CS 461: DesignDocument Activities

Prototype a web-based tool for creating and executing task-delineated, collaborative, AI-assisted assignments

Design Activity

Select a Database or Storage Technology:

Oracle

Pros:

- High scalability. Designed to handle high-volume workloads.
- Has many built in security features.
- Cross platform across Windows, Linux, etc.
- Good at storing predefined data types.

Cons:

- High licensing fees.
- Lack of open-source contributions/support.
- Used for large databases and has less application for smaller developers.
- Higher learning curve - None of our developers have much experience with Oracle.

MySQL

Pros:

- Open source, free software, good for small developers.
- Cross platform across Windows, Linux, etc.
- Good scalability for both small and large applications.
- Multiple open source plugins and solutions. Commonly used, so issues are easier to solve through community.
- Lower learning curve - All of our developers have experience with MySQL.

Cons:

- Certain features are locked behind MySQL's paid versions.
- Reports of concurrency issues that Oracle typically does not have.
- Can be very slow to alter large tables.
- In general can run slowly and can be difficult to manually modify tables.

Overall, I believe that our group will choose to use MySQL over the alternatives. The first and primary reason is simply that it is free and open source, meaning that we can use it without any issue for developing our software. The other primary factor to take into consideration is that all of our developers have had experience using MySQL previously due to taking courses previously during our OSU

curriculum. This means that we will have a level of familiarity with MySQL which can reduce the workload of understanding it and help development in other areas.

Furthermore, the primary alternative that I looked at was Oracle, which is commonly used by larger business models. However, as we are merely making a smaller-developed software, we are unlikely to need to store significantly large amounts of data for the time being. Furthermore, MySQL has decent scalability for both small and large projects, meaning that if our database is larger than expected, it is dynamic enough to handle our information.

Apply the Privacy by Design (PbD) Guidelines:

1. Proactive not reactive; preventive not remedial.
2. Privacy as the default setting.
3. Privacy embedded into Design.
4. Full functionality — positive-sum, not zero-sum.
5. End-to-end security — full lifecycle protection.
6. Visibility and transparency — keep it open.
7. Respect for user privacy — keep it user-centric.

1. During our implementation we will need to consistently conduct risk assessments. Our software should not gather any personal information or have any method to obtain sensitive information from our users that they themselves do not choose to share. Furthermore, we need to ensure any gathered data, such as student prompts and our responses are not accessible by third parties. We do not want to have to retroactively implement security measures, our software should be tested thoroughly beforehand.

2. During our implementation, we will gather as little user information as possible and ensure that user profiles cannot be publicly displayed or shown without permission. Professors will have access to student accounts but only be able to see information from them regarding their classes. Students only have to verify through name/email, and otherwise we will gather as little data as possible.

3. Our implementation will include encrypted communication channels. Fair information practices will be applied, and we will conduct rigorous security/risk assessments that will also be published to allow users/shareholders to understand possible security risks.

4. To ensure full functionality, our implementation will embed privacy in such a way that full-functionality is not impaired. For instance, our AI models will grade student submissions without gathering student information besides necessary associated accounts which will strictly and only be sent to the associated professor. Both students and professors will be able to delete or modify any information associated with their accounts at any time.

5. During our implementation, we will need to implement end-to-end security. We must ensure confidentiality of any gathered data to our users, such as through implementing data-deletion features after specific time periods, where such retained data is no longer necessary. In this way, we can ensure end-to-end security.

6. In our implementation, we must ensure users fully understand what data is being collected, and how and where it is being used through a privacy-policy. We must also ensure users can see what data has been collected from them, such as students being able to access and see what prompts are currently stored and shared with professors.

7. We must enable consent settings for our users so as for them to understand what of their data is being stored. For instance, we may allow students to hide their prompts from even professors (though professors will likely have knowledge that the specified student has requested their prompts be hidden from them). We want to ensure student privacy and safety at all costs, and we wish to ensure users have full control over all data that is being shared/stored.