SEI Project 2 - ReadMe

Description

SEI Project 2: Build an Express App using MongoDB! The aim of this group project was to build a site using Express framework and MongoDB as a NoSQL database. This was completed during Week 6 of 12.

Our chosen project is an ‘Album Review Express App’ - a site where users can add and then review music albums, and view others’ reviews.

Deployment link

```
<deployment link goes here>
```

Getting Started/Code Installation

If you would like to contribute, please fork and then submit a pull request.

Please see `package.json` for required dependencies and install with `npm`.

Timeframe & Working Team

The timeframe for this project was approx. 1 week: planning starting 08/09/22, build starting 09/09/22, project to be presented 16/09/22.

This was a paired project, the working team was Milos Jocic and I (Harry Philpotts). For this project, I took the role of Team Leader.

Technologies Used

- Languages: HTML, CSS, JavaScript, EJS, jQuery.
- Written using Visual Studio Code.
- Express framework, Mongoose middleware, Node.js runtime environment.
- Database: MongoDB / MongoDB Atlas.
- Other key dependencies: `bcrypt`, `express-ejs-layouts`, `multer`, `passport` (see `package.json` for full list).
- Planning and management: Figma, Trello.
- Project hosted on Heroku.

Brief

General Requirements:
Build a web application from scratch, must be your own work.
Use Express framework to build your application
Deploy on Heroku so application is live on the web
Create a README.md file that explains your app to the world

Technical Requirements:
User resource
User must have a profile
User must be able to edit their profile
User must be able to change password

Authentication:
User must be able to sign up
User must be able to sign in
User must be able to sign out

2 extra resources of your choice (other than User):
User must be able to create a resource
User must be able to edit a resource
User must be able to view all resources they created
User must be able to view a single resource they created
User must not be able to edit or delete other users' resources

Stretch Technical Goals (optional):

Make application responsive
Use a CSS library like Bootstrap
Add extra resources
Allow users to upload image files

Planning

We began the project with an idea for a 'DIY' app, where users could add instructions, and then upload projects linked to these instructions.

We then 'wireframed' our key pages and completed an ERD for the project here on Figma.

Our project direction changed significantly on Day 2 of building where we decided to build the final 'Album Review App', with 'instructions' and 'project' resources being replaced by 'albums' and 'reviews' respectively.

We set up a Trello board in order to organise workstreams and track progress.

Production

Day 01 - Production:
09/09/22

App production started. *Main focus today: basic app structure*.

GitHub Repo established and successfully forked & cloned.
Now setting up server config, installing key dependencies:

```
Express
mongoose
EJS
dotenv
express-ejs-layouts
```

Added MVC structure and other directories.

Split workstream into layouts/landing page and models. I will be creating models files:

- User model added - possibility of changing firstName/lastName to screenName if wanted.

- Post model added.
    - May need to check if image datatype should be `string`.
    - Also, I have decided to do away with difficulty property and instead access this from the relationed instructions database
- Instruction model added - needed further commit due to naming error.
- Milos completed layout.ejs including link buttons.

Had a few merging difficulties - discussed with Ana, we are to send any error logs across in future.

Milos building landing page and route etc.

I am now moving to setting up 'Create Instructions':
- added `add detail edit index .ejs` files, empty for now. Moving to `add.ejs`.
- `add.ejs` completed, *GET API* and controller set up. Had been missing `app.set('view engine', 'ejs');` from `server.js`. Now tested working ok.
- now *POST API* added, tested working ok.

First significant merge conflicts, on `server.js` *as expected, due to parallel edits. Eventually resolved through VSCode merge editor.*

Landing page integrated into auth views/ctrl/route.

Milos working on signup functionality.

I am now on *Read*, *Update* and *Delete* functionality for Instructions:
- Instructions index and detail read routes all working ok!
- Edit functionality is working ok, also populates with previous values.
- Delete functionality working ok.

Day 02 - Production, Rewrites:
12/09/22

Milos has added the majority of `auth` functionality over the weekend. Pair coded in order to debug, tested working ok.

Significant change made to project: we have decided to change direction in order to make an Album review app instead.

Completing rewrites: workload split between editing all directly 'instructions'-related files/folders and editing all other references to Post/Instruction outside of this.

Cue judicious use of `find + replace`...*taking great care to match case / singular or plural.*

Edits completed on both sides, attempting merge:
- *Eventually resolved one expected conflict in routes - we found the GitHub GUI to be preferable to CLI or VSC Merge Editor*

- Changes debugged using pair programming, latest changes pushed.
-

Next two workstreams: Review views & create review functionality, dynamic 'add input box' functionality to add track listing.

Firstly, I am attempting to achieve 'add input box' functionality through embedded 'vanilla' JS, using DOM manipulation:

```
     <%  const newInput = document.createElement("input");
         newInput.setAttribute('type', 'text');
         newInput.setAttribute('name', 'trackList');
         newInput.setAttribute('class', 'form-control');
         const parentElement = document.getElementById('track-parent');
         function newElement(){
             parentElement.appendChild(newInput);
         }
         document.getElementById("add-field").addEventListener(click,
 newElement);
     %>
```

Did not work, `document is not defined`, googling turned up: "*You can't use document inside your ejs tags because that code is executed on the server.*"

Using similar code but as frontend in `main.js` I am able to add an input box, but no more than one. On a positive note, both input boxes created accept and pass values correctly!

I'm now going to try *jQuery*:
- *Tested working, although the button floats above any added inputs!*

…and fixed with another div.

Tested working ok. Replicating functionality for Edit page:
- Done and tested working ok: input fields populate with values, add input fields working and updating in DB.

Views, paths, controllers and routes for Reviews well under way: *pushing to finish before close of day*.

Day 03 - Production:
13/09/22

Got `'/'` working - if not signed in, this brings a landing page. If signed in, this goes to reviews home.

Now I am updating the review title so that it populates from the album list. Completed after some difficulty. Needed to use `.populate()`...!

Two workstreams: the reverse of the above, where albums are updated with attached reviews; and reviews have a user attached based upon currentUser. Album update implemented as below:

```
review.save()
.then(() => {
    req.body.album.forEach(album => {
        Album.findById(album, (error, album) => {
            album.review.push(review);
            album.save();
        })
    })
    res.redirect("/review/index")
})
```

User attached to reviews by Milos, using currentUser in hidden input field as follows:

```
<input type="hidden" name="createdBy" value="<%= currentUser._id %>" />
```

User firstName and lastName combined into username.

Linked reviews within albums eventually linking to usernames.
This part required a solution from Saad, specifically:

```
Album.findById(req.query.id).populate({
    path: 'review',
    populate: {
      path: 'createdBy',
      model: 'User'
    }
```

Due to the fact that `createdBy` was set as an array, we required the following in `album/detail.ejs`:

```
<div> Reviewed by: <a href="/review/detail?id=<%= review._id %>"><%=
review.createdBy[0].username %></a>, rating: <%= review.rating %></div>
```

This also includes a hyperlink to the review document!

Splitting workstream into protecting routes with IsLoggedIn and IsCorrectUser, and basic CSS formatting.

Basic formatting completed using bootstrap and CSS on `layout.ejs`.

Milos: `IsCorrectUser` removed, instead 'edit'/'delete' buttons are hidden to non-creators using EJS.

Further formatting done across album index, detail.

Day 04 - Production:
14/09/22

Further formatting work done on reviews pages. Further links added, paths adjusted to make app more user-friendly.

Milos working on *Cloudinary* Image uploads. Ultimately decided to go with *Multer* - now up and running.

Formatting continues site-wide. Milos working on Bootstrap Cards for Review index.

Very pleased with this code snippet which dynamically punctuates the ‘genre’ array:

```
    <div><span class="med-text
highlighted-text">Genres:&nbsp;&nbsp;</span>
        <% let counter = 0;
        album.genre.forEach(function(genre) {
        counter++ %>
        <a href="">
            <span class="med-text highlighted-text">
            <%= genre %>
        <% if (counter < album.genre.length) { %>
            <%= ', ' %>&nbsp;
        <% } else {%>
            <%= '.' %>
        <% } %>
            </span>
        </a>
        <% }) %>
    </div>
```

I attempted to set a transparent background...without success. I have decided to focus instead on mobile responsiveness.

Day 05 - Final day of production:
15/09/22

A challenging day today for both of us:

Getting cards working through Bootstrap caused us such issues that on Ana's advice we made the call to switch the cards to ‘vanilla’ CSS. I think the issue lies with a) conflict between main.css styles and Bootstrap styles, and b) what we wanted was outside the scope of Bootstrap customisation.

The good news is that Milos has now got the cards working, looking good and responding to screen size changes.

On my side, I had a good start to the day, with great progress made on mobile responsiveness. This was largely achieved through removing CSS positioning and leaning on *Bootstrap* in order to resize.

In some cases, this was not enough, esp. on tables. In order to get around this, I used `` `display: hide` `` classes within a media query in order to hide unneeded information when viewed on a smaller screen.

Progress ground to a halt when we found that uploaded album art started disappearing. The cause was not clear which triggered a revert to earlier commits. This did not solve the problem - *not surprisingly as the code edited should not have affected images or Multer* - but meant losing a lot of work and time.

Progress further ground to a halt when attempting to merge branches / repos. As we had been editing in parallel - Milos working on the new review index view, I had been working on the old one - and I had made the mistake of editing in `master`, when we came to merge at the end of the day we saw conflict after conflict.

Finally at about 5:45pm (now the evening before presentation day) we managed to merge and host our code, albeit with minimal content.

Day 06 - Presentation
16/09/22

I'm making minor (and I mean **minor**) edits prior to presentation: added a couple of CSS classes and a `review.createdBy` EJS snippet to `/review/index.ejs`.

And of course we see problems: it turns out that when a new push is sent to heroku, album art uploaded since the last push is lost. Fortunately, we only lost a small amount of work. *And, crucially, we now should know what the problem is*!

I have promised **no more `master` edits or pushes** until after we present so that we don't lose any more content!

Initially, I had planned to clean up code in the `dev` branch, removing `console.log()` statements, taking out comments no longer needed, refactoring etc. *Ultimately, I changed my mind in the end and decided to leave things as they were!*

Presentation completed successfully!

A few hours after the presentation, the disappearing image issue happened again. Some research came up with the following from *stackoverflow.com*:

```
 I'm not sure why your uploads aren't being saved; you should be able to
```

```
save them temporarily.
But this won't work long-term. Heroku's filesystem is ephemeral: any
changes you make will be lost the next time your dyno restarts, which
happens frequently (at least once per day).
Heroku recommends storing uploads on something like Amazon S3. Here's a
guide for doing it specifically with Node.js.
Once you've stored your files on S3 you should be able to retrieve them
using an appropriate library or possibly over HTTP, depending on how
you've configured your bucket.
```

At least we finally know why the issue was happening! And crucially, the images remained stable for long enough to complete our presentation.

Challenges

- Managing git / GitHub forks and branches with multiple contributors for the first time.
- Avoiding conflicts / merging correctly when conflicts arise.
- Time management: finding the balance between committing time towards working together to solve issues, and to working and solving issues separately whilst the other person continues.
- Debugging backend code seemed much more daunting when compared to the frontend error messages we were already familiar with.
- Integrating image uploads, getting these uploads to stay on Heroku.
- Finding the balance between relying on Bootstrap for styling and adding CSS, resolving conflicts that arise between the two.

Wins

- Building confidence with Express framework, CRUD operations. Also confidence using EJS. Gaining further understanding around the concepts of ‘client-side’ and ‘server-side’.
- Building familiarity and confidence with working as a team rather than solo, working with git and GitHub
- Revisiting CSS and gaining better understanding as a result of this.
- Gaining experience of being in the ‘Team Leader’ role.

Key Learnings/Takeaways

- Git branching discipline is key! Things can go very wrong, very quickly when this falls down.
- Debugging as a pair can often be significantly quicker than working solo.
- Communication is crucially important, especially when avoiding unnecessary conflict during merges, and when avoiding duplication of work.
- It's fine to change your project idea and direction (if you do this early enough!).
- EJS is still JavaScript (albeit with some differences and limitations).

Bugs

- Uploaded images are lost from Heroku after < 24hrs. Alternative hosting (S3?) required.
- Upload image functionality on 'Edit Albums' not working.
- Sessions not consistently saved.

Future Improvements

- All image upload functionality fixed and stabilised.
- Full mobile-responsive formatting.
- Delete album functionality hidden, edit functionality protected.
- Spotify API for album info, artwork.
- Change 'add review' functionality where user can click 'add review' button on album.
- Search and filter by users, artists, genres.
- Refactor code and remove unneeded comments, `console.logs()` etc.
