Final Draft - AskMe Web Application

Team member: Jincheng Xu (Fr4nk), Jinyu Ding (Tina), Qi wu (Qi), Zhuoang Tao (Kasse1)

Pitch

“No more holding back, no more fear of judgment. Just pure honesty and pure fun! ”

The objective of AskMe is to provide individuals with a secure platform to pose personal or sensitive inquiries and receive responses, while maintaining anonymity. AskMe engenders a feeling of comfort and security, enabling individuals to express themselves openly and receive sincere answers.

By allowing anonymity, AskMe can also encourage people to ask more candid questions and provide more honest answers, leading to deeper conversations and better understanding between friends. Additionally, it can also serve as an entertaining activity, such as playing truth or dare games.

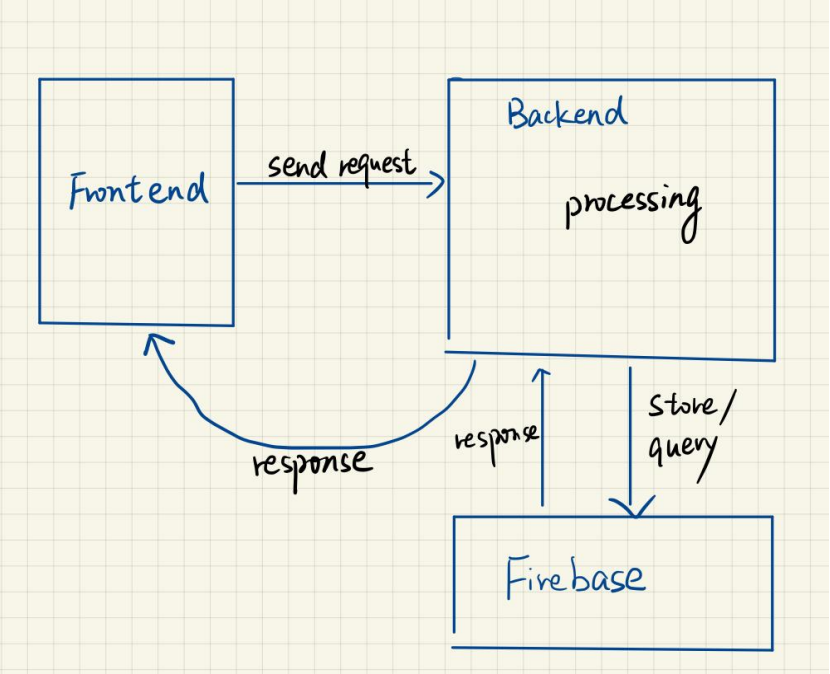
Functionality

1. Login page: Users can create their own account and login anytime.
2. Asking Page: Everyone is able to click the link to the question box shared by users to ask questions.
3. Answering Page / User Home Page: Users can see all questions in their question box and answer these questions.
4. Questioners can ask questions as guests without login.
5. Questioners are anonymous and their personal information is protected.

Components

- **Backend:** We will use Javascript and React framework to write our backend code, with Google Firebase.
    - JavaScript is a highly performant language and can handle a lot of concurrent connections, making it suitable for building scalable backend applications. Besides, JavaScript can be used on both the client and server side, allowing for a seamless transfer of data between the front-end and the back-end.
    - The React framework is built with components, which can be reused and combined to create complex UIs. Similarly, code written for the backend can also be easily reused and modularized, making it easier to maintain and scale the application.
    - JavaScript and React allow for rapid development and iteration, making it easier to quickly prototype and test ideas.

    - Firebase provides real-time data synchronization between clients, making it easy to create collaborative, real-time applications. Firebase's serverless architecture means you don't have to write any server-side code to get started. Firebase can easily scale to handle large volumes of data and traffic. Firebase can be easily integrated with other Firebase services, as well as third-party services like Google Cloud Platform.
- The backend should have the following responsibilities:
    - Store a list of user accounts
    - Store questions & replies from the question box
        - We will use Firebase to store questions & replies with corresponding questioners(anonymously, only visible to developers) and box owners.



- **Frontend:** We will use the HTML, CSS, JavaScript programming language and the React framework. We might use TypeScript as a transpiler but we haven't decided yet. We chose HTML, CSS and JavaScript due to its speed, simplicity, and popularity. The benefit of the React framework includes having potential to reuse components and easy to use for building a rich UI. We will test our functionalities using React Testing Library which provides light utility functions for us to interact with our React components so that we can experience as a website user and make meaningful updates.
- **Test:** We will use React Testing Library and any other necessary testing libraries like Jest or Cypress to test both our frontend and backend. We will configure the test environment to simulate the production environment as closely as possible. Then, we will write integration tests that test the frontend's interaction with the backend. These tests can simulate user

interactions and test the application's end-to-end functionality. By analyzing the test results and refine the tests over time, we will then be able to ensure the complete test coverage and application functionality.

Weekly Planning

1. learn basic knowledge of creating website
2. Create a login page and test
3. Create user homepage and test
4. Create question box and test
5. Create page with guest mode and test
6. Create database to store users personal information and corresponding questions data
7. Connect the frontend with the backend and test
8. Integrated Test
9. Record video for presentation

Potential Risks

1. Lack of experience: Since this may be the first time learning and using certain coding languages like HTML, JavaScript, CSS etc., finish dates of some tasks may be postponed due to unfamiliarity.
2. Inadequate security measures: Without proper understanding of security risks and best practices, we may not be able to implement adequate measures to protect user privacy and data security.
3. Inadequate testing: Without proper testing, we may miss important bugs or security vulnerabilities, putting our users at risk.
4. How to resolve merge conflicts?
    a. We will assign tasks specifically to make sure that we all work on different files. We will try our best to avoid merge conflicts.
    b. If there does have a merge conflict, we will follow the instruction in the following link: https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/addressing-merge-conflicts/resolving-a-merge-conflict-on-github

Teamwork

Based on our different computers and preference, our team will allow teammates to use editors that they preferred.

If we have different opinions on one function implementation (different algorithms for example), we will vote for the final result.

Since we are a team of four members, we will have two sub teams: a frontend team (Fr4nk and Kasse1), and a backend team (Q1 and Jinyu).

We chose this division of teams because Qi and Jinyu already know some about mySQL and MongoDB, which we are using for the backend, and Fr4nk and Kasse1 want to blend their insane creativity into web design.

We will create a to-do list for each period and complete the tasks within each subteam. If certain tasks need cooperation between subteams, a whole group meeting will be held.

Continuous Integration:

- **Testing library:** For the frontend part, we will use React Testing Library to do the unit testing for each function. As for the backend database, we will use MySQL Test Framework to write our test case.
- **Style guide:** Writing comments for each function, defining parameters with a proper name by following the snake case with clang-tidy.
- **Test coverage:** It will be computed by the number of lines that are covered by a test and divided by the total number of lines in your application.
- **Pull request workflow:** After one has finished his part and tested the code, push to the github and request review from another team member to double check the functionality. After peer review, we are able to merge the branch.