VoidCode

A live coding platform

Overview :-

Users login into VoidCode using email, password. Alternatively we can implement Google+ login or Github Login.

Users land on the dashboard, with collaborative projects listed. Opening any one collaborative project should make the user land on an editor cum folder explorer view, with their project code loaded up. Each dashboard will be loading the project from any source control (git, scm etc)

Live Users can edit their code simultaneously and run accordingly. Offline collaborators of the project should be receiving notifications or alerts on the changes and being made by whom.

DB :-

- **Users Table/Document** - Containing personal details of users along with crypto hashed passwords and map of dashboard PKs (1-m mapping with Dashboard DB)
- **Dashboards Table/Document** - All existing Dashboards along with a repository link and Notification topic. (1-1 mapping with Repository DB, 1-1 mapping with Notification DB)
- **Repositories Table/Document** - Contains all user created Repositories links
- **Notifications Table/Document** - stores the topics related to each dashboard and user ids in a map subscribed to it
- **Message Document** - contains a unique id, involved user ids and messages as json

Login Functionality :-

- Users sign in through username and password.
- On Successful login, we create a session by storing a SSO token in Frontend and validating each network request based on the token. The token will have an expiry.
- Requests must be driven through a SSO token proxy layer which will validate the requests or force sign out in case of invalidity.

Live Coding, Inviting Collaborators & Saving Work :-

- New users can create a dashboard, which will make them a subsequent blank repository in github or any other source control.
- Users can invite other collaborators (already registered users) through providing a link through their email - the link should contain encrypted dashboard id(pk) and also the invitees user id. Accepting (and clicking) the link would call an API which would :
    - Extract the dashboard id and user id and push it to the dashboard map under the user.
    - Provide access to the repository mapped with the dashboard.
- Online users can live code together. Opening any dashboard creates a socket connection, streaming changes from all other connected users and vice versa.
    - The socket serves the files in the repository to the online editor. It can support Open Request, Write Request, Commit Request. Each file opened will trigger a separate Socket connection.
    - Save and Run functionalities are User centric and run on frontend and user centric devices. Saving on users own device dosen't commit the changes into repository
    - Line based Locking - All live cursor positions of other users will be pushed by the socket to the frontend and will not allow editing on the lines which the cursors are. This is to eradicate conflict.
    - Users can run the code in their terminal provided to them, the terminal and editor and source control is part of a sandbox which contains the compiler too. (can be changed into a backend based code run as we have for online compilers. Backend based code runs will not pertain to the whole projects and only single files which limits use case).
    - Post all changes, users may choose to commit their code into their source control, which would save the work in the repository.

Line Locking and Simultaneous editing :-

Line locking is a necessity when implementing live coding platforms :

- cursors of other online users will be displayed to the user in real time and will disable the lines which they are. Every cursor position, characters typed will be streamed and broadcasted to other online users in real time.
- Cursors of offline users will not be considered.
- Cursors immediately on the next line adjacent to any other live user will be pushed one line down on encountering a newline anywhere across the file.
- All the above data will be pushed from Socket to the users frontend. Any changes from the user's system will be sent to the socket to be broadcasted. Data to travel in bytestreams.

Notifications, Alerts and Messages :-

Notifications will be pushed to offline users on registering any new commit on the repos they are collaborating on.

- Each dashboard creates a Notification topic, all collaborators if notifications enabled will be subscribed to the topic
- Users DB stores the preference - isNotificationEnabled
- Any hits on the Commit Command/API will trigger Notification service with Dashboard ID, triggering a Notification to every user subscribed under the topic.
- Message boxes create their own Socket connection if both users are online or push messages through API if one person is offline. Updates the Message Document

Data & Object Flow Diagram :-