

Kudos Project

Problem

As students and developers, our lives move fast, and accomplishments for our work often goes unnoticed. Since we are tasked with executing and delivering software on top of schoolwork and other responsibilities, it is easy to feel overworked and burnt out.

The Tech Team will create a tool that facilitates interconnectivity across project teams by allowing developers to leave feedback under each other's work during sprints.

Feedback can be positive or constructive, and will always be anonymous.

Features

The app consists of two main pages.

Features for page 1, the main page:

- a 4 x 4 card layout, where each card contains the following:
 - commit content headline
 - author
 - streak/trend info (if applicable)
 - a text input box
- a sidebar that allows users to navigate team archive. It has:
 - teams dropdown -> clickable sections to view the commits for each project team
 - team 1
 - team 2
 - team 3
 - archive dropdown -> clickable archives with commits and kudos from past sprints
 - 3/21 sprint
 - 3/14 sprint
 - 3/7 sprint 1

each of these components will have filter, sort, and search functionality

Features of page 2, the admin page:

Page 2 will have admin access only.

- dashboard layout where an admin can do the following:
 - **start a kudos session.** admins can open up the kudos board for developers to visit and interact with. the visibility of the board changes at the admin's discretion.
 - **review all comments.** admin can see comments on all commits (with username attached?) (identify user who made comment, but don't display content?)
 - **view comment analytics.** admin can see how many comments a developer has made by sprint or all time.
 - **set timed start and end.** admin can set a timer that will be displayed on the main page indicating when the next session will be. once a session is in progress, another timer will be displayed indicating when the session will close.
-

End-to-End User Workflow

1. General User (Developer) Experience

- Initial Access:

- The user opens the application in their web browser.
- The frontend makes an API call to the backend to fetch the latest commit data and active session status.
- The main page (4x4 card layout) is rendered, displaying commit headlines, authors, and any existing streak/trend information.
- The sidebar is populated with team and archive navigation options.

- Browsing Commits:

- The user can scroll through the 4x4 card layout to view different commits.
- They can use the sidebar to navigate to specific teams or archived sprints.
- Filtering, sorting, and searching functionalities allow them to find specific commits.
- The frontend sends the filter, sort, or search parameters to the backend via API calls, and the backend returns the filtered, sorted, or searched data.

- Leaving Feedback (Kudos):

- If a kudos session is active (indicated by a timer or clear visual), the user can enter feedback in the text input box within each commit card.
- When the user submits feedback, the frontend sends a POST request to the backend API, including the commit ID, user ID (anon), and the comment content.
- The backend stores the comment in the database, associating it with the commenter, the commit, and the author.
- (If using websockets, the comment is broadcasted to other users in real-time???)

- Viewing Session Status:

- The user can see the status of the current kudos session (active, inactive, time remaining) on the main page.
- This information is fetched from the backend and updated in real time if web sockets are implemented.

2. Admin User Experience

- Accessing Admin Page:

- The admin user logs in with their credentials.
- The frontend checks the user's role and redirects them to the admin page if authorized.

- Starting/Ending a Kudos Session:

- The admin uses the dashboard to start or end a kudos session.
- The frontend sends a POST request to the backend API to update the session status in the database.
- The backend updates the database and, if using WebSockets, broadcasts the session status change to all connected users.

- Reviewing Comments:

- The admin can view all comments on the dashboard.
- The frontend sends a GET request to the backend API to retrieve all comments.
- The backend retrieves the comments from the database and returns them to the frontend, possibly with user identification (for admin review only, not public display).

- Viewing Comment Analytics:

- The admin can view comment analytics, such as the number of comments per developer and per sprint.
- The frontend sends a GET request to the backend API to retrieve the analytics data.
- The backend performs the necessary data aggregation and returns the results to the frontend.

- Setting Timed Start/End:

- The admin can set the start and end time of a kudos session.
 - The frontend sends the start and end times to the backend via a POST or PUT request.
 - The backend stores the times in the database and uses them to control session status and display timers on the main page.
 - If web sockets are used, the time remaining will be updated in real time.
-

High Level Architecture

1. Frontend (Client-Side):
 - React/TSX UI
 - handling user interactions, and displaying data.
 - rendering the 4x4 card layout, sidebar navigation, and admin dashboard.
 - API calls to backend db.
2. Backend (Server-Side):
 - Django/Python for the APIs.
 - manages data retrieval from GitHub API
 - handles auth using GitHub API
 - implements logic for filtering, sorting, and searching commits and comments.
3. Database???:
 - MongoDB to store commit data, user information, comments, and session details.
4. Real-time Communication???:
 - websockets?

Outstanding Questions

should we store commit data? or just pull instances from API?

Revision #3

Created 7 April 2025 19:57:49 by Miguel Merlin

Updated 9 April 2025 21:43:05 by Emilio Cardillo Schrader