Team Meeting 5 Notes

Meeting minutes: 30

Place: virtual

Date: Oct 15th, 2024

Time: 8:30 pm to 9:00 pm

Notes:

* Discussed the reworking of the current sprint to complete the following epic

Client Meeting 2 Notes

* Backlog suggestions

Add more columns, have ceremonious meetings to discuss what exactly will be in the descriptions of the tasks in jira.

Mohand puts heavy emphasis on user stories

Tickets are almost empty aside from the title, descriptions need to be more accurate and precise

Tickets need to be shorter, no more than 2 days. Few days before the sprint, provide an agenda of what were going to present and whos going to present it

History for tickets needs to be implemented

Add tags for our tickets (UI, backend, etc). Makes it easier to keep track of tickets

Version control for our work (so we use the same version for our demo)

* Things clients want for next meeting/future

What other documents we have: project charter, architecture document (work in progress) to discuss technical aspect, requirements document to outline user stories/requirements

We need to introduce our client to the professor, and before that, we should ask the prof if he's okay to discuss success criteria with our clients

Add an agenda so they can add what they want to see in the meetings

Project dashboard so they can intervene whenever they see fit.

Send wireframes to show clients our progress/integrations for feedback

Collaborate on the wireframes (then use wireframes to demonstrate how we want to integrate features to the client)

* Miscellaneous

Don't take what they say too literally, take it as a nudge.

We should use our tasks in jira to organize our presentations in the future

We need to consider how our clients can help us (guidance/questions)

Vscodium? Same core as vscode but we should all be using the same IDE regardless

Next week: wireframes, documentation, architecture (including datamodel?).

Now we should pause, focus on our user stories, wireframes, fill our backlog, put more content in the backlog. The tester needs this to produce test cases

Figma? Tonnes of features to help generate our UI, increases productivity

JIRA Epic: User Authentication System

Epic Name: User Authentication System

Epic Summary:  
Implement a secure user authentication system for the application, using Google Sign-In with Gmail as the primary authentication provider. This epic will cover the backend and frontend integration, allowing users to sign up, log in, and log out securely. Additionally, user data will be stored in a PostgreSQL database, and the authentication process will be seamless and efficient.

Epic Description:  
The purpose of this epic is to provide users with a robust authentication system using Google OAuth for Gmail. The system will ensure that only authenticated users can access specific parts of the application, with their data securely stored in the backend. The goal is to complete the entire user authentication flow, from frontend UI to backend processing and database storage. This includes the integration of Google OAuth, the setup of user sessions, and the management of access tokens.

* Expand to include home page

Acceptance Criteria:

* Users can log in with their Gmail accounts via Google OAuth.
* Only authenticated users can access protected routes or features in the application.
* User details (email, name, login status) are stored securely in PostgreSQL.
* Proper error handling for failed login attempts.
* Users can log out and terminate their sessions.

User Story 1: Frontend - Google OAuth Integration/User Email (Ghait)

* Summary: As a user, I want to log in using my Gmail account via Google Sign-In, so that I can securely access my account.
* Description: Implement Google OAuth on the frontend using Next.js. Upon clicking the "Login with Gmail" button, users should be redirected to the Google authentication page. Once authenticated, they should be redirected back to the application.
* Acceptance Criteria:
  + A "Login with Gmail" button is available on the login page.
  + The user is redirected to Google's authentication page.
  + After successful authentication, the user is redirected back to the application and sees their account details.

User Story 2: Backend - Handle Google OAuth Tokens (Ghait)

* Summary: As a developer, I want to securely handle Google OAuth tokens on the backend, so that I can manage user authentication.
* Description: Implement a Spring Boot service to handle the receipt of OAuth tokens from Google. The service will validate the token, create or update the user in the PostgreSQL database, and establish a session for the user.
* Acceptance Criteria:
  + The backend receives the OAuth token from the frontend.
  + The token is validated using Google’s API.
  + The user's details (email, name) are stored in the database.
  + A session is created for the user, and access tokens are handled securely.

User Story 3: Frontend - Manage Session and User State (Jazia)

* Summary: As a user, I want to remain logged in after authentication, so that I can navigate the app without having to log in again.
* Description: Once a user logs in, manage their session state in Next.js. Ensure that users remain logged in during their session and can log out when desired. Store the session token and display the user's name or email on the dashboard.
* Acceptance Criteria:
  + After logging in, the user remains logged in until they log out or the session expires.
  + The user's name or email is displayed on the dashboard.
  + A "Logout" button is available that terminates the session.

User Story 4: Backend - Logout and Session Management (Isam)

* Summary: As a user, I want to log out of my account, so that my session is securely terminated.
* Description: Implement the logout functionality on the backend. When a user logs out, their session should be invalidated, and they should be redirected to the login page.
* Acceptance Criteria:
  + The user’s session is invalidated when they click the "Logout" button.
  + The user is redirected to the login page after logging out.
  + No protected routes can be accessed after logout.

User Story 5: Secure User Data in PostgreSQL (Kian)

* Summary: As a developer, I want to securely store and retrieve user data in PostgreSQL, so that user information is persisted across sessions.
* Description: Implement a PostgreSQL schema to store user data, such as email, name, and login status. Ensure that sensitive information is not stored directly and that data is encrypted where necessary.
* Acceptance Criteria:
  + User details (email, name) are stored securely in PostgreSQL.
  + The database is queried to check if the user exists during login.
  + Sensitive data is encrypted where necessary.