Media Engineering and Technology Faculty
German University in Cairo

Improved CMS System

Bachelor Thesis

Author: Ahmed Atef Askora
Supervisors: Dr.Milad Ghantous
Submission Date: 1 August, 2021

Media Engineering and Technology Faculty
German University in Cairo

Improved CMS System

Bachelor Thesis

Author: Ahmed Atef Askora
Supervisors: Dr.Milad Ghantous
Submission Date: 1 August, 2021

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor Degree

(ii) due acknowlegement has been made in the text to all other material used

---

Ahmed Atef Askora
1 August, 2021

Acknowledgments

First of all, I would sincerely to thank God for the power he gave me to accomplish this project. I would also like to express my gratitude and appreciation to my supervisor **Dr. Milad Ghantous**, who has been following up my work, helping and supporting me through the whole process. I would also like to thank him for his great effort and providing me all the facilities for this project. Finally, I would like to thank my parents and my friends for their continuous support.

Abstract

The concept of traditional education has changed completely within the last couple of years. Being physically present in a classroom isn't the only learning option anymore with the rise of the internet and new technologies also with the spreading of the COVID-19 pandemic. Due to this, German University in Cairo (GUC) has offered a new platform to access all the material while staying at home to reduce the spread of COVID-19. But the problem is there are a lot of disadvantages and missing modules in the website. So the aim of the project is to solve the problems of the current website and missing modules by adding new features and a better way to organize the content for easier access using the latest web technologies.

Contents

Chapter 1

Introduction

The Internet has significantly changed how we communicate with one another as well as how we access, share and facilitate information. The lecturers and teachers now acknowledge the way the world is already developing, and understand the significance of online teaching and the role that it plays in students learning and their future workplace environment. Online teaching's potential advantages involve increased educational access; it provides a high-quality learning opportunity, improves student outcomes and skills, and expands educational choice options. Therefore, location, time, and quality are no longer considered factors in seeking degree courses or higher education because of online teaching. In addition The COVID-19 pandemic has forced the world to engage in the use of virtual learning as it has resulted in schools shut all across the world. Globally, over 1.2 billion student are out of the physical classroom. As a result, education has changed dramatically, with the distinctive rise of e-learning, whereby teaching is undertaken remotely and on digital platforms. In other words online teaching nowadays especially in these current situations is indispensable.

1.1 Motivation

And due to pandemic situation the German University in Cairo developed a new website for Content Management System to help students continue studying online while the closure of schools and universities to reduce the growing number of infections. This website helps students alot in their studies as it contains all the material that students need (lecture slides, assignments, projects, etc.) in addition to the Video On Demand (VOD) of the recorded lectures and tutorials that the lecturers and teaching assistants record and upload to the website. In addition this platform has a feature that enables the lecturers and Teacher Assistant (TA)s to post an important announcements about the course(quiz timing, deadline submission ,etc.) and thus the students have to regularly check the announcements section of every course. Also this new platform has a discussion board that enables the student to discuss any ambiguous part with lecturers and TAs using posts and replies. So using all these useful features this platform has become

the only way for the student to follow with the university and all curriculum. But unfortunately there are some problems in the current platform which make the students and instructors be more confused while using CMS. One of the biggest problem is that the layout of contents is not organised and too ambiguous. There is no folders and all contents are ordered by weeks all under each other so the students must scroll too much to find what they want. Another problem there is no search for contents. The discussion board is separated from the contents there is no relation between each other. Also the interface is a bit confusing to deal with so in this project we are trying to develop an Improved CMS System.

1.2 Aim of the project

As a result of these problems in this project we are going to improve the interface of the CMS, reduce and fix bugs, add more features in a way to be more organized and more comfortable to both lecturers and students. We are going to organize the material into folders (folder for lecture, folder for tutorial, folder for assignments, etc.). We are going to make the discussion board related to content for example each lecture will have its own questions and answers and the student or the lecturer can view the discussion only for a specific lecture. Also in this project we will add a tags for videos and contents to make it easier for students to know what are the sub-topics included in the recorded video or inside the lecture. Also we are going to add search for these tags. And in order not to miss any important announcement for quizzes or deadlines, in the Improved CMS System we are going to send a desktop notification and an email notification to the students once the instructor post a new announcement. All this and more comes in the Improved CMS System.

1.3 Thesis outline

This thesis consists of five chapters. Chapter one is an introduction to the project. Chapter two is related work. It covers different online teaching platforms similar to our project. Chapter three is methodology. It shows the structure of the project in addition to description of new features and the user functionalities of the new website. Chapter four discusses in details the implementation of the new system and explains also the tools and technologies used in the project. Finally, chapter five concludes the thesis with an overall summary about the enhancements of the project and suggests some new features and improvements to be added as a future work.

Chapter 2

Related Work

After showing the importance of online teaching speacially in the COVID-19 time and giving a general idea about the problems of current CMS platform and how we are trying to fix them in a way to be more organized for students and lecturers, in this chapter we will talk about the most popular online teaching platform like Google classroom and moodle, discussing their features, describing how they look like, making a comparison between them to make the best use of the nice features they contain to make a nice CMS platform that satisfies all the students and lecturers.

2.1 Google Classroom

Google Classroom is a free web service developed by Google for schools that aims to simplify creating, distributing, and grading assignments. The primary purpose of Google Classroom is to streamline the process of sharing files between teachers and students. It is estimated between 40 to 100 million people use Google Classroom. [1]

Google Classroom integrates Docs, Sheets, Slides, Gmail, and Calendar into a cohesive platform to manage student and teacher communication. Students can be invited to join a class through a private code, or automatically imported from a school domain. Teachers can create, distribute and mark assignments all within the Google ecosystem. Each class creates a separate folder in the respective user's Drive, where the student can submit work to be graded by a teacher. Assignments and due dates are added to Google calendar, each assignment can belong to a category (or topic). Teachers can monitor the progress for each student by reviewing revision history of a document, and after being graded, teachers can return work along with comments.

2.1.1 Google Classroom Features

- **Stream**

Stream feature in Google Classroom is the main feature of the platform, it's considered the home page as it contains all contents, discussion and classwork questions about the class. Google Classroom gives the student or the teacher the ability to announce any thing to the class, the students can add any posts also the instructors can add contents and all of this is shown in the stream.

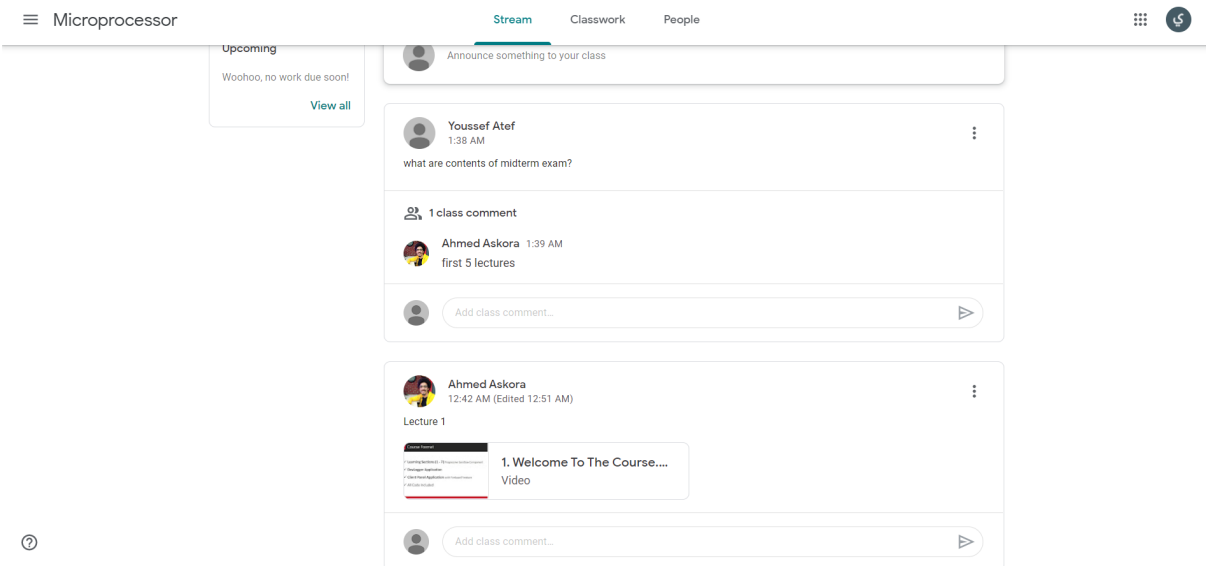


Figure 2.1: Stream Feature

- **Add Content**

Google Classroom enable the teachers to add content in different ways as they can add from their personal device, from google drive, from YouTube, or using any link also the teachers can add caption to the uploaded folder and style it as they want.

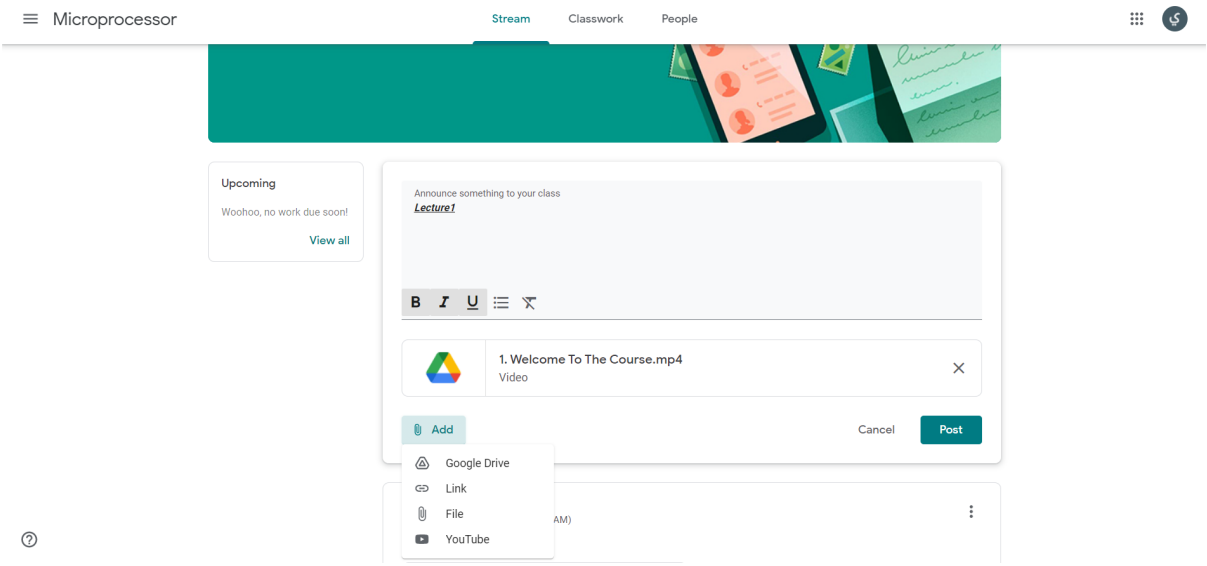


Figure 2.2: Add Content Feature

- **Classwork**

Google Classroom has a classwork feature that enables the instructor to add an assignment or a quiz or ask a question and specify the grade of each question. Also the instructors can choose the type of the exam weather it's a MCQ or written questions as well as they can select a deadline for the exam, and the exam will be assigned to the students to answer for it.

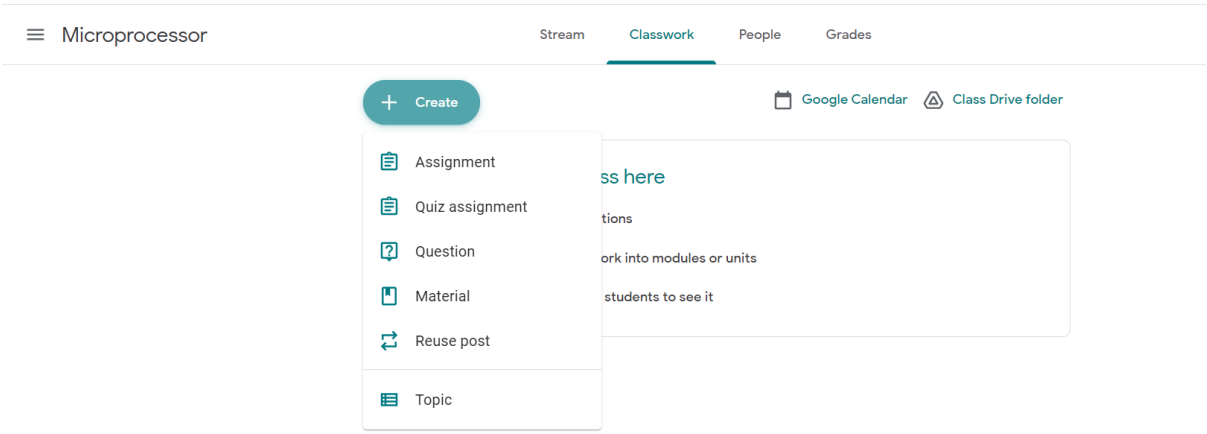


Figure 2.3: Classwork Feature

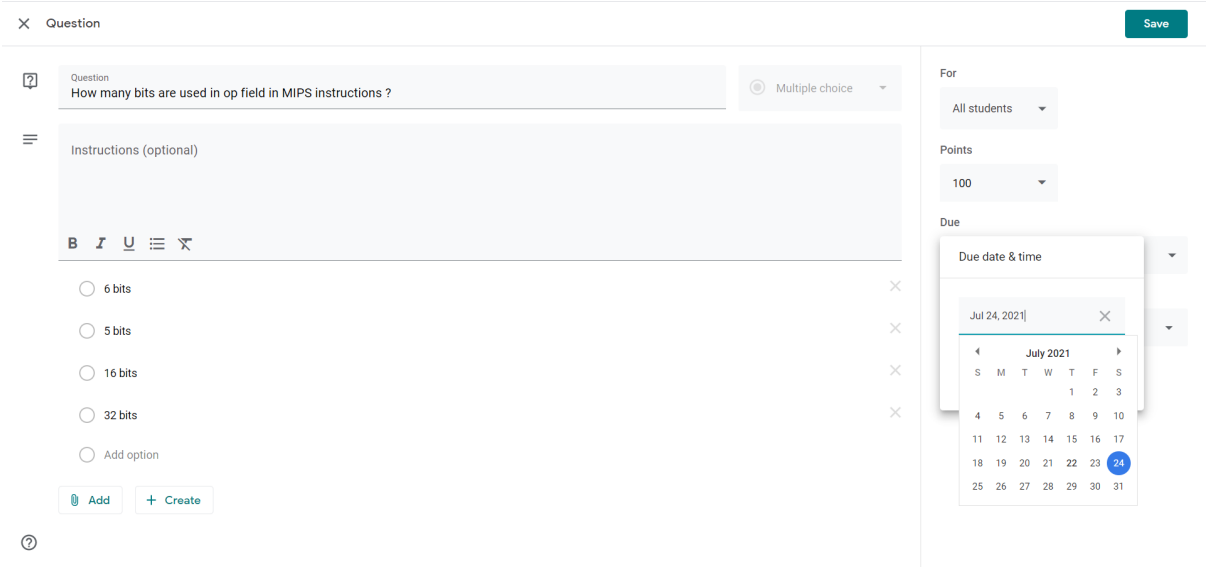


Figure 2.4: teacher asking a question

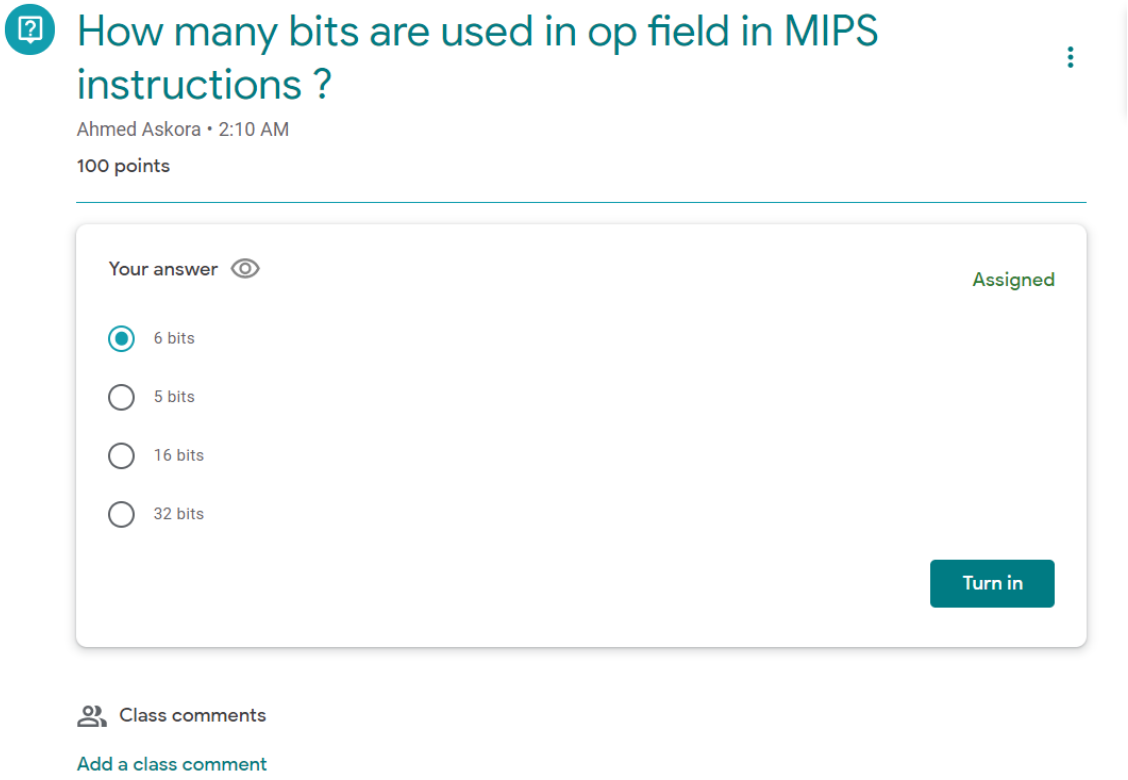


Figure 2.5: student answering the question

2.2 Moodle

Moodle is a free software, a learning management system providing a platform for e-learning and it helps the various educators considerably in conceptualizing the various courses, course structures and curriculum thus facilitating interaction with online students. Moodle was devised by Martin Dougiamas and since its inception, its primary agenda has been to contribute suitably to the system of e-learning and facilitate online education and attainment of online degrees. Moodle actually stands for Modular Object-Oriented Dynamic Learning Environment and statistics reveal that about 14 million consumers are engaged in about 1.4 million courses propagated by this learning management system. [2]

2.2.1 Moodle Features

- **Notification Feature**

Notifications when enabled, users can receive automatic alerts on new assignments and deadlines, forum posts and also send private messages to one another.

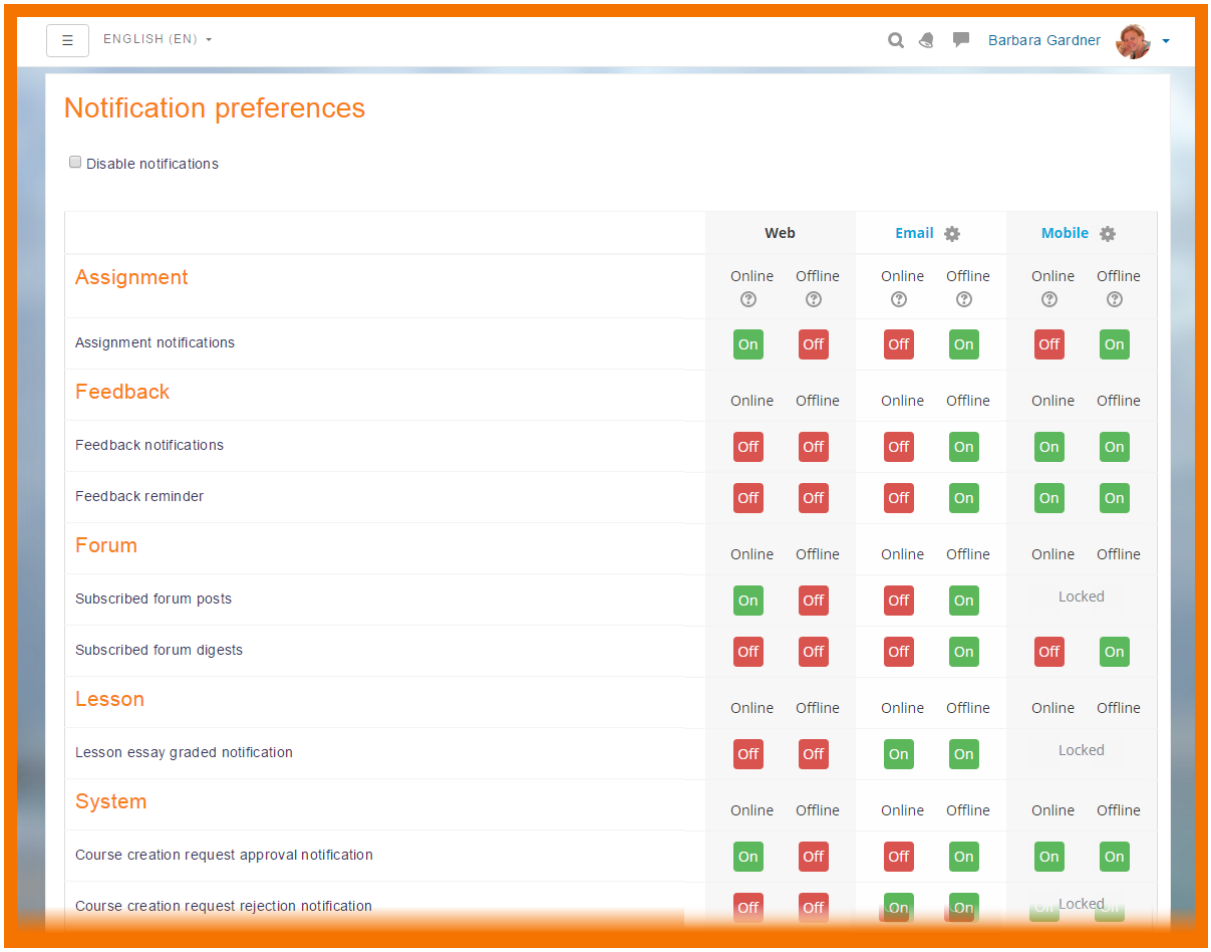


Figure 2.6: Notification Feature

- **Calendar Feature**

Moodle's calendar tool helps students keep track of their academic or company calendar, course deadlines, group meetings, and other personal events.

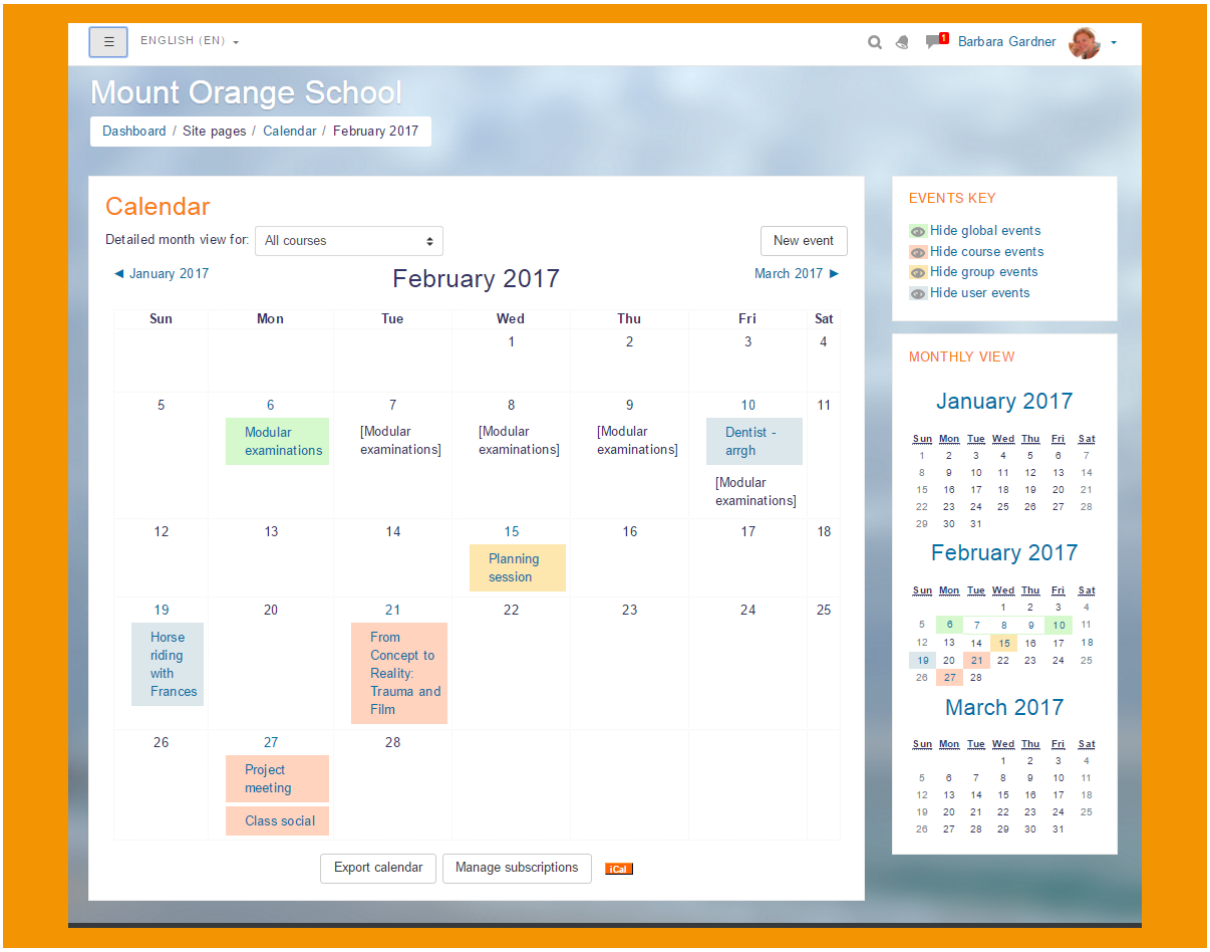


Figure 2.7: Calendar Feature

- **Forums Feature**

Forms or discussion board helps students to work and learn with other students and with lecturers.

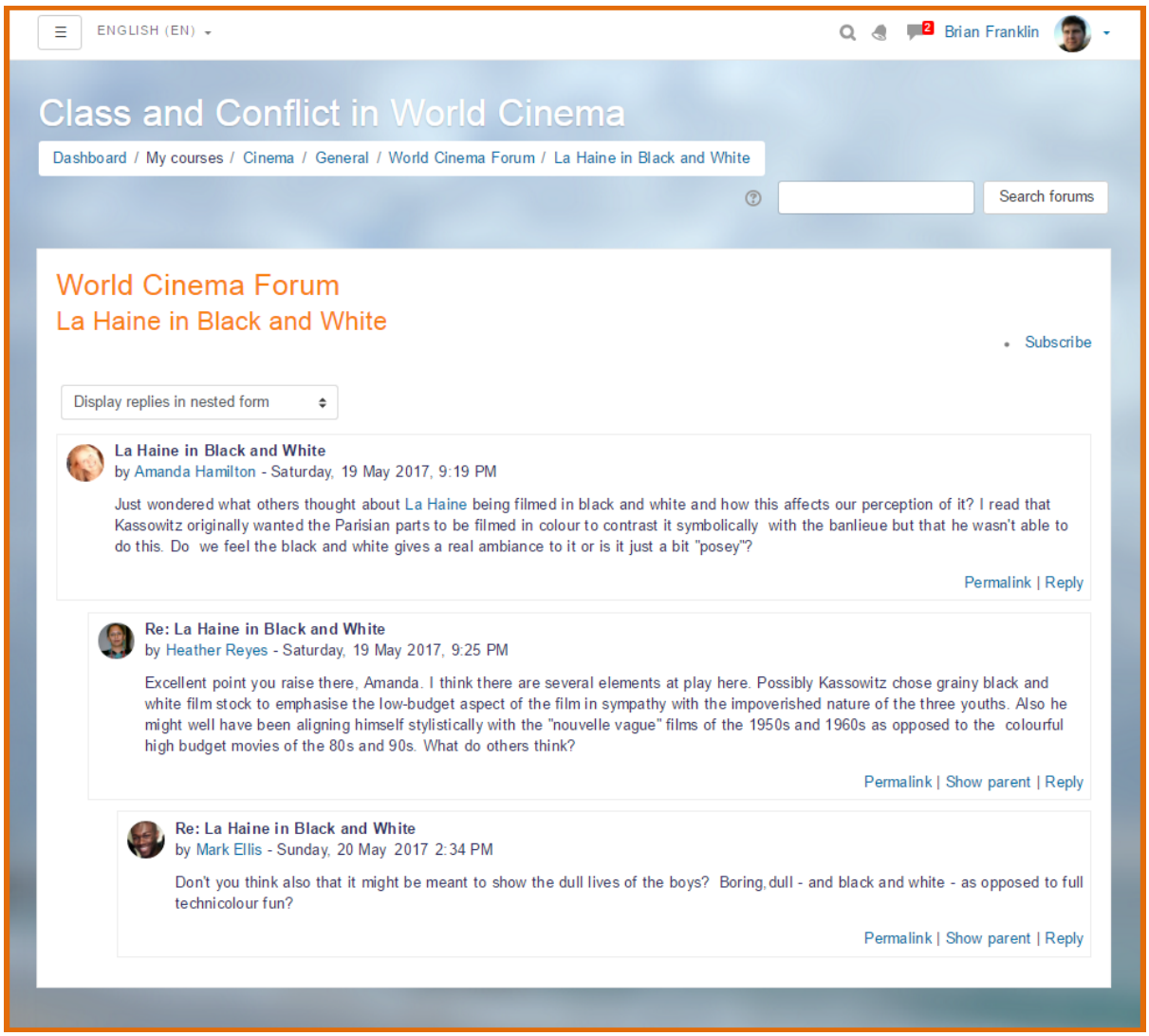


Figure 2.8: Forums Feature

- **Download Content Feature**

Download course content (if allowed) on student's personal device

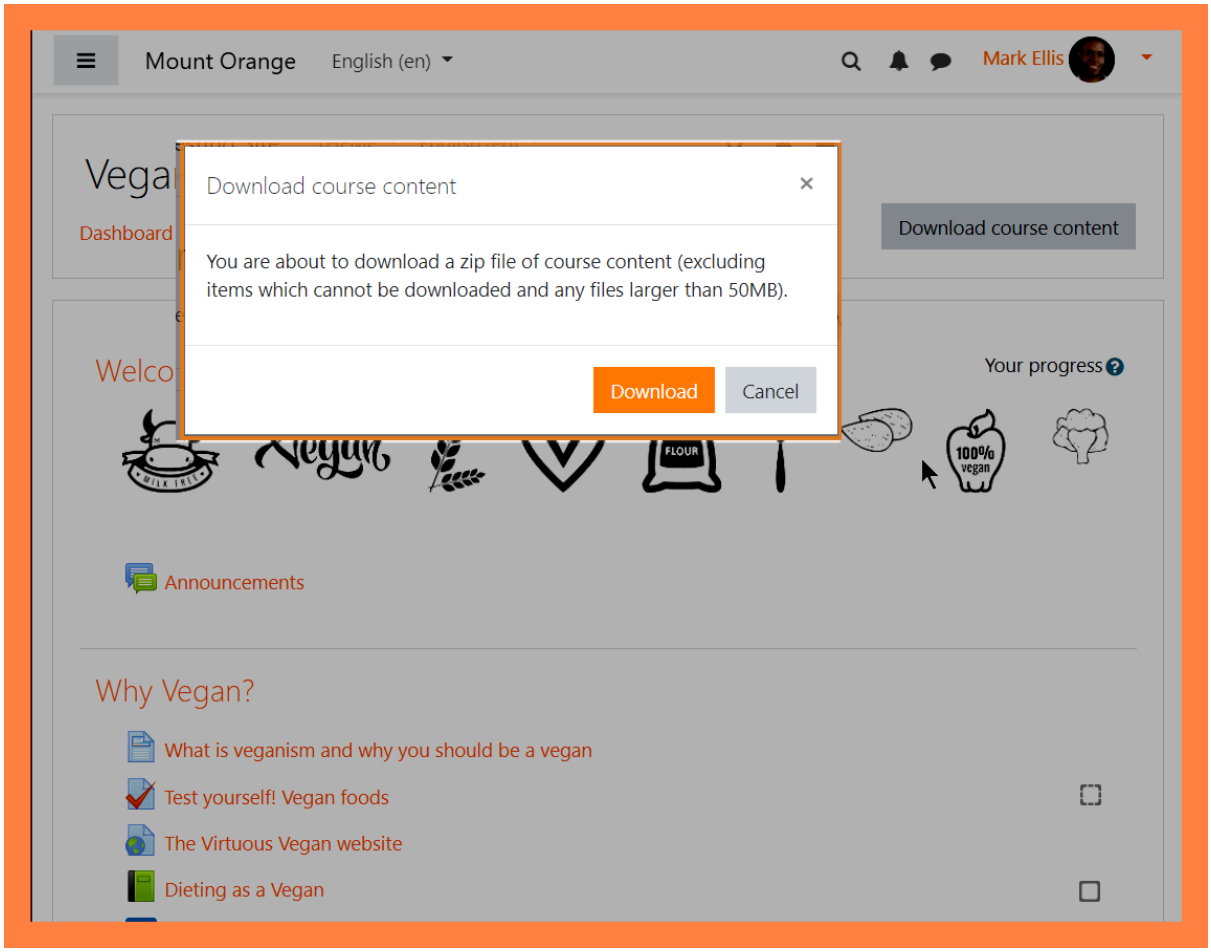


Figure 2.9: Download Content Feature

2.3 Current GUC CMS

This section represents the current content management system for the GUC, it has been developed recently to help students at the current pandemic situation of COVID-19 and it has some features for example:[3]

- **View Courses**

A student can see all enrolled courses by season.

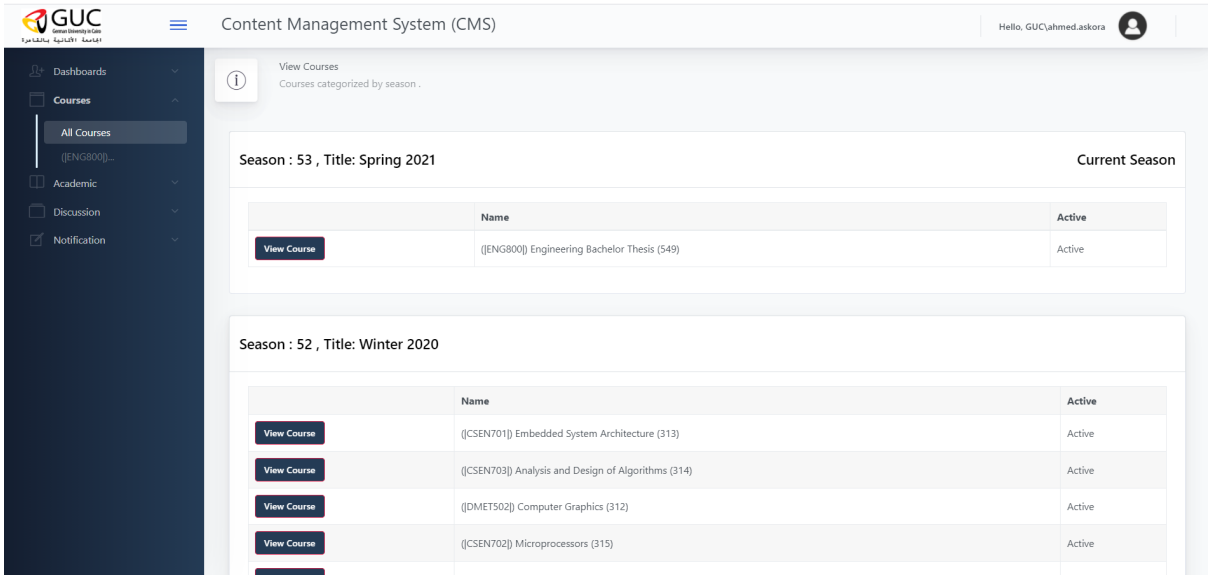


Figure 2.10: View Course Feature

- **Course Announcement**

An instructor can pin an announcement at the main page of the course to remind the students of new updates.

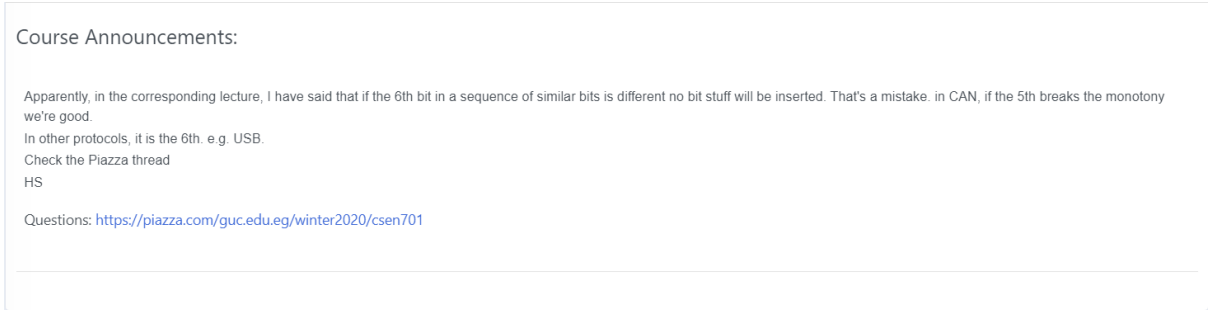


Figure 2.11: Course Announcement Feature

- **View Course Contents**

A student can see all contents of his/her enrolled courses labeled by weeks and the student can download these contents. But this is one of the disadvantages of current CMS as the all content of the course are in one page and the students have to scroll too much to find the content they want.

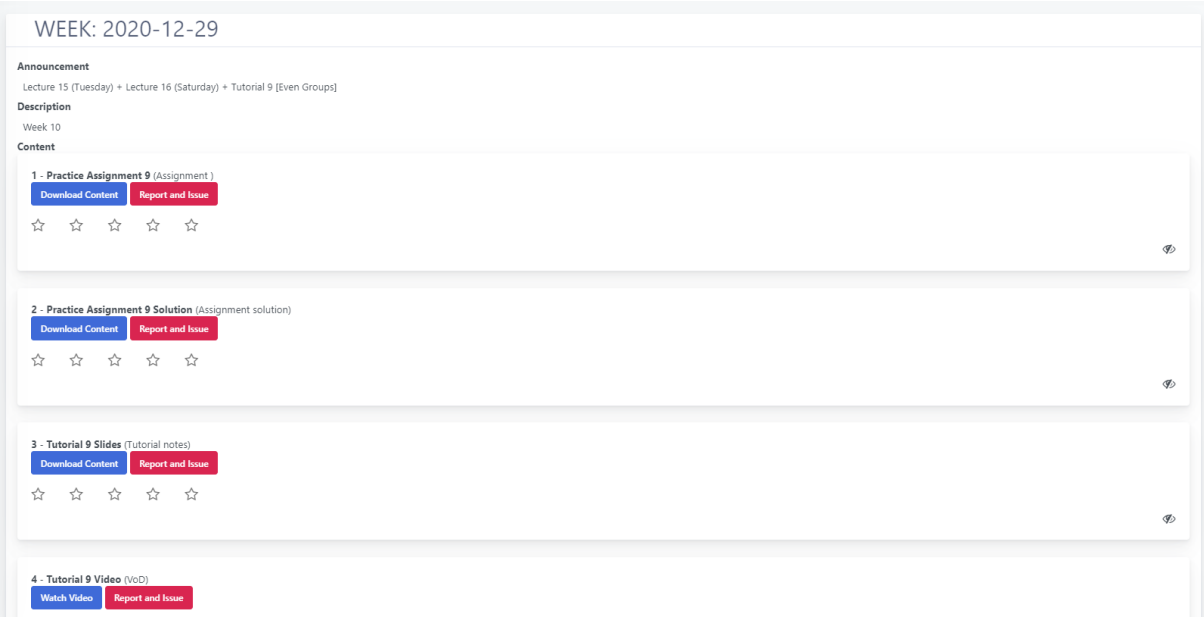


Figure 2.12: Course Contents Feature

- **Watch Video**

A student can watch a recorded videos of lectures and tutorials online by clicking on watch video button and a pop up video will be shown to the student. Also the pop up video is not efficient and hated for most of the GUC students so it considered a disadvantage more than a feature.

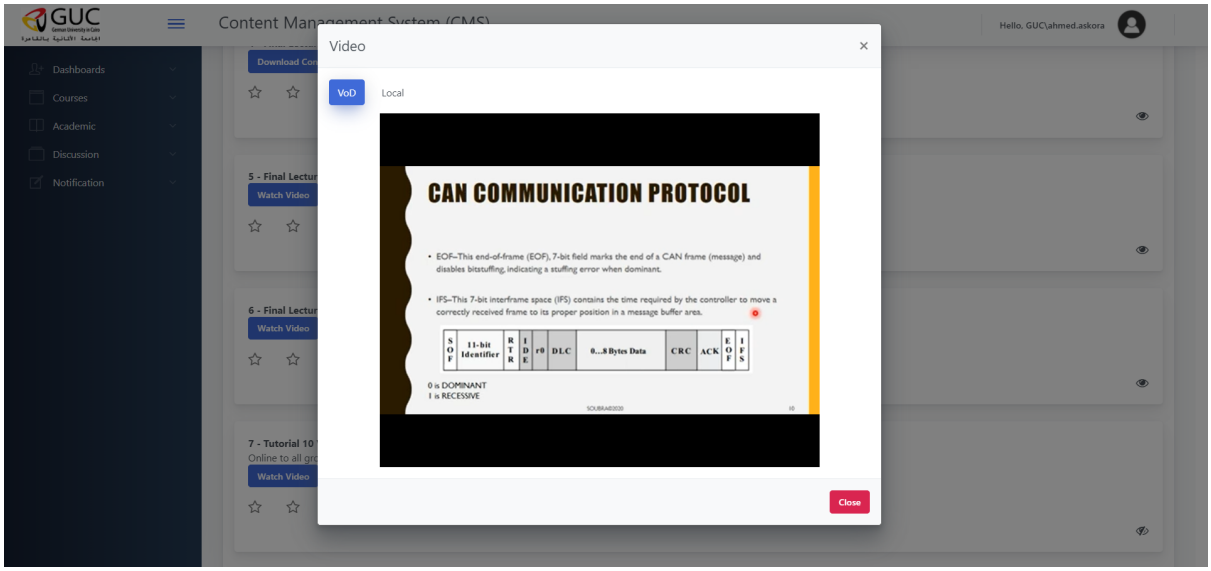


Figure 2.13: Watch Video Feature

- **Course Discussion**

A student can see all discussions of any course posted by an instructor or by another student.

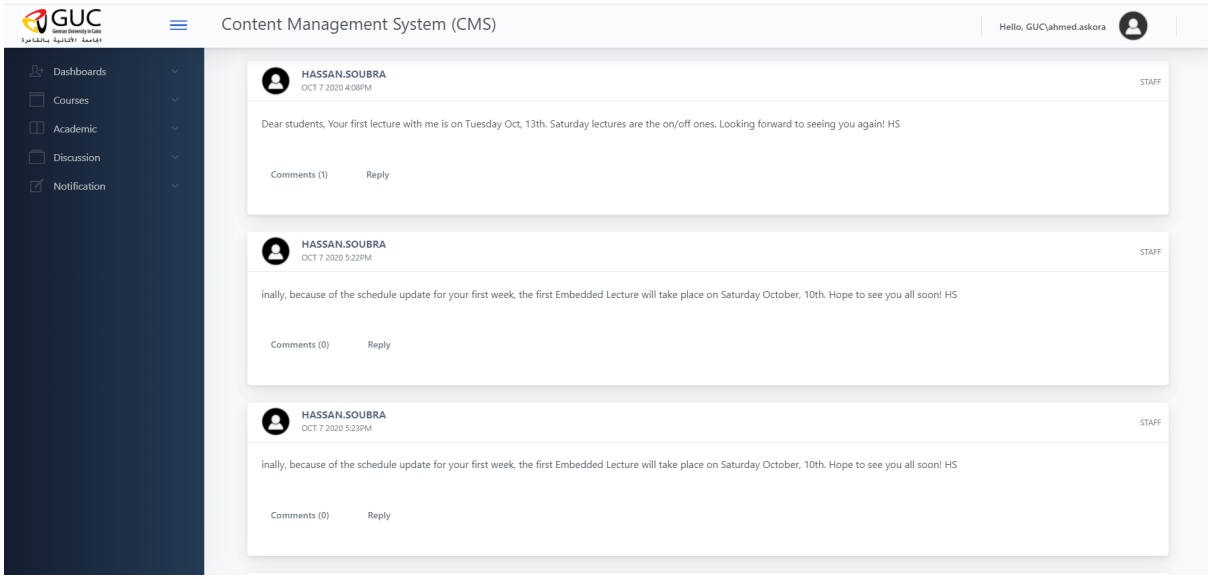


Figure 2.14: Course Discussion Feature

- **Reply to discussion**

A student or an instructor can reply to any posted discussion by clicking on reply and a pop up text filed will show to write the reply. Also this is considered one of the disadvantages of the current CMS as the pop up dialogue is not efficient and seems unfamiliar way to write a reply.

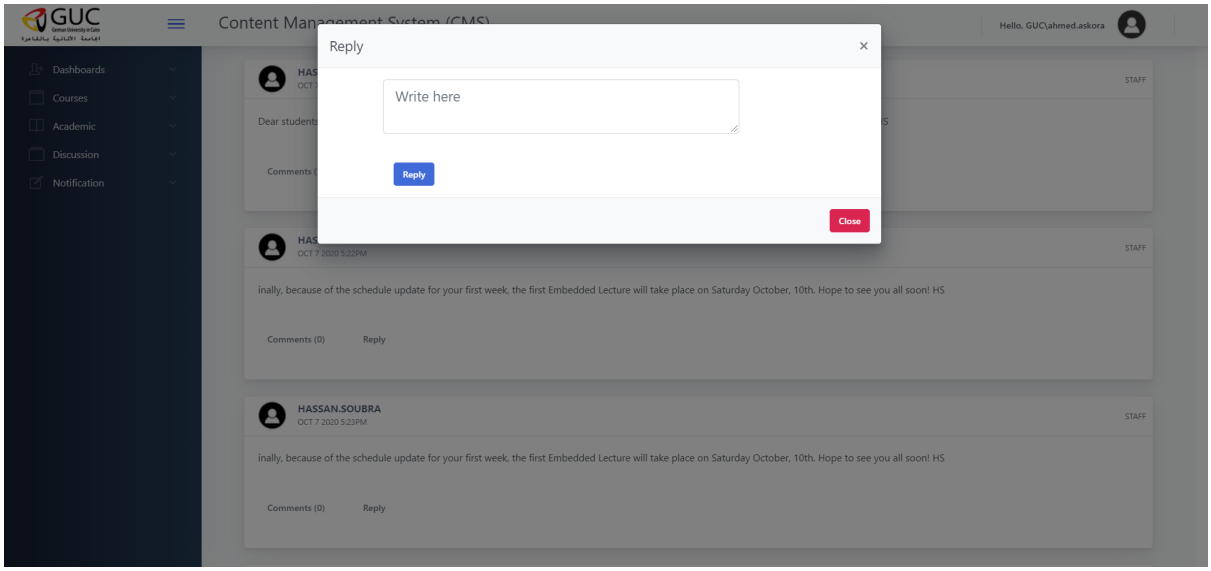


Figure 2.15: Reply to discussion Feature

Chapter 3

Methodology

As discussed about the problems of current CMS website and the missing modules the Improved CMS System is trying to fix them in a way to be more comfortable for students and instructors by making the instructor be able to organise course contents in folders, adding an improved discussion module in an easy and modern way to discuss thoughts between students and instructors, also by creating a notification module to notify user for any discussion has been posted or any content has been uploaded, adding search module to search for contents by tags or by title and this chapter will discuss the methodology and the structure of the website Entity Relationship Diagram and how all entities work together to build the new CMS website.

3.1 New CMS features

This section discuss in details the features of the Improved CMS System and how these features can help both instructors and students while interacting with the website.

3.1.1 Folders

Putting all material in one scrolling page is one of the most disadvantages of current CMS as it's very annoying to scroll down and down to reach for your goal, this way of scrolling wastes too much time and can distract the student, so to over come this problem the Improved CMS System has a nice feature of organizing contents in folders and sub folders as you deal with your personal computer, by this way the instructors can organize their contents as they like and the students can easily reach for their content in a comfortable and more comfortable way.

3.1.2 Tags

The Improved CMS System comes with new and nice feature which is tags. Before uploading any content the instructor has to add tags for this content describing what this video or this content is talking about, for example #assembly, #MIPS, #Object Oriented Programming (OOP), #recursion and the instructor can add more than one tag as well. This is useful as it makes the student has an idea or an overview of content before studying it also the student can use the tag to search for all contents that matches the tag as will be discussed in next feature .

3.1.3 Search

The improved CMS systems has an essential feature which is search for contents. The students can search for contents by content title or by content tags, and this feature can save the student time and make it much easier to reach for contents.

3.1.4 Discussion

Discussion feature in the new CMS can give user the ability to post a discussion of what they think about with familiar and modern user interface as each post has its own likes, comments and each comment has its own replies. There are two types of discussion in the new CMS website content discussion and course discussion. The first type is content discussion, students can post a content related discussion to ask questions about what they don't understand for example also the instructor can post a content discussion to add a note for this specific content or make an interactive discussion between students by asking question about this content as see the students' answers in comments and replies. The second type is course related discussion and this the students ask for any question about course in general what is the grading schema, what is the location of the quiz, what is the contents included in midterm exam and so on.

3.1.5 Announcements

Announcement feature gives the instructor the ability to announce important notes to the class for example announcement about the timing and location of the next quiz, announcement about deadlines or about the contents included in midterm exam. This module is contains all announcements of all courses in one place which makes the student more focus about important updates. This feature also can reduce the using of GUC mail as it's sometimes annoying for most of users.

3.1.6 Notifications

Notification feature is important to notify the user of important updates. For example a student is getting a notification when an instructor upload a new content or update a course announcement. Also an instructor is notified when another instructor uploads a new content on his/her assigned courses. And for both student and instructor they get a notification when someone post a new discussion or when they receive a comment on their own discussions.

3.2 User functionality

This section represents the three types of users in the Improved CMS System and they are admin, staff and student and also discuss the functionality of each user and how each user can interact with the website.

3.2.1 Admin functionality

- As an admin I can create a new department so that these departments represents GUC's departments.
- As an admin I can create a new semester.
- As an admin I can add a new user (staff/student) so that each user can interact with CMS.
- As an admin I can add a new course to each of the departments so that each user can interact with courses.
- As an admin I can assign courses to staff so that they can upload contents to students.
- As an admin I can assign courses to students so that they can interact with their courses and contents.

3.2.2 Staff functionality

- As a staff I can add a new folder so that I can organize contents in these folders.
- As a staff I can edit folder name.
- As a staff I can delete folder.
- As a staff I can upload content to specific folder.

- As a staff I can edit content title or content tags.
- As a staff I can delete a specific content.
- As a staff I can post a content discussion to specific content so that students can comment and reply to interact with each others about this specific content.
- As a staff I can post a general discussion to any course..
- As a staff I can post a course announcement to any course so that I can remind student of something about the course in general.
- As a staff I can delete a specific discussion of any student.
- As a staff I can receive a notification when a new content is uploaded by another staff of same assigned course.
- As a staff I can receive a notification when a new discussion is posted by any user to any content of my assigned course.
- As a staff I can receive a notification when a new discussion is posted by any user to any content of my assigned course there is a new comment or reply to my posted discussions.
- As a staff I can search for contents by title or tags.

3.2.3 Student functionality

- As a student I can see the contents of my enrolled courses are organized in folders.
- As a student I can watch video and its discussions in the same page.
- As a student I can download contents.
- As a student I can post content related discussion to ask about content so that I can receive an answer from the instructors to be fully understood of the material.
- As a student I can post a general discussion to ask questions about the course in general.
- As a student I can see all discussions of all enrolled courses and interact with them by comments or replies.
- As a student I can receive notifications when a new content is uploaded or when someone comments or replies to my discussions.
- As a student I can search for contents by title or tags.

3.3 Sequence Diagrams

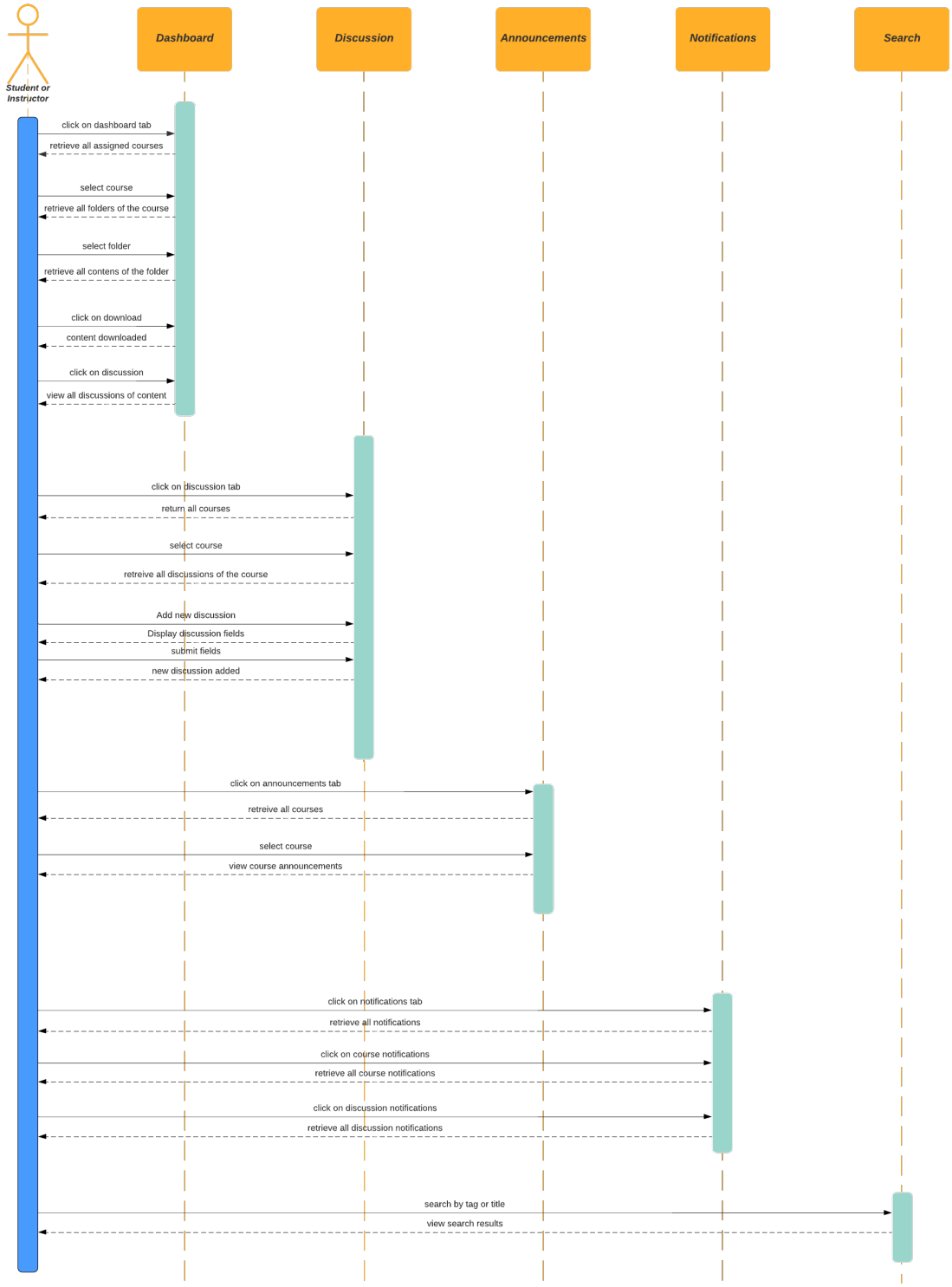


Figure 3.1: Sequence diagram of new CMS features

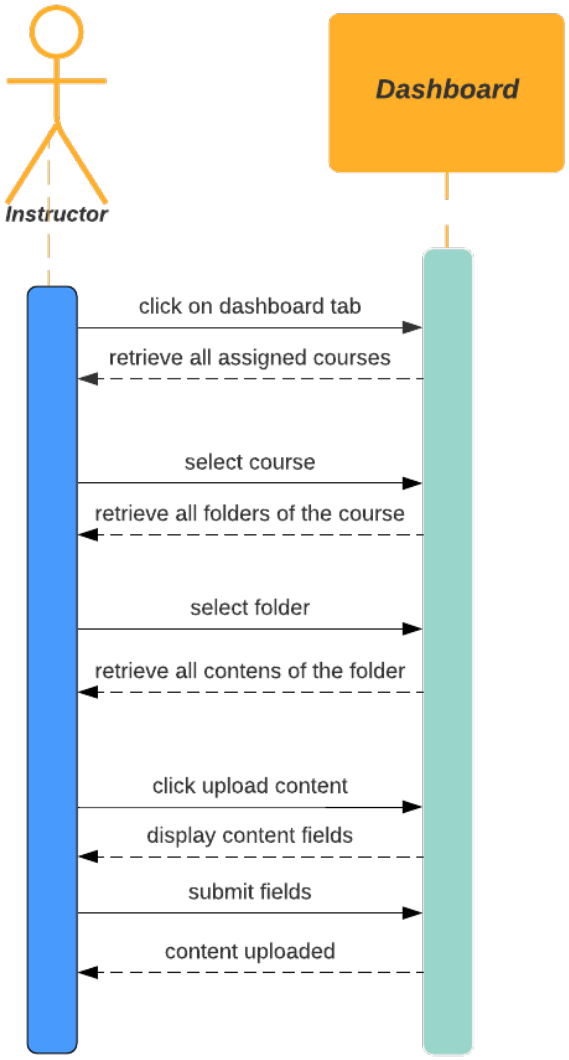


Figure 3.2: Sequence diagram uploading content

3.4 Entity Relationship Diagram

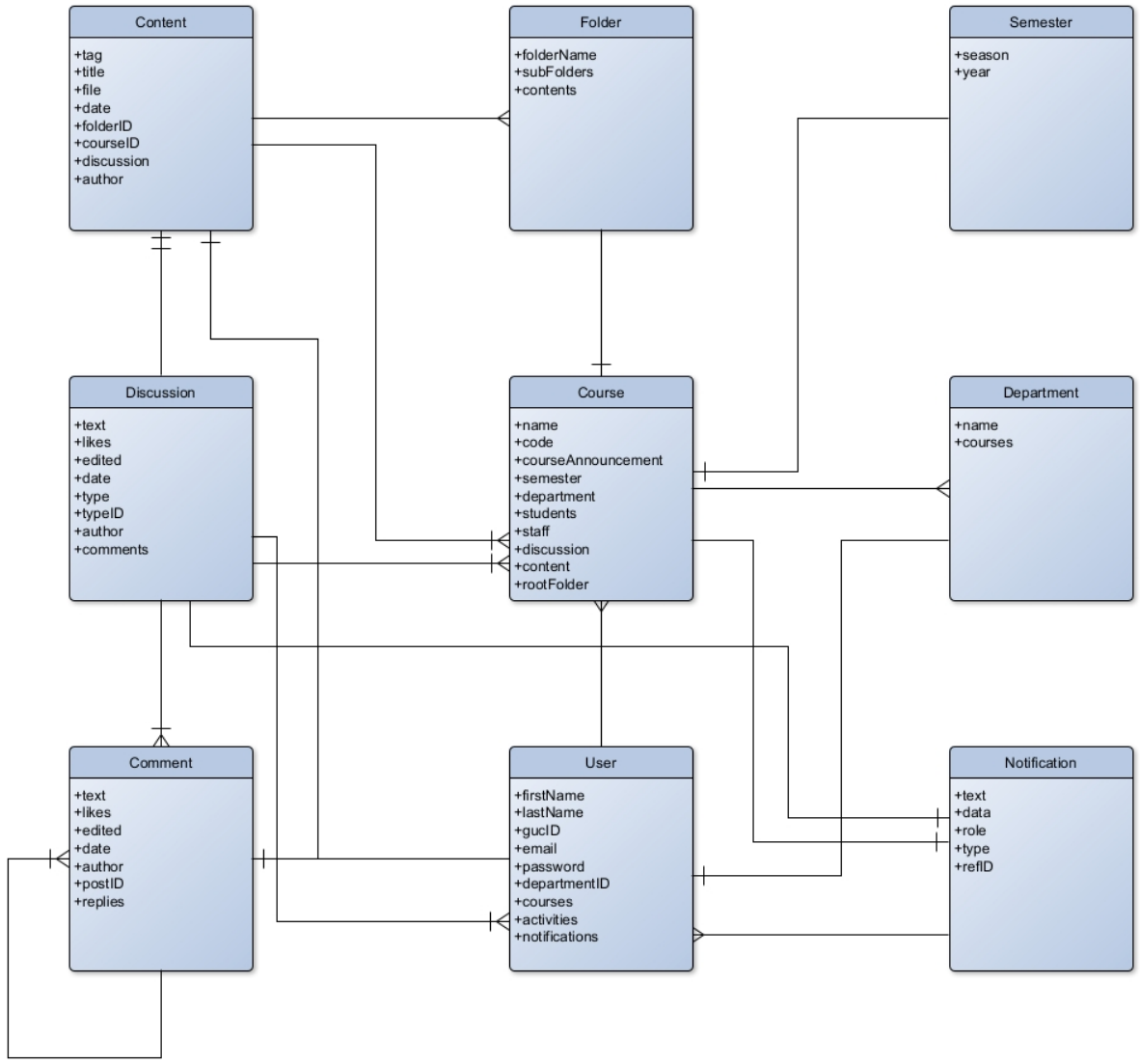


Figure 3.3: Database Entity Relationship Diagram (ERD)

Chapter 4

Implementation

This chapter will discuss in details the implementation of the system and how all the components and the database work together to improve the new features of the new website. Also this chapter will mention the languages, tools, packages, and technologies that were used to implement the new website.

4.1 Tools and technologies

To implement the new website and combine all the features and ideas that were discussed in the previous chapters I decided to work using MERN stack which stands for MongoDB, Express, React and Nodejs. **MongoDB** is an open-source, document-based database, **Express** is a web application framework for Nodejs known for its speed, **React** is a front end JavaScript library for creating user interfaces and finally **Nodejs** is a JavaScript run-time environment that executes JavaScript code outside of the browser(such as a server). And the reason for choosing MERN stack is that working in the MERN stack delivers powerful results simply and efficiently. The MERN stack's main advantage in web development is that each line of code is written in JavaScript, which is a nearly universal programming language since it is vital for both server-side and client-side code. By using a single programming language, the MERN stack eliminates the need for context switching and greatly simplifies the entire development process, giving web developers the tools to create efficient web applications with far less effort.

4.1.1 MERN Stack

1. **MongoDB**

   According to Mongo's homepage, "it's the database for modern application". It's a document database and commonly used in combination with Nodejs.

   And before diving into what MongoDB is let's first differentiate between the two types of database in the world Structured Query Language (SQL) and not only SQL - non-relational database (NoSQL) database.

   - **SQL Database**

     SQL database stands for Structured Query Language. It can be used while interacting with relational databases where data is stored in tables that have fixed columns and rows. SQL databases rose in popularity in the early 1970s. At the time, storage was extremely expensive, so software engineers normalized their databases in order to reduce data duplication. Software engineers in the 1970s also commonly followed the waterfall software development model which means projects should be planned in details and schema must be defined before adding anything into the database. Everything has to conform to a pattern. Software engineers painstakingly created complex entity-relationship (ER) diagrams to ensure they had carefully thought through all the data they would need to store. Due to this upfront planning model, software engineers struggled to adapt if requirements changed during the development cycle. As a result, projects frequently went over budget, exceeded deadlines and failed to deliver against user needs. And there are lots of very popular SQL databases, including things like MySQL, Postgres, Oracle, Microsoft SQL Server, SQLite.

Customer_id	Name	Address	Age
1	Billie	NY	22
2	Eilish	London	19
3	Ariana	Miami	18
4	Selena	New Jersey	32
5	Kety	Hawaii	42
6	Adele	Miami	29

Figure 4.1: SQL table example

- **NoSQL Database**
  When people use the term "NoSQL database", they typically use it to refer to any non-relational database. Some say the term "NoSQL" stands for "non SQL" while others say it stands for "not only SQL." Either way, most agree that NoSQL databases are databases that store data in a format other than relational tables. NoSQL databases are non tabular, and store data differently than relational tables. NoSQL databases come in a variety of types based on their data model.Unlike SQL databases there are many types of NoSQL databases like document type, key-value and graph. They provide flexible schemas and scale easily with large amounts of data and high user loads. NoSQL databases emerged in the late 2000s as the cost of storage dramatically decreased. Gone were the days of needing to create a complex, difficult-to-manage data model simply for the purposes of reducing data duplication. Developers (rather than storage) were becoming the primary cost of software development, so NoSQL databases optimized for developer productivity. Additionally, the Agile Manifesto was rising in popularity, and software engineers were rethinking the way they developed software. They were recognizing the need to rapidly adapt to changing requirements. They needed the ability to iterate quickly and make changes throughout their software stack—all the way down to the database model. NoSQL databases gave them this flexibility. In fact, when compared with SQL databases, many find modeling relationship data in NoSQL databases to be easier than in SQL databases, because related data doesn't have to be split between tables. NoSQL data models allow related data to be nested within a single data structure. And there are very popular NoSQL databases,like MongoDB, Couch DB, Neo4j, Cassandra and Redis.[4]

Figure 4.2: Document Type Example

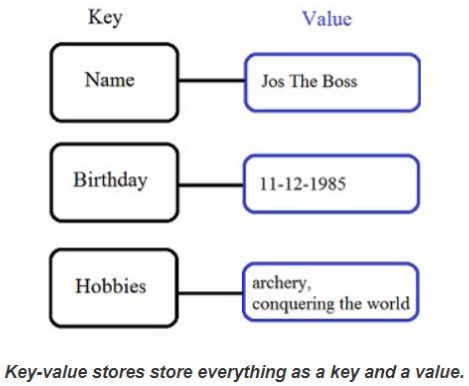


Figure 4.3: Key-Value Type Example

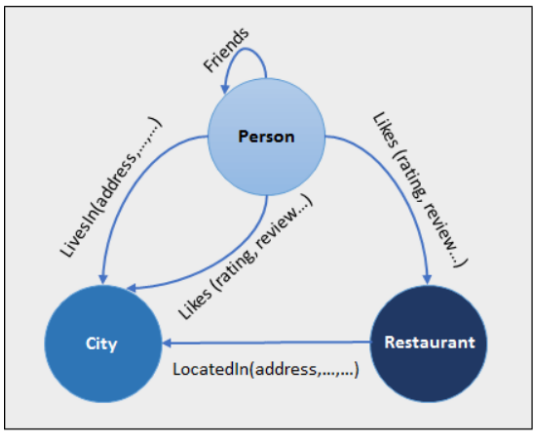


Figure 4.4: Graph Type Example

Now after discussing the difference between SQL and NoSQL databases, it's easier to talk about what MongoDB is. **MongoDB is a document database** designed for ease of development and scaling. A record in MongoDB is a document, which is a data structure composed of field and value pairs. MongoDB documents store data in documents similar to JavaScript Object Notation (JSON) (JavaScript Object Notation) objects. Each document contains pairs of fields and values. The values can typically be a variety of types including things like strings, numbers, booleans, arrays, or objects, and their structures typically align with objects developers are working with in code. Because of their variety of field value types and powerful query languages, document databases are great for a wide variety of use cases and can be used as a general purpose database. They can horizontally scale-out to accommodate large data volumes. The values of fields may include other documents, arrays, and arrays of documents which means the content, size, and number of fields in the documents can differ from one another and the data structure can be changed over time. And these embedded documents and arrays can reduce the need for expensive joins like in SQL databases. MongoDB framework is best known for its flexible and scalable features.[5]

Figure 4.5: MongoDB document

2. **Express**

Express is a fast, unopinionated, minimalist web framework for Nodejs that provides a robust set of features for web and mobile applications. It's a backend framework that simplifies the task of writing server code. For developers, there is no need to repeat the same code, as did earlier with the Nodejs Hypertext Transfer Protocol (HTTP) module. Express has lots of advantages, it's asynchronous and single-threaded. It's very easy and makes handling requests very simple using much less code. Express helps start up the server to listen for requests. Also it parse incoming request form string text to objects. It also helps match those requests to their particular routes so we can have different code and different functions that run depending on what that request is requesting for example route "/home" has a different functionality from the route "/about" so Express helps us do so. Also it helps deal with the response set the status code, set the content that we're responding with headers and so on. [6]

3. **React**

React is a JavaScript library created for building fast and interactive user interfaces for web and mobile applications. It is an open-source, component-based, front-end library responsible only for the application's view layer. In Model View Controller (Model View Controller (MVC)) architecture, the view layer is responsible for how the app looks and feels. React was developed by Facebook developers. React makes it easier to create dynamic web applications because it requires less coding and offers more functionality, as opposed to JavaScript, where coding often gets complex very quickly. React also uses **virtual Document Object Model (DOM)**, thereby creating web applications faster. But before diving into virtual DOM and what is it let's talk about real DOM. **DOM** stands for Document Object Model which is a programming interface for Hypertext Markup Language (HTML) and XML documents. It represents the page so that programs can manipulate the document structure and change its style or contents. DOM treats an XML or HTML document as a tree structure in which each node is an object representing a part of the document.

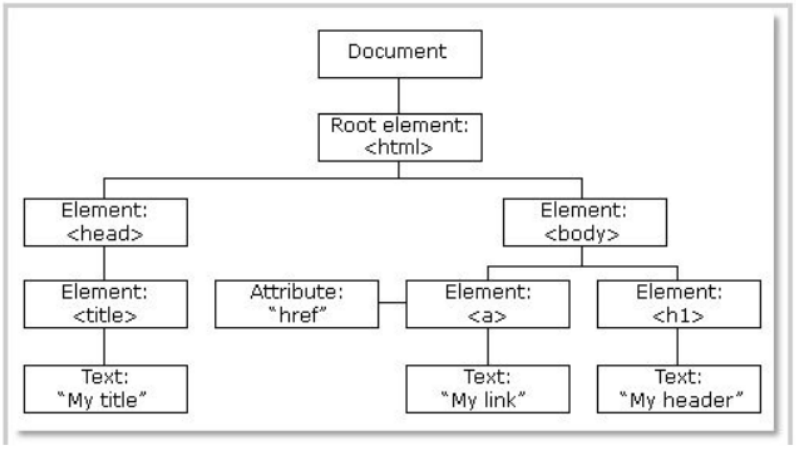


Figure 4.6: DOM of a web page

Now, React keeps a lightweight representation of the real DOM in the memory, and that is known as the **virtual DOM**. Manipulating real DOM is much slower than manipulating virtual DOM because nothing gets drawn on the screen. When the state of an object changes in a React application, virtual DOM gets updated. It then compares its previous state and then updates only those objects in the real DOM that were changed instead of updating all of the objects. This makes things move fast, especially when compared to other front-end technologies that have to update each object even if only a single object changes in the web application.

React uses **JavaScript XML (JSX)** syntax which stands for JavaScript XML. It's a syntax extension to JavaScript. It is used with React to describe what the user interface should look like. By using JSX, we can write HTML structures in the same file that contains JavaScript code. This makes the code easier to understand and debug, as it avoids the usage of complex JavaScript DOM structures. It also allows writing expression between . The expression can be any JavaScript expression, variable or function. For example this code can be written in React using JSX syntax,

```
const name = "Ahmed"
const greetMessage = <h1>Hello, {name}</h1>
```

Figure 4.7: JSX syntax example

JSX is an expression too as after compilation, JSX expressions become regular JavaScript function calls and evaluate to JavaScript objects. This means that JSX can be used inside of "if statements" and "for loops", assign it to variables, accept it as arguments, and return it from functions for example:

```
function getGreeting(user) {
    if (user === "Ahmed") {
      return <h1>Hello, {user}!</h1>;
    }
    return <h1>Hello, Stranger.</h1>;
  }
  getGreeting("Ahmed")
```

Figure 4.8: JSX expression example

React also uses **components** which are the building blocks of any React application, and a single app usually consists of multiple components. Components let the developer split the User Interface into independent, reusable pieces, and think about each piece in isolation. These components have their logic and controls, and they have the term "reusable components" which means they can be reused throughout the application, which in turn reduces the application's development time. Basically, components are like a JavaScript function that returns a JSX single element describing what should appear on the screen and they accept arbitrary inputs called "props" stands for properties which is an object contains data.

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

Figure 4.9: Functional Component example

Now this Welcome component can be treated as HTML tags and used as elements in DOM. And here is an example of how Welcome component can be rendered.

```
function App() {
  return (
    <div>
      <Welcome name="Sara" />
      <Welcome name="Cahal" />
      <Welcome name="Edite" />
    </div>
  );
}
```

Figure 4.10: Render Functional Component example

React follows a unidirectional data flow. This means that when designing a React app, developers often nest child components within parent components. Since the data flows in a single direction, it becomes easier to debug errors and know where a problem occurs in an application at the moment in question.[7]

4. **Nodejs**

   Nodejs is a JavaScript run-time environment that executes JavaScript code outside of the browser (such as a server). It's built on Google Chrome and open-sourced by Google. The framework is built on Chrome's JavaScript engine. All Application Programming Interface (API)s of Node.js library are asynchronous and it's very fast in code execution. The server simply never waits for an API to return data. It works great when enclosed with HTML page instead of using its module. [8]

The combination of technologies in MERN Stack makes it ideal for developing a wide variety of apps. It enables fast and easy deployment of apps also delivers powerful results simply and efficiently.

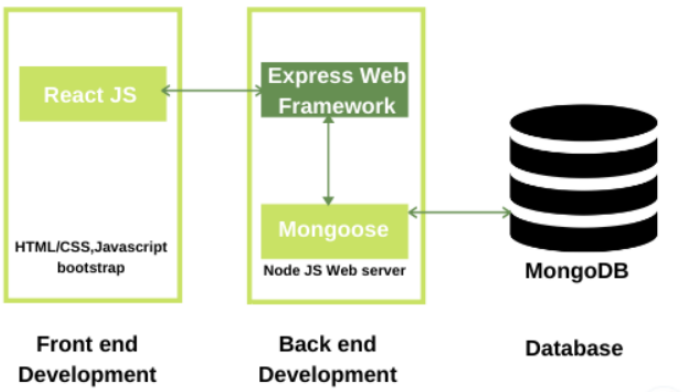


Figure 4.11: MERN stack

4.1.2 Installed Packages

Packages simply are a code written by other developers to make our life easier. This subsection will discuss the installed packages that were used in this project, what is the purpose of installing them and how they work.

- **express**
  As discussed above express is a fast, unopinionated, minimalist web framework for node. It provides small, robust tooling for HTTP servers, making it a great solution for single page applications, web sites, hybrids, or public HTTP APIs. Express helps start up the server to listen for requests. Express has routing methods (get, post, patch, delete). Routing refers to how an application's endpoints (URIs) respond to client requests. The following example respond with "Hello World" when a get request is made to home page.

```
const express = require('express')
const app = express()

app.get('/', function (req, res) {
  res.send('Hello World')
})

app.listen(3000)
```

Figure 4.12: express routing example

- **mongoose**
  Mongoose is an Object Data Mapper or an Object Data Modeling (ODM) library for MongoDB and Nodejs. It's like a driver that connects Nodejs to MongoDB. Object Data Mapper simply means it maps data into objects. It maps the data that comes back from MongoDB or the data that we want to insert into MongoDB into these usable JavaScript objects that we can access easily and add methods to. It also manages relationships between the data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB.[9]

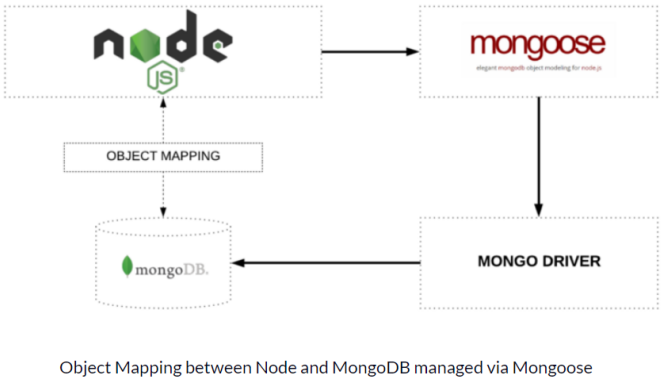


Figure 4.13: Mongoose Object Mapping

And to create Model in MongoDB using mongoose we must go through four easy steps.

1. First we need to require mongoose library.
2. Second we need to connect to our Mongo database using .connect() method which takes mongodb URL as a parameter.

```
// getting-started.js
const mongoose = require('mongoose');
mongoose.connect('mongodb://localhost:27017/test', {useNewUrlParser: true, useUnifiedTopology: true});
```

Figure 4.14: Getting start with mongoose

3. Third we need to define our Schema using .Schema() method.

```
import mongoose from 'mongoose';
const { Schema } = mongoose;

const blogSchema = new Schema({
  title:  String, // String is shorthand for {type: String}
  author: String,
  body:   String,
  comments: [{ body: String, date: Date }],
  date: { type: Date, default: Date.now },
  hidden: Boolean,
  meta: {
    votes: Number,
    favs:  Number
  }
});
```

Figure 4.15: Defining schema using mongoose

4. Finally we can create our Model using .model() method which takes the model name and schema defined in step3 as a parameters.

```
const Blog = mongoose.model('Blog', blogSchema);
// ready to go!
```

Figure 4.16: Creating Model using mongoose

After doing those four steps we successfully managed to create model in our MongoDB and now we can insert or retrieve data from our model using mongoose CRUD methods like .findByID() or findOneAndUpdate() and so on.

- **joi**
  Joi is the most powerful schema description language and data validator for JavaScript. It's a validation library that allows to build schemas to validate JavaScript objects. It also provides methods to easily validate common data types, like strings, numbers or booleans also it can validate more constrains such as this field is required and has min length of 5 characters for example. [10]

```
Joi.object({
    username: Joi.string()
        .alphanum()
        .min(3)
        .max(30)
        .required(),

    password: Joi.string()
        .pattern(new RegExp('^[a-zA-Z0-9]{3,30}$')),

    birth_year: Joi.number()
        .integer()
        .min(1900)
        .max(2013),

    email: Joi.string()
        .email({ minDomainSegments: 2, tlds: { allow: ['com', 'net'] } })
})
```

Figure 4.17: Joi Schema

Usage is a two steps process:

1. First, a schema is constructed using the provided types and constraints such as the above example
2. Second, the passed object is validated against the defined schema. And the result is an object with two keys value and error. If the input is valid, then the error will be undefined. But if the input is invalid, error is assigned a ValidationError object providing more information.

```
schema.validate({ username: 'abc', birth_year: 1994 });
// -> { value: { username: 'abc', birth_year: 1994 } }

schema.validate({});
// -> { value: {}, error: '"username" is required' }
```

Figure 4.18: Joi validation

- **moment.js**
  Moment.js is a JavaScript date library for parsing, validating, manipulating, and formatting dates. And this package is used in this project to format the date in a pretty nice way.[11]

```
moment.locale();         // en
moment().format('LT');   // 10:32 PM
moment().format('LTS');  // 10:32:32 PM
moment().format('L');    // 07/25/2021
moment().format('l');    // 7/25/2021
moment().format('LL');   // July 25, 2021
moment().format('ll');   // Jul 25, 2021
moment().format('LLL');  // July 25, 2021 10:32 PM
moment().format('lll');  // Jul 25, 2021 10:32 PM
moment().format('LLLL'); // Sunday, July 25, 2021 10:32 PM
moment().format('llll'); // Sun, Jul 25, 2021 10:32 PM
```

Figure 4.19: moment format methods

- **bcryptjs**
  Bcryptjs is a JavaScript library for hashing passwords instead of saving them as a plain text. The way it works is, first the password will hashed using hashing function.[12]

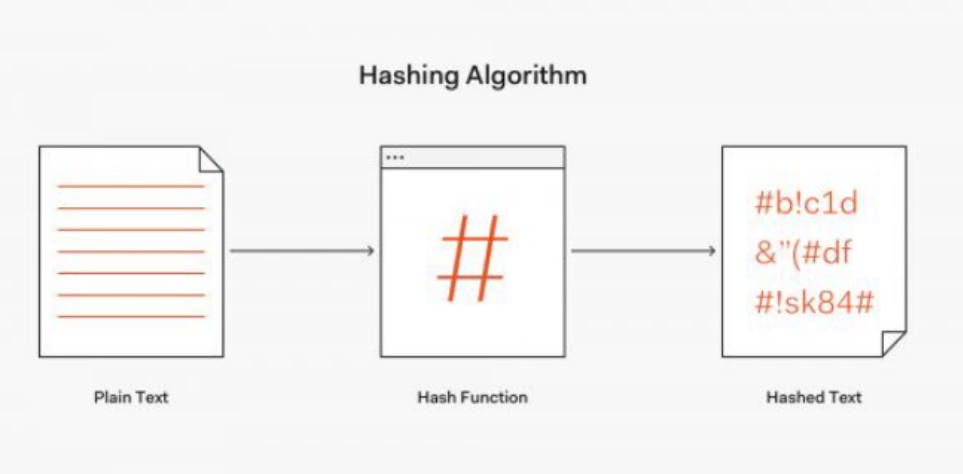


Figure 4.20: Hashing passwords

But the problem is same passwords will produce same hashes! So the solution for this problem is that a randomly generated text (called salt) will be generated first, then this salt will be appended to real password, after that both real password and salt will be hashed together using the hashing function.

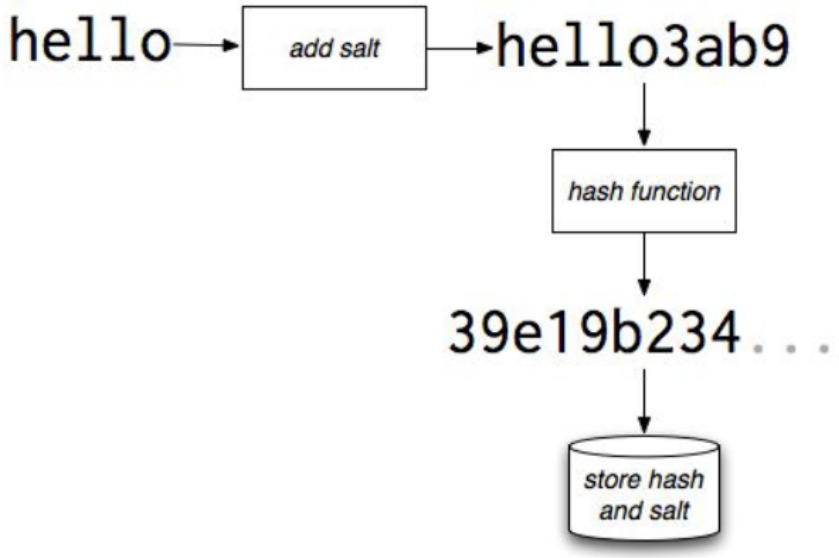


Figure 4.21: Hashing password and salt together

And this how bcryptjs helps us using .genSalt() and .hash() methods to generate salt and hash passwords.

```
const salt = await bcrypt.genSalt();
const hashPassword = await bcrypt.hash(password, salt);
```

Figure 4.22: Hashing password using bcryptjs

And there are two different users in MongoDB with and without hashing their passwords.

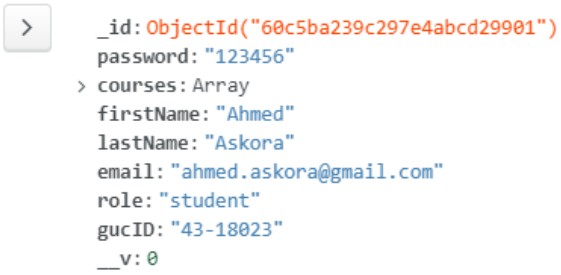


Figure 4.23: user password as a plain text example

Figure 4.24: user hashed password example

- **json web token(jwt)**
  JSON Web Token (JSON Web Token (JWT)) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed.[13] Json Web Tokens are useful in authorization. Once the user is logged in, each request will include the JWT in the header of that request, allowing the user to access routes, services, and resources that are permitted with that token. JSON Web Tokens consist of three parts separated by dots which are header, payload and signature. The most important part and we care about is the second part which is the payload or the data assigned to the user when he/she logs in. And this is an example of how we assign a token containing a payload of id and role of the user and this token is signed to the header of the request.

```
req.header.token = jwt.sign({ id: foundUser._id, role: foundUser.role },
    process.env.tokenSecret);
```

Figure 4.25: assign a token example

When the user tries to access any route of the website, with the help of that token we verify that if this user is authorized to use this route or not. And this is an example of verifying a token of a user trying to access a route of an admin.

```
const token = req.header("token")
if (!token)
    return res.status(401).send("please log in first");

try {
    const verified = jwt.verify(token, process.env.tokenSecret)
    if (verified.role !== "admin")
        return res.status(401).send("Access Denied You're NOT an Admin");
    req.user = verified
```

Figure 4.26: verifying a token example

- **multer**
  Multer is a Nodejs middleware for handling multipart/form-data, which is primarily used for uploading files. [14] This package is used to help instructors upload contents to the system. The way it works is very simple. Multer adds a body object and a file or files object to the request object. The body object contains the values of the text fields of the form, the file or files object contains the files uploaded via the form. And this is an example of uploading a single file.

```
var express = require('express')
var multer  = require('multer')
var upload = multer({ dest: 'uploads/' })

var app = express()

app.post('/profile', upload.single('avatar'), function (req, res, next) {
  // req.file is the `avatar` file
  // req.body will hold the text fields, if there were any
})
```

Figure 4.27: uploading file using multer

- **Bootstrap**
  Bootstrap is one of the most popular front-end framework used to create modern and responsive websites and web apps. It's open-source and free to use. Bootstrap is great for creating layouts, as its responsive Cascading Style Sheet (CSS) is designed to conform to different devices. It can be employed to ensure consistency, eliminate cross-browser issues, and so on.[15]

- **Material-UI**
  Material-UI is a library that allows developers to import and use different components to create a user interface in our React applications. This saves a significant amount of time since the developers do not need to write everything from scratch.[16]

4.2 System Implementation

After we discussed MERN stack technology and mentioned packages and libraries we installed to help in this project, this section will discuss the actual implementation of the system and how all these technologies combine together to get the final result of the Improved CMS.

4.2.1 Server Side

This subsection will discuss the implementation of server side or in other words the backend of the system, also will mention the schemas and models of the database.

As mentioned above, first of all we required express package to start up the server to listen for incoming requests, after that we required mongoose library to connect Nodejs to MongoDB and helps us define schemas and create models. And here are the schemas and models of the new website.

- **Department**

```
const departmentSchema = mongoose.Schema({
    name: {
        type: String,
        required: true,
        unique: true
    },
    courses: [{ type: mongoose.Types.ObjectId, ref: "Course" }],
});
```

(a) Department Schema

```
_id: ObjectId("60b89653513a493214cb9213")
courses: Array
  0: ObjectId("60c5bb45cf8bd0281c189758")
  1: ObjectId("60d2de1d00f73a46c06ee72b")
  2: ObjectId("60dc844c4e5365089c56a491")
  3: ObjectId("60dc8628a16e7035d47e8177")
  4: ObjectId("60e4a0f992fc1f34e8910646")
  5: ObjectId("60e6b98dc879bb47346ecc1c")
  6: ObjectId("60e6ba7a67d84c23a046409b")
name: "MET"
__v: 0
```

(b) Department Model

- **Semester**

```
const semesterSchema = mongoose.Schema({
    season: String,
    year: Number,
});
```

(a) Semester Schema

```
_id: ObjectId("60ba92707ccd4c2224f04bae")
season: "Spring"
year: 2021
__v: 0
```

(b) Semester Model

- **Course**

```
const courseSchema = mongoose.Schema({
    name: String,
    code: {
        type: String,
        unique: true,
    },
    courseAnnouncement: [String],
    department: {
        type: mongoose.Types.ObjectId,
        ref: "Department",
        required: true,
    },
    semester: {
        type: mongoose.Types.ObjectId,
        ref: "Semester",
        required: true,
    },
    students: [{
        type: mongoose.Types.ObjectId,
        ref: "User",
        // required: true,
    }],
    staff: [{
        type: mongoose.Types.ObjectId,
        ref: "User",
        // required: true,
    }],
    content: [{
        type: mongoose.Types.ObjectId,
        ref: "Content"
    }],

    discussion: [{ type: mongoose.Types.ObjectId, ref: "Discussion" }],
    rootFolder: {
        type: mongoose.Types.ObjectId,
        ref: "Folder"
    },
```

(a) Course Schema


```
_id: ObjectId("60dc8628a16e7035d47e8177")
students: Array
    0: ObjectId("60d963378868462a2445eaf9")
staff: Array
    0: ObjectId("60d9639e8868462a2445eafa")
content: Array
discussion: Array
    0: ObjectId("60e46b801324116154704fdc")
    1: ObjectId("60e486f0ba7c3743c88426ac")
    2: ObjectId("60e70fd39c71810f482a7080")
    3: ObjectId("60ec602789861a2ab8791702")
    4: ObjectId("60ec61d4c070692ea0d906f4")
    5: ObjectId("60ed8a25c148b04fb0c4b8d1")
    6: ObjectId("60ed8e53481c5465909133e9")
    7: ObjectId("60edef777ffb4a5e34110119")
    8: ObjectId("60edefef7ffb4a5e3411011d")
    9: ObjectId("60fae334981ac24b04571d71")
name: "Digital Logic Design"
code: "CSEN201"
department: ObjectId("60b89653513a493214cb9213")
semester: ObjectId("60ba92707ccd4c2224f04bae")
rootFolder: ObjectId("60dc8628a16e7035d47e8176")
__v: 0
courseAnnouncement: Array
    0: "<p><span class="ql-size-large">Dear All</span></p><p><span class="ql-s..."
    1: "Quiz 3 is held in H15"
    2: "Assignment 3 is postponed"
    3: "QUIZ 2 is on 20/9"
```


(b) Course Model

- **Content**

```
const contentSchema = mongoose.Schema({
    tag: {
        type: String,
        required: true,
    },
    title: {
        type: String,
        required: true,
    },
    folderID: {
        type: mongoose.Types.ObjectId,
        ref: "Folder",
        required: true,
    },
    courseID: { type: mongoose.Types.ObjectId, ref: "Course", },
    discussion: [{ type: mongoose.Types.ObjectId, ref: "Discussion" }],
    file: Object,
    date: {
        type: String,
        default: moment().format('llll')
    },
    author: {
        type: mongoose.Types.ObjectId,
        ref: "User",
    },
});
```

(a) Content Schema


```
_id: ObjectId("60f00362af2495765c0a6b37")
discussion: Array
    0: ObjectId("60f003d1af2495765c0a6b39")
date: "Thu, Jul 15, 2021 10:07 AM"
author: ObjectId("60d9639e8868462a2445eafa")
folderID: ObjectId("60f00323af2495765c0a6b36")
tag: "#lecture1, #java, #classes, #inheritance"
title: "Lecture 1"
courseID: ObjectId("60e6ba7a67d84c23a046409b")
file: Object
    fieldname: "file"
    originalname: "Improved CMS System Ch1.pdf"
    encoding: "7bit"
    mimetype: "application/pdf"
    destination: "D:/GUC/bachelor/client/public/uploads/"
    filename: "Improved CMS System Ch1-1626342242313.pdf"
    path: "D:\GUC\bachelor\client\public\uploads\Improved CMS System Ch1-16263422..."
    size: 65408
__v: 0
```


(b) Content Model

- **Folder**

```
const folderSchema = mongoose.Schema({
    folderName: String,
    subFolders: [{ type: mongoose.Types.ObjectId, ref: "Folder" }],
    contents: [{ type: mongoose.Types.ObjectId, ref: "Content" }],
});
```

(a) Folder Schema

(b) Folder Model

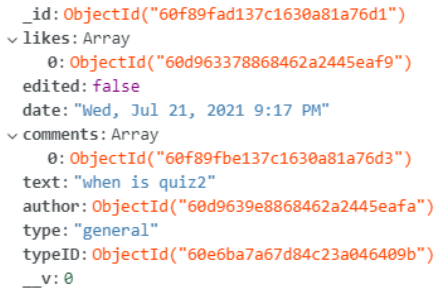
- **Discussion**

```
const discussionSchema = mongoose.Schema({
    author: {
        type: mongoose.Types.ObjectId,
        ref: "User",
    },
    text: {
        type: String,
        required: true
    },
    type: {
        type: String,
        enum: ["general", "content"]
    },
    //type == "general" ? typeID = courseID : typeID = contentID
    typeID: { type: mongoose.Schema.Types.ObjectId },

    // files: Object,
    likes: [{
        type: mongoose.Types.ObjectId,
        ref: "User",
    }],
    edited: {
        type: Boolean,
        default: false
    },
    date: {
        type: String,
        default: moment().format('llll')
    },
    comments: [{ type: mongoose.Schema.Types.ObjectId, ref: "Comment" }],

});
```

(a) Discussion Schema



(b) Discussion Model

- **Comment**

```
const commentSchema = mongoose.Schema({
    author: {
        type: mongoose.Types.ObjectId,
        ref: "User",
    },
    text: {
        type: String,
        required: true
    },
    // files: Object,
    likes: [{
        type: mongoose.Types.ObjectId,
        ref: "User",
    }],
    date: {
        type: String,
        default: moment().format('llll')
    },
    edited: {
        type: Boolean,
        default: false
    },
    postID: {
        type: mongoose.Schema.Types.ObjectId,
        ref: "Discussion",
        required: true
    },
    replies: [{
        type: mongoose.Schema.Types.ObjectId,
        ref: "Comment",
    }]

});
```

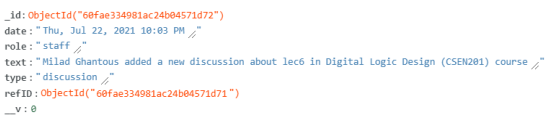
(a) Comment Schema

(b) Comment Model

- **Notification**

```
const notificationSchema = mongoose.Schema({
    text: {
        type: String,
        required: true,
    },
    role: {
        type: String,
        required: true,
    },
    type: {
        type: String,
        enum: ["course", "discussion"]
    },
    date: {
        type: String,
        default: moment().format('llll')
    },
    refID: { type: mongoose.Types.ObjectId },
});
```

(a) Notification Schema



(b) Notification Model

- **User**

```
const userSchema = mongoose.Schema({
  firstName: String,
  lastName: String,
  gucID: {
    type: String,
    unique: true,
  },
  email: {
    type: String,
    unique: true,
  },
  password: {
    type: String,
    default: "123456",
  },
  role: String,
  courses: [{ type: mongoose.Types.ObjectId, ref: "Course" }],
  activities: [{ type: mongoose.Types.ObjectId, ref: "Discussion" }],
  departmentID: {
    type: mongoose.Types.ObjectId,
    ref: "Department",
  },

  notifications: [{
    type: mongoose.Types.ObjectId,
    ref: "Notification",
  }],
});
```

(a) User Schema

```
_id: ObjectId("60d963378868462a2445eaf9")
password: "$2a$10$C85La/WL8vTQIJhIVuIXuOOhz4ZhVlLo.d8pr494kRR6InMNf3yqa"
courses: Array
  0: ObjectId("60dc8628a16e7035d47e8177")
  1: ObjectId("60e4a0f992fc1f34e8910646")
  2: ObjectId("60e6ba7a67d84c23a046409b")
firstName: "ahmed"
lastName: "ahmed"
email: "ahmed.ahmed@gmail.com"
role: "student"
gucID: "123123"
__v: 0
activities: Array
  0: ObjectId("60e46b801324116154704fdc")
  1: ObjectId("60e486f0ba7c3743c88426ac")
  2: ObjectId("60e4a19d5a7548665c4b2f4a")
  3: ObjectId("60e70fd39c71810f482a7080")
  4: ObjectId("60ed8e53481c5465909133e9")
  5: ObjectId("60edea7f82044b1ea4cebbc4")
  6: ObjectId("60edefef7ffb4a5e3411011d")
  7: ObjectId("60edef777ffb4a5e34110119")
  8: ObjectId("60f8a0b2137c1630a81a76d7")
notifications: Array
  0: ObjectId("60edeccb1a67bd5d78cc2c2b")
  1: ObjectId("60ededf54eae9f394816c93d")
  2: ObjectId("60edee3f78b463590c0e7a79")
  3: ObjectId("60edee733abdcd6d68db6714")
  4: ObjectId("60edef577ffb4a5e34110118")
  5: ObjectId("60edef777ffb4a5e3411011a")
  6: ObjectId("60f00363af2495765c0a6b38")
  7: ObjectId("60f003d2af2495765c0a6b3a")
```

(b) User Model

4.2.2 Client Side

This subsection will discuss the implementation of client side or in other words the front-end, also it will discuss the new features of the system.

The system consists of five main features which are dashboard, discussion, announcements, notifications and search.

- **Dashboard**
  As it's a CMS the main feature is the dashboard. The dashboard shows all the courses assigned to students or assigned to instructors. But in the new system the instructor has the ability to upload the course contents in folders instead being uploaded in on page. This folder feature overcomes the problem of scrolling too much that the students face while using the current CMS. Also organizing the contents into folders makes it more easier for the students to find the contents they want.

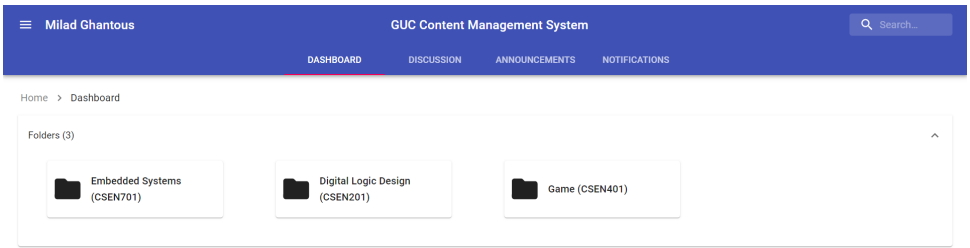


Figure 4.37: all courses

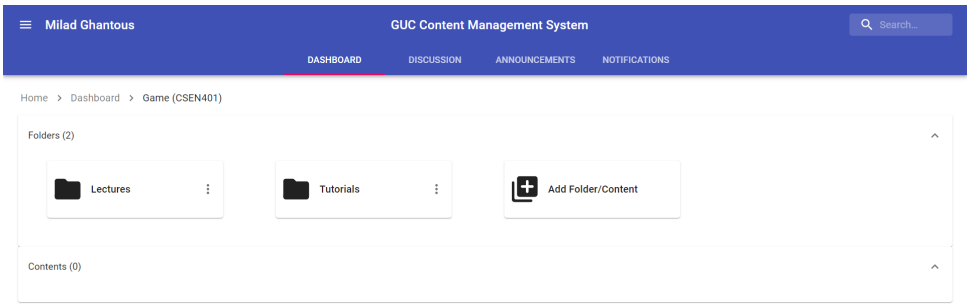


Figure 4.38: course folders example

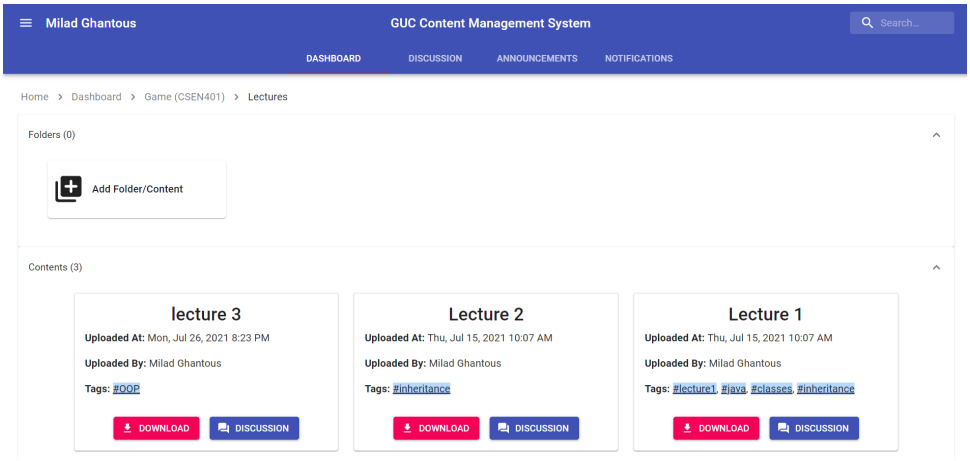


Figure 4.39: contents of folder example

One of the main instructors functionality is to add a new folder or upload a new content which they can do it easily by clicking on "Add Folder/Content" button beside the folders.

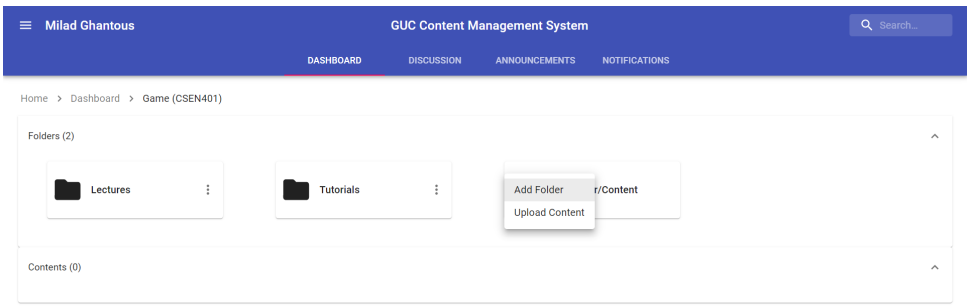


Figure 4.40: Add Folder/Content

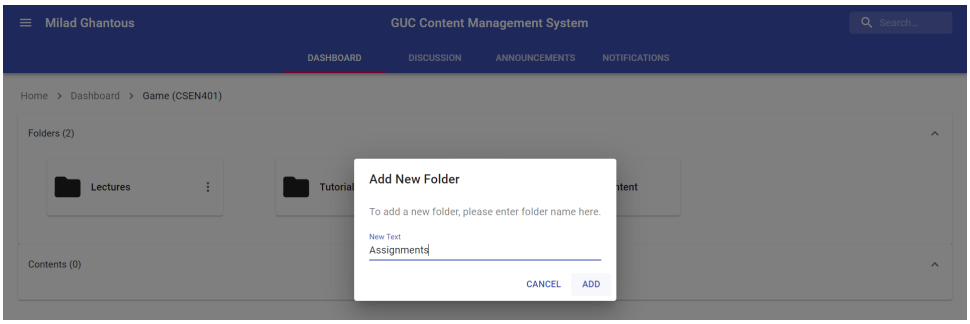


Figure 4.41: Add new folder

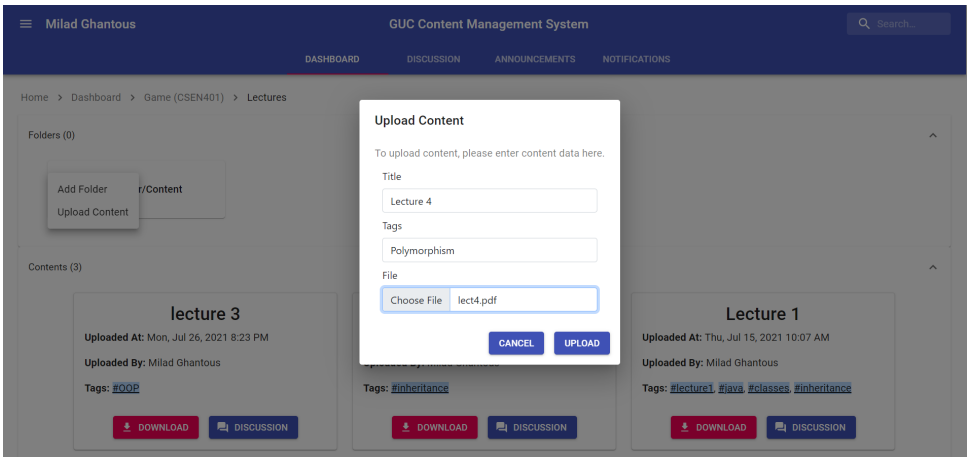


Figure 4.42: Upload content

- **Discussion**
  The second feature is discussion. Discussion feature is totally improved to user friendly design. With discussion the student and the instructor can interact virtually with each other. The students can use this feature to ask about any thing they think about. There are two types of discussion, content discussion and general discussion. For each uploaded content in the system there is a "discussion" button, when this button is clicked, it shows all discussions about this specific content also students can post a new discussion about this content.

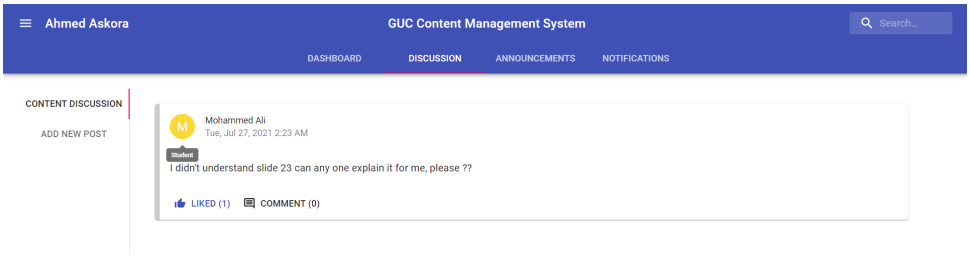


Figure 4.43: all discussions of a content

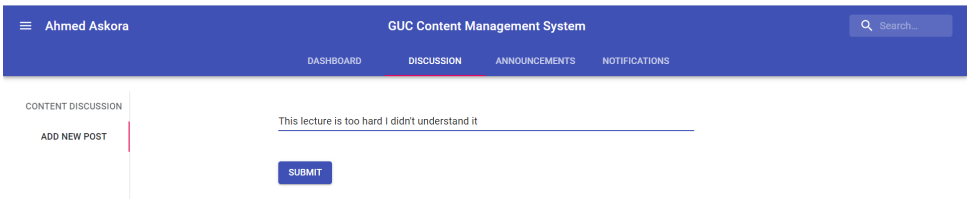


Figure 4.44: Add discussion to a content

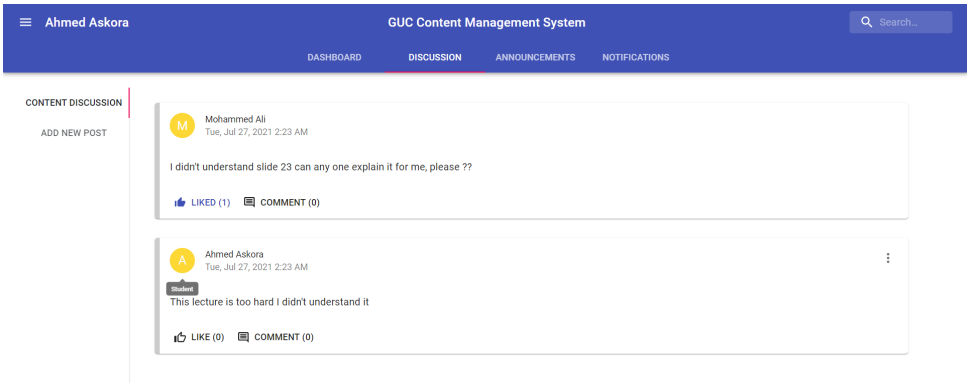


Figure 4.45: Add discussion to a content

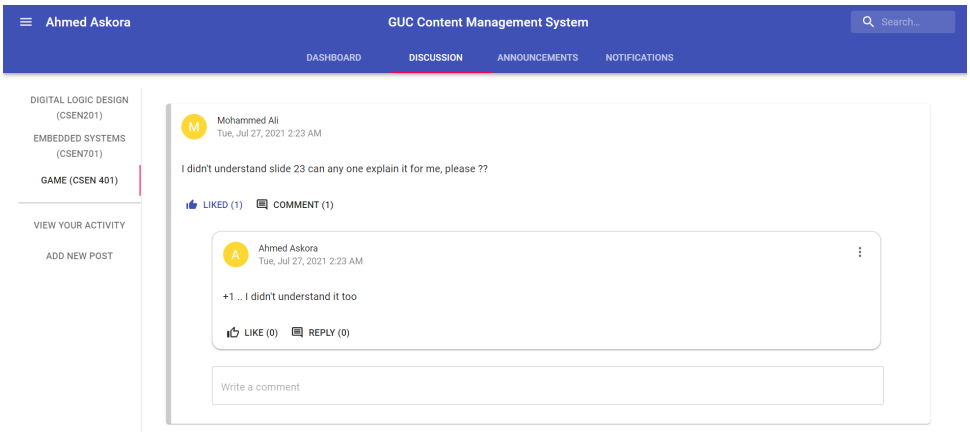


Figure 4.46: Add comment to a discussion

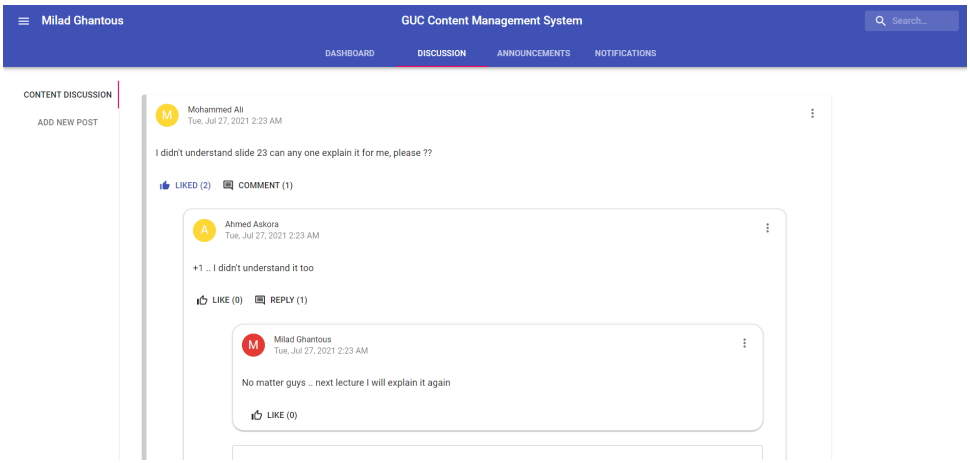


Figure 4.47: Reply to a comment

The other type of discussion is course discussion. By clicking on discussion button in the navigation bar it shows all discussion separated by each course. The student can post a course discussion to ask about any thing related to the course in general for example asking about the time of the next quiz or the deadline of an assignment. This can help all the students to communicate with their instructors instead of sending an email to the instructor using GUC mail which wastes more time but this way is more efficient and more user-friendly.

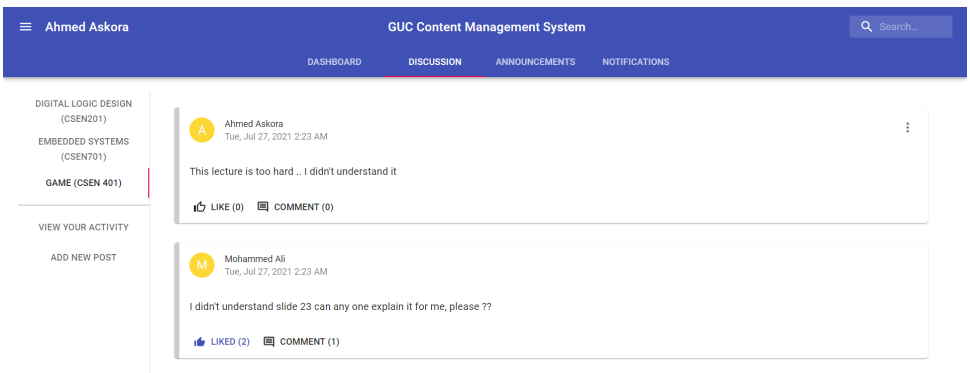


Figure 4.48: all discussions of a course

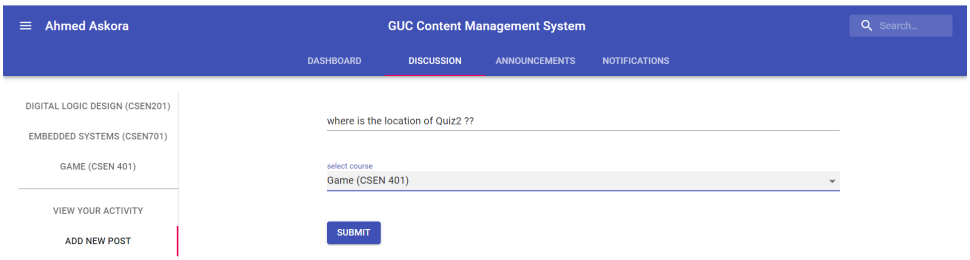


Figure 4.49: add a course discussion

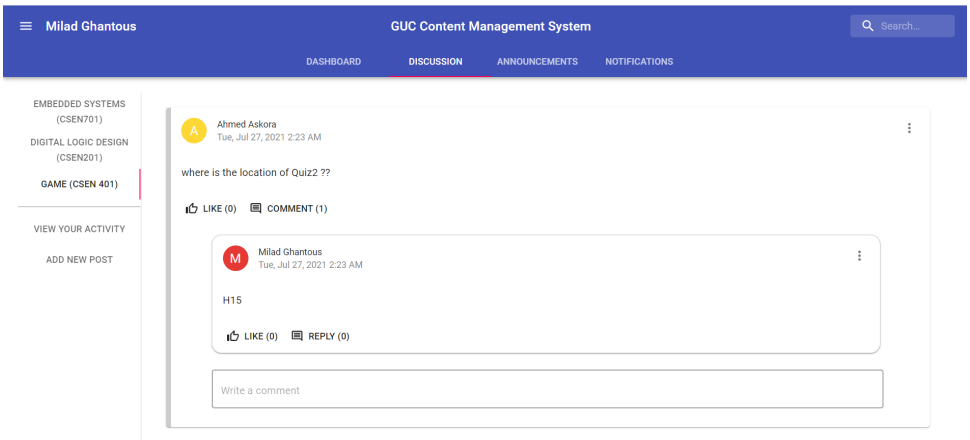


Figure 4.50: reply to course discussion

- **Announcements**
  The third main feature is announcements. Instructors can use this feature to announce any thing to their students like grading schema, timing of a quiz or content of midterm exam for example. This feature is like course announcement in current website but it has been improved to have its own module to show its importance, also it includes all announcements of all courses in one module to help students memorize important announcements and avoid distraction. This feature also can reduce the using of GUC mail which is annoying for most of the students and instructors and hard to keep track of all received e-mails.

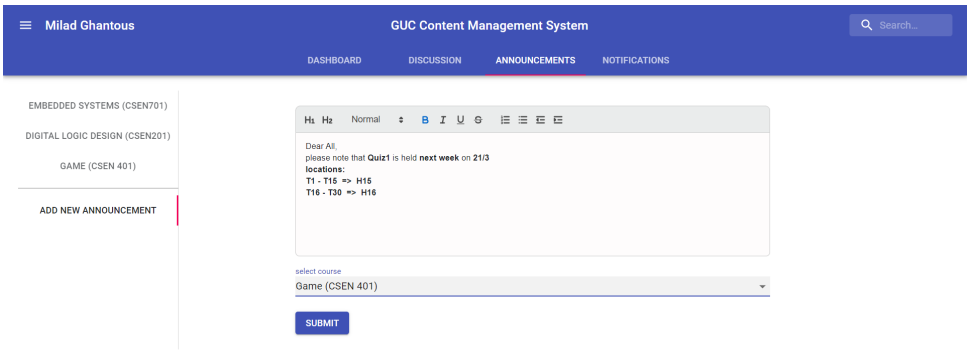


Figure 4.51: add new announcement

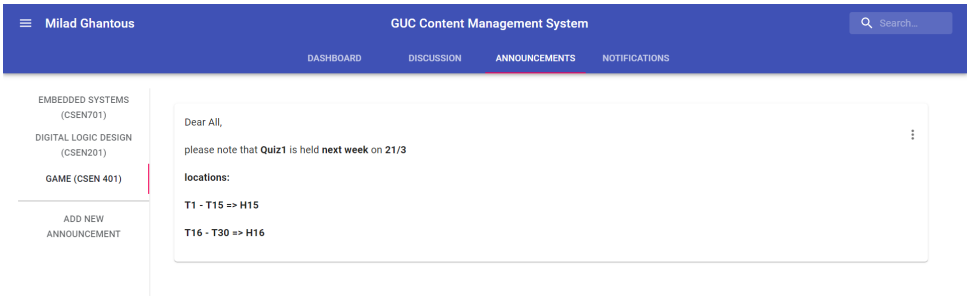


Figure 4.52: view course announcements

- **Notifications**
  The fourth main feature in the new system is notification. Notifications can notify the user of the new updates. There are two types of notifications in the system course notifications and discussion notifications. Course notifications includes any notification related to course for example when the instructor uploads a new content or adds a new announcement this is course notifications. Discussion notifications is any notification related to discussion for example when a student or an instructor posts a new discussion all students and instructors assigned to this course get a notification that there is a new discussion has been posted to that course or content. Also when there is a new comment to a discussion a notification is sent to the author of that discussion.

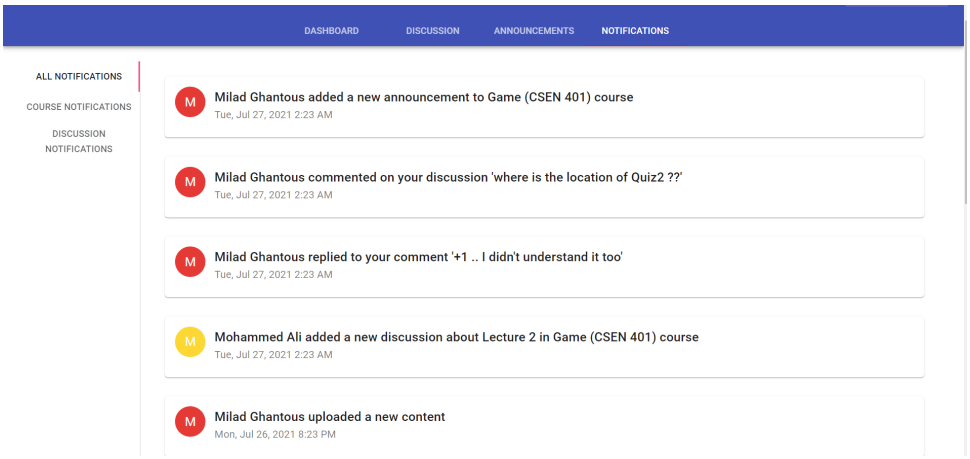


Figure 4.53: All notifications

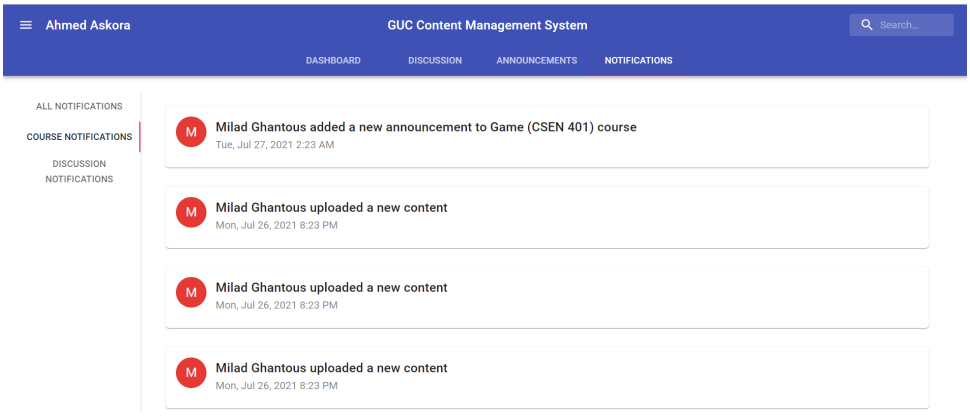


Figure 4.54: Course notifications

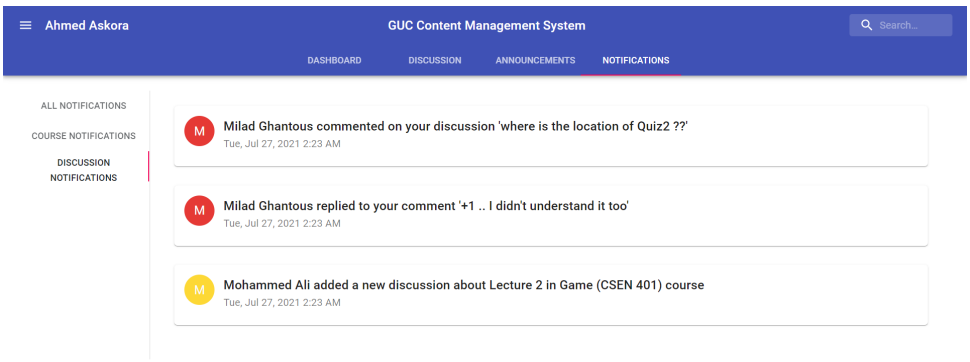


Figure 4.55: Discussion notifications

- **Search**
  The new system has a new feature which is search. Students can search for contents by content title or even content tags. This feature can help students find what they want in a quick and easy way.

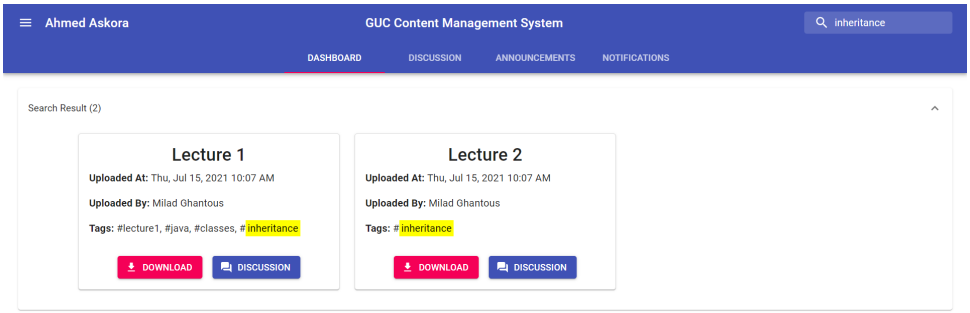


Figure 4.56: search

- **Watch Video**
  The new system has a new feature which is the student can watch a video and the discussion on this video at the same time. Also the pop up video problem is solved in the new system and the video looks nicer.

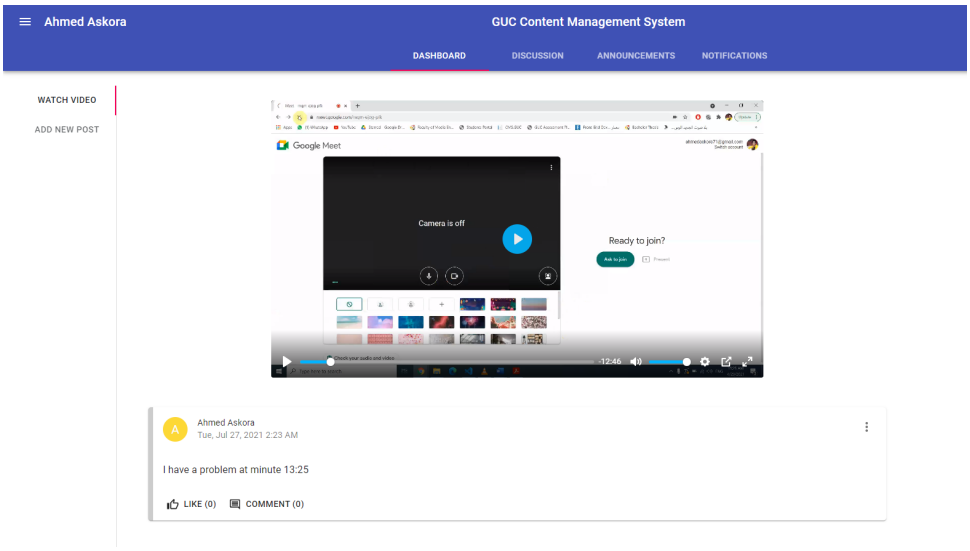


Figure 4.57: search

Chapter 5

Conclusion and Future Work

5.1 Conclusion

The main goal of this project is to enhance the current CMS by fixing the disadvantages and adding new features. The new system succeeded to solve the problem of unorganized content of the courses by adding folder feature which enables the instructors to organize contents in folders they want. The new system also improved the discussion module to have comments and replies in an user-friendly way. And this discussion module helps students and instructors spread thoughts between each other. Search by tags also is a new feature added to the system which make it easy for students to reach for contents they want. The new system improved course announcements feature by separating it to its own module and now all announcements of all courses are in one place. Notification feature is also added to the new website in order not to miss any important update.

5.2 Future Work

In the future, more work needs to be done for the website to be more enhanced and more user-friendly. This can be done by adding more features to the website. For example it's a good idea for GUC CMS to have a calendar showing all upcoming quizzes and deadlines. This can help students memorize important deadlines. Another feature can be added is instructor page. For example each instructor can have a page showing instructor info like GUC mail, office location and office hours. This can help students reach their instructor more easily.

Appendix

Appendix A

Lists

GUC	German University in Cairo
TA	Teacher Assistant
CMS	Content Management System
ERD	Entity Relationship Diagram
OOP	Object Oriented Programming
VOD	Video On Demand
DOM	Document Object Model
MVC	Model View Controller
JSON	JavaScript Object Notation
JWT	JSON Web Token
SQL	Structured Query Language
NoSQL	not only SQL - non-relational database
HTML	Hypertext Markup Language
CSS	Cascading Style Sheet
JSX	JavaScript XML
HTTP	Hypertext Transfer Protocol
API	Application Programming Interface

List of Figures

Bibliography

[1] Google Classroom. https://classroom.google.com/.

[2] Moodle. https://moodle.org/.

[3] GUC CMS. https://cms.guc.edu.eg/.

[4] SQL vs NoSQL. https://www.mongodb.com/nosql-explained/nosql-vs-sql.

[5] MongoDB. https://www.mongodb.com/.

[6] Express. https://expressjs.com/.

[7] React. https://reactjs.org/.

[8] Nodejs. https://nodejs.org/en/.

[9] mongoose. https://mongoosejs.com/.

[10] joi. https://joi.dev/.

[11] moment.js. https://momentjs.com/.

[12] bcryptjs. https://www.npmjs.com/package/bcryptjs.

[13] json web token. https://jwt.io/introduction.

[14] multer. http://expressjs.com/en/resources/middleware/multer.html.

[15] bootstrap. https://getbootstrap.com/.

[16] Material-UI. https://material-ui.com/.