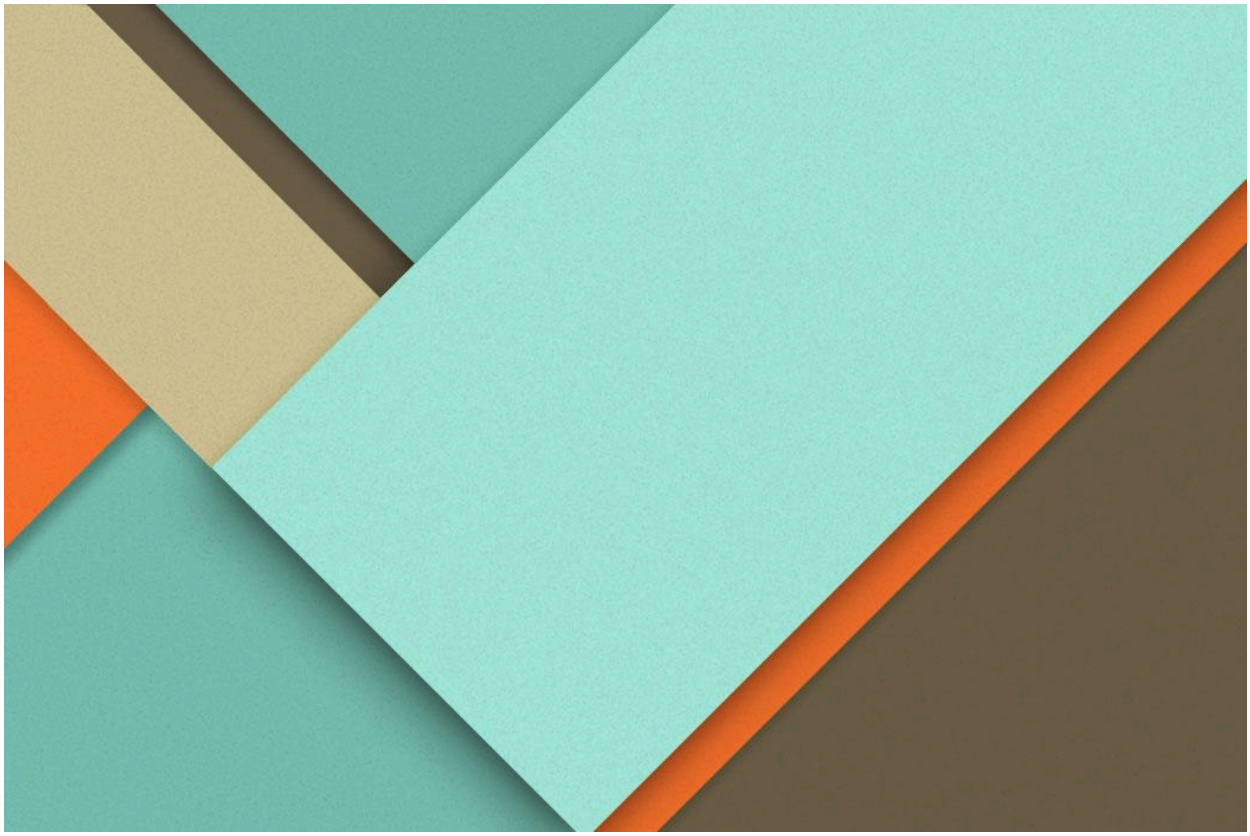

Autumn Assignment' 24

21.08.2024

—

4y
Information Management Group

Overview

Managing and reviewing assignments over multiple iterations can be a time-consuming process for both reviewees and reviewers. Currently, this entire process—including creating assignments, checking them over multiple iterations, and managing their status—is done manually using sheets and documents by reviewers and students alike. So...

As part of your Autumn Assignment this year, you will be building an in-house Assignment Review System (ARS) to streamline and manage this assignment-review cycle.

MVP Features

- **Authentication**: Users should be able to sign up and sign in through the conventional username-password auth.
- **OAuth**: Implement Channel I OAuth to allow Channel I based login in the application.
- **Roles**: A user can also be an admin, a reviewer, or a reviewee. You need to handle all these use-cases in your implementation.
- **Reviewer Role:**
  - **Assignment Creation:** A reviewer should be able to create an assignment along with the necessary attachments. These attachments will include pdf, docx, supporting images, links, etc.
  - **Assignment Sub Task Creation:** An assignment might have many sub tasks.
  - **Add reviewers:** The assignment creator will be the default reviewer for the assignment. A reviewer can invite other users to be added as reviewers for the assignments.
  - **Share:** Include share functionality so that an assignment creator can share assignments over mail.
  - **Allocation:** A reviewer should be able to allocate the assignment to users separately as well as in groups. The functionality should support hybrid group+individual allocation as well.
  - **Review:** A reviewer can review an assignment at any stage. There can be many reviewers for an assignment.
  - **Iterations:** A reviewer can review over multiple iterations, remember to include it in your implementation. Iterations may include comments for the reviewer.

not exactly sure what does this mean

- **Reviewee Role**
  - **Assignment Submission:** A reviewee should be able to submit an assignment. A submission might include comments and necessary attachments.
  - **Ask for Review:** Reviewee can tag reviewers any time after submitting the assignment.
  - **Resubmit:** Allow multiple submissions to support the iterative flow of review.
- **Admin Role:** Admin should be 'Admin' 🙂.
- **Access Rights:** Use Django permissions for managing roles and accesses. A single user can have multiple roles.
- **Team submission:** Implement a feature to allow team submission in an assignment.
- **User Sub Grouping:** Admin should be able to create sub groups which can be joined by reviewees. A reviewer can allocate the assignment to a complete group.
- **Realtime updates:** Implement real time data updation on frontend wherever necessary using Django websockets.
- **Profile:** A profile will include basic personal information, along with some more stats as mentioned below.
  - **Reviewer:** A reviewer should be able to see his/her review history, pending reviews, and other necessary stats.
  - **Reviewee:** A reviewee profile will include submission history, pending assignments, in-progress assignments, etc.
  - Reviewers can view the profile of reviewees linked to his/her assignment.
  - Admin can view anyone's profile.

Brownie Features

- **Authorization:**
  - **Email Verification:** You can further add email verification steps, using OTP for login purposes.
  - **Google OAuth:** Implement Google OAuth along with the Channel I OAuth.
- **Reviewer Role:**
  - **Assignment Reminder:** A reviewer can send reminder notifications for the assignment.
- **Chat:**
  - **Reviewer-Reviewee 1v1:** You can add a chatting system between the reviewer and reviewee.
  - **Group chatting:** Add a chat feature in the groups created by the admin.

- **Generalize IMG use-case:**
    - Generalize the use case to allow multiple workspaces, like your application can have many workspaces spreaded over multiple academic sessions so you are able to persist previous years' data. Manage permissions accordingly (a reviewee should be able to edit his/her own assignments and submission statuses after passing some ETEs 🙂). Using workspaces for academic years is just an example, workspaces can represent different clubs as well (another example).
- **Gamification:**
    - For encouraging them to complete assignments on time, gamify the portal. You can use your imagination and discuss with your mentors.

Timeline

Self Proposal Submission (23th August)

Submit your self proposals by 28th of August. Your proposal should include at least the sections given in this doc. You may also include wireframes or flowcharts to explain your idea in a better way.

DB Structure and Wireframes (6th September)

This is the very first and foremost step in your app development. You need to list down all the features in a document. Next, decide the database structure. For this, it is advisable to make ER diagrams. For better understanding, you can make the hand-drawn layout of the screens that you plan to build. All this has to be shared and will be reviewed by third years.

APIs / Backend (28th September)

Now comes the backbone of your app, the APIs. Read up on the Django Rest Framework. Needless to say, your code should be git tracked, and you will be evaluated on the progress that you make coding your API. It is expected that you follow all the specifications mentioned above strictly.

Frontend (27th October)

The most interesting and probably the most challenging part of your app. Once you are ready with your APIs, you should get started with the frontend. You are free to decide the design of your apps and make sure that it is simple to use. The hand-drawn screens made in the 1st part will come in handy and try to build on them.

Tech Specifications

1. Database Architecture

Use either PostgreSQL for storing your data and not SQLite. SQLite is a file system database and hence is not suitable for storing data in production. Make sure to do this else it will hinder your application flow and would make it difficult to evaluate.

2. API with Django REST framework

Tech stack: Django, Django REST framework

This will be the framework that should be used to write the backend part of your application.

3. Frontend:

Tech stack: React, Redux/Redux-toolkit

For the frontend of the application, you will be using React.js. For css libraries you can check out Material UI or Tailwind CSS, Bootstrap is for kids :)

4. Web-Sockets:

You can use Django Channels to implement Websockets in Django.

You can also use Websockets to implement a communication system between the reviewers and students. This approach will allow reviewers to instantly see the status of assignments, preventing multiple reviewers from assessing the same assignment simultaneously and at other necessary places.