Testing Documentation

Preface: Data will be synthesized in our consistency system, recommendation system, and prediction system. Each system performs a unique task and these systems are some of the most vital components of the application we are building. Thus, we should look for how compromises can affect these components. In addition, we need to think in terms of the overall application design as well. This includes evaluating the hosting service, SurrealDB database records, Flask application containing our synthesis system, and HTML with CSS frontend.

Consistency system: This will heuristically analyze any faulty input either via comparison to other anonymized datasets or by correlating information within an individual's profile.

Recommendation system: Recommends best diet and nutrition intake, as well as exercises and habits tailored to the body and goals.

Prediction system: This determines the degree of improvement of the individual and is used by the recommendation system to further accuracy.

Overview of Systems: First, the systems need to have proper input for proper output, thus ensuring valid and accurate input is important. As a result we have two general criteria already due to the presence of attack vectors in this area.

Goal:

1) Ensure valid input data
    a) Impose ranged limit on quantitative data
    b) When required, impose regular expression limit on qualitative data
    c) Impose predefined buttons for the form

2) Ensure accurate input data
    a) Approved facilities can input data with consent of the user
    b) The consistency system will notify of bad information and possibly reject information to be input again under necessary circumstances

Execution:

1) Ensure valid input data
    a) Differentiate between pounds and kg when indicating weight
    b) BMI, height, and weight cannot be drastically unproportional
    c) Recommendations take into account daily activity levels

2) Ensure accurate input data
    a) Ensure accuracy for gender data
    b) Monitor changes of a specific user by limiting duplicate accounts

This prevents any privilege escalation or false information from entering the database and thus our systems. Furthermore, our work regarding security and integrity doesn't need to lie in the systems any longer due to addressing these fundamental issues. Thus, we can move onto the more broad issues of the hosting service, SurrealDB database records, Flask application containing our synthesis system, and HTML with CSS frontend.

Overview of Hosting: The most important idea is to structure the directory properly and have a reliable host. Thus, a host with good DDOS protection is needed as well as a well-organized project.

Goal:

1) Obtain a reliable host with DDOS protection

2) Have a repository to grab stable or mainline code to pull into the website being hosted

Execution:

1) Mitigate risks of spam accounts and server sustainability

Overview of Database: The principle concept here is to ensure anonymity. Creating a distinct public dataset will help to minimize personal risk overall.

Goal:

1) Divide datasets into accounts, personal data, and public data

Execution:

1) Create clusters in data set to properly retrieve correct data for users

Overview of Backend: The backend should not tax the server and must be responsive to requests at all times. It should be able to handle multiple synchronous user queries to the server.

Goal:

1) Ensure algorithms perform efficiently by documenting design

2) Rate-limit the Flask system usage

Execution:

1) Form executable flow between back-end and front-end

Overview of Frontend: Writing simple and reliable code conforming to modern standards is the best way to secure the frontend. As input handling was discussed earlier and tied into this aspect, we can forgo repeated discussion in this section.

Goal:

1) Conform to web standards

Execution:

1) HTML and CSS with minimal JS

References:

https://www.google.com/search?q=how+to+ratelimit+with+flask&sourceid=chrome&ie=UTF-8
https://medium.datadriveninvestor.com/rate-limit-a-flask-api-with-2-lines-of-code-c7976fb13436
https://www.section.io/engineering-education/implementing-rate-limiting-in-flask/