**CEBU INSTITUTE OF TECHNOLOGY**

**UNIVERSITY**

COLLEGE OF COMPUTER STUDIES

Software Design Description

for

LOOP - Learning Object Organizer Plus

Signature

Aton, Marc Venry - Team Lead, Back-end Developer \_\_\_\_\_\_\_\_\_\_\_\_\_

Langga, Edwin - Systems Analyst, Back-end Developer \_\_\_\_\_\_\_\_\_\_\_\_\_

Roxas, Jessie - Front-end Developer, Quality Assurance Engineer \_\_\_\_\_\_\_\_\_\_\_\_\_

Quilaton, Krizette Gen n - Front-end Developer, Documentation Officer \_\_\_\_\_\_\_\_\_\_\_\_\_

Change History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reasons for Change** | **Version** |
| LOOP  [Software Project] | 06/23/14 | This is the continuation of the Learning Object Organizer Plus (LOOP) which was our Software Engineering project. | R1.0 |
|  |  | The documentation must be updated and finalized before it is to be submitted. | R2.0 |

Preface

This is the Software Design Description (SDD) document for the LOOP project or Learning Object Organizer Plus project. LOOP is part of the new Learning Objects Project headed by the CIT-University eLearning Center. It hosts Learning Object uploads and reviews. It is to be an organizing and filtering type of system designed to provide another system (Informatron CMS) with only Ready to Use downloadable Learning Objects.

The new Learning Objects Project is a reboot of the previous Learning Object system in which users view Learning Objects straight from the server. The project aims to create a cloud computing architecture out of the said system by providing the Learning Objects for download and make them playable offline through specialized Learning Object Players. Development on these players is being made on the following platforms – Windows, iOS, and Android.

The intended audiences for this document are the faculty, developers, project managers and anyone else concerned in the technical side of development of this project. Being the most technical of all documentations, full understanding of this document content requires a certain amount of knowledge and experience in the IT industry.

Table of Contents

Signature 2

Change History 3

Preface 4

Table of Contents 5

List of Figures 7

List of Tables 8

1. Introduction 9

1.1. Purpose 9

1.2. Scope 9

1.3. Definitions and Acronyms 9

2. References 11

3. Decomposition Description 12

3.1. Module Decomposition 12

3.1.1. Administrator 12

3.1.2. Developer 12

3.1.3. Reviewer 12

3.2. Concurrent Process Decomposition 13

3.2.1. Database Process 13

3.2.2. Controller Process 13

3.3. Data Decomposition 13

3.3.1. Site User 13

3.3.2. Learning Object 13

4. Dependency Description 14

4.1. Inter-module Dependencies 14

4.2. Inter-process Dependencies 14

4.3. Data Dependencies 14

5. Interface Description 15

5.1. Module Interface 15

5.1.1. Homepage/Login Module 15

5.1.2. Admin Panel Module 15

5.1.3 Developer Panel Module 16

5.1.4 Reviewer Panel Module 16

5.2. Process Interface 17

5.2.1. Home Page 17

5.2.2. Sign Up Pop-up 18

5.2.3. Login Pop-up 18

5.2.4. Main View Page for Developer 19

5.2.5. File Action Pop-up for Developer 19

5.2.6. Main View Page for Reviewer 20

5.2.7. Download Page for Reviewer 20

5.2.8. Review Later List for Reviewer (After Download) 21

5.2.9. Review Option for Reviewer (After Download) 21

5.2.10. Review Page (Only for Reviewer) 22

5.2.11. Advanced Search Page 23

5.2.12. Upload Page (Only for Developer) 24

5.2.13. Change Email Pop-up 24

5.2.14. Change Password Pop-up 25

5.2.15. New Account Requests Tab for Admin 25

5.2.16. User Action for New Account Requests Tab (Admin) 26

5.2.17. Main View Page for Admin (Developer Tab) 26

5.2.18. Main View Page for Admin (Reviewer Tab) 27

5.2.19. Main View Page for Admin (Inactive Accounts Tab) 27

5.2.20. User Action for Admin Pages (Developer, Reviewer, Inactive Accounts Tab) 28

5.2.21. Main View Page for Admin (Blocked Accounts Tab) 28

5.2.22. User Action for Admin Page (Blocked Accounts Tab) 29

5.2.23. Main View Page for Admin (Learning Objects Tab) 29

5.2.24. File Action for Admin Page (Learning Objects Tab) 30

5.2.25. Download Page for Admin (After you click Review) 30

5.2.26. Review Page for Admin (After Download) 31

5.2.27. Change Password for Admin 31

5.2.28 History View for LO's on all user…………………………………………………………………………………………………....32

6. Detailed Design 33

6.1. Module Detailed Design 33

6.1.1. Administrator User Module Detail 33

6.1.2. Developer User Module Detail 34

6.1.3. Reviewer User Module Detail 35

6.1.4. Other Operations 36

6.2. Data Detailed Design 36

6.2.1. Admin Entity Detail 36

6.2.2. Developer Entity Detail 37

6.2.3. Reviewer Entity Detail 37

6.2.4. Learning Object Entity Detail 37

7. Annexes 38

7.1. Class diagram 38

7.2. Use case realization (Sequence diagram / Communication diagram) 39

7.3. Entity-relationship diagram 41

List of Figures

Figure 5.2.1 - The Home/Index 17

Figure 5.2.2 – Sign Up Pop-up for all users including Admin 18

Figure 5.2.3 - Login Pop-up for all users including Admin 18

Figure 5.2.4 – Main View Page for Developer 19

Figure 5.2.5 - File Action Pop-up for Developer 19

Figure 5.2.6 – Main View Page for Reviewer 20

Figure 5.2.7 - Download Page for Reviewer 20

Figure 5.2.8 – Review Later List for Reviewer 21

Figure 5.2.9 – Review Option for Reviewer 21

Figure 5.2.10 – Review Page 22

Figure 5.2.11 – Advanced Search Page (Both Developer & Reviewer) 23

Figure 5.2.12 – Upload Page 24

Figure 5.2.13 – Change Email Pop-up 24

Figure 5.2.14 – Change Password Pop-up 25

Figure 5.2.15 – New Account Requests Tab for Admin 25

Figure 5.2.16 - User Action for New Account Requests Tab (Admin) 26

Figure 5.2.17 – Main View Page for Admin (Developer Tab) 26

Figure 5.2.18 - Main View Page for Admin (Reviewer Tab) 27

Figure 5.2.19 – Main View Page for Admin (Inactive Accounts Tab) 27

Figure 5.2.20 – User Action for Admin Pages (Developer, Reviewer, Inactive Accounts Tab) 28

Figure 5.2.21 – Main View Page for Admin (Blocked Accounts Tab) 28

Figure 5.2.22 – User Action for Admin Page (Blocked Accounts Tab) 29

Figure 5.2.23 – Main View Page for Admin (Learning Objects Tab) 29

Figure 5.2.24 – Action for Admin Page (Learning Objects Tab) 30

Figure 5.2.25 – Download Page for Admin (After you click Review) 30

Figure 5.2.26 – Review Page for Admin (After Download) 31

Figure 5.2.27 – Change Password for Admin 31

Figure 5.2.28 – History View of LO’s on all users……………………………………………………………………………………...32

Figure 5.2.29 – History View for Developers…………………………………………………………………………………………...32

Figure 5.2.30 – History View for Reviewer……………...……………………………………………………………………………...32

Figure 5.2.31 – History View for Admin……………………...………………………………………………………………………...33

Figure 5.2.32 – Assigning Reviewer …………………………….……………………………………………………………………...33

Figure 5.2.33 – View Learning Element for Admin …………………………..………………………………………………………...34

Figure 5.2.34 – View Learning Element for Developer……...………………………………………………………………………….34

Figure 5.2.35 – View Learning Element for Reviewer ………….……………………………………………………………………...34

Figure 5.2.36 – Upload of Learning Object… ………………….……………………………………………………………………...35

Figure 5.2.36 – Upload of Learning Elements ………………….……………………………………………………………………...35

Figure 7.1.1– Class Diagram 38

Figure 7.2.1 – Sequence Diagram for Developer 39

Figure 7.2.2 – Sequence Diagram for Reviewer 39

Figure 7.2.3 – Sequence Diagram for Admin 40

Figure 7.3.1 – Entity Relationship Diagram 41

List of Tables

Table 3.1.1 – Administrator Module Description 12

Table 3.1.2 – Developer Module Description 12

Table 3.1.3 – Reviewer Module Description 12

Table 4.1.1 – Reviewer Module Description 14

Table 4.1.2 – Developer Module Description 14

Table 5.1.1 – Homepage/Login Module Description 15

Table 5.1.2 – Admin Panel Module Description 15

Table 5.1.3 – Developer Panel Module Description 16

Table 5.1.4 – Reviewer Panel Module Description 16

Table 6.2.1 – Admin Entity Detail 35

Table 6.2.2 – Developer Entity Detail 36

Table 6.2.3 – Reviewer Entity Detail 36

Table 6.2.4 – Learning Object Entity Detail 36

# Introduction

## Purpose

This Software Design Description will lay out the technical specifics of the Learning Object Organizer Plus project or LOOP project. The systems architecture and the means of its implementation will be defined and explained in the succeeding pages of this document. All of which will guide both the developers and the project managers in the technical side of the software development process.

The document will also complement the system’s Software Requirements Specification (SRS) document by breaking down each use case or functional specification into detailed processes by means of different diagrams shown in the annexes.

## Scope

This document will provide a clear view of the LOOP or Learning Object Organizer Plus system design plan. Technologies covered in the development will be web based since the system is to be deployed on a server. Client server concepts will be taken into account and the user interfaces will be constructed using PHP, HTML5, CSS3. We will also be using frameworks such as Bootstrap and CodeIgniter. The necessary database components will be implemented using mySQL.

## Definitions and Acronyms

**Learning Object** – an interactive presentation of a certain subject/topic designed to help in academic development.

**Organizer –** a thing used for organizing. [Equipment](http://www.macmillandictionary.com/search/british/direct/?q=equipment) [that](http://www.macmillandictionary.com/search/british/direct/?q=that) [helps](http://www.macmillandictionary.com/search/british/direct/?q=helps) [you](http://www.macmillandictionary.com/search/british/direct/?q=you) [to](http://www.macmillandictionary.com/search/british/direct/?q=to) [arrange](http://www.macmillandictionary.com/search/british/direct/?q=arrange) [information](http://www.macmillandictionary.com/search/british/direct/?q=information).

**LOOP –** Learning Object Organizer Plus

**Cloud Framework** – is a concept of online computing that aims to make system resources available to a large number of users by storing them on a common network (typically the internet) as shared resources. Users download only the necessary or requested resources and can use these on their respective devices anytime regardless of network connectivity.

**eLearning** – learning with the aid of computers.

**AJAX (Asynchronous JavaScript and XML) -** A group of interrelated web development methods used on the client-side to create interactive web applications.

**Apache HTTP Server -** Commonly referred to as **Apache** is [web server](http://en.wikipedia.org/wiki/Web_server) software notable for playing a key role in the initial growth of the [World Wide Web](http://en.wikipedia.org/wiki/World_Wide_Web)

**JavaScript -** A prototype-based scripting language that is dynamic, weakly typed and has first-class functions. It is a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles.

**jQuery -** A cross-browser JavaScript library designed to simplify the client-side scripting.

**MySQL -** A [relational database management system](http://en.wikipedia.org/wiki/Relational_database_management_system) (RDBMS) that runs as a server providing multi-user access to a number of databases.

**PHP: Hypertext Preprocessor -** A widely used, general-purpose [scripting language](http://en.wikipedia.org/wiki/Scripting_language) that was originally designed for [web development](http://en.wikipedia.org/wiki/Web_development) to produce [dynamic web pages](http://en.wikipedia.org/wiki/Dynamic_web_page).

**CSS (Cascading Style Sheets) -** A [style sheet language](http://en.wikipedia.org/wiki/Style_sheet_language) used to describe the [presentation semantics](http://en.wikipedia.org/wiki/Presentation_semantics) (the look and formatting) of a document written in a [markup language](http://en.wikipedia.org/wiki/Markup_language). We will be using CSS3 which is the latest version of CSS.

**Hypertext Transfer Protocol (HTTP) -** A [networking protocol](http://en.wikipedia.org/wiki/Communications_protocol) for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the [World Wide Web](http://en.wikipedia.org/wiki/World_Wide_Web).

**UML (Unified Modeling Language) –** is a standardized general-purpose [modeling language](http://en.wikipedia.org/wiki/Modeling_language) in the field of [software engineering](http://en.wikipedia.org/wiki/Software_engineering).

**XAMPP -** A [free and open source](http://en.wikipedia.org/wiki/Free_software) [cross-platform](http://en.wikipedia.org/wiki/Cross-platform) [web server](http://en.wikipedia.org/wiki/Web_server) [solution stack](http://en.wikipedia.org/wiki/Solution_stack) package, consisting mainly of the [Apache HTTP Server](http://en.wikipedia.org/wiki/Apache_HTTP_Server), [MySQL](http://en.wikipedia.org/wiki/MySQL) [database](http://en.wikipedia.org/wiki/Database), and [interpreters](http://en.wikipedia.org/wiki/Interpreter_(computing)) for scripts written in the [PHP](http://en.wikipedia.org/wiki/PHP) and [Perl](http://en.wikipedia.org/wiki/Perl) [programming languages](http://en.wikipedia.org/wiki/Programming_language).

**HTML -** stands for [**Hypertext**](http://en.wikipedia.org/wiki/HyperText)  **Markup Language** is the predominant [markup language](http://en.wikipedia.org/wiki/Markup_language) for [web pages](http://en.wikipedia.org/wiki/Web_page). HTML is the basic building-blocks of web pages. A markup language is a set of markup tags, and HTML uses markup tags to describe [web pages](http://en.wikipedia.org/wiki/Web_page). We will be using HTML5 which is the latest version of HTML.

**Bootstrap –** Bootstrap is a front-end framework. It adds responsive design elements to our UI.

**CodeIgniter –** CodeIgniter is a back-end framework. This will force us to use the MVC model.

# References

* IEEE Std 1016-1998
* CCS427 SDD Template
* Cloud Computing - http://en.wikipedia.org/wiki/Cloud\_computing
* The Learning Objects – CIT-U local server: 172.31.11.208

# Decomposition Description

This section will define the different modules and concurrent processes that constitute the system. Modules will be divided based on primary system functions and user characteristics and the processes will be based on the client-server database paradigm.

## Module Decomposition

### Administrator

The administrator subsystem will be provided specific pages used for localized user management purposes.

|  |  |
| --- | --- |
| **Identification** | Administrator |
| **Type** | Module |
| **Purpose** | To perform localized user management functions such as: reviewing and deleting learning objects, blocking developers and reviewers, and monitoring account activity. |

**Table 3.1.1 Administrator Module Description**

### Developer

The Developer subsystem will be consisted of the main features of the system. The Developer subsystem has a GUI that allows developers to add, edit and delete learning objects.

|  |  |
| --- | --- |
| **Identification** | Developer |
| **Type** | Module |
| **Purpose** | To allow the Developers to perform functions such as: adding, editing, deleting learning object materials. |

**Table 3.1.2 Developer Module Description**

### Reviewer

The Reviewer subsystem will be consisted of the main features of the system. The Reviewer subsystem has a GUI that allows reviewers to view (download) and rate the learning objects added by the developers.

|  |  |
| --- | --- |
| **Identification** | Reviewer |
| **Type** | Module |
| **Purpose** | To allow the Reviewer to perform functions such as: viewing, searching, and rating the learning objects. |

**Table 3.1.3 Reviewer Module Description**

## Concurrent Process Decomposition

### Database Process

Identification : Database Process

Type : PHP file

Purpose : To access the database and perform all the queries.

Function : Waits for query request in managing entities.

Lifetime : Instantiated on demand. Available for the lifetime of the system.

Subordinates : Apache Web Server and MySQL Server.

### Controller Process

Identification : Controller Process

Type : PHP files

Purpose : To control the behavior of the system based on user interaction.

Function : Select Operation/s to perform

Prepare and display interface on requested command.

Perform queries or inter-site requests according to given commands.

Lifetime : Instantiated on demand. Available for the lifetime of the system.

Subordinates: Apache Web Server and MySQL Server.

## Data Decomposition

### Site User

Once users will be approved by the system admin and added to the user management system, the user will be added to another table named “siteUser”. Relevant information on the users will be stored on the

### Learning Object

Learning Objects are educational multimedia content that can be downloaded and viewed on any of the Learning Object players, which are applications being developed in parallel to this one. Learning Objects will be retrieved from the uploading system, which is another separate system outside of this system’s scope. A LO file will be wrapped in an entity class written in PHP code to enable organizing, searching and other relevant functions.

# Dependency Description

## Inter-module Dependencies

The admin module may indirectly affect activity in the user module through the user management actions but the former does not have any direct technical connection to the latter. A good example is when a developer or reviewer waits for the admin to accept the sign up request.

* **Reviewer Module**

|  |  |
| --- | --- |
| **Dependencies:** | **Reason/s:** |
| Administrator Module | An administrator controls reviewer’s activity, access rights of the reviewer is decided by the admin. |
|  |  |

**Table 4.1.1 Reviewer Module Dependencies**

* **Developer Module**

|  |  |
| --- | --- |
| **Dependencies:** | **Reason/s:** |
| Administrator Module | An administrator controls developer’s activity, access rights of the developer is decided by the admin. |

**Table 4.1.2 Developer Module Dependencies**

## Inter-process Dependencies

* Controller classes that would be using the dbConnection class would be directly dependent on the queries written/specified. The dbConnection class houses all database transactions and must therefore be used by the necessary controller classes.

## Data Dependencies

* In the database, site users are dependent but not strongly tied to the new users table. New users dictate incoming additions to the site users table, but do not affect the latter directly.
* The LOs and the LORIs however are completely isolated and unrelated. They are logically related though but have no technical dependencies.

# Interface Description

## Module Interface

Interfaces will follow the theme of the College of Computer Studies which is the colors: orange, white and black. The following interfaces will be implemented in the system.

### Homepage/Login Module

|  |  |
| --- | --- |
| **Identification** | * + - * 1. Homepage |
| * + - * 1. **Type** | * + - * 1. Form |
| * + - * 1. **Description** | * + - * 1. This will display the interface that informs the viewers all the informations about LOOP. This will also allow developers, reviewers or admin to login. |
| * + - * 1. **Function/s** | * + - * 1. **login()** |

**Table 5.1.1 – Homepage/Login Module Description**

### Admin Panel Module

|  |  |
| --- | --- |
| **Identification** | * + - * 1. Admin |
| * + - * 1. **Type** | * + - * 1. Form |
| * + - * 1. **Description** | * + - * 1. This will display all the functions that the Administrator can use. |
| * + - * 1. **Function/s** | * + - * 1. **addDeveloper(), addReviewer(), blockAccount(), unblockAccount(),**         2. **showLearningObject(), deleteLearningObject(), changePassword(),**         3. **reviewLearningObject(), downloadLearningObject(), searchAccount()** |

**Table 5.1.2 - Admin Panel Module Description**

### 5.1.3 Developer Panel Module

|  |  |
| --- | --- |
| Identification | Developer |
| Type | Form |
| Description | This will display all the functions that the Developer can use. |
| Function/s | addLearningObject(), updateLearningObject(), deleteLearningObject(), showLearningObject(), changePassword(), changeEmail(), uploadRevision(), searchLearningObject() |

**Table 5.1.3 - Developer Panel Module Description**

### 5.1.4 Reviewer Panel Module

|  |  |
| --- | --- |
| Identification | Reviewer |
| Type | Form |
| Description | This will display all the functions that the Reviewer can use. |
| Function/s | showLearningObject(), rateLearningObject(), changePassword(), changeEmail(), downloadLearningObject() |

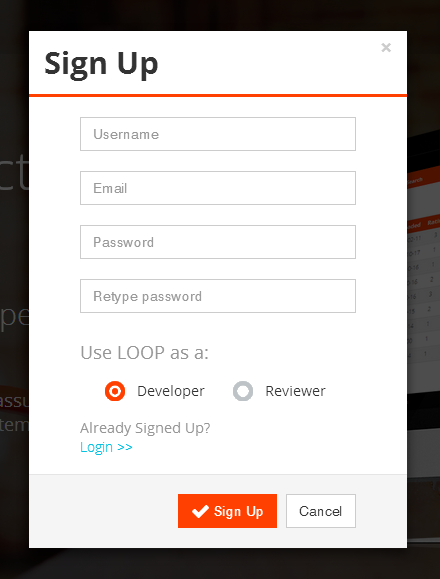
**Table 5.1.4 - Reviewer Panel Module Description**

## Process Interface

### LOOP Home Page Part 1LOOP Home Page Part 2Home Page

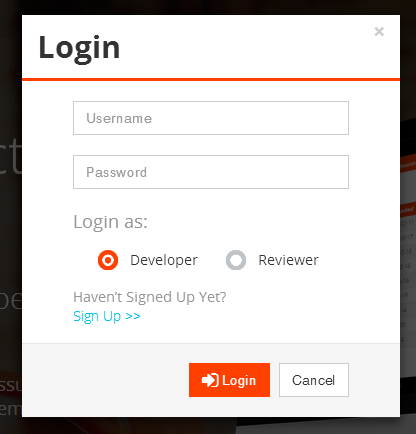
**Fig 5.2.1 – Home Page/Index**

### Sign Up Pop-up

****

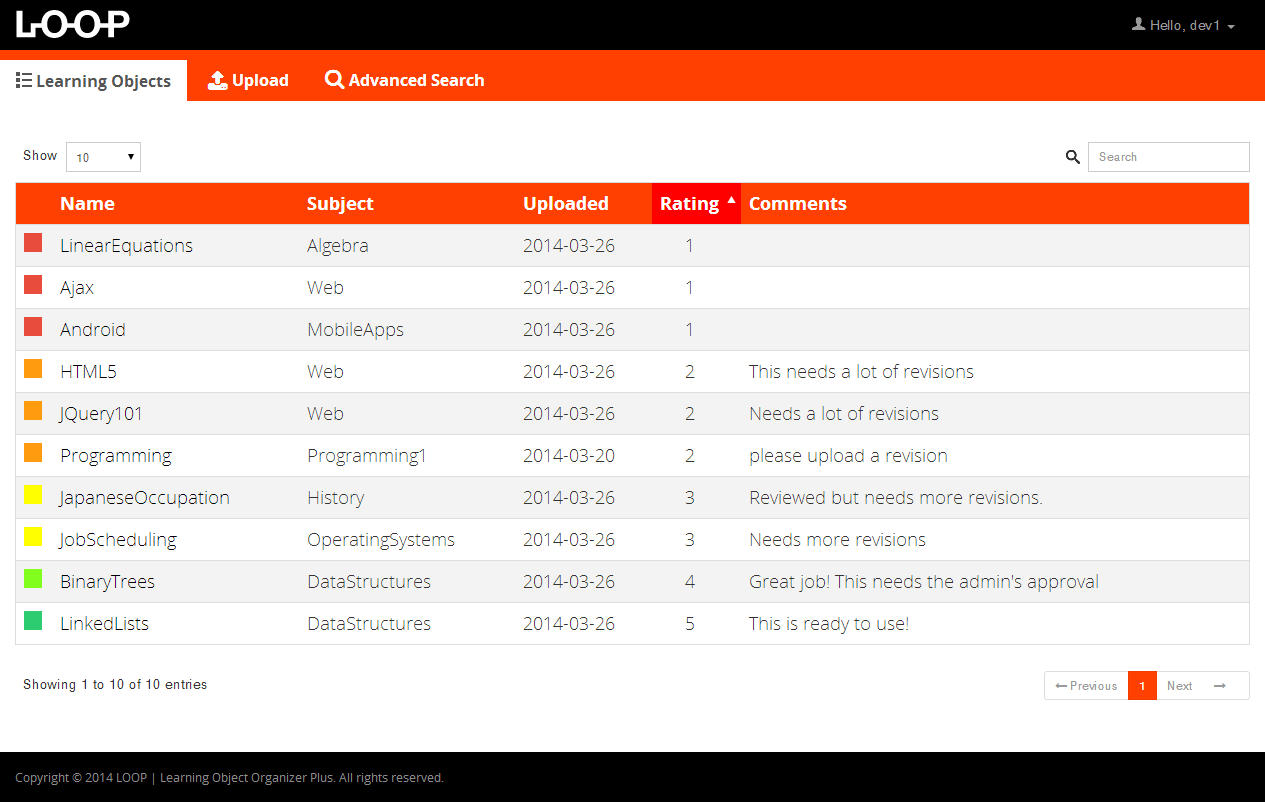
**Fig 5.2.2 – Sign Up Pop-up for all users including Admin**

### Login Pop-up

****

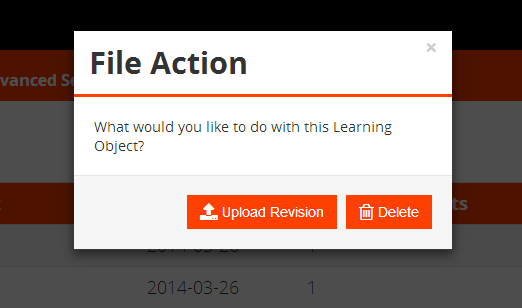
**Fig 5.2.3 – Login Pop-up for all users including Admin**

### Main View Page for Developer

****

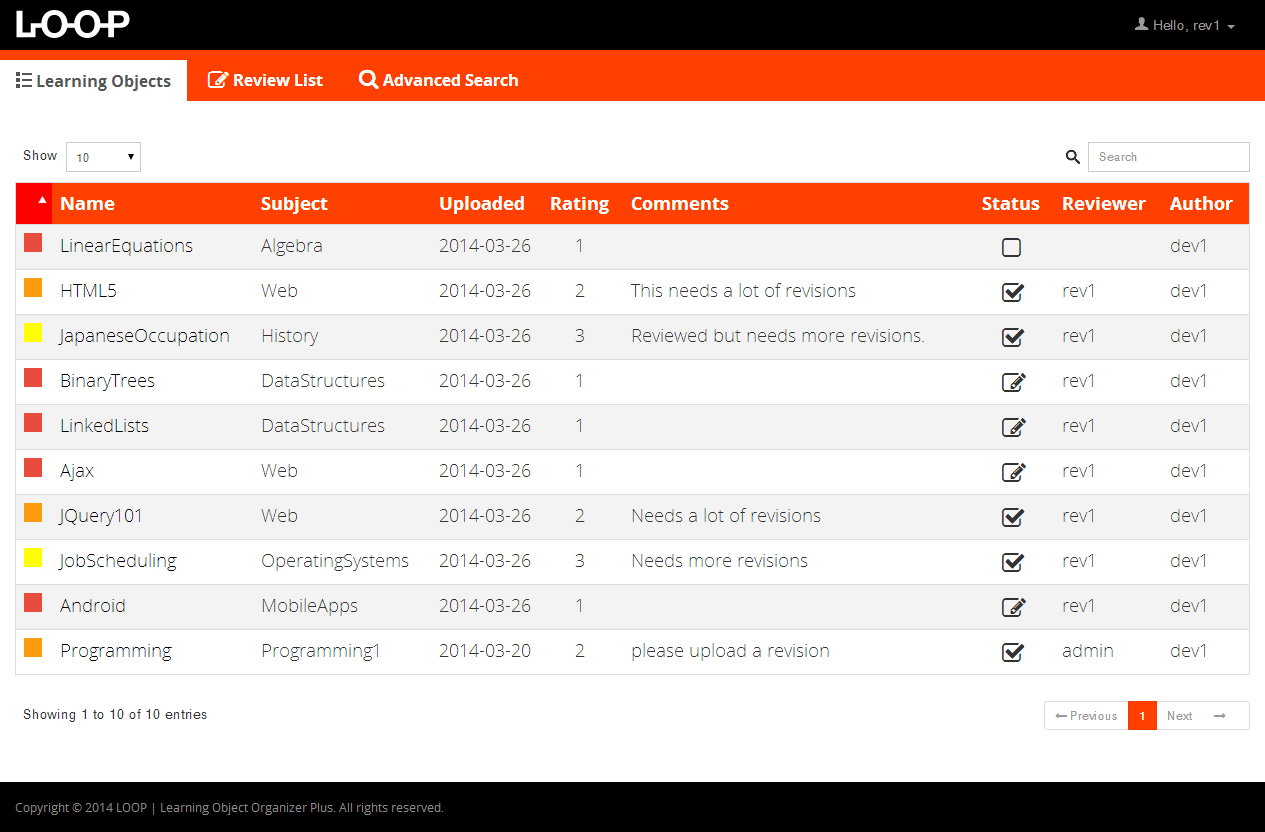
**Fig 5.2.4 – Main View Page for Developer**

### File Action Pop-up for Developer

****

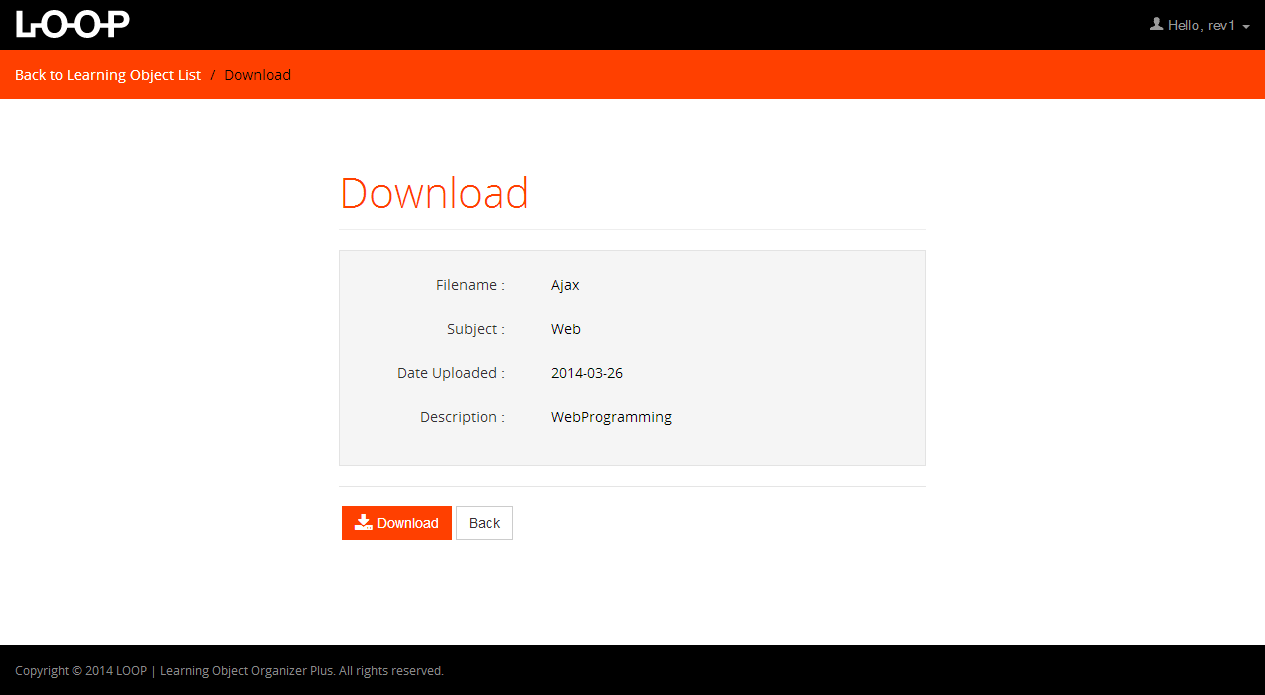
**Fig 5.2.5 – File Action Pop-up for Developer**

### Main View Page for Reviewer

****

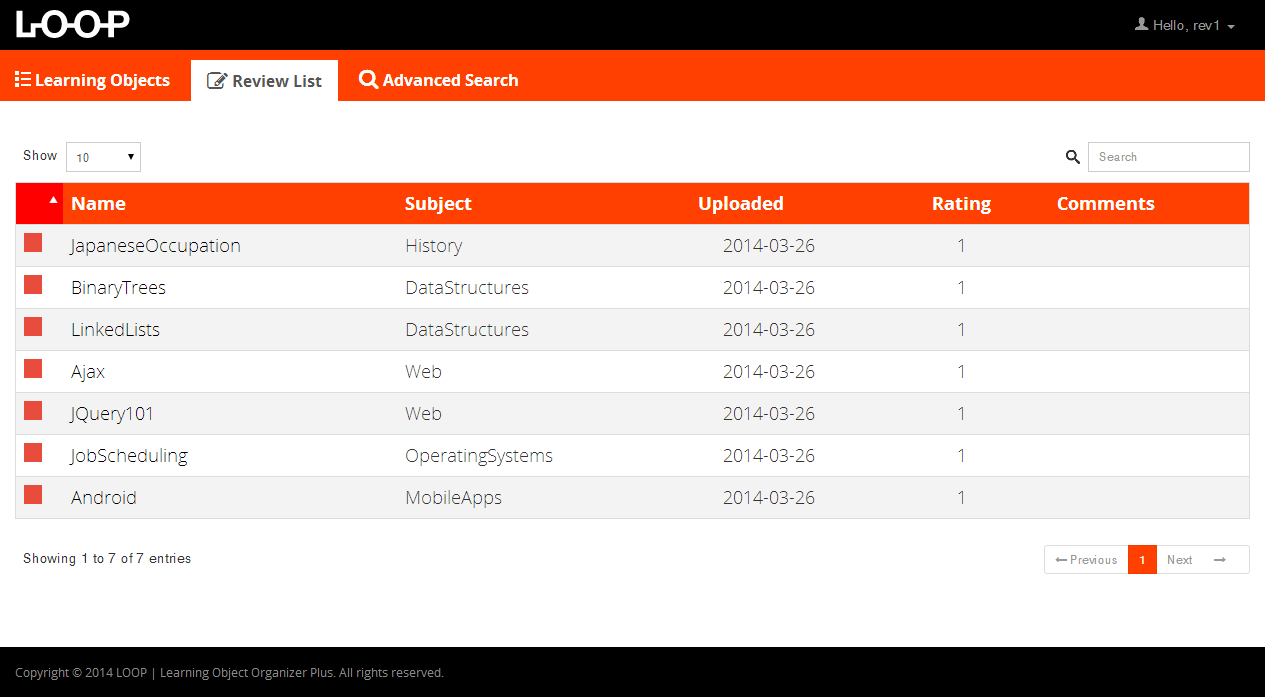
**Fig 5.2.6 – Main View Page for Reviewer**

### Download Page for Reviewer

****

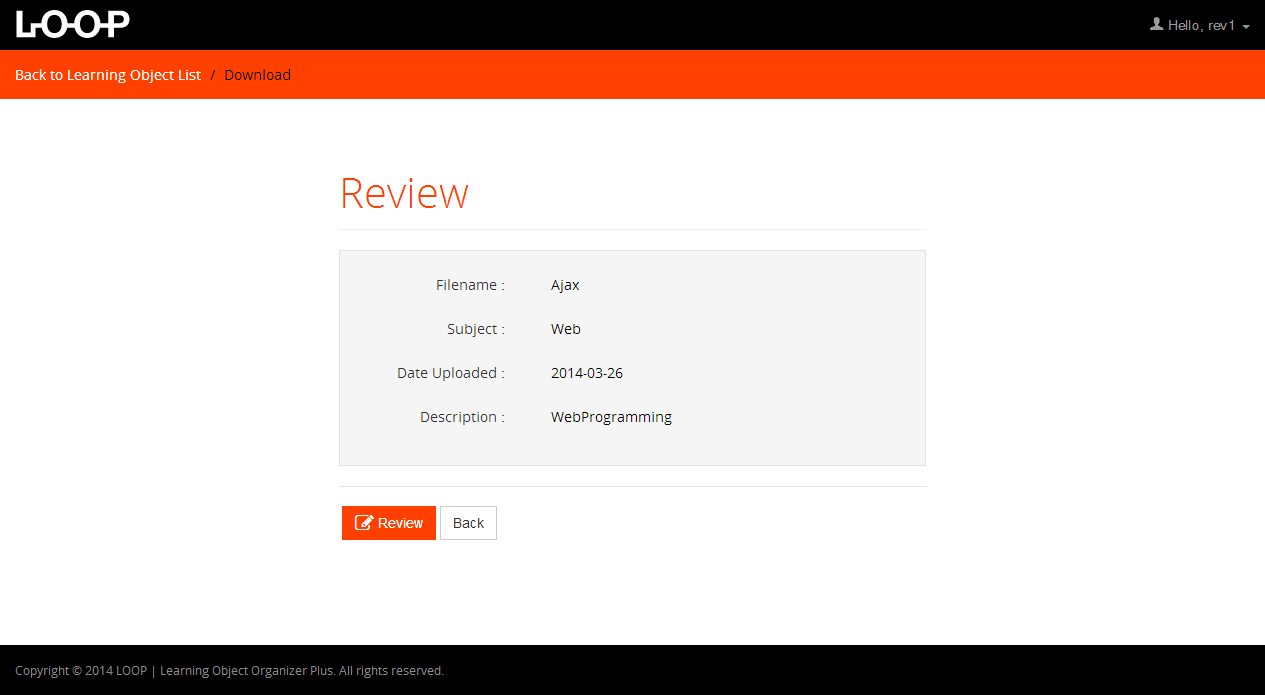
**Fig 5.2.7 – Download Page for Reviewer**

### Review Later List for Reviewer (After Download)

****

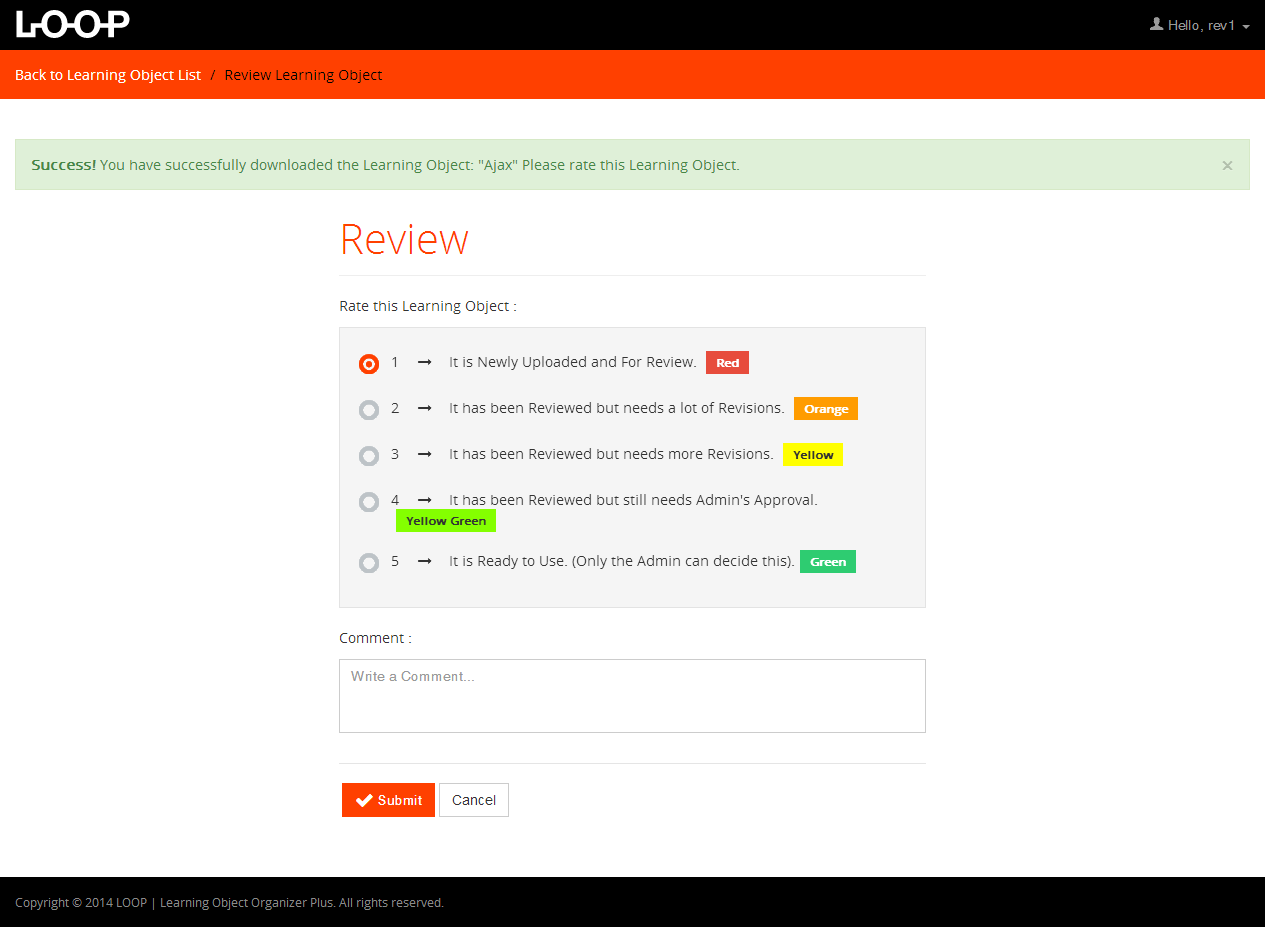
**Fig 5.2.8 – Review Later List for Reviewer (After Download)**

### Review Option for Reviewer (After Download)

****

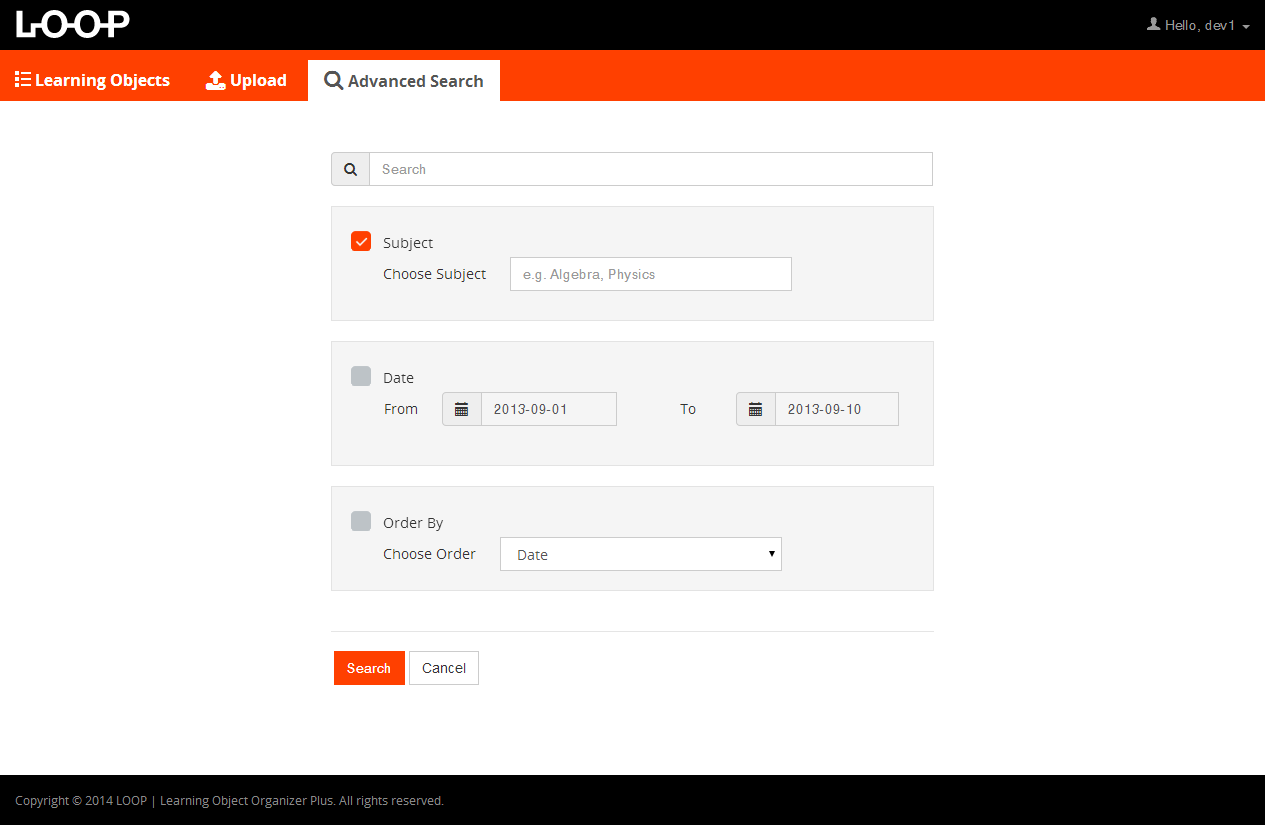
**Fig 5.2.9 – Review Option f** **or Reviewer (After Download)**

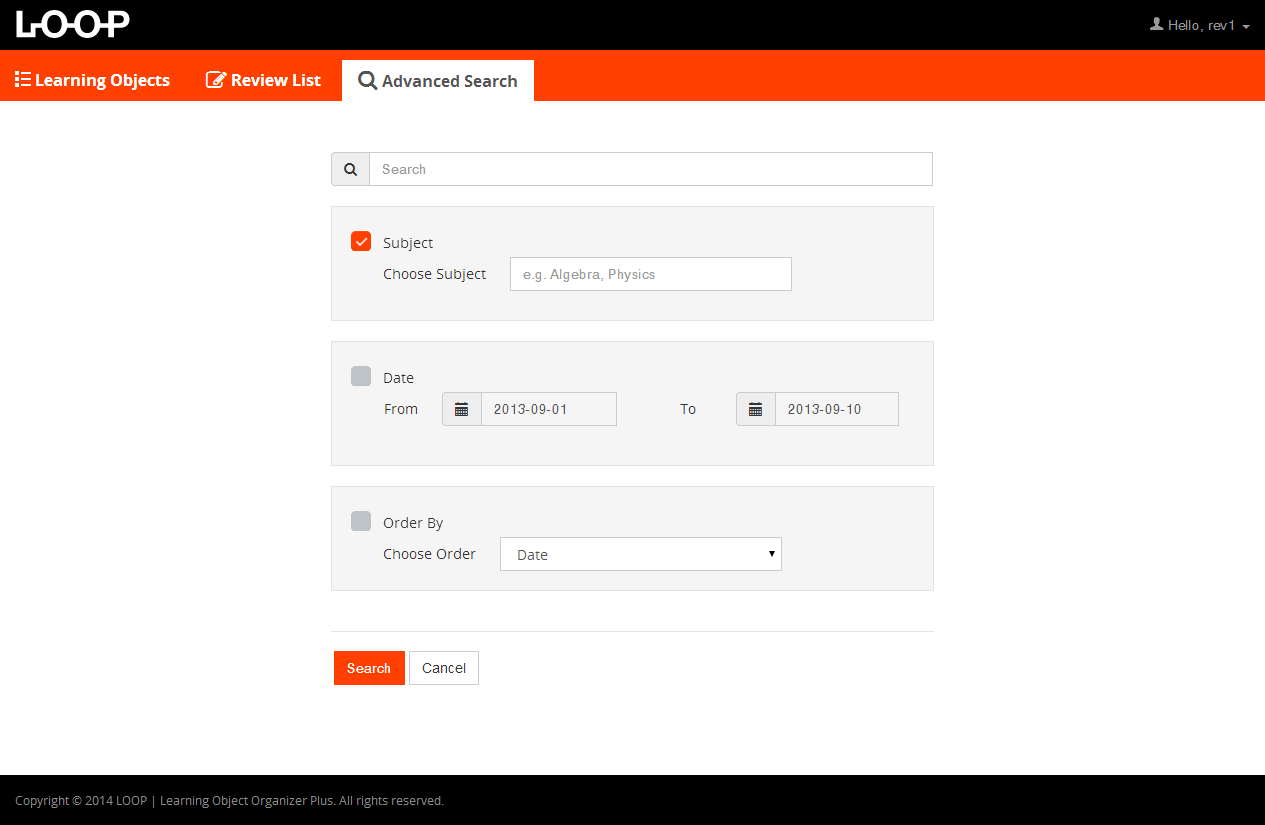
### Review Page (Only for Reviewer)

****

**Fig 5.2.10 - Review Page (Only for Reviewer)**

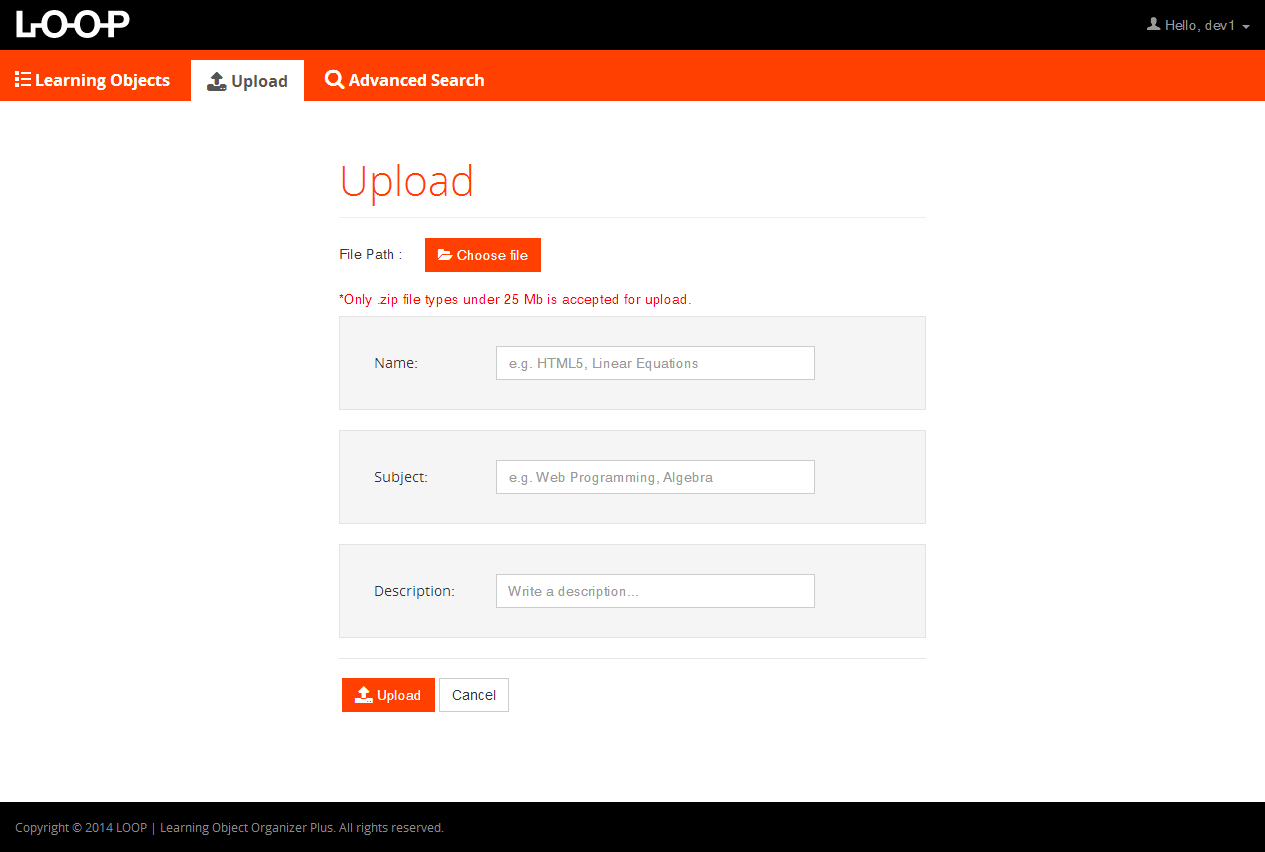
### Advanced Search Page

****

****

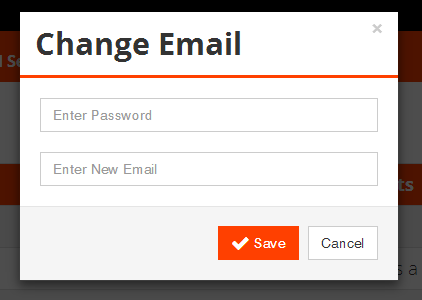
**Fig 5.2.11 – Advanced Search Page (Both Developer and Reviewer)**

### Upload Page (Only for Developer)

****

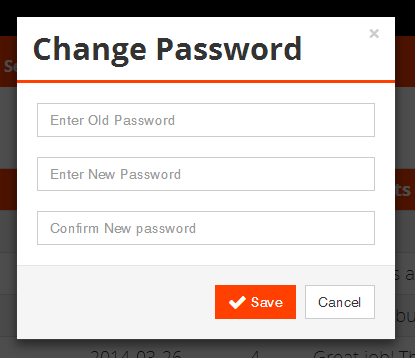
**Fig 5.2.12 – Upload Page (Only for Developer)**

### Change Email Pop-up

****

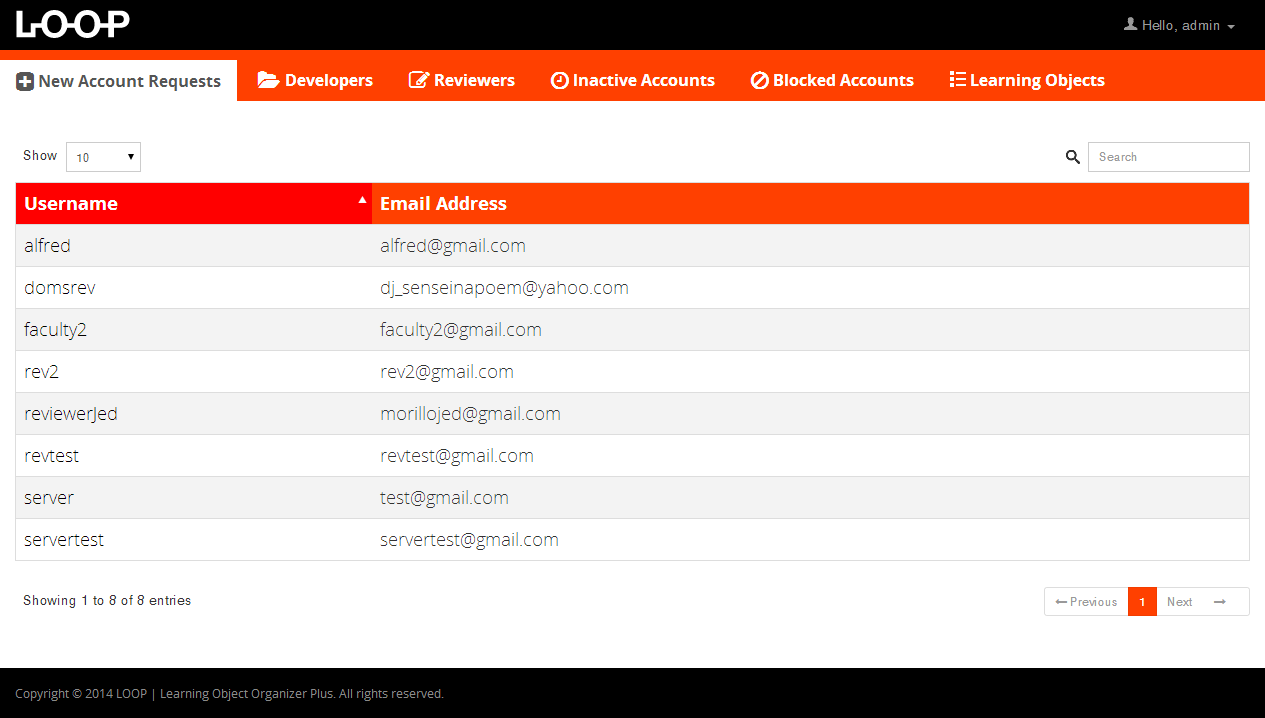
**Fig 5.2.13 – Change Email Pop-up**

### Change Password Pop-up

****

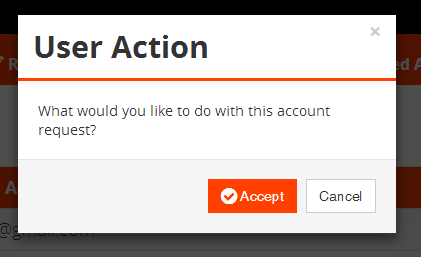
**Fig 5.2.14 – Change Password Pop-up**

### New Account Requests Tab for Admin

****

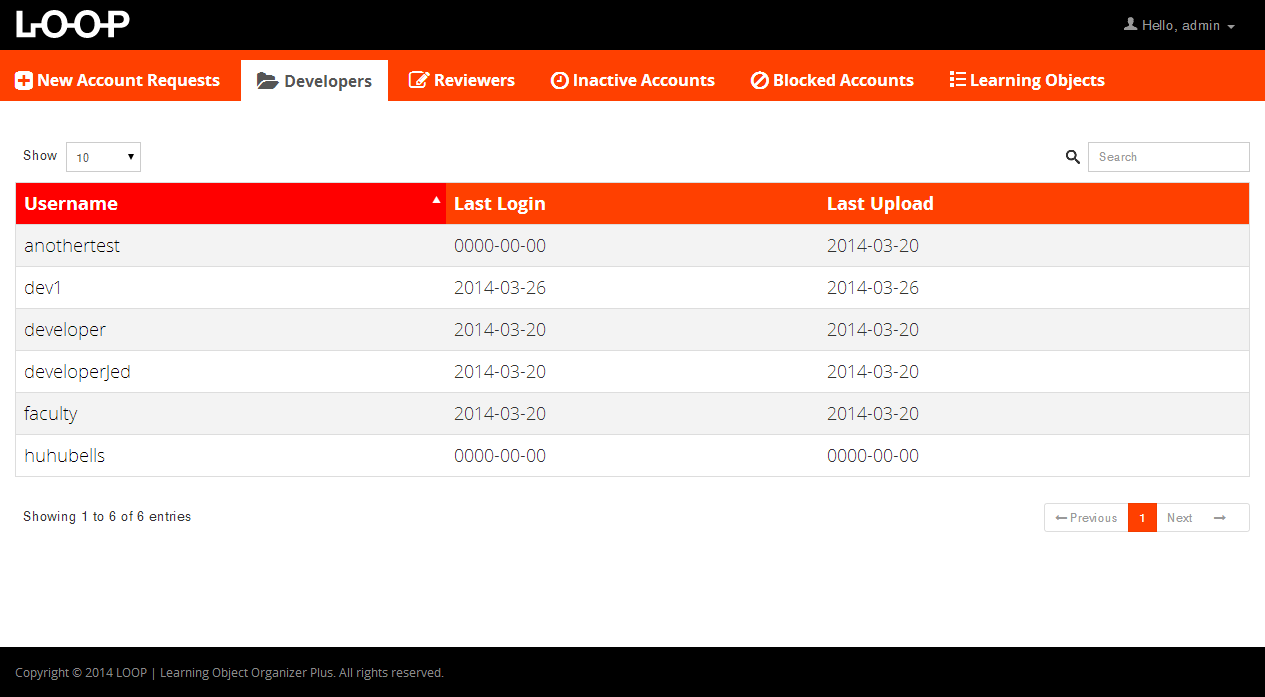
**Fig 5.2.15 – New Account Requests Tab for Admin**

### User Action for New Account Requests Tab (Admin)

****

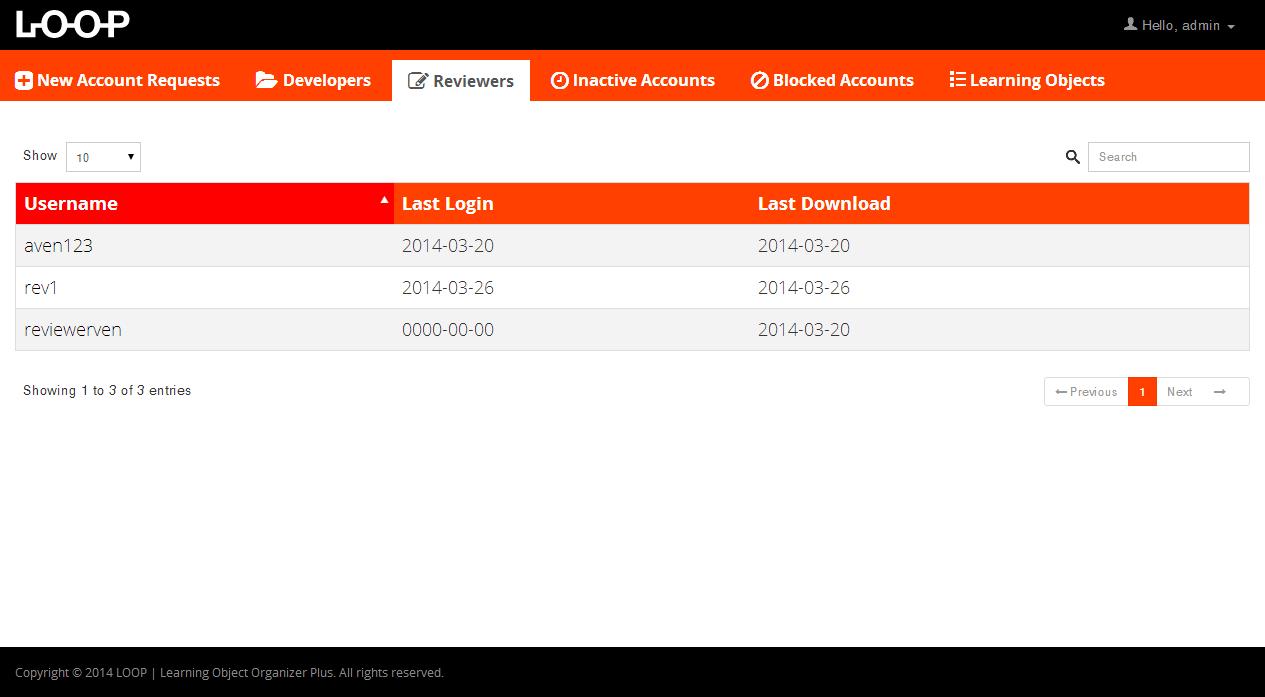
**Fig 5.2.16 – User Action for New Account Requests Tab (Admin)**

### Main View Page for Admin (Developer Tab)

****

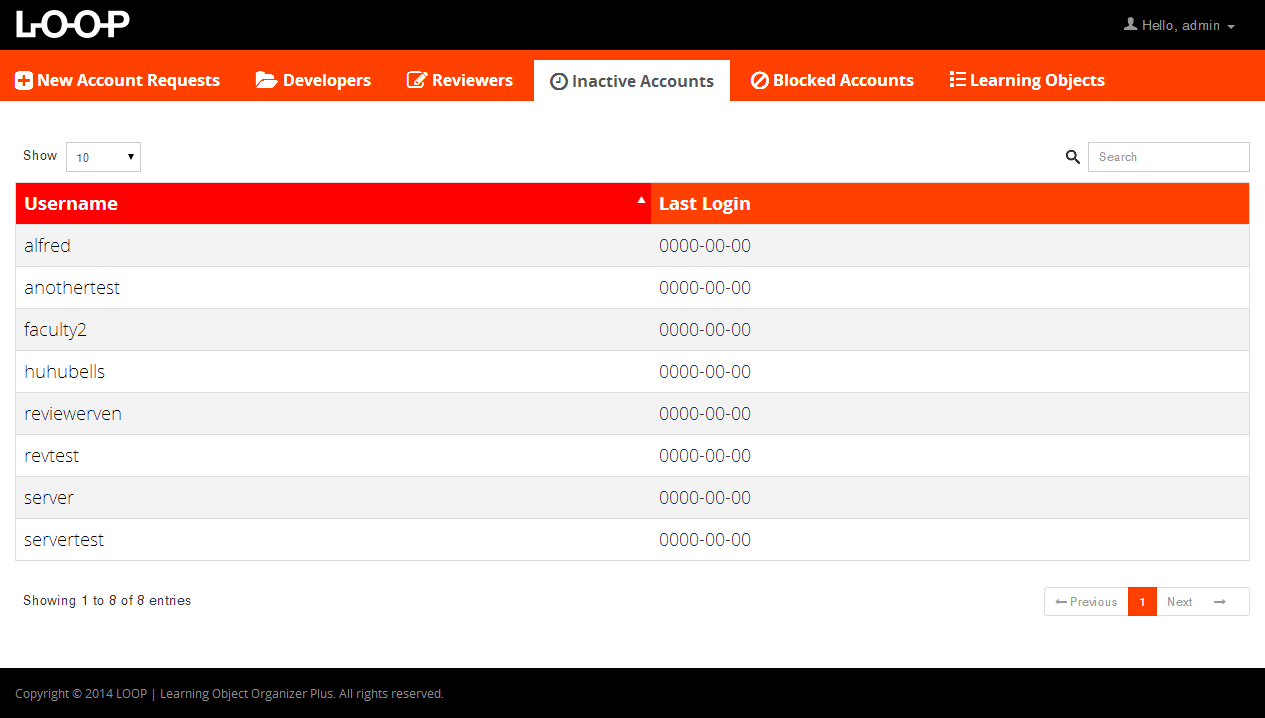
**Fig 5.2.17 – Main View Page for Admin (Developer Tab)**

### Main View Page for Admin (Reviewer Tab)

****

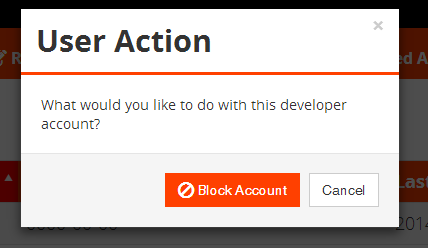
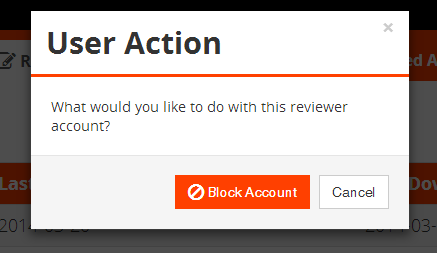
**Fig 5.2.18 – Main View Page for Admin (Reviewer Tab)**

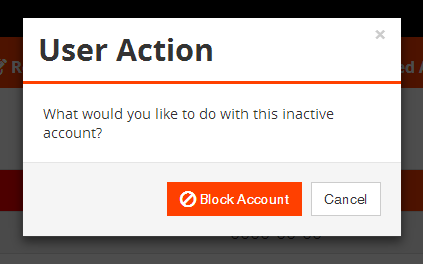
### Main View Page for Admin (Inactive Accounts Tab)

****

**Fig 5.2.19 – Main View Page for Admin (Inactive Accounts Tab)**

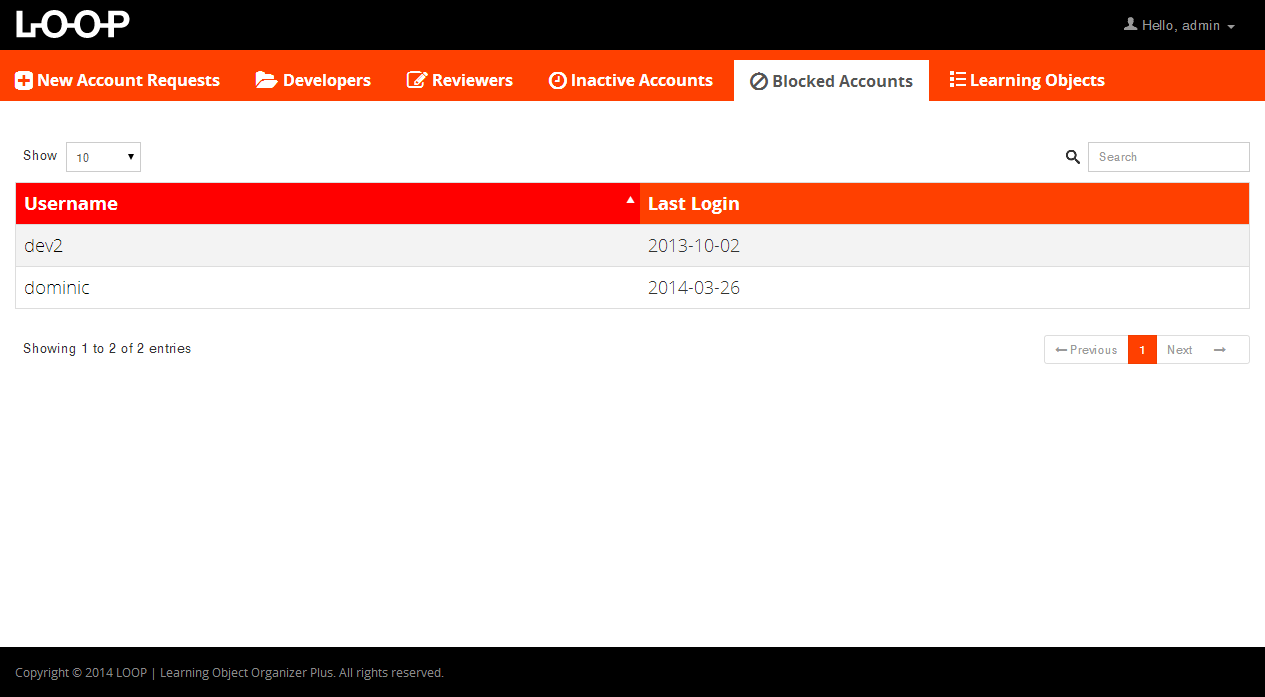
### User Action for Admin Pages (Developer, Reviewer, Inactive Accounts Tab)





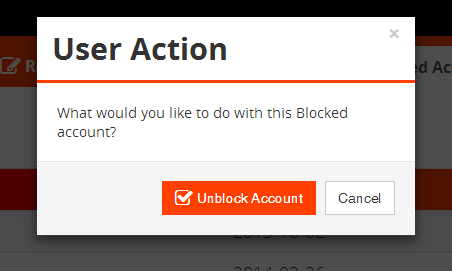
**Fig 5.2.20 – User Action for Admin Pages (Developer, Reviewer, Inactive Accounts Tab)**

### Main View Page for Admin (Blocked Accounts Tab)

****

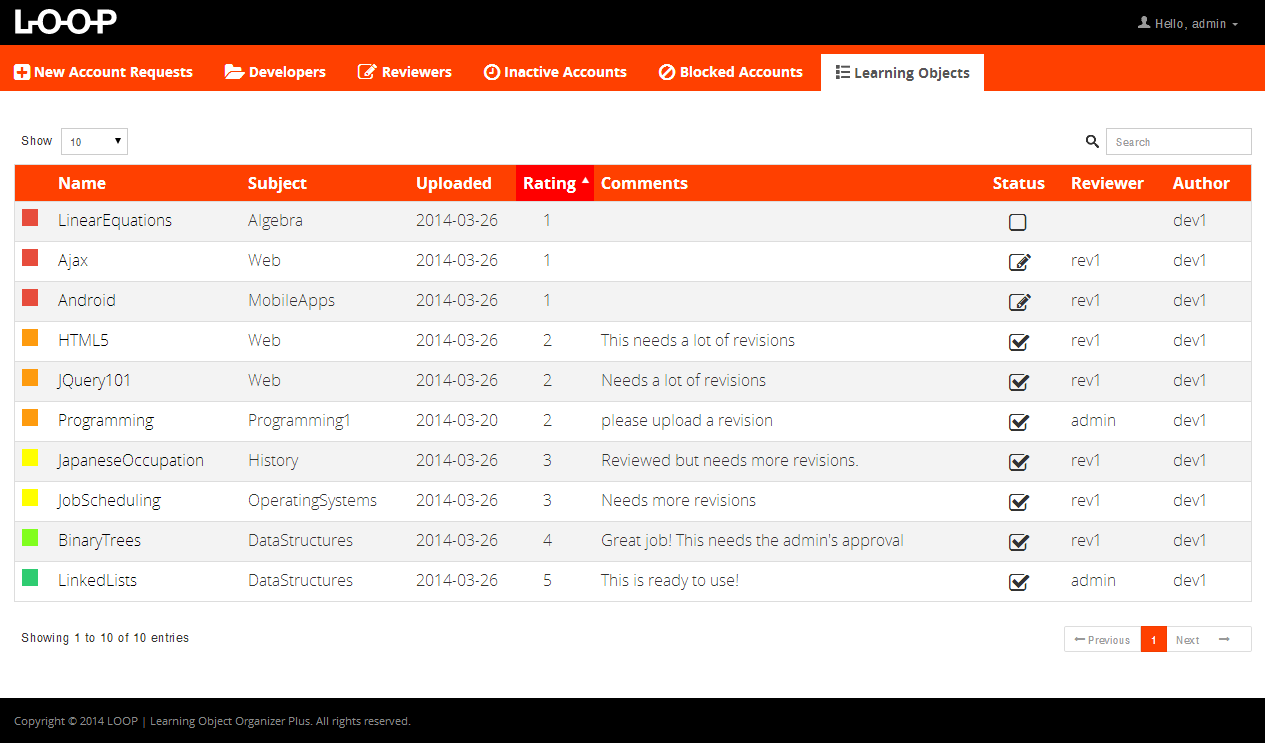
**Fig 5.2.21 – Main View Page for Admin (Blocked Accounts Tab)**

### User Action for Admin Page (Blocked Accounts Tab)

****

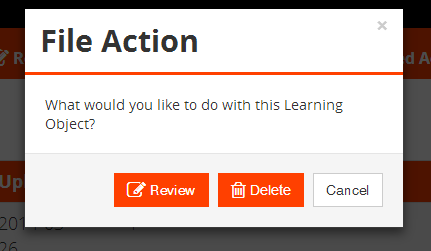
**Fig 5.2.22 – User Action for Admin Page (Blocked Accounts Tab)**

### Main View Page for Admin (Learning Objects Tab)

****

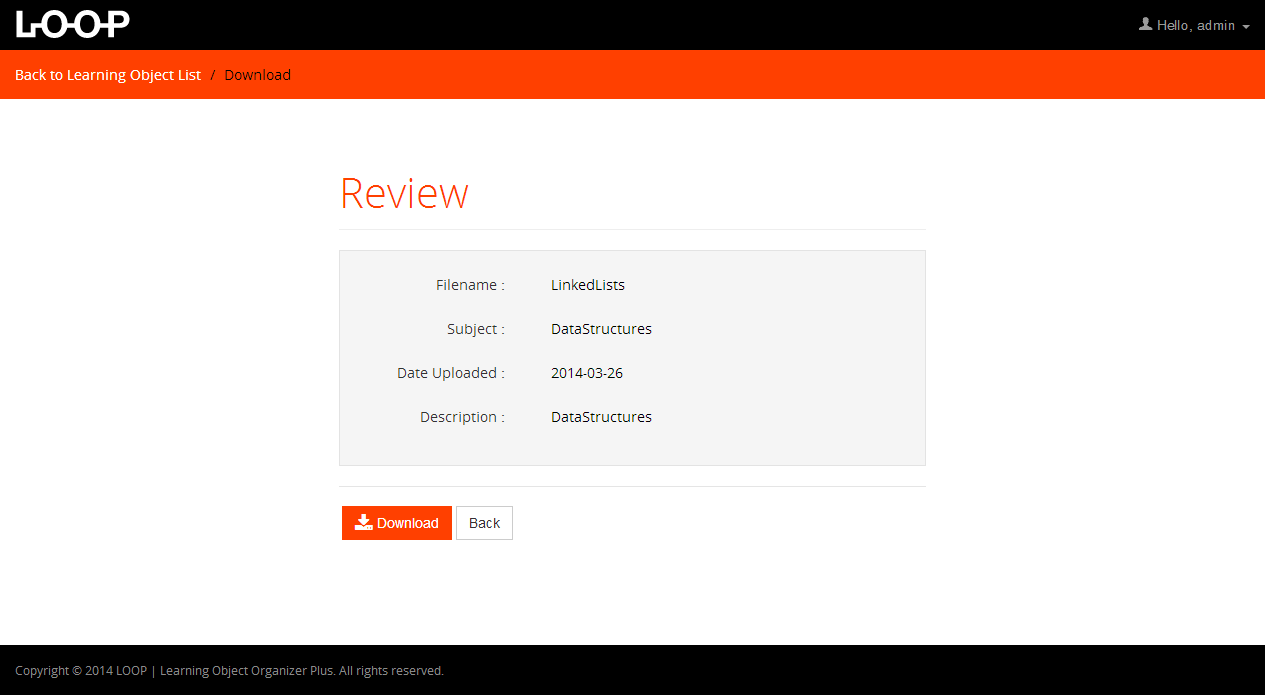
**Fig 5.2.23 – Main View Page for Admin (Learning Objects Tab)**

### File Action for Admin Page (Learning Objects Tab)

****

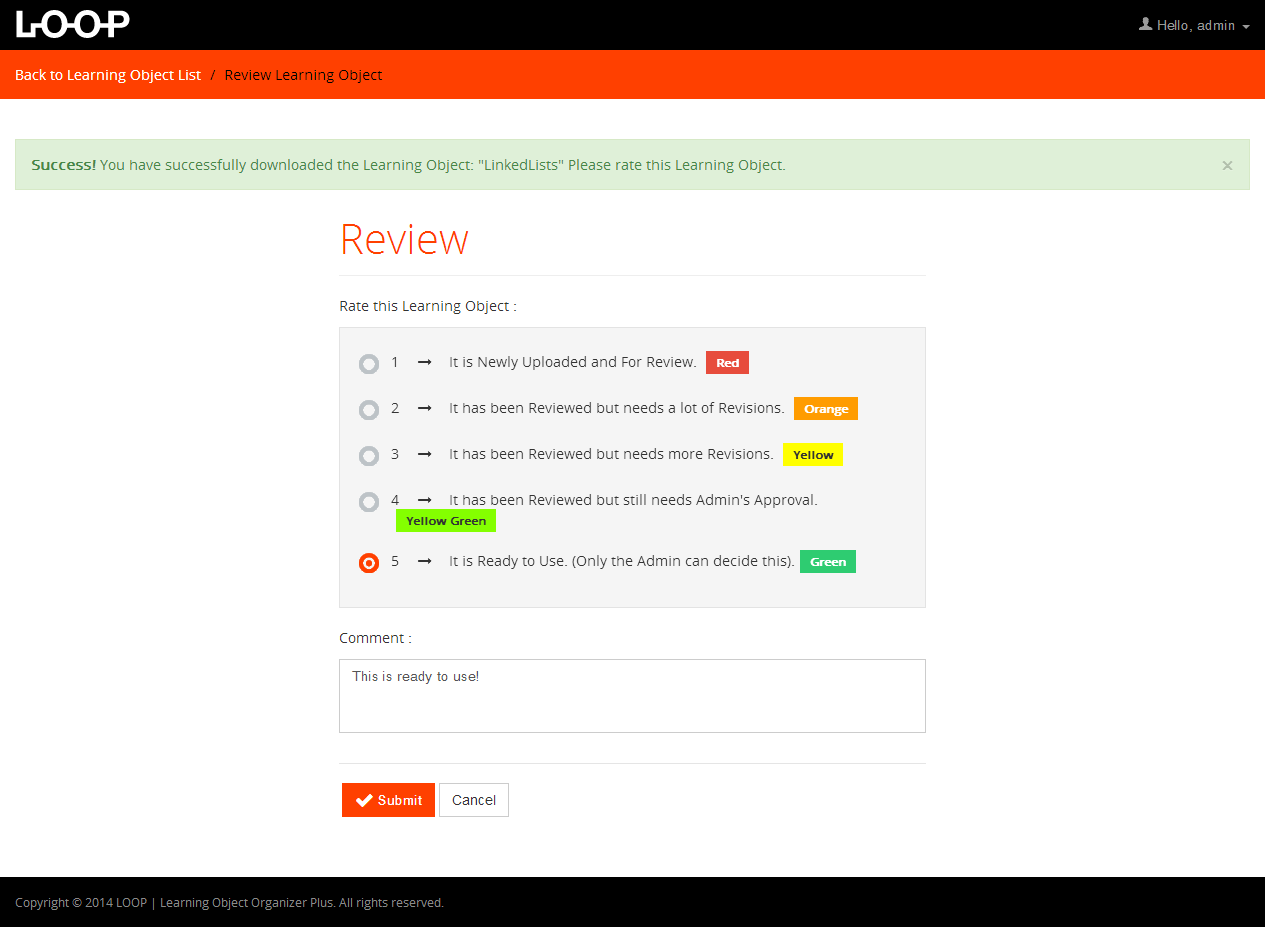
**Fig 5.2.24 – File Action for Admin Page (Learning Objects Tab)**

### Download Page for Admin (After you click Review)

****

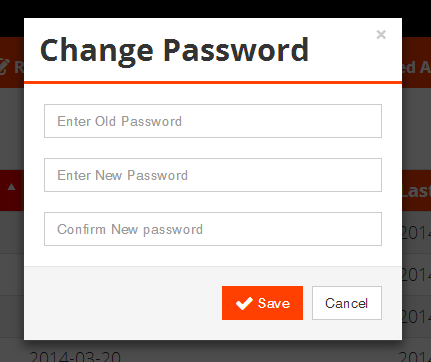
**Fig 5.2.25 – Download Page for Admin (After you click Review)**

### Review Page for Admin (After Download)

****

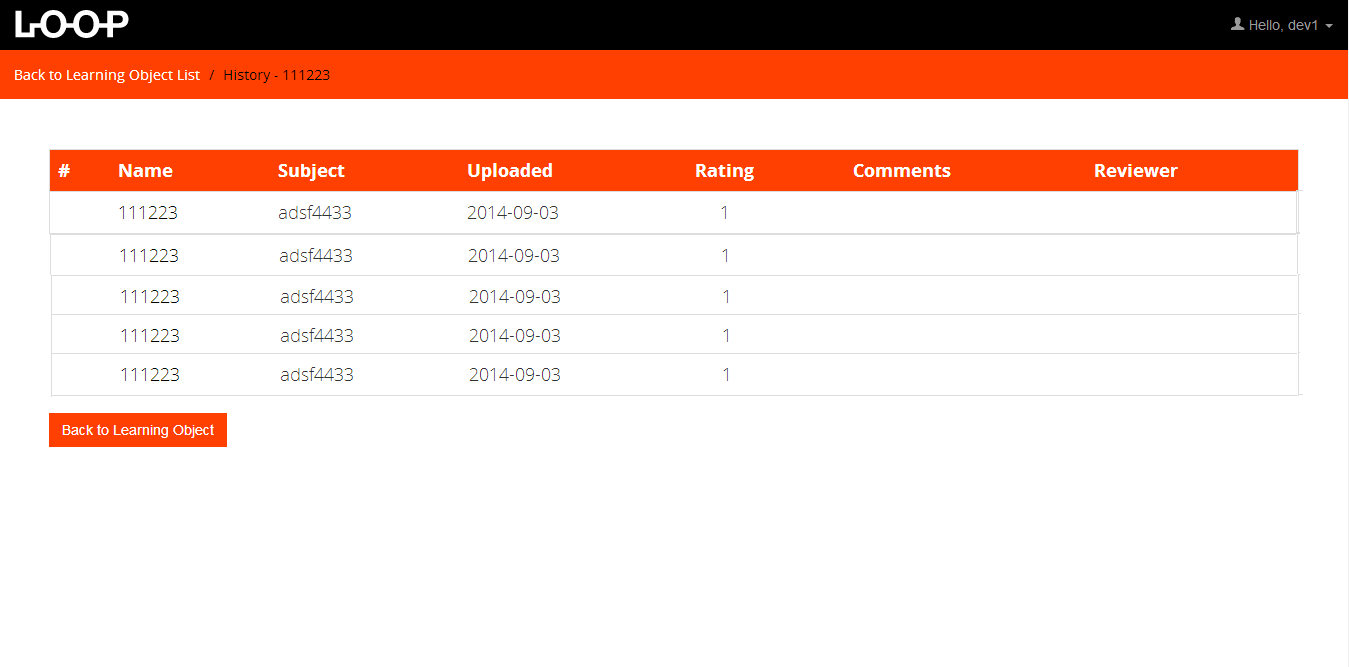
**Fig 5.2.26 – Review Page for Admin (After Download)**

### Change Password for Admin



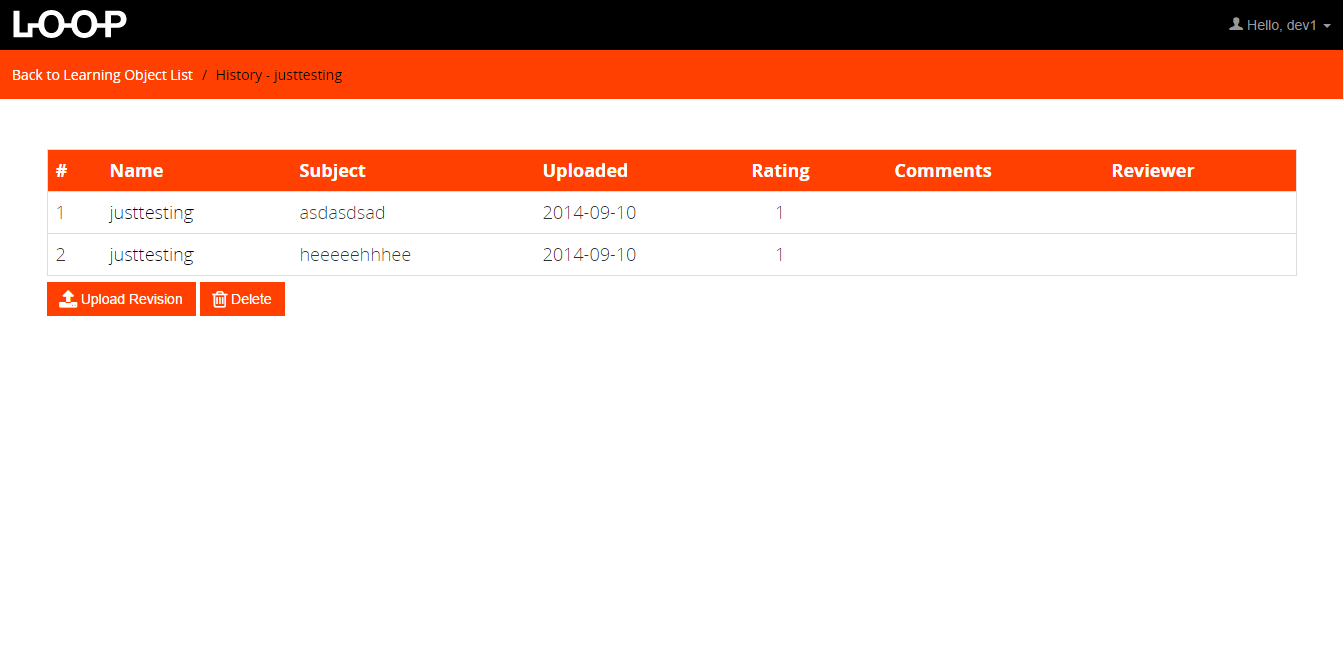
**Fig 5.2.27 – Change Password for Admin**

### 5.2.28 HistoryView Learning Object / Learning Element



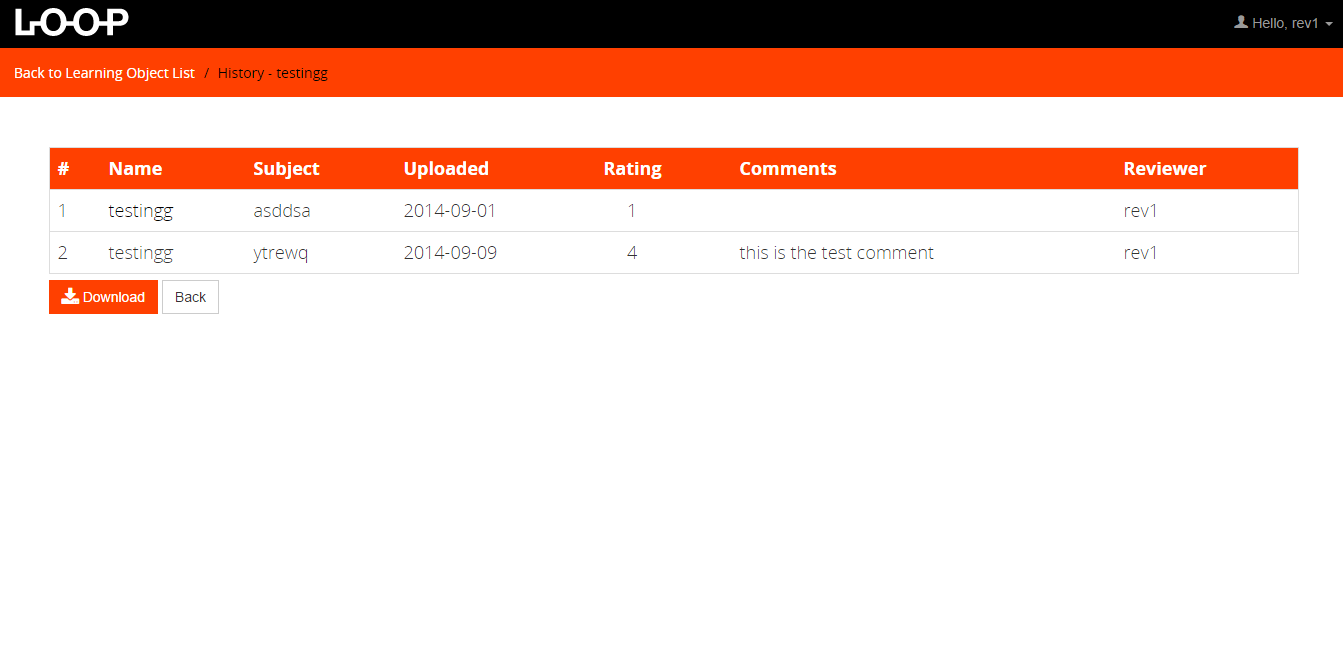
**Fig 5.2.28 – History View of LO’s on all users**

### 5.2.29 History View for Developer Learning Object / Learning Element

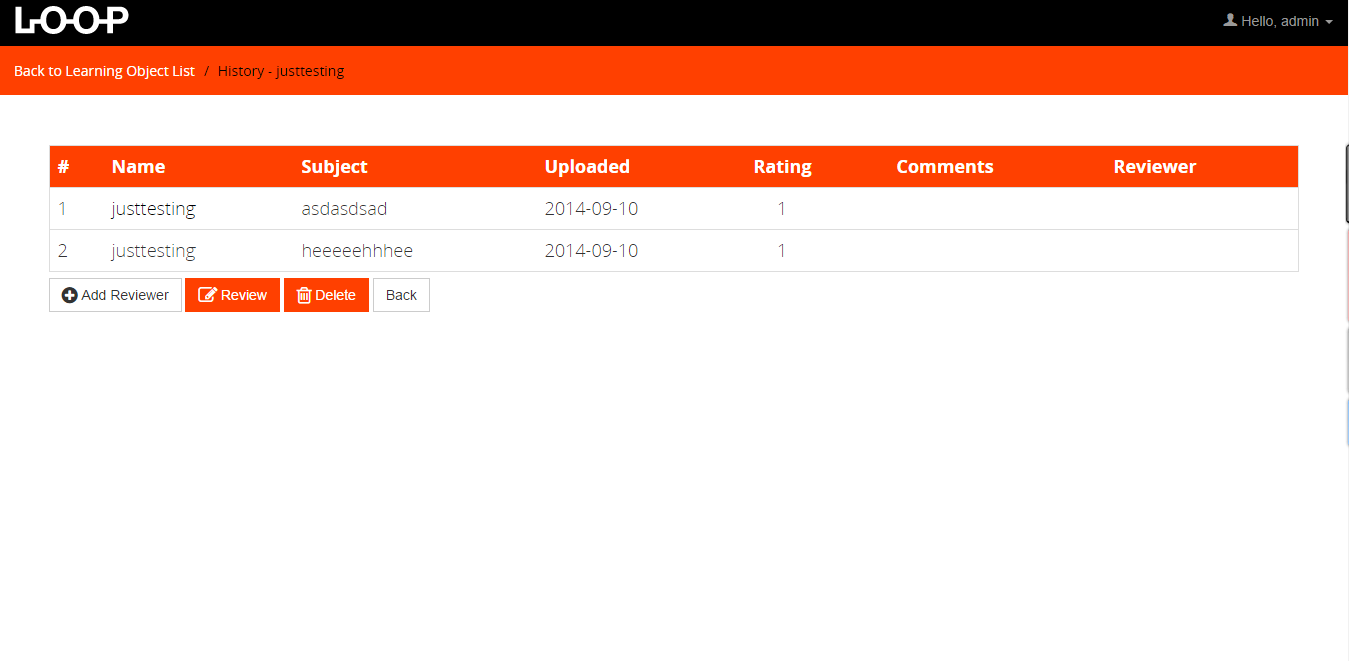


**Fig 5.2.29 – History View for Developer**

### 5.2.30 History View for Reviewer Learning Object / Learning Element

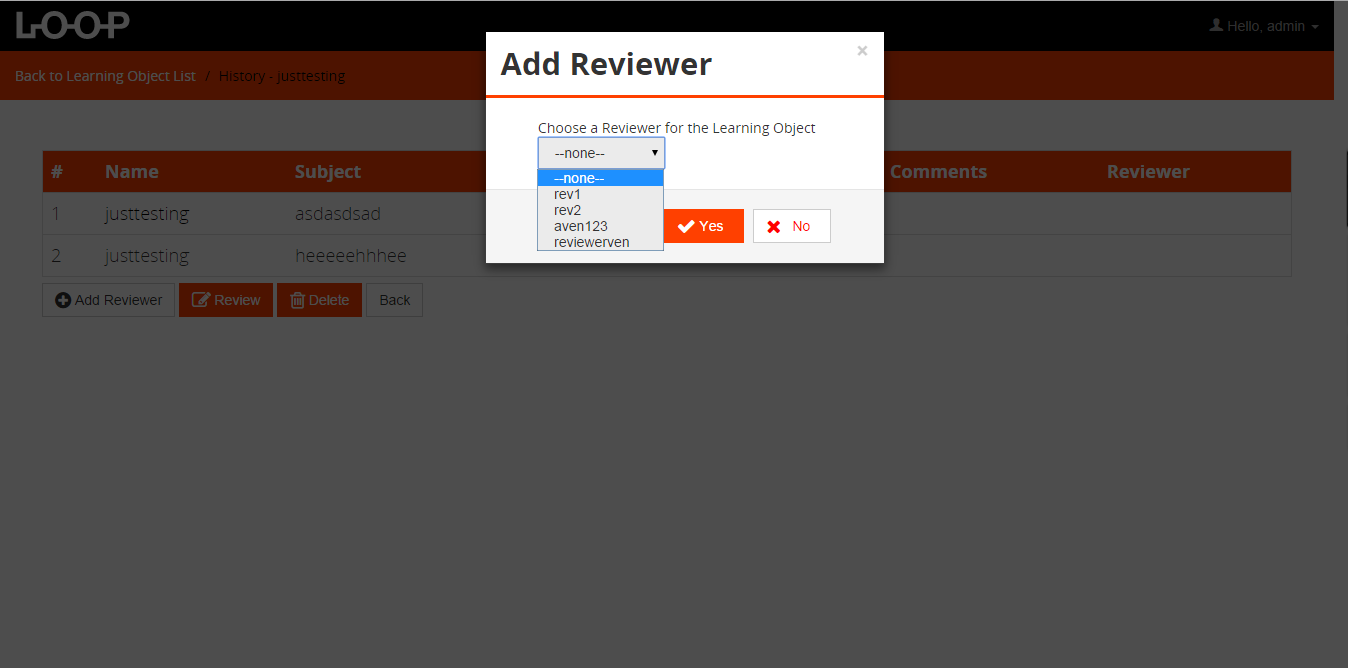
**Fig 5.2.30 – History View for Developer**

### 5.2.31 History View for Admin Learning Object / Learning Element



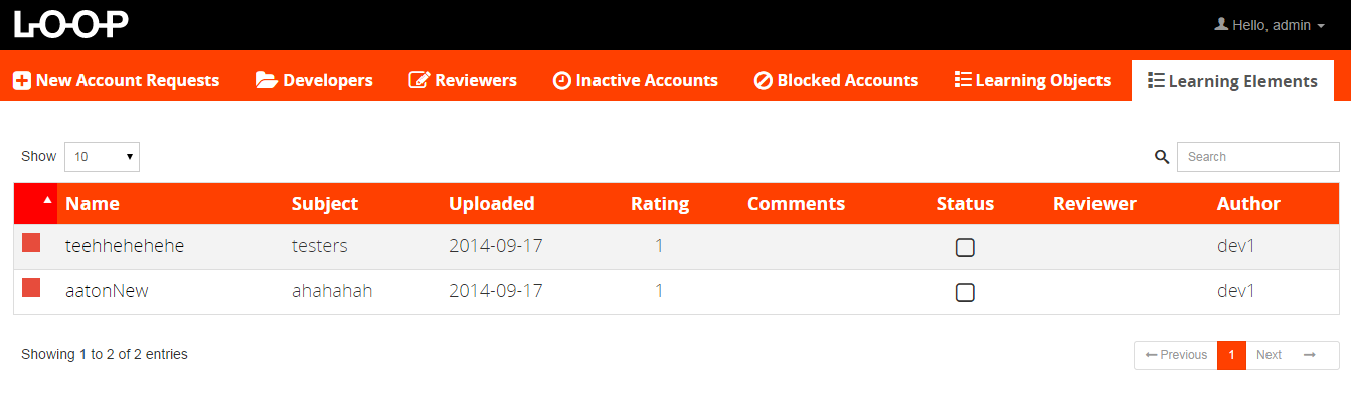
**Fig 5.2.31 – History View for Admin**

### 5.2.32 Assigning For Reviewer



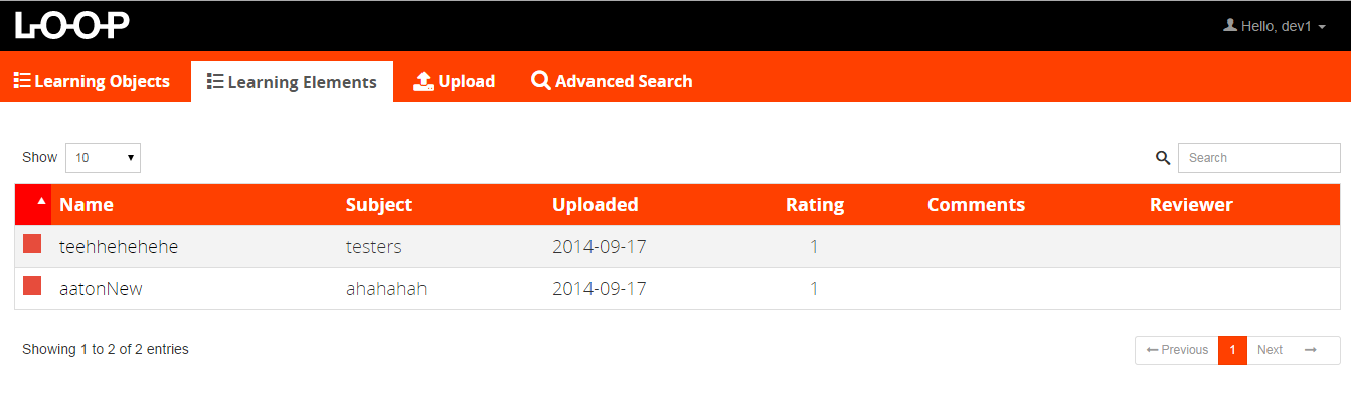
**Fig 5.2.32 – Assigning Reviewer**

### 5.2.33 View Learning Element for Admin



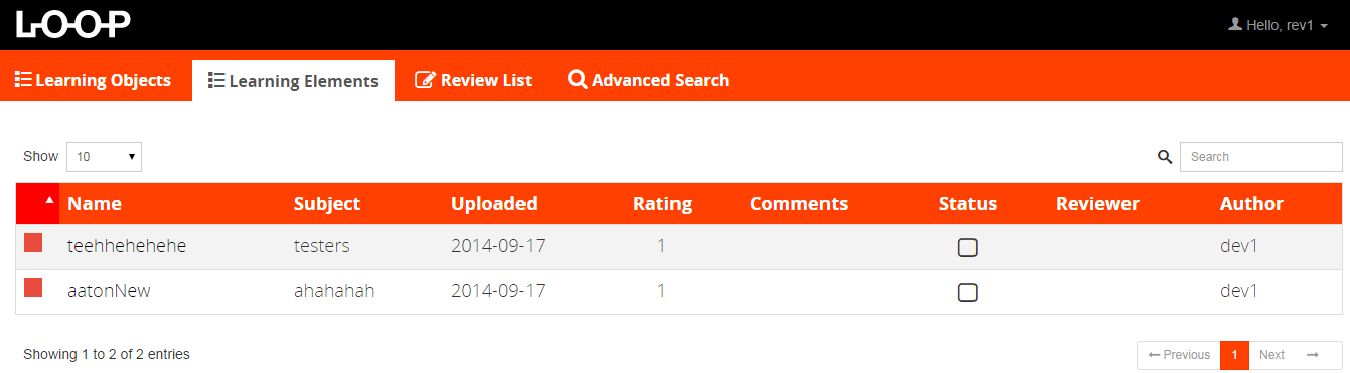
**Fig 5.2.33 – View Learning Element for Admin**

### 5.2.34 View Learning Element for Developer



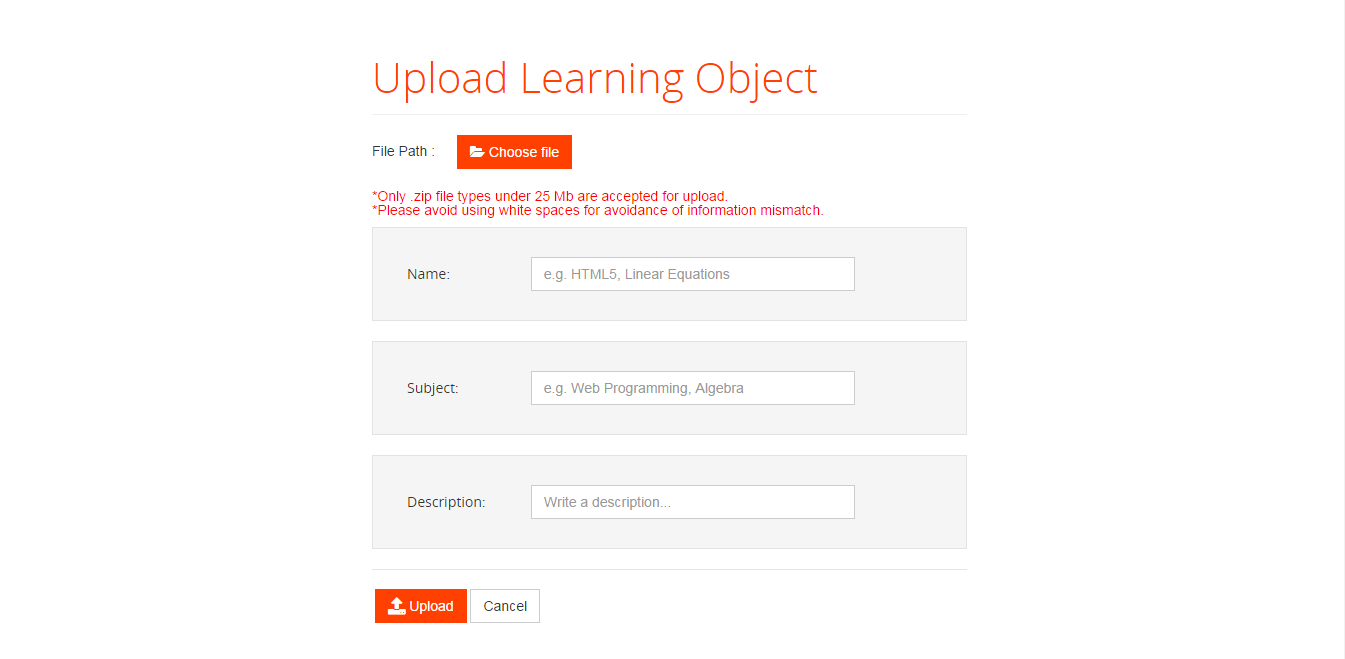
**Fig 5.2.34 – View Learning Element for Developer**

### 5.2.35 View Learning Element for Reviewer



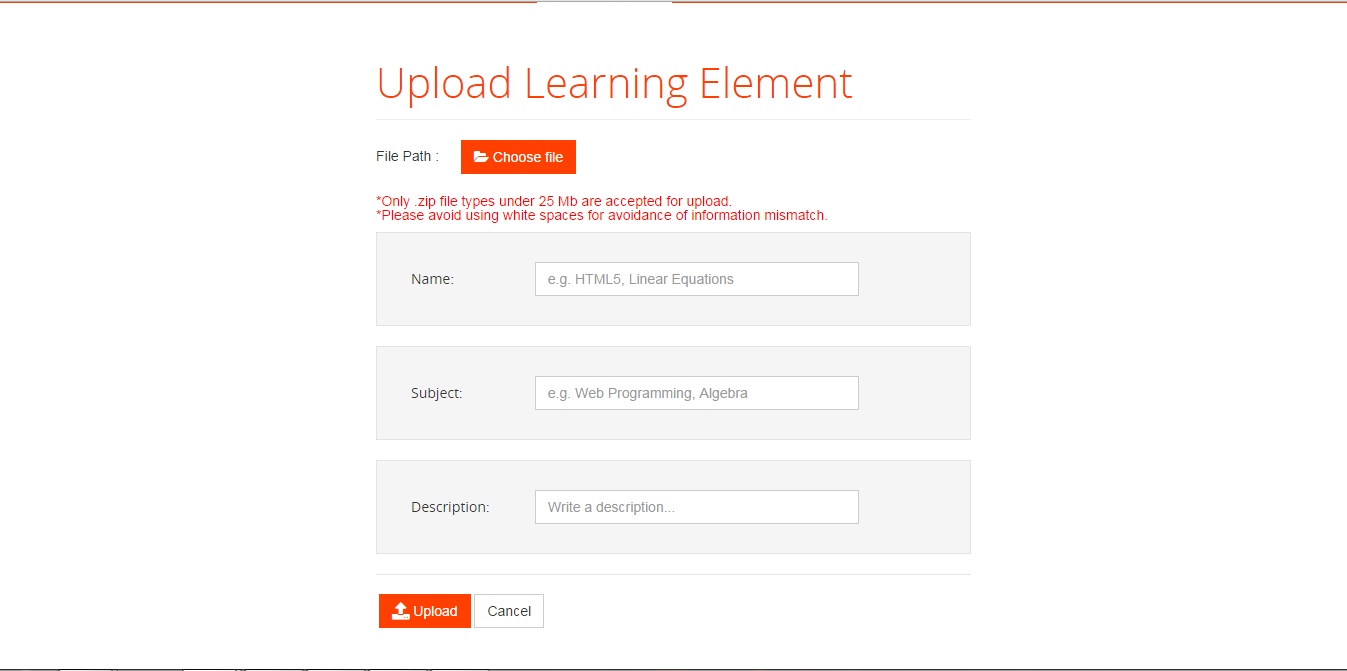
**Fig 5.2.35 – View Learning Element for Reviewer**

### 5.2.36 Upload of Learning Object



**Fig 5.2.36 – Upload of Learning Object**

### 5.2.37 Upload of Learning Element



**Fig 5.2.37 – Upload of Learning Element**

# Detailed Design

## Module Detailed Design

### Administrator User Module Detail

Name: **Admin Module**

Type: PHP Classes

Description: This part describes the operations done by the admin user-type in the system.

Attributes: (developer:Developer), (reviewer:Reviewer)

Operations:

Name : **changePassword**

Arguments : **changePassword**(pass:String, newPass:String)

Condition : Connected to database.

Admin user-type is logged in.

Flow : Admin clicks on change password and fills in the pop up accordingly.

Database is queried for admin info to compare with the parameters.

Result : if password matches database records, password is updated. Else, send failure notice to user

Name : **addDeveloper**

Argument : addDeveloper(developer:Developer)

Condition : Connected to database.

Admin user-type is logged in.

Flow : Admin encodes Developer’s username, password and email address.

System stores the entry into the database.

Result : A new Developer account is added.

Name : **removeDeveloper**

Argument : removeDeveloper(developer:Developer)

Condition : Connected to database.

Admin user-type is logged in.

Developer accounts exist in database.

Flow : Admin selects Developer accounts to delete.

System fetches the Developer account information, removes it from the database.

Result : A Developer account is deleted.

Name : **addReviewer**

Argument : addReviewer(reviewer:Reviewer)

Condition : Connected to database.

Admin user-type is logged in.

Flow : Admin encodes username, password and email address.

System stores the entry into the database.

Result : A new Reviewer account is added.

Name : **removeReviewer**

Argument : removeReviewer(reviewer:Reviewer)

Condition : Connected to database.

Admin user-type is logged in.

Reviewer accounts exist in database.

Flow : Admin selects Reviewer accounts to delete.

System fetches Reviewer account information, removes it from the database.

Result : A Reviewer account is deleted.

Name : **showLearningObjects**

Argument : showLearningObjects()

Condition : Connected to database.

Administrator user-type is logged in.

Developer’s learning objects are present in the database.

Flow : Fetches the learning objects uploaded by the developer in the database.

Result : The list of learning objects with ratings and comments is displayed and

Can be viewed one by one.

Name : **deleteLearningObject**

Argument : deleteLearningObject(learningObject:LearningObject)

Condition : Connected to database.

Developer user-type is logged in.

A current uploaded learning object exists in the database.

Flow : Developer can only delete the uploaded learning objects.

Result : Developer’s learning object is deleted from the database.

### Developer User Module Detail

Name: **Developer**

Type: PHP Classes

Description: This part describes the operations done by the developer user-type in the system.

Attributes: (learningObject:LearningObject)

Operations:

Name : **changePassword**

Arguments : **changePassword**(username:String, password:String, newPass:String)

Condition : Connected to database and external user management system.

Developer user-type is logged in.

Flow : Developer clicks on change password and fills out the pop up.

Use change email function of external user management system.

Result : if username and password matches, developer password will be updated, Else, display error message to the developer.

Name : **changeEmail**

Arguments : **changeEmail**(username:String, password:String, new:String)

Condition : Connected to external user management system.

Developer is logged in.

Flow : Developer clicks on change email and fills out the pop up.

Use change email function of external user management system.

Result : if username and password matches, developer email will be updated, Else, display error message to the developer.

Name : **addLearningObject**

Argument : **addLearningObject** (learningObject:LearningObject)

Condition : Connected to database.

Developer user-type is logged in.

Flow : Developer uploads a Learning Object.

Co-requisite : Reviewer must confirm learning object to be considered valid.

Result : Developer’s learning object is saved in the database.

Name : **updateLearningObject**

Argument : updateLearningObject(learningObject:LearningObject)

Condition : Connected to database.

Developer user-type is logged in.

A current uploaded Learning Object is saved in the database.

Flow : Developers can update the learning objects by deleting and re-uploading

the learning object.

Co-requisite : Reviewer must confirm learning object to be considered valid.

Result : Developer’s learning object is updated and saved in the database.

Name : **deleteLearningObject**

Argument : deleteLearningObject(learningObject:LearningObject)

Condition : Connected to database.

Developer user-type is logged in.

A current uploaded learning object exists in the database.

Flow : Developer can only delete the uploaded learning objects.

Result : Developer’s learning object is deleted from the database.

Name : **showLearningObjects**

Argument : showLearningObjects(developer:Developer)

Condition : Connected to database.

Developer user-type is logged in.

Developer’s learning objects are present in the database.

Flow : Fetches the learning objects uploaded by the developer in the database.

Result : The list of learning objects is displayed and can be viewed one by one.

### Reviewer User Module Detail

Name: **Reviewer**

Type: PHP Classes

Description: This part describes the operations done by the reviewer user-type in the system.

Attributes: (learningObject:LearningObject)

Operations:

Name : **changePassword**

Arguments : **changePassword**(username:String, password:String, newPass:String)

Condition : Connected to database and external user management system.

Reiewer user-type is logged in.

Flow : Reiewer clicks on change password and fills out the pop up.

Use change email function of external user management system.

Result : if username and password matches, reviewer password will be updated, Else, display error message to the reviewer.

Name : **changeEmail**

Arguments : **changeEmail**(username:String, password:String, new:String)

Condition : Connected to external user management system.

Reviewer is logged in.

Flow : Reviewer clicks on change email and fills out the pop up.

Use change email function of external user management system.

Result : if username and password matches, reviewer email will be updated, Else, display error message to the reviewer.

Name : **showLearningObjects**

Argument : showLearningObjects()

Condition : Connected to database.

Reviewer user-type is logged in.

Developer’s learning objects are present in the database.

Flow : Fetches the learning objects uploaded by the developer in the database.

Result : The list of learning objects with ratings and comments is displayed and

Can be viewed one by one.

Name : **rateLearningObject**

Argument : **rateLearningObject** (learningObject:LearningObject)

Condition : Connected to database.

Reviewer user-type is logged in.

Flow : Reviewer rates the Learning Object uploaded by the developers and

Comment on it.

Rating of the Learning Object is saved into the database.

Encoded comment is saved into the database.

Result : Rates and Comments are stored into the database.

### Other Operations

Name : **Login**

Arguments : login(username:String, password:String)

Condition : connected to database and external user management system.

Flow : User fills up the login form and clicks log in.

Database is queried first to check if admin is loggin in.

If true, compare passwords and return login success.

If false, use get users function of external user management system, compare username and passwords and return login success.

Result : if success, send user to respective main screen (user or admin).

Name : **History**

Arguments : (username:String, password:String)

Condition : connected to database and external user management system.

Flow : User fills up the login form and clicks log in.

Database is queried first to check if admin is loggin in.

If true, compare passwords and return login success.

If false, use get users function of external user management system, compare username and passwords and return login success.

Result : if success, send user to respective main screen (user or admin).

## Data Detailed Design

### Admin Entity Detail

**Storage Medium:** mySql Database

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data Type** | **Purpose** |
| adminID | int | Primary key for querying purposes |
| username | String | Will be the user’s personal identification |
| password | String | Will serve as secret key for accessing user’s account |
| email | String | Will serve as contact to the user |

**Table 6.2.1 – Admin Entity Detail**

### Developer Entity Detail

**Storage Medium:** mySql Database

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data Type** | **Purpose** |
| developerID | Int | Primary key for querying purposes |
| username | String | Will be the user’s personal identification |
| password | String | Will serve as secret key for accessing user’s account |
| email | String | Will serve as contact to the user |

**Table 6.2.2 – Developer Entity Detail**

### Reviewer Entity Detail

**Storage Medium:** mySql Database

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data Type** | **Purpose** |
| reviewerID | Int | Primary key for querying purposes |
| username | String | Will be the user’s personal identification |
| password | String | Will serve as secret key for accessing user’s account |
| email | String | Will serve as contact to the user |

**Table 6.2.3 – Reviewer Entity Detail**

### Learning Object Entity Detail

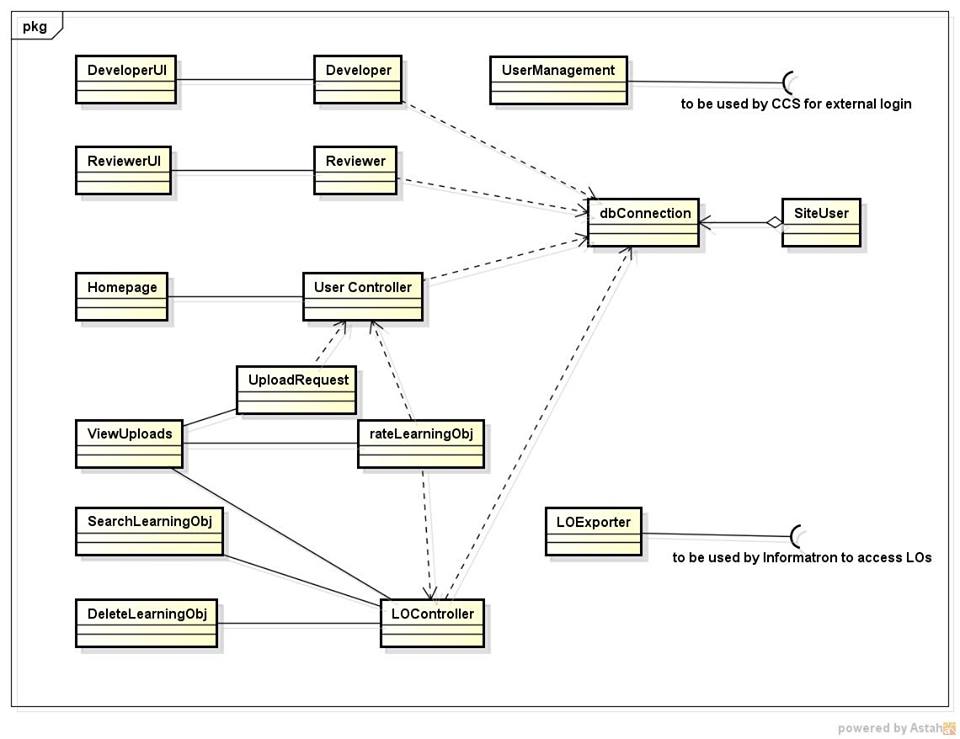
**Storage Medium:** server file system

**Wrapper Class: LO**

|  |  |  |
| --- | --- | --- |
| **Data Member** | **Data Type** | **Purpose** |
| name | String | Name of the learning object to be displayed |
| subject | String | Subject/topic touched or taught by the Learning Object |
| description | String | Description of the learning object |
| rating | Int | Shows the rating on the learning object. |
| developerID | Int | Foreign key for querying purposes |
| LO\_id | Int | Primary key for querying purposes |

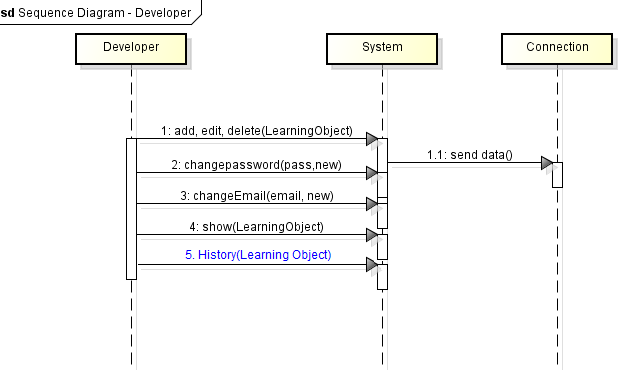
**Table 6.2.4 – Learning Object Entity Detail**

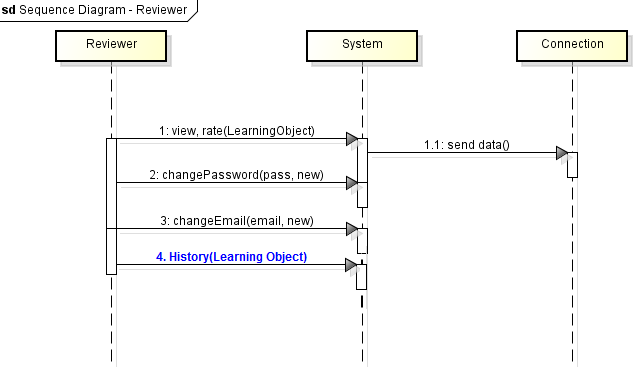
## 7.1. Class diagram



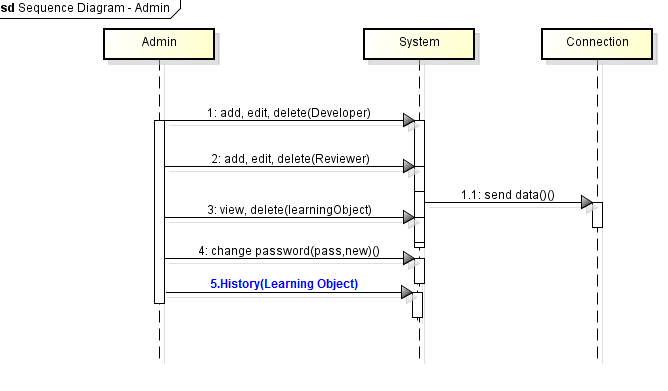
**Fig 7.1.1 – Class Diagram**

## 7.2. Use case realization (Sequence diagram / Communication diagram)



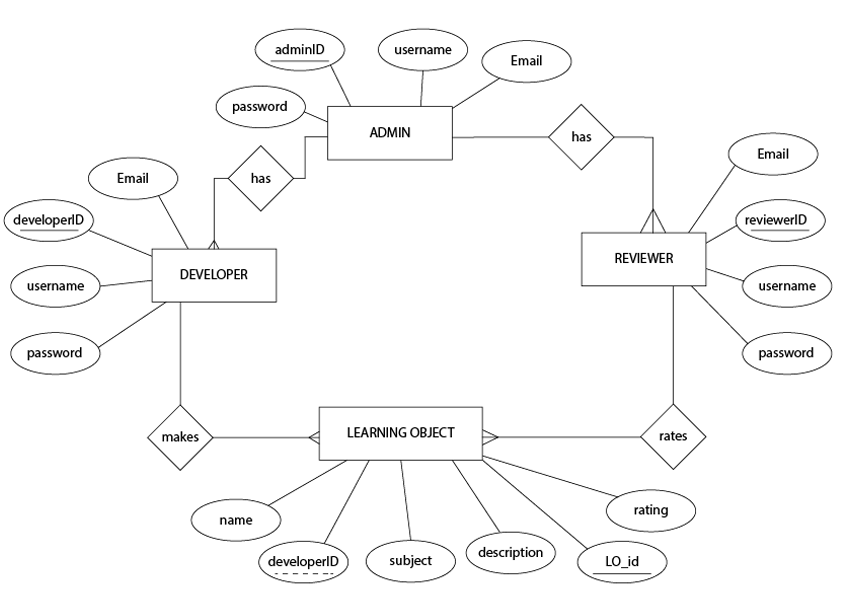
**Fig 7.2.1 – Sequence Diagram for Developer**

**Fig 7.2.2 – Sequence Diagram for Reviewer**



**Fig 7.2.3 – Sequence Diagram for Admin**

## 7.3. Entity-relationship diagram



**Fig 7.3.1 – Entity Relationship Diagram**