System and Software Architecture Description (SSAD)

The Los Angeles Community Garden Inventory and Locator

Team 13

Ardalan Yousefi	**Project Manager**
Cole Cecil	**Integrated Independent Verification & Validation**
Jeff Tonkovich	**Implementer**
Shi-Xuan Zeng	**Tester**

April 22, 2012

Version History

Date	Author	Version	Changes made	Rationale
10/06/11	Ardalan	0.1	• Completed section 1 and 2.1.	• Initial draft of SSAD.
10/07/11	Ardalan, Matt	0.2	• Edited section 1.1. • Completed section 2.2.	• The system analysis models make understanding the proposed system easier and will be later used as a basis for creating class diagrams
10/14/11	Matt, Ardalan	0.3	• Fixed the defects. • Edited section 2.1.4,and 2.2. • Corrected the UML diagrams of section 2.1	• Modified by the IIV&V's review. • First draft for section 2.2. • TA suggested some corrections for the diagrams.
10/16/11	Ardalan, Matt	0.4	• Made a few more corrections to the UML diagrams. • Fixed the defects found by the IIV&V and the TA.	• A few good points about the SSAD were made by the TA in class. • Preparing the final version for the ARB meeting.
10/20/11	Ardalan	0.5	• Edited the Artifacts and Info Diagram and added the new artifacts to Table 2.	• The TA thought the different types of users (DB manager and Admin) should be clear in the Artifacts and Info Diagram.
10/23/11	Matt, Ardalan	1.0	• Modified the preconditions of each use case. • Modified the Artifact & Info Diagram.	• Based on suggestions made during the ARB meeting.
11/20/11	Matt	1.1	• Modified the exception of UC3. Modified the post conditions of use cases.	• Based on TA's comment.
11/21/11	Ardalan, Matt	1.2	• Added sections 3 and 4.	• Completed the technology-specific design of the system for Draft DCP.
12/02/11	Ardalan, Matt	1.3	• Update Use Case Diagram and Use Cases. • Modified the Software component diagram and deployment diagram. • Updated some class diagrams in section 3.1.2.	• Update for DCP ARB. • IIV&V's and the TA's comments.
12/05/11	Ardalan	1.4	• Add the supported operating systems to the hardware components and deployment diagrams. • Changed the names of "add/delete column" use cases to "add/delete garden table column"	• Comments from the DCP ARB meeting.
02/05/12	Ardalan	2.0	• Changed the association relationship between control and boundary classes to dependencies.	• TA's grading comments

Date	Author	Version	Changes made	Rationale
02/15/12	Ardalan	2.1	• Added View DB Log, Add and Remove Garden Image, View Garden Picture, and View Driving Directions use cases. • Added Log File and Garden Image Profile to the Artifacts & Info Diagram.	• comments from RDCR ARB meeting.
03/23/12	Ardalan	3.0	• Updated a couple of process descriptions. • Added a new use case “Search Garden Table”.	• the bugs found by IIV&V.
03/26/12	Ardalan, Gary	3.1	• Modified the actors description. • Updated almost every part of section 3.	• The architecture and the implementation need to be consistent with each other.
04/14/12	Ardalan	3.2	• Added the new actor, database manager. • Modified several UML diagrams and tables.	• New requirement added by the client during the CCD.
04/22/12	Ardalan	3.3	• Modified section 4. • Modified Internal Report Generation Class Diagram.	• TA’s feedback from code-design review session.

Table of Contents

Table of Tables

Table of Figures

1. Introduction

1.1 Purpose of the SSAD

The purpose of the SSAD is to document the results of the object-oriented analysis and design of the Los Angeles Community Garden and Locator system. SSAD specifies the architecture of the system using numerous UML diagrams and tables. The implementation should be faithful to this architecture. Furthermore, the SSAD is used by the clients to help understand the structure of the system once the proposed system is delivered.

1.2 Status of the SSAD

The status of the SSAD is currently at the Initial Operational Capability Package version 3.3. A note was added to section 4, describing the deviation of the implementation from the architecture design. In addition, another bug which was found during the Code-Review design was fixed.

2. System Analysis

2.1 System Analysis Overview

The main purpose of the Los Angeles Community Garden Inventory and Locator is to provide the public with an up-to-date, easily accessible inventory of all the community gardens in Los Angeles via an online, user-friendly mapping tool. The organizations responsible for maintaining and managing these gardens will be able to update the database and generate reports from the database securely and easily.

2.1.1 System Context

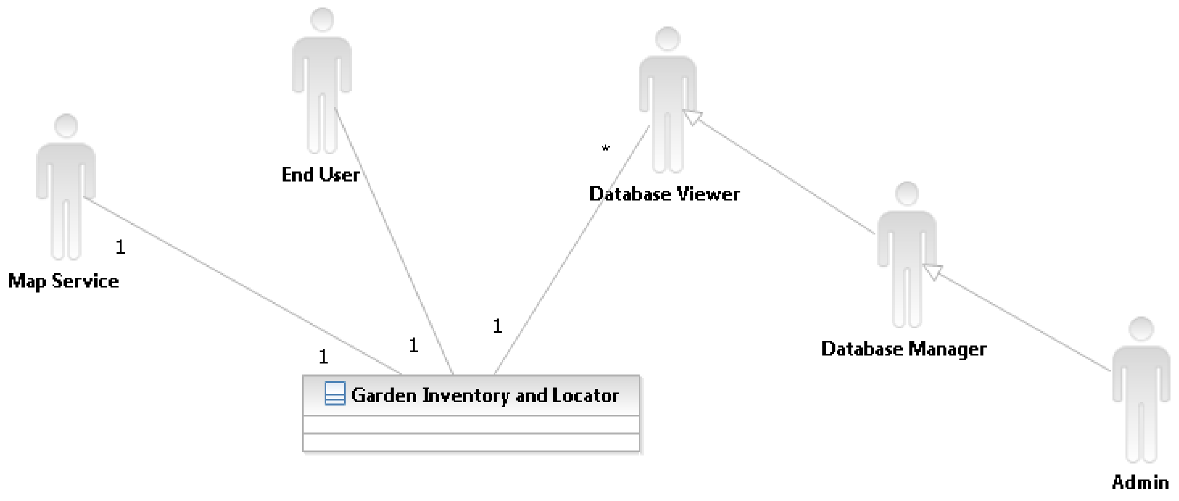

Figure 1: System Context Diagram

Table 1: Actors Summary

Actor	Description	Responsibilities
End User	People who use this system to search information about community gardens.	• Searches gardens and views their information. • Downloads reports of the gardens.
Database Manager	A member of the four organizations. He/she manages the current data of gardens.	• Updates the data of gardens. • Generates reports as needed.

Actor	Description	Responsibilities
Database Viewer	Researchers and other people who want to view all the columns of the garden table, but should not be able to edit the table.	• Views the complete data of gardens. • Generates reports as needed.
Admin	A member of the four organizations. He/she manages the data managers.	• Updates the list of database managers. • Updates the data of gardens. • Generates reports as needed.
Map Service	The online map service used by the Garden Locating System.	• Look for an address. • Pinpoint an address on the map. • Display gardens on the map.

2.1.2 Artifacts & Information

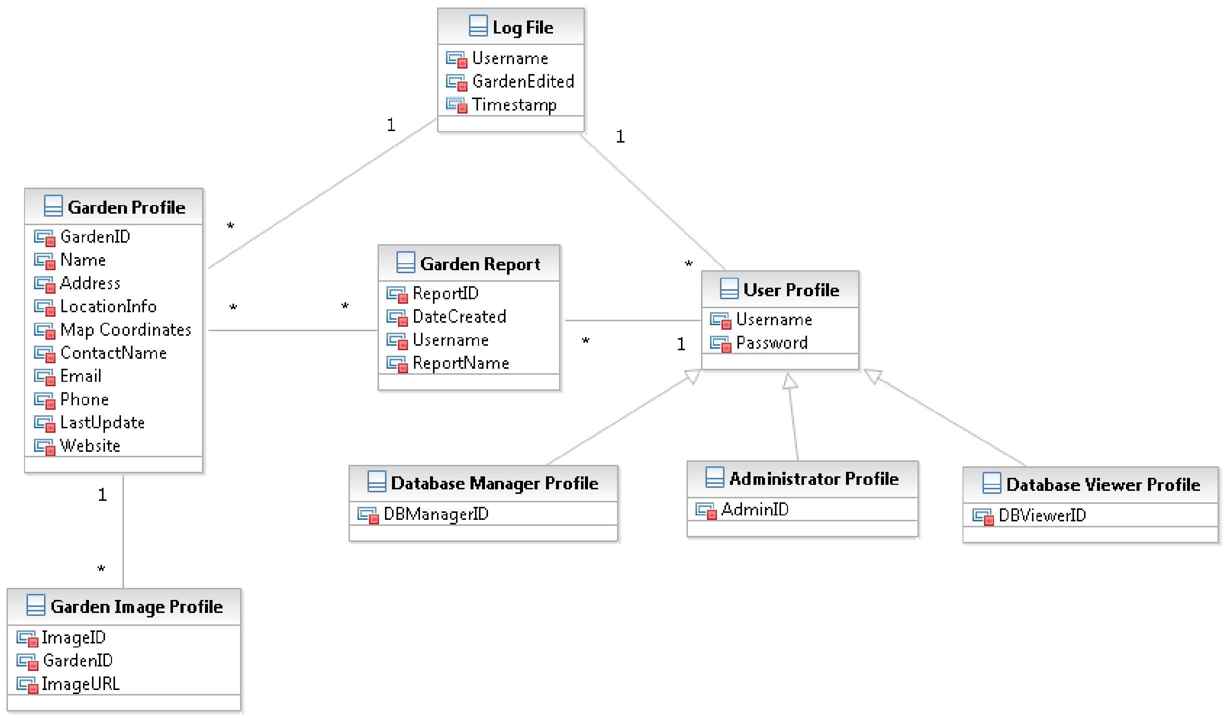

Figure 2: Artifacts and Information Diagram

Table 2: Artifacts and Information Summary

Artifact	Purpose
ATF-1: Garden Report	A report containing the information of all the gardens. Only

	the database manager can generate this report. The database manager can select what columns appear in the report. The report includes the name of the user generating the report as well as the date it was created.
ATF-2: Garden Profile	Contains the information of the community garden such as the name of the garden, the address, information about its location like on what corner it is or it is next to what shop/restaurant, X and Y coordinates of the garden to be used in the online map, the name, email, and phone number of the person responsible for this garden, the date this information was updated, the URL of the garden's website, and so forth.
ATF-3: Database Manager Profile	Represents the database managers who can update the database. Only the administrator has access to this list.
ATF-4: User Profile	Includes the username and passwords of all the users who are authorized to work with the database management and report generating modules of the system.
ATF- 5: Administrator Profile	Represents the administrator. The admin is a fixed username and password stored in the database.
ATF-6: Log File	Represents the file that logs the changes made to the database. Stores the name of the user, the name of the garden that was changed, and the time and date of change.
ATF-7: Garden Image Profile	Represents the URL of the images of each Garden.
ATF-8: Database Viewer Profile	Represents the database viewers who have read-only access to the complete garden table. Only the administrator has access to this list.

2.1.3 Behavior

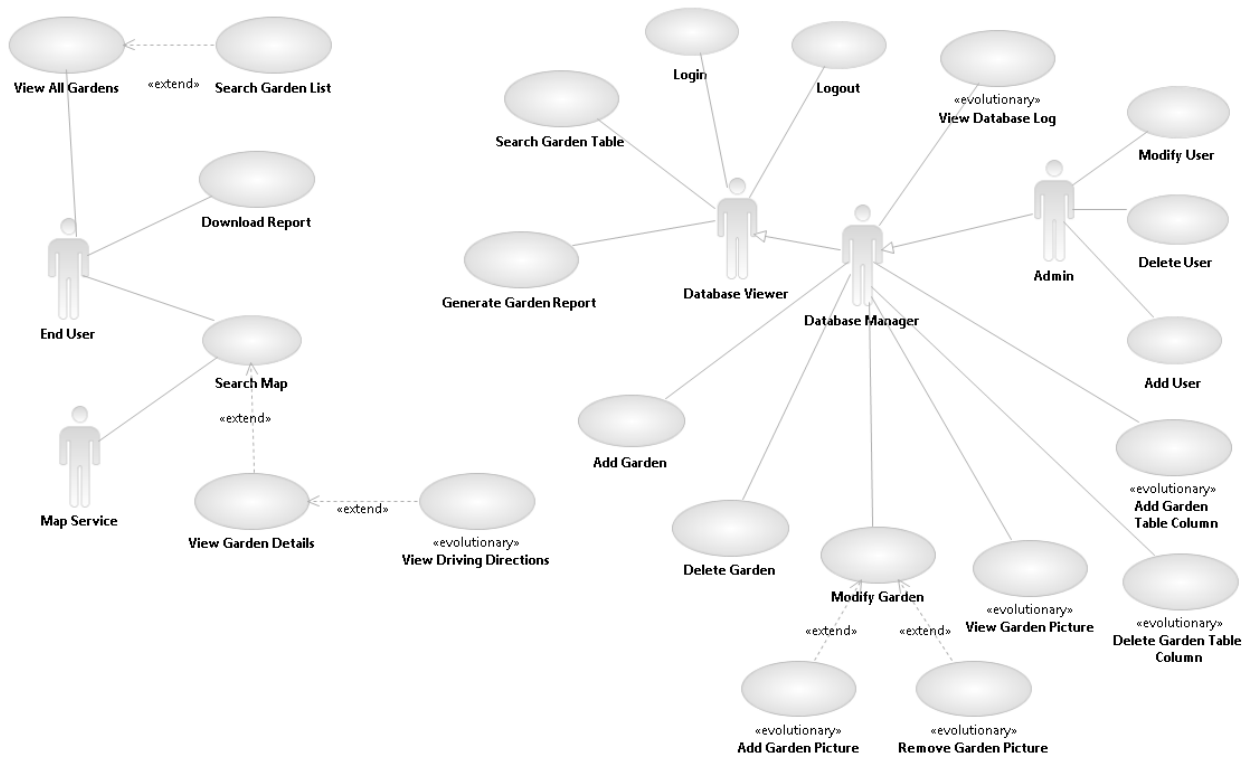

Figure 3: Process Diagram

2.1.3.1 Login/Logout

2.1.3.1.1 Login

Table 3: Process Description - Login

Identifier	UC-1 : Login
Purpose	To determine if a person is authenticated when he/she is logging in to the system. If so, it will determine the person as a database manager or admin.
Requirements	CR-3 Login to the System
Development Risks	None
Pre-conditions	System database is properly initialized.
Post-conditions	If a person is authorized, he/she will access to system with appropriate privileges; otherwise, he/she will be denied by the system.

Table 4: Typical Course of Action – Login: Successful

Seq#	Actor's Action	System's Response
1	Enters a user name and password	
2	Clicks Login button	
3		Redirects the actor to Database Manager home page.

Table 5: Alternate Course of Action – Login: Failure

Seq#	Actor's Action	System's Response
1	Enters a user name and password	
2	Clicks Login button	
3		Displays an error message: "Username or password is wrong" in a dialog box.
4	Clicks OK button	
5		Redirects the actor to the login page.

Note: Entering only a username, a password, or blanks will be treated the same as entering an invalid username and password.

2.1.3.1.2 Logout

Table 6: Process Description - Logout

Identifier	UC-2 : Logout
Purpose	To log out of the system
Requirements	CR-4 Logout of the System
Development Risks	None
Pre-conditions	The user is a database manager or admin. The user's login session still exists.
Post-conditions	The user's login session is terminated.

Table 7: Typical Course of Action – Logout

Seq#	Actor's Action	System's Response
1	Clicks the "log out" hypertext	
2		Redirect the actor to the login page.

Alternate Course of Action for Logout is not applicable.

Table 8: Exception Course of Action – Logout

Seq#	Actor's Action	System's Response
1	User tries to directly load logout page	
2		Display an error message: "You have not logged in yet."

2.1.3.2 Garden Locator

2.1.3.2.1 Search Map

Table 9: Process Description – Search Map

Identifier	UC-3 : Search Map
Purpose	To search an address entered by an end-user. If the address exists, the system will show nearby gardens on a map.
Requirements	CR-18 View Public Garden Information Map and CR-19 Search Public Garden Information
Development Risks	None.
Pre-conditions	System database is properly initialized.
Post-conditions	The screen will show the nearby gardens on a map.

Table 10: Typical Course of Action – Search map: Valid Address

Seq#	Actor's Action	System's Response
1	[End User] Enters an address and a radius.	
2	[End User] Clicks on "Find Nearby Gardens" button.	
3		Send the address and radius to the map service.
4	[Map Service] Find the address on the map.	
5	[Map Service] Pinpoint the address on the map and mark the gardens in the radius.	
6	[Map Service] Send the map back to the system.	
7		Display the map to the user.

Table 11: Alternate Course of Action – Search map: Invalid Address

Seq#	Actor's Action	System's Response
1	[End User] Enters an address and a radius.	
2	[End User] Clicks "Find Nearby Gardens" button.	
3		Send the address and radius to the map service.
4	[Map Service] Find the address is not existed or invalid.	
5	[Map Service] Report back to the system.	
		Display an error message: "The address cannot be found." in a dialog box.

Table : Alternate Course of Action – Search map: Negative radius

Seq#	Actor's Action	System's Response
1	[End User] Enters an address and a radius.	
2	[End User] Clicks "Find Nearby Gardens" button.	
3		Send the address and radius to the map service.
4	[Map Service] Find the radius is negative.	
5	[Map Service] Report back to the system.	
6		Display an error message: "The radius must be greater than zero." in a dialog box.

Table 12: Exception Course of Action – Search map: Google Maps is not available

Seq#	Actor's Action	System's Response
1	[End User] Enters an address.	
2	[End User] Clicks "Find Nearby Gardens" button.	
		Send the address and radius to the map service.
		Display an error message: "Google

		Maps is not available now." in a dialog box.

2.1.3.2.2 View all gardens

Table 13: Process Description – View all Gardens

Identifier	UC-4 : View all Gardens
Purpose	To provide a list of existing gardens to an end user.
Requirements	CR-17 View Public Garden Information
Development Risks	None
Pre-conditions	System database is properly initialized. At least one garden exists in the database.
Post-conditions	The end user will be redirected to a page with a list of gardens

Table 14: Typical Course of Action – View all Gardens

Seq#	Actor's Action	System's Response
1	Click the "List of Gardens" hypertext.	
2		Redirect the end user to a page with a list of existing gardens.
3		Retrieve the information of all gardens from the database.
4		Display the retrieved info on the list.

Alternate course of action for View All Gardens is not applicable.

Exceptional course of action for View All Gardens is not applicable.

2.1.3.2.3 View Garden Details

Table 15: Process Description – View Garden Details

Identifier	UC-5 : View all Garden Details
Purpose	To provide a more detailed information of a specific garden to an end user.
Requirements	CR-20 View Public Garden Information Detail
Development Risks	None.

Pre-conditions	System database is properly initialized.
Post-conditions	Display the detailed information of a specific garden.

Table 16: Typical Course of Action – View Garden Details: From Map

Seq#	Actor's Action	System's Response
1	Click on a garden on the map.	
2		Display the detailed information of the garden in a bubble.

Exceptional course of action for search map is not applicable.

Exceptional course of action for search map is not applicable.

2.1.3.2.4 Download Report

Table 17: Process Description – Download Report

Identifier	UC-13 : Download Report
Purpose	Download a report file of the gardens in PDF format.
Requirements	CR-21 Download Garden Report for End-Users
Development Risks	None.
Pre-conditions	System database is properly initialized.
Post-conditions	The information of the gardens is saved to a file and passed on as a download link to the end-user.

Table 18: Typical Course of Action – Download Report

Seq#	Actor's Action	System's Response
1	Click "Download PDF" button.	
2		Create a report file.
3		Navigate to the report file.

Alternate course of action is not applicable.

Exceptional course of action is not applicable.

2.1.3.2.5 View Driving Directions

Table 19: Process Description – View Driving Directions

Identifier	UC-16 : View Driving Directions
Purpose	To view the driving directions from a specific address to one of the gardens.
Requirements	CR-28 View Garden Driving Direction
Development Risks	None.
Pre-conditions	A garden is selected.
Post-conditions	The driving directions will be shown on the map.

Table 20: Typical Course of Action – View Driving Directions: Valid Address

Seq#	Actor's Action	System's Response
1	[End User] Enters a "From" address.	
2	[End User] Clicks on Go button.	
3		Send the address to the map service.
4	[Map Service] Find the address on the map.	
5	[Map Service] Pinpoint the address on the map and show the direction from the garden to the address.	
6	[Map Service] Send the map back to the system.	
7		Display the map to the user.

Table 21: Alternate Course of Action – View Driving Directions: Invalid Address

Seq#	Actor's Action	System's Response
1	[End User] Enters a "From" address.	
2	[End User] Clicks Go button.	
3		Send the address to the map service.
4	[Map Service] Cannot find the address.	
5	[Map Service] Report back to the system.	
6		Display an error message: "The address cannot be found." in a dialog box.

Table 22: Exception Course of Action – View Driving Directions: Google Maps is not available

Seq#	Actor's Action	System's Response
1	[End User] Enters an address.	
2	[End User] Clicks Search button.	
3		Send the address to the map service.
4		Display an error message: "Google Maps is not available now." in a dialog box.

2.1.3.2.6 Search Garden List

Table 23: Process Description – Search Garden Table

Identifier	UC-22 : Search Garden List
Purpose	Search for particular keywords in the garden list.
Requirements	CR-7 Search Garden Information
Development Risks	None
Pre-conditions	The database is properly initialized.
Post-conditions	The search results, if any, are shown to the end user.

Table 24: Typical Course of Action – Search Garden List

Seq#	Actor's Action	System's Response
1	Enters search keyword(s) in the search box.	
2	Clicks the search button.	
3		Looks for the keyword(s) in the entire table.
4		Shows results on the screen.

Table 25: Alternate Course of Action – Search Garden List: keyword(s) not found in table

Seq#	Actor's Action	System's Response
1	Enters search keyword(s) in the search box.	
2	Clicks the search button.	
3		Looks for the keyword(s) in the entire table.

4		Shows the message "the keyword(s) you entered did not match any garden.".

Exceptional course of action is not applicable.

2.1.3.3 Garden Management

2.1.3.3.1 Add Garden

Table 26: Process Description – Add Garden

Identifier	UC-6 : Add Garden
Purpose	Add new gardens to the garden inventory.
Requirements	CR-9 Add Garden Record
Development Risks	None
Pre-conditions	The user has logged in as an administrator or a database manager.
Post-conditions	The garden information is saved to the database.

Table 27: Typical Course of Action – Add Garden

Seq#	Actor's Action	System's Response
1	Enter the appropriate information of garden	
2	Click Add.	
3		Validate the entered information.
4		[valid] save the new garden to the database.
5		Display the newly added garden in the list of gardens.

Table 28: Alternate Course of Action – Add Garden

Seq#	Actor's Action	System's Response
1	Enter the appropriate information of garden	
2	Click Add.	
3		Validate the entered information.
4		[invalid] Highlight the invalid fields and display a message saying why the entered values are invalid. The fields

		are still populated.

Exceptional course of action is not applicable.

2.1.3.3.2 Delete Garden

Table 29: Process Description – Delete Garden

Identifier	UC-7 : Delete Garden
Purpose	Delete a garden from the garden inventory.
Requirements	CR-11 Delete Garden Record
Development Risks	None
Pre-conditions	The user has logged in as an administrator or a database manager.
Post-conditions	The garden entry is deleted from the database.

Table 30: Typical Course of Action – Delete Garden

Seq#	Actor's Action	System's Response
1	Click Delete on a row which should be deleted.	
2		Display a confirmation dialog box.
3	Click Yes.	
4		Delete the row of data from the database.
5		Update the list of gardens to show this change.

Table 31: Alternate Course of Action – Delete Garden

Seq#	Actor's Action	System's Response
1	Click Delete on a row which should be deleted.	
2		Display a confirmation dialog box.
3	Click No.	
4		Cancel the action, and display the screen as the same page.

Exceptional course of action is not applicable.

2.1.3.3.3 Modify Garden

Table 32: Process Description – Modify Garden

Identifier	UC-8 : Modify Garden
Purpose	Modify and update the information of a garden in the inventory.
Requirements	CR-10 Modify Garden Record
Development Risks	None
Pre-conditions	The user has logged in as an administrator or a database manager. At least one garden exists in the database.
Post-conditions	The garden information is updated in the database.

Table 33: Typical Course of Action – Modify Garden

Seq#	Actor's Action	System's Response
1	Click the edit button next to a garden row in the list of gardens.	
2		Change the fields of that row into text boxes.
3	Change the information.	
4	Click the update button.	
5		Validate the entered information.
6		[valid] Update the garden entry in the database.
7		Update the list of gardens to show this change.

Table 34: Alternate Course of Action – Modify Garden

Seq#	Actor's Action	System's Response
1	Click the edit button next to a garden row in the list of gardens.	
2		Change the fields of that row into text boxes.
3	Change the information.	
4	Click the update button.	
5		Validate the entered information.
6		[invalid] Highlight the invalid fields and display a message saying why the entered values are invalid.
7		Display the row only with the valid value.

Exceptional course of action is not applicable.

2.1.3.3.4 Generate Garden Report

Table 35: Process Description – Export Gardens

Identifier	UC-9 : Generate Garden Report
Purpose	Export the information of the gardens to a XLSX file.
Requirements	CR-8 Export Garden Information
Development Risks	None.
Pre-conditions	The user has logged in as an administrator, a database manager, or a database viewer.
Post-conditions	The information of the gardens is saved to a file whose link is made available to the admin/DB manager/DB viewer.

Table 36: Typical Course of Action – Export Gardens

Seq#	Actor's Action	System's Response
1	Click Report button.	
2		Navigate to the Report page.
3	Select fields to be included in the report file	
4	Click Generate Report	
5		Navigate to the report file.

Alternate course of action is not applicable.

Exceptional course of action is not applicable.

2.1.3.3.5 Add Garden Table Column

Table 37: Process Description – Add Garden Table Column

Identifier	UC-14 : Add Garden Table Column
Purpose	Add a new column name to the table of database.
Requirements	CR-12 Add Garden Table Column
Development Risks	None
Pre-conditions	The user has logged in as an administrator or a database manager.
Post-conditions	The new column name is saved to the database.

Table 38: Typical Course of Action – Add Garden Table Column

Seq#	Actor's Action	System's Response
1	Click Add Column button.	
2		Navigate to the Add Column page.
3	Enter the name of the new column, and select the type of column.	
4	Click Add button.	
5		Alter table in the database to add a new column.
6		Navigate back to Garden Management Page.

Table 39: Alternate Course of Action – Add Garden Table Column

Seq#	Actor's Action	System's Response
1	Click Add Column button.	
2		Navigate to the Add Column page.
3	Enter a duplicated name of the new column, and select the type of column.	
4	Click Add button.	
5		Show a message box with message, "The column name already exists."
6	Click OK button.	
		Redirect back to Add Column page.

Exceptional course of action is not applicable.

2.1.3.3.6 Delete Garden Table Column

Table 40: Process Description – Delete Garden Table Column

Identifier	UC-15 : Delete Garden Table Column
Purpose	Delete a column from the table of database.
Requirements	CR-13 Delete Garden Table Column
Development Risks	None.
Pre-conditions	The user has logged in as an administrator or a database manager.
Post-conditions	The column name is deleted from the database.

Table 41: Typical Course of Action – Delete Garden Table Column

Seq#	Actor's Action	System's Response
1	Click on a checkbox next to a column heading.	
1	Click Delete Column button.	
2		Display a confirmation box.
3	Click Yes.	
5		Alter table in the database to delete the column.
6		Refresh the Garden Management Page.

Alternate course of action is not applicable.

Exceptional course of action is not applicable.

2.1.3.3.7 Add Garden Picture

Table 42: Process Description – Add Garden Picture

Identifier	UC-17: Add Garden Picture
Purpose	Add the URL of a picture to a garden.
Requirements	CR-25 Add Picture to Garden Record
Development Risks	None.
Pre-conditions	The user has logged in as an administrator or a database manager. The database is properly initialized.
Post-conditions	The picture URL is stored in the database.

Table 43: Typical Course of Action – Add Garden Picture

Seq#	Actor's Action	System's Response
1	Enter a URL for the picture.	
2	Click "Add Picture".	
3		Check if the URL exists.
4		Add the picture URL to the database.

Alternate course of action is not applicable.

Table 44: Exception Course of Action – Add Garden Picture: URL does not exist

Seq#	Actor's Action	System's Response
1	Enter a URL for the picture.	

2	Click "Add Picture".	
3		Check if the URL exists.
4		Display a "picture does not exist" message.

2.1.3.3.8 Delete Garden Picture

Table 45: Process Description – Delete Garden Picture

Identifier	UC-18: Delete Garden Picture
Purpose	Delete the URL of a garden picture.
Requirements	CR-26 Delete Picture from Garden Record
Development Risks	None.
Pre-conditions	The user has logged in as an administrator or a database manager. The database is properly initialized.
Post-conditions	The picture URL is deleted from the database.

Table 46: Typical Course of Action – Add Garden Picture

Seq#	Actor's Action	System's Response
1	Choose a picture.	
2	Click "Delete Picture".	
3		Display a confirmation box.
4	Click Yes.	
5		Delete the picture's URL from the database.
6		Display a "picture deleted successfully" message.

Alternate course of action is not applicable.

Exceptional course of action is not applicable.

2.1.3.3.9 View Garden Picture

Table 47: Process Description – View Garden Picture

Identifier	UC-19 : View Garden Picture
Purpose	View the pictures of a garden.
Requirements	CR-27 View Pictures of Garden Record
Development Risks	None
Pre-conditions	The user has logged in as an administrator or a database manager.

	The database is properly initialized.
Post-conditions	The garden pictures are displayed to the database manager.

Table 48: Typical Course of Action – View Garden Picture

Seq#	Actor's Action	System's Response
1	Clicks the "View Images" button next to a garden name.	
2		Navigate to Images web page.
3		Get image URLs from the database.
4		Get images from the URLs.
5		Display images on the web page.

Alternate course of action is not applicable.

Exceptional course of action is not applicable.

2.1.3.3.10 Search Garden Table

Table 49: Process Description – Search Garden Table

Identifier	UC-21 : Search Garden Table
Purpose	Search for particular keywords in the garden table.
Requirements	CR-7 Search Garden Information
Development Risks	None
Pre-conditions	The user has logged in as a database manager or a database viewer. The database is properly initialized.
Post-conditions	The search results, if any, are shown to the user.

Table 50: Typical Course of Action – Search Garden Table

Seq#	Actor's Action	System's Response
1	Enters search keyword(s) in the search box.	
2	Clicks the search button.	
3		Looks for the keyword(s) in the entire table.
4		Shows results on the screen.

Table 51: Alternate Course of Action – Search Garden Table: keyword(s) not found in table

Seq#	Actor's Action	System's Response
1	Enters search keyword(s) in the search box.	
2	Clicks the search button.	
3		Looks for the keyword(s) in the entire table.
4		Shows the message "the keyword(s) you entered did not match any garden.".

Exceptional course of action is not applicable.

2.1.3.3.11 View Database Log

Table 52: Process Description – View Database Log

Identifier	UC-20 : View Database Log
Purpose	View the log of changes made to the database.
Requirements	CR-24 View Database Log
Development Risks	None
Pre-conditions	The user has logged in as an administrator or a database manager. The log file is available to be read.
Post-conditions	The contents of the log file is displayed to the database manager.

Table 53: Typical Course of Action – View Database Log : log file accessible

Seq#	Actor's Action	System's Response
1	Clicks the "View Database Log" link.	
2		Read from the log File.
3		Display contents on screen.

Alternate course of action is not applicable.

Table 54: Exception Course of Action – View Database Log : log file not accessible

Seq#	Actor's Action	System's Response
1	Clicks the "View Datbase Log" link.	
2		Cannot read from the log File.

3		Display “cannot read from log file” message.

2.1.3.4 User Management

2.1.3.4.1 Add Database Manager

Table 55: Process Description – Add Database Manager

Identifier	UC-10 : Add Database Manager
Purpose	Add a new database manager.
Requirements	CR-14 Add User
Development Risks	None
Pre-conditions	The user has logged in as an administrator.
Post-conditions	A new database manager is created in the database.

Table 56: Typical Course of Action – Add Database Manager

Seq#	Actor’s Action	System’s Response
1	Enters the information of the new user and click “Create Database Manager” button.	
2		Validate the entered information.
3		[valid] save the new user to the database.
4		Display the newly added user in the list of Database Managers.

Table 57: Alternate Course of Action – Add Database Manager

Seq#	Actor’s Action	System’s Response
1	Enters the information of the new user and click “Create Database Manager” button.	
2		Validate the entered information.
3		[invalid] Highlight the invalid fields and display a message saying why the entered values are invalid.

Exceptional course of action is not applicable.

2.1.3.4.2 Delete Database Manager

Table 58: Process Description – Delete Database Manager

Identifier	UC-11 : Delete Database Manager
Purpose	Delete a database manager.
Requirements	CR-16 Delete User
Development Risks	None
Pre-conditions	The user has logged in as an administrator.
Post-conditions	The database manager is deleted from the database.

Table 59: Typical Course of Action – Delete Database Manager

Seq#	Actor's Action	System's Response
1	Click "Remove" on a row which should be deleted	
2		Display a confirmation dialog box.
3	Click OK.	
4		Delete the row of data from the database.
5		Update the list of database managers to show this change.

Table 60: Alternate Course of Action – Delete Database Manager

Seq#	Actor's Action	System's Response
1	Click "Remove" on a row which should be deleted.	
2		Display a confirmation dialog box.
3	Click Cancel.	
4		Cancel the action, and display the screen as the same page.

Exceptional course of action is not applicable.

2.1.3.4.3 Modify Database Manager

Table 61: Process Description – Modify Database Manager

Identifier	UC-12 : Modify Database Manager
Purpose	Modify a database manager's password.

Requirements	CR-15 Modify User
Development Risks	None
Pre-conditions	The user has logged in as an administrator. At least one database manager exists in the database.
Post-conditions	The database manager’s password is updated in the database.

Table 62: Typical Course of Action – Modify Database Manager

Seq#	Actor’s Action	System’s Response
1	Click the “Change Password” button next to a database manager row in the list of the database managers.	
2		Navigate to “change password” page.
3	Enter new, and confirm passwords.	
4	Click the “Change Password” button.	
5		Validate the entered information.
6		[valid] Update the database manager entry in the database.
7		Update the list of database managers to show this change.

Table 63: Alternate Course of Action – Modify Database Manager

Seq#	Actor’s Action	System’s Response
1	Click the “Change Password” button next to a database manager row in the list of the database managers.	
2		Navigate to “change password” page.
3	Enter new, and confirm passwords.	
4	Click the “Change Password” button.	
5		Validate the entered information.
6		[invalid] Highlight the invalid fields and display a message saying why the entered values are invalid.

Exceptional course of action is not applicable.

2.1.4 Modes of Operation

The Los Angeles Community Garden Inventory and Locator works in only one mode of operation.

2.2 System Analysis Rationale

Fundamentally, the Los Angeles Community Garden Locator and Inventory is comprised of two subsystems:

1. The online garden map system to which the general public have access.
2. The database management system to which only database managers and administrators have access.

In addition, based on our analysis of how user interacts with the system as shown in the Behavior section of System Behavior Analysis, four types of users have been identified:

1. **End User**: These users search and view information of gardens from the system. They do not have to login to the system.
2. **Database Viewer**: These users have read-only access to the complete garden table. They can also generate reports from the table.
3. **Database Manager**: These users update the database and generate reports. They have to login to the system in order to do these actions.
4. **Admin**: This admin is authorized to manage the user accounts and passwords. He/She is able to update the database and generate reports, too. The admin also has to login the system in order to do these actions.

There is one external actor interacting with the system.

1. **Map Service**: It helps us to search the nearby gardens and display the search results on the map.

3. Technology-Specific System Design

3.1 Design Overview

3.1.1 System Structure

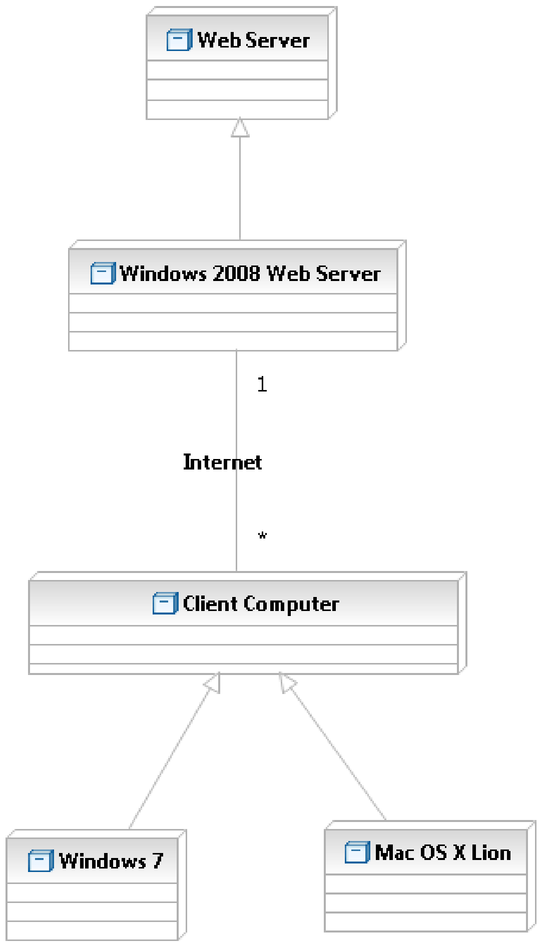

Figure 4: Hardware Component Class Diagram

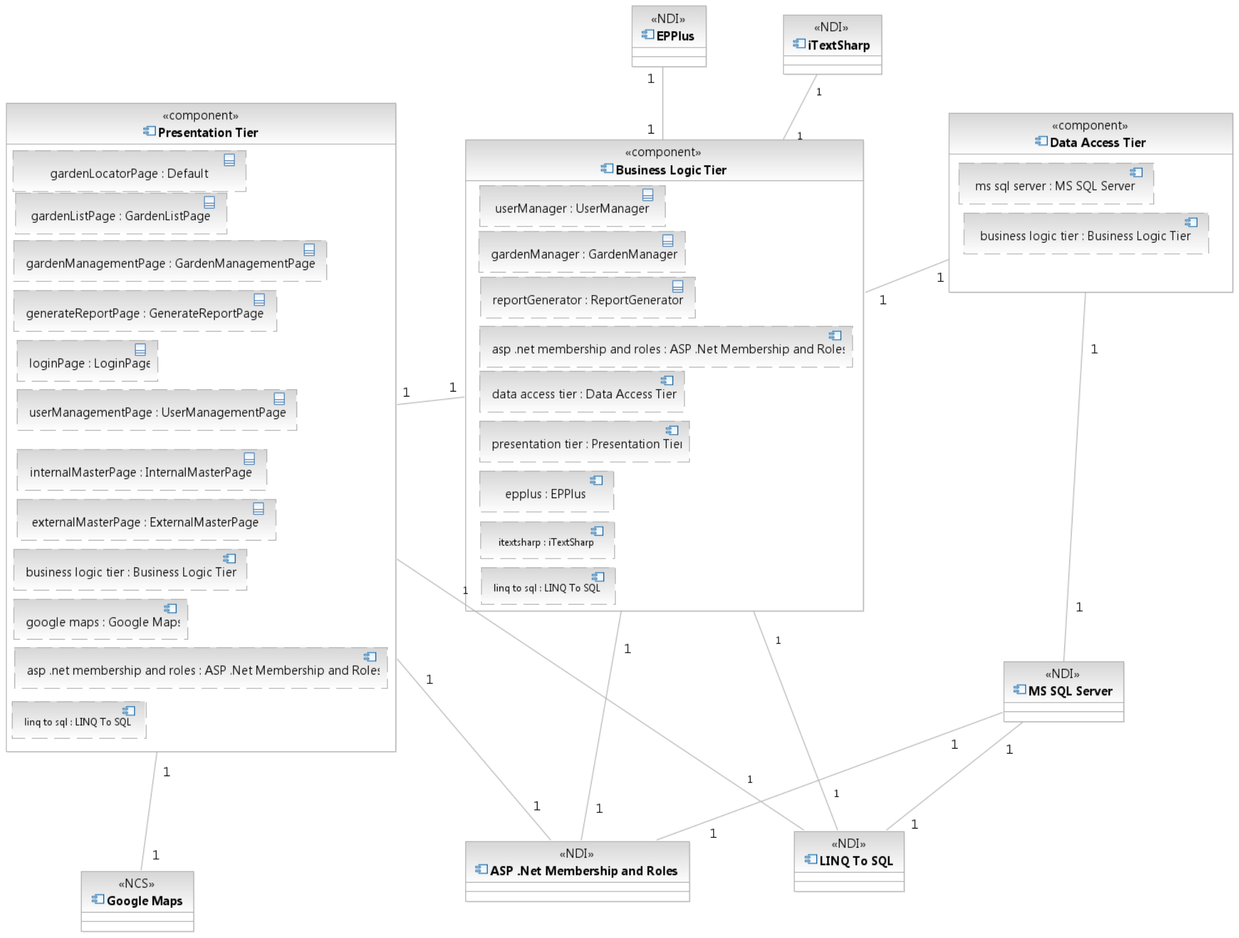
«NDI» EPPlus
«NDI» iTextSharp
«component» Presentation Tier
gardenLocatorPage : Default
gardenListPage : GardenListPage
gardenManagementPage : GardenManagementPage
generateReportPage : GenerateReportPage
loginPage : LoginPage
userManagementPage : UserManagementPage
internalMasterPage : InternalMasterPage
externalMasterPage : ExternalMasterPage
business logic tier : Business Logic Tier
google maps : Google Maps
asp .net membership and roles : ASP .Net Membership and Roles
linq to sql : LINQ To SQL
«component» Business Logic Tier
userManager : UserManager
gardenManager : GardenManager
reportGenerator : ReportGenerator
asp .net membership and roles : ASP .Net Membership and Roles
data access tier : Data Access Tier
presentation tier : Presentation Tier
epplus : EPPlus
itextsharp : iTextSharp
linq to sql : LINQ To SQL
«component» Data Access Tier
ms sql server : MS SQL Server
business logic tier : Business Logic Tier
«NDI» MS SQL Server
«NCS» Google Maps
«NDI» ASP .Net Membership and Roles
«NDI» LINQ To SQL

Figure 5: Software Component Class Diagram

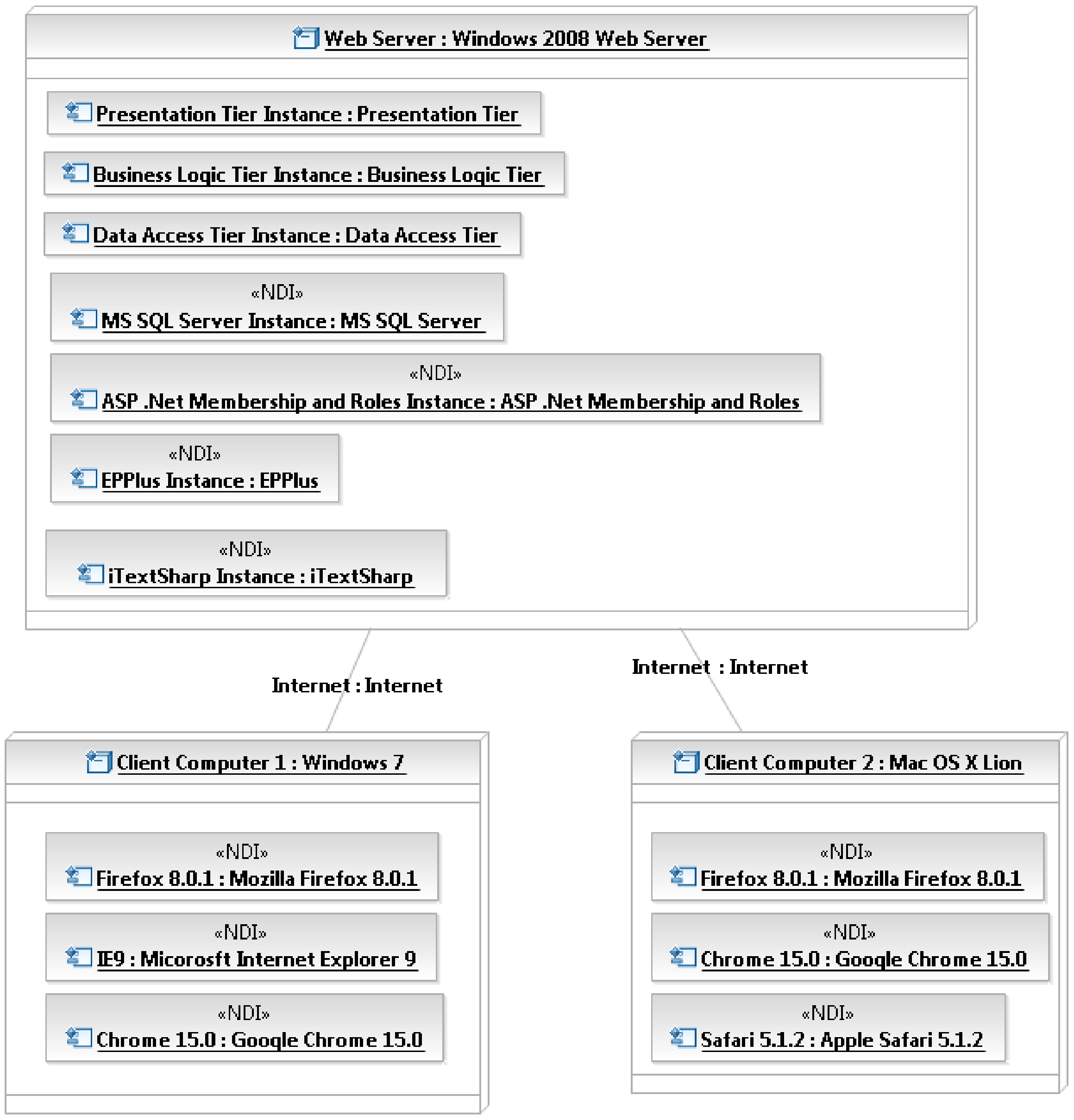
Web Server : Windows 2008 Web Server
Presentation Tier Instance : Presentation Tier
Business Logic Tier Instance : Business Logic Tier
Data Access Tier Instance : Data Access Tier
«NDI»
MS SQL Server Instance : MS SQL Server
«NDI»
ASP .Net Membership and Roles Instance : ASP .Net Membership and Roles
«NDI»
EPPlus Instance : EPPlus
«NDI»
iTextSharp Instance : iTextSharp
Internet : Internet
Internet : Internet
Client Computer 1 : Windows 7
«NDI»
Firefox 8.0.1 : Mozilla Firefox 8.0.1
«NDI»
IE9 : Micorosft Internet Explorer 9
«NDI»
Chrome 15.0 : Google Chrome 15.0
Client Computer 2 : Mac OS X Lion
«NDI»
Firefox 8.0.1 : Mozilla Firefox 8.0.1
«NDI»
Chrome 15.0 : Google Chrome 15.0
«NDI»
Safari 5.1.2 : Apple Safari 5.1.2

Figure 6: Deployment Diagram

Table 64: Hardware Component Description

Hardware Component	Description
Client Computer	A computer that browses the Garden Inventory and Locator website via the Internet.
Windows 7	A client computer with Windows 7 as its operating system.
Mac OS X Lion	A client computer with Mac OS X Lion as its operating system.
Web Server	A web server that hosts the Garden Inventory and Locator website.
Windows 2003 Web	A web server running on Microsoft Windows Server 2003 that

Server	hosts the Garden Inventory and Locator website.

Table 65: Software Component Description

Software Component	Description
Presentation Tier	This component contains all the web pages and their code-behind files.
Business Logic Tier	This component holds all the classes that contain the business logic of the system.
Data Access Tier	This component is comprised of classes that connect to the database and read/write data from/to it.
Google Maps	A web service that is used by the presentation tier to display the gardens on the map of Los Angeles.
ASP .Net Membership and Roles	This component is one of the features of ASP .Net that facilitates authentication, authorization, and user management.
EPPlus	This NDI is used to create reports in excel format from the gardens in the database.
iTextSharp	This NDI is used to create reports in pdf format from the gardens in the database.
MS SQL Server	A DBMS that stores all the information of gardens and users.
LINQ to SQL	A feature of .NET that facilitates connecting to the database and managing the data in the database; eliminates the need to write classes for connecting to the database from scratch.

3.1.2 Design Classes

The design classes describe the relationships among the boundary, control, and entity classes of the Garden Inventory and Locator website.

3.1.2.1 Interface Classes

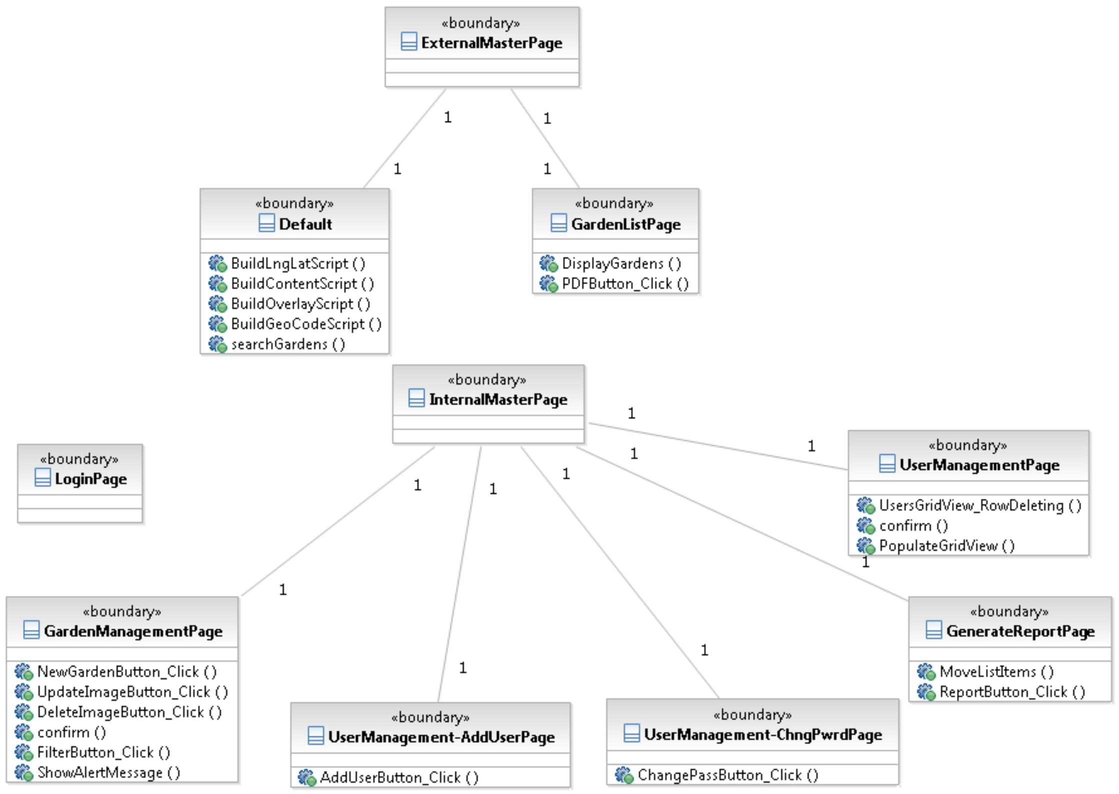
«boundary» ExternalMasterPage
«boundary» Default
BuildLngLatScript ()
BuildContentScript ()
BuildOverlayScript ()
BuildGeoCodeScript ()
searchGardens ()
«boundary» GardenListPage
DisplayGardens ()
PDFButton_Click ()
«boundary» InternalMasterPage
«boundary» LoginPage
«boundary» UserManagementPage
UsersGridView_RowDeleting ()
confirm ()
PopulateGridView ()
«boundary» GardenManagementPage
NewGardenButton_Click ()
UpdateImageButton_Click ()
DeleteImageButton_Click ()
confirm ()
FilterButton_Click ()
ShowAlertMessage ()
«boundary» UserManagement-AddUserPage
AddUserButton_Click ()
«boundary» UserManagement-ChngPwrdPage
ChangePassButton_Click ()
«boundary» GenerateReportPage
MoveListItems ()
ReportButton_Click ()
1

Figure 7: Interface Class Diagram

Table 66: Interface Class Description

Class	Type	Description
ExternalMasterPage	Boundary	A master page that creates a consistent layout for the external section of website.
Default	Boundary	This page enables the end user to search for gardens near an address and shows the results on the Google Maps.
GardenListPage	Boundary	This page shows a list of all the gardens in the database to the end user. In addition, the end user can download a report of gardens in PDF format through this page.
InternalMasterPage	Boundary	A master page that creates a consistent layout for the internal section of website.
LoginPage	Boundary	The database managers and the administrator can login to the system via this page.

GardenManagementPage	Boundary	The database manager can add, delete, or modify a garden through this page.
GenerateReportPage	Boundary	This page lets the database manager select a number of columns and export the information of the gardens to an excel spreadsheet.
UserManagementPage	Boundary	The administrator can manage the database managers using this page.
UserManagement-AddUserPage	Boundary	The administrator can enter the information of a new database manager in this page.
UserManagement-ChngPwdPage	Boundary	The administrator can change the password of a database manager in this page.

3.1.3 Garden Locator Classes

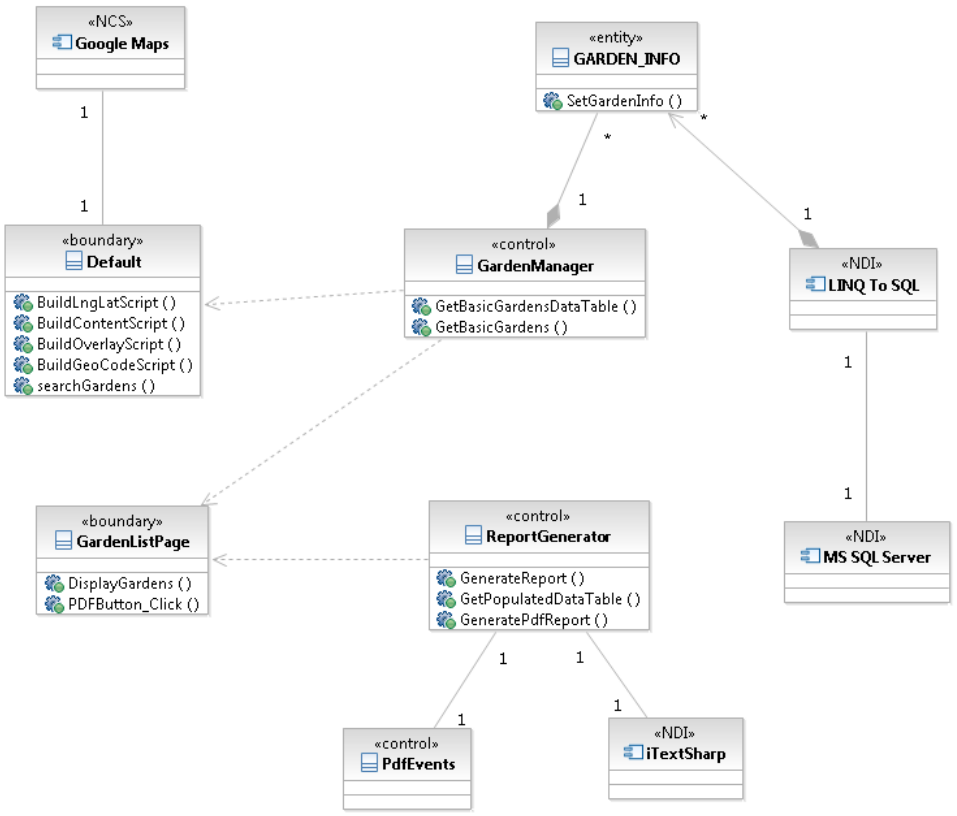

Figure 8: Garden Locator Class Diagram

Table 67: Garden Locator Class Description

Class	Type	Description

GARDEN_INFO	Entity	This class holds the information of a garden. This class is generated by LINQ to SQL.
Default	Boundary	This page enables the end user to search for gardens near an address and shows the results on the Google Maps.
GardenListPage	Boundary	This page lists all the gardens with limited information. In addition, the end user can generate a report of gardens in PDF format through this page.
GardenManager	Control	This class contains the business logic of selecting gardens from the database.
LINQ To SQL	NDI	A feature of .NET that facilitates connecting to the database and managing the data in the database; eliminates the need to write classes for connecting to the database from scratch.
Google Maps	NCS	A web service that is used by the GardenLocatorPage to display the gardens on the map of Los Angeles.
PdfEvents	Control	This class helps the ReportGenerator class to format the header and footer of the PDF file and add pictures to them.

3.1.4 Garden Management Classes

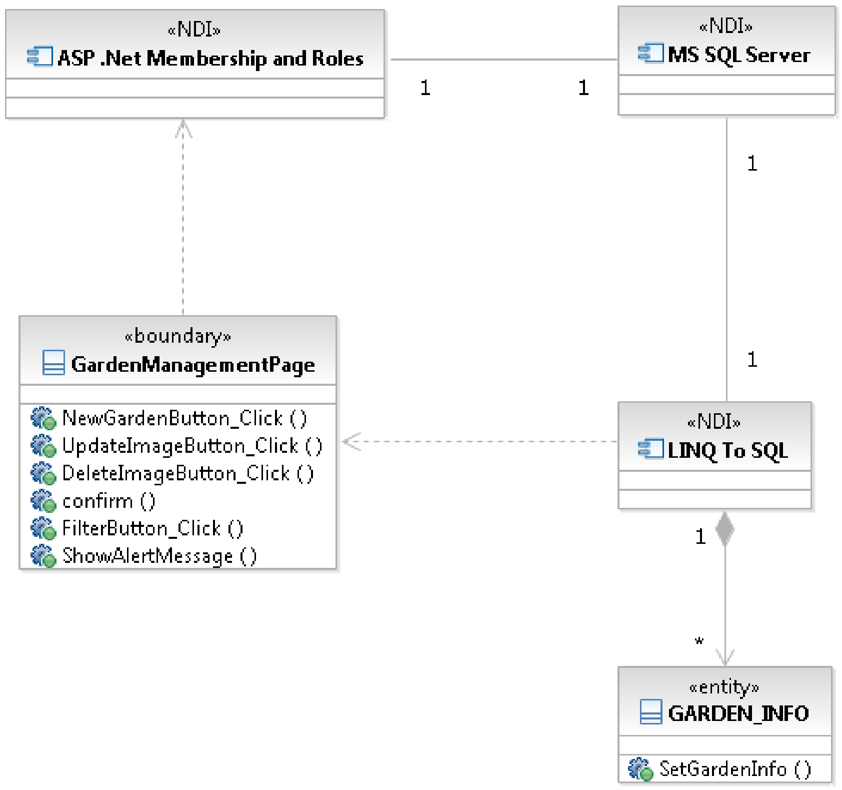

Figure 9: Garden Management Class Diagram

Table 68: Garden Management Class Description

Class	Type	Description
ASP .Net Membership and Roles	NDI	This component checks the user's authentication and authorization.
GardenManagementPage	Boundary	The database manager can add, delete, or modify a garden through this page.
LINQ To SQL	NDI	A feature of .NET that facilitates connecting to the database and managing the data in the database; eliminates the need to write classes for connecting to the database from scratch.
GARDEN_INFO	Entity	This class holds the information of a garden. This class is generated by LINQ to SQL.
MS SQL Server	NDI	A DBMS that stores all the information of gardens and users.

3.1.5 Internal Report Generation Classes

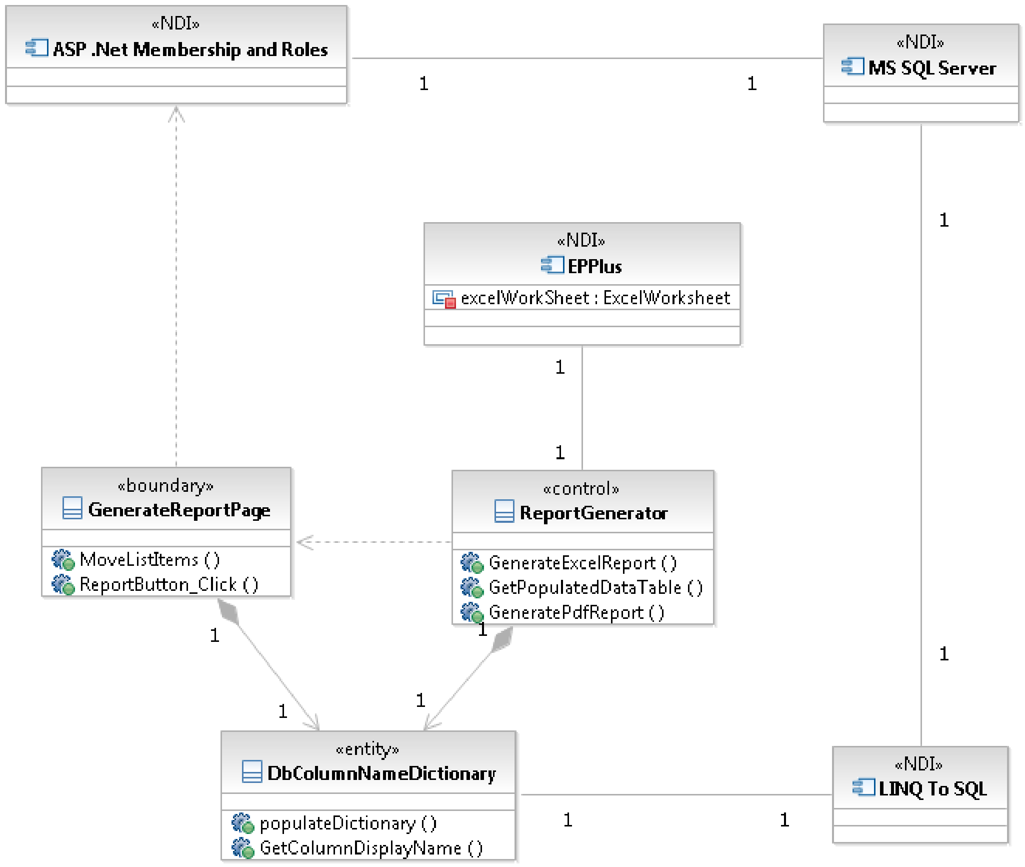

Figure 10: Internal Report Generation Class Diagram

Table 69: Internal Report Generation Class Description

Class	Type	Description
ASP .Net Membership and Roles	NDI	This component checks the user’s authentication and authorization.
GenerateReportPage	Boundary	This page lets the database manager select a number of columns and export the information of the gardens to an excel spreadsheet.
ReportGenerator	Control	This class contains the business logic of generating a report from the gardens in the database.
LINQ To SQL	NDI	A feature of .NET that facilitates connecting to the database and managing the data in the database; eliminates the need to write classes for connecting to the database from scratch.
EPPlus	NDI	This NDI is used to create reports in excel

		format from the gardens in the database.
MS SQL Server	NDI	A DBMS that stores all the information of gardens and users.
DbColumnNameDictionary	Entity	A class that stores the name of the columns of the garden table in a dictionary.

3.1.6 User Management Classes

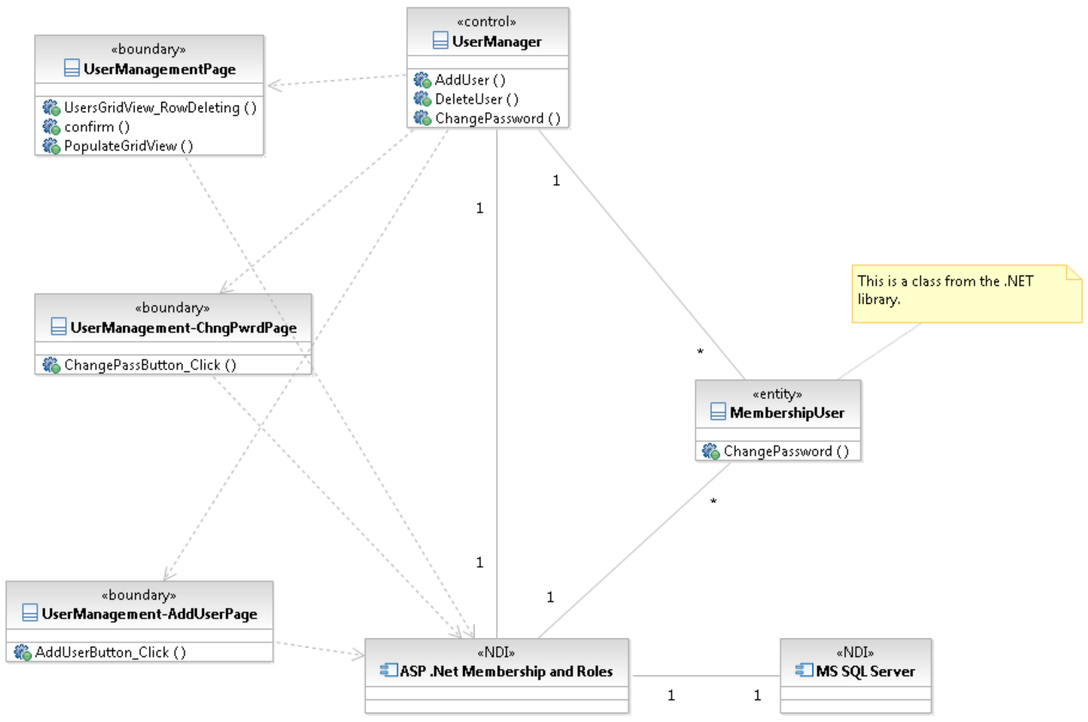

Figure 11: User Management Class Diagram

Table 70: User Management Class Description

Class	Type	Description
ASP .Net Membership and Roles	NDI	This component makes sure that the user is an administrator. In addition, it adds, updates, and deletes database managers.
MembershipUser	Entity	A .NET class holding the information of a user (administrator, database manager, or database viewer).
UserManagementPage	Boundary	The administrator can manage the database managers using this page.
UserManagement-AddUserPage	Boundary	The administrator can enter the information of a new database manager in this page.
UserManagement-ChngPwdPage	Boundary	The administrator can change the password of a database manager in this page.

UserManager	Control	This class contains the business logic of adding, updating, and deleting database managers.
MS SQL Server	NDI	A DBMS that stores all the information of gardens and users.

3.1.7 Process Realization

This section shows how the use cases described in section 2.1.3 are going to be realized using sequence diagrams. All the sequence diagrams can also be found in the RAS file available on the team website.

Since the beginning of the implementation, the team decided to use the LINQ To SQL feature of the .Net framework. This decision led to most of the sequence diagram becoming simpler because LINQ To SQL handles the interactions with the database itself.

The sequence diagram in Figure 12 depicts how the end user can search for gardens using Google Maps.

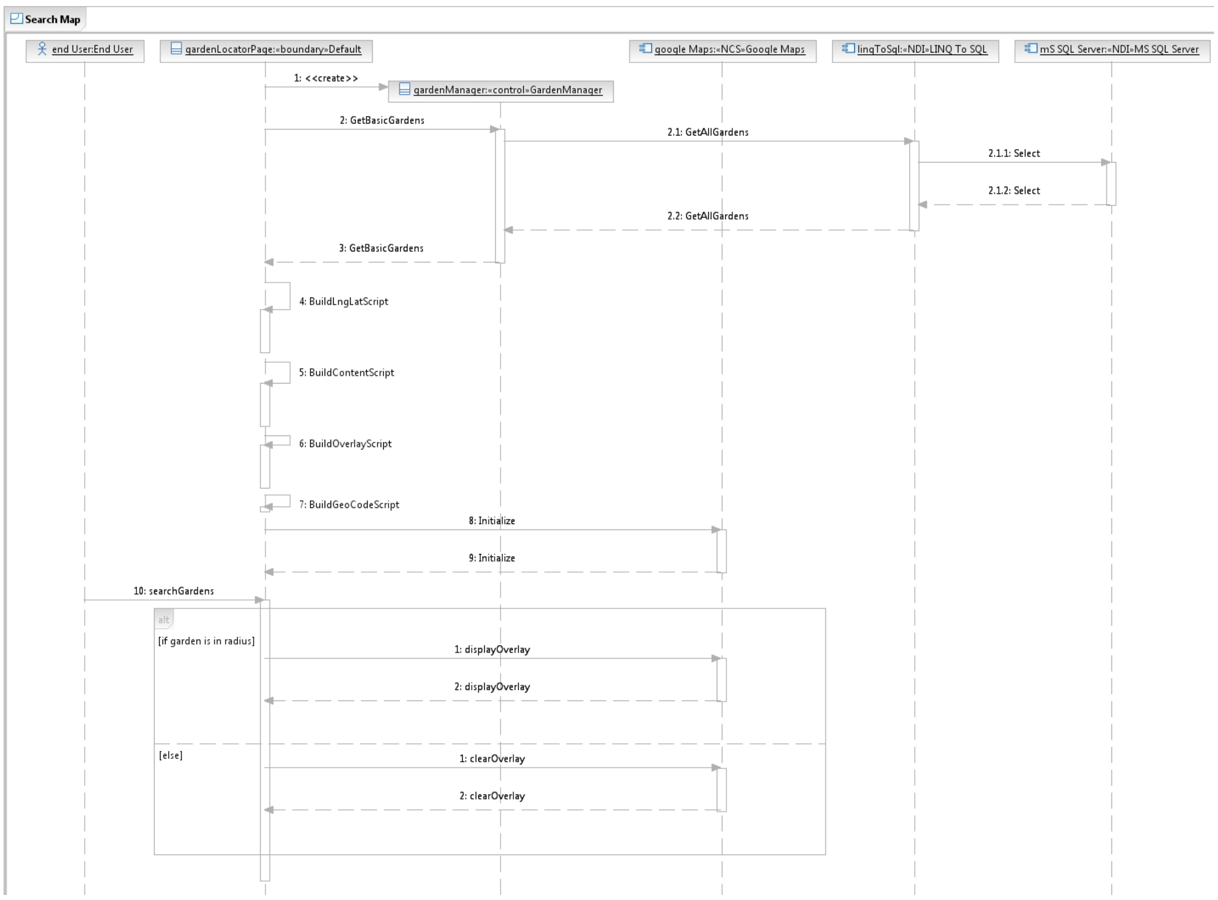

Figure 12: Search Map Sequence Diagram

The sequence diagram in Figure 13 shows how the end user can see a list of all the gardens in the database.

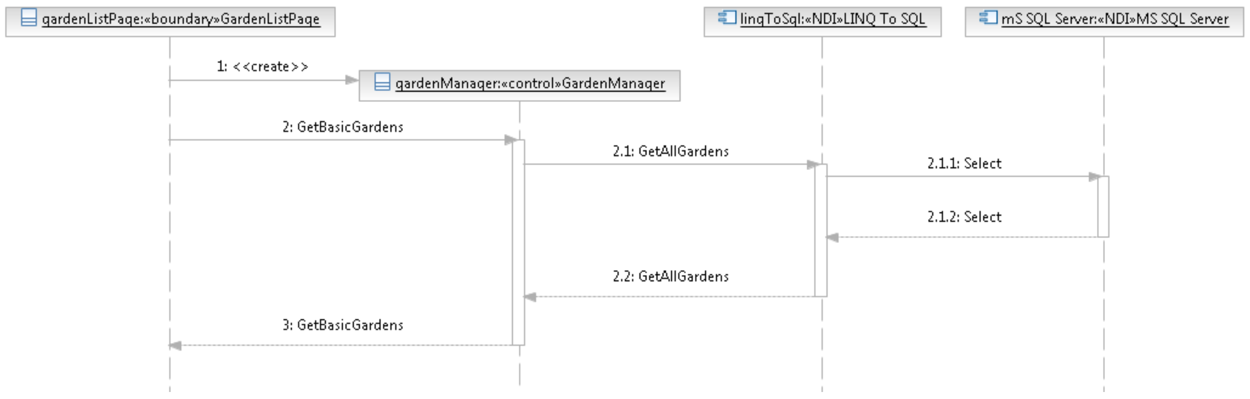
gardenListPage:«boundary»GardenListPage
linqToSql:«NDI»LINQ To SQL
mS SQL Server:«NDI»MS SQL Server
1: <<create>>
gardenManager:«control»GardenManager
2: GetBasicGardens
2.1: GetAllGardens
2.1.1: Select
2.1.2: Select
2.2: GetAllGardens
3: GetBasicGardens

Figure 13: View All Gardens Sequence Diagram

The sequence diagram in Figure 14 shows how the database manager can add a garden to the database.

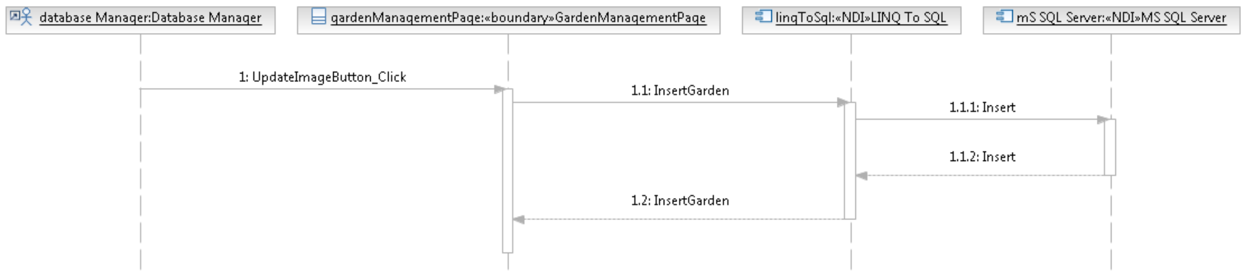
database Manager:Database Manager
gardenManagementPage:«boundary»GardenManagementPage
linqToSql:«NDI»LINQ To SQL
mS SQL Server:«NDI»MS SQL Server
1: UpdateImageButton_Click
1.1: InsertGarden
1.1.1: Insert
1.1.2: Insert
1.2: InsertGarden

Figure 14: Add Garden Sequence Diagram

The sequence diagram in Figure 15 shows how the database manager can delete a garden from the database.

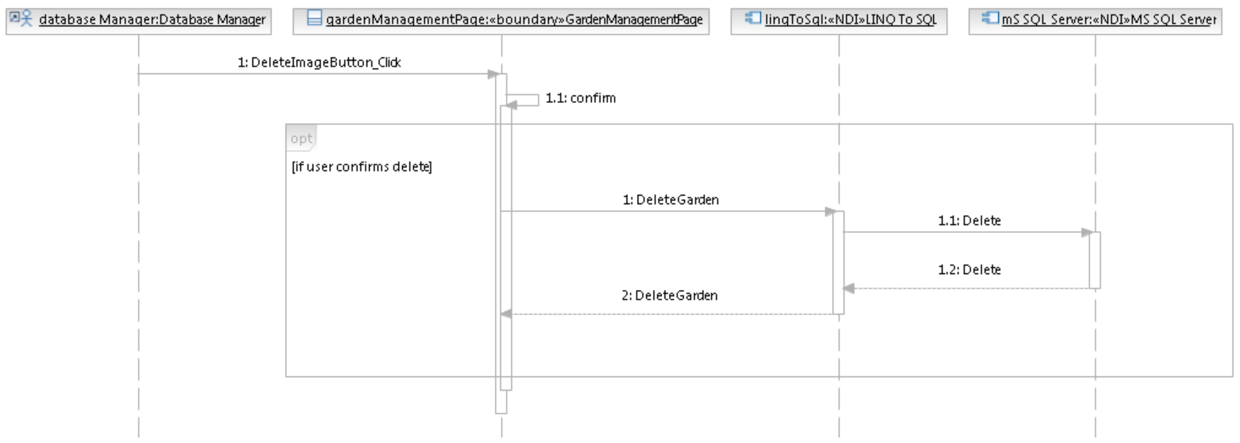
database Manager:Database Manager
gardenManagementPage:«boundary»GardenManagementPage
linqToSql:«NDI»LINQ To SQL
mS SQL Server:«NDI»MS SQL Server
1: DeleteImageButton_Click
1.1: confirm
opt
[if user confirms delete]
1: DeleteGarden
1.1: Delete
1.2: Delete
2: DeleteGarden

Figure 15: Delete Garden Sequence Diagram

The next sequence diagram shows how the database manager can modify the information of a garden. This diagram is shown in figure 16.

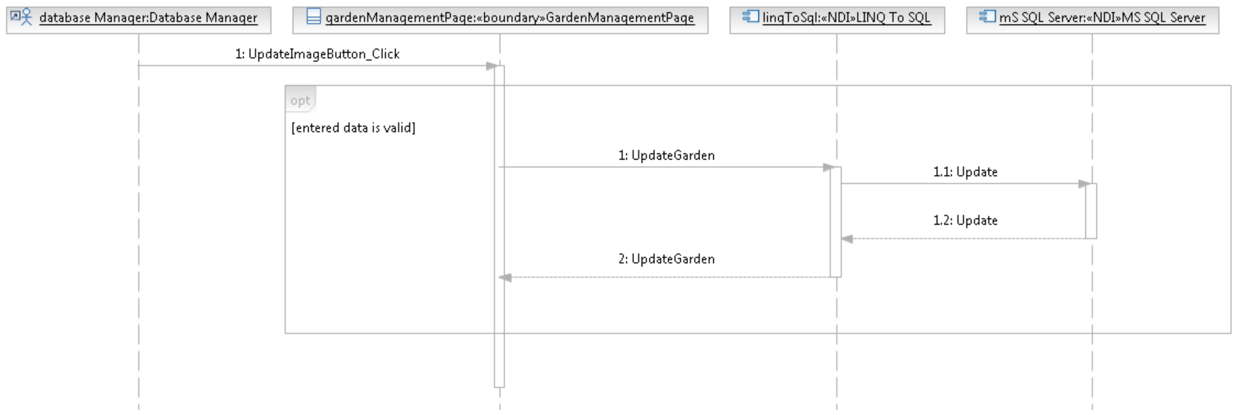
database Manager:Database Manager
gardenManagementPage:«boundary»GardenManagementPage
linqToSql:«NDI»LINQ To SQL
mS SQL Server:«NDI»MS SQL Server
1: UpdateImageButton_Click
opt
[entered data is valid]
1: UpdateGarden
1.1: Update
1.2: Update
2: UpdateGarden

Figure 16: Modify Garden Sequence Diagram

The sequence diagram in Figure 17 shows how the database manager can search the garden table for a particular keyword.

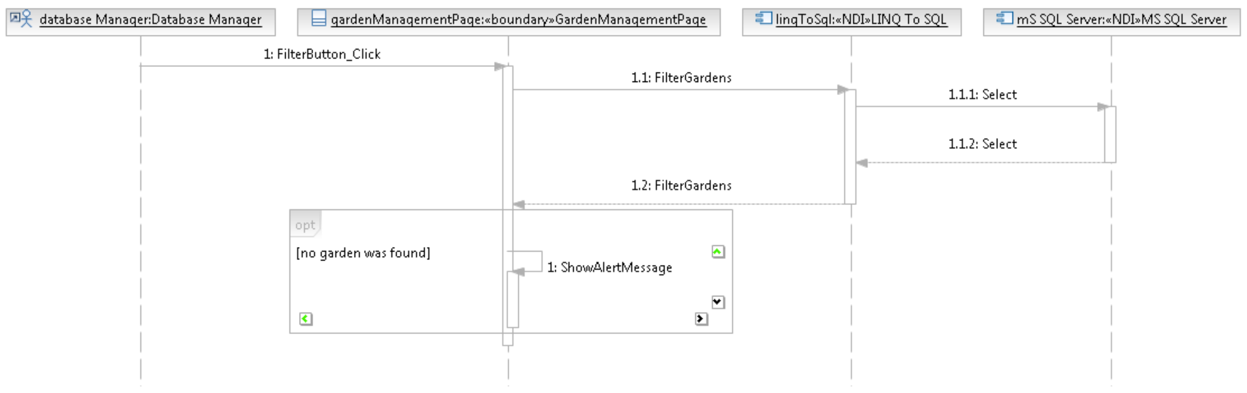
database Manager:Database Manager
gardenManagementPage:«boundary»GardenManagementPage
linqToSql:«NDI»LINQ To SQL
mS SQL Server:«NDI»MS SQL Server
1: FilterButton_Click
1.1: FilterGardens
1.1.1: Select
1.1.2: Select
1.2: FilterGardens
opt
[no garden was found]
1: ShowAlertMessage

Figure 17: Search Garden Table Sequence Diagram

The sequence diagram in Figure 18 shows how the end user can download a report file of all the gardens with a limited number of columns.

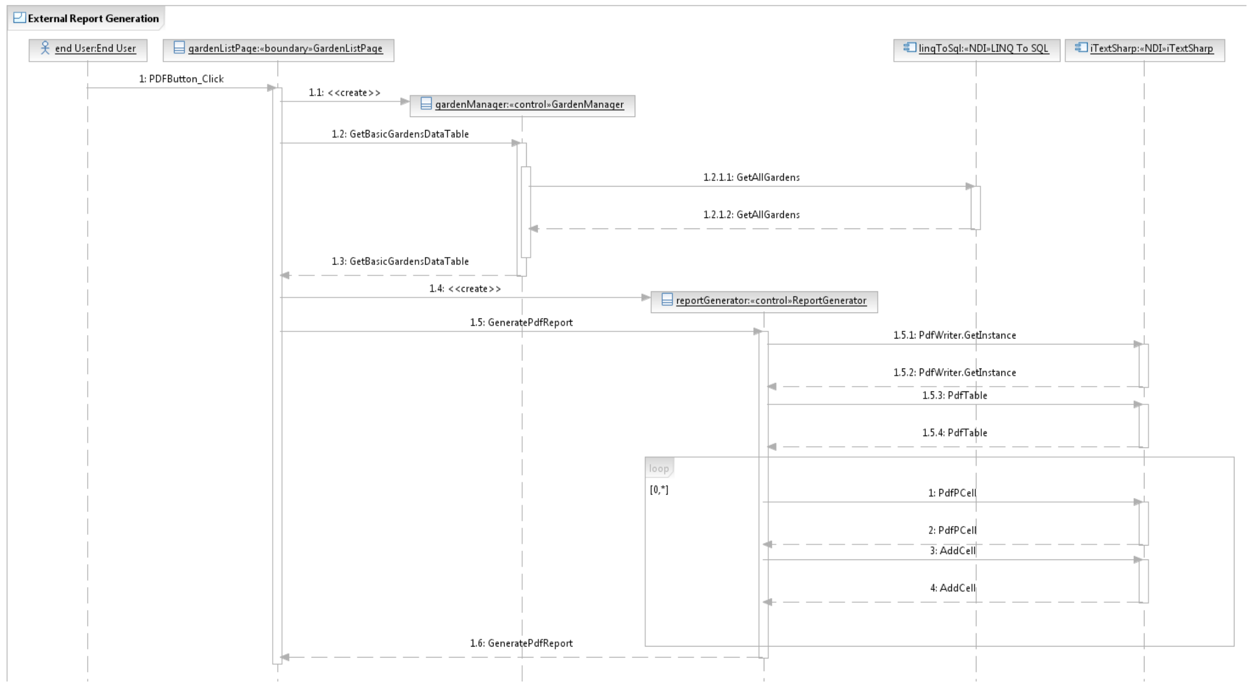
External Report Generation
end User:End User
gardenListPage:«boundary»GardenListPage
linqToSql:«NDI»LINQ To SQL
iTextSharp:«NDI»iTextSharp
1: PDFButton_Click
1.1: <<create>>
gardenManager:«control»GardenManager
1.2: GetBasicGardensDataTable
1.2.1.1: GetAllGardens
1.2.1.2: GetAllGardens
1.3: GetBasicGardensDataTable
1.4: <<create>>
reportGenerator:«control»ReportGenerator
1.5: GeneratePdfReport
1.5.1: PdfWriter.GetInstance
1.5.2: PdfWriter.GetInstance
1.5.3: PdfTable
1.5.4: PdfTable
loop
[0,*]
1: PdfPCell
2: PdfPCell
3: AddCell
4: AddCell
1.6: GeneratePdfReport

Figure 18: External Report Generation Sequence Diagram

The sequence diagram in Figure 19 shows how the database manager can download a report file of all the gardens with all the columns.

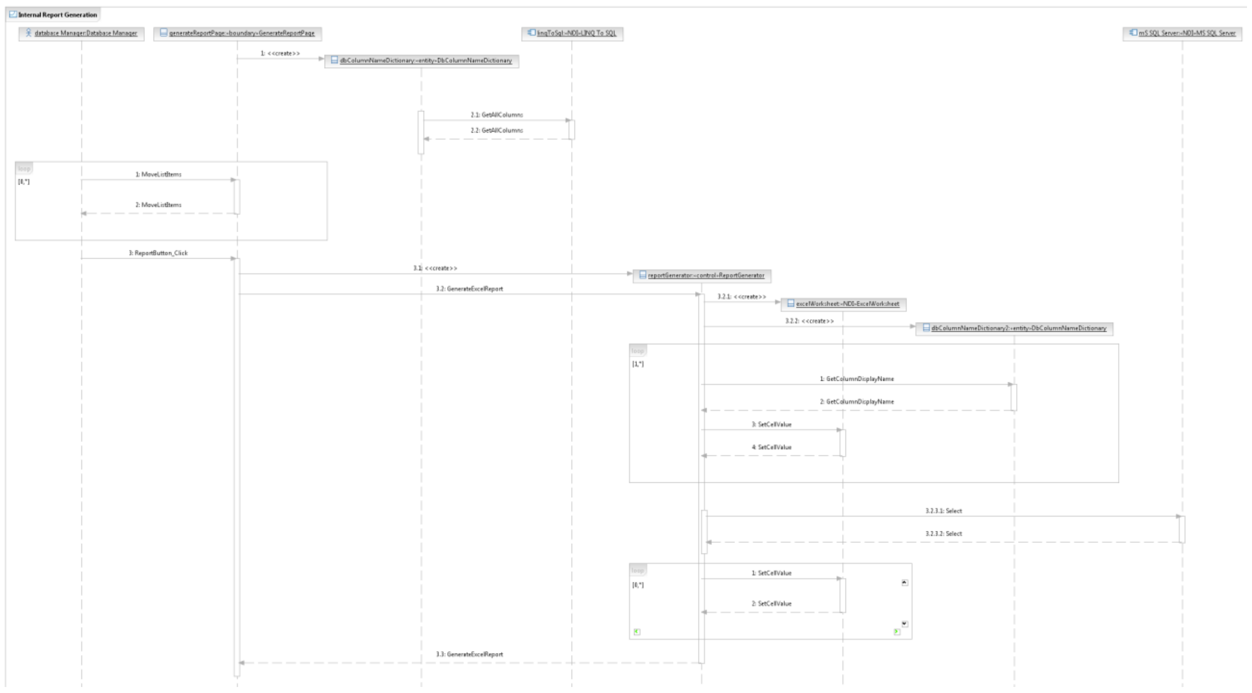

Figure 19: Internal Report Generation Sequence Diagram

The sequence diagrams in Figures 20, 21, and 22 show how the administrator can add, delete, and modify database managers.

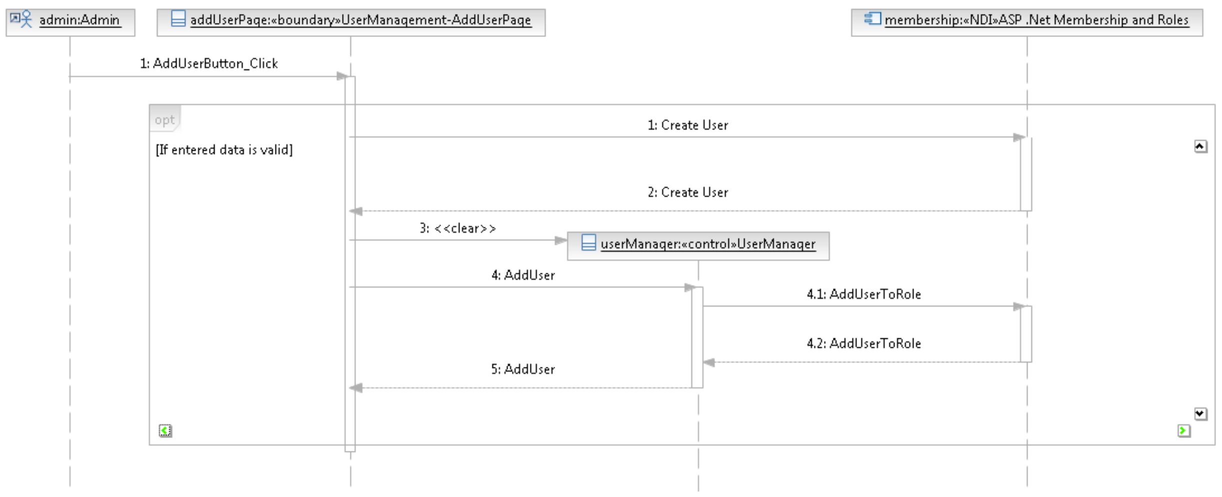
admin:Admin
addUserPage:«boundary»UserManagement-AddUserPage
membership:«NDI»ASP .Net Membership and Roles
1: AddUserButton_Click
opt
[If entered data is valid]
1: Create User
2: Create User
3: <<clear>>
userManager:«control»UserManager
4: AddUser
4.1: AddUserToRole
4.2: AddUserToRole
5: AddUser

Figure 20: Add Database Manager Sequence Diagram

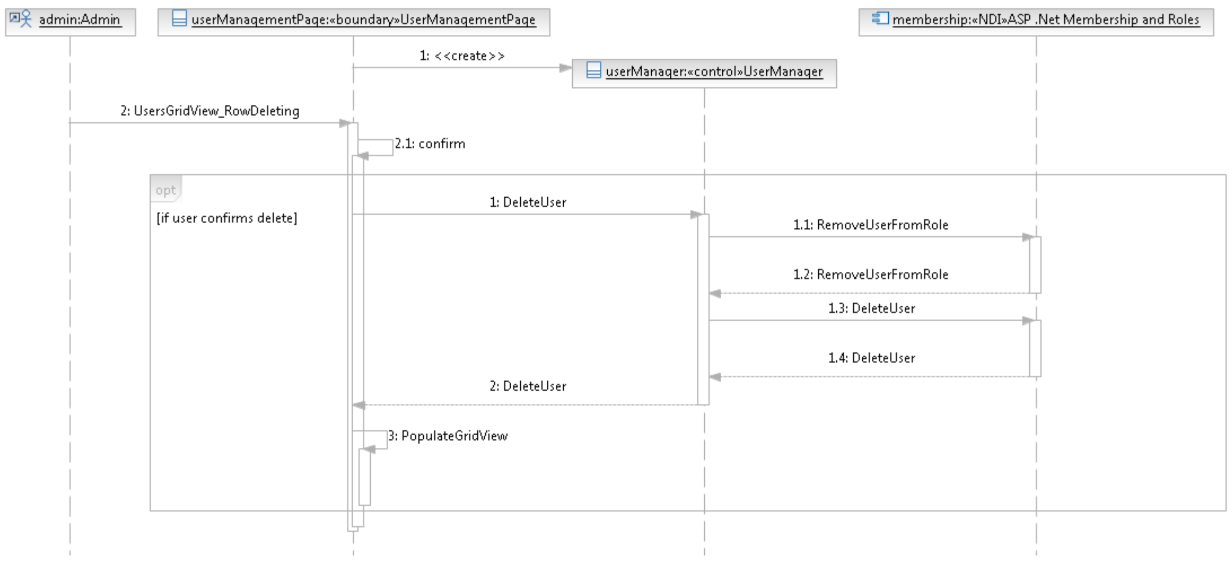
admin:Admin
userManagementPage:«boundary»UserManagementPage
membership:«NDI»ASP .Net Membership and Roles
1: <<create>>
userManager:«control»UserManager
2: UsersGridView_RowDeleting
2.1: confirm
opt
[if user confirms delete]
1: DeleteUser
1.1: RemoveUserFromRole
1.2: RemoveUserFromRole
1.3: DeleteUser
1.4: DeleteUser
2: DeleteUser
3: PopulateGridView

Figure 21: Delete Database Manager Sequence Diagram

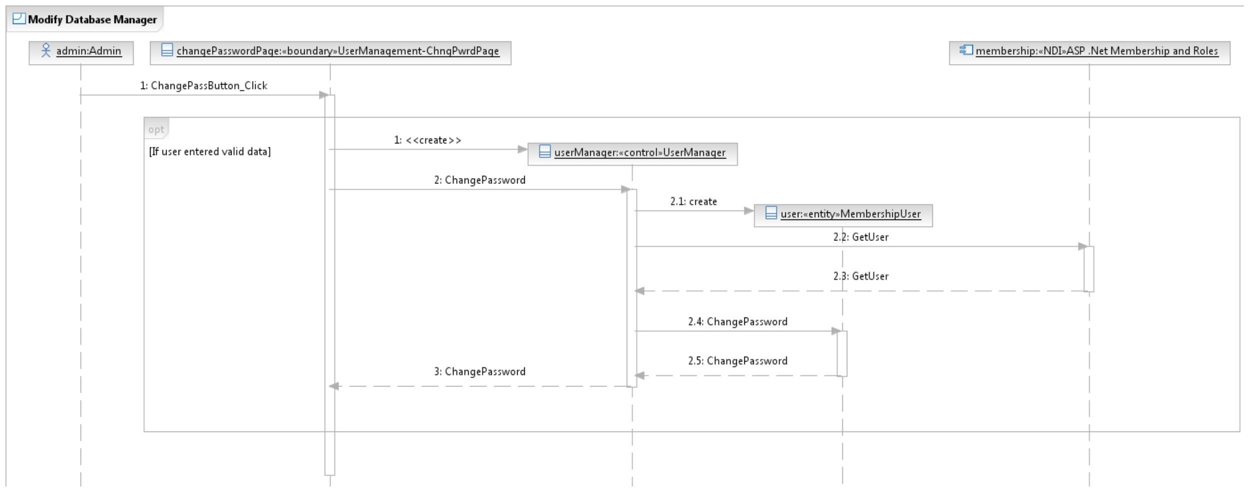
Modify Database Manager
admin:Admin
changePasswordPage:«boundary»UserManagement-ChngPwrdPage
membership:«NDI»ASP .Net Membership and Roles
1: ChangePassButton_Click
opt
[If user entered valid data]
1: <<create>>
userManager:«control»UserManager
2: ChangePassword
2.1: create
user:«entity»MembershipUser
2.2: GetUser
2.3: GetUser
2.4: ChangePassword
2.5: ChangePassword
3: ChangePassword

Figure 22: Modify Database Manager Sequence Diagram

3.2 Design Rationale

As depicted in Figure 5, the Garden Inventory and Locator website will have a 3-tier architecture. This architectural style makes future changes to the system easier. The three tiers, or layers, of this website are listed below:

- Presentation Tier
- Business Logic Tier
- Data Access Tier

The description of each tier can be found in Table 46.

Microsoft SQL Server will be used as the DBMS because it is the most convenient DBMS to use with ADO .NET classes; they are both manufactured by the same company and are very compatible with each other. In addition, LINQ To SQL will be used to create the necessary classes for the data access layer.

Instead of coding the user management functionality from scratch, we decided to utilize the ASP .NET Membership and Roles feature. This service of ASP .NET automatically creates database tables for storing usernames and passwords, hashes passwords for higher security, and checks users for both authentication and authorization.

4. Architectural Styles, Patterns and Frameworks

Table 71: Architectural Styles, Patterns, & Framework

Name	Description	Benefits, Costs, and Limitations
3-Tier Architecture	The 3-tier architecture separates the application into 3 different layers: presentation, business logic, and data access. Each layer can only communicate with the layer immediately above or below itself.	This architecture is an excellent example of separation of concerns principle: each layer is independent of the other. Therefore, any internal changes to the classes of a layer does not affect other layers. As a result, a layer can be completely replaced with a different set of classes if they have the same interface; for instance, the system can have multiple presentation layers. The main disadvantage of this architecture is the increase in the size of the project.

Note: We tried to make the implementation comply with the 3-tier architecture as much as possible. However, because of using LINQ To SQL for facilitating the data access layer, in a few classes, there is direct connection between the presentation layer and the data access layer.