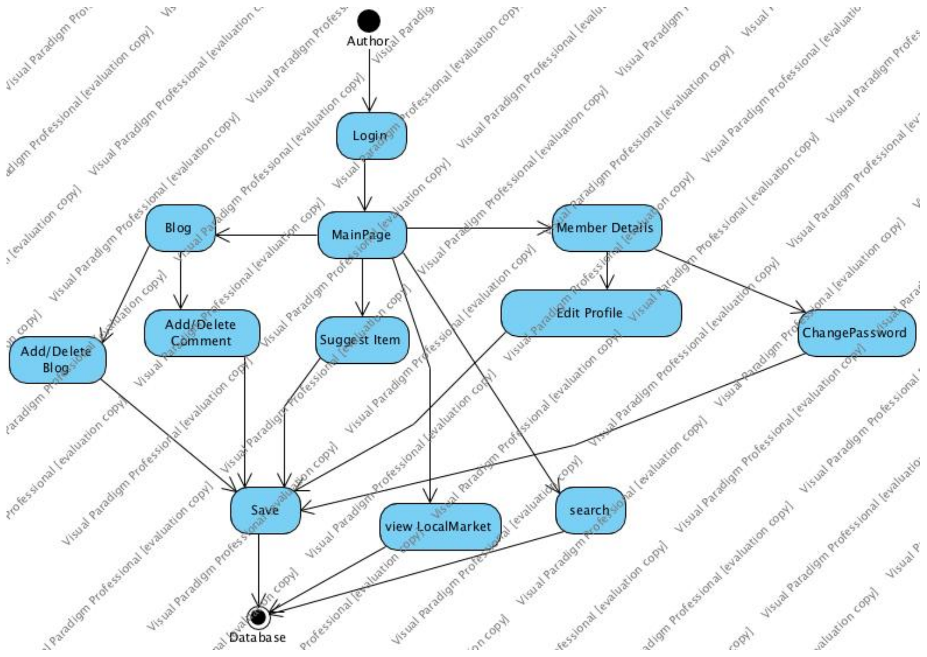
- Activity diagram is for system administrator:



- Activity diagram is for normal user:



- Activity diagram is for author:



3.5. Project Schedule

The project schedule was prepared on Google Sheets and it is shared with team members and instructors.

4. Glossary

- Admin panel: It is a panel that admin has only. It consists of a menu that includes management function for admin.
- Authentication: It is membership right and it is related to logging in to the system.
- Author: It is a member type that is approved by system administrator.
- Back end: That is combination of functions.
- Database: The collection of large data set/stack.
- Execute: Running a code line.
- Function: Function is a mathematical expression of something. That is also called method.
- GET-POST methods: They are message types of hiper text transfer protocol.
- HTTP Message: It is explained below.
- Input form: That is form that receives variables from user, and calls relevant method in back end.
- Input text-field: That is like input form.
- Linked text-label: A text-label that goes to a URL.
- MVC: It is a design methodology: model-view-controller.
- Normal user: It is member.
- Query: That is SQL (standard query language) code pair.

- ❖ Ratio button: It is a button that is just selected one time.
- ❖ Registered people, member: For this system, they are author and normal user.
- ❖ Statement: Executing code line.
- ❖ System administrator: It is a person who manages the system over admin panel.
- ❖ Unregistered people, visitor: Anyone who is not registered on the system.

5. References

1. Bruegge B. & Dutoit A.H.. (2010). *Object-Oriented Software Engineering Using UML, Patterns, and Java*, Prentice Hall, 3rd ed.
2. TazeDirekt.com – e-shopping for fresh foods, www.tazedirekt.com
3. TazeMasa.com, www.tazemasa.com
4. MemlekettenGelsin.com, www.memlekettengelsin.com
5. VitaminRehberi.org, www.vitaminrehberi.org