ASHESI UNIVERSITY COLLEGE

Designing a mimic of Ashesi Courseware

that Integrates

Peer Learning Techniques

Applied Project

B.Sc. Computer Science

Ali PF Njie

2016

Designing a mimic of Ashesi Courseware that Integrates Peer Learning Techniques

Applied Project

Applied Project submitted to the Department of Computer Science, Ashesi University College in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Science

Ali PF Njie

April 2016

DECLARATION

I hereby declare that this Applied Project is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

……

Candidate's Name:

…….....

Date: ……...……...……

I hereby declare that preparation and presentation of this Applied Project were supervised in accordance with the guidelines on supervision of Applied Project laid down by Ashesi University College.

Supervisor's Signature:

……

Supervisor's Name:

……

Date: ……

Acknowledgement

I wish to thank various people for their contribution to this project; Mr. and Mrs. Njie for their continuous and valuable support in this project; Mr. Jasseh, staff of Qcell Ltd, The Gambia, for introducing me to Course and Student Management Systems.

Special thanks should be given to Mr. Kwadwo Osafo-Maafo, my project supervisor for his professional guidance and valuable critics, and to all lecturers in the Ashesi Computer Science Department for who introduced me to the technology used to develop this project.

Abstract

This projects seeks to expand the usage of peer learning in tertiary educational institutions. It focuses on designing and a developing a simplified mimic of the Ashesi Courseware that shall make use of peer learning techniques.

This is done by granting both lecturers and students the permission to share coursework materials they feel relevant to a given course. It also includes extending its uses to allowing students to give feedback on the usefulness of materials shared in understanding class concepts and the recommendation of materials to students through the rating of uploaded materials.

The result is a sharing application that has the potential to be a vibrant information sharing platform.

Table of Contents

1. Introduction

With the global increase in the benefits of peer learning and its impact on global education, it has become essential to spend more resources on this venture so as to unlock its full potential.

"Meaningful learning emphasizes active, constructive, intentional, authentic and cooperative learning (Jonassen *et al*, 2003). Peer learning is one method to encourage meaningful learning which involves students teaching and learning from each other", M. Keppell *et al.*

This concept has been successfully integrated into most universities' teaching systems. In Ashesi, it has manifested in group assignments and projects, class presentations, peer review, etc. It acts as one of Ashesi's learning goals in a subtler form, **teamwork** (Ashesi Learning Goals).

As encouraging as it is to see such a group-oriented learning method in use in this noble institution, its application is yet to be integrated fully into the Ashesi learning experience. The fact remains that some of the technology in used in this institution to streamline the interaction and communication between lecturers and students have failed to use this concept.

Perhaps the most used of this tools are **Ashesi Courseware.** The primary tool used by faculty to share course materials, delegate assignments, ease assignment submissions, and in some rare cases conduct online quizzes, have failed to utilize or deliberately ignored peer learning concepts.

Uploading/Sharing course material in courseware is an exclusive right of faculty. Also, it lacks an interaction platform for students to discuss course materials, and its rather complex design has made a lot of its functionalities unknown to student or too exasperating to explore.

These limitations have limited courseware's purpose to an assignment submission forum, and closed discussion on the vibrant knowledge sharing platform it ought to be.

In order to integrate peer learning fully into the Ashesi learning experience, there is the need to redesign courseware in the image of what Ashesi stands for, one of which is teamwork. Courseware needs to evolve into a vibrant knowledge sharing platform for students, under the supervision of faculty.

Courseware+ is an education platform that will incorporate peer learning techniques into a more simplified Ashesi courseware mimic. It is a tool that shall remove the exclusivity of coursework material uploading of lecturers and allow all participants in a given class to share materials they think relevant to a course.

In addition to giving students the permission to share materials they think will help their peers understand class concepts better, it will also allow all participants in a given class the freedom to provide feedbacks on shared course materials. This will enable other participants to properly gauge the effectiveness of shared materials in explaining relevant concepts.

Peer learning can help "the development of learning outcomes related to collaboration, teamwork, and becoming a member of a learning community", D. Boud *et al*. However, if under supervised the results could be disastrous. As such, Courseware+ will provide for faculty members an interface to monitor student activities, understanding of shared materials, and tracking students' fluency in covered concepts in class.

It is hoped that Courseware+ will spur a discussion on the potential of a vibrant knowledge sharing platform that can integrate peer learning concepts into this virtual learning environment. A platform with a minimalistic design, whose primary focus shall be to facilitate engagement among participants in a given course.

2. Requirements Specification

The purpose of this application is to provide a medium through which lecturers and students can get share relevant course materials. It also features a fluency component that shall allow lecturers to monitor student understanding in the different lessons of the course, and a schedule generator which shall make use of student understanding to generate an efficient study time-table to address students concerns.

The requirements for the application can be divided into three broad categories:

- Content Sharing
- Student Fluency
- Study Schedule Generator

The content sharing functionality enables both tutors and students to share materials with other participators in the course, and to also give feedback on those materials for future users. Materials shared are organized based on the course they were shared on and the section of the course to which they were shared.

The student fluency requirement shall provide students the ability to rate their understanding of the different sections taught in class. These ratings are presented to the lecturer so as to give him/her an overview of how well students are faring in his/her class.

As the name implies, the study schedule generator shall provide the system the ability to use the student fluency ratings of students to generate a study schedule to address student's problems in the different courses the student is partaking in. This feature is exclusive to students.

2.1. Purpose

The purpose of this chapter is to present a detailed description of the **Courseware+** platform. It will explain the purpose and features of the system, its parameters and goals, the interfaces of the system, the constraints under which it must operate, and how the system will react to external stimuli.

This chapter will provide an overview of the software product. It shall describe the audience and how they see the product and its functionality, while also giving the developer a breakdown of the system's functionalities.

2.2. Scope of Project

The first iteration of Courseware+ will focus on the content sharing potential of the system. As such it will be designed to maximize the student's productivity by providing tools to assist in gauging material relevance and usefulness to other students, and allowing lecturers to monitor student understanding of concepts covered in class.

2.3. Overview

This section contains the requirements for Courseware+. The process involves first, the listing of general user requirements, this is followed by the functional requirement, generated from the general user requirements, with detailed descriptions and diagrams of each use case.

2.4. User Requirements

The Courseware+ system has two active users:

I. Students

II. Faculty/Tutors

Students can interact with content sharing module of the system. Lecturers, in addition to the interacting with the content sharing module can also access the fluency module of the application.

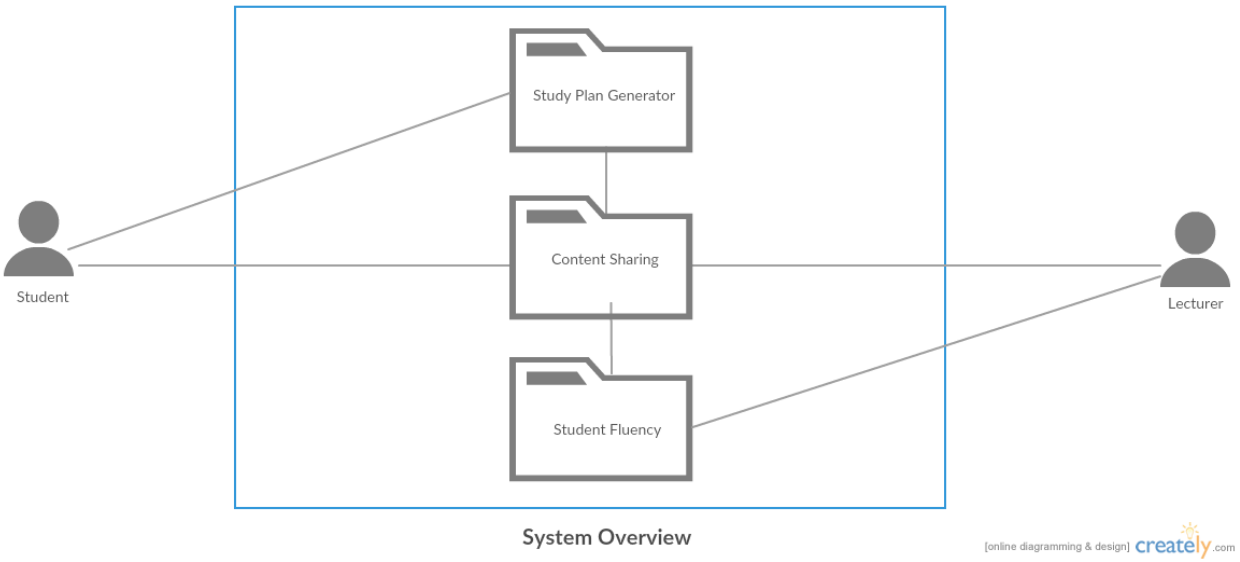


Figure 2.1: System Overview

Users of Courseware+ should possess the ability to do the following:

- All users should be able to share course materials with their peers and/or lecturers doing the same course.
- All users should be able to give feedbacks and/or see reviews of shared materials in a given course.
- Students should be given the permission to rate their fluency in concepts covered in class.
- Lecturers should have a structured way of organizing course materials into clutters.
- Lecturers should have the permission to see the fluency of students in concepts covered in class.
- The right to create a course should be granted exclusively to lecturers.

2.5. Use Case Scenarios

This section outlines the specific use cases for the active users in detail.

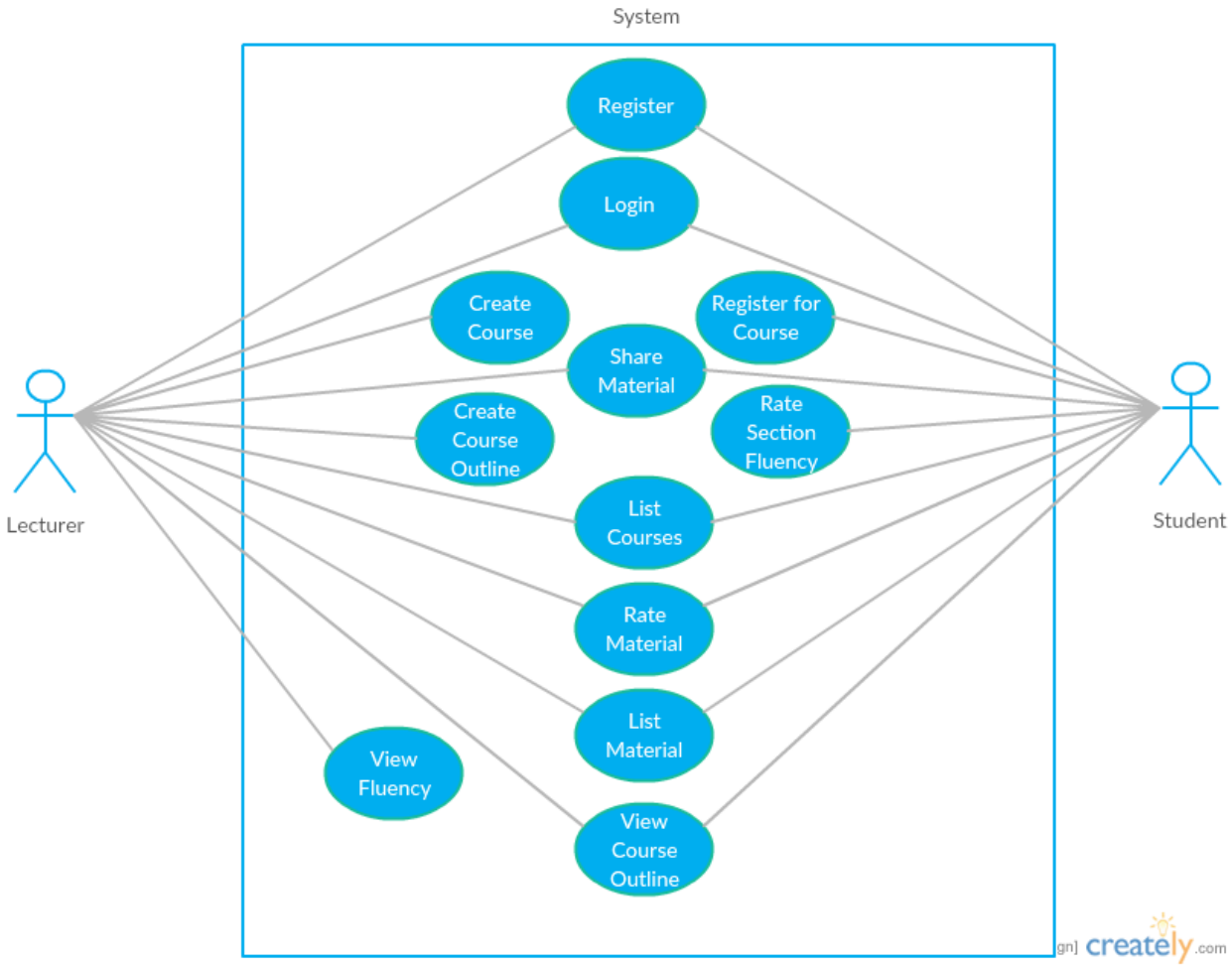


Figure 2.2: Use Case Scenarios

2.6. Functional Requirements

Use Case: Register

Brief Description

This use case describes the process by which the users create an account with Courseware+. It also sets up access permission for various categories of users.

Actors

- Lecturer
- Student

Event Flow

i. The use case starts when the user starts the application.
ii. The system will display the register account screen
iii. The user enters a full name and an email address, selects a user category, and enters a password and confirms the password.
iv. The system will verify the information given.
v. The system will then set access permissions for user.
vi. The system will display the Courseware+ dashboard on the screen.

Use Case: Login

Brief Description

This use case describes the process by which the users sign in to Courseware+.

Actors

- Lecturer
- Student

Event Flow

i. The use case starts when the user starts the application.
ii. The system will display the login account screen
iii. The user enters an email address and a password.
iv. The system will verify the information given.

v. The system will display the Courseware+ dashboard on the screen.

Use Case: Create Course

Brief Description

This use case describes the process by which a user can create a course with Courseware+.

Actors

- Lecturer

Event Flow

i. The use case starts when the user clicks on the create course link on the dashboard.

ii. The system will display a create course screen.

iii. The user enters a title for the course and its description.

iv. The system will verify the information given.

v. The system will display the Courseware+ dashboard on the screen with the newly added course listed.

Use Case: Create Course Plan

Brief Description

This use case describes the process by which a user can create a plan of action for an already created course.

Actors

- Lecturer

Event Flow

i. The use case starts when the user clicks on the create course plan on a course's course board.
ii. The system will display a create course plan screen.
iii. The user enters the five broad section which the course will structured around.
iv. The system will verify the information given.
v. The system will display the course's course board on the screen with course plan displaying

Use Case: Share Material

Brief Description

This use case describes the process by which a user can share material for a given course.

Actors

- Lecturers
- Students

Event Flow

i. The use case starts when the user clicks on the share course material on the course board of a particular course.
ii. The system will display a share course form.
iii. The user selects the section of the course to which the material should be added to.
iv. The user selects the file he/she wants to share.
v. The system verifies the information given.
vi. The system displays the course board with the newly uploaded material.

Use Case: Rate Material

Brief Description

This use case describes the process by which a user can rate a material on Courseware+.

Actors

- Lecturer
- Students

Event Flow

i. The use case starts when the user clicks on the course material.
ii. The system displays an option to rate the material.
iii. The user selects from the available list of ratings.
iv. The system verifies the information provided.
v. The system displays the course board of the course.

Use Case: Display Material Rating

Brief Description

This use case describes the process by which a user can view the average rating of a course material.

Actors

- Lecturers
- Students

Event Flow

i. The use case starts when a user clicks on a link to view course's course board.

ii. The system displays the course material and its average rating on its right side.

Use Case: Rate Fluency

Brief Description

This use case describes the process by which a user can rate his/her fluency on a section of the course.

Actors

- Students

Event Flow

i. This use case starts when the user clicks on the dropdown link attached to a section header.

ii. The dropdown displays a list of possible ratings.

iii. The user selects one of the possible ratings.

iv. The system verifies the information.

v. The system loads the course's course board with the newly rated section's rating attached to the section header.

Use Case: View Course Plan

Brief Description

This use case describes the process by which a user can view the course plan of a given course.

Actors

- Lecturers
- Students

Event Flow

i. This use case starts when the user clicks on the link to view a course's course board.

ii. The system displays the course's course board with the course plan listed in the view.

Use Case: List Course Materials

Brief Description

This use case describes the process by which a user can view the shared materials on a given course.

Actors

- Lecturers
- Students

Event Flow

i. This use case starts when the user clicks on the link to view a course's course board.

ii. The system displays the course's board with the shared course materials listed in the section to which they were added to.

Use Case: List Courses

Brief Description

This use case describes the process by which a user can see a list of the courses that he/she is associated with.

Actors

- Lecturers
- Students

Event Flow

i. This use case is initiated when the users signs in to the system.
ii. The system displays to him the dashboard which contains a list of all courses associated with him.

Use Case: Register for Course

Brief Description

This use case describes the process by which a user can register for a course.

Actors

- Students

Event Flow

i. This use case is initiated when the user clicks on the register for course link on the navigation bar.
ii. The system presents to the user a register for course form.
iii. The user enters the token for the course.
iv. The system verifies the information.

v. The system displays the dashboard with newly registered course displaying.

Use Case: Display Class Fluency

Brief Description

This use case describes the process by which a user can view the fluency of a class in the different sections of the course.

Actors

- Lecturers

Event Flow

i. This use case is initialized when the user clicks on the fluency tab of the dashboard.
ii. The system lists all the courses the user is associated with.
iii. The user selects a course.
iv. The system displays the percentage of students who selected the available ratings for the different sections.

Use Case: Display Individual Fluency

Brief Description

This use case describes the process by which a user can view the fluency of student in the different sections of the course.

Actors

- Lecturers

Event Flow

i. This use case is initialized when the user clicks on the fluency tab of the dashboard.

ii. The system lists all the courses the user is associated with.

iii. The user selects a course.

iv. The system presents two tabs, one with individual written on it.

v. The user clicks on the individual tab.

vi. The system presents to the user the fluency of the student on the different sections in the course.

3. Architecture

This chapter will provide a high level overview of the system, and explain the architecture of Courseware+. It shall explain how a lecturer will be able to create and monitor classes, and will include the underlying architecture of the system.

This chapter provides a sophisticated description of the goals of the architecture, the use cases supported by the system and the architectural styles and components that have been carefully chosen to best achieve the use cases.

3.1. Purpose

The chapter provides a comprehensive architectural overview of Courseware+.

It presents diverse architectural views to illustrate aspects of the system. Its purpose is to capture and convey the significant judgements which have been made on the system.

3.2. Scope

This chapter describes the aspects of Courseware+ design that are considered to be architecturally significant; that is, those features and behaviors that most central for guiding the peer learning process and understanding the system as a whole.

3.3. Architectural Representation

3.3.1. Use Case View

Overview

This section describes the set of scenarios and /or use cases that represent some significant, central functionality of the system. It describes the actors and use cases for the system. It presents the needs of the user.

Below is a list of use-cases that represent major functionality of the system.

- User Register
- User Login
- Create Course
- Register for Course
- Create Course Plan
- Share Material
- Rate Material
- Rate Section
- View Class Fluency
- View Individual Fluency

Refer to use-case diagram in requirements specification chapter.

Actors

A web user of Courseware+ could be one of three types:

- **Lecturer** has enhanced privileges to monitor general class fluency and individual student understanding of covered sections in class.
- **Student** can share materials and review material uploaded/shared by peers.
- **System** is third type of actor. It handles all physical and logical process of the software.

Use-Case Realization

The diagram below shows the organization of the use-cases based on the user's category, and how this use-cases interact with other components of the system.

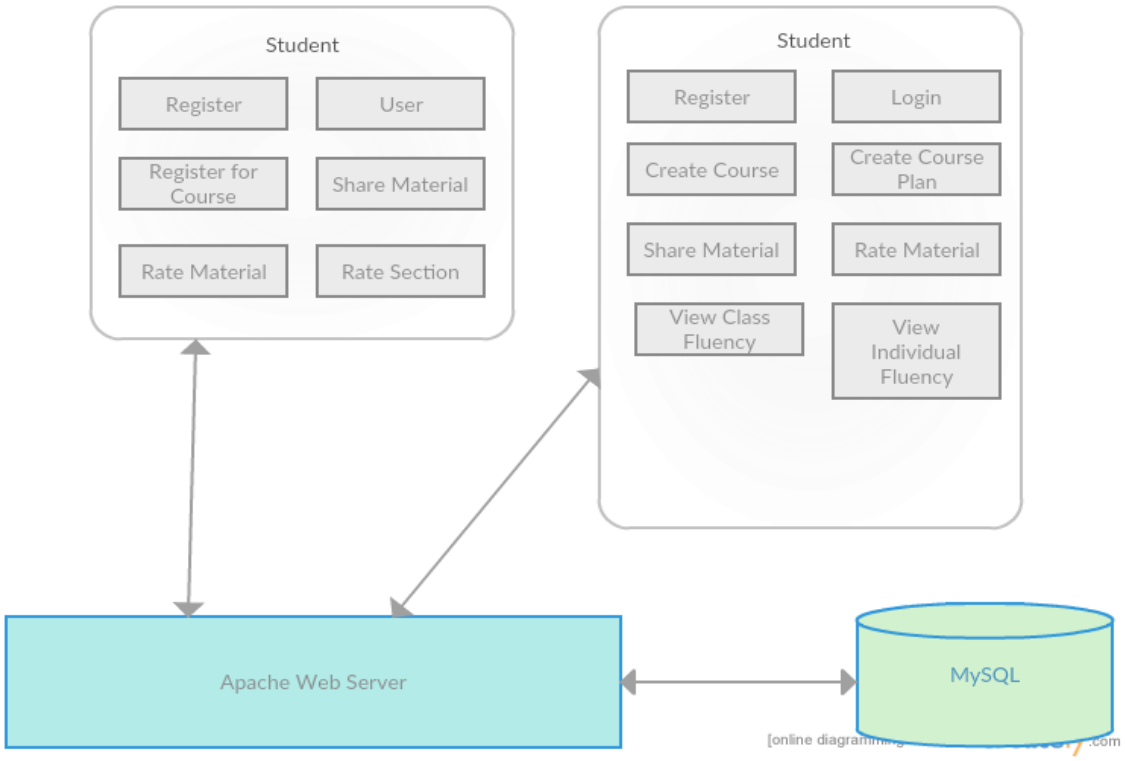


Figure 3.1: Use Case Realization

3.3.2. Logical View

Overview

Courseware+ is generally divided into layers based on the N-tier architecture.

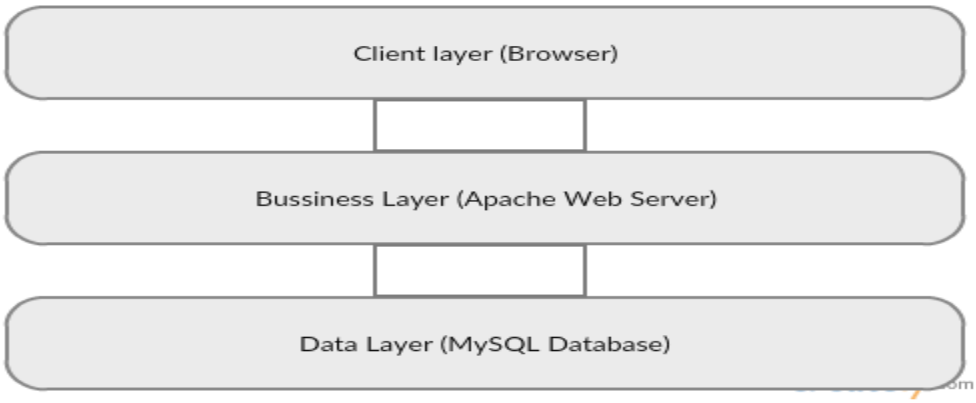


Figure 3.2: Layer Architecture

This layering approach of Courseware+ application is based on a responsibility layering scheme that associates each of the layers with a particular responsibility. This strategy makes it possible to isolate the various responsibilities of the system from one another to optimize development and maintenance.

Class Diagram

This section describes the attributes and operations that define each class in the system, its relationship with the other classes, and the constraints they impose on the system.

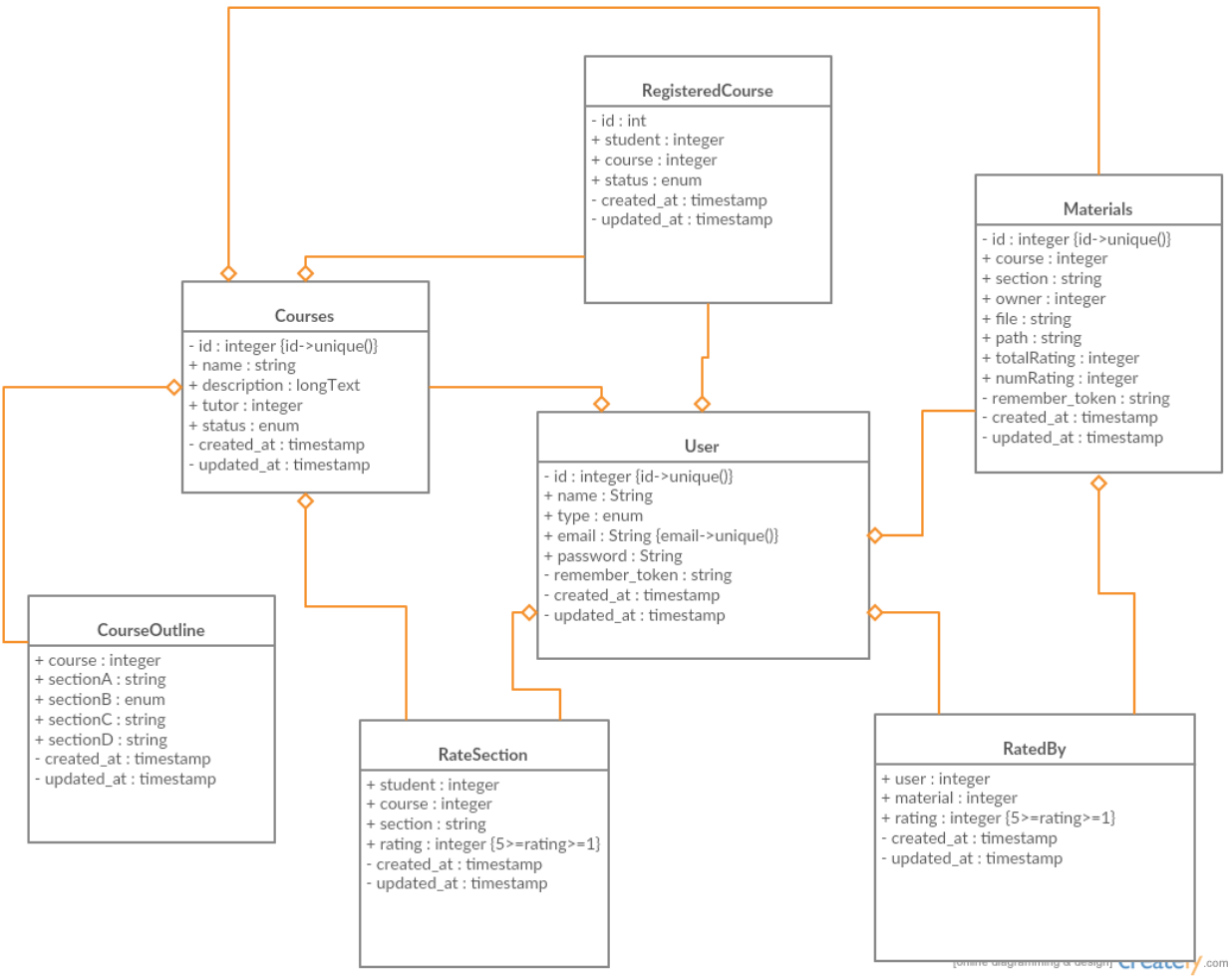


Figure 3.3: Class Diagram

State Machine Diagram

The behavior of the objects of the classes above at different states are as follows:

i. User

The User object doesn't exist in many different states. From creating a user object, all attributes are initialized except for the *remember_token*, and the *updated_at* attributes.

At initialization, that is upon registration, the user sets the value of all his/her attributes with exception of the two mentioned above. Every time the user makes an attempt to login, the

view presents a “remember me” option. If the user checks the option, a string is generated and stored in the *remember_token* field. Because an attribute of the user object has been altered, the value of the *updated_at* attribute is changed to timestamp the event took place.

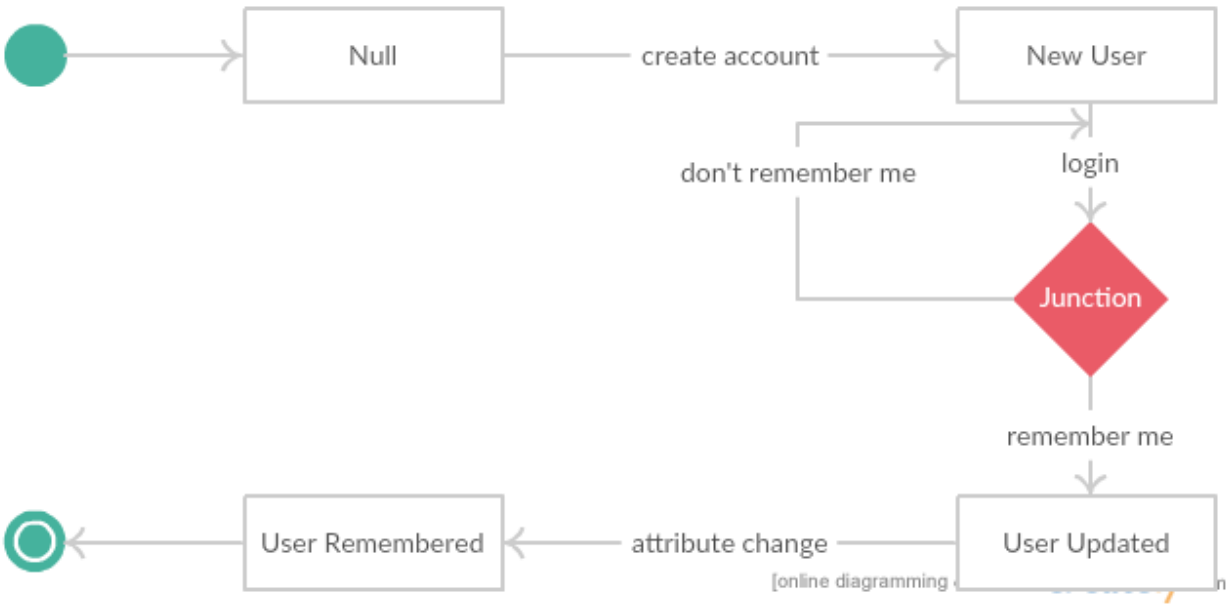


Figure 3.4: State Machine Diagram of User object

ii. Courses

Upon creation/initialization, that is when a user creates a course, all its attributes with the exception of *updated_at* are initialized. However, the status attribute is set to a default value of “inactive”. When the user has already created a plan for it, he/she can enable it. However, if no plan exists for that course, the course will stay inactive.

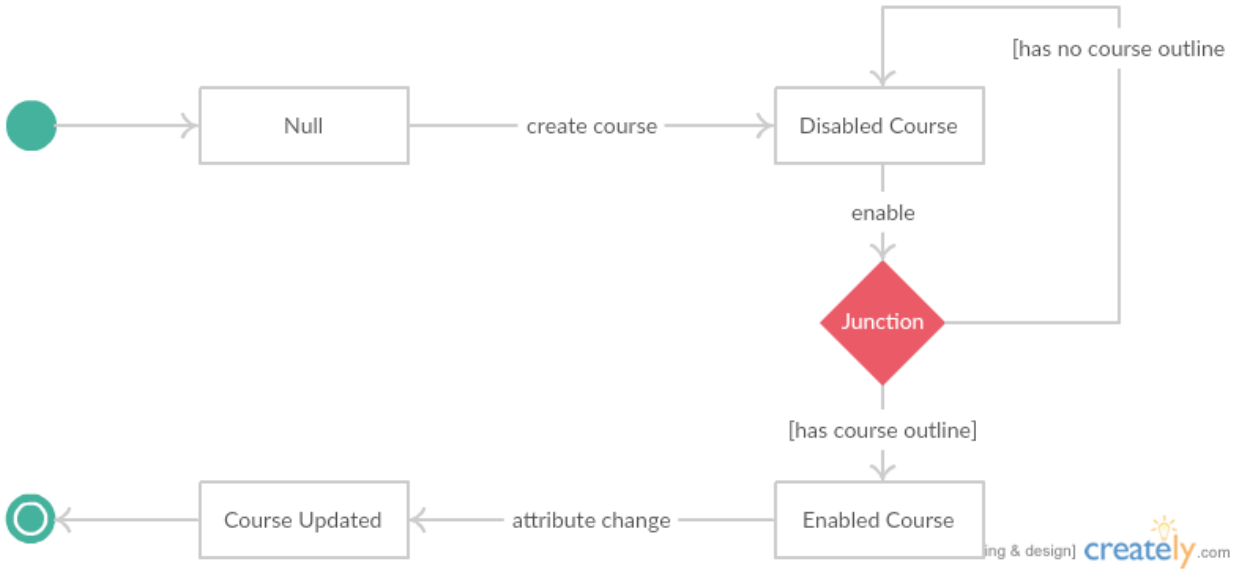


Figure 3.5: State Machine Diagram of Course object

iii. Material

The material object is perhaps the object that goes through the most state changes. The user adds a material. Its name, path, owner and the course to which it belongs are recorded. The total ratings for that material and its number of rating are initialized to zero (0). Whenever, a user rates the material object, the rating is added to total rating attribute and number of rating attribute is increased by a value of one (1).

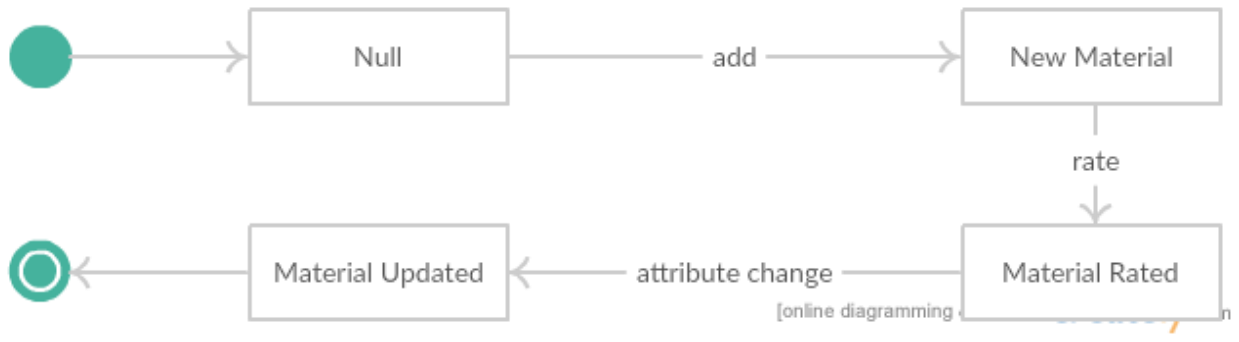


Figure 3.6: State Machine Diagram of Material

For the rest of the classes, their objects undergo little or no state changes.

3.3.3. Process View

Leveraging on the separated nature of HTTP request/response and the ability of relational database, Courseware+ will handle multiple users simultaneously.

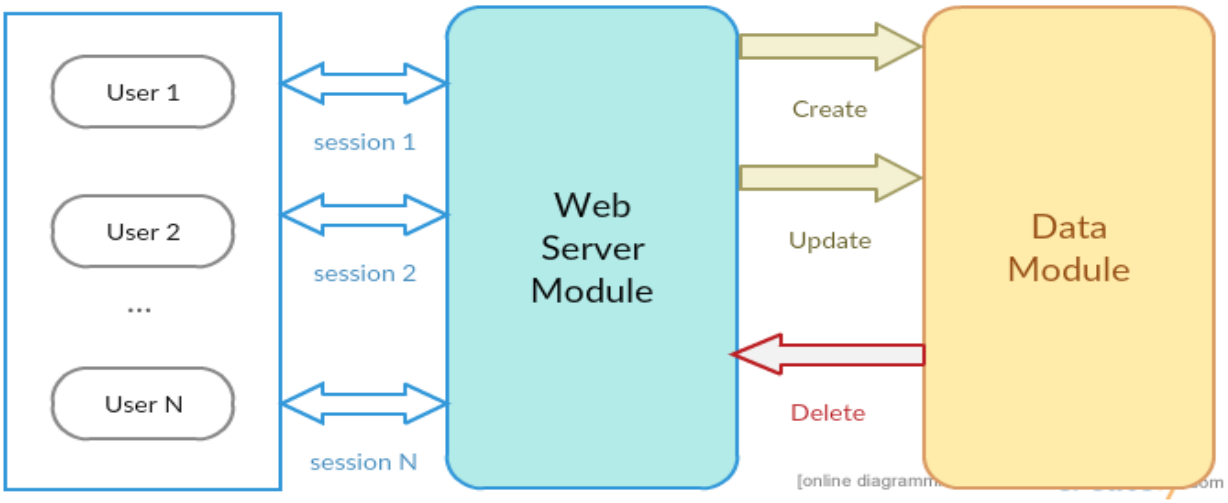


Figure 3.7: Session Handling Overview

Activity Diagram

This section describes the workflow of crucial activities of the system.

i. User Registration and Login

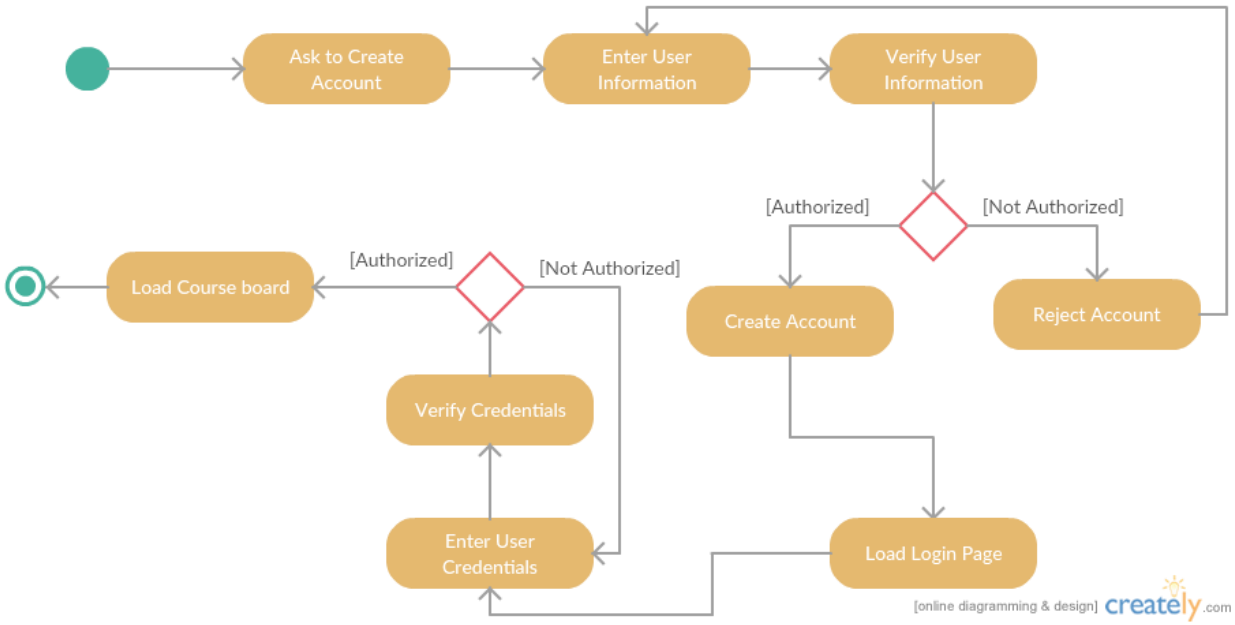


Figure 3.8: User Registration and Login Activity Diagram

ii. Course Creation and Enabling

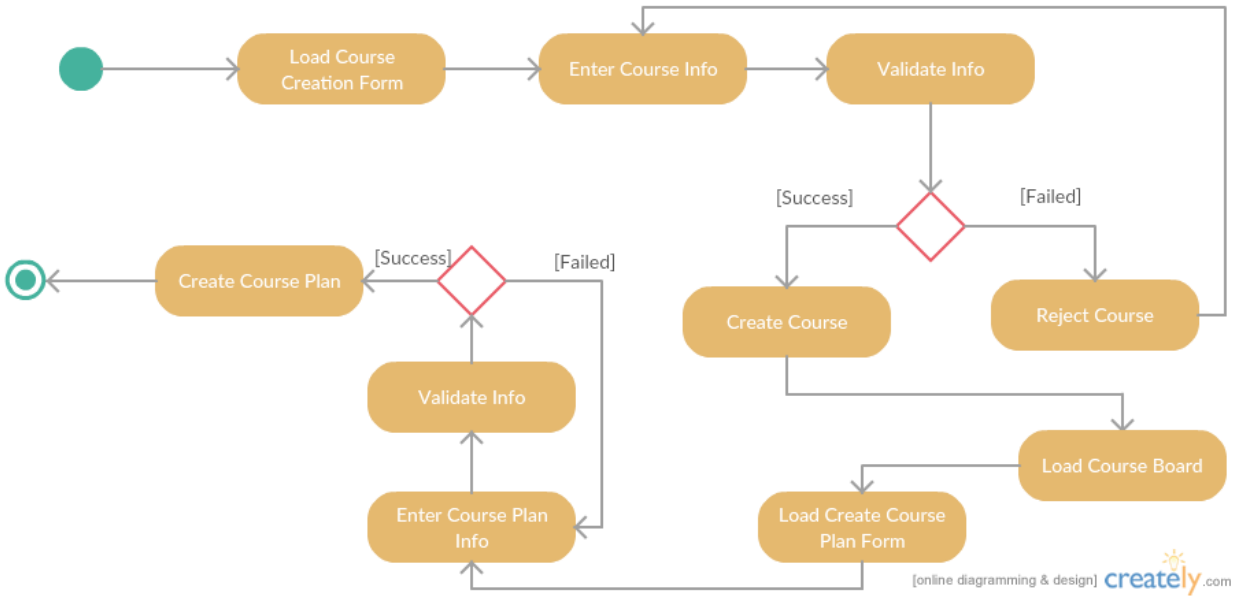


Figure 3.9: Course Creation and Enabling Activity Diagram

iii. Material Sharing and Rating

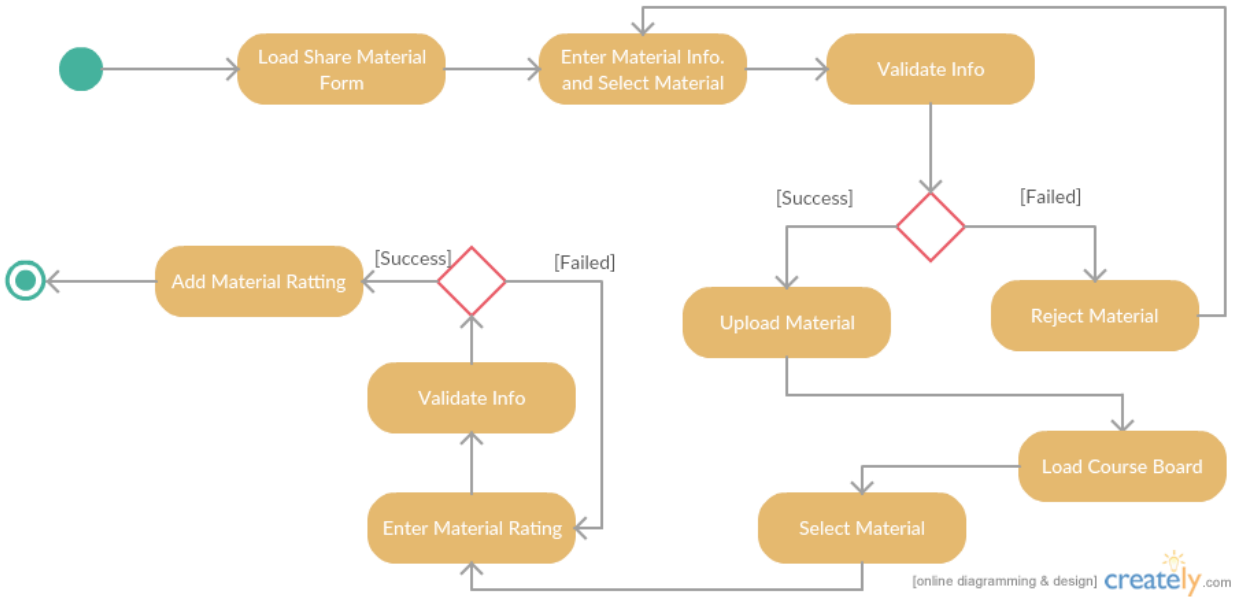


Figure 3.10: Material Sharing and Rating Activity Diagram

Sequence Diagram

This section describes the interaction between objects in the sequential order that those interactions occur

The sequential order of the interaction is the same for all use cases and as such a general diagram will be used to show how this interaction takes place.

All use cases/functionalities are initiated through routing, which is clicking or sending a link (synchronous HTTP request to the server). The link/route is received by the routes file which contains a list of all operational routes that the system makes use of. The route file checks whether the HTTP request that was made is part of the list. If the requests cannot be found, and error is returned to the presentation layer, the browser. If the request is found in

the list, the routes file sends the parameters that was sent with the request to the appropriate controller specifying the function to call.

The controller initializes the appropriate models, which runs a query on the database and pass the data obtained to the controller.

The controller loads the appropriate view to send back to the presentation layer together with an array of the data requested.

The presentation layer presents the information/data to the user.

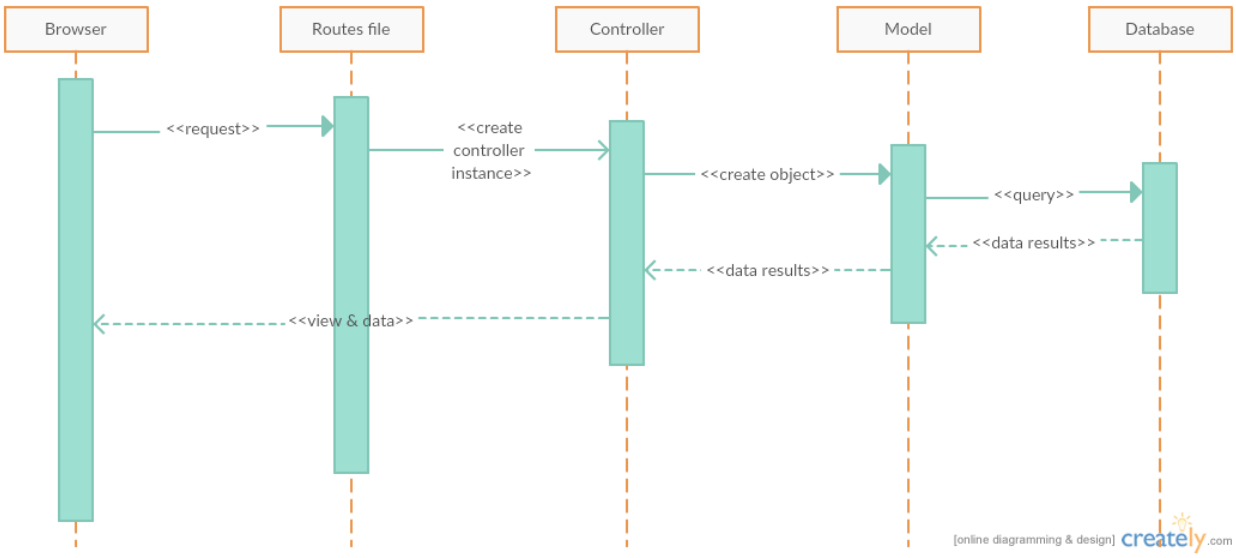


Figure 3.8: Sequence Diagram

3.4. Architectural Goals and Constraints

Server Side

Courseware+ will be hosted on an Apache web server running on a Linux platform, and connecting to a MySQL database server. All communications with client has to comply with public HTTPS, TCP/IP communication protocol standards.

Client Side

Clients will be able to access Courseware+ only online. Clients are required to use a modern web browser, preferably Mozilla Firefox 10 or later, Internet Explorer 9, and/or latest versions of Google Chrome and Safari.

Performance

It is expected that the system should respond to any request well under standard database and web server script timeouts. Note that the performance of the system is highly dependent on the available hardware, the strength of the network, and internet connection capabilities.

In addition, the time it takes to upload or download files from the system depends on data size which in turn depends on user input.

4. Implementation

This section, as the name implies describes the implementation process of Courseware+. It shall include description of the platform on which it is build; the programming language used, and framework that was used.

This section will also describe the user interface of the system. It shall also include the organization of project folders and the description of the contents of the relevant folders.

4.1. Platform

The platform chosen for the development of Courseware+ is the web platform. Alternatives included desktop and mobile applications, however the former was preferred. Web platform was preferred because intended users of Courseware+ are already familiar with using Ashesi Courseware which is a web application.

Also web applications are accessible on desktops, laptops, tablet, and phones. Thus, this make its accessibility universal.

4.2. Language(s)

The language that could be used to implement Courseware+ are numerous. PHP was chosen to build this project. The reason is mostly due to familiarity with the language and a desire to explore it more.

4.3. Frameworks(s)

To build a robust application like Courseware+, one needs a framework to build upon. This is because frameworks make the development process faster and easier. Also, it provides a better structure for organizing code.

In addition to that, it provides a common ground between current developer(s) and developers who might want to work on it in the future.

The PHP framework that was chosen to build Courseware+ is **Laravel**. The main reason is that it is the most widely used PHP framework in the world. This means that it has a bigger developer community to provide support.

4.4. File & Folder Structure

The application is structured into nine folders, out of which most of the interaction occurs in only four folders which are: app, database, public, and resources.

Figure 4.1: Folder Structure

The app folder contains the model classes. It also contains a **Http** folder, which contains within in the **routes** file and a **Controller** folder which contains all controllers.

The database folder contains a **migration** folder which holds migration files (used to create database tables).

The public folder holds a **css** and a **js** folder which contains the stylesheets and JavaScript files.

The resources folder is where the view folder is located, which is where all work related to the view are done.

As explained in the Architecture chapter. The work flow starts with a HTTP request which is captured by the routes file in the app/Http/ folder. This files then initializes the appropriate Controller located in the app/Http/Controllers folder giving it the appropriate function call to handle the request.

The Controller creates an object of the appropriate model which runs the query on the database and returns to the Controller the data result. The data result is processed by the controller and sent to the view which is presented to the user.

4.5. User Registration & Login

Laravel has a command than ones run to create a default user registration and login functionality. The user registration has four fields which are **name**, **email**, **password**, and **confirm password**.

The user registration field was tweaked to include an account type field, which could take only two possible values: tutor, or student.

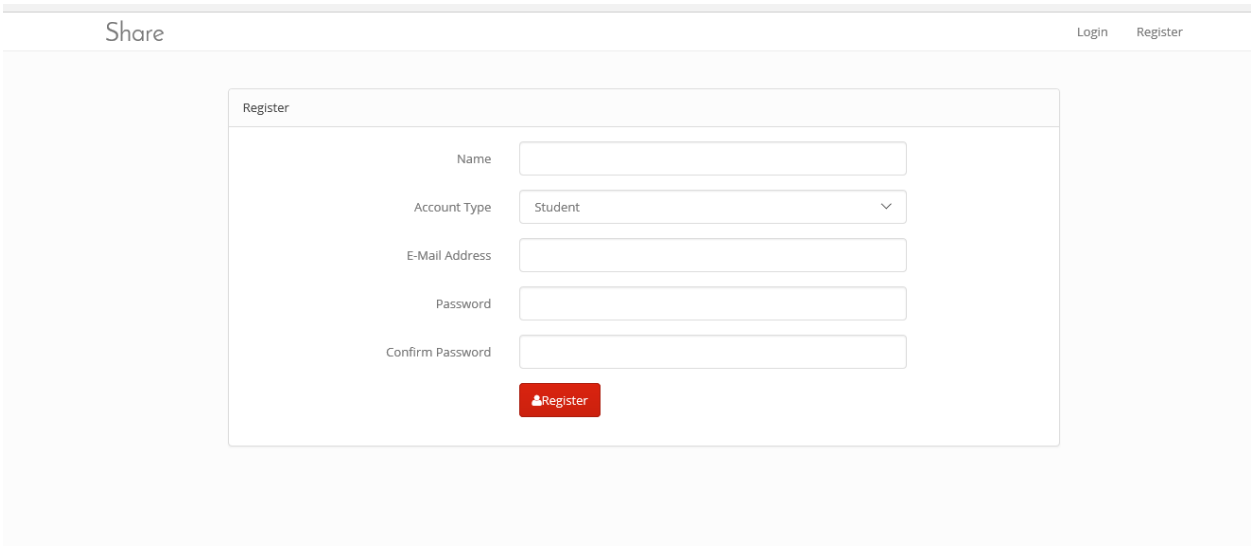


Figure 4.2: User Registration

The auto generated login view was tweaked to include validation.

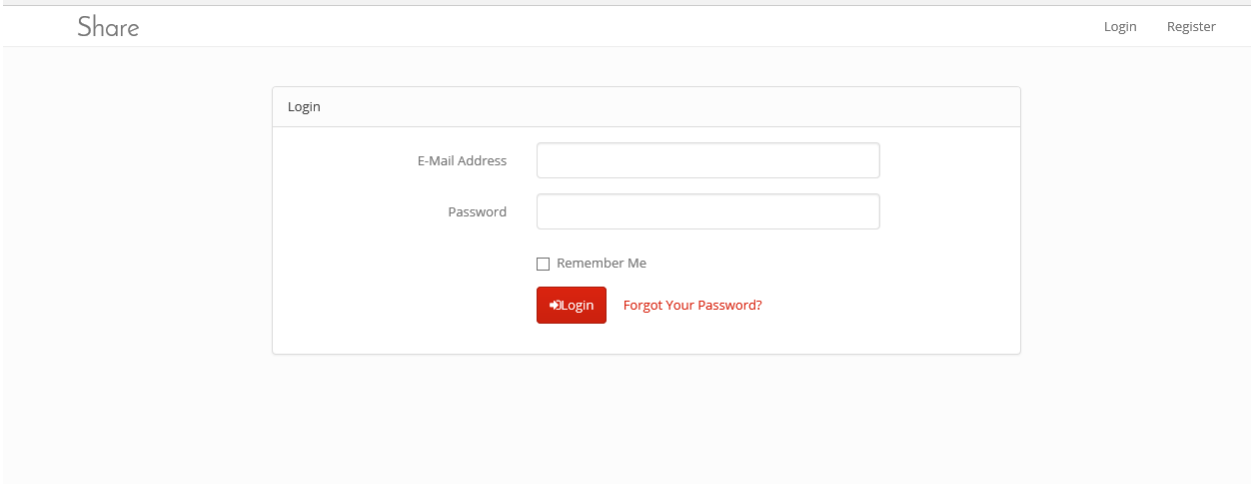
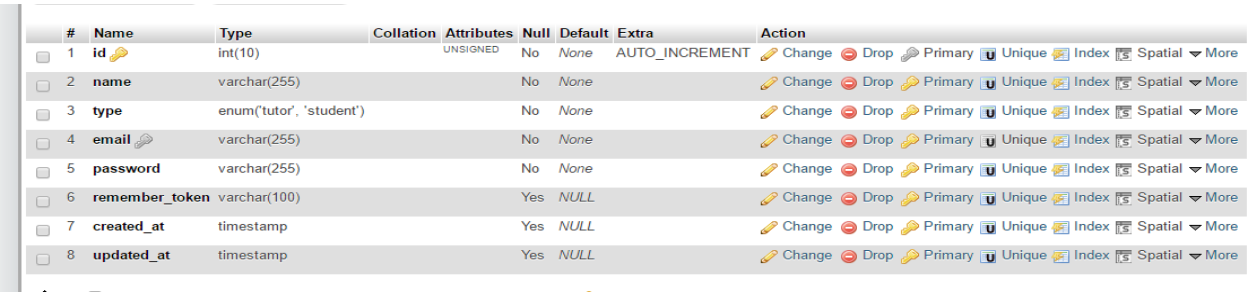


Figure 4.3: User Login

Laravel installation comes with a user model and a migration file for creating a user table. The account type attribute was added to this file before running the migrate prompt on the terminal.



#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	id	int(10)		UNSIGNED	No	None	AUTO_INCREMENT	Change Drop Primary Unique Index Spatial More
2	name	varchar(255)			No	None		Change Drop Primary Unique Index Spatial More
3	type	enum('tutor', 'student')			No	None		Change Drop Primary Unique Index Spatial More
4	email	varchar(255)			No	None		Change Drop Primary Unique Index Spatial More
5	password	varchar(255)			No	None		Change Drop Primary Unique Index Spatial More
6	remember_token	varchar(100)			Yes	NULL		Change Drop Primary Unique Index Spatial More
7	created_at	timestamp			Yes	NULL		Change Drop Primary Unique Index Spatial More
8	updated_at	timestamp			Yes	NULL		Change Drop Primary Unique Index Spatial More

Figure 4.4: User Database Table

4.6. Course Creation & Course Outline Creation

The welcome screen upon login was also changed to list courses that the user is involved with. The navigation included a create course for lecturers and a register for course for students.

The home.blade.php file in resources/view/ is used as the general template for the dashboard. Depending on the request that was send, a file in the resources/view/tutor/ is loaded which extends the home.blade.php (read more about template on the Laravel documentation site).

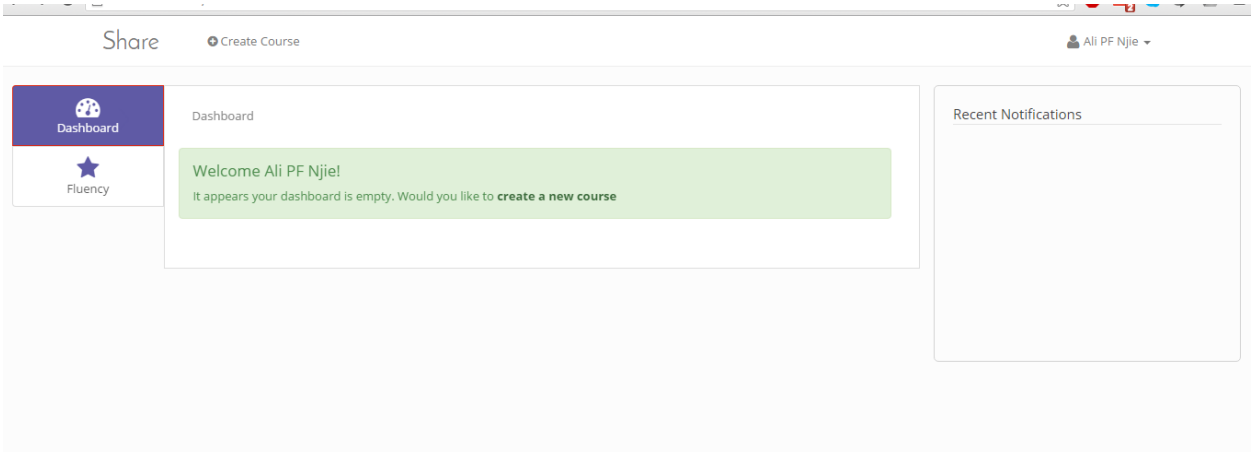
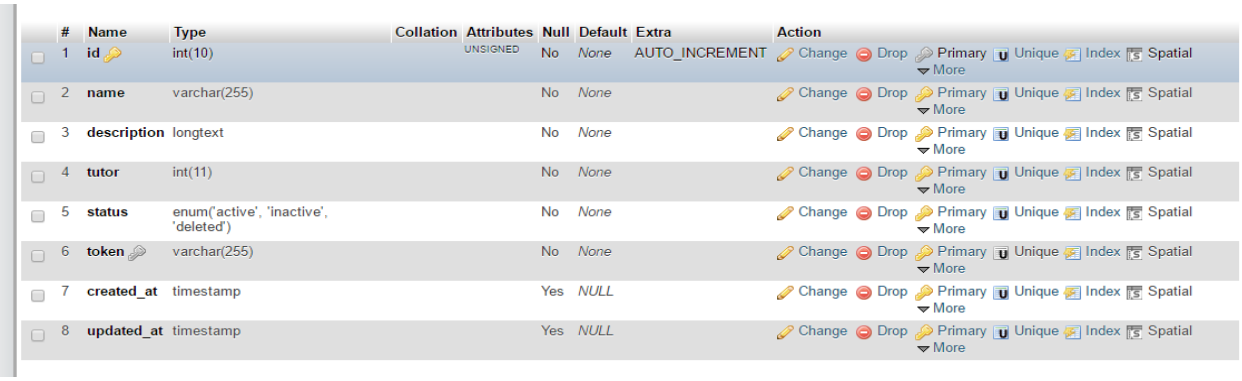


Figure 4.5: Empty Dashboard

All functionalities that are associated with courses are handled by the **CourseController** in the Controller folder.

A create course migration is located in the **migration** folder which creates the courses table in the database. The structure of the courses table is as follows.



#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	id	int(10)		UNSIGNED	No	None	AUTO_INCREMENT	Change Drop Primary Unique Index Spatial More
2	name	varchar(255)			No	None		Change Drop Primary Unique Index Spatial More
3	description	longtext			No	None		Change Drop Primary Unique Index Spatial More
4	tutor	int(11)			No	None		Change Drop Primary Unique Index Spatial More
5	status	enum('active', 'inactive', 'deleted')			No	None		Change Drop Primary Unique Index Spatial More
6	token	varchar(255)			No	None		Change Drop Primary Unique Index Spatial More
7	created_at	timestamp			Yes	NULL		Change Drop Primary Unique Index Spatial More
8	updated_at	timestamp			Yes	NULL		Change Drop Primary Unique Index Spatial More

Figure 4.6: Course Database Table

Upon clicking a create course link, the CourseController directs the user to a create course form which validates the form inputs before sending the data to the CourseController which then adds the course to the database and loads dashboard.

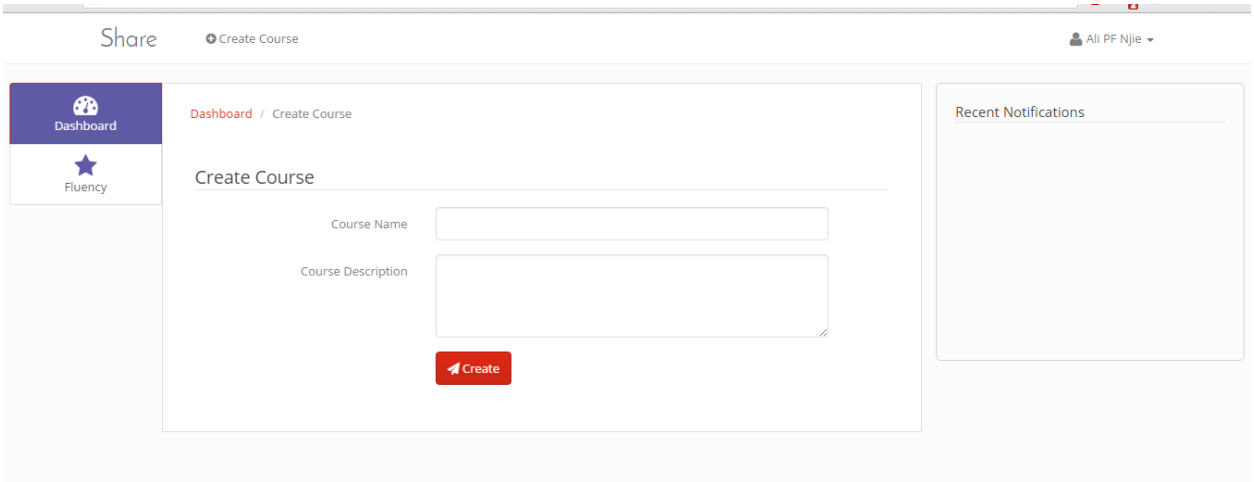


Figure 4.7: Create Course Form

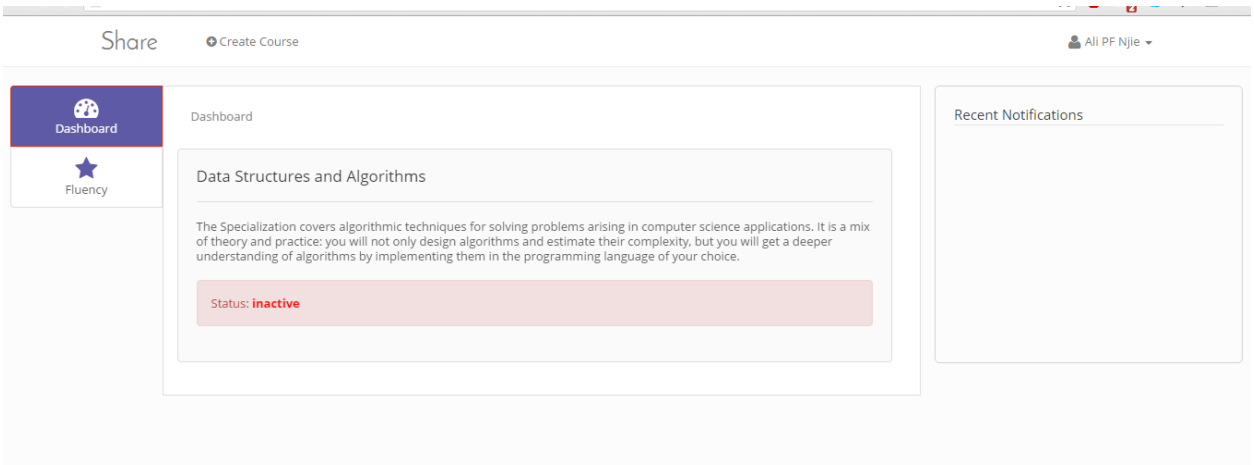


Figure 4.8: Dashboard

A newly created course is marked inactive. To make it active, a lecturer must first create an outline for the course and then enable it. Each course has a unique token assigned to it which students must use to register for the course.

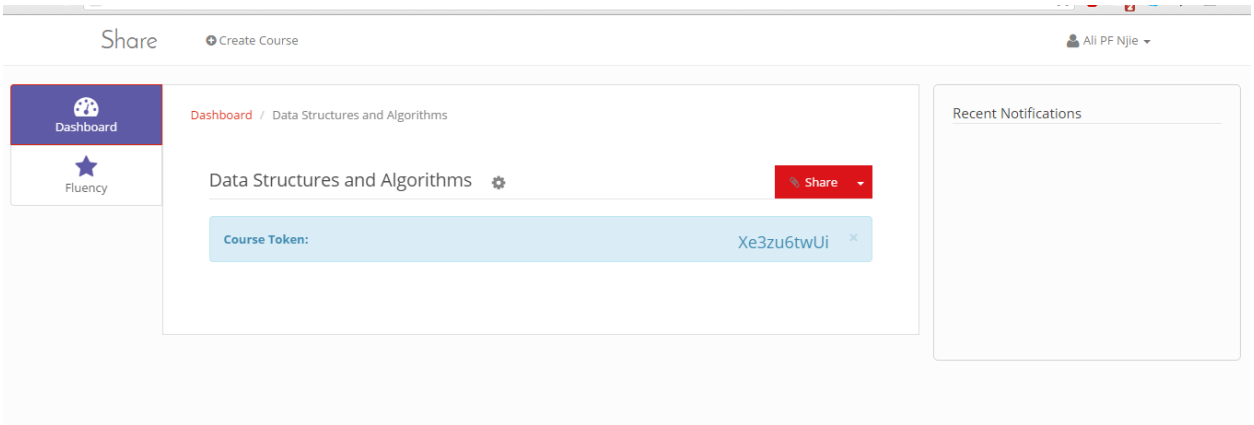


Figure 4.9: Courseboard

Each course should be divided into five broader section which make up the outline. This is used to better organize the shared materials. After creating a plan for the course and then

activating it. The courseboard will include a listing of the different sections. Participants can then share materials with each other.

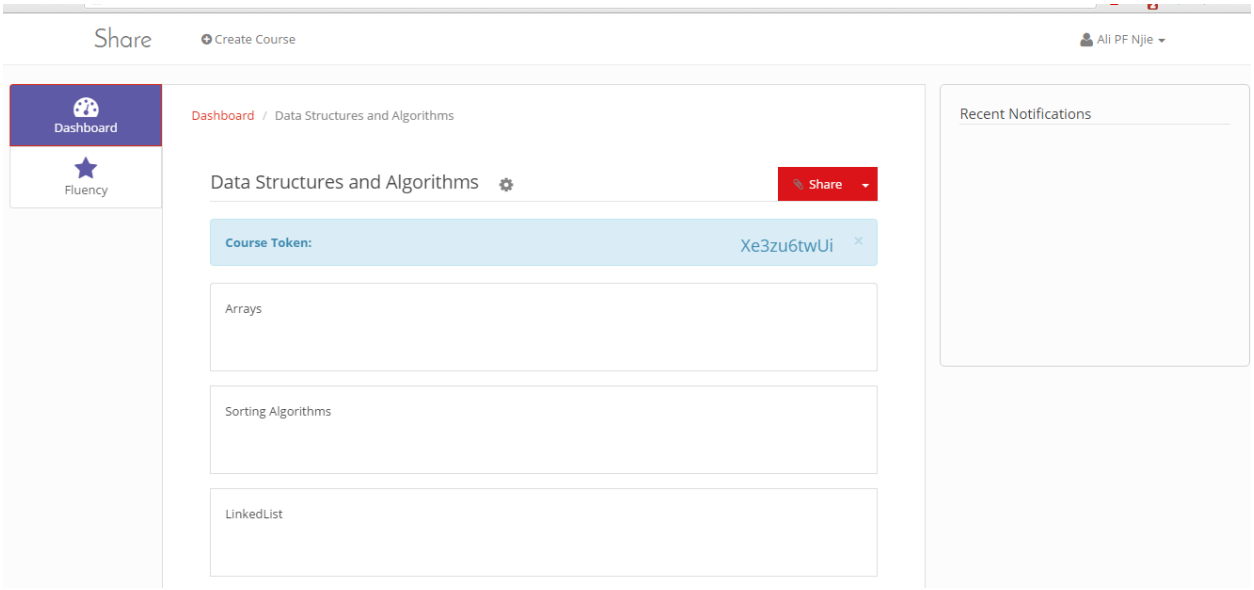


Figure 4.10: Course Plan

The database structure for the course outlines is as follows.

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	course	int(11)			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values More
2	sectionA	varchar(255)			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values More
3	sectionB	varchar(255)			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values More
4	sectionC	varchar(255)			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values More
5	sectionD	varchar(255)			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values More
6	sectionE	varchar(255)			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values More
7	created_at	timestamp			Yes	NULL		Change Drop Primary Unique Index Spatial Fulltext Distinct values More
8	updated_at	timestamp			Yes	NULL		Change Drop Primary Unique Index Spatial Fulltext Distinct values More

Figure 4.11: Course Plan Table

4.7. Material Sharing & Review

This makes up important component of Courseware+. Request that are associated with material sharing and review are handled by the MaterialController in the Controllers folder. Its

view files are located in the resources/views/tutor/ for the tutor, and the resources/views/student folder for students.

File sharing in this iteration Courseware+ has been limited to text and pdf files.

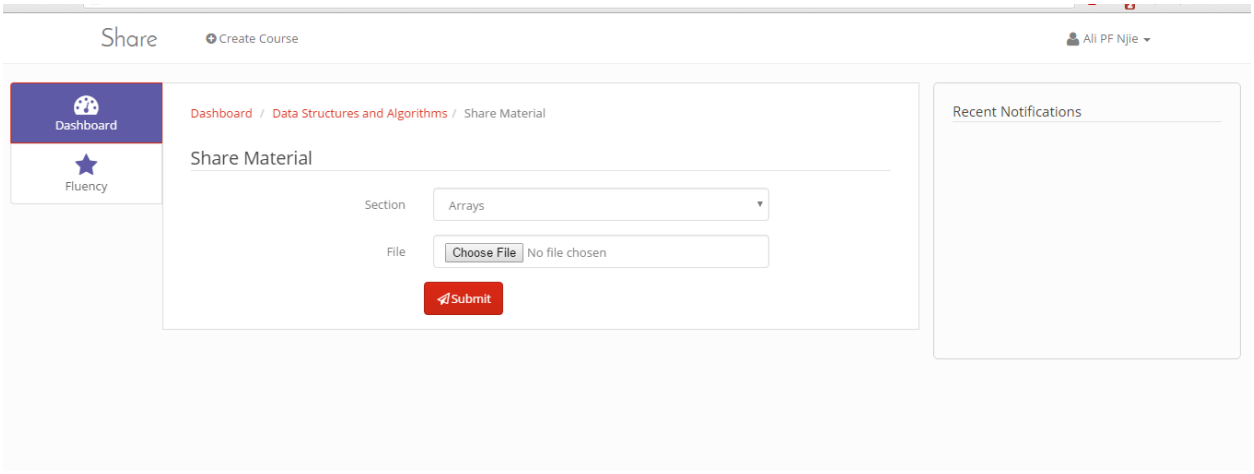


Figure 4.12: Share File

On successfully sharing a file, the courseboard view changes such that the file shared is displayed under the section to which it was shared.

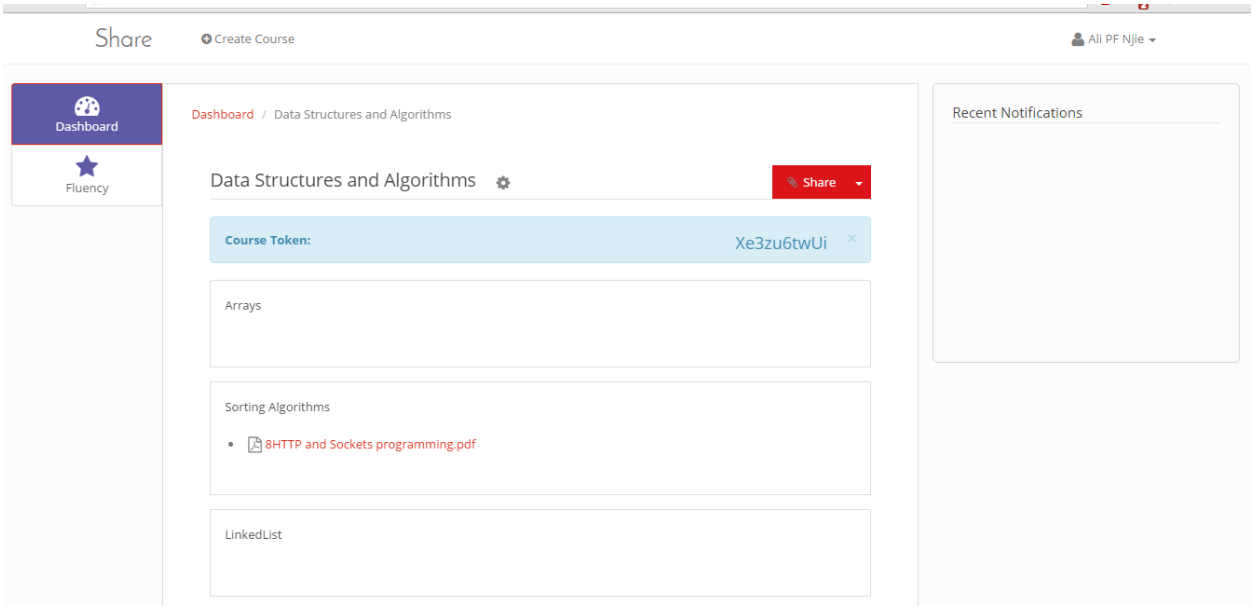


Figure 4.13: File in Section

Review has been limited to star ratings in this iteration of Courseware+. A user selects between a minimum value of 1 to maximum value of 5 stars to show how useful the material has been to them.

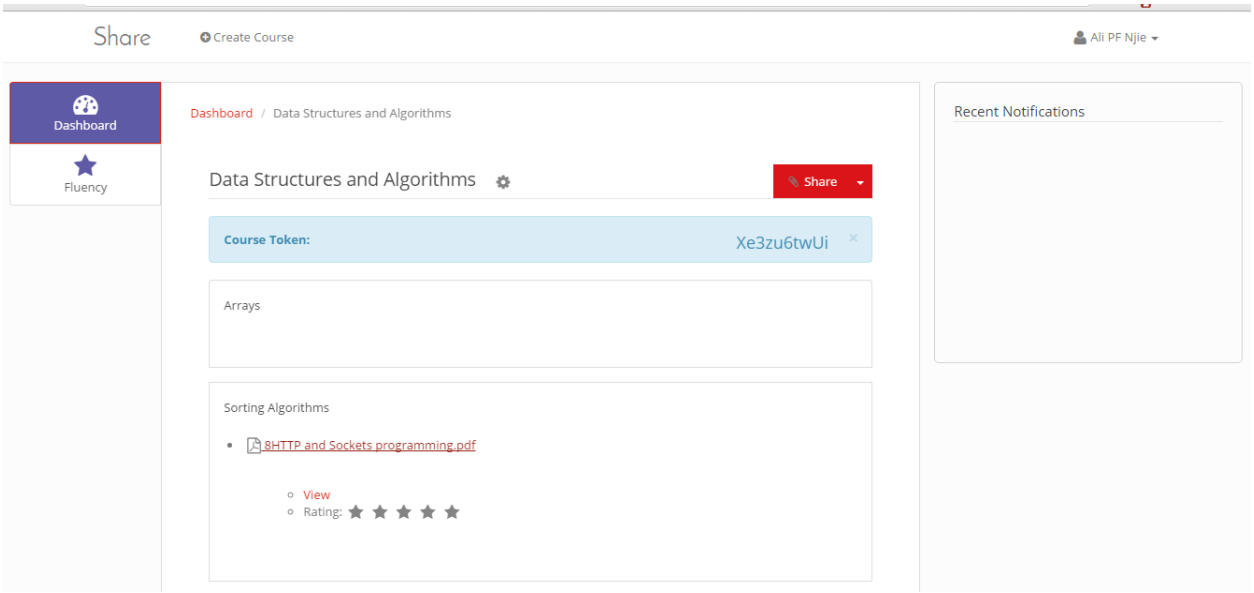


Figure 4.14: Rate File

After receiving at least one rating, the average rating of the material is displayed on its right side. This is to give the users and overview of how useful the material has been to previous users.

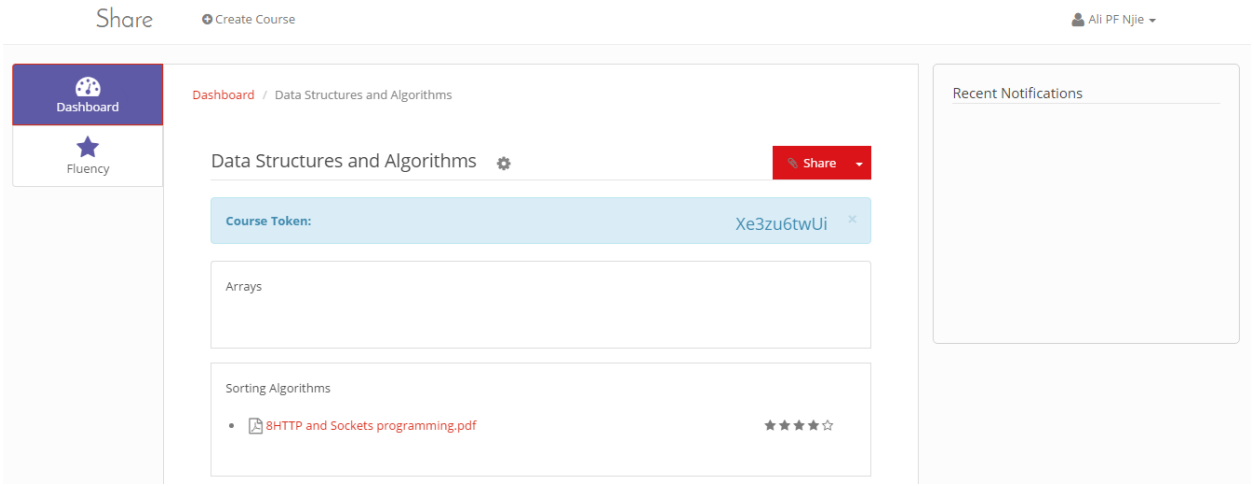


Figure 4.15: Display Rating

The files shared are stored in a folder in the public folder called **materials**. Each course has a folder create for it with its id as the folder name.

The database table used to track the materials shared by users is as follows.

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	id	int(10)		UNSIGNED	No	None	AUTO_INCREMENT	Change Drop Primary Unique Index Spatial Fulltext More
2	course	int(11)			No	None		Change Drop Primary Unique Index Spatial Fulltext More
3	section	varchar(255)			No	None		Change Drop Primary Unique Index Spatial Fulltext More
4	owner	int(11)			No	None		Change Drop Primary Unique Index Spatial Fulltext More
5	file	varchar(255)			No	None		Change Drop Primary Unique Index Spatial Fulltext More
6	path	varchar(255)			No	None		Change Drop Primary Unique Index Spatial Fulltext More
7	totalRating	int(11)			No	None		Change Drop Primary Unique Index Spatial Fulltext More
8	numRating	int(11)			No	None		Change Drop Primary Unique Index Spatial Fulltext More
9	created_at	timestamp			Yes	NULL		Change Drop Primary Unique Index Spatial Fulltext More
10	updated_at	timestamp			Yes	NULL		Change Drop Primary Unique Index Spatial Fulltext More

Figure 4.16: Material Database Table

The database table used to track which user rated which material is as follows.

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	user	int(11)			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values More
2	material	int(11)			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values More
3	rating	int(11)			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values More
4	created_at	timestamp			Yes	NULL		Change Drop Primary Unique Index Spatial Fulltext Distinct values More
5	updated_at	timestamp			Yes	NULL		Change Drop Primary Unique Index Spatial Fulltext Distinct values More

Figure 4.17: User Rating Table

4.8. Extra Resources

The user interface was design to be simplistic and intuitive. This was accomplished using the **Bootstrap** library. The bootstrap theme used is **Simplex.** The system also made use of the **font-awesome icons** to better communicate with its users.

The git version control tool was used to manage the project. The project is hosted on github by user apfnjie.

5. Testing & Results

This section will describe the testing that was performed on the Courseware+ system, its results and the implication of those results. The section will also define the testing goals, the method/type of testing that was used and the parameters of the testing. Below is a summary of a combination of unit and component testing for the system.

Feature	Front-End	Back-end	Database
User Registration & Login			
User Registration	✓	✓	✓
User Login	✓	✓	✓
Course Creation & Registration			
Course Creation	✓	✓	✓
Course Registration	✓	✓	✓
Course Plan Creation	✓	✓	✓
Content Sharing			
Share Material	✓	✓	✓
Rating Materials	✓	✓	✓
Display Ratings	✓	✓	✓
Fluency			
Rate Sections	✓	✓	✓
View General Fluency	✓	✓	✓

The Laravel framework caters for testing of application. Test files are located at the tests directory of the application folder.

5.1. Testing Objectives & Goals

The goal of Courseware+ Testing is to ensure that the system performs as per the functional requirements specified in the Requirements Specification chapter.

5.2. Unit Testing

This section will focus on isolating functional requirements and testing it to see if it behaves as expected.

User Registration & Login

A RegistrationTest class was created which extends the laravel built in TextCase class, and information was provided to it.

```
class RegistrationTest extends TestCase
{

    use DatabaseTransactions;

    public function testNewUserRegistration()
    {
        $this->visit('/register')
            ->type('Modu Sowe', 'name')
            ->type('hello1@in.com', 'email')
            ->type('hello1', 'password')
            ->type('hello1', 'password_confirmation')
            ->type('type', 'lecturer')
            ->seePageIs('/home');
    }

    public function testUserLogin()
    {
        $this->visit('/login')
            ->type('email', 'apfnjie@gmail.com')
            ->type('password', 'chicharito14')
            ->seePageIs('/home');
    }

}
```

Figure 5.1: Registration Test File

After which the phpunit command was run on the windows terminal to test the Registration and Login functional requirements and the following result was obtained.

```
C:\xampp\htdocs\projects\courseware+>phpunit
PHPUnit 4.8.24 by Sebastian Bergmann and contributors.

...

Time: 2.16 seconds, Memory: 12.00Mb

OK (3 tests, 10 assertions)

C:\xampp\htdocs\projects\courseware+>
```

Figure 5.2: Registration Test Result

Course Creation, Registration & Create Outline

A class was created for testing course creation, course registration, and create course outline functional requirements.

```php
class CourseTest extends TestCase
{
    use DatabaseTransactions;

    public function testCreateCoursePlan()
    {
        $this->visit('/course/courseboard/plan/2')
            ->type(1, 'course')
            ->type('Intro', 'sectionA')
            ->type('First', 'sectionB')
            ->type('Second', 'sectionC')
            ->type('Third', 'sectionD')
            ->type('Conclude', 'sectionE')
            ->press('Create')
            ->seePageIs('/course/courseboard/2');
    }

    public function testRegisterCourse()
    {
        $this->visit('/course/register')
            ->type('AGsgTRdsBe', 'token')
            ->press('Register')
            ->seePageIs('/home');
    }
}
```

Figure 5.3: Create Course Test File

After running the phpunit command on the terminal, the following result was obtained.

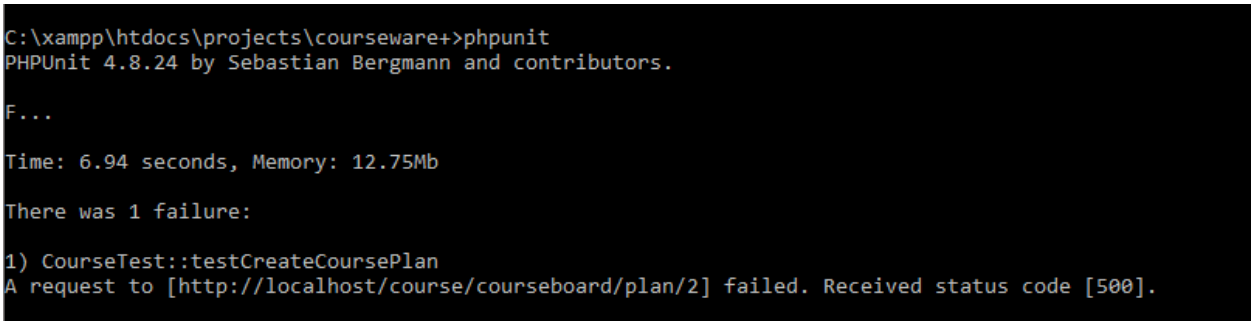


Figure 5.4: Course Test Result

The reason for the failure in the test as is the case in the remaining test is that it requires a user to login before it can run the test successfully. The system is developed such that for every make request, the system first checks to make sure a that there is an active user session, otherwise the request will have failed.

For that reason, the testing was done manually for the rest of the functionalities.

The result for the testing is as follows.

Functionality	**Inputs**	**Results**	**Implications**
Create Course	• **Name** = Human Computer Interaction • **Description** = Course to teach how computers	Course successfully created. **Note**: When either field was empty, it returned validation errors to the view.	Course creation implementation successfully

	and humans interact		
Course Registration	• **Token =** LkdUYxMdli	Course registration successful.	Course Registration successfully implemented
Create Course Plan	• **Course** = 4 • **sectionA =** Intro • **sectionB =** Web • **sectionC =** Mobile • **sectionD =** Virtual Reality • **sectionE =** Conclusion	Plan was successfully created. Note: when any of the fields was empty, validation returned error message to view.	Plan creation successfully implemented.

Table 5.1: Course Test

Material Sharing & Reviewing

Because of the requirement to check if there is an active user session before processing any request. Laravel unit testing returned failures.

However, each functionality was manually tested and the results are as follows.

Functionality	**Inputs**	**Results**	**Implications**
Share Material	• **Section =** Intro	The result was unsuccessful for the	The functionality works as expected

	• **Material =** test.pdf (size = 4.58MB) & test2.pdf (size = 458KB)	first file. This is because it exceeded the file size limit. The second one was successful	
Rate Material	• **Material =** test.pdf • **User =** 5 • **Rating = 4**	The rating was successfully implemented.	The functionality works as expected.

Figure 5.2: Material Test

6. Conclusion & Recommendation

6.1. Overview

The first iteration of Courseware+ focused more on integrating basic peer learning concepts into tools used for learning. As a result, enough emphasis was not placed on some of the broader uses of courseware like assignment submission, online quiz creation, and other type of users who are engaged in the educational process such as Faculty Interns and the Registry department.

The system has successfully integrated the basic concepts of peer learning. It promotes the sharing of information in the form of materials among student. Since lecturers can also upload materials, it has improved on the existing system courseware uses.

In addition to the universalization of sharing/uploading materials, the system has also incorporated reviewing of shared information in the form of a five-star rating. This review can also

be thought of as recommendation to peers on what material they might find helpful and what material they might find not helpful.

6.2. System Limitation

As far as information sharing is concerned, only pdf and other text files are allowed due to file size restrictions by the server. The system does not allow for the sharing of online articles, videos, audio podcast, etc. which are all tools used in modern day education.

Also, lecturers are supposed to teach courses and not create them. The privilege of creating courses and enabling them should be assigned to another user group. Also, Courseware+ only has two types of users. The educational system however has more actors than lecturers and students and the system failed to address that.

6.3. Recommendation

Future work should focus on creating other relevant user groups that are involved in the education process like Faculty Assistants. Provisions should also be made for guest users to be able to observe courses on Courseware+. In should also cater separately for students who opt to audit a given course separately.

In addition to adding more user groups, future work should also focus on integrating assignment submissions and online quiz to the Courseware+ system. Likewise, it should allow for more interaction on the platform at the lecturer's discretion. This is because some courses encourage students to discuss on given course materials. Thus it will be beneficial if a discussion component can be added to the shared materials at the lecturer's discretion.

Finally, as helpful as it is for the user to use student ratings of their understanding to get an opinion on the fluency of the class in covered topics, more can be achieved with those ratings. A

study time table generator algorithm should be designed to use the ratings and suggest a weekly time table for students.

Bibliography

Topping, K. J. (2005). Trends in peer learning. *Educational psychology*,*25*(6), 631-645.

Boud, D., Cohen, R., & Sampson, J. (1999). Peer learning and assessment. *Assessment & Evaluation in Higher Education*, *24*(4), 413-426.

Components. (n.d.). Retrieved April 17, 2016 from http://getbootstrap.com/components/

Installation. (n.d.). Retrieved April 17, 2016, from https://laravel.com/docs/5.2

Our Learning Goals: Skills for a lifetime. (n.d.). Retrieved April 17, 2016, from http://www.ashesi.edu.gh/academics/learning-goals.html