

DPPL-xx

SOFTWARE DESIGN DESCRIPTION

eLab

for:

RPLGDC Laboratory

Prepared by:

Aulia Rahman Arif Wahyudi (1301194195)

Putu Budi Sukarya (1301194252)


Khaidir Mauladan (1301192327)

Anggi Rodesa Sasabella (1301193161)

Informatics Study Program

Faculty of Informatics

Jl. Telekomunikasi 1, Dayeuhkolot Bandung

	Informatics Study Program Telkom University	Document Number		Page
		<i>DPPL-xx</i> <xx:no grp>		<#>/<number #
		Revision	<revision number>	<i>Date: <fill in date></i>

LIST OF CHANGES

Revision	Description
A	
B	
C	
D	
E	
F	
G	

INDEX DATE	-	A	B	C	D	E	F	G
Written by								
Review by								
Approve d by								

List of Changes

Pages	Revised	Pages	Revised

Table of Contents

1. Introduction	6
1.1. Purpose of Document Writing	6
1.2. Scope of Problem	6
1.3. Definitions and Terms	6
1.4. References	7
1.5. Systematics of Discussion	7
2. Description of Global Design	8
2.1. Implementation Environment Design	8
2.2. Architectural	8
2.3. Component Description	8
3. Detailed Design	9
3.1. Use Case Realization	9
3.1.1. Use Case: Register an account	9
3.1.1.1. Class Identification	10
3.1.1.2. Sequence Diagram	10
3.1.1.3. Class Diagram	10
3.1.2. Use Case: View an Archive	10
3.1.2.1. Class Identification	10
3.1.2.2. Sequence Diagram	11
3.1.2.3. Class Diagram	11
3.1.3. Use Case: Search an Archive	11
3.1.3.1. Class Identification	11
3.1.3.2. Sequence Diagram	12
3.1.3.3. Class Diagram	13
3.1.4. Use Case: Save an Archive	13
3.1.4.1. Class Identification	13
3.1.4.2. Sequence Diagram	14
3.1.4.3. Class Diagram	14
3.1.5. Use Case: Create an Archive	14
3.1.5.1. Class Identification	14
3.1.5.2. Sequence Diagram	15
3.1.5.3. Class Diagram	15
3.1.6. Use Case: Modify an Archive	15
3.1.6.1. Class Identification	15
3.1.6.2. Sequence Diagram	16
3.1.6.3. Class Diagram	16
3.1.7. Use Case: Remove an Archive	16

3.1.7.1. Class Identification	16
3.1.7.2. Sequence Diagram	17
3.1.7.3. Class Diagram	17
3.1.8. Use Case: Authorize a Registered Account	17
3.1.8.1. Class Identification	17
3.1.8.2. Sequence Diagram	18
3.1.8.3. Class Diagram	18
3.1.9. Use Case: Manage Users	18
3.1.9.1. Class Identification	18
3.1.9.2. Sequence Diagram	8
3.1.9.3. Class Diagram	8
3.2. Detailed Class Design	8
3.2.1. Class eLab User	8
3.2.2. Class Normal User	8
3.2.3. Class Authorized User	8
3.2.4. Class Admin	8
3.3. Overall Class Diagram	9
3.4. Algorithm/Query	9
3.5. Statechart Diagram	9
3.6. Interface Design	9
3.7. Class Persistence Representation Design	10
4. Traceability Matrix	10

After the Table of Contents There may be a list of tables and a list of pictures ar

1. Introduction

1.1 Purpose of Document

The purpose of this Software Design Description (SDD) document is to present a detailed description of the application-based archive for Telkom University laboratories known as eLab. This document will help in describing the software specifications with object oriented design.

This SDD document is targeted for the use of all individuals that are involved in the development of the eLab software. This document will be utilized as guidance and reference for the software development as well as an evaluation device in the final process of development as well as during the development process.

1.2 Scope of the Problem

eLab is an application-based method of archiving documentations of Telkom University laboratories. Its main purpose is to allow users, in this case the students at Telkom University to find, recap, or upload documentations of previous and upcoming projects or achievement of a lab in Telkom University with ease in the form of Instagram-like posts that can be filtered.

1.3 Definitions and Terms

1. User

An object that uses the application, initially they are guest user in which they can register to become a normal user and can be authorized by the admin to become an authorized user.

2. Admin

An object that acts as the super user of the application that focuses on user management

3. Archive

An object that contains description, tags and pictures that explains about the documentation of an event or activity that has happened in the past

4. SDD

Stands for Software Design Description which is a document that describes and explain the details about the planning of the software that is being developed

5. SRS

Stands for Software requirement Specification which is a document that specifies the specifications requirements of the software that is being developed

6. DBMS

Stands for Database Management System which is an organizing and processing system for a database in a computer system

7. Unity Engine

A cross-platform engine developed by Unity Technologies. The engine has gradually extended to support a variety of desktop, mobile, console and virtual reality applications.

8. C#

A general-purpose, multi-paradigm, programming language. It encompasses static typing, strong typing, lexically scoped, imperative, declarative, functional, generic, object-oriented, and component-oriented programming disciplines

1.4 References

- Khaidir M., Aulia R. A., P. Budi S. “*eLab SRS*”.

1.5 Systematic Discussion

This section will describe the general systemic discussion of this SDD Document. Chapter 1 will introduce our app, the eLab. Chapter 2 will then proceed to outline the description of the eLab at large. We will give a detailed elaboration of the design of eLab in chapter 3

2 Description of Global

2.1 Design Implementation Environment Design

eLab	Specification
Operating System	Windows 10
DBMS	MySQL
Development tools	Unity Engine
Filing System	Date, name, type (video, photo, document), ...
Bahasa Pemrograman	C#, PHP

2.2 . Architectural Description

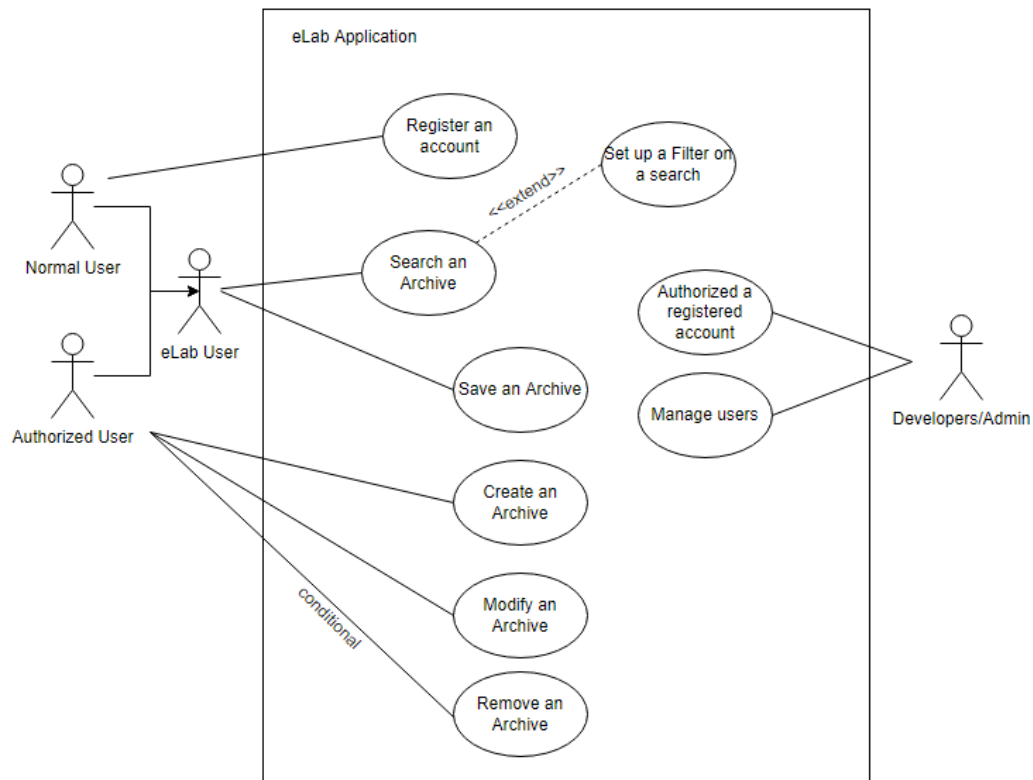
Give a brief description of the /L architecture to be built. Draw it in the form of a component diagram.

2.3 Component Description

No	Component Name	Detailed
1	Admin	Super user focused on user management
2	Normal user	Basic user, can be guest or registered user
3	Authorized user	Authorized by admin. Usually lab assistant
4	Login	Login to use more features
5	Register	Register to open more features
6	Search Archive	Search function for uploaded archives
7	View Archive	View function for searched archives
8	Create Archive	Create function specified for authorized user to create new archives
9	Modify Archive	Modify function specified for authorized user to modify archives
10	Remove Archive	Remove function specified* for authorized users to remove archives. *Can be registered user conditionally
11	Save Archive	Save function for viewed archive available to registered users

3 Design

3.1 Realization Use Case

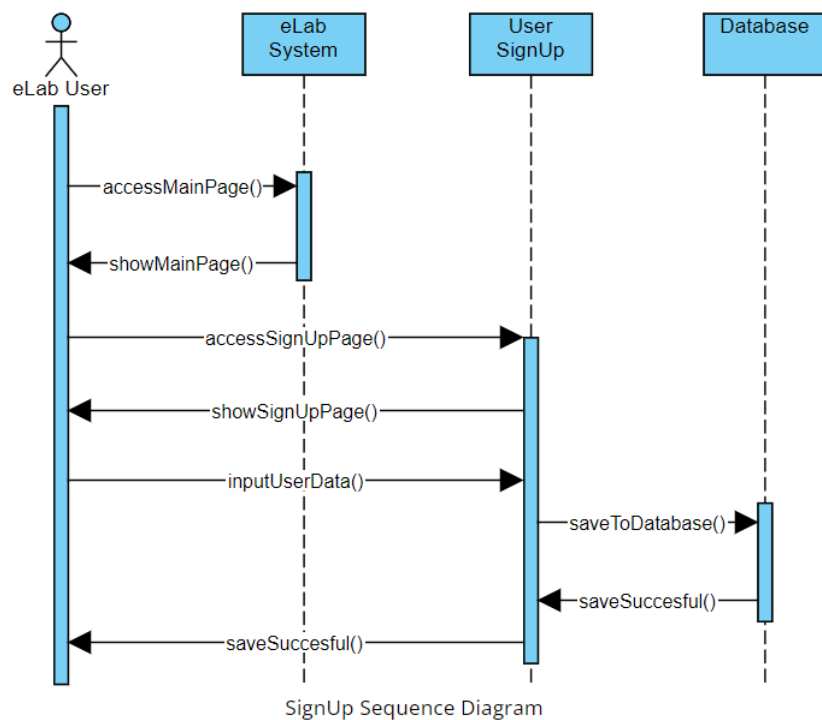


3.1.1 Use Case Register an Account

3.1.1.1 Class

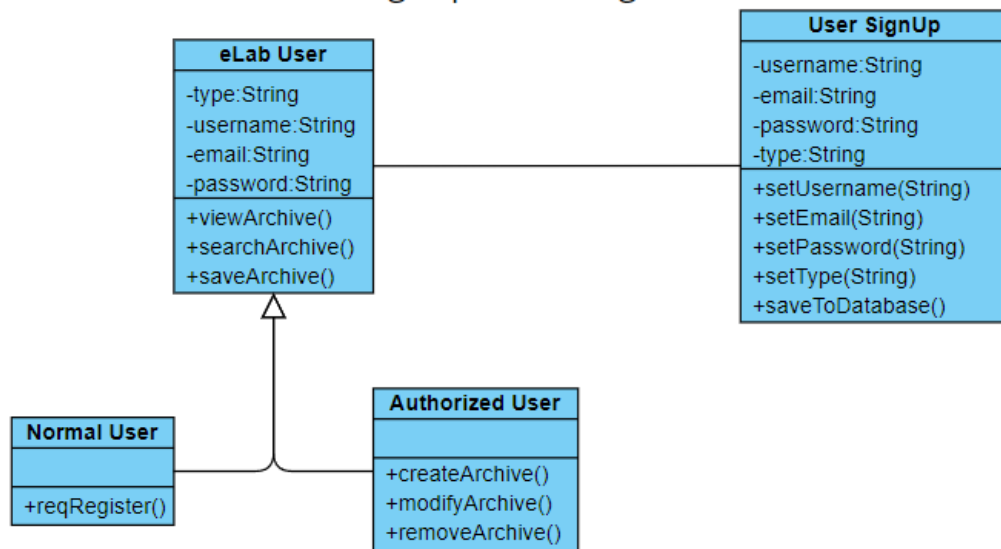
No	Class Name Design Class	Type
1	User	User
2	Admin	Admin
3	Database	Database

3.1.1.2 Sequence Diagram



3.1.1.3 Class Diagram

SignUp Class Diagram

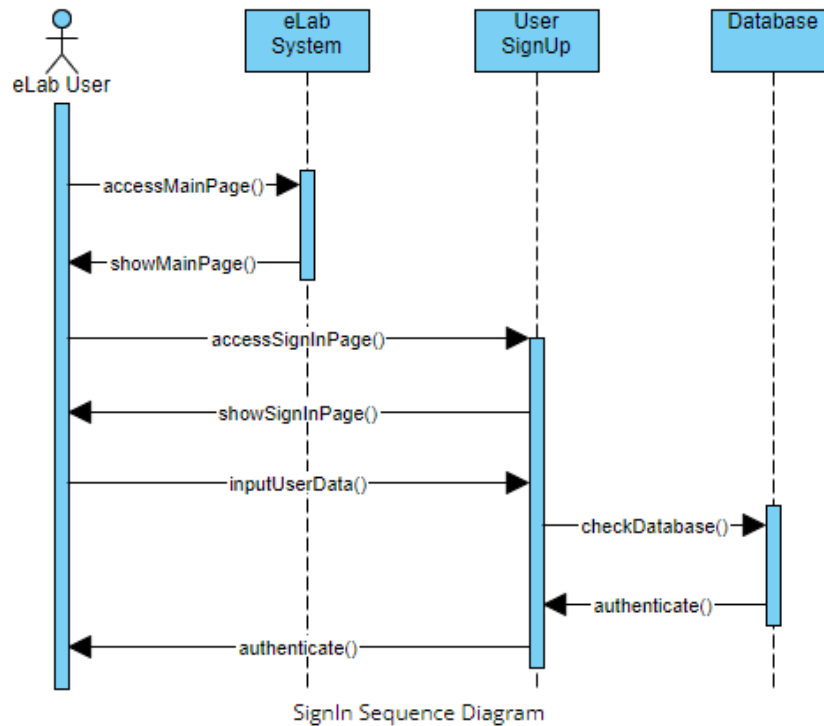


3.1.2 Use Case Sign in to an Account

3.1.2.1 Class

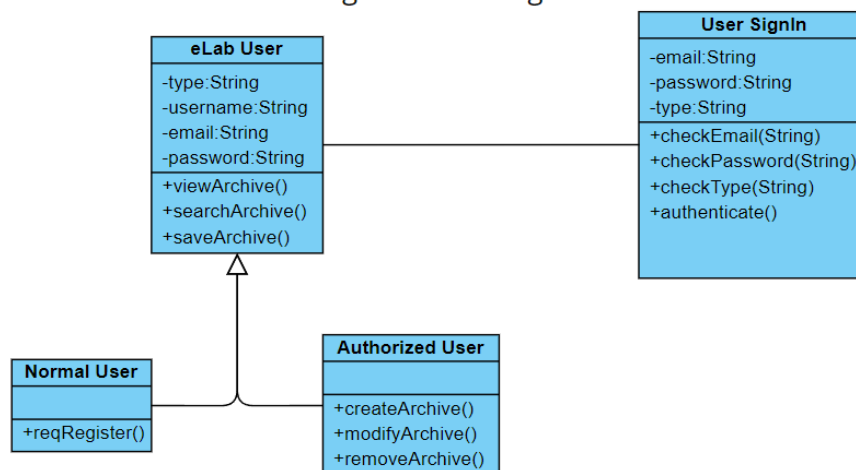
No	Class Name Design Class	Type
1	User	User
2	Admin	Admin
3	Database	Database

3.1.2.2 Sequence Diagram



3.1.2.3 Class Diagram

SignIn Class Diagram

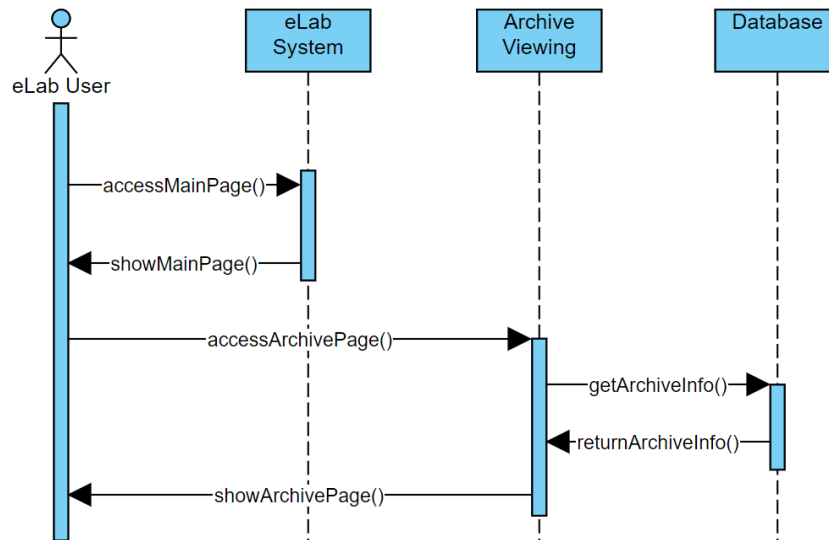


3.1.3 Use Case View an Archive

3.1.3.1 Class

No	Class Name Design Class	Type
1	User	User
2	Admin	Admin
3	Database	Database

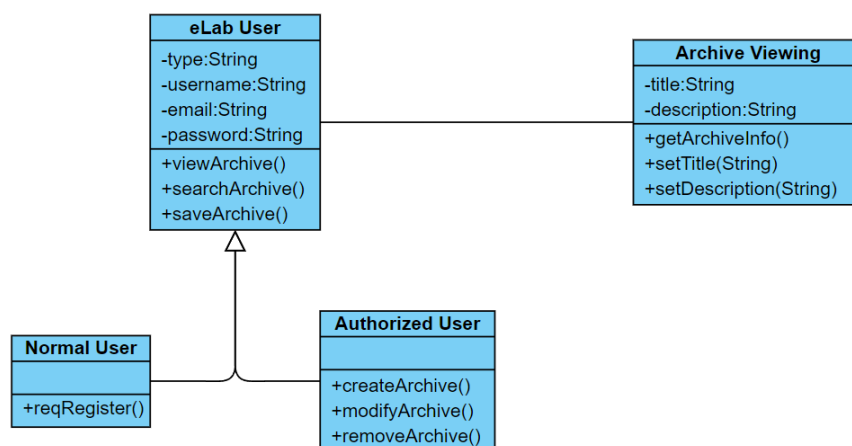
3.1.3.2 Sequence Diagram



Archive Viewing Sequence Diagram

3.1.3.3 Class Diagram

Archive Viewing Class Diagram

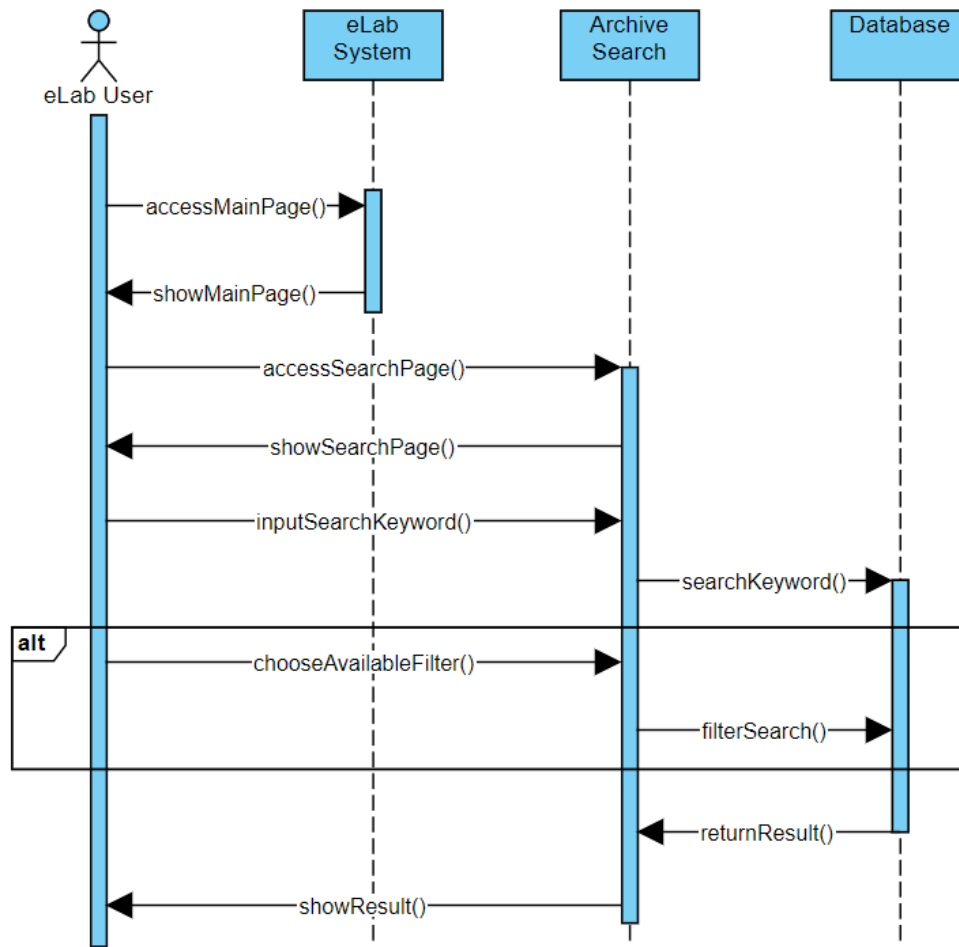


3.1.4 Use Case Search an Archive

3.1.4.1 Class

No	Class Name Design Class	Type
1	User	User
2	Admin	Admin
3	Database	Database

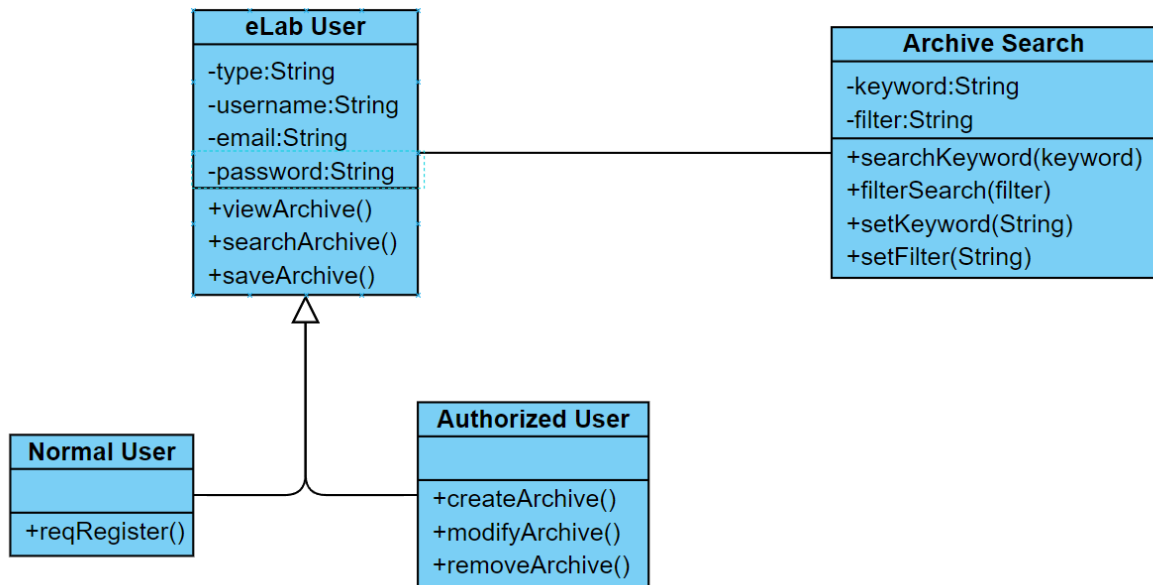
3.1.4.2 Sequence Diagram



Archive Search Sequence Diagram

3.1.4.3 Class Diagram

Archive Search Class Diagram

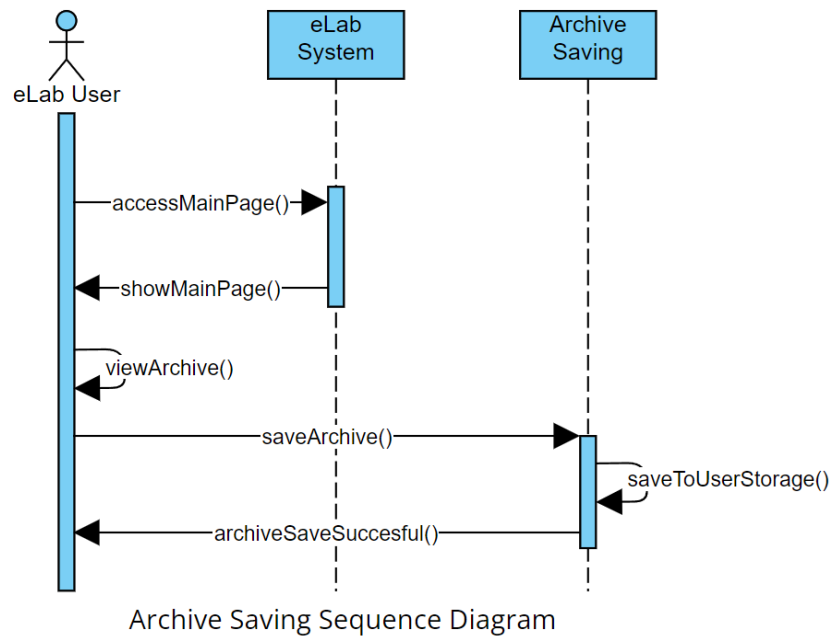


3.1.5 Use Case Save an Archive

3.1.5.1 Class

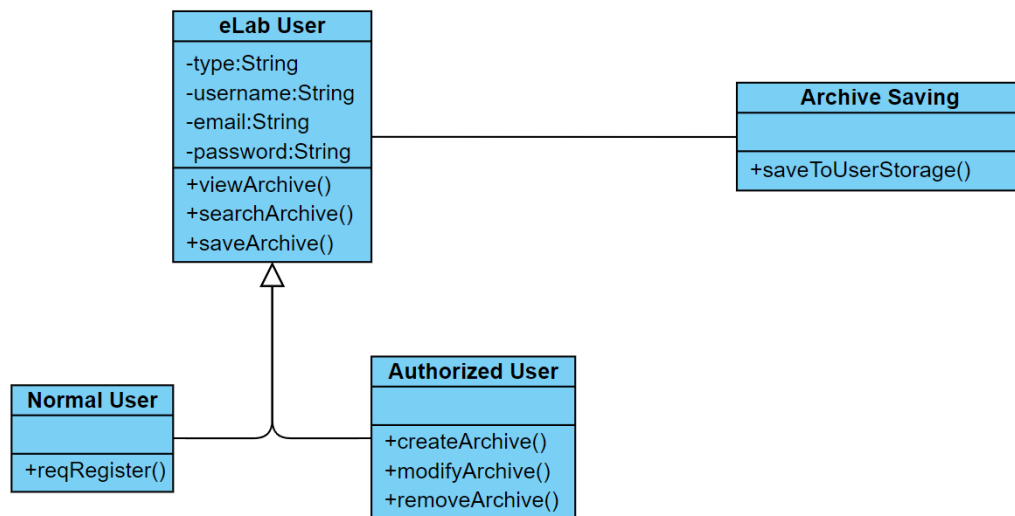
No	Class Name Design Class	Type
1	User	User
2	Admin	Admin
3	Database	Database

3.1.5.2 Sequence Diagram



3.1.5.3 Class Diagram

Archive Saving Class Diagram

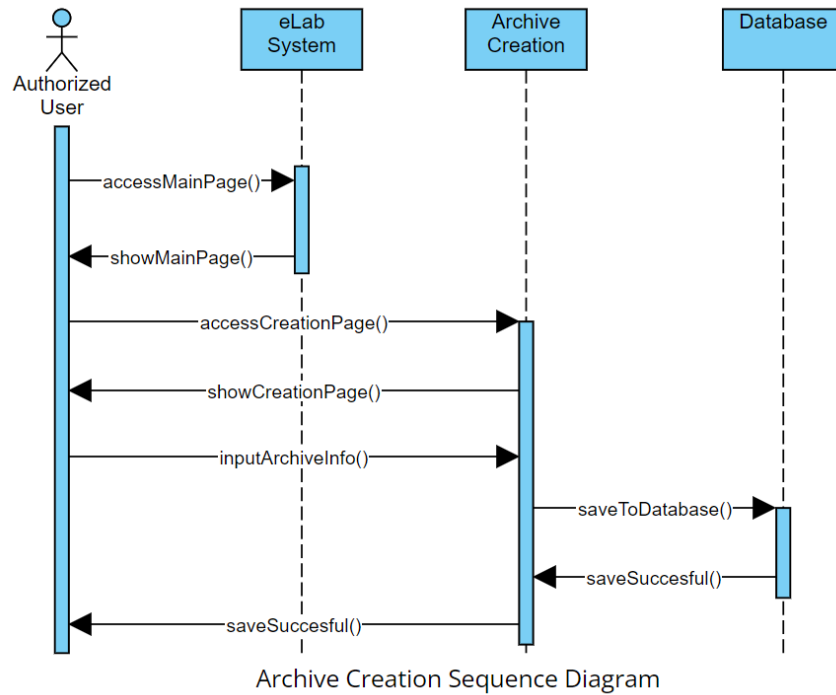


3.1.6 Use Case Create an Archive

3.1.6.1 Class

No	Class Name Design Class	Type
1	User	User
2	Admin	Admin

3.1.6.2 Sequence Diagram



3.1.6.3 Class Diagram

Archive Creation Class Diagram

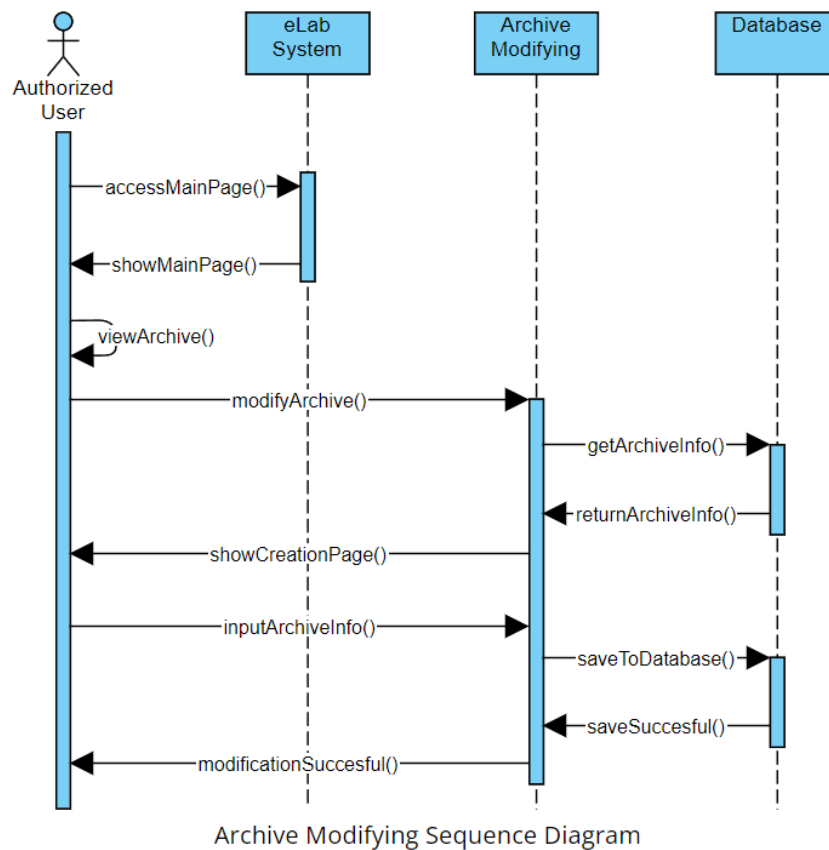


3.1.7 Use Case Modify an Archive

3.1.7.1 Class

No	Class Name Design Class	Type
1	User	User
2	Admin	Admin
3	Database	Database

3.1.7.2 Sequence Diagram



3.1.7.3 Class Diagram

Archive Modifying Class Diagram

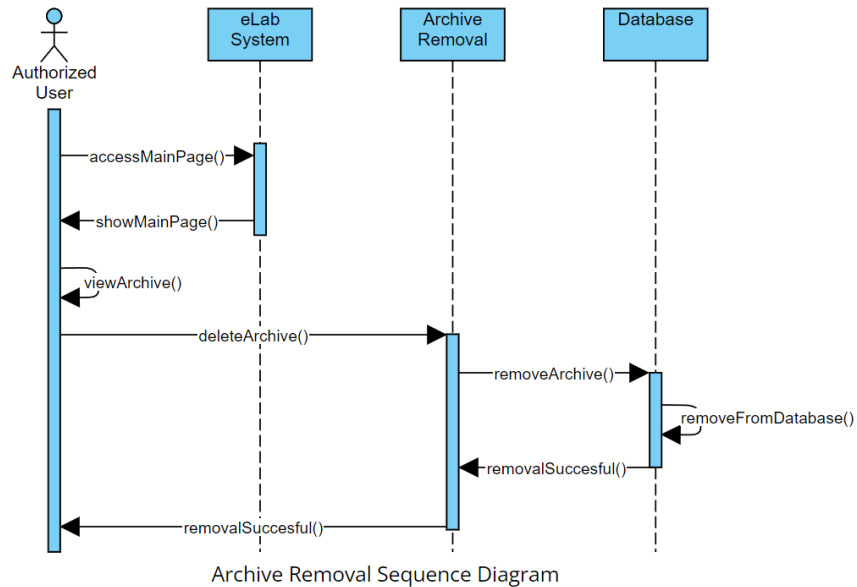


3.1.8 Use Case Remove an Archive

3.1.8.1 Class

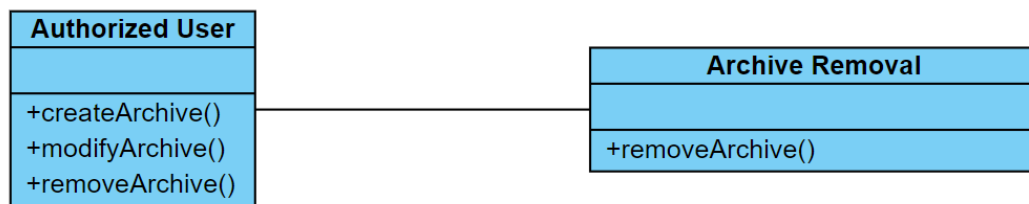
No	Class Name Design Class	Type
1	User	User
2	Admin	Admin
3	Database	Database

3.1.8.2 Sequence Diagram



3.1.8.3 Class Diagram

Archive Removal Class Diagram

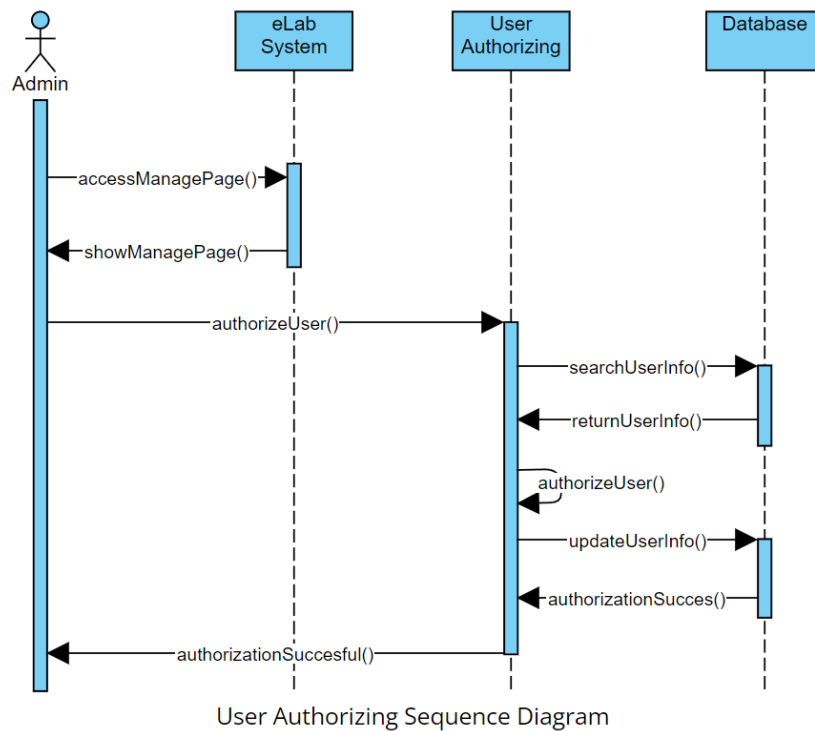


3.1.9 Use Case Authorize a Registered Account

3.1.9.1 Class

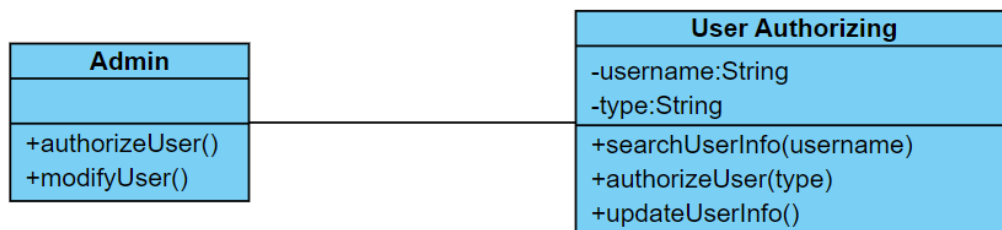
No	Class Name Design Class	Type
1	User	User
2	Admin	Admin
3	Database	Database

3.1.9.2 Sequence Diagram



3.1.9.3 Class Diagram

User Authorizing Class Diagram



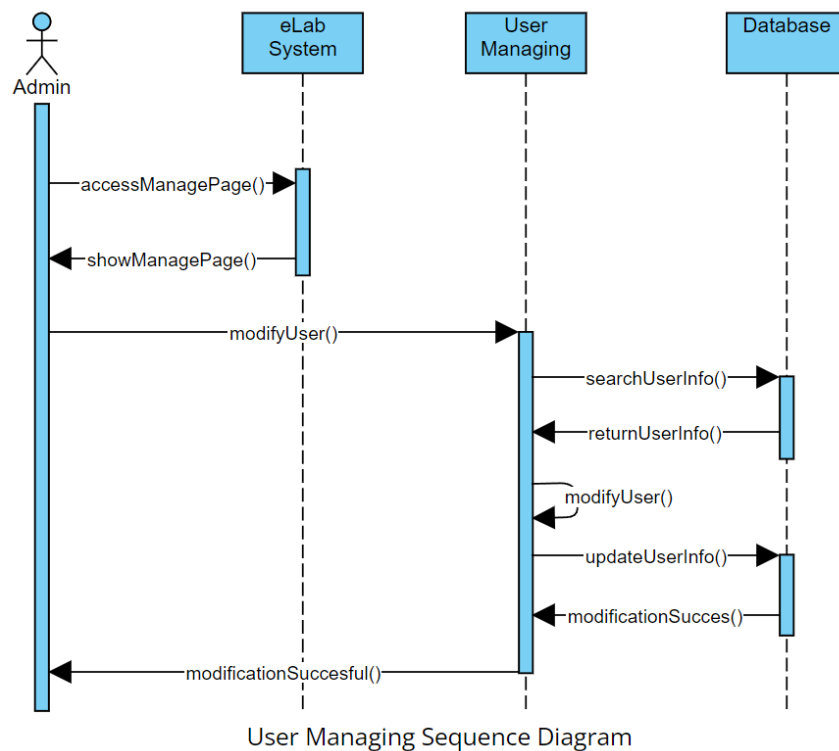
3.1.10 Use Case Manage Users

3.1.10.1 Class

No	Class Name Design Class	Type
1	User	User
2	Admin	Admin
3	Database	Database

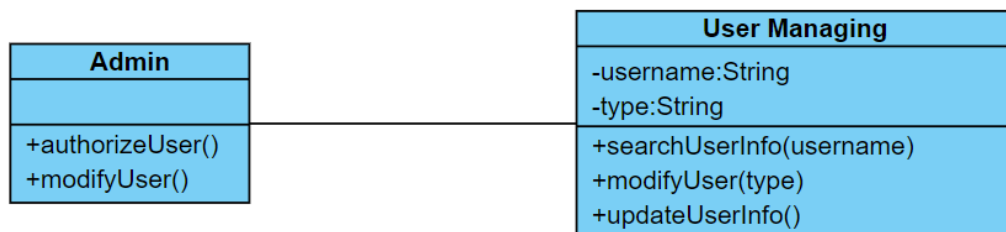
*Class types such as Boundary(Interface), Entity(Database), Controller

3.1.10.2 Sequence Diagram



3.1.10.3 Class Diagram

User Managing Class Diagram



3.2 Design Detailed Classes

No	Design Class	Name Related Analysis
1	eLab User	eLabUser
2	Normal User	Normal User
3	Authorized User	Authorized User
4	Admin	Admin

3.2.1 Class eLab User

Name of Class : eLab User

Operation Name	Visibility (private, public)	Description
<i>viewArchive()</i>	<i>public</i>	<i>View an Archive</i>
<i>searchArchive()</i>	<i>public</i>	<i>Search an Archive</i>
<i>saveArchive()</i>	<i>public</i>	<i>Save an Archive</i>
Attribute Name	Visibility (private, public)	Type
<i>type</i>	<i>private</i>	<i>String</i>
<i>username</i>	<i>private</i>	<i>String</i>
<i>email</i>	<i>private</i>	<i>String</i>
<i>password</i>	<i>private</i>	<i>String</i>

3.2.2 Class Normal User

Name of Class : Normal User

Operation Name	Visibility (private, public)	Description
<i>reqRegister()</i>	<i>public</i>	<i>Request to authorize account</i>
Attribute Name	Visibility (private, public)	Type

3.2.3 Class Authorized User

Name of Class : Authorized User

Operation Name	Visibility (private, public)	Description
<i>createArchive()</i>	<i>public</i>	<i>Create an Archive</i>
<i>modifyArchive()</i>	<i>public</i>	<i>Modify an Archive</i>
<i>removeArchive()</i>	<i>public</i>	<i>Remove an Archive</i>
Attribute Name	Visibility (private, public)	Type

3.2.4 Class Admin

Name of Class : Admin

Operation Name	Visibility (private, public)	Description
<i>authorizeUser()</i>	<i>public</i>	<i>Authorize a normal User</i>
<i>modifyUser()</i>	<i>public</i>	<i>Modify the privilege of a User</i>
Attribute Name	Visibility (private, public)	Type

3.3 Overall Class Diagram

3.4 Algorithms/Query

3.4.1 Login Algorithm

Class : Normal User

Operation Name : Login

Algorithm : (Algo-001)

```
<?php
    include( dirname(__FILE__) . '/database.php');
    if (isset($_POST["email"]) && !empty($_POST["email"]) &&
        isset($_POST["password"]) && !empty($_POST["password"])){

        $email = $_POST["email"];
        $password = $_POST["password"];

        $sql = "SELECT id, name, email, password, isAuthorized FROM user WHERE email='$email'";
        $result = $conn->query($sql);
        if($result !== FALSE){
            if ($result->num_rows > 0) {
                $row = $result->fetch_assoc();
                if($row["password"] != $password) echo "Password is Incorrect";
                else{
                    echo "GRANTED:";
                    echo "|";
                    echo $row["name"];
                    echo "|";
                    echo $row["email"];
                    echo "|";
                    echo $row["isAuthorized"];
                }
            }
            else {
                echo "User Not Found";
            }
        }
        else echo "MySQL ERROR";
    }
?>
```

{If referring to a specific query, complete the query table below}

No Query	Query	Description
Q-001	SELECT id, name, email, password, isAuthorized FROM user WHERE email='\$email'	Check the account information where the email is equal to 'email'

3.4.2 Register Algorithm

Class : eLab User

Operation Name : Register

Algorithm : (Algo-002)

```
<?php
    include( dirname(__FILE__) . '/database.php');
```

```

        if (isset($_POST["fullname"]) && !empty($_POST["fullname"]) &&
            isset($_POST["email"]) && !empty($_POST["email"]) &&
            isset($_POST["password"]) && !empty($_POST["password"]) &&
            isset($_POST["cpassword"]) && !empty($_POST["cpassword"])){

            $fullname = $_POST["fullname"];
            $email = $_POST["email"];
            $password = $_POST["password"];
            $cpassword = $_POST["cpassword"];

            $sql = "SELECT email FROM user WHERE email='$email'";
            $result = $conn->query($sql);
            if($result !== TRUE){
                $sql = "INSERT INTO `user`(`email`, `name`, `password`, `isAuthorized`, `saved_archive`) VALUES
($email', '$fullname', '$password', 0, ')";
                $result = $conn->query($sql);
                if($result !== FALSE){
                    echo "GRANTED.";
                    echo "|REGISTRATION COMPLETE|";
                    echo $fullname;
                }else{
                    echo "MySQL ERROR";
                }
            }else {
                echo "Account already existed!";
            }
        }else echo "Form ERROR";

?>

```

{If referring to a specific query, complete the query table below}

Query

No Query	Query	Description
Q-002	INSERT INTO `user`(`email`, `name`, `password`, `isAuthorized`, `saved_archive`) VALUES (\$email', '\$fullname', '\$password', 0, ')	Register the account into the database

3.4.3 Create Archive Algorithm

Class : Authorized User

Operation Name : Create Archive

Algorithm : (Algo-003)

```

<?php
    include( dirname(__FILE__) . '/database.php');
    if(isset($_POST["title"]) && !empty($_POST["title"]) &&
        isset($_POST["desc"]) && !empty($_POST["desc"]) &&
        isset($_POST["type"]) && !empty($_POST["type"]) &&
        isset($_POST["date"]) && !empty($_POST["date"])){

        $title = $_POST["title"];
        $desc = $_POST["desc"];
        $type = $_POST["type"];
        $date = $_POST["date"];
        $author = $_POST["author"];
    }

```

```

    $sql = "INSERT INTO `archive`(`title`, `desc`, `type`, `date`, `authorName`) VALUES
('title','$desc','$type','$date', '$author')";
    $result = $conn->query($sql);
    if($result !== FALSE){
        echo "SUCCESS";
    }else echo $conn->error;
    }else echo "Form ERROR";
?>

```

{If referring to a specific query, complete the query table below}

Query

No Query	Query	Description
Q-003	INSERT INTO `archive`(`title`, `desc`, `type`, `date`, `authorName`) VALUES ('title','\$desc','\$type','\$date', '\$author')	Create a new archive in the database

3.4.4 Read Archive Algorithm

Class : eLab User

Operation Name : Read Archive

Algorithm : (Algo-004)

```

<?php
    include( dirname(__FILE__) . '/database.php');
    if(isset($_POST["date"]) && isset($_POST["filter"])){
        $date = $_POST["date"];
        $filter = $_POST["filter"];

        $sql = "";

        if(empty($_POST["date"]) && empty($_POST["filter"])){
            $sql = "SELECT * FROM archive";
        }else if(!empty($_POST["date"]) && !empty($_POST["filter"])){
            $sql = "SELECT * FROM archive WHERE date='$date' AND type='$filter'";
        }else if(empty($_POST["date"]) && !empty($_POST["filter"])){
            $sql = "SELECT * FROM archive WHERE type='$filter'";
        }else if(!empty($_POST["date"]) && empty($_POST["filter"])){
            $sql = "SELECT * FROM archive WHERE date='$date'";
        }

        $result = $conn->query($sql);
        if($result !== FALSE){
            if ($result->num_rows > 0) {
                $counter = 0;
                while($row = $result->fetch_assoc()){
                    echo $row["title"];
                    echo "|";
                    echo $row["desc"];
                    echo "|";
                    echo $row["type"];
                    echo "|";
                    echo $row["authorName"];
                    echo "|";
                    if(empty($row["image"])) echo "null";
                    else echo base64_encode($row['image']);
                }
            }
        }
    }

```



```

        if (++$counter < $result->num_rows) echo "#";
    }
    }else echo "No Archives Found!";
    }else echo "MySQL ERROR";
    }else echo "Form ERROR";
?>

```

{If referring to a specific query, complete the query table below}

Query :

No Query	Query	Description
Q-004	SELECT * FROM archive	Obtain all of the archive from the database
Q-005	"SELECT * FROM archive WHERE date='\$date' AND type='\$filter'"	Obtain archive with a certain date and filter
Q-006	SELECT * FROM archive WHERE type='\$filter'	Obtain archive with a certain filter
Q-007	SELECT * FROM archive WHERE date='\$date'	Obtain archive with a certain date

3.4.5 Update Archive Algorithm

Class : Authorized User

Operation Name : Update Archive

Algorithm : (Algo-005)

```

<?php
    include( dirname(__FILE__) . '/database.php');
    if(isset($_POST["id"]) && !empty($_POST["id"])){

        $id = $_POST['id'];
        $title = $_POST['title'];
        $desc = $_POST['desc'];
        $type = $_POST['type'];
        $date = $_POST['date'];

        $sql = "UPDATE archive SET `title`='$title', `desc`='$desc', `type`='$type', `date`='$date' WHERE `archive_id`='$id'";
        $result = $conn->query($sql);
        if($result !== FALSE){
            echo "SUCCESS";
        }else echo $conn->error;
        }else echo "Form ERROR";
?>

```

{If referring to a specific query, complete the query table below}

Query :

No Query	Query	Description
Q-008	UPDATE archive SET `title`='\$title', `desc`='\$desc', `type`='\$type', `date`='\$date' WHERE `archive_id`='\$id'	Update the values of a certain archive in the database

3.4.6 Remove an Archive Algorithm

Class : Authorized User

Operation Name : Remove an Archive

Algorithm : (Algo-006)

```
<?php
    include( dirname(__FILE__) . '/database.php');
    if(isset($_POST["id"]) && !empty($_POST["id"])){

        $id = $_POST["id"];

        $sql = "DELETE FROM archive WHERE archive_id='$id'";
        $result = $conn->query($sql);
        if($result !== FALSE){
            echo "SUCCESS";
        }else echo $conn->error;
        }else echo "Form ERROR";
    ?>
```

{If referring to a specific query, complete the query table below}

Query

No Query	Query	Description
Q-009	DELETE FROM archive WHERE archive_id='\$id	Remove an archive with a certain id from the database

3.5 Interface Design

This section is filled with the initial version of the interface prototype .

Next, for each interface/screen, write down the detailed specifications, for example as below:

Interface : Log In Page

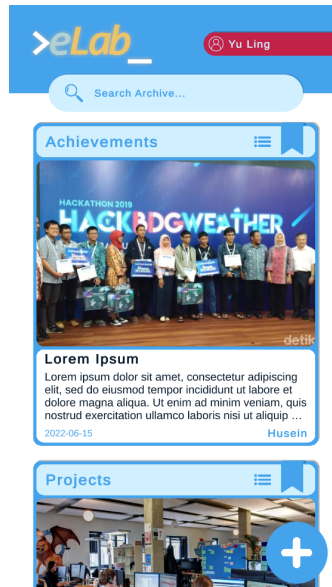
Id Objek	Type	Name	Description
BTN1	Button	Login	If clicked, will activate the Log In Process.
BTN2	Button	Sign Up	If clicked, move to Sign Up Page
TX1	Text Input	Email	Box to input a user email
TX2	Text Input	Password	Box to input a corresponding password to a user

Interface : Sign Up Page

Id Objek	Type	Name	Description
----------	------	------	-------------

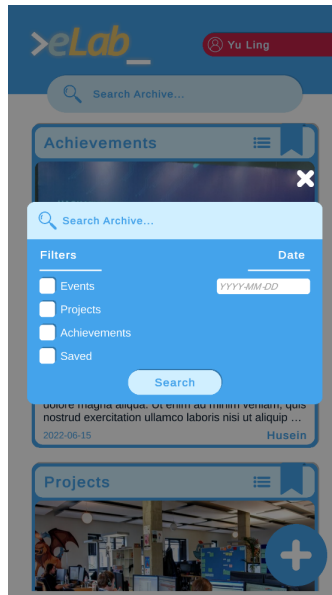
BTN1	Button	Register	If clicked, will activate the SignUp Process
BTN2	Button	Sign In	If clicked, move to Log In Page
TXI1	Text Input	FullName	Box to input a user's name
TXI2	Text Input	Email	Box to input a user's email
TXI3	Text Input	Password	Box to input a password for the user account
TXI4	Text Input	Confirm Password	Box to input a password for the user account (must be the same with the previously inputted password)

Interface : Main Page



Id Objek	Type	Name	Description
BTN1	Button	Create	If clicked, will move to the create archive page (only available for Authorized User)
BTN2	Button	Account	If clicked, will go to the profile page
TX1	Text Input	Search Bar	Box to input a keyword term for a search
ArchivePage	Object	Archive Page	If clicked, will open the detailed page for the corresponding archive

Interface : Filter Overlay



Id Objek	Type	Name	Description
<i>BTN1</i>	<i>Button</i>	<i>Apply Filter</i>	<i>If clicked, will activate the Filter Process</i>
<i>CBX1</i>	<i>CheckBox</i>	<i>Archive Type</i>	<i>If check, will mark for filter process</i>
<i>CBX2</i>	<i>CheckBox</i>	<i>Laboratory</i>	<i>If check, will mark for filter process</i>
<i>CBX3</i>	<i>CheckBox</i>	<i>Tags</i>	<i>If check, will mark for filter process</i>
<i>NMI1</i>	<i>Num Input</i>	<i>Start Date</i>	<i>A box to input number for the initial date range</i>
<i>NM2</i>	<i>Num Input</i>	<i>End Date</i>	<i>A box to input number for the final date range</i>

3.6 Design of Class Persistence Representation

*This section is filled with database schema design and its traceability to the entity class.
(RELATIONSHIP SCHEME DEVELOPMENT)*

4 Traceability Matrix

Mapping use cases with related classes

Requirements	Related Use Cases	Class
FR-01	Register account	eLab user
FR-02	Log in account	eLab user
FR-03	Create an archive	Authorized user
FR-04	View an archive	eLab user
FR-05	Modify an archive	Authorized user
FR-06	Remove an archive	Authorized user
FR-07	Save an archive	eLab user
FR-08	Search an archive	eLab user
FR-09	Manage users	Admin
FR-10	Authorized users	Admin