

Florida International University

Knight Foundation School of Computing and Information Sciences

Software Engineering Focus

Final Deliverable

Project Title: DemoEd: Content Creation

Team Members: Shanice Bando, Alisa Chavez, David Giles, Franklin Neves, Batiah Sinepha

Product Owner(s): Dr. Sat Gavassa

Mentor(s): N/A

Instructor: Masoud Sadjadi

April 17, 2024

The MIT License (MIT)

Copyright (c) 2016 *Florida International University*

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

ABSTRACT

This document presents the information necessary to gain a good understanding of the DemoEd application and its existing framework, along with further implementation and development process from the Spring 2024 Capstone II team.

TABLE OF CONTENTS

Abstract	3
Table of Contents	4
Introduction	7
Current System.....	7
Purpose of New System	9
User Stories	10
Implemented User Stories	10
Pending User Stories	11
Project Plan	11
Hardware and Software Resources	12
Sprints Plan	13
Sprint 1	11
Sprint 2	11
System Design	20
Architectural Patterns.....	20
System and Subsystem Decomposition	20
Deployment Diagram.....	21
Design Patterns	22

System Validation	26
Performance Evaluation:	26
Functionality Testing:	26
Cross-Browser and Device Compatibility:	27
User Acceptance Testing (UAT):	27
Unit Testing:	27
Glossary	29
Appendix.....	31
Appendix A - UML Diagrams	31
Appendix B - User Interface Design.....	32
Created Activities.....	32
Drag And Drop Creation.....	33
Multiple Choice	35
Interactive Video.....	20
Appendix C - Sprint Review Reports	39
Sprint 1.....	39
Sprint 2.....	40
Sprint 3.....	41
Sprint 4.....	42
Sprint 5.....	43

Sprint 6.....	43
Appendix D - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents.....	44
DEMOED React App	44
References.....	59

INTRODUCTION

The DemoED platform empowers educators to effortlessly craft and oversee courses and activities while providing students with an engaging and dynamic learning environment.

At its core, our application enables teachers to create and manage courses and activities with ease. Our intuitive interface streamlines the process, allowing educators to focus on what truly matters: fostering knowledge and growth.

Students benefit from the seamless experience our platform offers. With the freedom to engage with activities an unlimited number of times during the availability period, learners can reinforce their understanding and master concepts at their own pace.

Under the hood, our application boasts a robust technological framework. Leveraging the power of React for the front-end, and Node.js and Express for the back-end, we deliver a seamless and responsive user experience. Meanwhile, MongoDB serves as the backbone of our database infrastructure, providing reliability and scalability to meet the demands of modern education.

Current System

The DemoEd application is a web-based learning system in which its goal is provide a flexible learning environment to students and instructors via offering a plethora of

educational tools, i.e., instructors can create classes that allow the creation of assignments, grade checking and file management. DemoEd is viewable on a standard web browser for personal computers and laptops. The existing system operated on a client-server architecture, with the application's frontend operating primarily with React Javascript, HTML, and CSS. The backend server is connected and managed with NodeJS (specifically Express.js) framework and Nodemon library, with a database viewable through MongoDB.

Purpose of New System

The system's framework was not significantly altered but rather additions were made to the existing product instead. The product owner expressed interest in wanting higher learning flexibility via configurable activities. The goal of this semester's project was to implement such instructor-editable content under the "Create Content" tab on the instructor-side of the application. Most of the activity features are implemented using React JS, but there were several other modifications made to the application to improve and enhance performance, which were based on the product owner's needs. Some features include adding editable components, improvements to backend structure and requests, and providing documentation.

USER STORIES

The following section provides the detailed user stories that were implemented in this iteration of the DemoEd project. These user stories served as the basis for the implementation of the project's features. This section also shows the user stories that are to be considered for future development.

Implemented User Stories

Story Title	Assigned To	Estimate
Postman Collection	Franklin Neves	10 hours
README file (React)	Franklin Neves	4 hours
README file (Express)	Franklin Neves	4 hours
Design of Drag-and-Drop Instructor side	Franklin Neves	6 hours
Design of the Drag and drop student side	Franklin Neves	6hrs
Drag and drop implementation	Franklin Neves	15 hours
Create and Design Multiple-Choice Activity	Alisa Chavez Batiah Sinepha	7 hours
Design an Interactive-Video Activity (frontend)	David Giles Shanice Bando	6 hours
Multiple-Choice Demo	Alisa Chavez	10 hours

	Batiah Sinepha	
Interactive-Video Demo	David Giles Shanice Bando	10 hours
Drag-and-Drop Demo	Franklin Neves	10 hours

Pending User Stories

Story Title	Estimate
Modules Design (Instructor side)	20 hours
Refactor pre-made activities	30 hours
Modules Design (Students side)	15 hours
Modules implementation (Instructor side)	30 hours
Modules implementation (Students side)	25 hours

PROJECT PLAN

As a team, we incorporated Agile development techniques and required the sprints to be planned. Although the project outline was established as a team, we updated it to fit modern needs. This section highlights everything needed to continue the development of this product.

Hardware and Software Resources

The following is a list of all hardware and software resources that were used in this project:

- Integrated Development Environment (IDE)
 - IntelliJ IDEA
 - Webstorm
 - VSCode
- Postman
- MongoDB Compass
- NodeJS 14 (npm) or higher installed in environment
- Java 17 SDK
- Personal computer or laptop
- Web browser (Firefox, Chrome, Microsoft Edge)
- Local connection to your machine for testing

SPRINTS PLAN

Sprint 1

Our team accomplished several tasks. We met with the product owner to obtain project specifications and reviewed sections of the code repository collaboratively. Looking ahead to the next scrum meeting, we had planned to review the back end of the code repository to ensure its integrity. However, we faced a hurdle as the previous product owner had abandoned the project, prompting us to switch to a different project. This transition added complexity to our workflow, but we navigated it together as a team.

Sprint 2

During this sprint, our team came up with the following stories:

Story Title	Assigned To
Postman Collection	Franklin Neves
Readme File	Franklin Neves
Front-end planning for drag-and-drop feature	Shanice Badoo Batiah Sinepha
Create mockups for drag-and-drop interactions	Shanice Badoo Alisa Chavez
Back-end planning	David Giles Alisa Chavez
Front-End Planning for updating existing question format	David Giles Batiah Sinepha

The main goal of this sprint was to get a better understanding of what the product owner was expecting. She kept insisting on implementing an already existing library called H5P.

Sprint 3

During this sprint, our team came up with the following stories:

Story Title	Assigned To
H5P Research & Plugin	David Giles Shanice Bando
Create Instructor UI to create activities.	David Giles Alisa Chavez
Implement the requests to create drag-n-drop activity in express application	Franklin Neves Shanice Bando
Implement sending requests and receiving information about drag and drop application.	Franklin Neves Batiah Sinepha
Design the structure of the drag-n-drop activity	Batiah Sinepha Franklin Neves
Research on how to implement an interactive video activity	Franklin Neves David Giles Shanice Bando
Design an interactive video activity front-end	Franklin Neves David Giles Shanice Bando
Design the back-end implementation for interactive video activity	David Giles Batiah Sinepha

	Shanice Badoo
Create and design multiple choice activity	Alisa Chavez Batiah Sinepha
Create the requests for multiple choice activity	Alisa Chavez Batiah Sinepha

This Sprint's main goal is to have a basis setup for the product, within the instructor's side of the application to create different activities.

Sprint 4

During this sprint, our team came up with the following stories:

Story Title	Assigned To
Multiple Choice Demo	Alisa Chavez Batiah Sinepha
Interactive Video Demo	Sue David Alisa
Express requests	Franklin Neves
Assigning Drag and drop to students	Franklin Neves Batiah Sinepha
Created Activities screen	Franklin Neves
Interactive Video Instructor Page	Sue David

	Alisa
Interactive Video Student Page	Sue David Alisa Batiah Sinepha

As a team, this Sprint was meant to start implementing the planned execution from the previous one.

Sprint 5

During this sprint, our team came up with the following stories:

Story Title	Assigned To
Multiple Choice Styling	Batiah Sinepha Franklin Neves
Multiple Choice Implementation	Batiah Sinepha Alisa Chavez
Multiple Choice Demo	Batiah Sinepha
Module Creation	Franklin Neves
Module Update	Franklin Neves
Update current assignment creation	Franklin Neves
Interactive Video component storage and retrieval	David Giles Shanice Badoo

As a team, we planned to familiarize ourselves with the necessary resources, source code, and the current version of the product. After completing this stage, we began implementing the requested functions given by the product owner.

Sprint 6

SYSTEM DESIGN

Most of the design decisions had already been established for this pre-existing website. We were able to understand the existing system design and aggregate our own models and system into it.

Architectural Patterns

The current project supports a variety of architectural patterns, from singleton to a command pattern. Our job meant we needed to navigate through this project to fully comprehend its complexities and be able to provide the product owner with her required results.

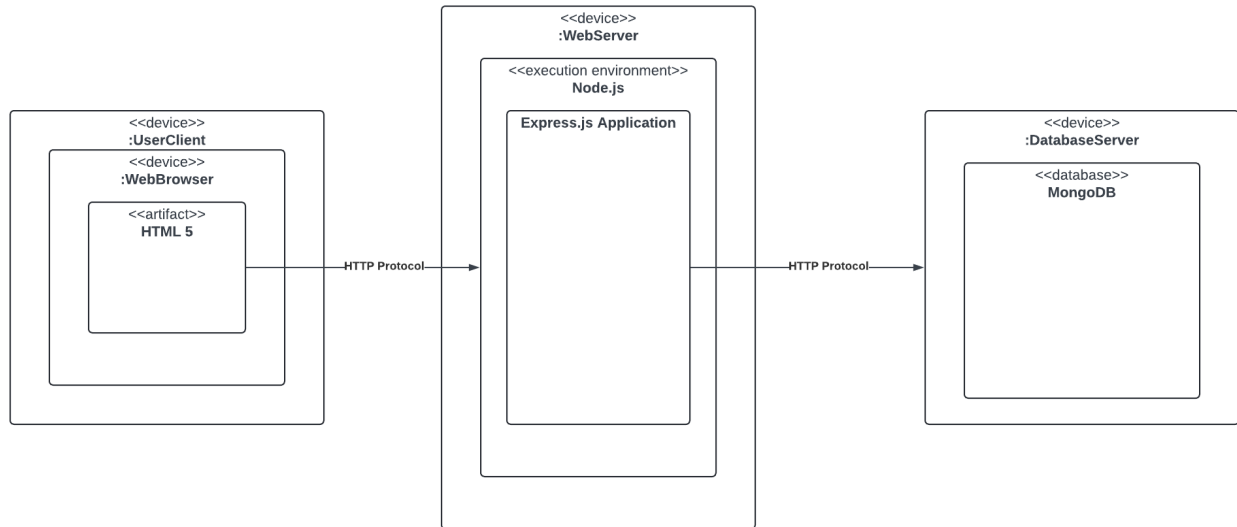
As a team, we took advantage of the non-relational database and created an activities object serving as the Decorator pattern. The activities object can fulfill all the requirements of any activity type, while allowing for only a single call to the backend. Thus, all activities can use the same calls, even though they have different attributes.

System and Subsystem Decomposition

As a team, we identified and created the necessary subsystems to fulfill our tasks. The Product owner insisted on many different activity types, which led us to split up and work on those activities individually.

We broke down our major system, which was content creation, into a Drag and drop activity, an interactive video activity, and a multiple-choice activity. Each activity was assigned to a pair of team members so that we minimize the development time.

Deployment Diagram



Design Patterns

In our software development design process, we decided to integrate the “Structural Design” patterns in our software development cycle. This technique allowed us to organize classes and objects in a way that guaranteed flexibility, reusability, and the overall maintainability of our code. Here's how we approached each feature using these patterns:

Video Feature:

Modular Component Design:

- We utilized ReactPlayer, a modular component library, to display the video on the website. This choice aligns with the principles of structural design patterns, where components are designed to be interchangeable and composable.
- By encapsulating the video display functionality within ReactPlayer, we followed the concept of encapsulation, a key principle of structural design patterns, which promotes modularity and separation of concerns.

Clear Separation of Concerns:

- The use of ReactPlayer for video rendering and management helped maintain a clear separation of concerns. This separation aligns with the goals of structural design patterns, which emphasize organizing code in a way that makes it easy to understand and modify.
- By accepting the start and stop times for cropping as input parameters, we ensured that the logic for cropping the video remained distinct from the video rendering

functionality provided by ReactPlayer. This separation enhances code readability and maintainability.

Flexible and Extensible Design:

- The design allowed for flexibility and extensibility; characteristics often associated with structural design patterns. While we didn't explicitly apply patterns like Adapter or Decorator, the modular design of ReactPlayer and the separation of concerns facilitated easy integration of new features or enhancements in the future.
- For example, if we wanted to add additional video editing capabilities beyond cropping, the modular structure provided by ReactPlayer would make it relatively straightforward to incorporate new functionalities without significantly modifying existing code.

Drag and Drop Feature:

Modular Component Design:

- The "react-beautiful-dnd" library itself follows a modular component design approach. It provides separate components for draggable items, droppable containers, and drag handles, promoting reusability and composability.
- By utilizing these modular components, we inherently apply the principles of structural design patterns, where components are designed to be interchangeable and self-contained.

Clear Separation of Concerns:

- The drag and drop logic are encapsulated within the components provided by the "react-beautiful-dnd" library. This encapsulation ensures a clear separation of concerns, with the library handling the complexities of drag and drop interactions independently from the rest of the application logic.
- Our implementation likely involves defining the structure of draggable items and droppable containers within the application's UI components. This separation of concerns aligns with the goals of structural design patterns, promoting code readability and maintainability.

Flexibility and Extensibility:

- The use of a specialized library like "react-beautiful-dnd" allows for flexible and extensible design. While we may not explicitly apply structural design patterns like Adapter or Decorator, the modular nature of the library facilitates easy integration of drag and drop functionality into our application.
- Additionally, the library's API likely provides hooks or callbacks for customizing behavior, such as handling drop events or defining constraints. This flexibility enables us to tailor the drag and drop behavior to suit our specific requirements without tightly coupling it to other parts of the application.

Multiple Choice Feature:

Modular Component Design:

- We likely encapsulated the dynamic input boxes and buttons as reusable React components. This modular component design promotes reusability and composability, reflecting the principles of structural design patterns.
- By breaking down the UI elements into smaller, self-contained components, we adhere to the idea of modular design, which encourages building complex systems from simpler, interchangeable parts.

Clear Separation of Concerns:

- Each dynamic input box and button likely has its own functionality and behavior encapsulated within separate components. This clear separation of concerns ensures that each component is responsible for a specific aspect of the multiple-choice feature.
- By adhering to this separation, we enhance code readability and maintainability, as developers can easily understand and modify individual components without affecting the functionality of other parts of the application.

Flexibility and Extensibility:

The use of dynamic input boxes and buttons allows for flexible and extensible design. While we may not explicitly apply structural design patterns, the modular nature of React components enables us to easily add or remove input fields and buttons as needed.

Additionally, we may implement functionality such as validation or data processing logic within these components, further enhancing their flexibility and extensibility.

SYSTEM VALIDATION

To validate the effectiveness and reliability of our interactive educational platform BioDemoEd developed using React and Node.js, we conducted a series of tests to evaluate the system's performance across different scenarios. The system is designed to facilitate active learning through interactive activities, quizzes, and course management features.

Performance Evaluation:

We conducted performance tests by simulating multiple user interactions simultaneously to assess the system's response time and stability. This involved automated scripts that mimicked student and instructor behaviors on the platform. The results were benchmarked against typical standards for educational platforms, ensuring our system can handle peak loads efficiently during critical usage times such as examinations or assignment submissions.

Functionality Testing:

Extensive functionality tests were performed to verify all features of the platform, including user authentication, course enrollment processes, interactive activities, and data management. We ensured that the user interface is intuitive and that all navigational elements

function as expected. Each feature was tested in isolation and in combination to ensure they integrate seamlessly.

Cross-Browser and Device Compatibility:

The platform was tested across multiple web browsers and devices to ensure a consistent user experience. This included testing on Chrome, Firefox, Safari, and Edge. Adjustments were made to accommodate various screen sizes and resolutions, ensuring accessibility and usability.

User Acceptance Testing (UAT):

We conducted UAT with actual users including students (team members) and instructors (Dr. Sat) to receive direct feedback on the system's functionality and user experience. This feedback was crucial in refining the platform, making it more user-friendly, and aligning it more closely with user needs.

Unit Testing:

We also tested to ensure that individual components function correctly both in isolation and when integrated into the platform.

These validation efforts demonstrate that our educational platform is not only functional, efficient, and secure but also robust and user-friendly, making it suitable for educational institutions seeking to enhance the learning experience.

GLOSSARY

Framework: Already written code, tools, or set of libraries used as the structure for developing applications.

Client-server architecture: A computing model where a client requests services or resources and servers then provide these services or resources

React JS: This is a JavaScript library utilized in building user interfaces, particularly for web applications.

NodeJS: NodeJS can be described as a runtime environment for executing JavaScript code outside of web browser environments.

Express.js: This is a web application framework for Node.js which simplifies the process of developing web servers and handling HTTP requests and responses.

MongoDB: MongoDB is a NoSQL database management system that uses a document-oriented data model to store data in flexible, JSON-like documents.

Agile Development: Agile development is a software development methodology that emphasizes iterative development, collaboration, and flexibility to adapt to changing requirements and deliver value to customers quickly.

Integrated Development Environment (IDE): An Integrated Development Environment is a software application that provides comprehensive tools and features to facilitate software development.

IntelliJ IDEA: This is a popular IDE that offers advanced coding assistance, productivity features, and support for various programming languages and frameworks.

VSCode: Visual Studio Code is a lightweight and extensible source-code editor known for its wide range of extensions, powerful editing features, and support for multiple programming languages.

Postman: Postman is a popular API development tool used for testing, debugging, and documenting APIs.

Architectural Patterns: Architectural patterns are high-level design principles or blueprints that guide the organization and structure of software systems

Decorator Patterns: The Decorator pattern, as a structural design pattern, enables the dynamic addition of behavior to individual objects without altering the behavior of other objects belonging to the same class.

Subsystem: A subsystem is a standalone module in a software system, executing a specific set of tasks, often collaborating with other subsystems to fulfill the system's goals.

ReactPlayer: An open-source library for integrating media players into React applications

APPENDIX

Appendix A - UML Diagrams

Appendix B - User Interface Design

Created Activities

Created Activities			
Title	Description	Activity Type	Delete
Eukaryote VS. Prokaryote	This is my description	dragAndDrop	Actions ▾ EditPreview X

Drag And Drop Creation

PreviewSave

Activity Title

Activity Description

+

Container Title -

Ordered Answers:

Extra Answers:

Double click to edit:

New Draggable

Remove Draggable

Created Activities

Text And Images

Multiple Choice

Drag and Drop

Video

H5P

Other Activities

Preview

Save

Eukaryote VS. Prokaryote

This is my description

Eukaryote

Membrane - embedded nucleus

Nucleolus

Mitochondria

Prokaryote

Cell Wall

Flagellum

Capsule

Ordered Answers:

Extra Answers:

Double click to edit:

New Draggable

Remove Draggable

Multiple Choice

Created Activities	Text And Images	Multiple Choice	Drag and Drop	Video	H5P	Other Activities
--------------------	-----------------	-----------------	---------------	-------	-----	------------------

< Back to courses

✕ Cancel

Edit

Preview: Biology Quiz 1

"Question 1: Which of these is the powerhouse of the cell?"

☐ Nucleus☒ Mitochondria☐ Ribosome☐ Cell Lipid Layer

Begin Quiz

Created Activities	Text And Images	Multiple Choice	Drag and Drop	Video	H5P	Other Activities
--------------------	-----------------	-----------------	---------------	-------	-----	------------------

< Back to courses

✕ Cancel

Question 1: Question 1: Which of these is the powerhouse of the cell?

☐ Nucleus☒ Mitochondria☐ Ribosome☐ Cell Lipid Layer

Submit Quiz

< Back to courses

✖ Cancel

Created Activities

Text And Images

Multiple Choice

Drag and Drop

Video

H5P

Other Activities

Quiz Completed!

Your score is 1 out of 1.

Create a New Quiz

< Back to courses

✖ Cancel

Created Activities

Text And Images

Multiple Choice

Drag and Drop

Video

H5P

Other Activities

Create Multiple Choice Assessment

Biology Quiz 1

Question 1: Which of these is the powerhouse of the cell?

Nucleus

Mitochondria

Ribosome

Cell Lipid Layer

☐Correct Answer(s)

☒Correct Answer(s)

☐Correct Answer(s)

☐Correct Answer(s)

+ ADD QUESTION

Preview

Interactive Video

 Biology DemoEd

Courses Shan's profile Log Out

< Back to courses ✕ Cancel

Edit this course

Modules

Add assignments

Add students

View students

View assignments

View grades

Add Instructors

View Instructors

Created Activities

Text And Images

Multiple Choice

Drag and Drop

Video

H5P

Other Activities

Crop Your Preferred Video

Please enter the video link you want to use:

Load Video Metrics

Reset

 Add assignments
 Add students
 View students
 View assignments
 View grades
 Add Instructors
 View Instructors
 Create Content
 Delete this course



Duration of the video: 807.00 seconds

Current Time: 12.18 seconds

Start Time: 10

Stop Time: 15

Title:

Intro to Biology

Description:

eneral concept of Biology.

Save Video

Reset

APPENDIX C - SPRINT REVIEW REPORTS

Sprint 1

After a show-and-tell presentation, we consolidated our findings as follows:

1. Drag and drop questions can be implemented
 - a. Using classic JS, CSS, HTML

Features/Functions that were tested thoroughly and worked as expected without any errors/issues:

1. Express application endpoints were successful.

Features/Functions that were tested thoroughly and did NOT work as expected. Here is the list of issues/errors observed:

1. When running the front and backend application a user with a student email was not able to register showing this error: “Invalid email address.”

Features/Functions that were tested thoroughly and would benefit from some improvements/enhancements. Here is the list of suggested enhancements/improvements:

1. Organization of routes of the application
2. Improve the CSS file organization
3. Authentication process
 - a. Possibly generate tokens for authorization.

Sprint 2

After a show-and-tell presentation, we consolidated our findings as follows:

1. Drag and drop questions can be implemented.
 - a. Using classic JS, CSS, HTML

Features/Functions that were tested thoroughly and worked as expected without any errors/issues:

1. Express application endpoints were successful.

Features/Functions that were tested thoroughly and did NOT work as expected. Here is the list of issues/errors observed:

1. When running the front and backend application a user with a student email was not able to register showing this error: “Invalid email address.”

Features/Functions that were tested thoroughly and would benefit from some improvements/enhancements. Here is the list of suggested enhancements/improvements:

1. Organization of routes of the application
2. Improve the CSS file organization.
3. Authentication process
 - a. Possibly generate tokens for authorization.

Sprint 3

After a show-and-tell presentation, we consolidated our findings as follows:

1. Drag and drop questions can be implemented.
 - a. Using classic JS, CSS, HTML

Features/Functions that were tested thoroughly and worked as expected without any errors/issues:

1. Express application endpoints were successful.

Features/Functions that were tested thoroughly and did NOT work as expected. Here is the list of issues/errors observed:

1. When running the front and backend application a user with a student email was not able to register showing this error: “Invalid email address.”

Features/Functions that were tested thoroughly and would benefit from some improvements/enhancements. Here is the list of suggested enhancements/improvements:

1. Organization of routes of the application
2. Improve the CSS file organization.
3. Authentication process
 - a. Possibly generate tokens for authorization.

Sprint 4

After a show-and-tell presentation, we consolidated our findings as follows:

1. Team members learned how to create and work with existing components for activities
2. Packages needed for multiple choice questions may not be required
3. Interactive video
4. The instructors side of creating activities works as intended
5. Creating a drag and drop activity has been approved my product owner.
6. Documentation along with explanations on how to add personalized activities for the application has been created.

Features/Functions that were tested thoroughly and worked as expected without any errors/issues:

1. Instructor Drag and drop activity creation

Features/Functions that were tested thoroughly and did NOT work as expected. Here is the list of issues/errors observed:

- Assigning any activities have not been implemented

Sprint 5

After a show-and-tell presentation, we consolidated our findings as follows:

1. The task to create modules so teachers can assign multiple activities to students ended unfinished.

Features/Functions that were tested thoroughly and worked as expected without any errors/issues:

1. Drag and drop activity creation.
2. Created activities screen.

Features/Functions that were tested thoroughly and did NOT work as expected. Here is the list of issues/errors observed:

- The multiple-choice creation is not being saved.
- The Interactive video creation is not being saved to the database.

Features/Functions that were tested thoroughly and would benefit from some improvements/enhancements. Here is the list of suggested enhancements/improvements:

- The interactive video could use better styling, to become more functional for the user.

Sprint 6

After a show-and-tell presentation, we consolidated our findings as follows:

1. The task to create modules so teachers can assign multiple activities to students ended unfinished.

Features/Functions that were tested thoroughly and worked as expected without any errors/issues:

1. Drag and drop activity creation.
2. Created activities screen.

Features/Functions that were tested thoroughly and did NOT work as expected. Here is the list of issues/errors observed:

- The Multiple-choice question adding functionality would not accept text input

Features/Functions that were tested thoroughly and would benefit from some improvements/enhancements. Here is the list of suggested enhancements/improvements:

1. Video Creation
 - a. Styling needs improvement
 - b. Object structure needs to be modified to work with Database schema.
2. Multiple Choice
 - a. Even though the issue was eventually resolved, additional bug fixes are needed.

APPENDIX D - USER MANUALS, INSTALLATION/MAINTENANCE

DOCUMENT, SHORTCOMINGS/WISHLIST DOCUMENT AND OTHER DOCUMENTS

DEMOED React App

This is a web application that allows teachers to create and manage courses and activities. Students can work on the activities unlimited times during the availability period. Scores are recorded only when students have complete a successful attempt without any mistakes.

The application is built using React and Ant Design for the front-end, and Node.js and Express for the back-end while using MongoDB as the database.

The application is divided into two main sections:

- The course section
- The activity section

There are two main views depending on user privileges:

- Student view
- Instructor view

The course section:

The course section allows teachers to create and manage courses. Teachers can create new courses, view all courses, view a single course, edit a course, and delete a course.

The activity section:

- The activity section allows teachers to create and manage activities.
- Teachers can create new activities, view all activities, view a single activity, edit an activity, and delete an activity.

They are able to assign activities to students and view the results of the activities. Current editing function only allows to change the name of the activity and availability dates but not to change the activity itself, but we are working to include that.

Although the application comes with a few pre-built activities, teachers can create custom activities to suit their needs.

Teachers are able to create:

- Text and Images activities
- Multiple Choice activities
- Drag and Drop activities
- Interactive Video activities

How to create custom activities

Step 1:

- Create a new folder in the `src/components/course/createContent` directory with the name of the activity you want to create. For example, if you want to create a new activity called `MyActivity`, create a new folder called `MyActivity` in the directory.

Step 2:

- Every activity needs to have a title, description, and courseId when being saved.
- The activity object looks like this:

```
{  
  
  title: 'MyActivity',  
  
  description: 'This is a description of my activity',  
  
  activityParams: {  
    'param1': 'value1'  
  },  
  
  activityType: 'MyActivity',  
  
  activityId: Auto generated when saved,  
  
  courseId: The course id of the course the activity is being saved to,  
}
```

Step 3:

- Add your activity to the create content page by adding it to the `tabList` array in the `CreateContent` component. For example:

```
tabList = [  
  { name: "Created Activities", value: "createdActivities" },  
  { name: "Text And Images", value: "textImages" },  
  { name: "Multiple Choice", value: "multipleChoice" },  
  { name: "Drag and Drop", value: "dragAndDrop" },  
  { name: "H5P", value: "h5p" },  
  { name: "MyActivity", value: "myActivity" },  
  { name: "Other Activities", value: "otherActivities" },  
]
```

- Also add your activity to the `renderForms` function with any props you want to pass to it.

For example:

```
renderForms = () => {  
  switch (this.state.whatToShow) {  
    case "multipleChoice":  
      return <MultipleChoice />;  
    case "textImages":  
      return <TextImages onValueEmitted={this.showItem} />;  
    case "h5p":  
      return <H5P />;  
    case "otherActivities":  
      return <OtherActivities />;  
    case "dragAndDrop":  
      return <DragAndDropCreation  
        onSave={this.saveActivity}  
        editActivity={this.state.editThisActivity}  
      />;  
    case "myActivity":  
      return <MyActivity />;  
  
    default:  
      return <CreatedActivities
```

```

    getActivityInfo={this.getActivityInfo}
    getActivity={this.getActivity}
    showItem={this.showItem}
    setEdit={this.editThisActivity}
    {...this.props}/> // So that we can redirect to the activity page with edit
  }
};

```

- Already implemented, there is an `editActivity` function as well as `saveActivity` function that you can use to save and edit your activity. You can also pass props to your activity by adding them to the `renderForms` function. For example:

```

saveActivity = async (activity, then) => {
  let newActivity
  // If the ID exists, then we are updating the activity
  if (activity._id) {
    newActivity = {
      ...activity,
      courseId: this.props.theCourse._id,
    }
    return this.service.put("/") + activity._id, newActivity)
    .then((response) => then(response))
  } else {

```

```

// If the ID does not exist, then we are creating a new activity

    newActivity = {
        ...activity,
        courseId: this.props.theCourse._id,
        activityType: this.state.whatToShow,
    }

    return this.service.post("/create", newActivity)
        .then((response) => then(response))
    }
}

editThisActivity = (activity) => {
    this.setState({ whatToShow: activity.activityType, editThisActivity: activity });
};

-----

case "myActivity":
    return <MyActivity
        onSave={this.saveActivity}
        editThisActivity={this.editThisActivity}/>;

```

- For the edit activity feature, you will need to add a check within the constructor of your activity to see if the activity is being edited. For example:

```
constructor(props) {  
  super(props);  
  this.state = {  
    title: "",  
    description: "",  
    activityParams: {},  
  };
```

```
if(this.props.editThisActivity) {  
    const { title, description, activityParams } = this.props.editThisActivity;  
  
    this.state = {  
        title: title,  
        description: description,  
        activityParams: activityParams,  
    };  
}  
  
}
```

Preview Component

The preview component is a reusable component that can be used to preview any activity.

It is essentially a wrapper component that will display whatever is passed as a child component, within a modal.

How to use the preview component

- Import the preview component into your activity component
- Create a state variable to keep track of whether the modal is open or not
- Create a function to open the modal
- Create a function to close the modal (toggling the state variable)
- Pass the close function to the preview component
- This is an example using the preview component in the `MyActivity` component:

```
import React, { Component } from 'react';

import { Button } from 'antd';

import PreviewActivityModal from "../../utils/previewActivity/PreviewActivityModal";

class MyActivity extends Component {

  constructor(props) {

    super(props);

    this.state = {

      previewModal: false,

    };

  }

  closePreviewModal = () => {

    this.setState({ previewModal: false });

  };

  render() {

    return (

      <div>

        <Button onClick={() => this.setState({ previewModal: true })}>Preview</Button>

        {
```

```
this.state.previewModal &&  
<PreviewActivityModal onClose={this.displayPreview}>  
<myActivityComponent  
{...this.props}/>  
</PreviewActivityModal>  
  
}  
  
);  
  
}  
  
}
```

Toggle Switch Component

The toggle switch component is a reusable component that can be used to create a toggle switch.

The component was created due to lack of a built-in toggle switch with the Bootstrap version.

How to use the toggle switch component

- Import the toggle switch component into your activity component
- Create a state variable to keep track of the toggle switch value
- Create a function to handle the toggle switch value change
- Pass the value and the function to the toggle switch component
- This is an example using the toggle switch component in the `MyActivity` component:

```
import React, { Component } from 'react';
```

```
import ToggleSwitch from "../../utils/toggleSwitch/ToggleSwitch";
```

```
class MyActivity extends Component {
```

```
  constructor(props) {
```

```
    super(props);
```

```
    this.state = {
```

```
      toggleSwitchValue: false,
```

```
      toggleSwitchValue2: false,
```

```
};  
}
```

```
handleToggleSwitchChange = (value) => {  
  this.setState({ toggleSwitchValue: value });  
};
```

```
render() {  
  return (  
    <div>  
      <ToggleSwitch  
        value={this.state.toggleSwitchValue}  
        onChange={this.handleToggleSwitchChange}  
      />  
  
      <ToggleSwitch  
        value={this.state.toggleSwitchValue2}  
        onChange={(value) => this.setState({ toggleSwitchValue2: value })}  
      />  
    </div>  
  );  
};
```

}

}

REFERENCES

<https://youtu.be/UX5HlrxBRUc?si=MdoVZ-HrIKFmJttT>

<https://youtu.be/VMZ7lcSdVnY?si=MVkhROWvsHEFrXD>

<https://reactnative.dev/docs/environment-setup>