Technical Manual

Crystal Cockpit

University Of Wollongong

CSCI321-JG1

Members: Nan Noon Noon, Shan Mey Yew, Andre Khoury and Behzad Garekani

Date: 29-October-2013

Revision Sheet

Release No.	Date	Revision Description
Rev. 0	20/05/2013	Plan and added draft information based on week 6 research
Rev. 1	31/05/2013	Added design documentation
Rev. 2	05/06/2013	Report design and make small changes on design and plan.
Rev.4	01/10/2013	Update the uml diagram to match with current program
Rev 5	20/10/2013	Change the information to match with current programs
Rev 6	29/10/2013	Finalize and add more features.

Contents

1 Introduction

This document provides the detail plan of the project, detail design based on functionality, other technical part of the project and how to implement the project. In document show the designs of the project such as project breakdown diagram, activity network, use cases, sequences, and architecture design and deployment diagrams. Each diagram represents the flow of the project and how we will go to implement on project.

Moreover, this document shows how to control the project (project management) such as project plan (Gantt chart) and milestone and risk management. These plans are helpful when members encounter the problem and provide idea how we can solve those problems. In additional, technical manual stated product breakdown structure, the purpose of this diagram is if developers want make maintenance or want to make some changes on some part of the project, they can follow the structure and easy to maintain the related part of that project.

2 System Overview

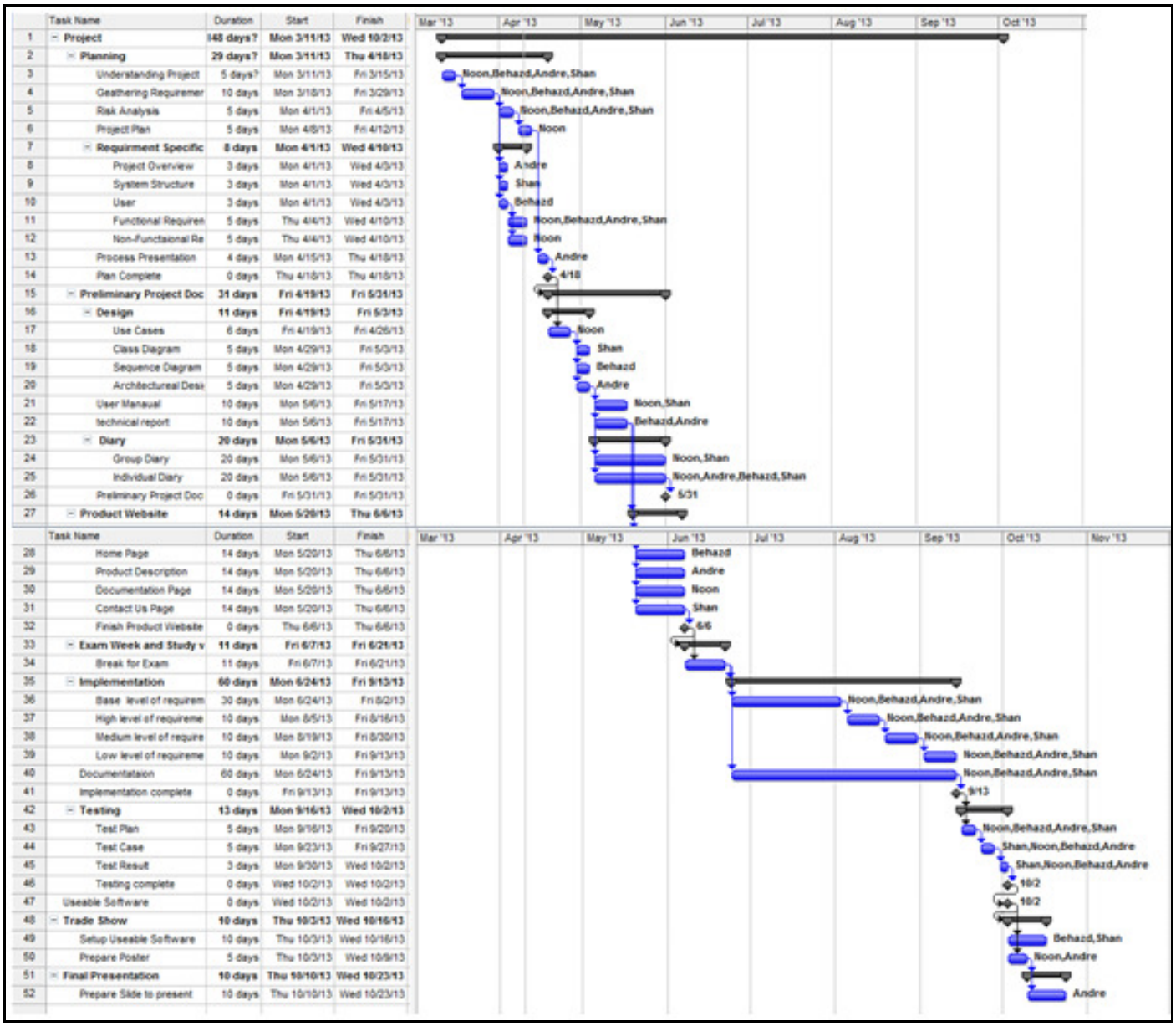
This project provide user friendly interface to maintain the database easily. The application name is called Crystal Cockpit. This is reflected with our statement. The mission of the project was providing nice and clean interface to control complex or sample database. The goal of the project was delivery high quality software to user with user-friendly interface which will allow for direct create, delete, migration and defragmentation on logical data objects. In additional Crystal Cockpit provide user control level to admin, to create and delete users, assign roles and grand privileges.

3 System Development Plan

3.1 Major Milestones

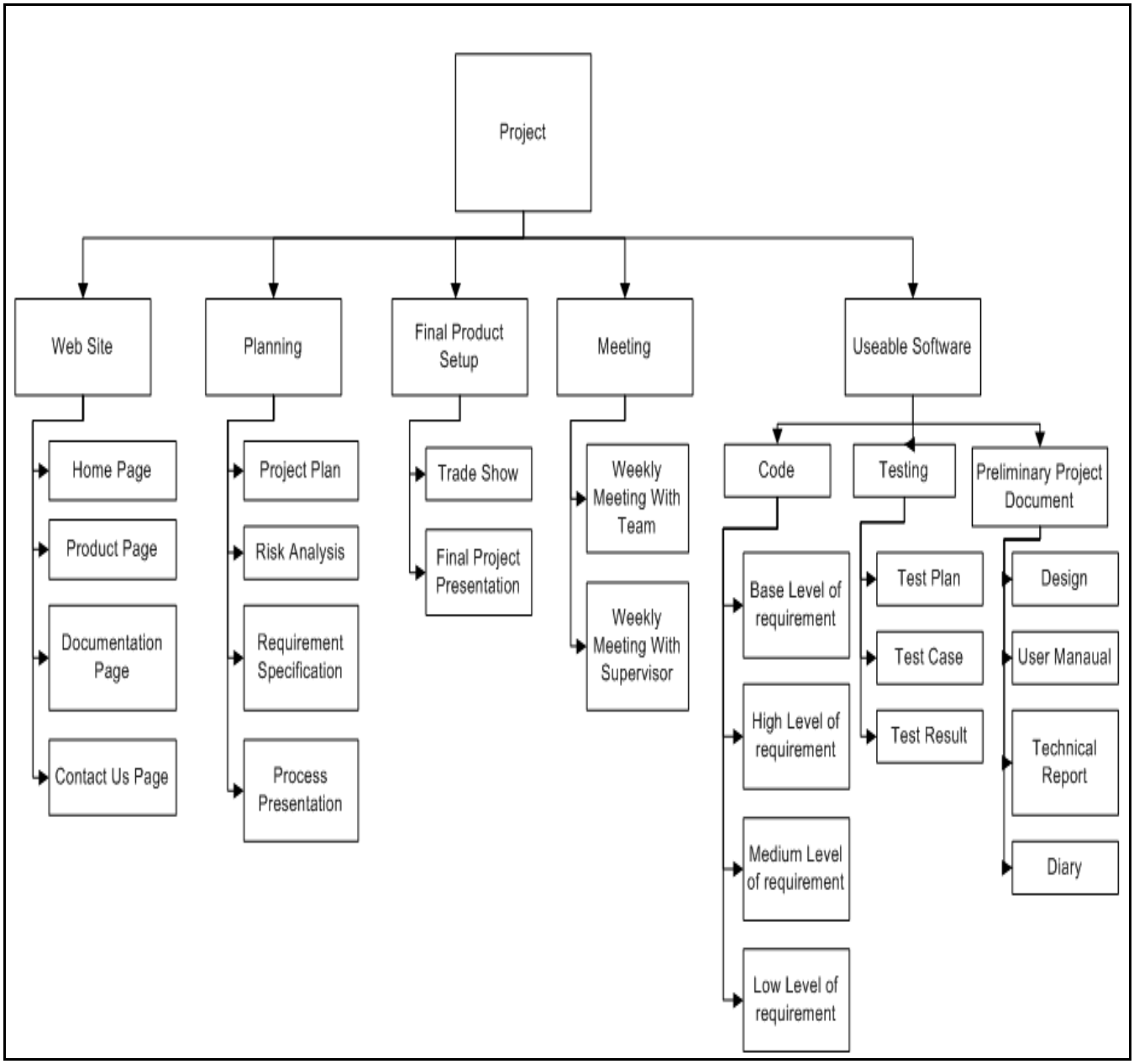
Milestone	Milestone description	Dependency	Resources	Due Date
1	Requirements Analysis	0	4	18 April 2013
2	Preliminary Project Documentation	1	2	31 May 2013
3	Draft User Manual	1	1	31 May 2013
3	Product Website	1,2,3	2	6 June 2013
4	Implementation Base level of requirement	3	3	2 Aug 2013
5	Implementation High level of requirement	4	3	16 Aug 2013
6	Implementation Medium level of requirement	5	3	30 Aug 2013
7	Implementation Low Level of requirement	6	3	13 Sep 2013
8	Testing	4,5,6,7	2	25 Sep 2013
9	Delivery whole project	8	4	30 Oct 2013

3.2 Project Plan (Grantt Chart)

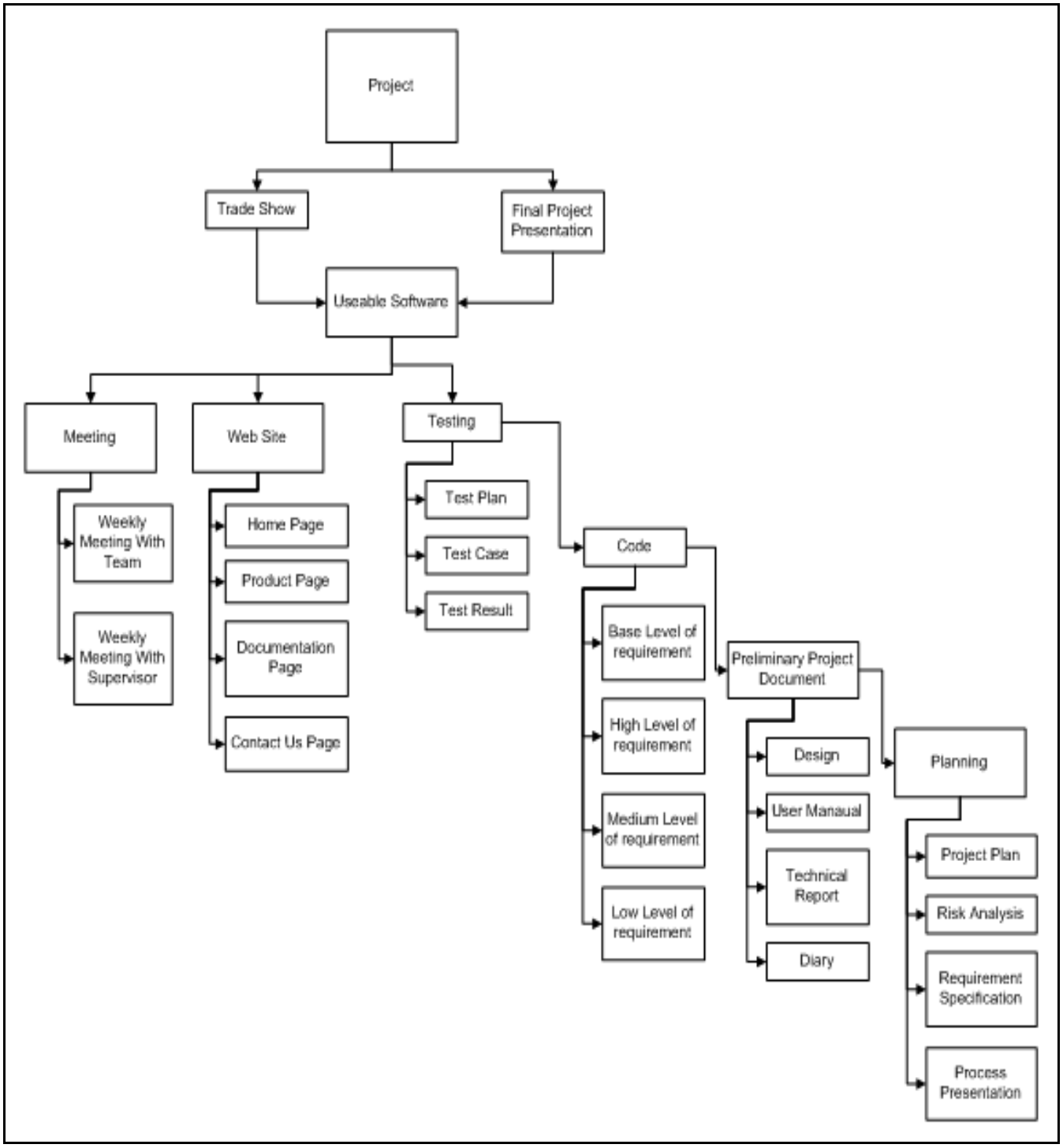


	Task Name	Duration	Start	Finish
1	Project	148 days?	Mon 3/11/13	Wed 10/2/13
2	Planning	29 days?	Mon 3/11/13	Thu 4/18/13
3	Understanding Project	5 days?	Mon 3/11/13	Fri 3/15/13
4	Geathering Requiremer	10 days	Mon 3/18/13	Fri 3/29/13
5	Risk Analysis	5 days	Mon 4/1/13	Fri 4/5/13
6	Project Plan	5 days	Mon 4/8/13	Fri 4/12/13
7	Requirment Specific	8 days	Mon 4/1/13	Wed 4/10/13
8	Project Overview	3 days	Mon 4/1/13	Wed 4/3/13
9	System Structure	3 days	Mon 4/1/13	Wed 4/3/13
10	User	3 days	Mon 4/1/13	Wed 4/3/13
11	Functional Requiren	5 days	Thu 4/4/13	Wed 4/10/13
12	Non-Functaional Re	5 days	Thu 4/4/13	Wed 4/10/13
13	Process Presentation	4 days	Mon 4/15/13	Thu 4/18/13
14	Plan Complete	0 days	Thu 4/18/13	Thu 4/18/13
15	Preliminary Project Doc	31 days	Fri 4/19/13	Fri 5/31/13
16	Design	11 days	Fri 4/19/13	Fri 5/3/13
17	Use Cases	6 days	Fri 4/19/13	Fri 4/26/13
18	Class Diagram	5 days	Mon 4/29/13	Fri 5/3/13
19	Sequence Diagram	5 days	Mon 4/29/13	Fri 5/3/13
20	Architectureal Desi	5 days	Mon 4/29/13	Fri 5/3/13
21	User Manaual	10 days	Mon 5/6/13	Fri 5/17/13
22	technical report	10 days	Mon 5/6/13	Fri 5/17/13
23	Diary	20 days	Mon 5/6/13	Fri 5/31/13
24	Group Diary	20 days	Mon 5/6/13	Fri 5/31/13
25	Individual Diary	20 days	Mon 5/6/13	Fri 5/31/13
26	Preliminary Project Doc	0 days	Fri 5/31/13	Fri 5/31/13
27	Product Website	14 days	Mon 5/20/13	Thu 6/6/13
28	Home Page	14 days	Mon 5/20/13	Thu 6/6/13
29	Product Description	14 days	Mon 5/20/13	Thu 6/6/13
30	Documentation Page	14 days	Mon 5/20/13	Thu 6/6/13
31	Contact Us Page	14 days	Mon 5/20/13	Thu 6/6/13
32	Finish Product Website	0 days	Thu 6/6/13	Thu 6/6/13
33	Exam Week and Study v	11 days	Fri 6/7/13	Fri 6/21/13
34	Break for Exam	11 days	Fri 6/7/13	Fri 6/21/13
35	Implementation	60 days	Mon 6/24/13	Fri 9/13/13
36	Base level of requirem	30 days	Mon 6/24/13	Fri 8/2/13
37	High level of requireme	10 days	Mon 8/5/13	Fri 8/16/13
38	Medium level of require	10 days	Mon 8/19/13	Fri 8/30/13
39	Low level of requireme	10 days	Mon 9/2/13	Fri 9/13/13
40	Documentataion	60 days	Mon 6/24/13	Fri 9/13/13
41	Implementation complete	0 days	Fri 9/13/13	Fri 9/13/13
42	Testing	13 days	Mon 9/16/13	Wed 10/2/13
43	Test Plan	5 days	Mon 9/16/13	Fri 9/20/13
44	Test Case	5 days	Mon 9/23/13	Fri 9/27/13
45	Test Result	3 days	Mon 9/30/13	Wed 10/2/13
46	Testing complete	0 days	Wed 10/2/13	Wed 10/2/13
47	Useable Software	0 days	Wed 10/2/13	Wed 10/2/13
48	Trade Show	10 days	Thu 10/3/13	Wed 10/16/13
49	Setup Useable Software	10 days	Thu 10/3/13	Wed 10/16/13
50	Prepare Poster	5 days	Thu 10/3/13	Wed 10/9/13
51	Final Presentation	10 days	Thu 10/10/13	Wed 10/23/13
52	Prepare Slide to present	10 days	Thu 10/10/13	Wed 10/23/13

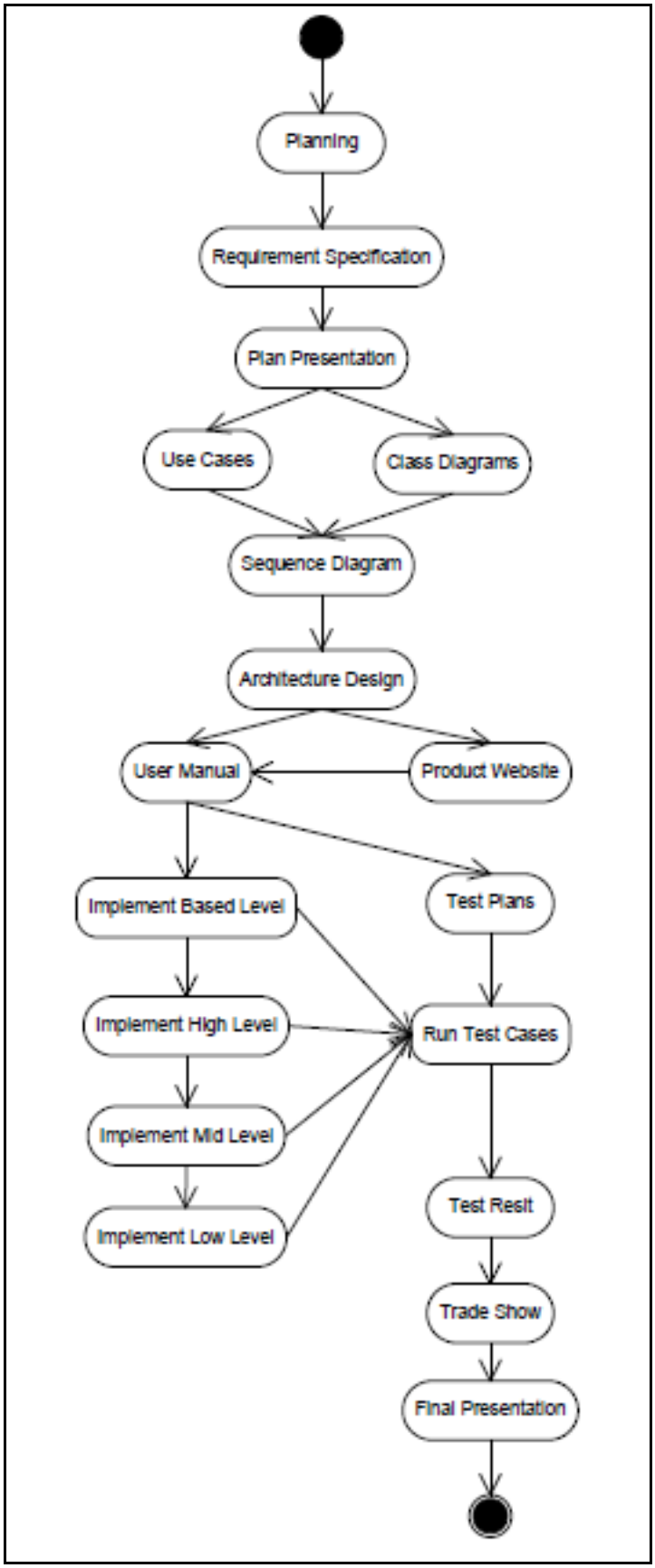
3.3 Project High Level Work Breakdown Structure



3.4 Product Flow Diagram

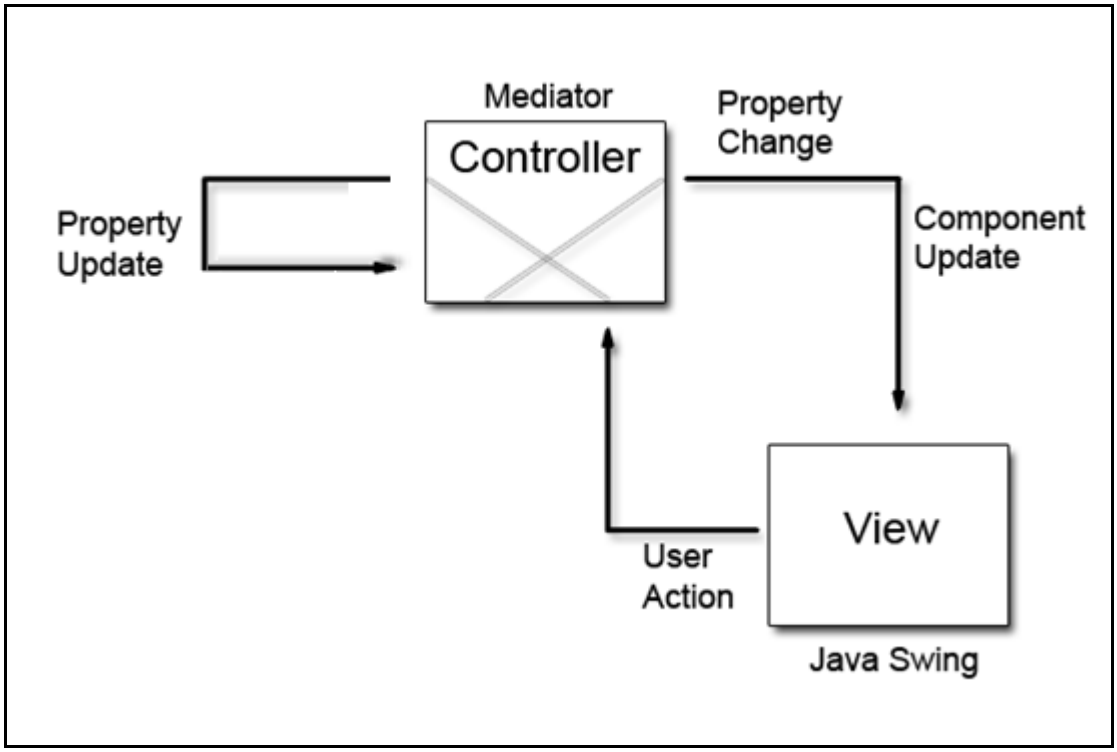


3.5 Activity Network Diagram



4 Procedures and Design

4.1 Technology Architectural Design



A controller is sending commands to the view to the user's recent changes such as scrolling and also it is connected to database to update the information.

A controller informs view if there are any changes in its state. As we can see in the picture Controller send messages to View if there is any changes requested by the user.

A view is connected to the controller to generate the data to the user. Most of the GUI classes contain in this part.

4.1.1 Technology Specification

Database	Oracle 11 g r2 and MySQL
Technology	Java and JDBC
Client technology	JRE 7.1
Development Environment	Window 7
Test Environment	Window
Platform Environment	Window

4.2 Design Aspects

This project needed user-friendly interface design. Therefore, developers are using Java to create GUI and allow user to login for security purpose to product their database from someone make changes.

Below (Figure 1) is user login design and it providing user to login with their username and password and database location. If username and password is incorrect system will show error message and will allow user to login again.

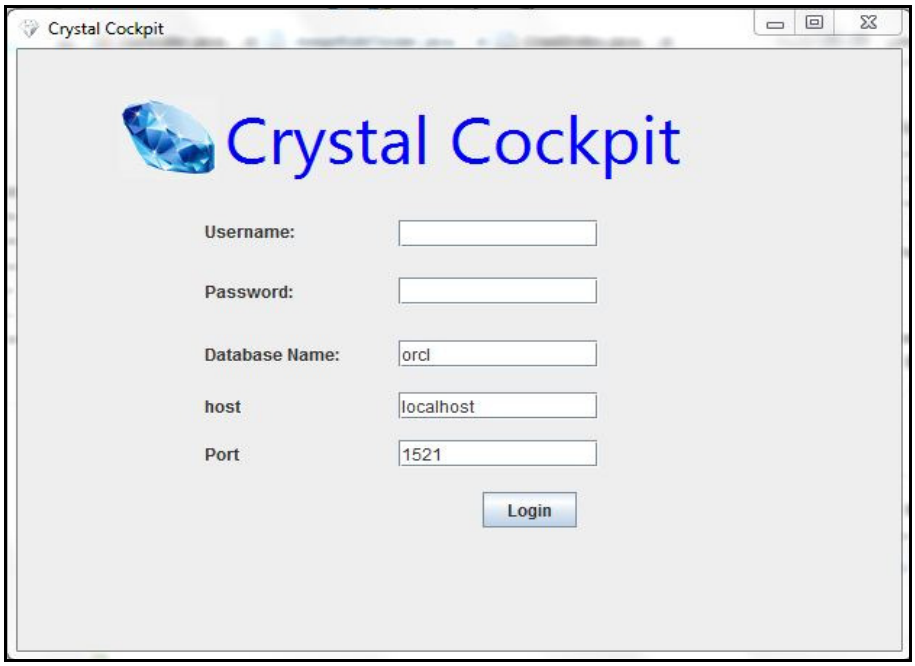


Figure 1: Login

Below (Figure 2) is main frame that user can see after they login. And its provide functions on the menus bar. These are File, Create, Delete, Defragmentation, Migration, User Control and Help. Each tab menus have their own sub-functions. At the left pane of the main frame show the data structure and right panel show the preview of the selected structure from menu bar. Besides that, in the centre of main frame, the text box on top is let user to execute the query while the table below will list the history of executed query by user. There are few functions we are added to let user clear the script box, save script, load script from local drive/ temp drive.

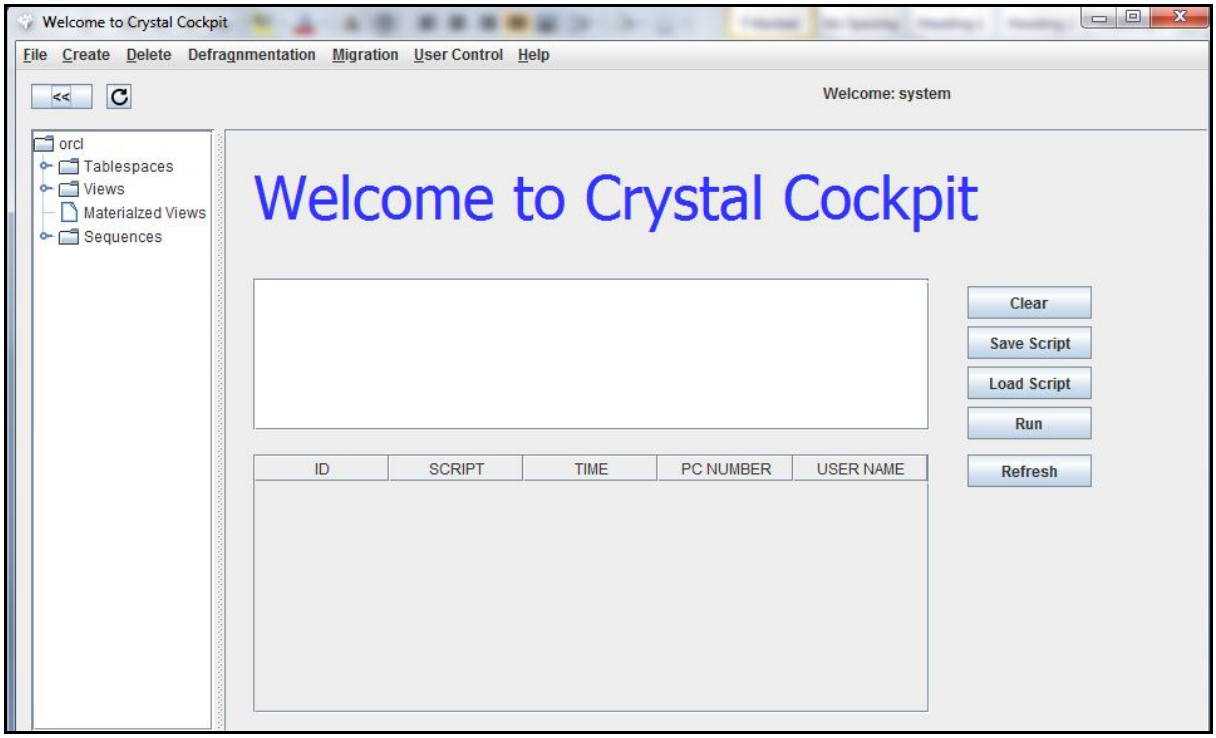
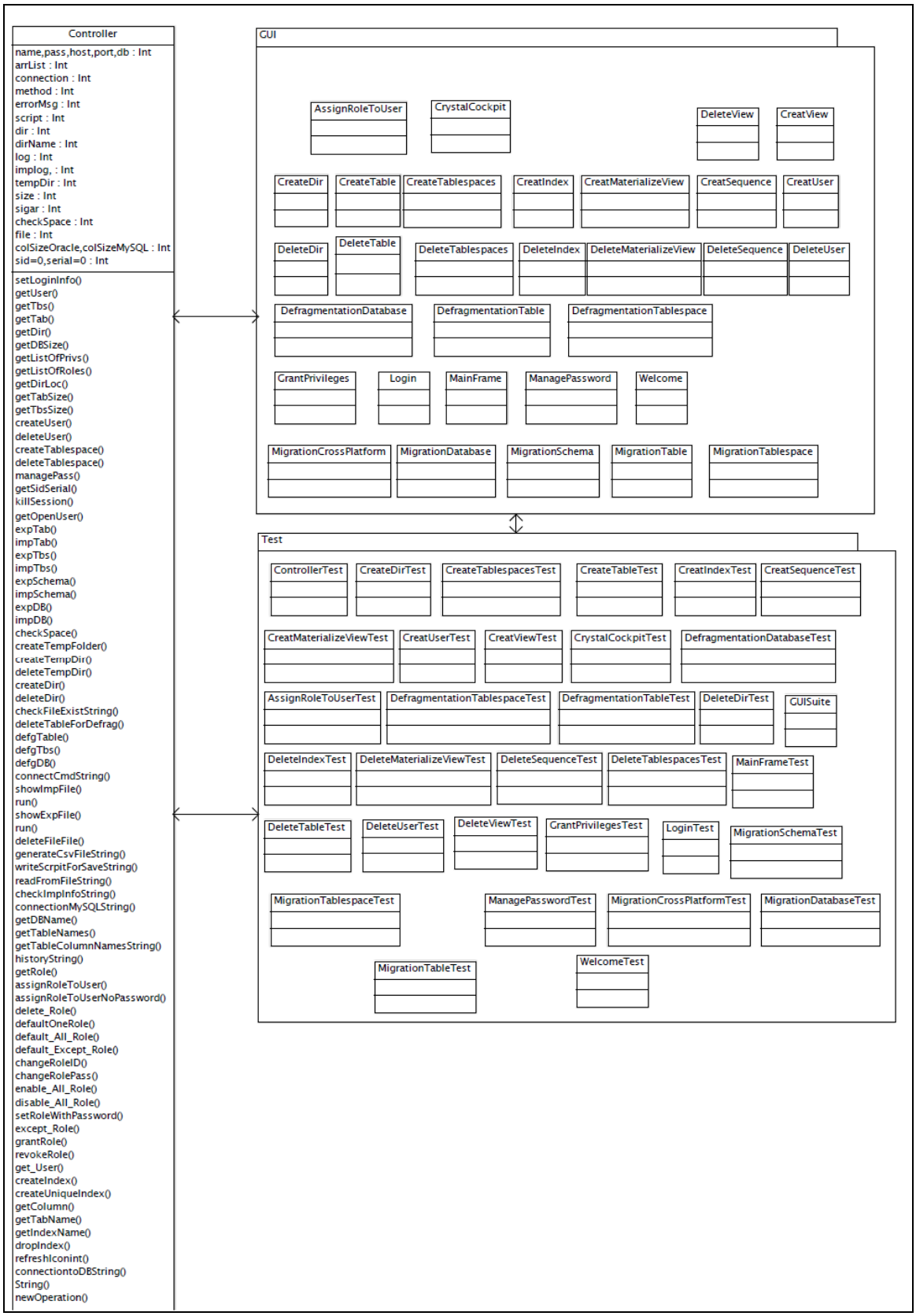


Figure 2: Welcome/History Page

4.3 UML Model

4.3.1 Class Diagram



Description

In Classs Diagram, Controller class is the main class to connect to database. In Crystal Cockpit, developers do not need to record information such as database name because data is recorded in database. However, to increase speed and avoid redundant usage of database, some information will be saved temporary.

All the GUI classes are independent. Changing in one class does not affect other classes. They are connected to the controller directly. Controller is responsible to open connection for each GUI. GUI can invoke functions in controller and update their content or only ask for connection to the database and then manipulate data from database directly. All the user interface classes are existed inside GUI package.

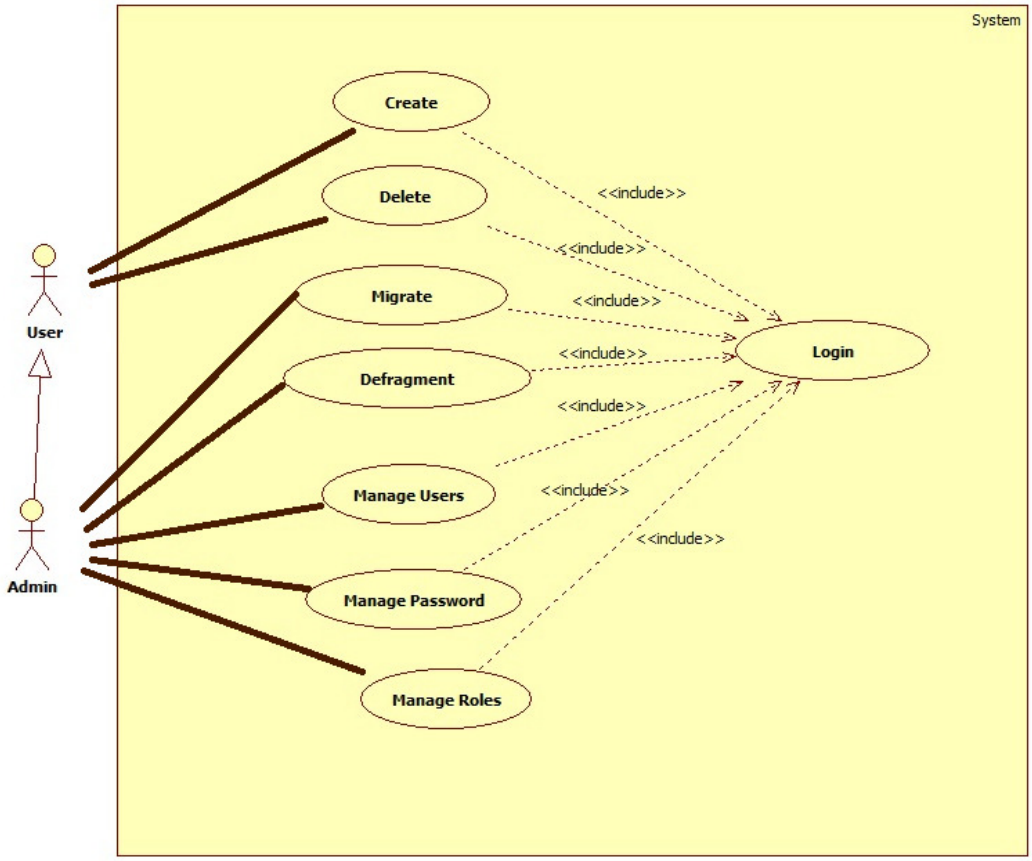
There is a MainFrame class, which is the frame all information is added. We have only one frame and the remaining classes are Panels. Which are added to the Frame by using CardLayout. This design helped us to increase the speed of the program.

This design helps us to add any functionality without effecting on others and increase independently of classes. Beside this if there is an unexpected error in one class does not crash the program and other functions are working properly.

Beside GUI package there is a test package, which is responsible for testing classes. It is connected to Controller and GUI package. This design helps us in order when we add more classes to GUI we can add a class to check functionality of that class.

4.4 Use Case

4.4.1 Use Case for Overall System



Description

Use Case Name	Login	
Actor	Normal User and Admin	
Triggering Events	The actor click on "Login" Button	
Pre-conditions	Actor much has their own user name and password.	
Post-conditions	Admin can access their home page Normal user can access their home page	
	Actor	System
Basic Flow	1. Actors input their own user name and password	1. The system gets user name and password.

	2. Actors can see their own page	2. Check the valid or not. 3. If valid show the welcome pages. If not show the login page again.
Alternative Flow	1. Actors input their own user name and password 2. Input user name or password is wrong. Back to login page again.	1. The system gets user name and password. 2. Check the valid or not. 3. If valid show the welcome pages. If not show the login page again.

Use Case Name	**Create**	
Actor	Normal User and Admin	
Triggering Events	The actor click on "Create" Menu	
Pre-conditions	Actor much has their own right.	
Post-conditions	Actors can see list of the available functions	
	Actor	System
Basic Flow	1. Actors click on "Create" menu 2. Actors can see list of "create" functions.	1. Check actors action right, show the available functions
Alternative Flow	n/a	n/a

Use Case Name	**Delete**
Actor	Normal User and Admin
Triggering Events	The actor click on "Delete" Menu

Pre-conditions	Actor much has their own right.	
Post-conditions	Actors can see list of the available functions	
	Actor	System
Basic Flow	1. Actors click on "Delete" menu 2. Actors can see list of "Delete" functions.	1. Check actors action right, show the available functions
Alternative Flow	n/a	n/a

Use Case Name	**Migrate**	
Actor	Admin	
Triggering Events	The actor click on "Migration" Menu	
Pre-conditions	Actor much has their own right.	
Post-conditions	Actors can see list of the available functions	
	Actor	System
Basic Flow	1. Actors click on "Migration" menu 2. Actors can see list of "Migration" functions.	1. Check actors action right, show the available functions
Alternative Flow	n/a	n/a

Use Case Name	**Defragment**
Actor	Admin
Triggering Events	The actor click on "Defragment" Menu

Pre-conditions	Actor much has their own right.	
Post-conditions	Actors can see list of the available functions	
	Actor	System
Basic Flow	1. Actors click on "Defragment" menu 2. Actors can see list of "Defragment" functions.	1. Check actors action right, show the available functions
Alternative Flow	n/a	n/a

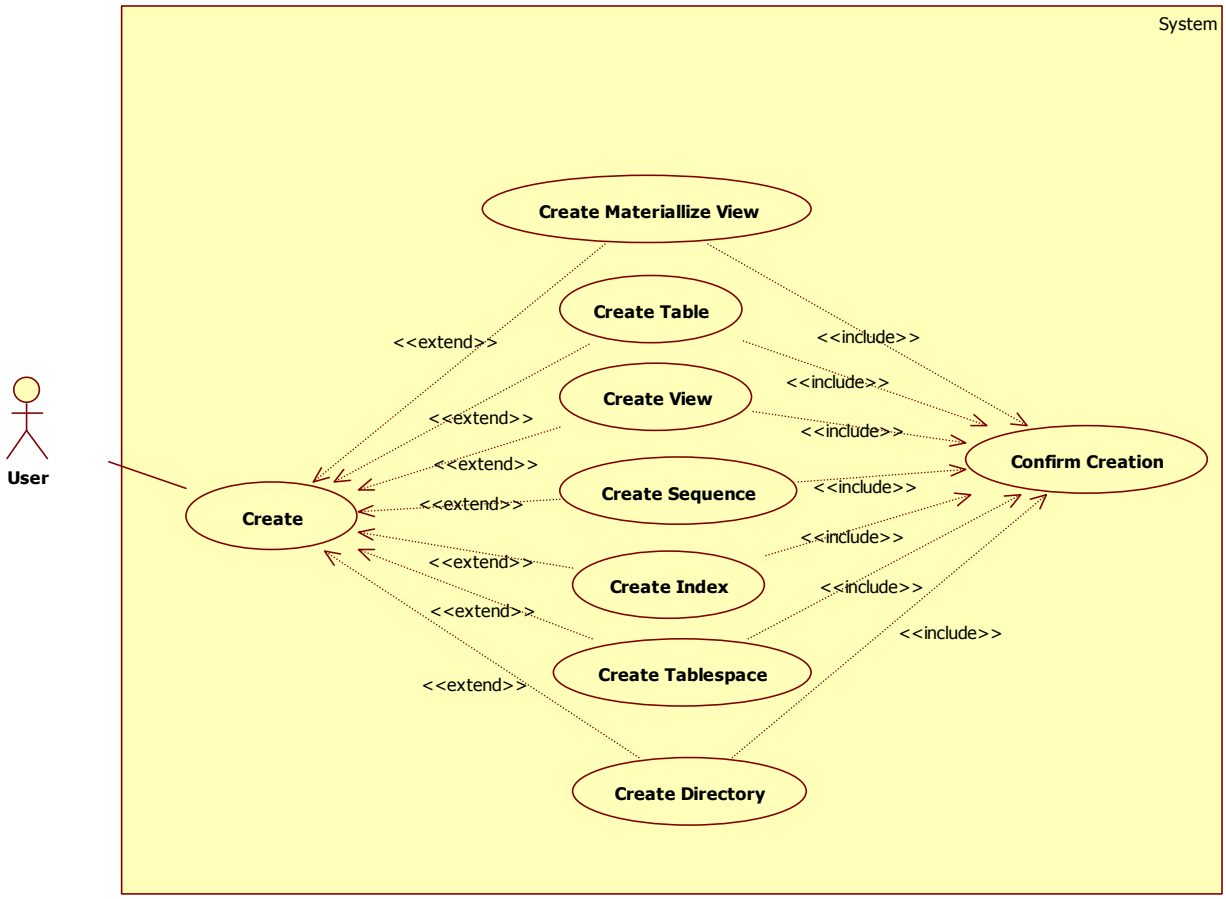
Use Case Name	**Manage User**	
Actor	Admin	
Triggering Events	The actor click on "User Control" Menu	
Pre-conditions	Actor much has their own right.	
Post-conditions	Actors can see list of the available user to manage	
	Actor	System
Basic Flow	1. Actors click on "User Control " menu 2. Actors can see list of "User" to manage and choose the user.	1. Check actors action update user right information.
Alternative Flow	n/a	n/a

Use Case Name	**Manage Password**
Actor	Normal User, Admin
Triggering	The actor click on "User Control" Menu

Events		
Pre-conditions	Actor much has their own right.	
Post-conditions	Actors can reset their own password.	
	Actor	System
Basic Flow	1. Actor click on the "User Control" menu 2. Actor can change password.	1. Check actors action update user right information.
Alternative Flow	n/a	n/a

Use Case Name	**Manage Roles**	
Actor	Normal User, Admin	
Triggering Events	The actor click on "User Control" Menu	
Pre-conditions	Actor much has their own right.	
Post-conditions	Actors can see list of the available functions	
	Actor	System
Basic Flow	1. Actor click on the "User Control" menu 2. Actor can grant privileges.	1. Check actors action right, show the available functions
Alternative Flow	n/a	n/a

4.4.2 Use Case for Creation



Description

Use Case Name	Create Tablespace	
Actor	User	
Triggering Events	The actor click on "Create - > Create Tablespace" Menu	
Pre-conditions	Actor much has their own right.	
Post-conditions	Actors can see create tablespace form to key in information that they would like to create.	
	Actor	System
Basic Flow	1. Actors click on "Create Tablespace" menu item. 2. Actors can see form to	1. Show "Create Tablespace" Form and let user to fill up information

	crate tablespace. 3. Fill up information and click "Create" to confirm.	2. Check information is correct or not. 3. If correct store in database. If not show the incorrect place.
Alternative Flow	1. Actors click on "Create Tablespace" menu item. 2. Actors can see form to create tablespace. 3. Fill up in-correct information and click "Create". 4. Re-fill correct information	1. Show "Create Tablespace" Form and let user to fill up information 2. Check information is correct or not. 3. If correct store in database. If not show the incorrect place.

Use Case Name	**Create Table**	
Actor	User	
Triggering Events	The actor click on "Create - > Create Table" Menu	
Pre-conditions	Actor much has their own right.	
Post-conditions	Actors can see create table form to key in information that they would like to create.	
	Actor	System
Basic Flow	1. Actors click on "Create Table" menu item. 2. Actors can see form to crate table. 3. Fill up information and click "Create" to confirm.	1. Show "Create Table" Form and let user to fill up information 2. Check information is correct or not. 3. If correct store in database. If not show the incorrect place.
Alternative	1. Actors click on "Create Table" menu item.	1. Show "Create Table" Form and let user to fill up

Flow	2. Actors can see form to create table. 3. Fill up in-correct information and click "Create". 4. Re-fill correct information	information 2. Check information is correct or not. 3. If correct store in database. If not show the incorrect place.

Use Case Name	**Create Index**	
Actor	User	
Triggering Events	The actor click on "Create - > Create Index" Menu	
Pre-conditions	Actor much has their own right.	
Post-conditions	Actors can see create Index form to key in information that they would like to create.	
	Actor	System
Basic Flow	1. Actors click on "Create Index" menu item. 2. Actors can see form to crate Index. 3. Fill up information and click "Create" to confirm.	1. Show "Create Index" Form and let user to fill up information 2. Check information is correct or not. 3. If correct store in database. If not show the incorrect place.
Alternative Flow	1. Actors click on "Create Index" menu item. 2. Actors can see form to create Index. 3. Fill up in-correct information and click "Create". 4. Re-fill correct	1. Show "Create Index" Form and let user to fill up information 2. Check information is correct or not. 3. If correct store in database. If not show the incorrect place.

	information	

Use Case Name	Create View	
Actor	User	
Triggering Events	The actor click on “Create - > Create View” Menu	
Pre-conditions	Actor much has their own right.	
Post-conditions	Actors can see create View form to key in information that they would like to create.	
	Actor	System
Basic Flow	1. Actors click on “Create View” menu item. 2. Actors can see form to crate Index. 3. Fill up information and click “Create” to confirm.	1. Show “Create View” Form and let user to fill up information 2. Check information is correct or not. 3. If correct store in database. If not show the incorrect place.
Alternative Flow	1. Actors click on “Create View” menu item. 2. Actors can see form to create View. 3. Fill up in-correct information and click “Create”. 4. Re-fill correct information	1. Show “Create View” Form and let user to fill up information 2. Check information is correct or not. 3. If correct store in database. If not show the incorrect place.

Use Case Name	Create Sequence
Actor	User

Triggering Events	The actor click on "Create - > Create Sequence" Menu	
Pre-conditions	Actor much has their own right.	
Post-conditions	Actors can see create Sequence form to key in information that they would like to create.	
	Actor	System
Basic Flow	1. Actors click on "Create Sequence" menu item. 2. Actors can see form to crate Sequence. 3. Fill up information and click "Create" to confirm.	1. Show "Create Sequence" Form and let user to fill up information 2. Check information is correct or not. 3. If correct store in database. If not show the incorrect place.
Alternative Flow	1. Actors click on "Create Sequence" menu item. 2. Actors can see form to create Sequence. 3. Fill up in-correct information and click "Create". 4. Re-fill correct information	1. Show "Create Sequence" Form and let user to fill up information 2. Check information is correct or not. 3. If correct store in database. If not show the incorrect place.

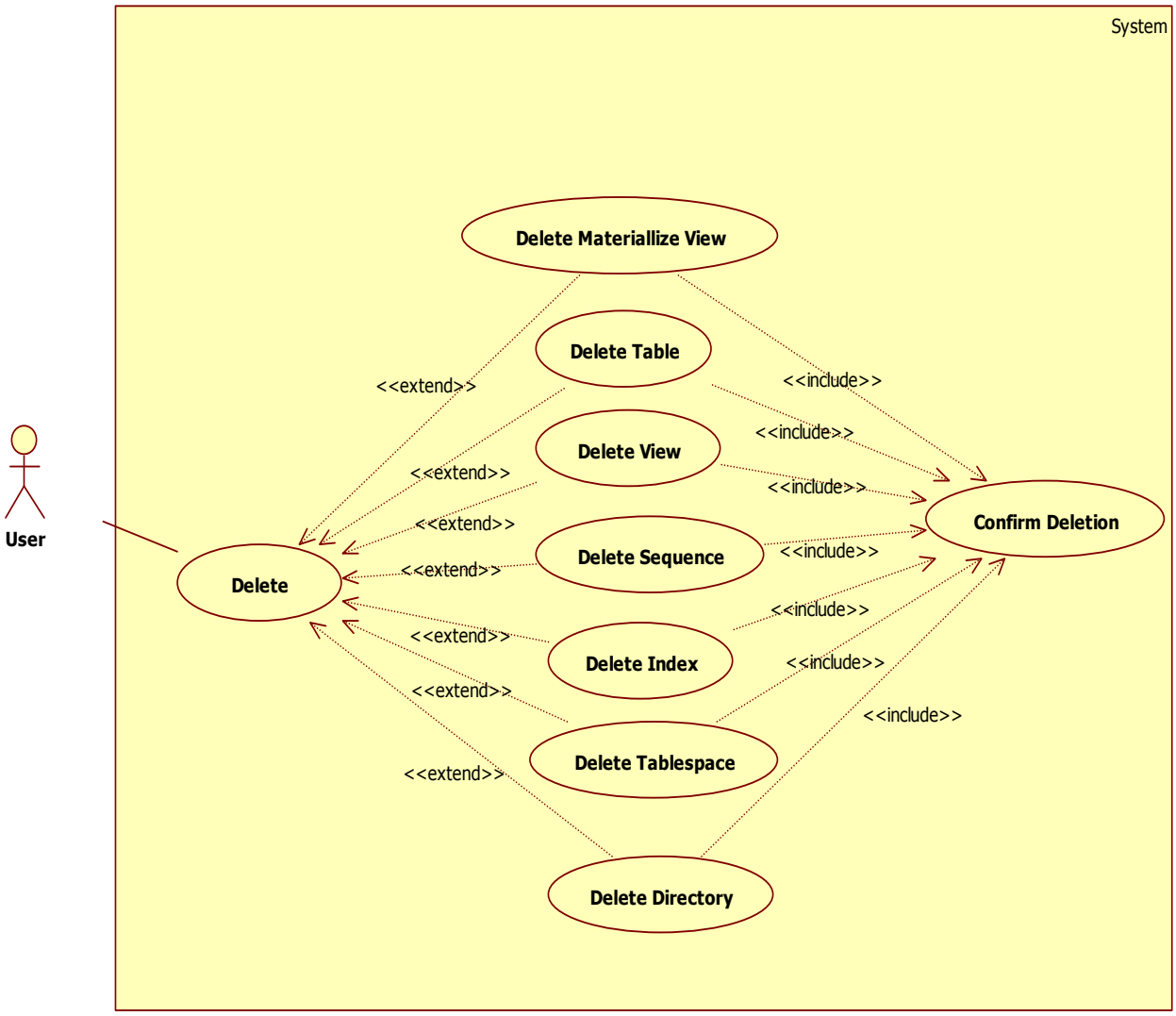
Use Case Name	**Create Materialize View**
Actor	User
Triggering Events	The actor click on "Create - > Create Materialize View" Menu
Pre-conditions	Actor much has their own right.
Post-conditions	Actors can see create Materialize View form to key in information that they would like to create.

	Actor	System
Basic Flow	1. Actors click on "Create Materialize View" menu item. 2. Actors can see form to crate Materialize View. 3. Fill up information and click "Create" to confirm.	1. Show "Create Materialize View" Form and let user to fill up information 2. Check information is correct or not. 3. If correct store in database. If not show the incorrect place.
Alternative Flow	1. Actors click on "Create Materialize View" menu item. 2. Actors can see form to create Materialize View. 3. Fill up in-correct information and click "Create". 4. Re-fill correct information	1. Show "Create Materialize View" Form and let user to fill up information 2. Check information is correct or not. 3. If correct store in database. If not show the incorrect place.

Use Case Name	**Create Directory**	
Actor	User	
Triggering Events	The actor click on "Create Directory " button	
Pre-conditions	Actor much has their own right.	
Post-conditions	Actors can see create directory form to key in information that they would like to create.	
	Actor	System
Basic Flow	1. Actors click on "Create Directory" button.	1. Show "Create Directory" Form and let user to fill up

	2. Actors can see the form of Create Directory. 3. Fill up information and click "Create" to confirm.	information 2. Check information is correct or not. 3. If correct store in database. If not show the incorrect place.
Alternative Flow	1. Actors click on "Create Directory" menu item. 2. Actors can see form to create Directory. 3. Fill up in-correct information and click "Create". 4. Re-fill correct information	1. Show "Create Directory" Form and let user to fill up information 2. Check information is correct or not. 3. If correct store in database. If not show the incorrect place.

4.4.3 Use Case for Deletion



Description

Use Case Name	Delete Table
Actor	User
Triggering Events	The actor click on “Delete - > Delete Table” Menu
Pre-conditions	Actors much have their own right.
Post-conditions	Actors can see Delete Table form to select information that they would like to delete.

	Actor	System
Basic Flow	1. Actors click on "Delete Table" menu item. 2. Actors can see form to Delete Table. 3. Select information and click "Delete" to confirm.	1. Show "Delete Table" Form and let user to choose information 2. Check action and if user confirm store into database.
Alternative Flow	n/a	n/a

Use Case Name	**Delete Tablespace**	
Actor	User	
Triggering Events	The actor click on "Delete - > Delete Tablespace" Menu	
Pre-conditions	Actors much have their own right.	
Post-conditions	Actors can see Delete Tablespace form to select information that they would like to delete.	
	Actor	System
Basic Flow	1. Actors click on "Delete Tablespace" menu item. 2. Actors can see form to Delete Tablespace. 3. Select information and click "Delete" to confirm.	1. Show "Delete Tablespace" Form and let user to choose information 2. Check action and if user confirm store into database.
Alternative Flow	n/a	n/a

Use Case Name	Delete Index	
Actor	User	
Triggering Events	The actor click on “Delete - > Delete Index” Menu	
Pre-conditions	Actors much have their own right.	
Post-conditions	Actors can see Delete Index form to select information that they would like to delete.	
	Actor	System
Basic Flow	1. Actors click on “Delete Index” menu item. 2. Actors can see form to Delete Index. 3. Select information and click “Delete” to confirm.	1. Show “Delete Index” Form and let user to choose information 2. Check action and if user confirm store into database.
Alternative Flow	n/a	n/a

Use Case Name	Delete Sequence	
Actor	User	
Triggering Events	The actor click on “Delete - > Delete Sequence” Menu	
Pre-conditions	Actors much have their own right.	
Post-conditions	Actors can see Delete Sequence form to select information that they would like to delete.	
	Actor	System
Basic Flow	1. Actors click on “Delete Sequence” menu item. 2. Actors can see form to	1. Show “Delete Sequence” Form and let user to choose information

	Delete Sequence. 3. Select information and click "Delete" to confirm.	2. Check action and if user confirm store into database.
Alternative Flow	n/a	n/a

Use Case Name	**Delete View**	
Actor	User	
Triggering Events	The actor click on "Delete - > Delete View" Menu	
Pre-conditions	Actors much have their own right.	
Post-conditions	Actors can see Delete View form to select information that they would like to delete.	
	Actor	System
Basic Flow	1. Actors click on "Delete View" menu item. 2. Actors can see form to Delete View. 3. Select information and click "Delete" to confirm.	1. Show "Delete View" Form and let user to choose information 2. Check action and if user confirm store into database.
Alternative Flow	n/a	n/a

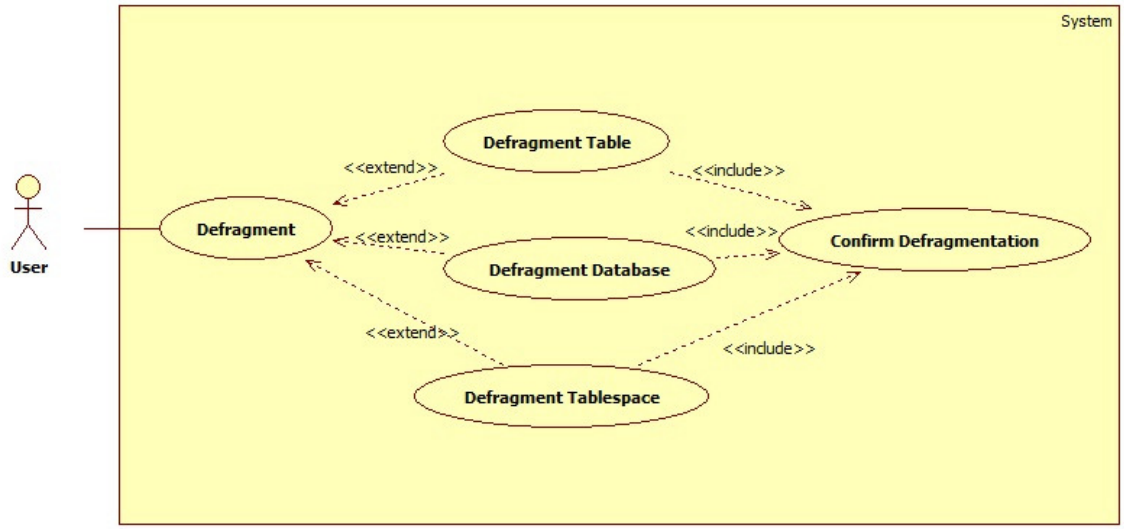
Use Case Name	**Delete Materialize View**
Actor	User
Triggering Events	The actor click on "Delete - > Delete Materialize View" Menu

Pre-conditions	Actors much have their own right.	
Post-conditions	Actors can see Delete Materialize View form to select information that they would like to delete.	
	Actor	System
Basic Flow	1. Actors click on "Delete Materialize View" menu item. 2. Actors can see form to Delete Materialize View. 3. Select information and click "Delete" to confirm.	1. Show "Delete Materialize View" Form and let user to choose information 2. Check action and if user confirm store into database.
Alternative Flow	n/a	n/a

Use Case Name	**Delete Directory**	
Actor	User	
Triggering Events	The actor click on "Delete Directory" Button	
Pre-conditions	Actors much have their own right.	
Post-conditions	Actors can see Delete Directory form to select information that they would like to delete.	
	Actor	System
Basic Flow	1. Actors click on "Delete Directory" Button to delete. 2. Actors can see form of Delete Directory. 3. Select information and	1. Show "Delete Directory" Form and let user to choose information 2. Check action and if user confirm store into database.

	click "Delete" to confirm.	
Alternative Flow	n/a	n/a

4.4.4 Use Case for Defragment



Description

Use Case Name	**Defragment Table**	
Actor	User	
Triggering Events	The actor click on "Defragment - > Defragment Table" Menu	
Pre-conditions	Actors much have their own right.	
Post-conditions	Actors can see Defragmentation selection that they would like to defragment	
	Actor	System
Basic Flow	1. Actors click on "Defragment Table" menu item. 2. Actors can see selection	1. Show selection and let user to choose 2. Check action and if user confirm do defragment.

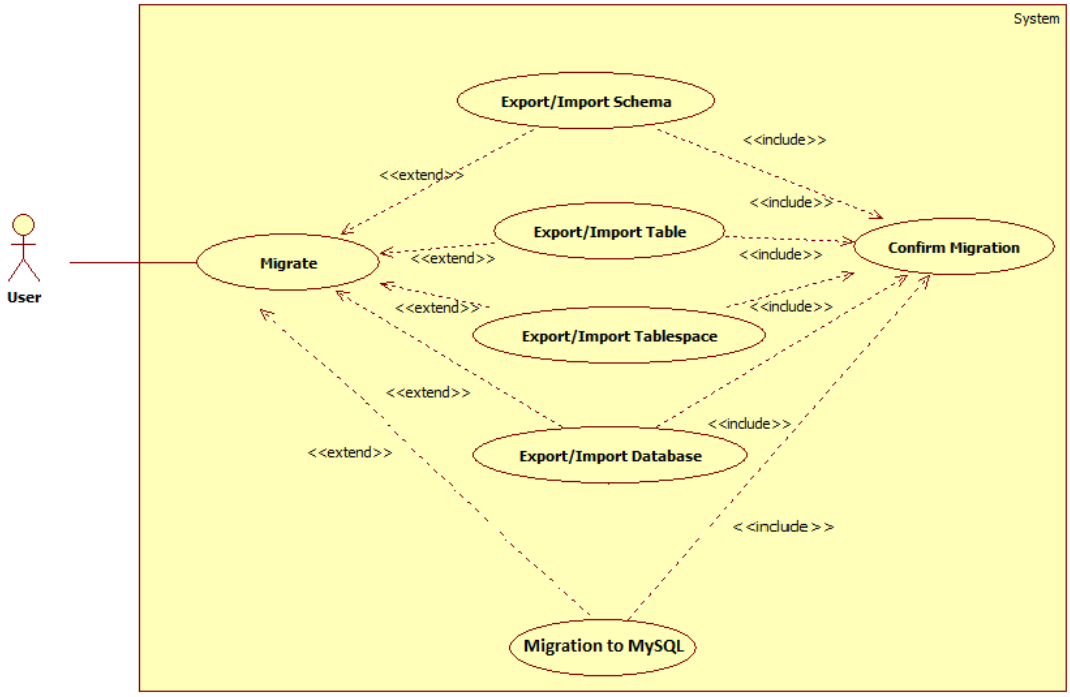
	and select "Select Table" and click "Defragment" to confirm.	
Alternative Flow	n/a	n/a

Use Case Name	**Defragment Tablespace**	
Actor	User	
Triggering Events	The actor click on "Defragment - > Defragment Tablespace" Menu	
Pre-conditions	Actors much have their own right.	
Post-conditions	Actors can see Defragmentation selection that they would like to defragment	
	Actor	System
Basic Flow	1. Actors click on "Defragment Tablespace" menu item. 2. Actors can see selection and select "Select Tablespace" and click "Defragment" to confirm.	1. Show selection and let user to choose 2. Check action and if user confirm do defragment.
Alternative Flow	n/a	n/a

Use Case Name	**Defragment Database**
Actor	User
Triggering Events	The actor click on "Defragment - > Defragment Database" Menu

Pre-conditions	Actors much have their own right.	
Post-conditions	Actors can see Defragmentation selection that they would like to defragment	
	Actor	System
Basic Flow	1. Actors click on "Defragment Database" menu item. 2. Actors can see selection and select "Select Database" and click "Defragment" to confirm.	1. Show selection and let user to choose 2. Check action and if user confirm do defragment.
Alternative Flow	n/a	n/a

4.4.5 Use Case for Migration



Description

Use Case Name	**Export/Import Schema**	
Actor	User	
Triggering Events	The actor click on "Migrate - > Export/Import Schema" Menu	
Pre-conditions	Actors much have their own right.	
Post-conditions	Actors can see Export/Import Schema form to key in information that they would like to migrate.	
	Actor	System
Basic Flow	1. Actors click on "Export/Import Schema" menu item. 2. Actors can see Export Schema form and Import Schema form. 3. Key in information and choose the directory. 4. Actors click "Import" to confirm. Or Actors click "Export" to confirm.	1. Show the export/import Schema form and let user to fill up information. 2. Check information is correct or not. 3. If yes, save information to database and do migration. If no, key in correct information.
Alternative Flow	1. Actors click on "Export/Import Schema" menu item. 2. Actors can see Export Schema form and Import Schema form. 3. Key in wrong information. 4. Allow actor to re-migrate	1. Show the export/import Schema form and let user to fill up information. 2. Check information is correct or not. 3. If yes, save information to database and do migration. If no, key in correct information.

	after correction	

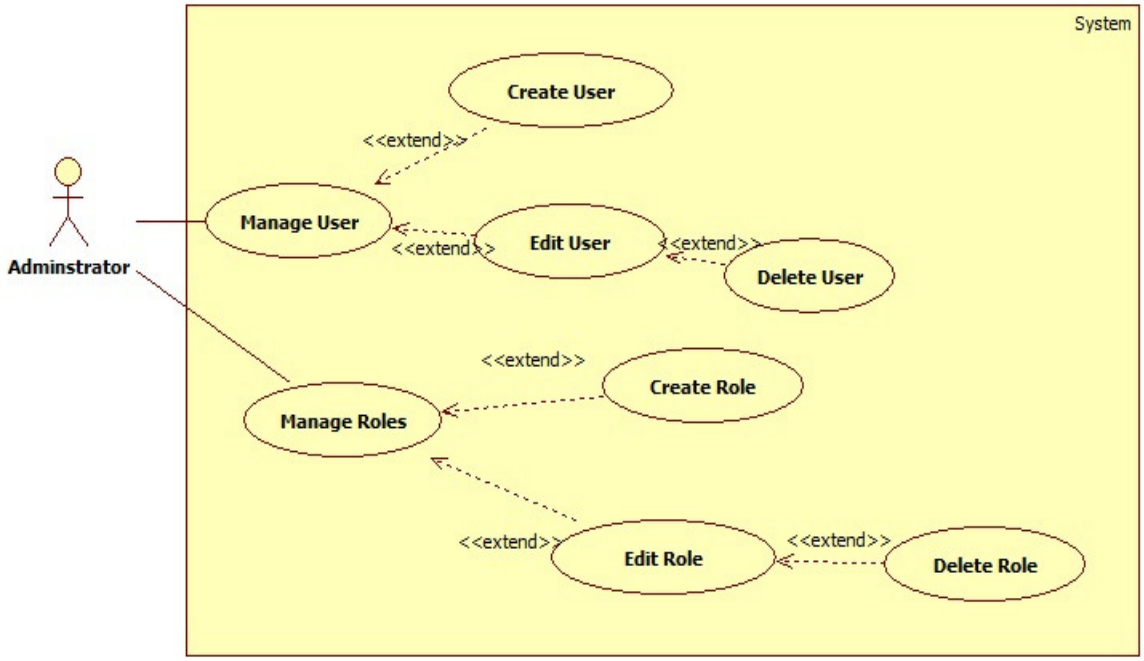
Use Case Name	Export/Import Table	
Actor	User	
Triggering Events	The actor click on “Migrate - > Export/Import Table” Menu	
Pre-conditions	Actors much have their own right.	
Post-conditions	Actors can see Export/Import Table form to key in information that they would like to migrate.	
	Actor	System
Basic Flow	1. Actors click on “Export/Import Table” menu item. 2. Actors can see Export Table form and Import Table form. 3. Key in information and choose the directory. 4. Actors click “Import” to confirm. Or Actors click “Export” to confirm.	1. Show the export/import Table form and let user to fill up information. 2. Check information is correct or not. 3. If yes, save information to database and do migration. If no, key in correct information.
Alternative Flow	1. Actors click on “Export/Import Table” menu item. 2. Actors can see Export Table form and Import Table form. 3. Key in wrong information. 4. Allow actor to re-migrate after correction after correction	1. Show the export/import Table form and let user to fill up information. 2. Check information is correct or not. 3. If yes, save information to database and do migration. If no, key in correct information.

Use Case Name	Export/Import Tablespace	
Actor	User	
Triggering Events	The actor click on "Migrate - > Export/Import Tablespace" Menu	
Pre-conditions	Actors much have their own right.	
Post-conditions	Actors can see Export/Import Tablespace form to key in information that they would like to migrate.	
	Actor	System
Basic Flow	1. Actors click on "Export/Import Tablespace" menu item. 2. Actors can see Export Tablespace form and Import Tablespace form. 3. Key in information and choose the directory. 4. Actors click "Import" to confirm. Or Actors click "Export" to confirm.	1. Show the export/import Tablespace form and let user to fill up information. 2. Check information is correct or not. 3. If yes, save information to database and do migration. If no, key in correct information.
Alternative Flow	1. Actors click on "Export/Import Tablespace" menu item. 2. Actors can see Export Tablespace form and Import Tablespace form. 3. Key in wrong information. 4. Allow actor to re-migrate after correction after correction after correction	1. Show the export/import Tablespace form and let user to fill up information. 2. Check information is correct or not. 3. If yes, save information to database and do migration. If no, key in correct information.

Use Case Name	Export/Import Database	
Actor	User	
Triggering Events	The actor click on "Migrate - > Export/Import Database" Menu	
Pre-conditions	Actors much have their own right.	
Post-conditions	Actors can see Export/Import Database form to key in information that they would like to migrate.	
	Actor	System
Basic Flow	1. Actors click on "Export/Import Database" menu item. 2. Actors can see Export Tablespace form and Import Tablespace form. 3. Key in information and choose the directory. 4. Actors click "Import" to confirm. Or Actors click "Export" to confirm.	1. Show the export/import Database form and let user to fill up information. 2. Check information is correct or not. 3. If yes, save information to database and do migration. If no, key in correct information.
Alternative Flow	1. Actors click on "Export/Import Database" menu item. 2. Actors can see Export Tablespace form and Import Tablespace form. 3. Key in wrong information. 4. Allow actor to re-migrate after correction after correction after correction	1. Show the export/import Database form and let user to fill up information. 2. Check information is correct or not. 3. If yes, save information to database and do migration. If no, key in correct information.

Use Case Name	**Migration to MySQL**	
Actor	User	
Triggering Events	The actor click on "Migrate - > Migrate To MySQL" Menu	
Pre-conditions	Actors much have their own right.	
Post-conditions	Actors can see Export/Import different platform form to select information that they would like to migrate.	
	Actor	System
Basic Flow	1. Actors click on "Migrate To MySQL" menu item. 2. Actors can see Export form and Import form. 3. Key in information and choose the directory. 4. Actors click "Import" to confirm. Or Actors click "Export" to confirm.	1. Show the Migrate To MySQL form and let user to fill up information. 2. Check information is correct or not. 3. If yes, save information to database and do migration. If no, key in correct information.
Alternative Flow	1. Actors click on "Migrate Cross Platform" menu item. 2. Actors can see Export form and Import form. 3. Key in information and choose the directory. 4. Actors click "Import" to confirm. Or Actors click "Export" to confirm.	1. Show the Migrate Cross Platform form and let user to fill up information. 2. Check information is correct or not. 3. If yes, save information to database and do migration. If no, key in correct information.

4.4.6 Use Case for Manage User and Manage Role



Description

Use Case Name	Create User	
Actor	Administrator	
Triggering Events	The actor click on "User Control- >Create User" Menu	
Pre-conditions	Actors much have their own right.	
Post-conditions	Actors can see Create User form to key in information that they would like to create.	
	Actor	System
Basic Flow	1. Actors click on "Create User" menu item. 2. Actors can see form to Create User. 3. Key in information and click "Create" to confirm.	1. Show "Create User" Form and let user to key in information 2. Check information is correct or not. 3. If yes store into database and if no let user to key in again.
Alternative	1. Actors click on "Create	1. Show "Create User" Form and

Flow	User" menu item. 2. Actors can see form to Create User 3. Key in wrong information and click "Create" 4. Re-key in correct information.	let user to key in information 2. Check information is correct or not. 3. If yes, store into database and if no let user to key in again.

Use Case Name	**Edit User**	
Actor	Administrator	
Triggering Events	The actor click on "User Control - > Grant Privileges" Menu	
Pre-conditions	Actors much have their own right.	
Post-conditions	Actors can see Grant Privileges form to change information that they would like to change.	
	Actor	System
Basic Flow	1. Actors click on "Grant Privileges" menu item. 2. Actors can see form to Grant Privileges and grant privileges to user. 3. Select information and click "OK" to confirm.	1. Show "Grant Privileges" Form and let user to grant or revoke the privileges. 2. Check information is correct or not. 3. If yes, store into database and if no let user to key in again.
Alternative Flow	1. Actors click on "Grant Privileges" menu item. 2. Actors can see form to Grant Privileges and grant privileges to user. 3. Key in wrong information and click	1. Show "Create User" Form and let user to key in information 2. Check information is correct or not. 3. If yes, store into database and if no let user to key in again.

	"Ok" 4. Re-key in correct information.	

Use Case Name	Delete User	
Actor	Administrator	
Triggering Events	The actor click on "User Control- >Delete User" Menu	
Pre-conditions	Actors much have their own right.	
Post-conditions	Actors can see Delete User form to key in information that they would like to delete.	
	Actor	System
Basic Flow	1. Actors click on "Delete User" menu item. 2. Actors can see form to Delete User. 3. Key in information and click "Delete" to confirm.	1. Show "Delete User" Form and let user to key in information 2. Check information is correct or not. 3. If yes store into database and if no let user to key in again.
Alternative Flow	1. Actors click on "Delete User" menu item. 2. Actors can see form to Delete User 3. Key in wrong information and click "Delete" 4. Re-key in correct information.	1. Show "Delete User" Form and let user to key in information 2. Check information is correct or not. 3. If yes, store into database and if no let user to key in again.

Use Case Name	Create Role	
Actor	Administrator and Normal User	
Triggering Events	The actor click on “User Control - >” Assign Role” Menu	
Pre-conditions	Actors much have their own right to do this.	
Post-conditions	Actors can see the lists of functions.	
	Actor	System
Basic Flow	1. Actors click on “Assign Role” menu item. 2. Actors can see the “Create Role” tab and input name or password (Optional). 3. Click “OK” to confirm.	1. Show “Assign Role” Form and let user to create role. 2. Check information is correct or not. 3. If yes, store into database and if no let user to key in again.
Alternative Flow	1. Actors click on “Assign Role” menu item. 2. Actors can see the “Create Role” tab and input name or password (Optional). 3. Key in wrong information and click “Delete” 4. Re-key in correct information.	1. Show “Assign Role” Form and let user to create role. 2. Check information is correct or not. 3. If yes, store into database and if no let user to key in again.

Use Case Name	Edit Role
Actor	Administrator and Normal User
Triggering Events	The actor click on “Manage User - > “Assign Role” → “Set/Alter Role” Tab

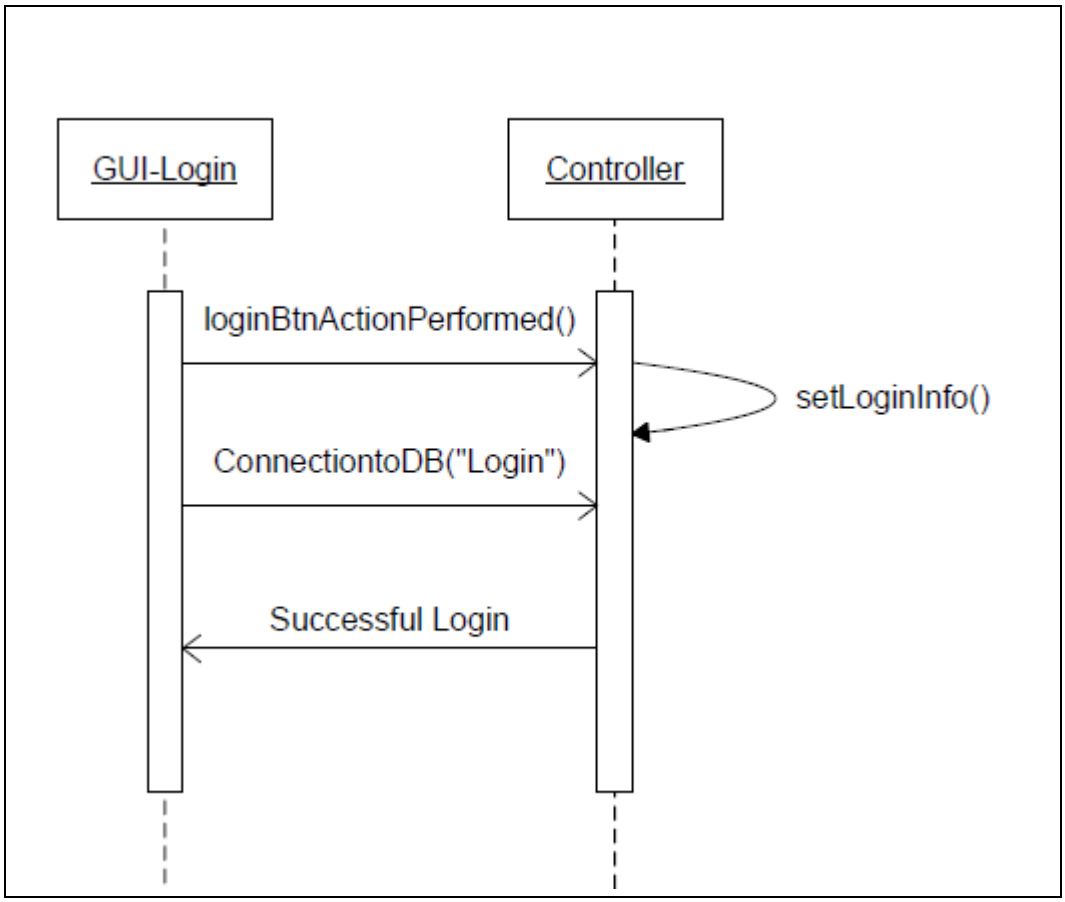
Pre-conditions	Actors much have their own right to do this.	
Post-conditions	Actors can see the lists of functions.	
	Actor	System
Basic Flow	1. Actors click on "Set/Alter Role" tab. 2. Actors can see form to edit role. 3. Click "OK" to confirm.	1. Show "Set/Alter Role" Form and let user to edit role. 2. Check information is correct or not. 3. If yes, store into database and if no let user to key in again.
Alternative Flow	1. Actors click on "Set/Alter Role" tab. 2. Actors can see form to edit role. 3. Key in wrong information and click "OK" 4. Re-key in correct information.	1. Show "Set/Alter Role" Form and let user to edit role. 2. Check information is correct or not. 3. If yes, store into database and if no let user to key in again.

Use Case Name	**Manage Password**	
Actor	Administrator and Normal User	
Triggering Events	The actor click on "Manage User - > "Manage Password" Menu	
Pre-conditions	Actors much have their own user name and password.	
Post-conditions	Actors can see change password form to change their password.	
	Actor	System
Basic Flow	1. Actors click on "Manage Password" menu item. 2. Actors can see form to Change the password and	1. Show "Change Password" Form and let user to change their password. 2. Check information is correct or

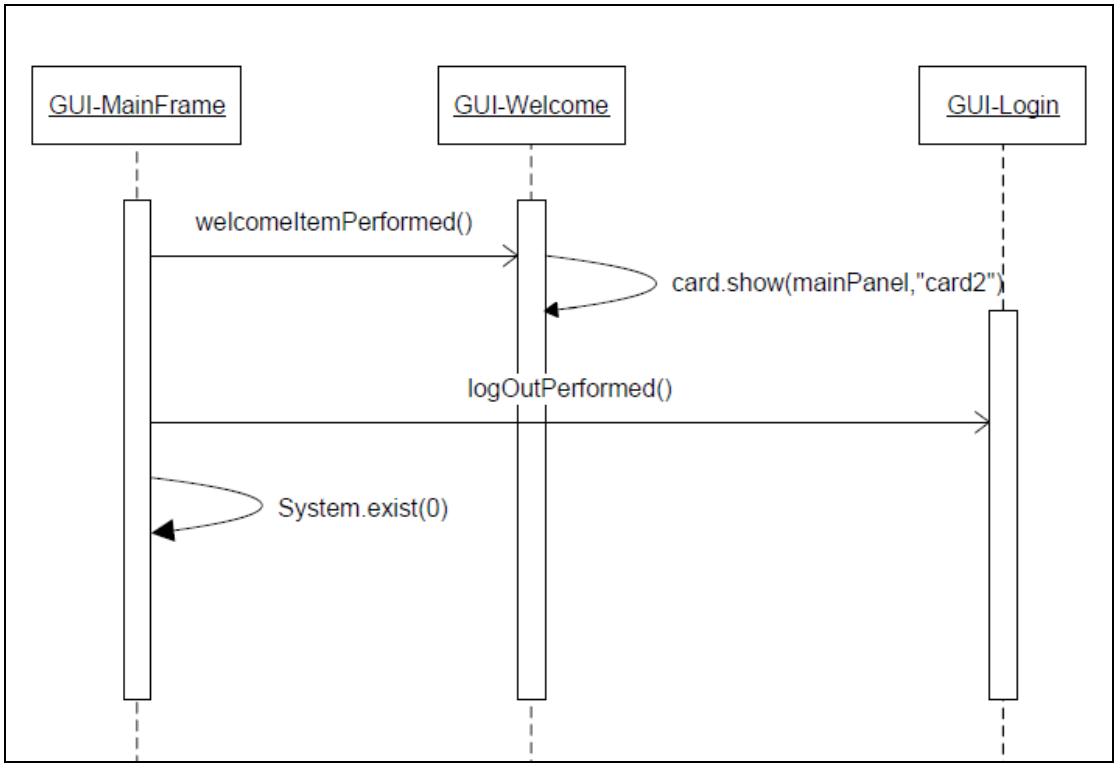
	key in old password follow by new password and confirm password. 3. Click "OK" to confirm.	not. 3. If yes, store into database and if no let user to key in again.
Alternative Flow	1. Actors click on "Manage Password" menu item. 2. Actors can see form to Change the password and key in wrong information or password not match 3. Allow user to re-key in correct information.	1. Show "Manage Password" Form and let user to change information 2. Check information is correct or not. 3. If yes, store into database and if no let user to key in again.

4.5 Sequence Diagrams

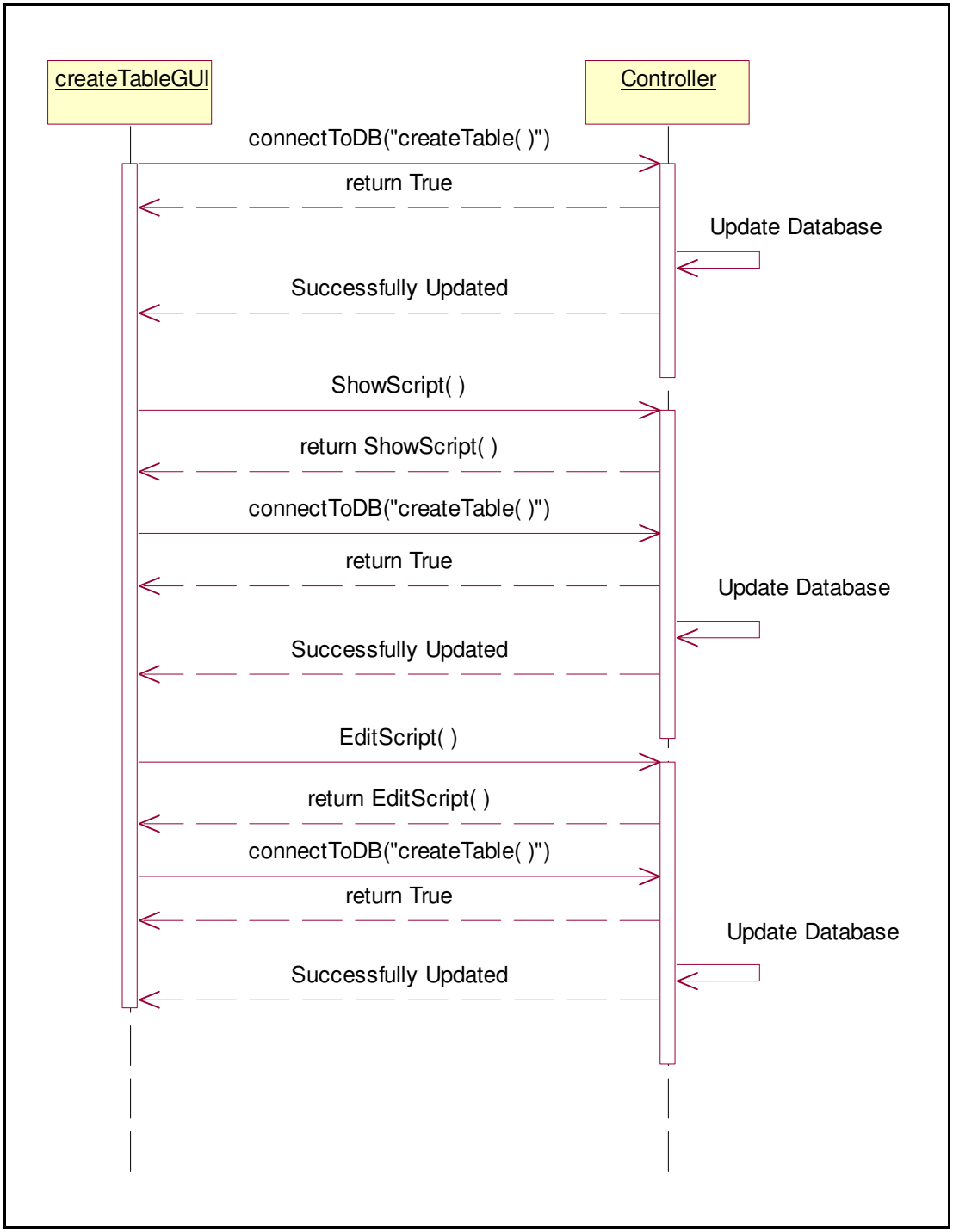
4.5.1 Sequence Diagram for Login



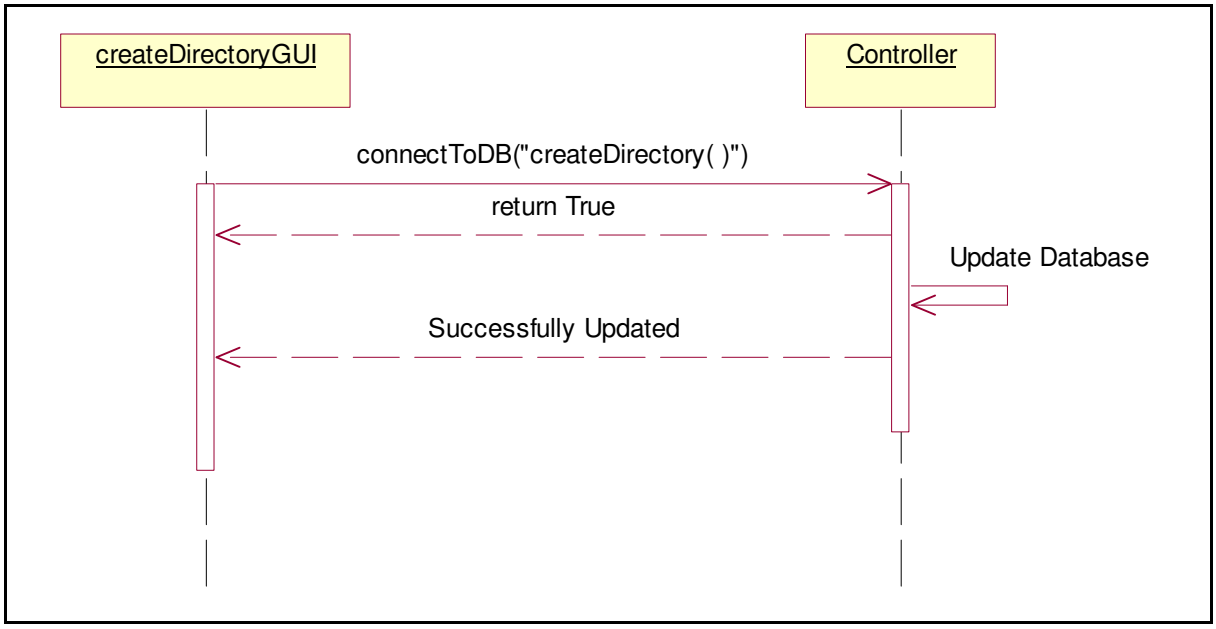
4.5.2 Sequence Diagram for File



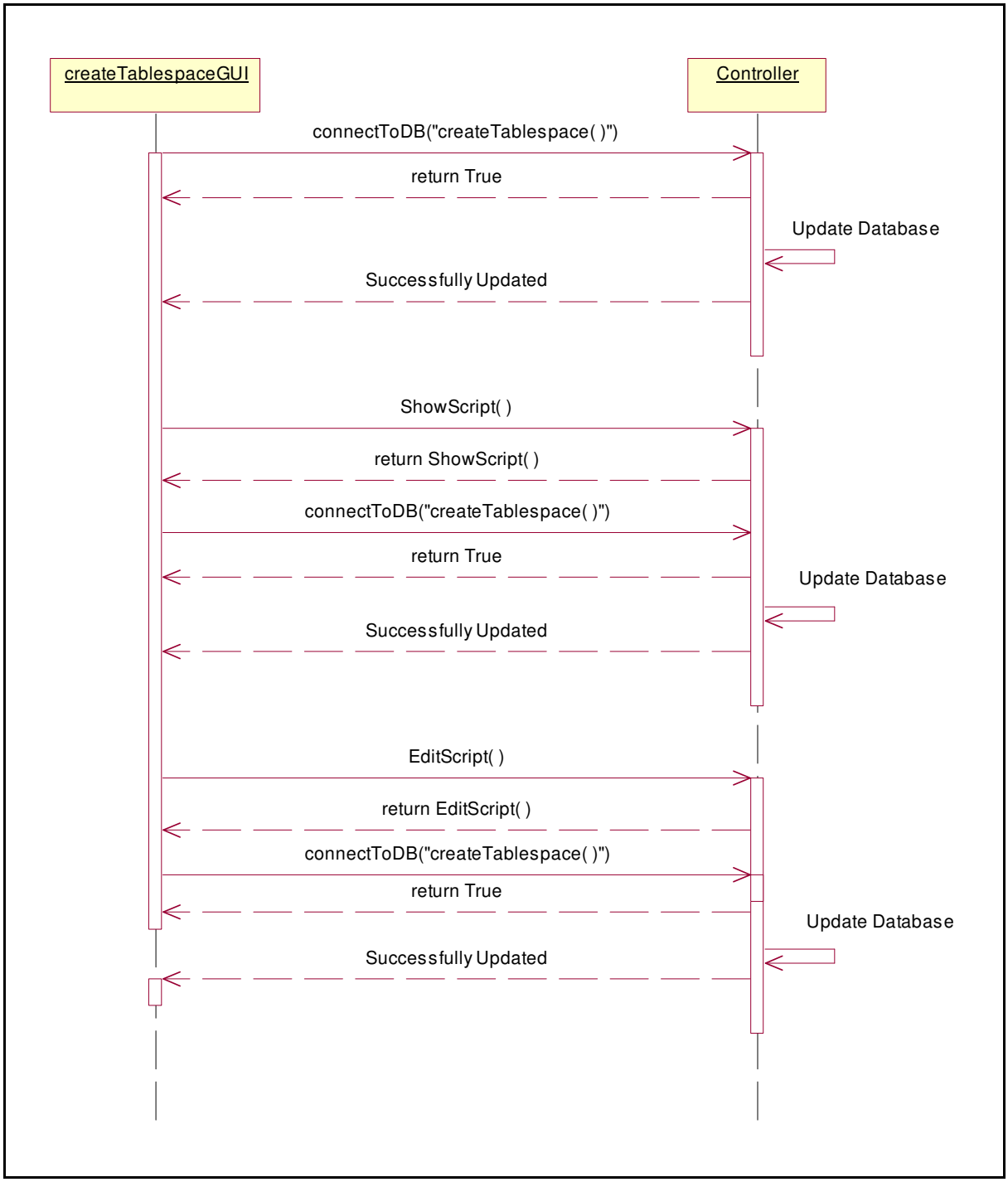
4.5.3 Sequence Diagram for Create Table



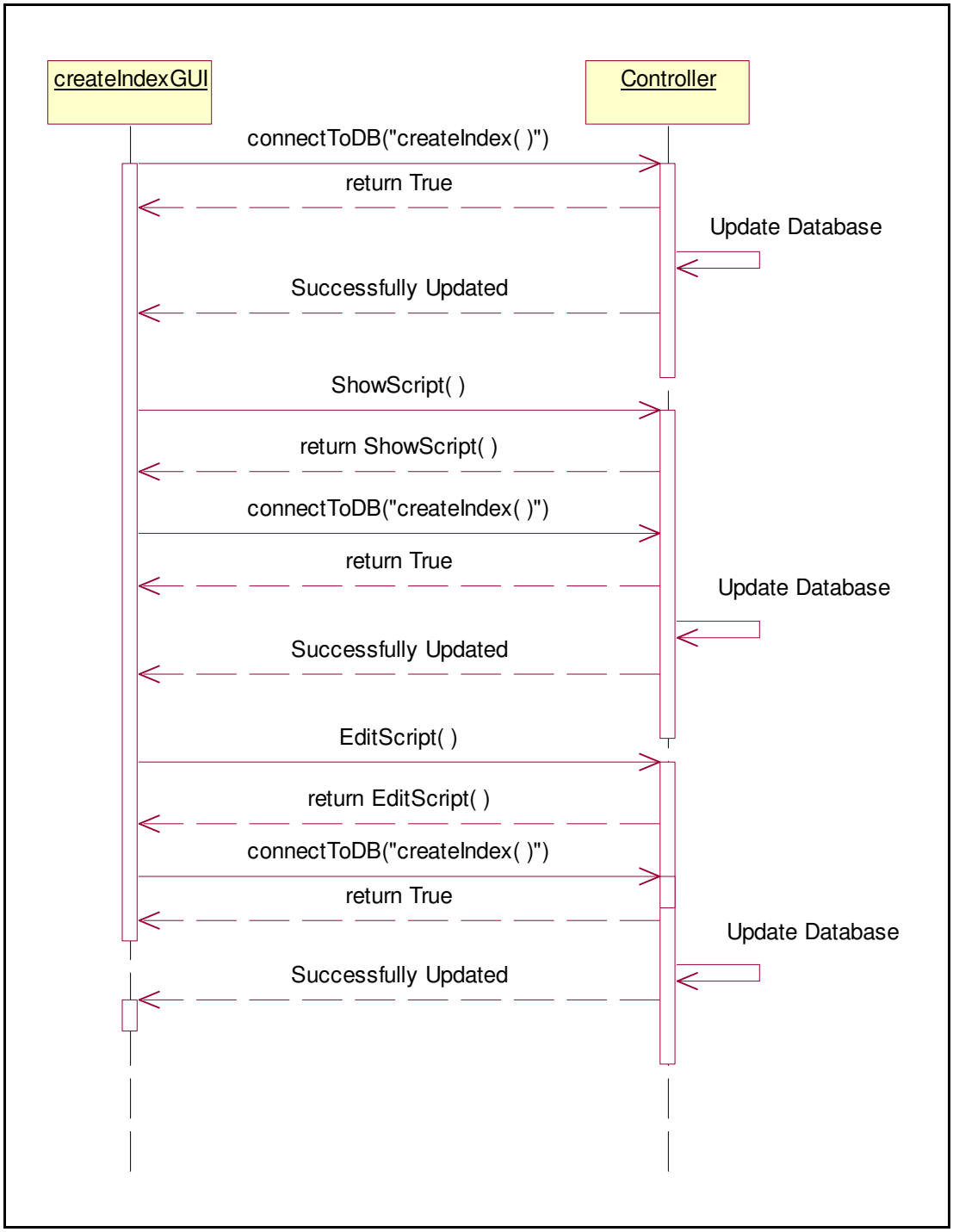
4.5.4 Sequence Diagram for Create Directory



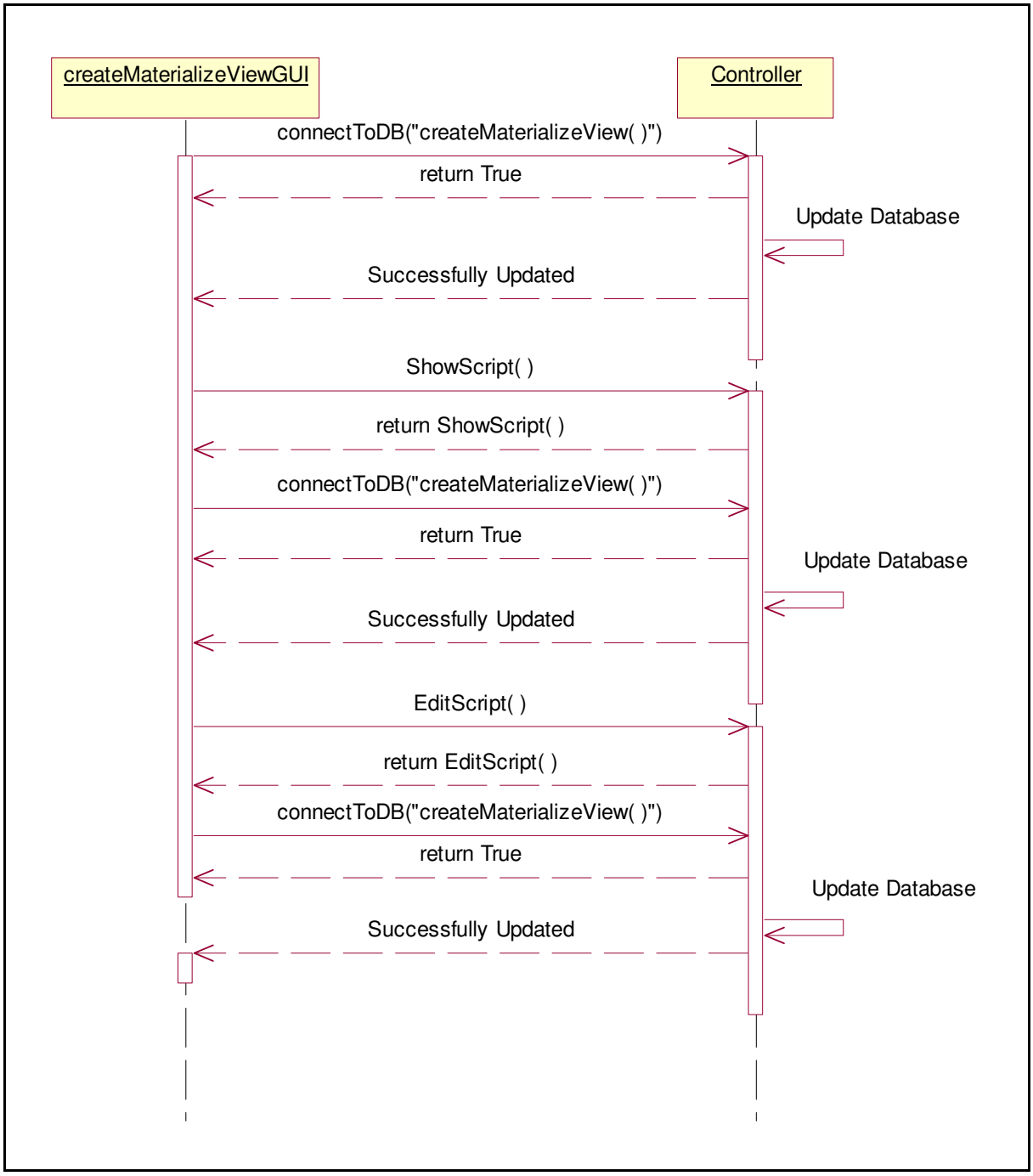
4.5.5 Sequence Diagram for Create Tablespace



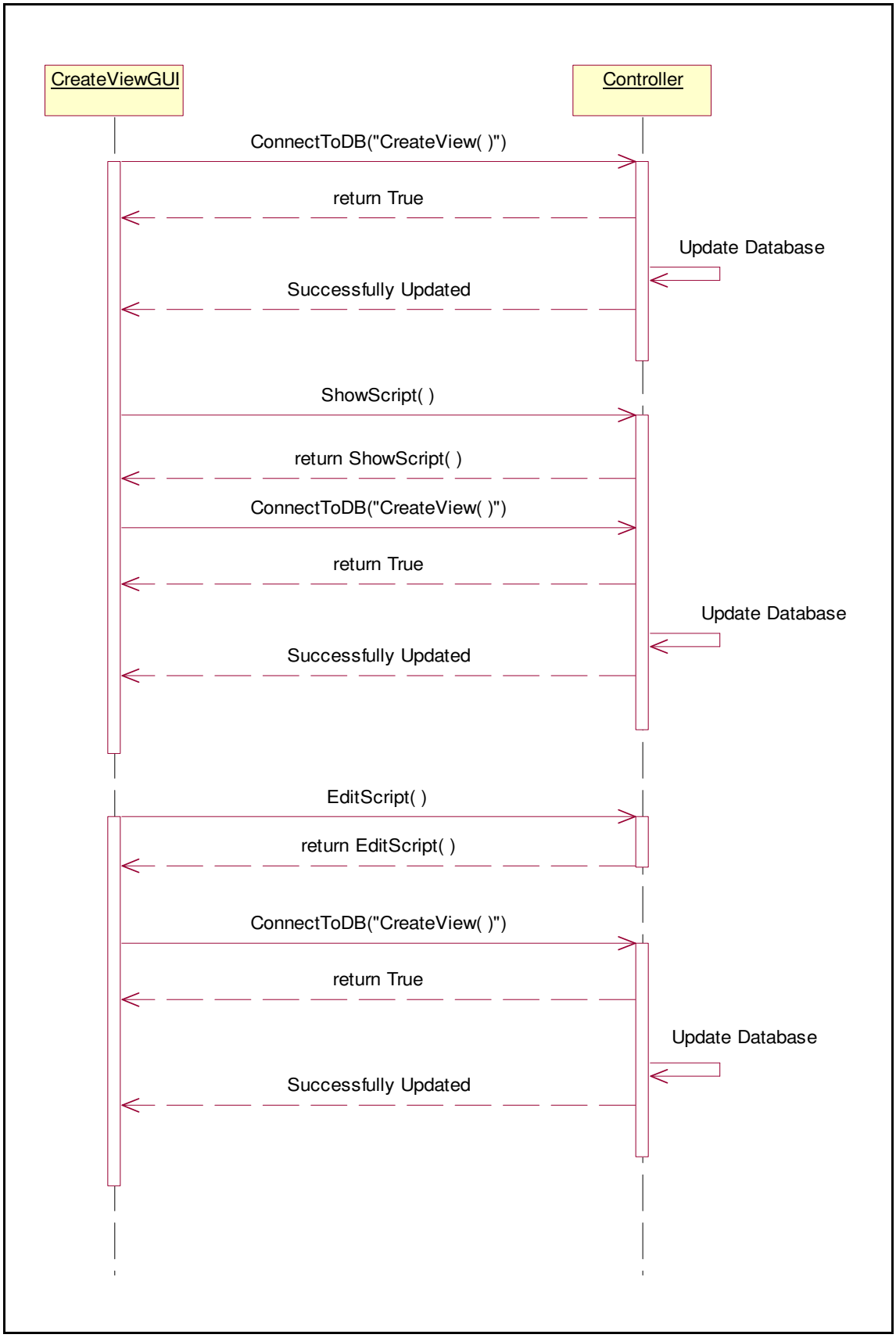
4.5.6 Sequence Diagram for Create Index



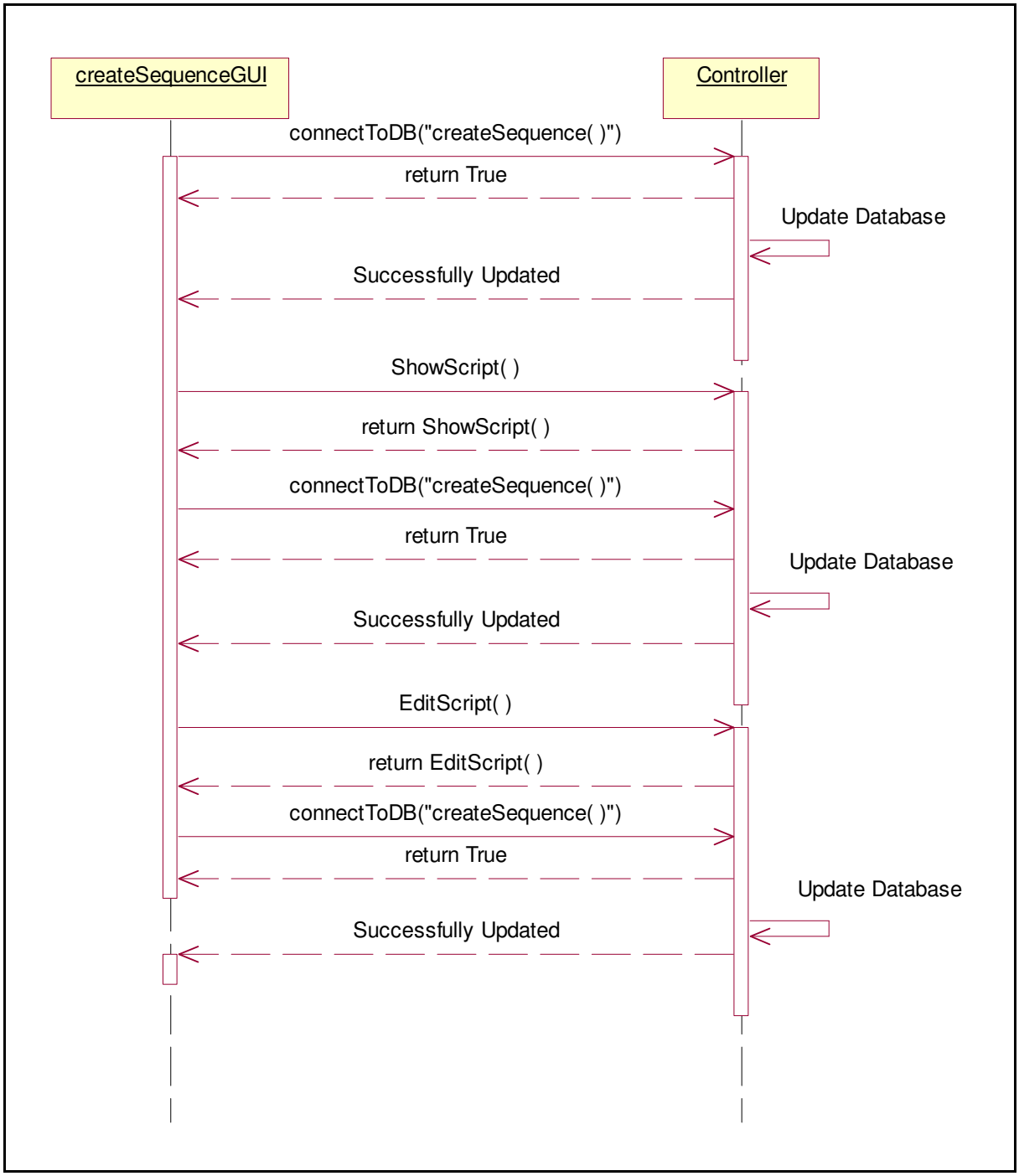
4.5.7 Sequence Diagram for Create Materialize View



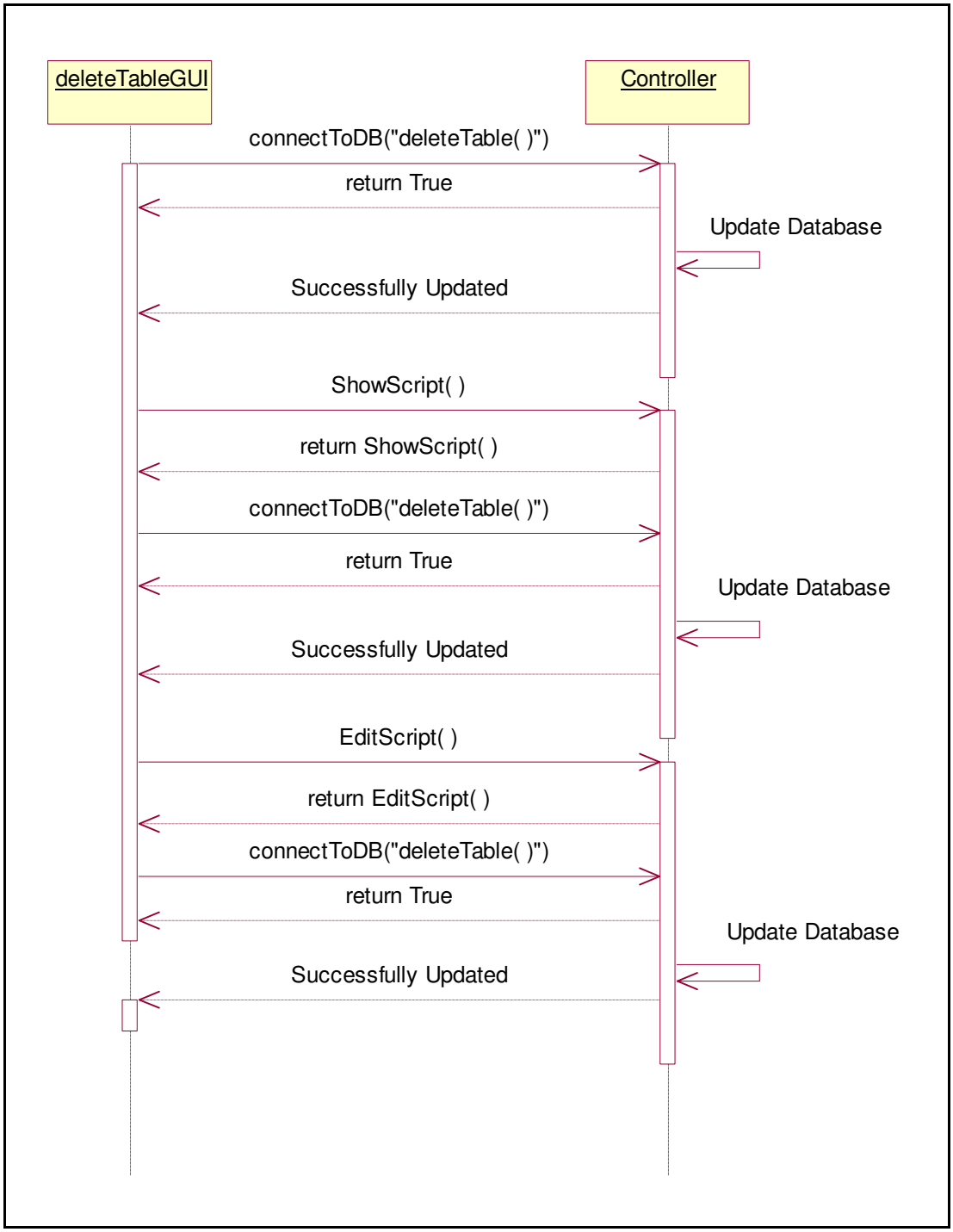
4.5.8 Sequence Diagram for Create View



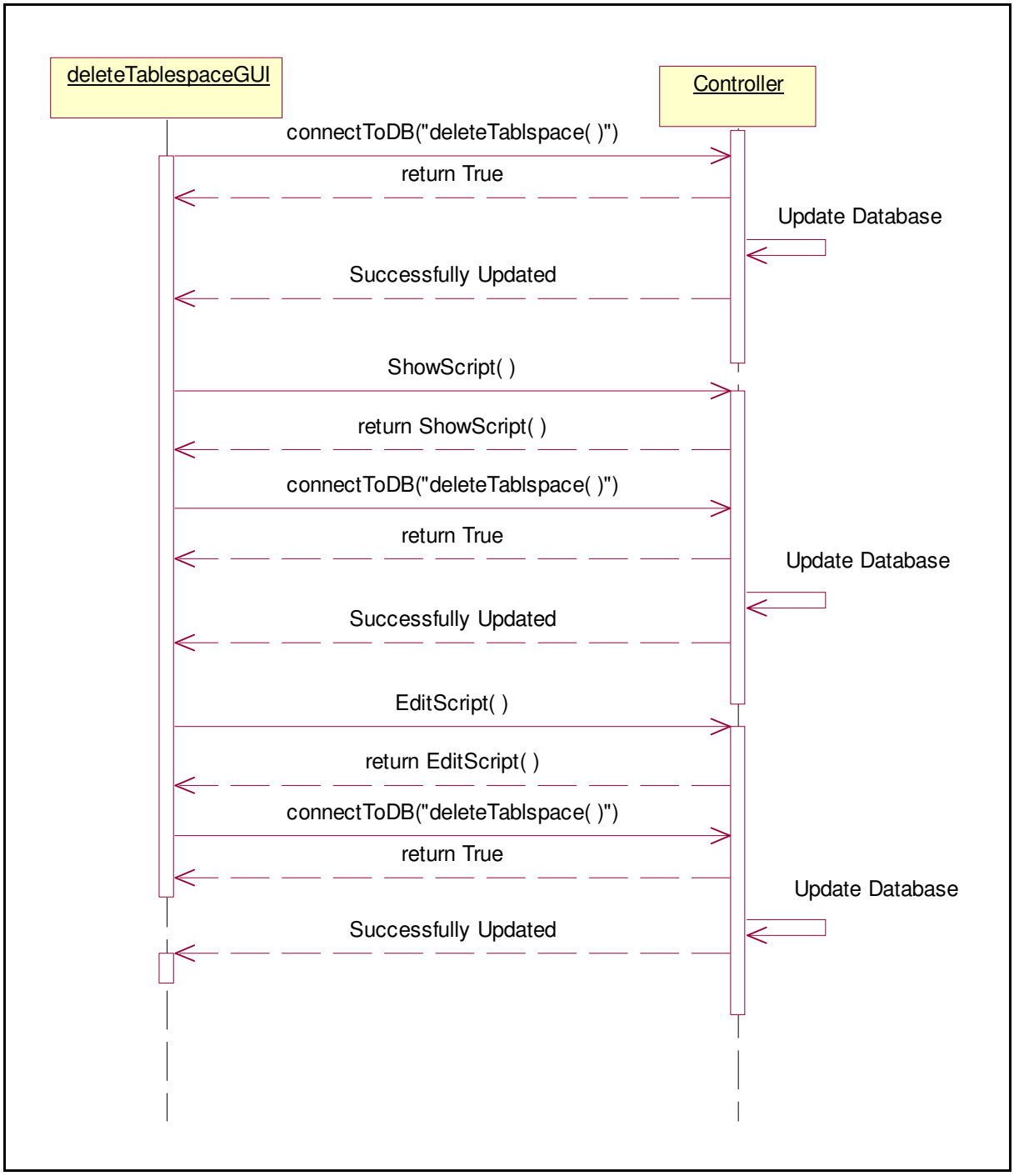
4.5.9 Sequence Diagram for Create Sequence



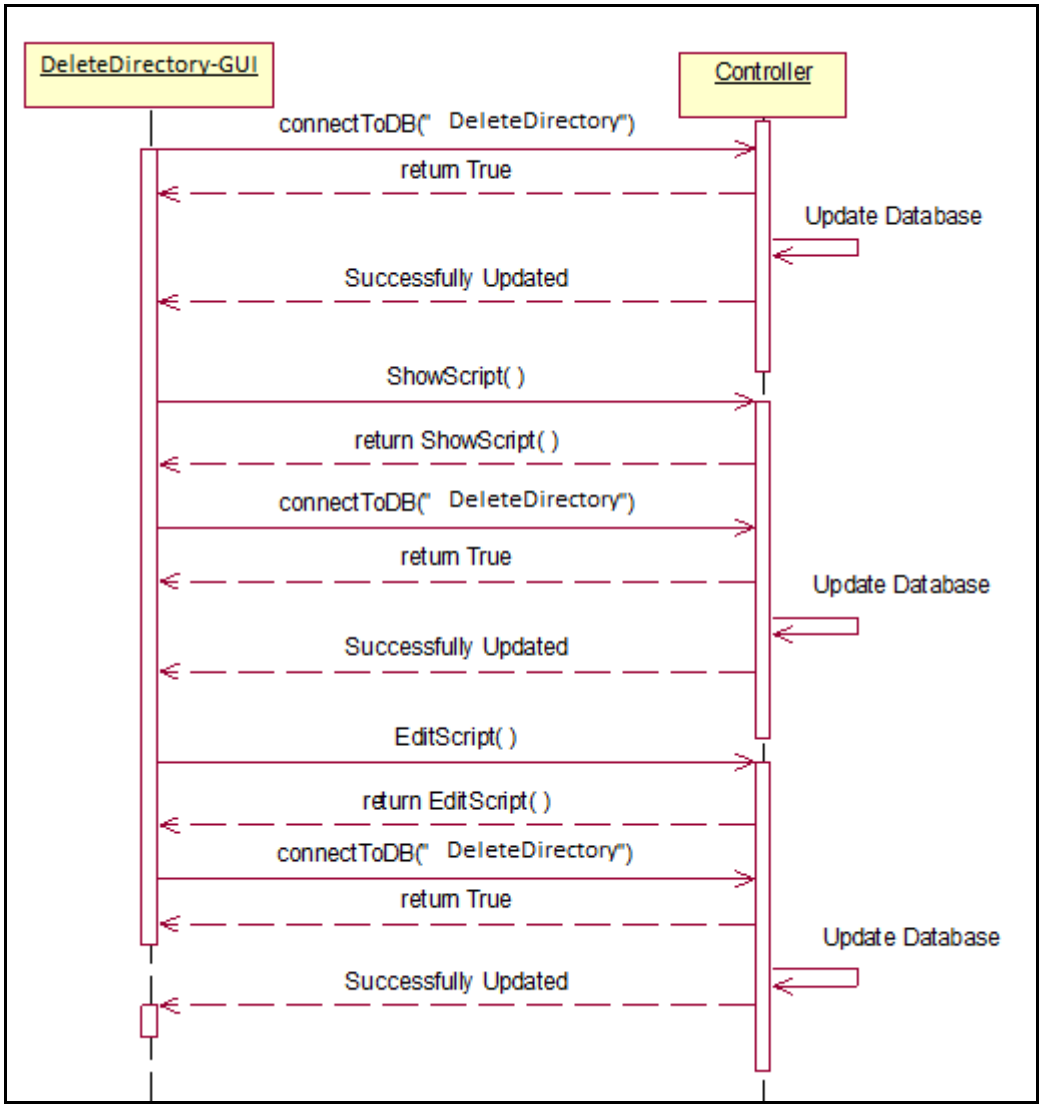
4.5.10 Sequence Diagram for Delete Table



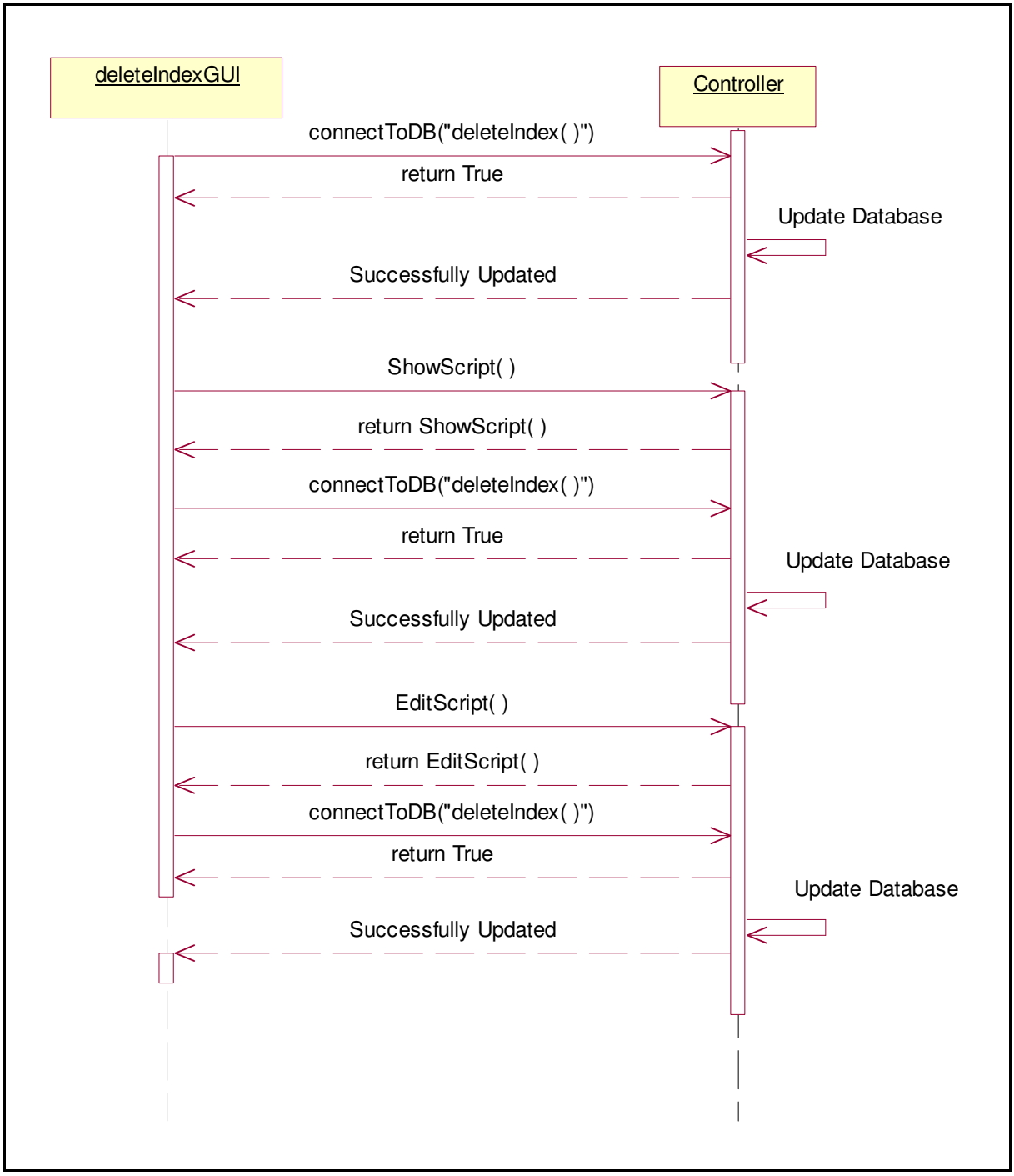
4.5.11 Sequence Diagram for Delete Tablespace



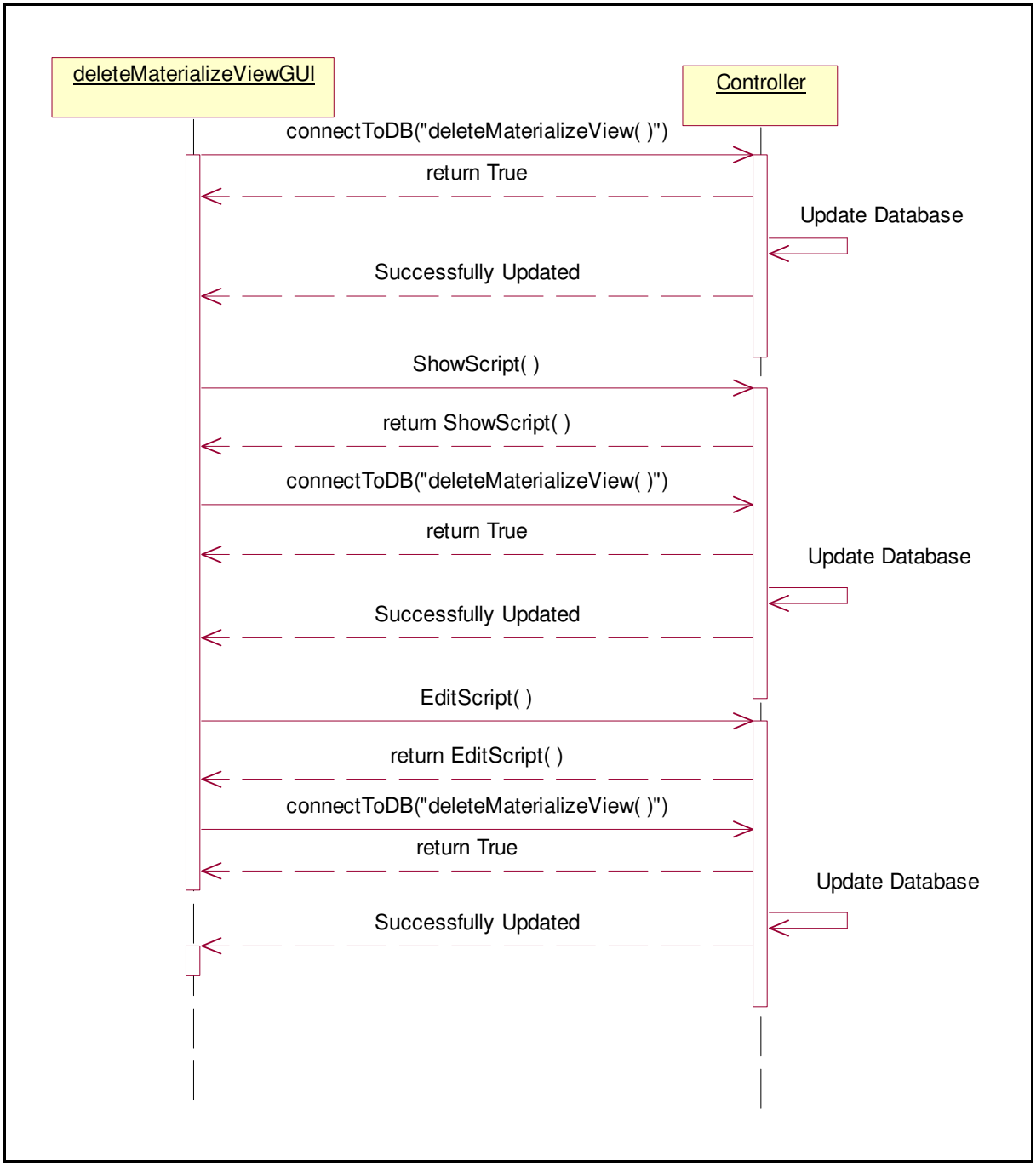
4.5.12 Sequence Diagram for Delete Directory



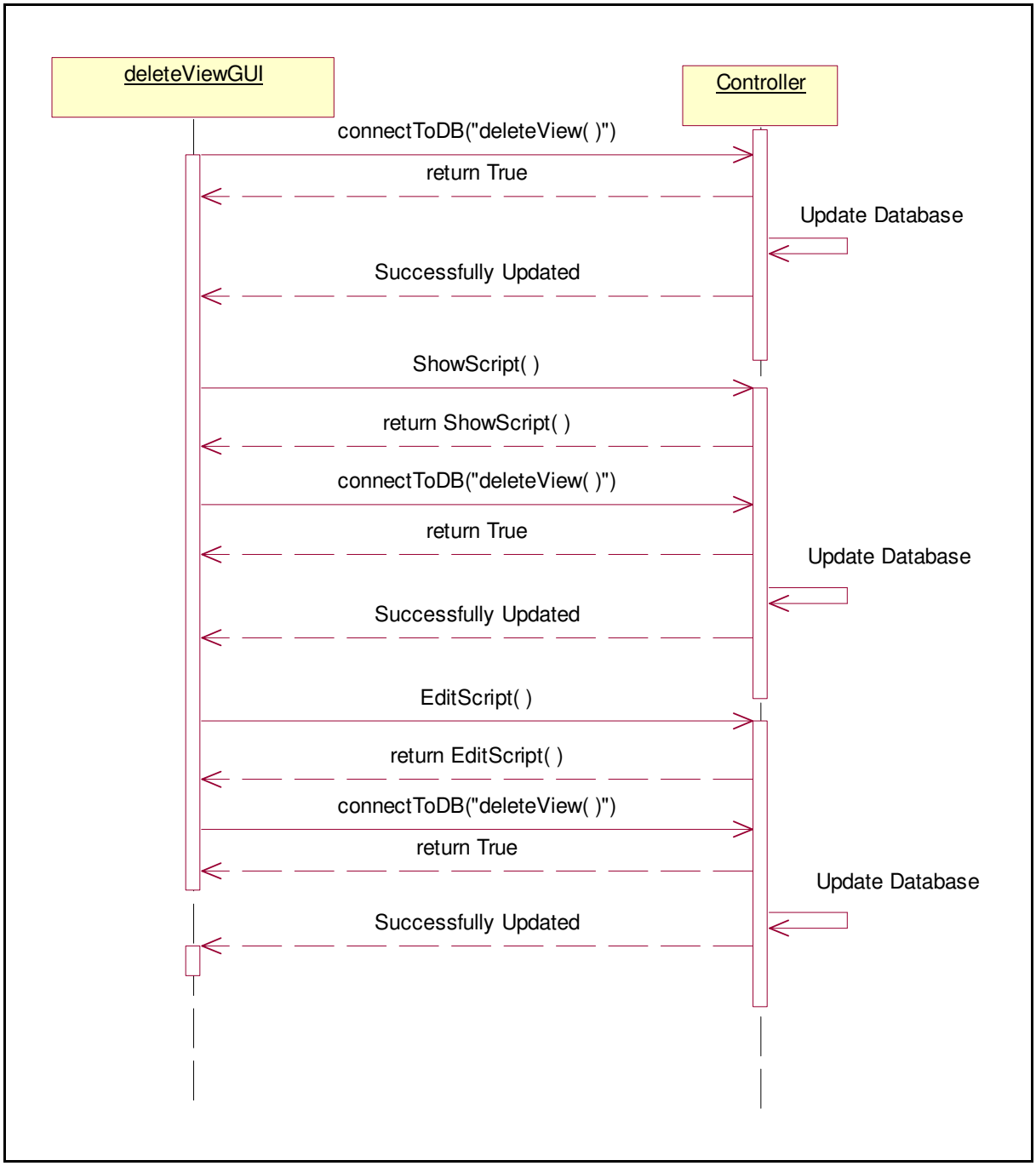
4.5.13 Sequence Diagram for Delete Index



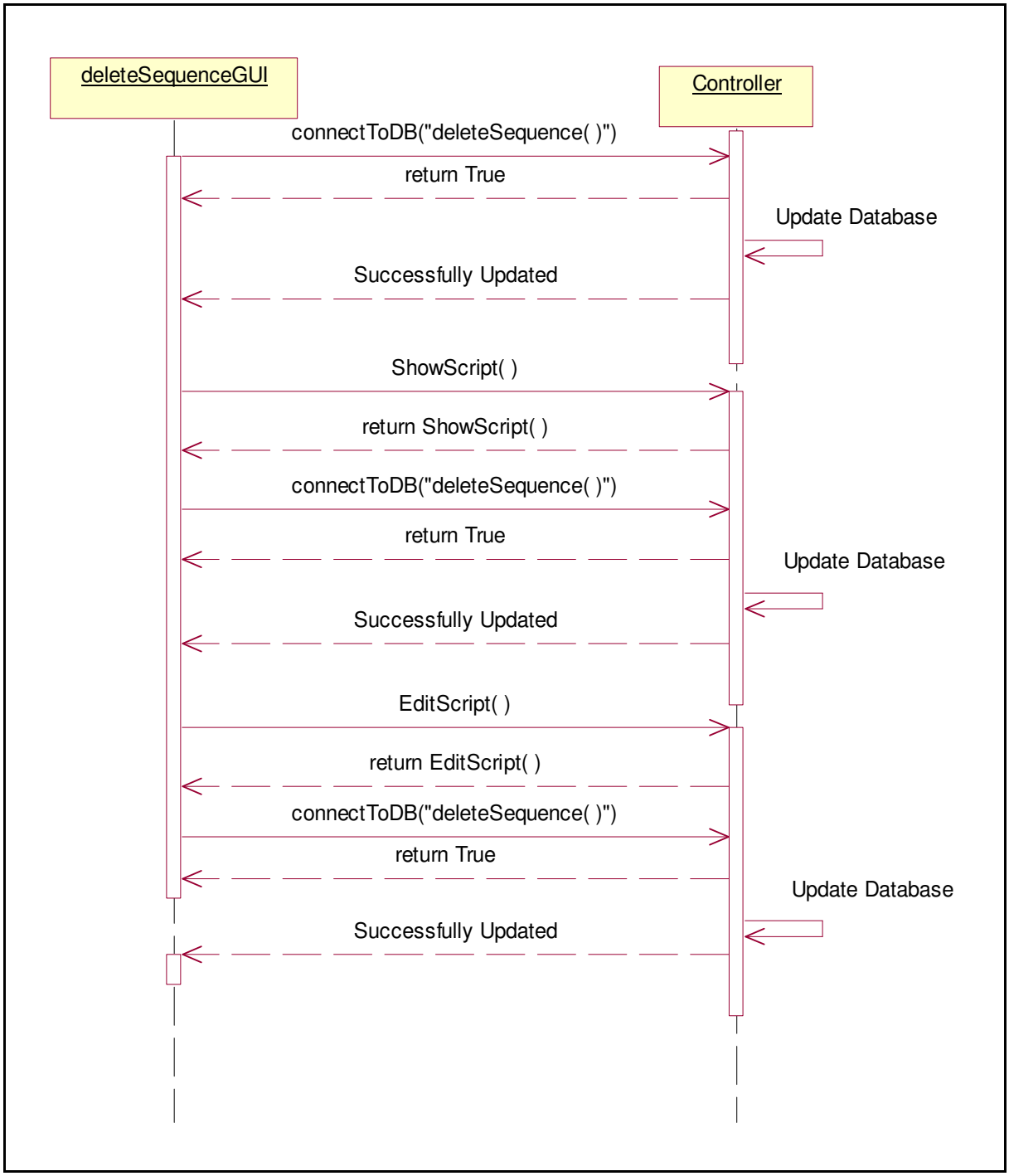
4.5.14 Sequence Diagram for Delete Materialize View



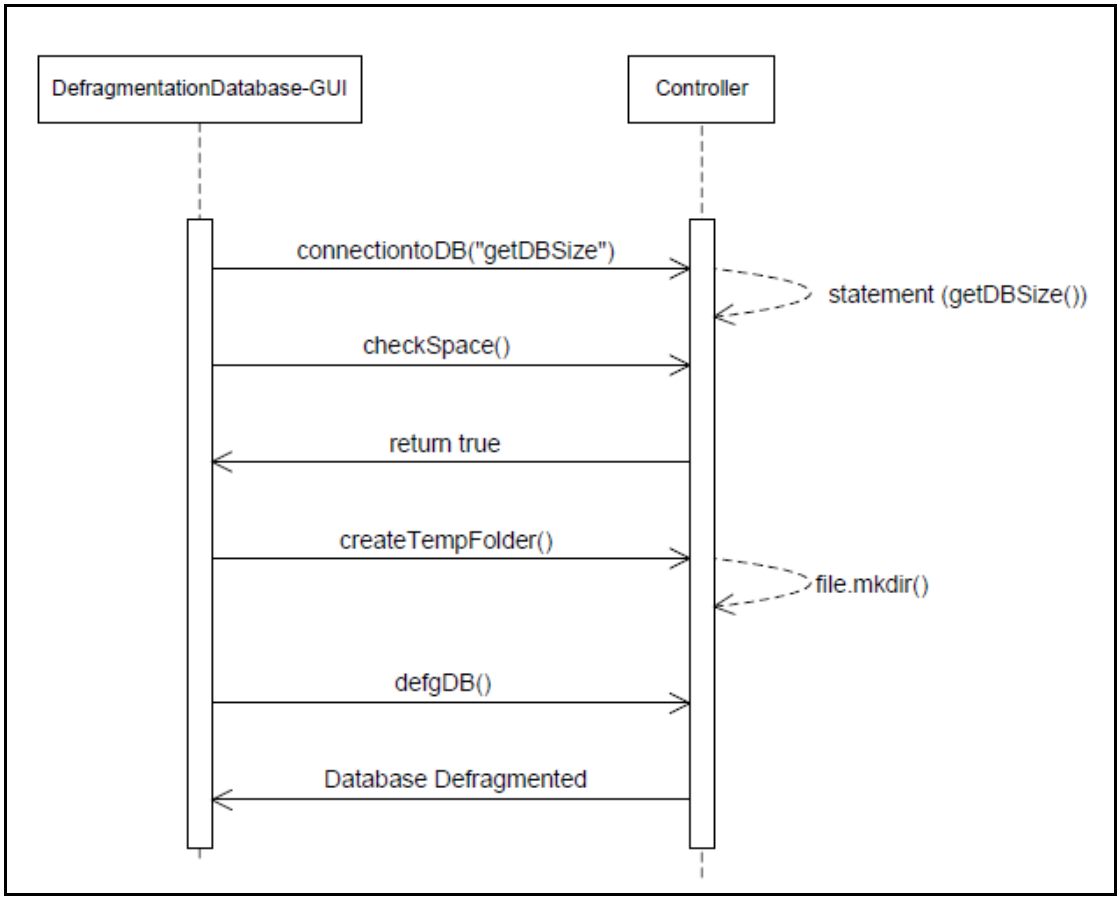
4.5.15 Sequence Diagram for Delete View



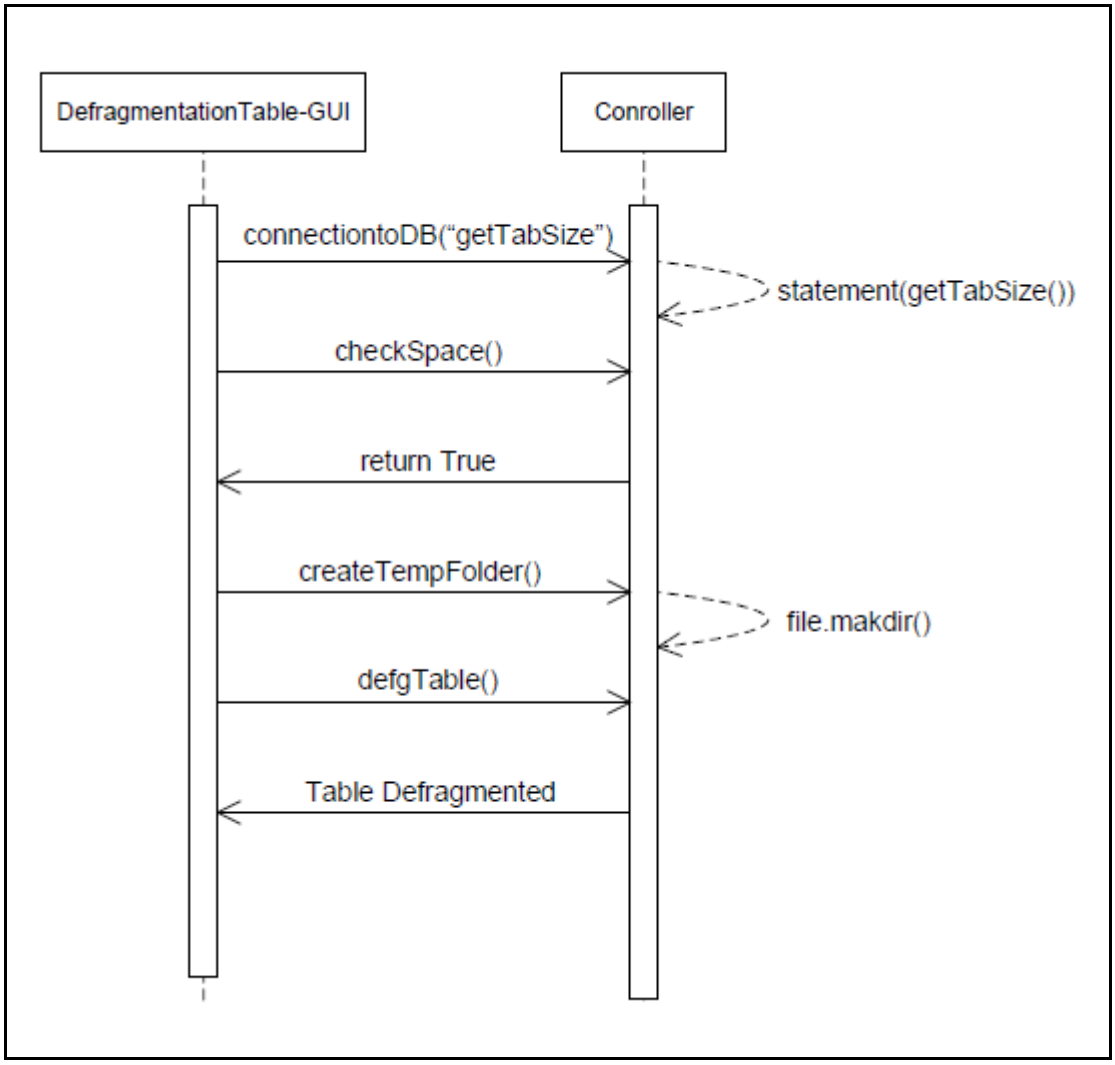
4.5.16 Sequence Diagram for Delete Sequence



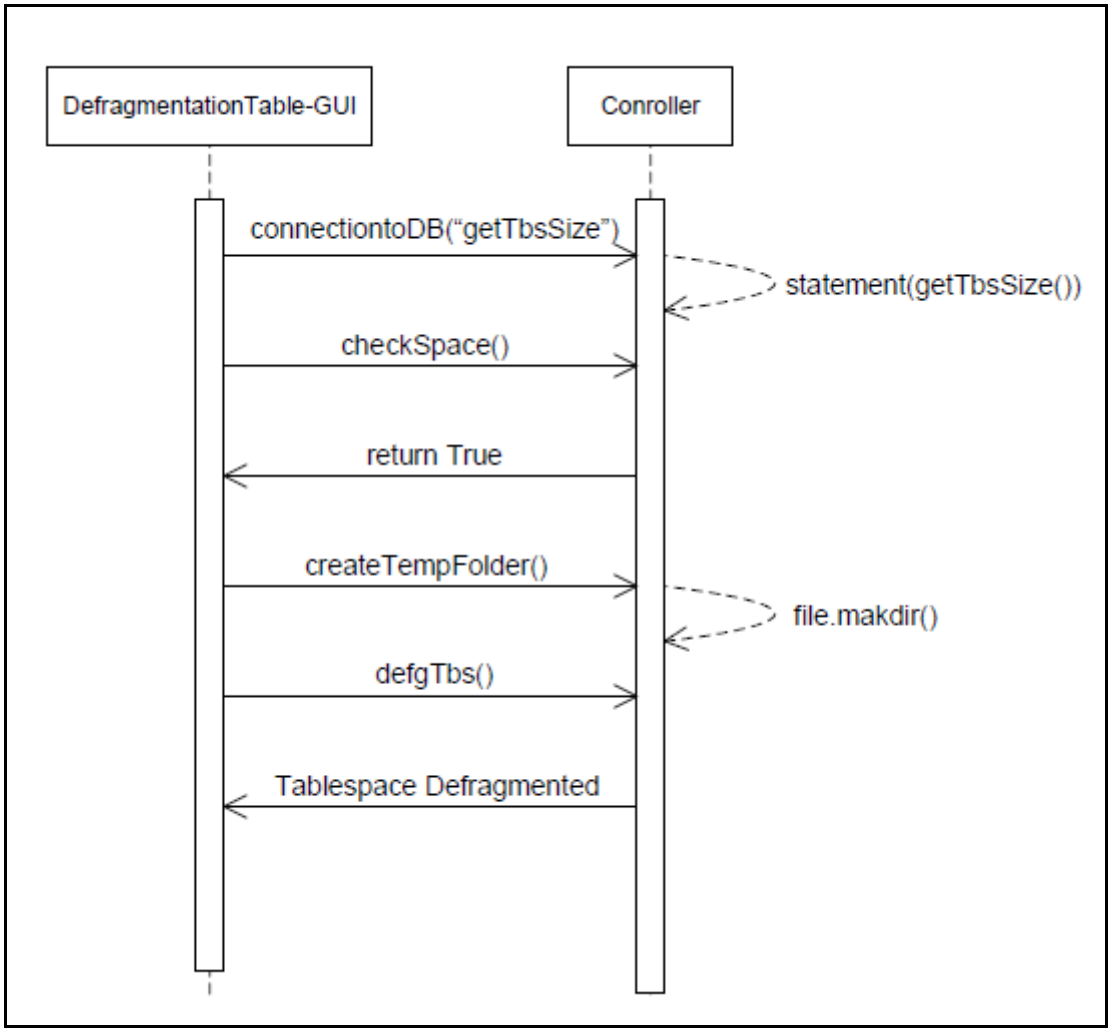
4.5.17 Sequence Diagram for Defragnmentation Database



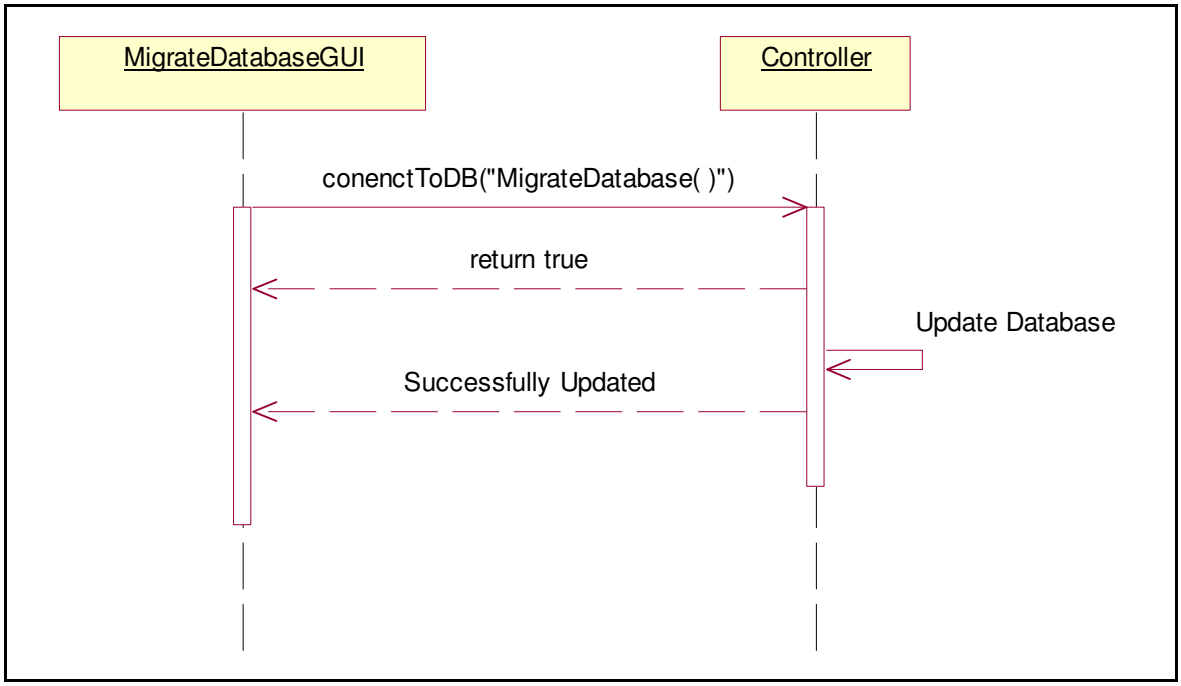
4.5.18 Sequence Diagram for Defragnmentation Table



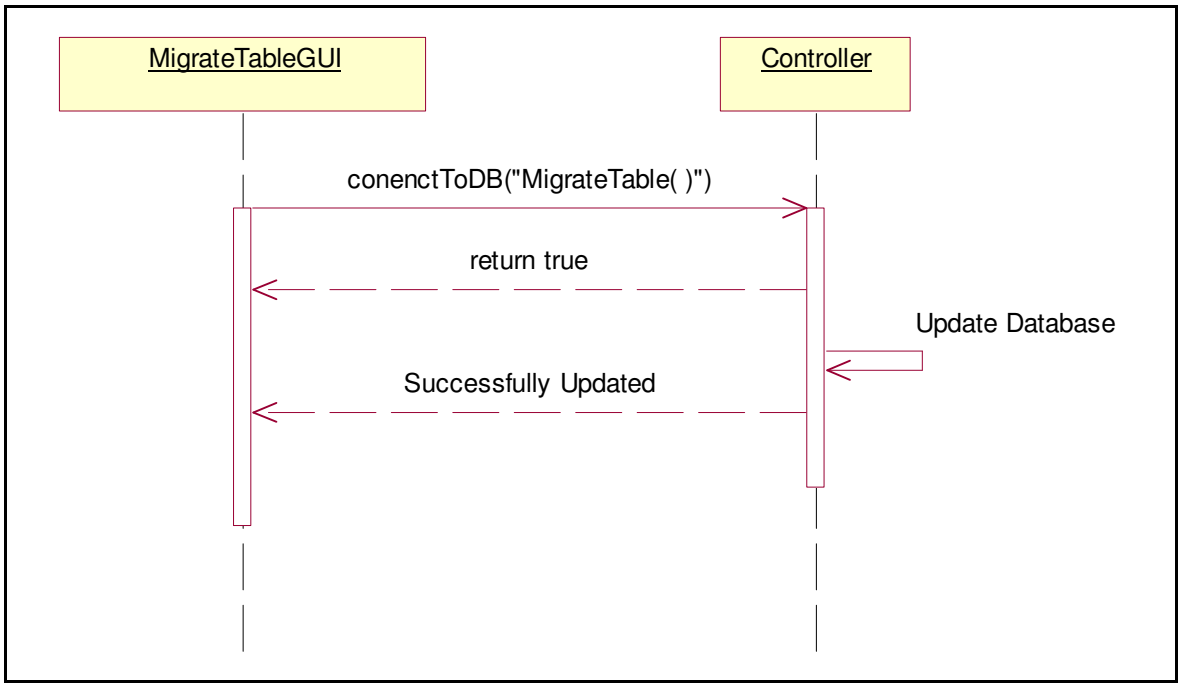
4.5.19 Sequence Diagram for Defragnmentation Tablespace



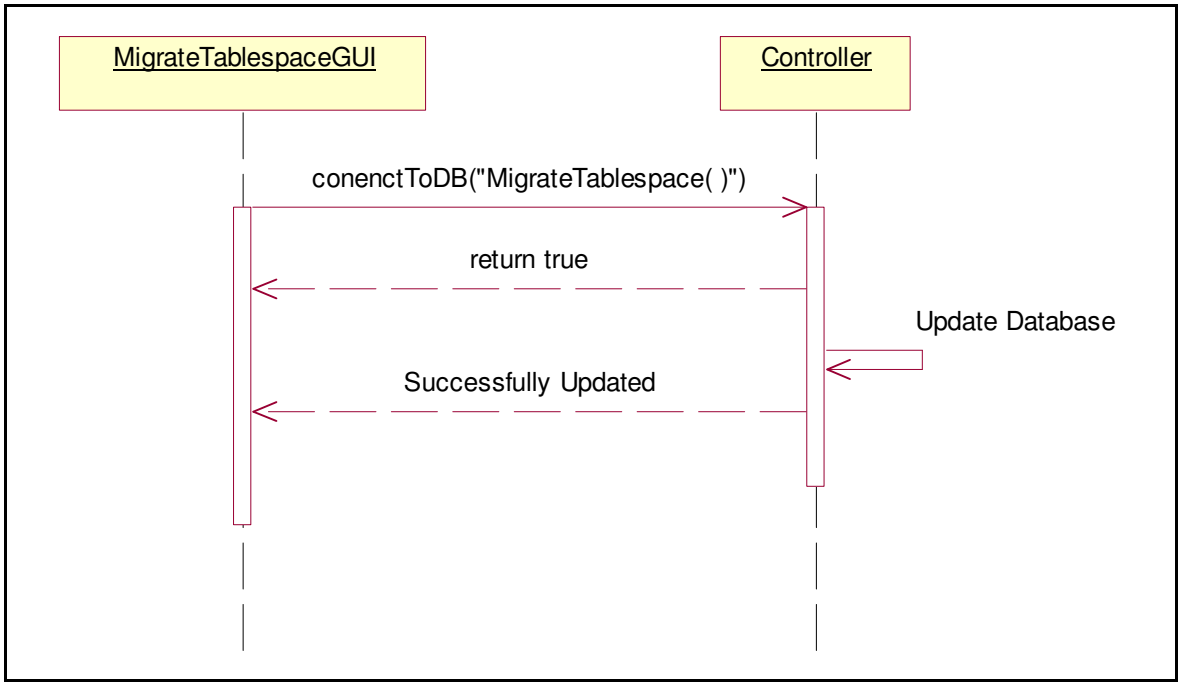
4.5.20 Sequence Diagram for Migration Database



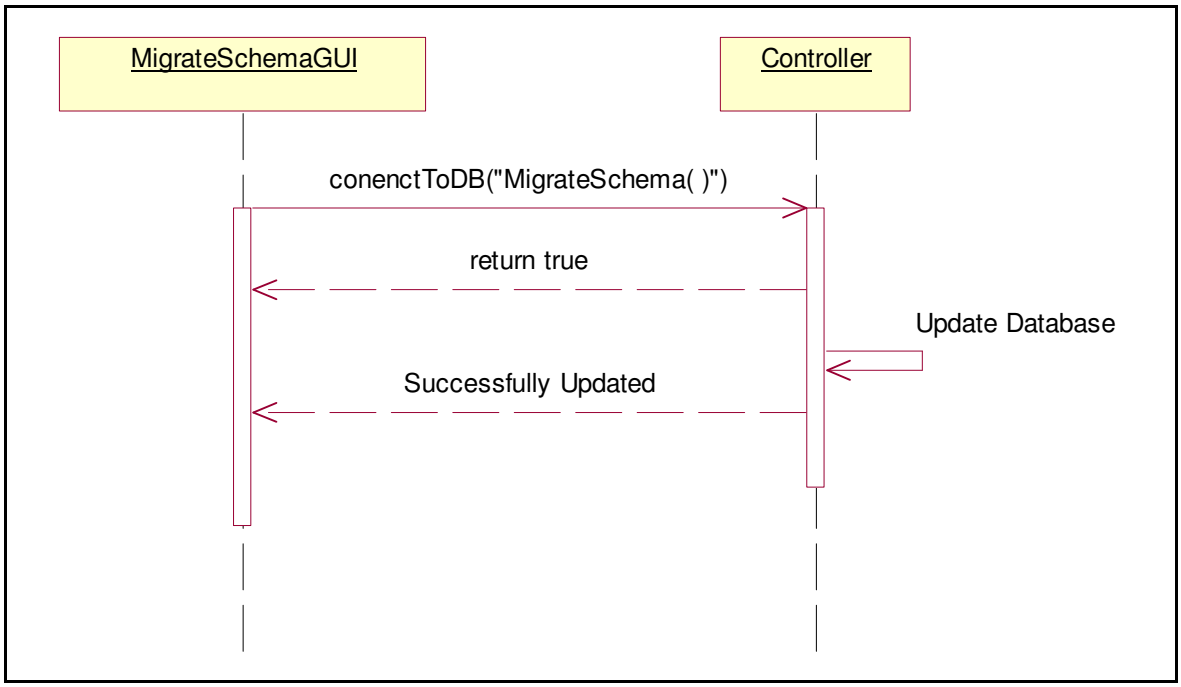
4.5.21 Sequence Diagram for Migration Table



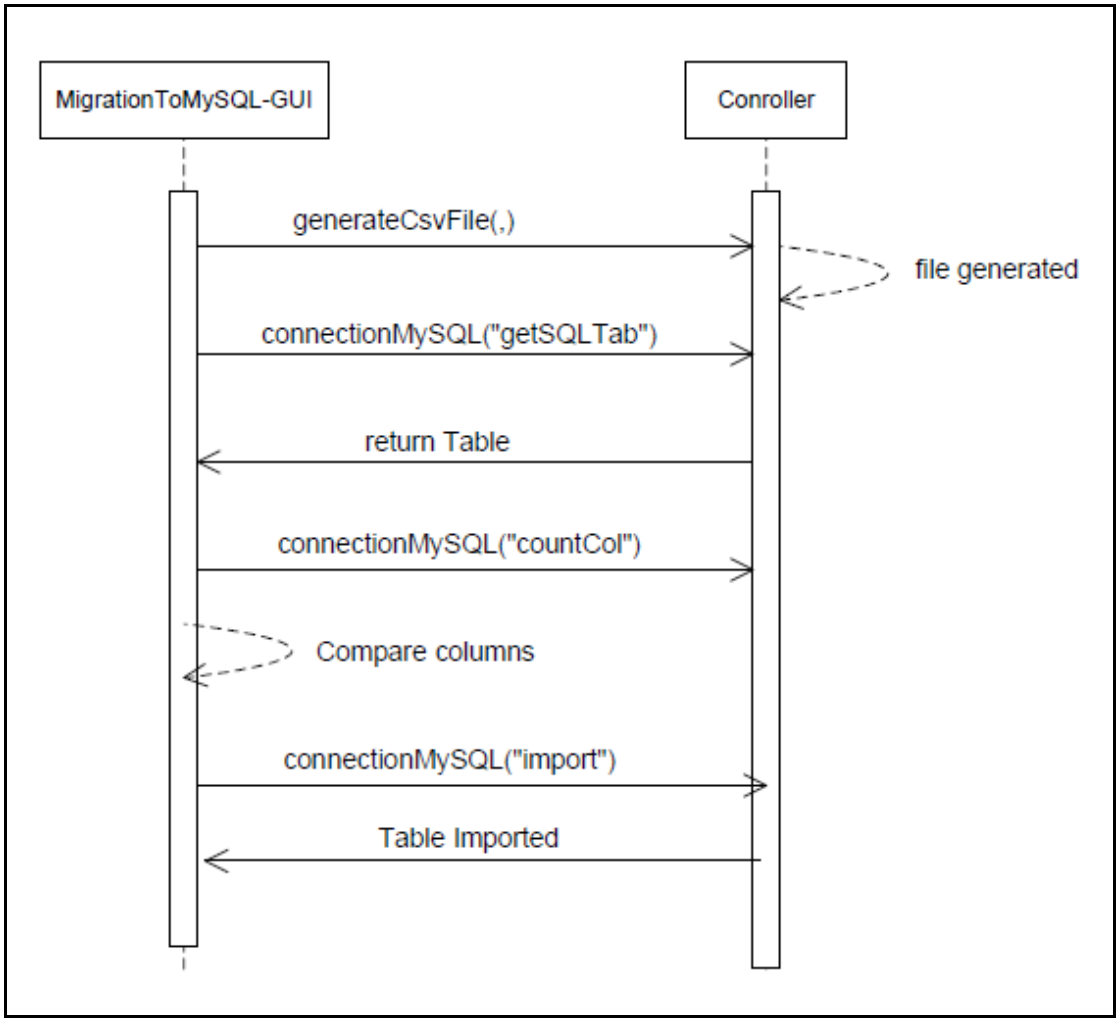
4.5.22 Sequence Diagram for Migration Tablespace



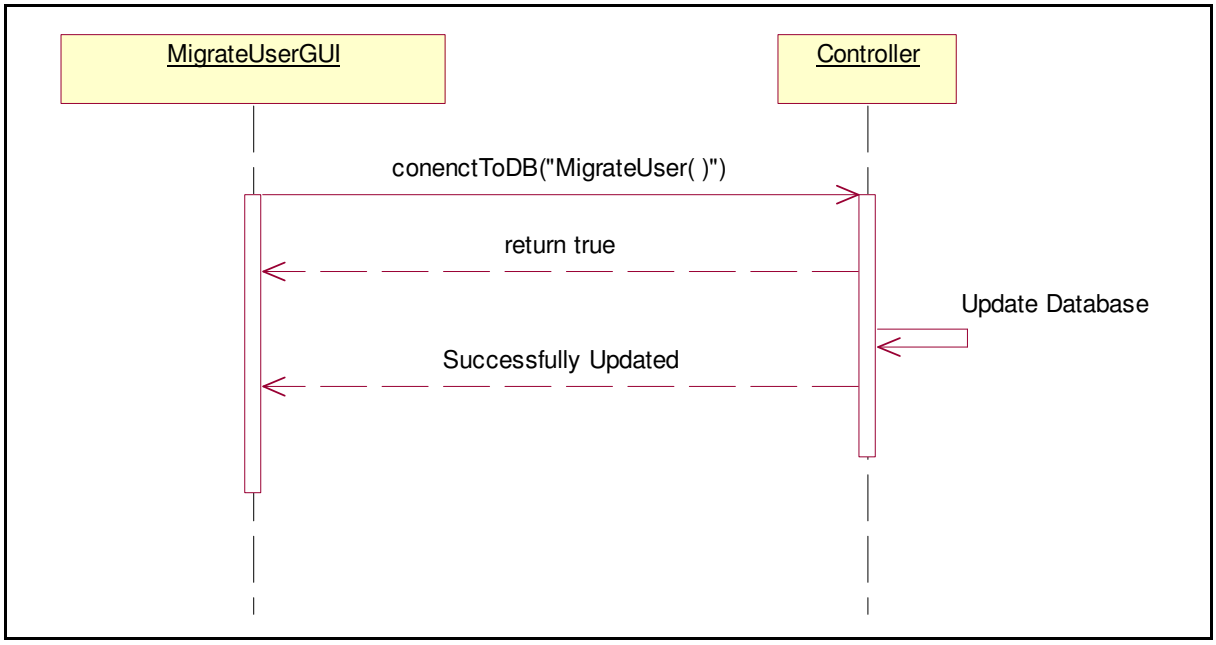
4.5.23 Sequence Diagram for Migration Schema



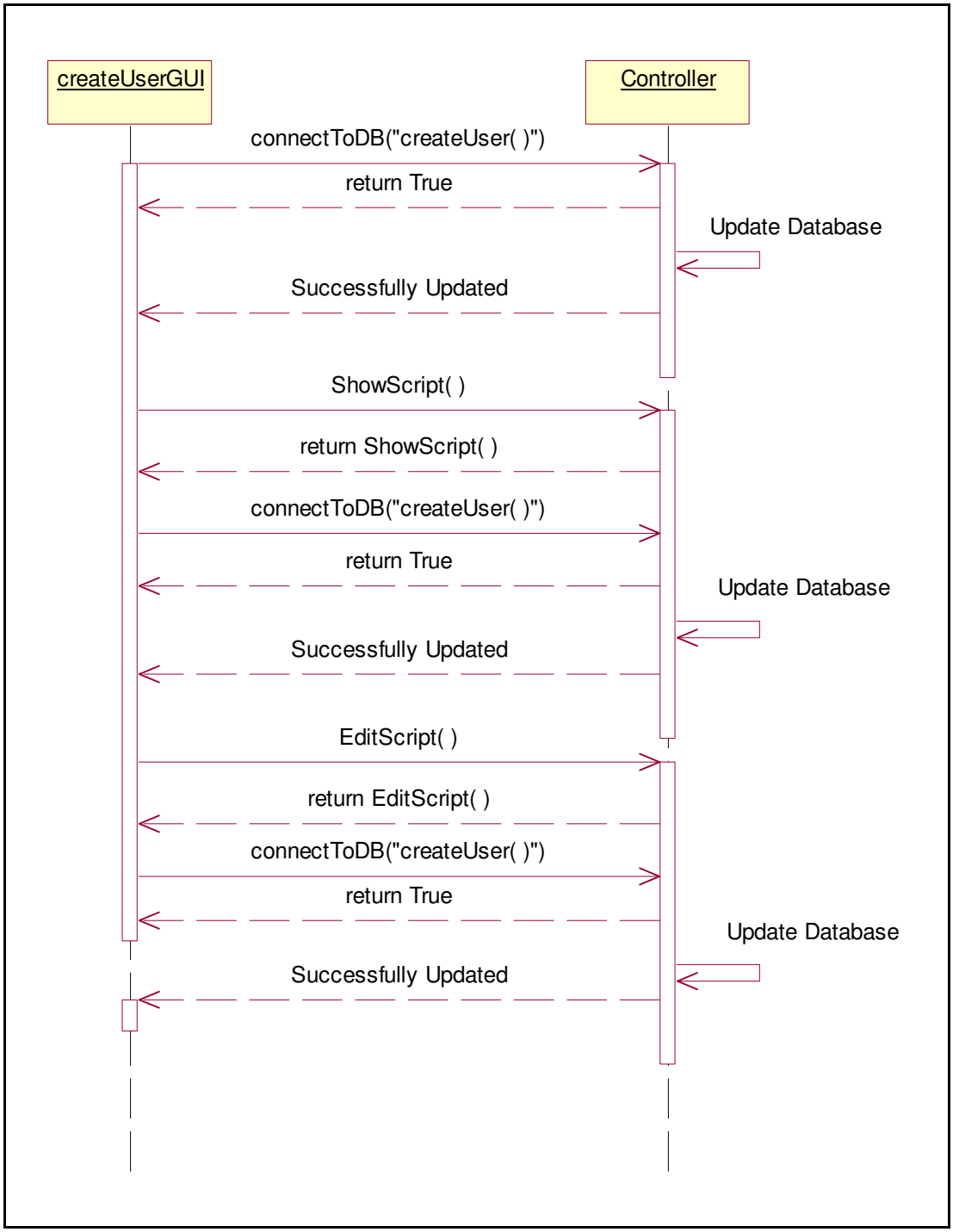
4.5.24 Sequence Diagram for Migration To MySQL



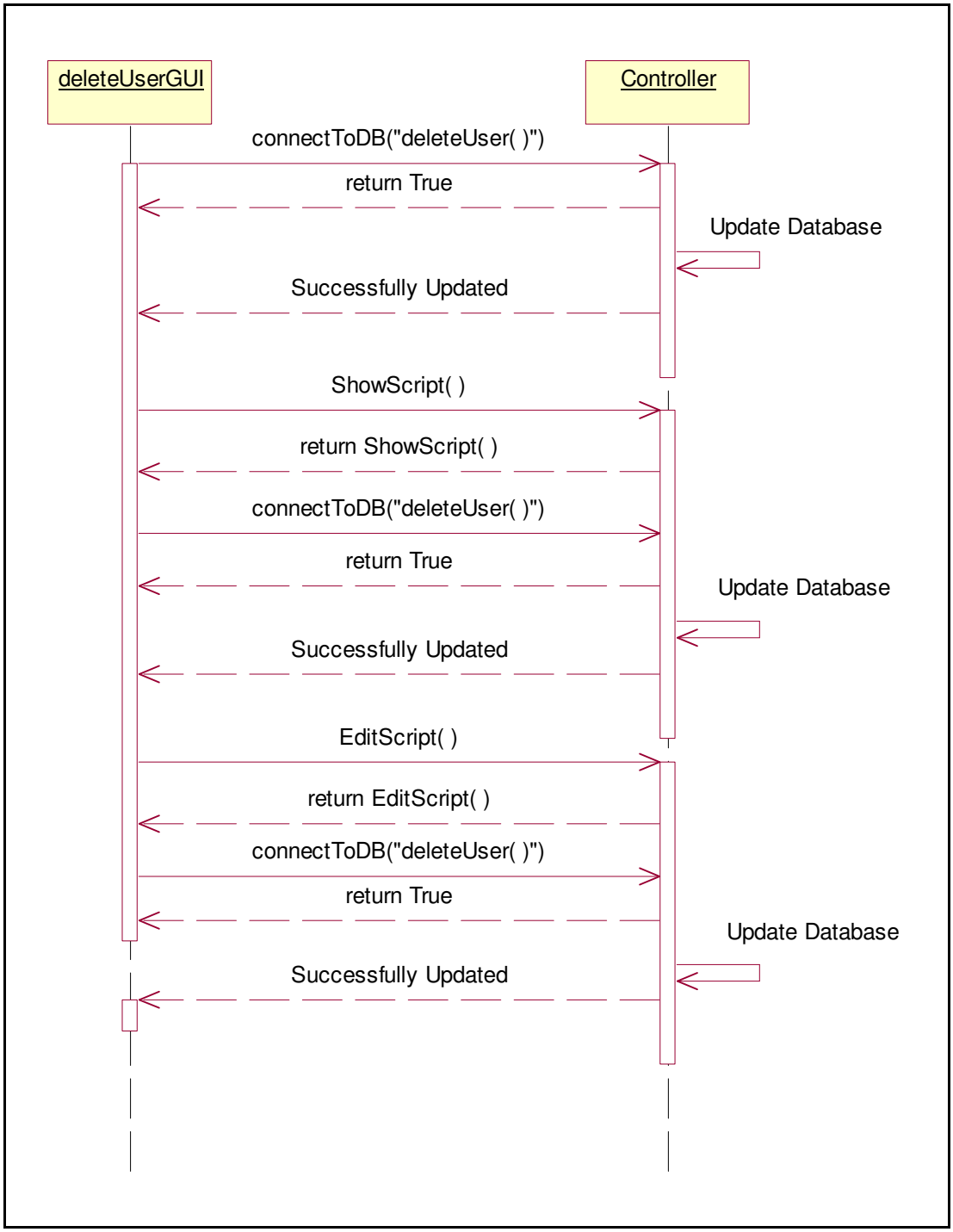
4.5.25 Sequence Diagram for Manage Password



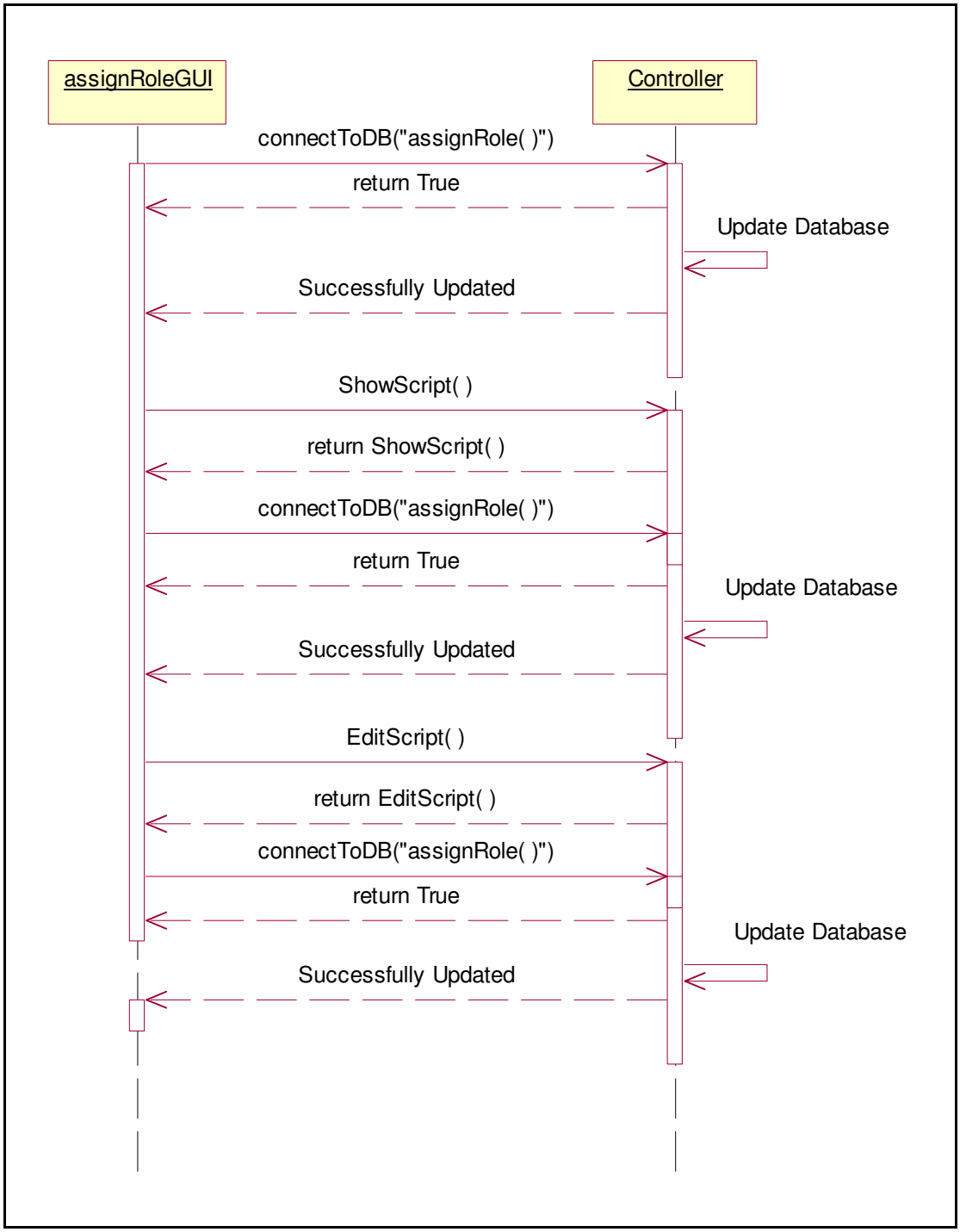
4.5.26 Sequence Diagram for Create User



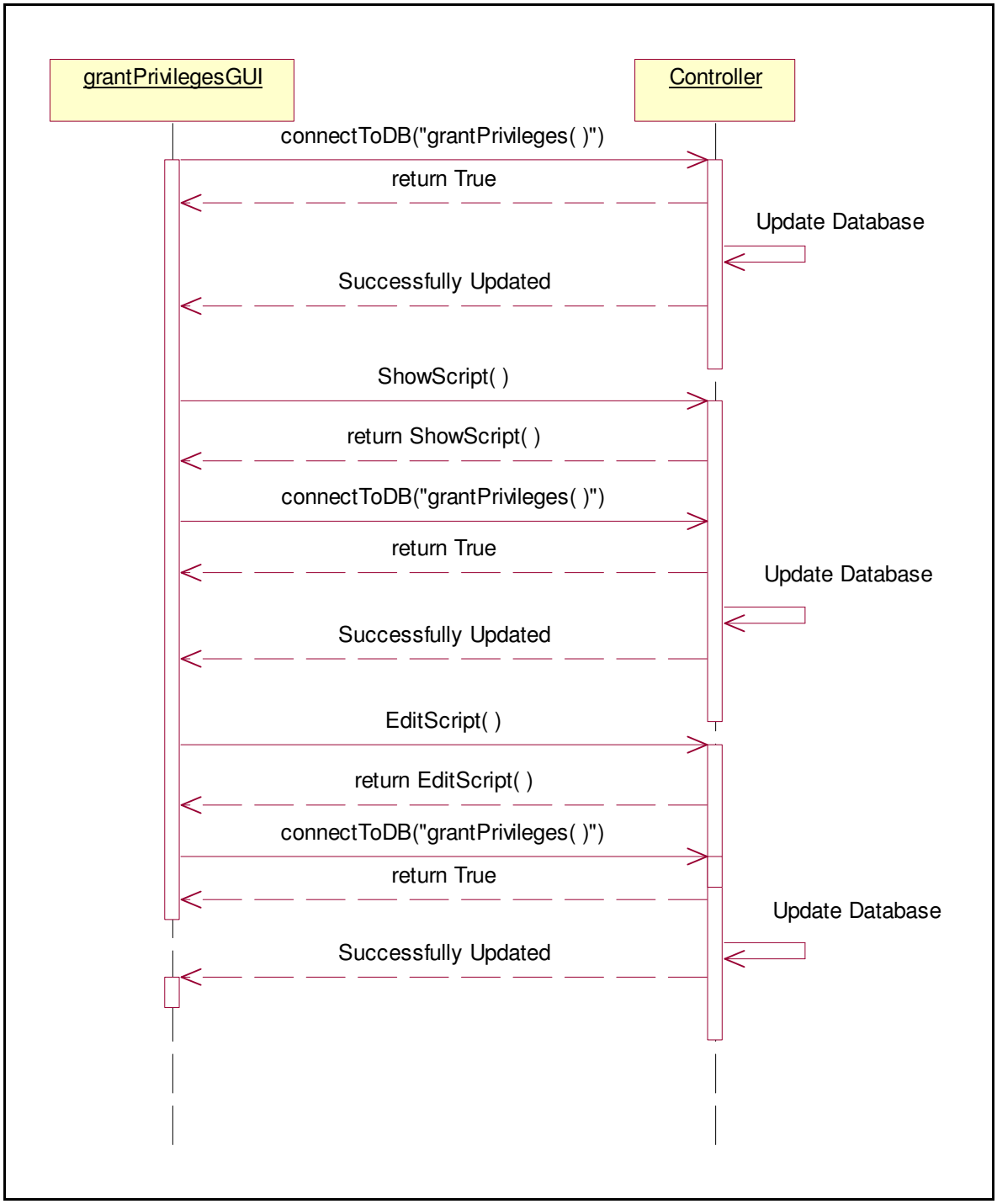
4.5.27 Sequence Diagram for Delete User



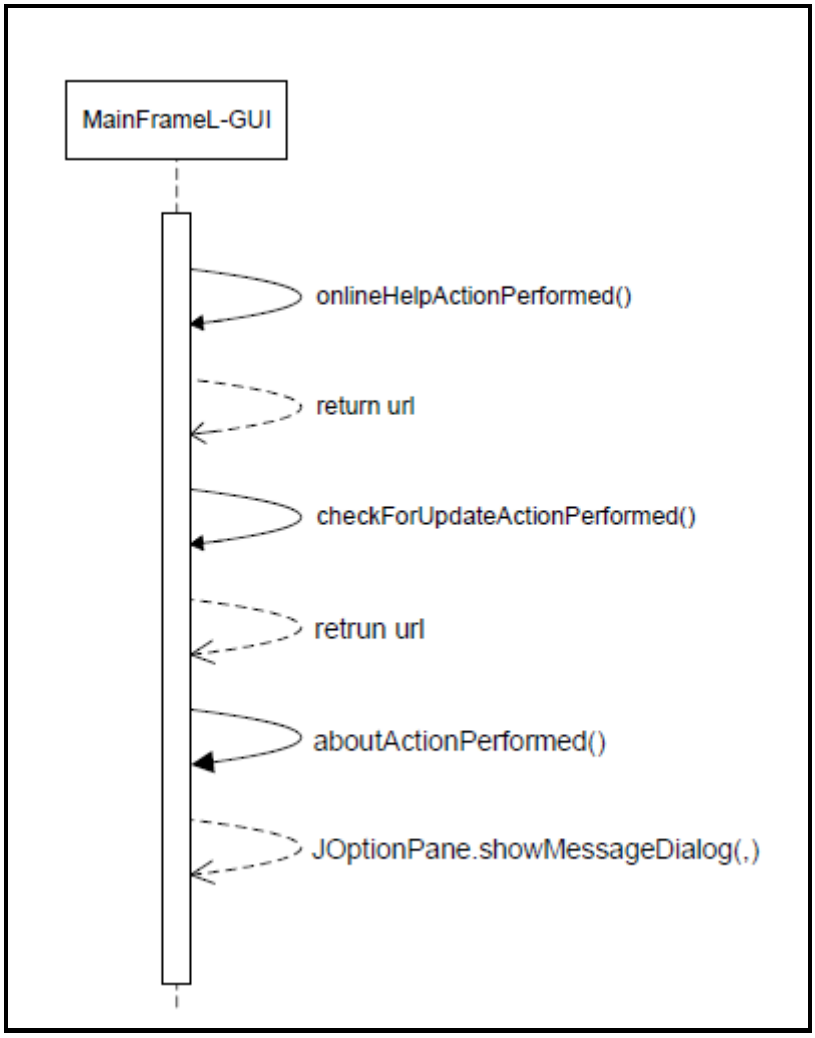
4.5.28 Sequence Diagram for Assign Role



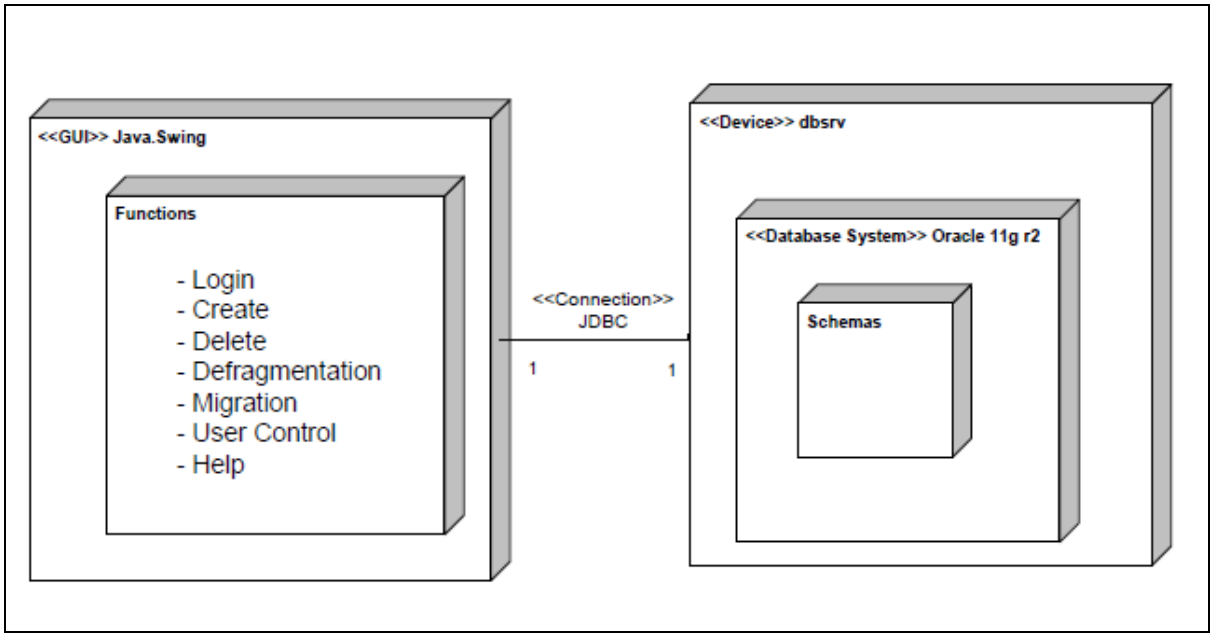
4.5.29 Sequence Diagram for Grant Privileges



4.5.30 Sequence Diagram for Help



4.6 Deployment Diagram



4.6.1 Deployment Environment

4.6.1.1 Nature of Project

This project is based on objective and makes development based on plan. And also record problem when encounter and provide solution how developer solved. This project is information system characteristics and provide user-friendly interface to maintain their own database without using coding. The project develops on Window platform, and will make improvement multi-platform (such as Linux and Mac OSX) based on time constrain. The project required user-friendly interface to manage the database. Therefore developers are using Java language for GUI and oracle (SQL plus) for database management.

4.6.1.2 Development and Support Environment

According to time consumption, the software was going to develop on Windows 7. As for optional, developers will trying to write in flexible way to on vary platforms such as Linux, Mac, and so on. All of the development software is available at the

http://zim.cs.uow.edu.au:50321/~cs321jg1a/index.html . The minimum system hardware specification to run:

Hardware	Requirement
CPU	Pentium 1 (GHz) minimum;
Memory (RAM)	512 megabytes (MB) minimum; 1 GB or more recommended
Hard disk space	20 MB minimum

Software	Requirement
Operating System	Microsoft Windows XP minimum
Oracle	11g R2
MySQL	5.5.8
NetBeans	7.3
JDK	1.7
Classes 1.2.jar and Ojbc6.jar	Oracle connection
Sigar	1.6.4
MySQLconnector	J5.1.26
Swingx	0.9.2

If the system is not meeting the above requirement will not able to run the application. Therefore, strongly advices to review the system requirement before download the software to install.

5 Project Management and Controlling

5.1 Project Controlling

Project manager controlling the progress of each activity based on project plan (Gantt chart). Before reach each millstone project manager look through each member they

are working process can finish on time or not. If member who can't finish on time for his/her task, project manager looking for other members who can finish his/her task and let him/her to help those member who are delay his/her task. If no one free to help his/her check his/her task is on high level requirement or not if it is not high level requirement let his/her to tidy up his/her task to complete the project. And try to take extra time (e.g.: work on Saturday and Sunday) by doing this way we can avoid from project delay.

We split our resources between implementation and design therefore maximizing our resource allocation. In addition, requirements will be split into base, high, mid and low level requirements which will be designed and implemented in the order. The main requirement is that GUI design and management database, for GUI we will develop using Java and for database management we using SQL plus.

5.2 Method of Communication

The project teams and supervisor will communicate using following methods:

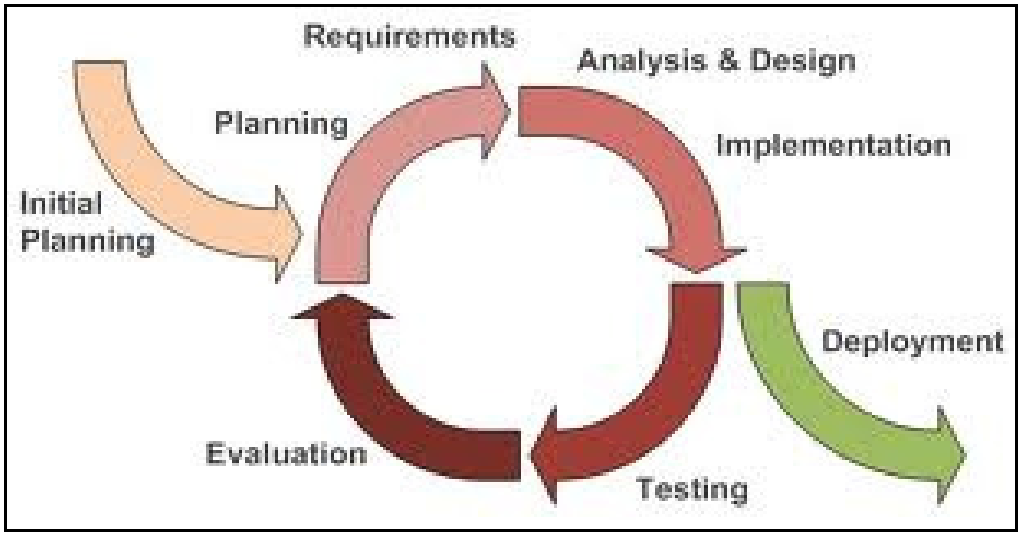
- At least 1 hour meeting per week with teams members. Extra meeting will require based on our process.
- At least 1 hour meeting per week with supervisor (Janusz).
- Facebook teams to communicate outside of meeting.
- Google code and drive to share code and document as subversion.

5.3 Design Methodology

The main software structure for implementing Crystal Cockpit is incremental and iterative model. System has designed, implemented and tested incrementally until it produces the satisfied output. Functionality of the program has been divided to sub functions. When the sub functions are implemented and tested they are combined together and created a larger component. Next step is to analysis, design, code, and test the next iterative.

To explain in detail, at the starting point we have designed a basic user interface so we can add functionalities to the system and discuss them with user. We are starting from connecting program to database and create a simple Tablespace. In this point we

do not consider immigration or defragmentation. Then we add the other features for creation function. Each database creation element is sub-function. We start from container elements such as first Tablespace and then assign new Table to that new Tablespace. Now we can test it and find the bugs and fix them and finally evaluate it if it is needed. This can be an iterative loop for us. We show it to supervisor to make sure the system is meeting user requirement. Next step is for deletion and then migration and defragmentation is final step for high priority functions.



The best benefit of this methodology is, if there is an error occurs within the program the coder knows what the last changes were so tracking bugs is easier. Beside this for testing also we can test smaller component and then combine them and test the main component. It can save a lot of time for implementation and testing part. After finishing high priority objectives we can add evaluated GUI and in same time we add User Control feature with medium priority. Now the problem rises with implementing code and evaluation of GUI. We need a pattern to separate user's interaction from representation of information. If we change user interface we don't want to be worry for connection with oracle. The best solution is that we are dividing our code to three parts.

5.4 Development and Quality Standards

All the code is to be tested by each members and users. All the design, documentation will conform to the teams and supervisor and move to implementation. All changes will put into the version control not only design document but also codes via

subversion. For the testing project, we are using unit testing, conforming to standard of the project.

Functionality: In the requirement specification, each function separated by priority. And members are implementing from base to low according to user requirement and testing each stage follow when each function finish.

Reliability: Implemented by functional requirement level and testing is follow according to implementation. When happen error in testing part record the error and solve it straight away.

Usability: Try to design the GUI as possible as simple. And also it is included all the functions, that requirement by user. User can understand how to use when his/her see the GUI design. And also provide help option to help user to use system easily.

Efficiency: Testing system respond time according to user uploaded file/folder. Try to reduce response time as possible as system does. And implement the memory usage of the program, as less as possible.

Maintainability: Using Model View Controller to make sure all the part of the coding is clean and maintainable. Each of the stage is tested by JUnit standard. And also recorded when error is occurring and prevent next time to happen same error again.

5.5 Project Risk Management

Impact Category	Definition
Critical	If it occurred, it can cause project fail.
Serious	If it occurred, can cause project late and may not meet the requirement.
Moderate	If it occurred, can cause project late but still can meet the requirement.
Minor	If it occurred, can cause project late or may not late. And requirement still can be achieved.

Negligible	If it occurred, no effect on project schedule and requirement.

Probability Category	**Definition**
Very High	It can happen 100%
High	It can happen 80%
Medium	It can happen 50%
Low	It can happen 20%
Very Low	It can happen 0%

Name	**Team members missing requirement**
Probability	Medium
Impact	Serious
Avoid risk	Member who missing the requirement, should discuss with other members and supervisor to know clearly all the requirements. Give more time spend to understand the requirement.

Name	**Team members Leaving**
Probability	Low
Impact	Serious
Avoid risk	Member, who wants to leave from the teams, should finish all his/her tasks or before leaving should find someone to replace.

Name	**Team members do not know how to develop**
Probability	Low
Impact	Critical
Avoid risk	Member who don't know how to start to develop the software. Should read books that related to project and learn new thing if member don't know. Spend more time on searching resources.

Name	**Member follow the project time line**
Probability	Low
Impact	Moderate
Avoid risk	Member should know their own task and its deadline based on time line of the project. If member can't finish his/her task based on time line should let other members to know to cover his/her part or discuss difficulties of his/her problem. Other members give more time to cover quickly.

Name	**Change Project Specification**
Probability	Medium
Impact	Moderate
Avoid risk	Make project breakdown structure easily to modifiable.

Name	**Supervisor not happy with the project**
Probability	Low
Impact	Critical

Avoid risk	If supervisor not happy with project, team should communicate with supervisor frequently and make changes quickly based on his requirement. And inform him go get high result.

Appendix

Coding Structure

Example -> Login()

```
package GUI;

import Controller.*;

import java.awt.Image;

import java.awt.event.KeyEvent;

import java.io.IOException;

import java.util.ArrayList;

import javax.swing.*;

@SuppressWarnings("serial")

public class Login extends javax.swing.JFrame {


   public Controller thisController = new Controller();

   public ArrayList<String> privsList=new ArrayList<String>();


   public Login() throws IOException {


      CreateTable.i = 1;

      CreatView.i = 1;

      DeleteMaterializeView.i = 1;

      DeleteTable.i = 1;

      DeleteView.i = 1;

      CreatMaterializeView.i=1;

      DeleteSequence.i =1;

      CreatSequence.i=1;

      Welcome.i=1;
```

```
    initComponents();
}



public Image Crystal() {
    ImageIcon icon = new ImageIcon(getClass().getResource("/Resources/Crystal Cockpit.png"));
    Image img = icon.getImage() ;
    return img;
}
/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {


    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    username = new javax.swing.JTextField();
    password = new javax.swing.JPasswordField();
    loginBtn = new javax.swing.JButton();
    jLabel3 = new javax.swing.JLabel();
    dbName = new javax.swing.JTextField();
    jLabel4 = new javax.swing.JLabel();
    url = new javax.swing.JTextField();
    jLabel5 = new javax.swing.JLabel();
    port = new javax.swing.JTextField();
```

```
jLabel6 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("Crystal Cockpit");
setBackground(new java.awt.Color(255, 255, 255));
setIconImage(Crystal());
setLocationByPlatform(true);

jLabel1.setText("Username:");

jLabel2.setText("Password:");

username.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyPressed(java.awt.event.KeyEvent evt) {
        usernameKeyPressed(evt);
    }
});

password.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyPressed(java.awt.event.KeyEvent evt) {
        passwordKeyPressed(evt);
    }
});

loginBtn.setText("Login");
loginBtn.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        OpenWelcome(evt);
    }
});
```

```
loginBtn.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyPressed(java.awt.event.KeyEvent evt) {
        pressEnter(evt);
    }
});

jLabel3.setText("Database Name:");

dbName.setText("orcl");
dbName.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyPressed(java.awt.event.KeyEvent evt) {
        dbNameKeyPressed(evt);
    }
});

jLabel4.setText("host");

url.setText("localhost");
url.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyPressed(java.awt.event.KeyEvent evt) {
        keyPressedAction(evt);
    }
});

jLabel5.setText("Port");

port.setText("1521");
port.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyPressed(java.awt.event.KeyEvent evt) {
        portKeyPressed(evt);
```

```
        }
    });

    jLabel6.setBackground(new java.awt.Color(255, 255, 255));
    jLabel6.setFont(new java.awt.Font("Segoe UI", 0, 48)); // NOI18N
    jLabel6.setForeground(new java.awt.Color(0, 0, 255));
    jLabel6.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
    jLabel6.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Resources/Crystal
Cockpit.png"))); // NOI18N
    jLabel6.setText("Crystal Cockpit");
    jLabel6.setToolTipText("");
    jLabel6.setIconTextGap(5);

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
            .addContainerGap(162, Short.MAX_VALUE)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addGap(71, 71, 71)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(jLabel1)
                        .addComponent(jLabel2)
                        .addComponent(jLabel3)
                        .addComponent(jLabel4)
                        .addComponent(jLabel5))
                    .addGap(41, 41, 41)
```

```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(username, javax.swing.GroupLayout.PREFERRED_SIZE, 138, javax.swing.GroupLayout.PREFERRED_SIZE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                    .addComponent(password, javax.swing.GroupLayout.DEFAULT_SIZE, 138, Short.MAX_VALUE)
                    .addComponent(dbName, javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(url, javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(port, javax.swing.GroupLayout.Alignment.TRAILING)))
                .addGap(57, 57, 57))
            .addComponent(jLabel6, javax.swing.GroupLayout.Alignment.TRAILING))
        .addGap(152, 152, 152))
    .addGroup(layout.createSequentialGroup()
        .addGap(321, 321, 321)
        .addComponent(loginBtn)
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(32, 32, 32)
            .addComponent(jLabel6)
            .addGap(29, 29, 29)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(username, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGroup(layout.createSequentialGroup()
                    .addComponent(jLabel1)
                    .addGap(26, 26, 26)
```

```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(password,          javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(26, 26, 26)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel3)
                    .addComponent(dbName,            javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel4)
                    .addComponent(url,               javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(15, 15, 15)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel5)
                    .addComponent(port,              javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))
            .addGap(18, 18, 18)
            .addComponent(loginBtn)
            .addContainerGap(47, Short.MAX_VALUE))
        );
        setSize(new java.awt.Dimension(717, 430));
        setLocationRelativeTo(null);
    }// </editor-fold>
    private void OpenWelcome(java.awt.event.ActionEvent evt) {
        actions();
```

```
}
private void pressEnter(java.awt.event.KeyEvent evt) {
    if(evt.getKeyCode()==KeyEvent.VK_ENTER){
        loginBtn.doClick();
    }
}
private void keyPressedAction(java.awt.event.KeyEvent evt) {
    if(evt.getKeyCode()==KeyEvent.VK_ENTER){
        loginBtn.doClick();
    }
}
private void portKeyPressed(java.awt.event.KeyEvent evt) {
    if(evt.getKeyCode()==KeyEvent.VK_ENTER){
        loginBtn.doClick();
    }
}
private void dbNameKeyPressed(java.awt.event.KeyEvent evt) {
    if(evt.getKeyCode()==KeyEvent.VK_ENTER){
        loginBtn.doClick();
    }
}
private void usernameKeyPressed(java.awt.event.KeyEvent evt) {
    if(evt.getKeyCode()==KeyEvent.VK_ENTER){
        loginBtn.doClick();
    }
}
private void passwordKeyPressed(java.awt.event.KeyEvent evt) {
    if(evt.getKeyCode()==KeyEvent.VK_ENTER){
        loginBtn.doClick();
    }
```

```
    }
    private void actions(){
        thisController.setLoginInfo();
        //check successfull login or not
        Boolean check = thisController.connectiontoDB("Login");
        if(check == true){
            //if login successful open main frame
            new MainFrame().setVisible(true);
            dispose();
        }else{
            //if not successful show errormsg
            JOptionPane.showMessageDialog(this, thisController.errorMsg,
                    "Error",JOptionPane.ERROR_MESSAGE);
        }
    }

    // Variables declaration - do not modify
    public static javax.swing.JTextField dbName;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JButton loginBtn;
    public static javax.swing.JPasswordField password;
    public static javax.swing.JTextField port;
    public static javax.swing.JTextField url;
    public static javax.swing.JTextField username;
}
```