Data Driven Web Development

Working Plan

Milestone project 3

Classapp

Jussi Nousiainen
Code Institute - Full Stack Web Development Course
Module 3

Table of content

Project brief

The project is following my own desire to bring about a digital tool for the community I myself belong to, the dance teachers. The project will be built using data-driven application technologies I have learned on the course module. Classapp intends to provide a mobile and easy tool to plan, store and manage dance classes in a digital form which would be accessible wherever they might be teaching.

Contextualizing the need

The idea for the application is a direct response to my own experiences as a dance teacher which I believe I share with many others in this occupation. Classapp is designed for dance teachers, to design, save and archive their dance classes.

There are dozens of different dance styles and genres taught in most diverse settings all over the world. Be it street dance,folk, classical, contemporary, tap, ballroom to start by mentioning just a few, each class is a result of a dedicated work by a dance teacher. A dance class is just the tangible and experiential manifestation of the skill and creativity of the dance teacher front of the class. Much of the teacher's work is hidden. Making, or designing classes happens outside the teaching of a class itself, competing for time from other necessary tasks that come with the job.

Dance class material is often tailored for specific occasions with their own defined goals and needs. Teachers have to keep in mind the level of the students taking the class, wishes of the organizations where they teach or needs of educational institutions curriculums when designing classes or series of classes. For example, designing a series of dance classes for higher educational institutions, the classes should provide thematically coherent experience for the dance students, while filling all kinds of other equally important objectives and requirements set by the institution. Teaching in an open studio, in a dance company, for amateurs, for children or for professionals have all very different kinds of implications and requirements.

Class material tends to accumulate with time and good material and well working classes can get lost and forgotten. Information and valuable insights, such as the first hand experiences with the students, tend to vanish with the forgotten classes. Especially when working with groups of students in schools, in order to help the students improve over time, notes and remarks from the past are the most useful resource of feedback. They are also good material for reflection and self assessment for teachers themselves, who often can only rely on their own judgement.

Teachers time is often limited and the time for planning, personal improvement and developing new ideas is rarely paid. Sometimes, to come up with new material can be taxing and time consuming. Resorting to already tested material is often helpful and can be a huge time saver.

Teachers tend to have an enormous capacity to store a full catalogue of classes they have created and recall them when needed. However, this might not be always the case. Furthermore, for teachers' own personal development it is important to change the habits every now and then and look for new perspectives and ideas. Revisiting past classes and notes is often a good point of departure when looking for new inspiration. It helps to recognize habitual patterns, find new angles and gain new insights. Building a comprehensive archive of classes can thus be a time saver, but also a source of inspiration, reflection and a valuable tool in maintaining a live relationship to teaching the art of dance.

Project Overview

Foreword

I needed to make a decision to deviate from the original project purpose to complete the project I envisioned. Instead of a shared data user can create his/ her own archive, which is accessible only for him /her. I made the decision in order to respect the teachers personal investment, creative capital and rights for their own intellectual property.

To realize a tool that would address the above mentioned points, I considered following features essential for application to fulfill its purpose:

1. Users can create his/ her own archive, which is accessible only for him /her.

   User can:

   1. create an account with a username and password
   2. login with the username and password

2. Users can create classes and they can easily handle the class archives. They should have enough freedom to formulate their classes, yet have a clearly structured interface to create and edit classe.

   Once a class is created user can:

   a. view class
   b. edit class
   c. delete class
   d. duplicate class
   e. associate the class with class series user has created
   f. de-associate the class with class series user has created

3. Users can create exercises for a class. They should have enough freedom to formulate their classes, yet have a clearly structured interface to create and edit exercises.

Once an exercise is created, user can:

a. view exercise
b. edit exercise
c. delete exercise
d. add and delete links to music tracks and videos

4. Users can create logs for the classes.

Once a log is created, user can:

a. view log
b. edit log
c. delete log

5. Users can create series of classes for various purposes

Once a series is created, user can:

a. view series
b. view classes associated with the series
c. edit series
d. delete series

Project Requirements

Practical Python and Data-Centric Development Milestone Project

Project purpose:

In this project, you'll build a full-stack site that allows your users to manage a common dataset about a particular domain.

Value provided:

1. Users make use of the site to share their own data with the community, and benefit from having convenient access to the data provided by all other members.
2. The site owner advances their own goals by providing this functionality, potentially by being a regular user themselves. The site owner might also benefit from the collection of the dataset as a whole.

Main Technologies

HTML, CSS, JavaScript, Python+Flask, MongoDB

Additional libraries and external APIs

Mandatory Requirements

A project violating any of these requirements will FAIL

Data handling: Build a MongoDB-backed Flask project for a web application that allows users to store and manipulate data records about a particular domain. If you are considering using a different database, please discuss that with your mentor first and inform Student Care.
Database structure: Put some effort into designing a database structure well-suited for your domain. Make sure to put some thought into the nesting relationships between records of different entities.
User functionality: Create functionality for users to create, locate, display, edit and delete records (CRUD functionality).
Use of technologies: Use HTML and custom CSS for the website's front-end.
Structure: Incorporate a main navigation menu and structured layout (you might want to use Materialize or Bootstrap to accomplish this).
Documentation: Write a README.md file for your project that explains what the project does and the value that it provides to its users.
Version control: Use Git & GitHub for version control.
Attribution: Maintain clear separation between code written by you and code from external sources (e.g. libraries or tutorials). Attribute any code from external sources to its source via comments above the code and (for larger dependencies) in the README.
Deployment: Deploy the final version of your code to a hosting platform such as Heroku. Make sure to not include any passwords or secret keys in the project repository.
Important Notes

No authentication is expected for this project. The focus is on the data, rather than any business logic.

What I would like to accomplish

I would like to create an application that allows users to at minimum be able to do tasks mentioned in the project overview. The application should be as straightforward to use as possible. It should also give teachers as much creative liberties as possible, avoid being cluttered with too many visuals and be pleasant to use.

Although the authentication is not part of the project requirements, I strongly feel it is a necessary feature in the application. Therefore, I intend to integrate a login and registration feature to the application.

UX

Who is this application for ?

Possible users types

- A freelance teacher working with companies, schools and/or dance studios.
- A curriculum teacher in an educational institution/ permanent school staff.python3 app.py
- Recurring teacher in different educational institutions.
- Company teachers.
- Dance teacher, movement coach, body worker, gym teacher and so forth in any class or workshop setting.

User stories

1. I really need a way to organize the class information so I can look at the classes I have taught, with my notes and logs to set sensible longer term goals for my students.

2. I would like to be able to recall some of my classes from the past, which would give me an idea how my class has evolved and what kind of information was produced and shared - a bit like a diary, but easier to use - to have some inspiration for new classes.

3. Would be great to have videos from my classes at hand sometimes to refresh my memory and to see what I could do differently or better.

4. I would need a way to archive my classes and be able to write detailed logs so I can follow the progress of my students over time and share some insights with my colleagues.

5. It would be handy to be able to easily design a series of classes in advance for specific purposes, which I could then copy and modify depending where I am teaching. I am teaching a lot in different kinds of places, but I do not have time to always come up with a new plan, sometimes a little alteration to an old class is good enough, but after a while I'm getting confused which version I am teaching.

6. I would like to be able to store my classes so that I can find the best fit in each situation, depending on the level of the people I will be teaching, or even the size of the studio or amount

of the expected participants. For example, when I need to teach a class with small adjustments to the usual exercise material to accommodate some situations, like a very small studio.

7. It would be practical to have the associated music tracks and playlists with links stored with the classes and exercises and possibility to write remarks, such as, this track is good for a slower tempo, or this is good energetic music to pump up the class etc.

8. I really need a way to organize the class information so I can look at the classes I have taught, with my notes and logs to set sensible longer term goals for my students.

9. Would be really handy to be able to have a digital document, where I could put images and stuff together with my class and exercises, and everything. I could then show things like, how is the anatomy or demonstrate what I mean with other kinds of media etc. But all in the same place... and that is easy to take to the class.

10. I would like to have a tool that would structure and unify my class designs to make it easier to visit them later - digital would be great so I don’t need to be guessing what I was writing. Actually, something that would help notating my classes would be amazing.

Planning

Project Task List

	Task description	Definition/ objective
	Create application	Define, design and create application for dance teachers
I.	Planning phase	
1	Define what it is	Chart needs and make user cases
2	Design the functionality	Design database schema and function flow
3	Make working plan	List the tasks and define the phases of the working process
4	Design the features and CRUD operations to include	Make a green, yellow, red planning for what is essential, not so important and what is maybe nice to have but not necessary.
5.	Decide technologies to use	Chart what technologies to use to accomplish the application built.
II.	Building phase	
1.	Prepare IDE	Prepare the development environment. Create app.py file and test.py file and any other file and folder to start building the application. Test the connections to the files as needed.
2	Prepare version control	Create GitHub repository and link the project
4	Create database	Create a database and add classes collection to it. Test the connection to the application.
5	View classes hub	Build the classes hub and test it with dummy data.
4	Add class	Build create class view and route and insert class function. Create Save and Cancel buttons. Test classes are created.
5	Add tool buttons and routes for CRUD operation on class	Add over all route schema for edit, delete and duplicate functions and create master tool button
6	View class	Build a class view and route. Test with the database. (Create additional dummy entries in MonogDB)
7	Edit class	Build edit class view and route to edit and update class function. Create Save and Cancel buttons. Test it with the database (class is updated).
8	Delete class	Create delete class from database function. Test create, edit and delete class routes.
9	Duplicate class	Create a duplicate class route and function to insert the new duplicate document. Create Save and Cancel buttons. Test duplicate is created.

10	Test CRUD.	Test all routes till now.
11	View Series hub	Add series collection to the database and add dummy data. Create Series hub view and route. Test with database.
12	Add series	Create add series view and route and insert series function. Create Save and Cancel buttons. Test series are created.
13	Add tool buttons and routes for CRUD operation on series	Add over all route schema for edit and delete series functions and create master tool button
14	Edit serie	Create edit series view and route and update series function. Create Save and Cancel buttons. Test series are edited.
15	Delete series	Create delete series function. Test series are deleted.
16	Select series	Create a reference connection between class and series. Test that the series selection renders on the edit class selection field and on the view class page series section.
17	View classes in series	Create view classes in series view and route. Create more classes to test the select series feature. Test the route from series to view classes in series.
18	Add tool buttons and routes for CRUD operation on classes in series	Create a new master tool button to view, edit, delete and duplicate a class and test the routes and CRUD operations on classes.
19	Add tool buttons and routes for CRUD operation on class view	Create a new master tool button to add exercise, add log, edit class, delete claa and duplicate class. Test the existing routes and CRUD operations on classes. Add routing schema for add exercise and add log.
20	Add log	Create add log view and route and insert log function. Create Save and Cancel buttons. Test log is added (class is updated).
21	Edit log	Create edit log view and route and update log in class function. Create Save and Cancel buttons. Test log is updated (class is updated).
22	Delete log	Create delete log in class function. Test
23	Add exercise	Create an exercise view and route and insert exercise in class function. Create Save and Cancel buttons. Test exercise is inserted (class is updated).
24	Edit exercise	Create an edit exercise button. Create edit exercise view and route and update exercise in class function. Create Save and Cancel buttons. Test exercise is updated (class is updated).
25	Delete exercise	Create a delete exercise button. Create delete exercise in class function. Test exercise is deleted (class is updated).
26	Add music track link	Create an add track button. Create add music track view and route and insert track function. Create Save and Cancel buttons. Test track is inserted (class is updated).
27	Delete music track link	Create a delete track button. Created a delete track function. Test track is deleted (class is updated).

28	Add video link	Create an add link button. Create add video link view and route and insert link function. Create Save and Cancel buttons. Test link is inserted (class is updated).
29	Delete video link	Create a delete link button. Created a delete link function. Test link is deleted (class is updated).
30	Login	Create a user collection in the database and create users. Create a login page and add Login to the navigation bar. Create a submit button. Create a user authentication route.
31	Logout	Create a logout function. Test Login and Logout.
32	Register	Create a register view and route and insert user function function. Test users are created. Test users that are created can login and logout.
33	Navigation	Separate the navigation view for the users who are logged in and users who are not logged in. Test the routes.
34	Feedback	Create a messaging system to give feedback to users. Test the messages are shown.
35	Validation	Create validation function for username, email and password validation. Test the validation.
36	Information	Create a modal to inform users what are the requirements for a valid password
III.	Testing and deployment phase	
37	Cleaning the code	Check and clean the code.
38	Test to deploy	Test all the visuals, features and functionality manually.
39	Deploy the project	Prepare the IDE for deployment. Deploy. Test that the application is running.
40	Go over the testing plan and write the testing report.	
IV.	Finalizing phase	
41	Complete writing the documentation	Make sure all documentation is included, especially README.md!!
42	Go over the project checklist	
43	Last adjustments	Make only small changes to the visuals.
44	Last check	Make sure the project is running and all needed links are working, also in documentation.
45	Submit the project	

Database Schema

The project uses document oriented database structure and MongoDB Atlas database service. Classapp database in MongoDB consists of the following collections.

- users
- series
- classes

Users are separated from the classes, to avoid repeating data. The same is the case with series.

1. users collection contains user documents with the registration data, namely username, email and password. The data is used to identify the user and as a permission layer before entering the users personal class archives. Username and user id are used as reference points to and between other collections. Users can have many classes and series.

2. classes collection contains class documents with all data of individual classes. Embedded into a class document are exercises and logs. Within an exercise object are embedded links to music tracks and videos. Username and user id are used as reference points to other collections. One class can belong to many series, but to only one user.

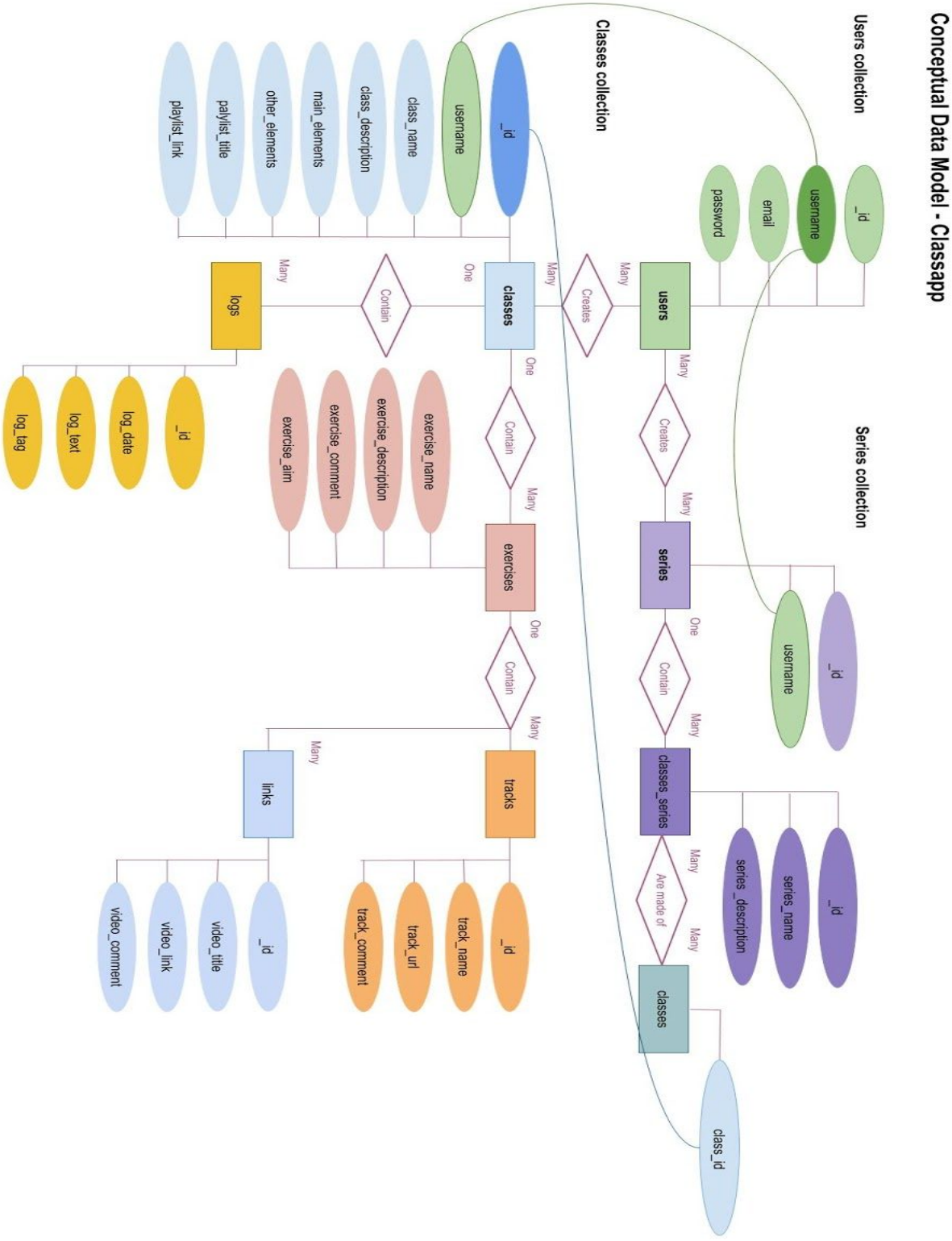
3. series collection contains series documents. Embedded in a series are class series. Within a class series is embedded a list of associated classes. A series can be associated with many classes, but only one user.

The embedded structure is justified by the cardinality of the relationships. The class data will need to be accessible only in the context of an individual user. The exercises data will need to be accessible only in the context of the class they belong to. The music track and video data will need to be accessible only in the context of the exercise they belong to. Document data is also not expected to grow ad infinitum. Denormalized data is a MongoDB recommendation in this case.

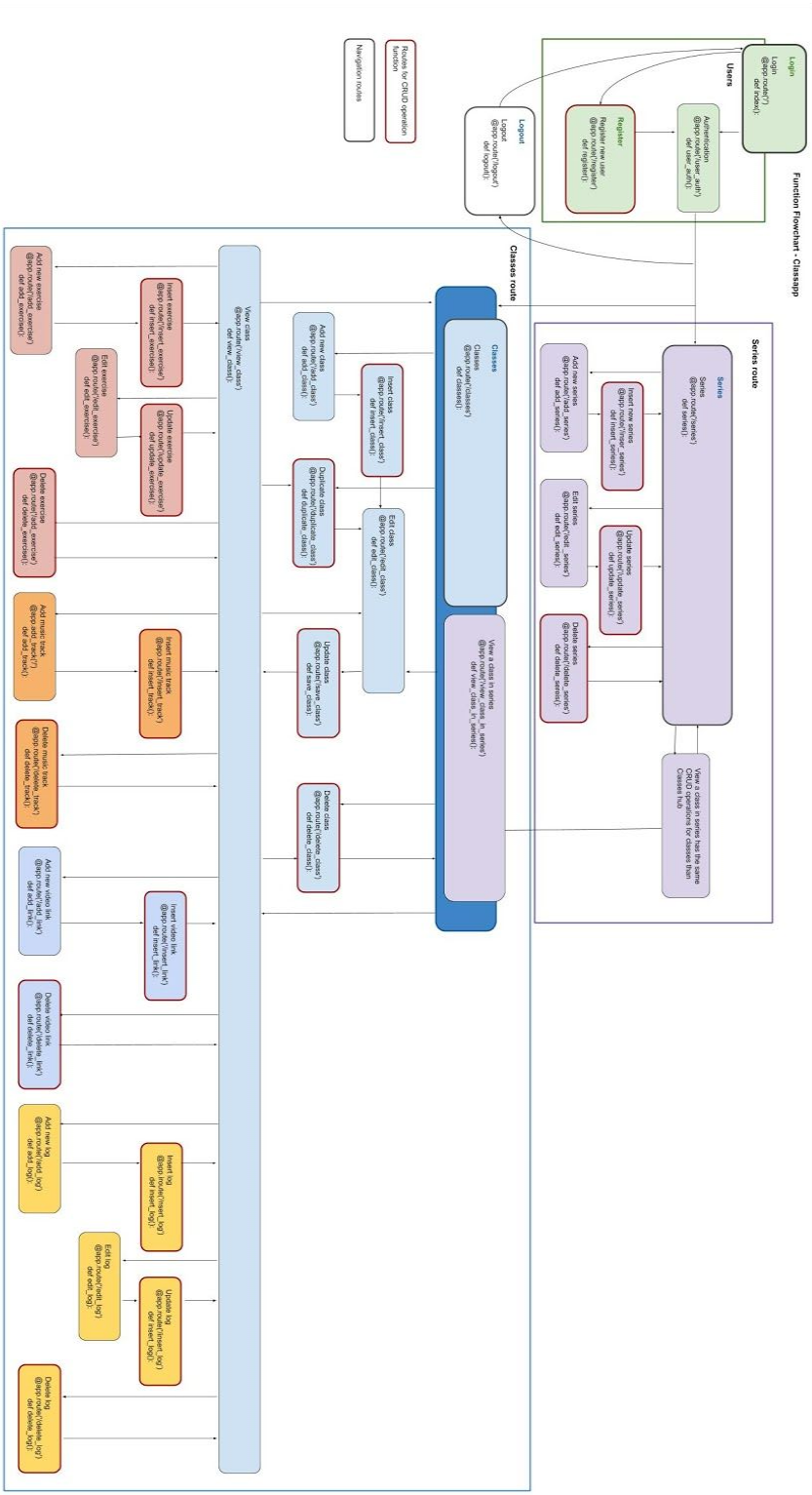
The series is really used only as a filter for document reading and to create perhaps temporary association. They also represent a clear many-to-many relationships regarding the classes. According to MongoDB documentation, data denormalization would not be required in this case. It also makes accessing the data and managing the documents less complex in practice.

Reference:
https://docs.mongodb.com/manual/core/data-modeling-introduction/

Conceptual Data Model



Function flowchart



Data sample per collection

Users

```
_id:ObjectId("5e568ade0938db6d0519c69f")
    username:"userOne"
    email:"one_user@time.com"
    password:"pdkdf2:sha364..."
```

Classes

```
_id:ObjectId("5e519eb9ba2dfb000006dfe1")
    class_name:"Monday Groove"
    class_description:"Let it loose and sweat it out!"
    main_elements:"Shaky shake moves"
    other_elements:"Shovel the toes, Slide with grace"
    playlist_title:"Billie Eilish"
    playlist_url:"https://www.youtube.com/watch?v=xNV38nq1fqc"
    class_notes:"This class makes you SWEAT, bring water and a towel"
    exercises:Array
        0:Object
            _id:ObjectId("5e51a370ba2dfb000006dfe6")
            exercise_name:"Get it going"
            exercise_description:"Wiggle the toes, while you touch your nose"
            exercise_comment:"Slow cooking makes the best stew"
            exercise_aim:"Get your toes moving, while maintaining focus to that which is the nearest to you"
            tracks:Array
                0:Object
                    _id:ObjectId("5e51a3ebba2dfb000006dfe7")
                    track_title:"Billie Eilish/ Bad Guy"
                    track_link:"https://www.youtube.com/watch?v=DyDfgMOUjCI"
                    track_comment:"It ain't salsa, but... I like Billie Eilish, ok"
            links:Array
                0:Object
                    _id:ObjectId("5e51a454ba2dfb000006dfe8")
                    video_title:"Say salsa/ Quick Feet Studio"
                    video_link:"https://www.youtube.com/watch?v=vfpajx2uemE"
                    video_comment:"The Quick & Dirty Guide to Salsa, Part 1. All the lessons put together..."
    logs:Array
        0:Object
            _id:ObjectId("5e51a32dba2dfb000006dfe5")
            log_date:"07/01/2020"
            log_text:"Was a wonderful class. Remember to start slow and ALWAYS smile"
            log_tag:"1st class for DanceFloor afternoon group"
            user_id:"5e568ade0938db6d0519c69f"
            username:"userOne"
```

Series

```
_id:ObjectId("5e519eb9ba2dfb00000c35fdf")
    user_id:"5e568ade0938db6d0519c69f"
    username:"userOne"
    class_series:Array
        0:Object
            _id:ObjectId("5e51a370ba2dfb00001a90of")
            series_name:"Mondays for advanced in DanceFloor"
            series_description:"Advanced classes for the afternoon group. Challenge level 10"
            classes:Array
                0:Object
                    0:"5e519eb9ba2dfb000006dfe1"
```

Features

Accessing the app

Users will access the application via the index page. At Index users can either Login or via the navigation bar link access Register page for registration.

Feature list

- Responsive navigation bar, with navigation links to Register and Login
- Login form with input fields for username and password, and a submission button
- Registration form with input fields for username, email, password and confirmation password and a registration submission button.
- Flash message to give user feedback and instruction about login status and how to create username and password
- Addition instruction feature for the password via information button embedded into the registration form
- Information card which can be opened with the information button and closed with a close button in the information card
- User authentication and user input validation for the username, email and password

After either Login in or Registering, a user is directed to the Home page. In the navigation bar a user has access to two separate routes to manage his/her class archives, and to create classes and class series. Route one is the path from the Home page, and route two is the path through Series. Also Logout is available after Login/ Register at all times. The Classapp log navigates the user back to the Home page.

Route 1/ Home page

Classes view and adding classes

Route one starts from the classes Home page, which is an access hub for the individual classes in classes collection and allows a user to add new classes using an add button. Clicking the add button opens an Add class form with empty fields. Users can fill in a form and save the content or cancel and go back. Submitting a new class with the Save button will create a new class document in the classes collection and takes the user to the class view page. Cancel button takes user back to the Home page

Feature list

- Accordion to display the classes if there are any created yet.
- Message if the classes list is empty.
- Add a new class button that takes the user to the add class form.

- Add a new class view with a form with following fields: Class name, Class description, Elements, Helpers, Notes, Select class series selection field, Add A Playlist Title and Add A Playlist Link.
- Elements text area is a Summernote editor, which has the following feature buttons set for the application as default: button to change text style, buttons to make the text bold, italic, or underline, undo and redo buttons, ordered list and unordered list buttons and a button to enter the full screen editor mode. The editor also allows dragging and dropping images and text.
- Save button and Cancel buttons.

Other options

Hovering over the accordion highlights the classes. Users can view a class in accordion list by clicking the accordion header. This reveals the class description and a master tool button at the right down corner of the page and highlights the selected class header. Users may click the description field to view a class. Alternatively, hovering over the tool button, reveals a set of options to choose from: View class, Edit class, Delete class and Duplicate class.

1. View class directs the user to the class view page.
2. Edit button directs the user to the edit class page.
3. Delete button deletes the class and returns to the classes page.
4. Duplicate class creates a duplicate of the class content and opens the class in edit class view.

Feature list

- Accordion element
- Highlighting the accordion headers.
- Highlighting the accordion content when hovering.
- Highlighting the accordion headers when open.
- Content field navigation to the class view.
- Master tool button.
- Tool buttons in master button.

Class view

Class view displays the class content:

1. The page header displays the name of the class the user is viewing.
2. A mastre tool button at the right hand bottom of the page with tools to add log, add exercise, edit class, delete class and duplicate class:
    2.1. Add log directs the user to the add log page
    2.2. Add exercise directs user to the add exercise page
    2.3. Edit class directs user to the edit class page
    2.4. Delete class deletes the class and returns to the classes page

2.5. Duplicate class creates a duplicate of the class content and opens the class in edit class view.

3. A button to navigate back to the classes view at the left side bottom of the class content.
4. Element content is created with the Summernote editor and is wrapped inside an accordion element. It can be opened or closed as it might contain a relatively large amount of information. The header highlighted when hovered over and stays highlighted when opened.
5. The series the class is associated with are displayed in collection elements and can be clicked to go to the series view.
6. Playlist is presented in a collection element and can be clicked to go to the playlist. Playlist opens on a blank page.
7. Exercises are displayed in an accordion element, which can be opened and closed.
   7.1. Clicking an exercise accordion header opens the exercise. The header highlights when hovered and stays highlighted when open.
   7.2. Exercise content is displayed in the form of accordion, so that each section can be opened or closed to save space. The first element, exercise description is open when exercise is opened. The headers highlight on hover and stay highlighted when open.
   7.3. Exercise description displays the content created with Summernote editor.
   7.4. Music tracks are displayed in collection elements and can be clicked to follow the link. Link opens on a blank page.
   7.5. Video links are displayed in collection elements and can be clicked to follow the link. Link opens on a blank page.
   7.6. Users can add Music tracks by clicking the add music track button and add video links by clicking the add video link button.
   7.7. Users can delete tracks and links by clicking the delete button in a collection element of each item.
   7.8. Users can edit exercise by clicking the edit exercise button.
8. Logs are presented in an accordion list. The accordion headers highlight when hovered and stay highlighted when opened. When opened logs display content created in Summernote. Users can edit logs by clicking an edit button at the left side bottom of the log content or delete the log by clicking the delete log button.

Feature list

- Page header with the name of the class
- Accordion elements for Elements, Exercises, exercise Description, Comment and Aim and Logs.
- Highlighting the accordion headers.
- Highlighting the accordion content when hovering.
- Highlighting the accordion headers when open.
- Content field navigation to the class view.
- Master tool button.
- Tool buttons in master button.

- Button to navigate back to classes
- Collection elements for Class series, Playlist, Music Tracks and Video Links.
- Buttons for adding Music Tracks and Video Links.
- Buttons to delete Music Tracks and Video Links.
- Buttons to Edit Exercise and Log
- Buttons to Delete Exercise and Log

Edit class view

Edit class view allows users to edit the content of prefilled input fields to make changes to a class. The Element field is again Summernote editor allowing the default editor options. Users can Save the changes to the class or click Cancel, which both direct the user back to the class view. The page header displays Edit - and the class name. Also the Exercises accordions are displayed, as well as Logs but for viewing only.

Feature list

- Page header with the name of the class
- A form with following fields: Class name, Class description, Elements, Helpers, Notes, Select class series selection field Add A Playlist Title and Add A Playlist Link.
- Elements text area is a Summernote editor with the default options.
- Accordion elements for the exercises and Logs
- Accordion elements for the exercises content
- Collection elements for the Music Tracks and Video Links in Exercise
- Save and Cancel buttons.

Duplicate class view

Duplicate class view is using the same form as edit the class view. The only difference is that the page header displays this time Edit - the name of the class (copy). Users can Save the changes to the class or click Cancel, which both direct the user to the class view of the copied class.

Feature list

- Header with the postfix (copy) added after the class name.
- Save and Cancel buttons.

Add exercise view

Add exercise view allows users to create an exercise for a class. The view displays an empty form for users to fill in. Users can save the exercise by clicking Save or cancel by Clicking Cancel, which both direct the users back to the class view.

Feature list

- A form with following fields: Exercise name, Comment and Aim.

- Exercise text area is a Summernote editor with the default options.
- Save and Cancel buttons.

Edit exercise view

Edit exercise view allows users to edit the content of prefilled input fields to make changes to an exercise. The Description field is again Summernote editor allowing the default editor options. Users can Save the changes to the exercise or click Cancel, which both direct the user back to the class view. The page header displays Edit - and the exercise name. Also the Music Tracks and Video Links collection elements are displayed, but the links do not work.

Feature list

- Page header with Edit - and the name of the exercise.
- A form with following fields: Exercise name, Comment and Aim.
- Elements text area is a Summernote editor with the default options.
- Collection elements for the Music Tracks and Video Links in Exercise
- Save and Cancel buttons.

Add log view

Add log view allows users to create a log for a class. The view displays an empty form for users to fill in. Users can select a log date from a date picker, tag the log for easier reference when later and write a log with a rich text editor. Users can save the log by clicking Save or cancel by Clicking Cancel, which both direct the users back to the class view.

Feature list

- Datepicker for selecting a date
- A form with following fields: Comment and Aim.
- Log text area is a Summernote editor with the default options.
- Save and Cancel buttons.

Edit log view

Edit log view allows users to edit the content of prefilled input fields to make changes to a log. The Log field is again Summernote editor allowing the default editor options. Users can Save the changes to the log or click Cancel, which both direct the user back to the class view. The page header displays Edit log from - and the log date. Also the Music Tracks and Video Links collection elements are displayed, but the links do not work.

Feature list

- Page header with Edit log from - and the log date.
- Datepicker for selecting a date
- A form with following fields: Comment and Aim.

- Log text area is a Summernote editor with the default options.
- Save and Cancel buttons.

Add music track view

Add music track view allows users to add a music track link to an exercise. The view displays an empty form for users to fill in. Users can save the track link by clicking Save or cancel by Clicking Cancel, which both direct the users back to the class view.

- A form with following fields: Music Track Title, Music Track Link and Comment.
- Save and Cancel buttons.

Add video link view

Add video link view allows users to add a video link to an exercise. The view displays an empty form for users to fill in. Users can save the exercise by clicking Save or cancel by Clicking Cancel, which both direct the users back to the class view.

- A form with following fields: Video Title, Video Link and Comment.
- Save and Cancel buttons.

Route 2/ Series page

Series view and adding series

Route two starts from the series page, which is an access hub for the class series in series collection and allows a user to add a new series using an add button. Series are displayed in accordion elements. Clicking the add button opens an Add series form with empty fields. Users can fill in a form and save the content or cancel and go back. Submitting a new series with the Save button will create a new series document in the classes collection and takes the user to the series view page. Cancel button takes users back to the series page as well.

Feature list

- Accordion to display the series if there are any created yet.
- Message if the series list is empty.
- Add a new series button that takes the user to the add series page.
- Add a new series view with a form with following fields: Class series name, Class series description.
- Save button and Cancel buttons.

Other options

Hovering over the accordion highlights the series. Users can view a series in the accordion list by clicking the accordion header. This reveals the series description and a master tool button at the right down corner of the page and highlights the selected series header. Users may click the description field to view the series. Alternatively, hovering over the tool button, reveals a set of options to choose from: View series, Edit series and Delete series.

5. View class directs the user to the class view page.
6. Edit button directs the user to the edit class page.
7. Delete button deletes the class and returns to the classes page.
8. Duplicate class creates a duplicate of the class content and opens the class in edit class view.

Feature list

- Accordion element
- Highlighting the accordion headers.
- Highlighting the accordion content when hovering.
- Highlighting the accordion headers when open.
- Content field navigation to the class view.
- Master tool button.
- Tool buttons in master button.

Edit series view

Edit log view allows users to edit the content of prefilled input fields to make changes to a log. Users can Save the changes to the series or click Cancel, which both direct the user back to the series view. The page header displays Edit - and series name.

Feature list

- Page header with Edit series - and series.
- A form with following fields: Comment and Aim.
- Save and Cancel buttons.

View classes in series view

View classes in series view is an access hub for the individual classes in series, which are displayed in accordion elements. Users can select a class from the series by clicking the accordion header, which opens the element and displays the cass description. Users can navigate back to the series view by clicking the Back to serie button at the bottom left of the View classes in series content.

Feature list

- Accordion to display the classes if there are any created yet.
- Message if the classes list is empty.
- Navigation button.

Other options

Hovering over the accordion highlights the classes. Users can view a class in accordion list by clicking the accordion header. This reveals the class description and a master tool button at the right down corner of the page and highlights the selected class header. Users may click the description field to view a class. Alternatively, hovering over the tool button, reveals a set of options to choose from: View class, Edit class, Delete class and Duplicate class.

1. View class directs the user to the class view page.
2. Edit button directs the user to the edit class page.
3. Delete button deletes the class and returns to the View classes in series view.
4. Duplicate class creates a duplicate of the class content and opens the class in edit class view.

Feature list

- Accordion element
- Highlighting the accordion headers.
- Highlighting the accordion content when hovering.
- Highlighting the accordion headers when open.
- Content field navigation to the class view.
- Master tool button.
- Tool buttons in master button.

Features and functions planned but not implemented yet

1. Incorporate class and exercises into the same form for creating and editing. Imply using embedded forms which technique I am not comfortable using yet. The class view would be ultimately turned to an edit form with a click of a button, which would reveal the edit buttons and turn the display fields into inputs.
2. For now only the registration form uses validation before sending the data to the server. Validate all input fields, especially for the input length. The summernote editor data is saved in html form, which makes it vulnerable for malicious intentions. The data should be checked for non html code for security and CSP compliance. Based on the conversation I observed on the Summernote Slack pages, currently there is no good ready out of box solution around which would accomplish this without interfering with the editors features.
3. Add an URL validator for the Playlist Link, Music Track links and Video Links.

4. Error catching for error 404 and 500. Error pages with navigation back.
5. Error handling for database not found.
6. Error handling for no Javascript/ jQuery available.
7. Incorporate YouTube API for playlists, music tracks and videos
8. Add a smiley API for the Summernotes editor to provide teachers more diversity when they are describing and notating their classes and exercises.
9. Make navigation button link dynamic, so that the user is directed back to the route they come from to the class view. Now the user is directed always back to classes, even when they actually arrive from the series route. This might feel a bit confusing and unpractical.
10. The application uses both Materialize and Bootstrap. This can currently potentially cause some overlapping and redundancy, which might slow down the performance. Bootstrap has a No conflict feature to remedy namespace collisions, which should be implemented where needed.
11. Summernotes editor now relies on Summernote with Bootstrap library. I discovered a conflict with the editor view using Materialize and Bootstrap together when building and testing the forms. Current setup was the best fit and works. However, the buttons are missing the tooltips. I had to turn them off, as they did not work properly. This should be solved somehow, if the culprit for the problem can be found and remedied with css or js. Otherwise one of the libraries should be dropped or the editor changed.
12. There should be two sets of flash messages, some which have to be closed manually (as is the case with flash messages at the moment) and others which fade out automatically.
13. Add search and organize features for the Classes and Series, so that the users can search for classes by name or arrange the list alphabetically or by date created. Mongo DB uses ObjectId which can be used as a timestamp.
14. Add a landing page with call to action content for a commercial version.
15. Add About page containing user manual and information about the application.

UI

Since the application is meant for creating and archiving classes the views mainly consists of different kinds of forms and displays of user input data. The elements, such as buttons, collapsibles and collections showing the data should be clearly structured and convey their purpose and meaning clearly. The layout should also be simple and give space for the teachers own creative process.

To help with the visual outlook and responsiveness, I will mainly use Materialize library. Summernotes rich text editor will be embedded in some form fields to give the user more options and creative freedom as the editor maintains the input outlook when displayed back to the user. Editor also enables the user to drag and drop images and other content they might see useful for their teaching.

The color scheme will be simple. Only the Login and Register page will have an image, and I will use background color to give the site a clear identity. The background color is incorporated to the logo element in the navigation bar as well, to create visual unity. The elements will have a bright and informative color scheme fitting to their purpose. Navigation elements on pages will have a written text telling where they take the user. Editing buttons will have indicative symbols to convey their purpose, and tooltips will be used to give users further information about the element's utility. The elements repeat throughout the pages to help users learn how to navigate the site and use its features.

Buttons

Master tool button will be a floating Materialize button, where the option tools are embedded. There are seven different groups of buttons.

1. Buttons to edit Classes are all round and have an indicative icon.
2. Buttons to add Classes and Series are rectangular with indicative text and icon.
3. Buttons to edit Elements in classes (Exercise and Logs) and in Exercises (Music Tracks and Video Links) are all rectangular and have an indicative icon.
4. Buttons to add Classes and Series are rectangular with indicative text and icon.
5. Buttons to Save and Cancel are rectangular and have indicative text.
6. Buttons to navigate back are rectangular and have indiactive text and icon.
7. Buttons to Login and Logout are rectangular and have indicative text.

Accordions

Materialize Accordion elements will be used to display the classes, series and classes in series for selection.

In classes view accordion will be used to display Exercises and Exercise’s content in classes.

Accordion elements will be used to display any elements which display Summernote content.

Collections

Materialize Collection elements will be used to display selectable elements (Playlist linsk, Music Track, Video Link and Series the class is associated with).