Project Report:

Please list out changes in the directions of your project if the final project is different from your original proposal (based on your stage 1 proposal submission).

The original project aimed to create a system that analyzed user attributes and preferences to provide dynamic, personalized exercise and diet recommendations. In contrast, the revised project centers on building a calculator that allows users to log their daily calorie intake and exercise activities. The primary goal of this version is to compute the total caloric balance for each log, helping users track their progress rather than offering proactive suggestions.

The scope of the project becomes more specific and focused, moving away from the broader ambition of creating an adaptive recommendation system. Instead, the revised project concentrates on the accurate calculation of calories consumed and burned, streamlining the functionality to meet these goals without requiring complex algorithms or machine learning components.

In terms of features, the personalized recommendation aspect is replaced by functionality that enables users to input food items, caloric intake, types of exercise, and estimated calories burned. The system then calculates and displays the total caloric balance for each log. This practical approach eliminates the need for user profiling and dynamic adaptation while emphasizing immediate feedback on logged activities.

Technologically, the revised project simplifies backend and computational requirements. The original proposal might have required integrating databases to store user preferences and employing recommendation algorithms to tailor fitness plans. The revised project instead involves basic arithmetic operations and potentially a lightweight database or storage solution to maintain a history of user logs.

The user interface shifts from a dynamic design aimed at guiding users through personalized recommendations to a straightforward input-output system. Users will interact with fields for logging food and exercise details, and the system will display the resulting caloric calculations. This streamlined interface prioritizes clarity and ease of use over the dynamic interactions required in the original design.

Evaluation metrics also change in the revised project. Success in the original plan depended on the accuracy and relevance of recommendations, as well as user satisfaction with personalized insights. In the revised version, the key metrics are the accuracy of caloric calculations, the usability of the logging and tracking system, and user engagement in maintaining their logs over time.

This shift in focus stems from constraints in time or resources, making the original proposal's complexity impractical within the project's scope. By narrowing the objectives, the revised project achieves a functional and valuable outcome, providing users with essential tools for monitoring their fitness journey while laying a foundation for potential future enhancements.

Discuss what you think your application achieved or failed to achieve regarding its usefulness.

The application successfully achieves its goal of providing a functional and user-friendly fitness tracking platform. It allows users to log workouts, view workout histories, and manage profiles, addressing essential fitness tracking needs. The inclusion of guest access makes the app more accessible by lowering the barrier to entry, and its responsive design ensures a seamless experience across various devices. Basic security measures, such as authentication and session management, contribute to a safe environment for user data, while the modular design supports future scalability.

However, the application has several limitations that impact its overall usefulness. It lacks advanced features like personalized fitness recommendations, progress visualizations, and integration with external fitness services, which could significantly enhance user engagement. Additionally, the app does not provide insights into fitness trends or tools for tracking long-term progress, such as data visualizations or statistical summaries. The absence of gamification or social features, such as achievement badges or workout sharing, limits user retention and engagement. Furthermore, the reliance on a stable internet connection restricts its usability for offline workout logging, and the lack of multilingual support and accessibility features narrows its potential audience. Guest sessions, while functional, do not allow data persistence, which could improve the transition from guest to registered user.

In conclusion, while the application meets basic fitness tracking requirements effectively, there is significant room for growth. Future iterations should focus on adding advanced analytics, personalization, and engagement features to make the app more comprehensive and appealing to a broader user base.

Discuss if you changed the schema or source of the data for your application

No changes.

Discuss what you change to your ER diagram and/or your table implementations. What are some differences between the original design and the final design? Why? What do you think is a more suitable design?

No changes were made to the ER diagram from Stage 3. We keep our ER diagram to be clear and straight forward so that our database will not be too complicated but still satisfy requirements.

Discuss what functionalities you added or removed. Why?

The final project includes a set of functionalities tailored to its revised goal of serving as a fitness calculator focused on tracking caloric intake and expenditure. Key functionalities added include the ability for users to create accounts with personal information, enabling the creation of individualized profile pages. This feature helps users maintain their personal fitness records and access their data conveniently. Additionally, a Goal page allows users to set their fitness objectives, such as weight loss or maintenance, giving the app a personal touch and providing context for their logged activities. Another essential functionality is the workout log page, where users can search and select one or more food items or exercises to record their caloric intake and calories burned. This feature streamlines the logging process and ensures that users have an easy and intuitive way to track their progress. Finally, users can view and

delete stored workout logs via their profile page, granting them full control over their data and the flexibility to update or revise their records as needed.

On the other hand, certain functionalities were removed to align with the project's revised scope and simplify its implementation. The recommendation system, initially planned as a core feature, was excluded. This decision was driven by the shift away from a personalized guidance model to a tool focused solely on accurate caloric tracking. Eliminating this feature also reduced the technical complexity, making the project more manageable within the given timeline. Additionally, the creative component of incorporating video and music content was removed. While this feature could enhance user engagement, it was deemed non-essential to the primary goal of the project and would have required additional resources to implement effectively.

Explain how you think your advanced database programs complement your application.

The advanced database programs in the application, such as using MySQL to enforce age restrictions during registration, ensure data integrity and enhance functionality. By embedding rules that prevent users below or above certain age thresholds from registering, the database helps maintain a valid and appropriate user base for the fitness calculator. This approach offloads validation from the application layer, simplifying the code and ensuring consistent enforcement of rules. It complements the application's goal by tailoring the system to its target audience while maintaining a secure and reliable data structure.

Each team member should describe one technical challenge that the team encountered. This should be sufficiently detailed such that another future team could use this as helpful advice if they were to start a similar project or where to maintain your project.

Chenhan Luo:

One technical challenge I met during this project is how to initialize a React project. I initialized and configured the React project according to some online tutorials, which made our front-end code work. But during the later maintenance, I found that because we used JavaScript code to edit the entire React project, the code readability was not high enough. So my suggestion is that when you work with the React project later, you can use .jsx files instead of JavaScript files, which can greatly improve the readability of the files.

Ze Yang:

One of the significant technical challenges we encountered was implementing a robust workout logging system that could handle multiple related database operations atomically while maintaining data integrity. This was particularly challenging because each workout log entry needed to be associated with multiple exercises through an affiliations table, and we needed to ensure that partial entries wouldn't be created if something went wrong.

The key challenges included:

1. Handling variable numbers of exercises per workout

2. Ensuring atomicity across multiple table insertions
3. Maintaining referential integrity
4. Proper error handling and rollback

Our initial naive approach was to simply insert records one at a time, but this led to issues where we'd have orphaned records if a later insertion failed. We solved this by developing a comprehensive stored procedure (`log_workout`) that:

1. Uses transaction management to ensure atomicity
2. Implements a dynamic string-splitting approach for handling variable numbers of exercises
3. Includes proper error handling with custom error messages
4. Validates data before attempting insertions

Nianze Guo:

Ensuring Session Consistency Between Frontend and Backend for Authentication and State Management
Description of the Challenge:

One significant challenge encountered in this project was ensuring that the session state between the frontend and backend remained consistent, especially during user transitions such as logging out, logging in as a guest, or switching between authenticated and unauthenticated states. This was particularly problematic because the frontend displayed outdated or incorrect user information due to lingering session data or improper state clearing. This challenge became evident when: The previous user's data persisted in the frontend even after logging out and continuing as a guest. Session cookies and state were not being properly synchronized, leading to discrepancies between the backend session data and the frontend's rendering logic.

Root Causes
Partial Session Clearing:
The backend originally cleared only specific session keys (e.g., user_id, username) instead of clearing the entire session. This left behind potentially sensitive session data.
State Management in Frontend:

The frontend was relying solely on backend responses without explicitly resetting the local state when logging out or switching to a guest session. This caused the profile page to display outdated user information.

Backend Response Expectations:
The frontend assumed that a successful logout would immediately invalidate all session data, but due to session persistence in cookies, the frontend sometimes retained outdated information.

Solution
Full Session Clearing in Backend:

Updated the /logout route to use session.clear(), ensuring that all session data was removed. This included session cookies and any metadata associated with the user session.
Explicit Frontend State Reset:

Added logic in the frontend's Login.js and Profile.js components to explicitly reset the user state to null when logging out or continuing as a guest.
Example in Profile.js:

```
useEffect(() => {
  fetch('http://localhost:5000/getuser', {
    method: 'GET',
    credentials: 'include',
    headers: {
      'Content-Type': 'application/json',
    },
  })
    .then(async (response) => {
      const data = await response.json();
      if (response.ok && data.user) {
        setUser(data.user);
      } else {
        setUser(null); // Clear user data for guest sessions or errors
      }
    })
    .catch(() => {
      setUser(null); // Handle errors gracefully by treating as guest
    })
    .finally(() => {
      setLoading(false);
    });
}, []);
```

Ensuring Backend and Frontend Synchronization:

Updated the handleGuest function in Login.js to call the /logout endpoint before navigating to the home page as a guest. This ensured that any residual session data was cleared.
Testing Workflow:

Tested various scenarios, such as:
Logging in, logging out, and continuing as a guest.
Navigating to the profile page as a guest.
Switching between multiple user accounts.
Advice for Future Teams
Backend Session Management:

Always use session.clear() for logout functionality to prevent lingering session data from affecting subsequent logins.
Ensure that your session configuration (e.g., cookies, timeouts) is consistent with the authentication flow.
Frontend State Management:

Reset local state explicitly when the session changes, especially for sensitive components like the profile page.
Use a centralized state management solution (e.g., Context API or Redux) to synchronize session states across components.
End-to-End Testing:

Test session workflows extensively, including edge cases like switching accounts, guest sessions, and invalid logins.
Simulate backend failures (e.g., session timeouts) to ensure graceful handling in the frontend.
Documentation:

Document the session flow and key endpoints (e.g., /login, /logout, /getuser) to help future teams understand how session management works in the application.
By addressing these issues and implementing robust solutions, the project now ensures consistent behavior for session-sensitive functionalities. This approach can serve as a blueprint for similar projects or future maintenance.

Jiaxin Liu:

Technical Challenge: Implementing Secure Session Authentication with CORS in a Flask-React Stack
Challenge Description
During the development of our fitness tracking application, we encountered a significant technical challenge when implementing session-based authentication between our Flask backend and React frontend. The core issue was maintaining secure user sessions while properly handling Cross-Origin Resource Sharing (CORS) requests, as our development environment ran the Flask backend on port 5000 and React frontend on port 3000.

Specific Problems Encountered
Session Cookies Not Being Set: Initially, even though our backend was setting session cookies correctly, they weren't being stored in the browser when making requests from our React frontend. The cookies were being blocked due to cross-origin restrictions.

Inconsistent Authentication State: Users would appear logged in when making requests directly through tools like Postman, but requests from the React frontend would fail authentication checks.

CORS Preflight Requests: OPTIONS preflight requests were failing for authenticated routes, causing POST, PUT, and DELETE requests to fail before they even reached our endpoint handlers.

Solution Implementation

Here's how we resolved these issues:

1. Proper CORS Configuration
We needed to explicitly configure CORS to handle credentials and specify the exact origin:

```
from flask_cors import CORS

app = Flask(__name__)
CORS(app,
    supports_credentials=True,
    origins=["http://localhost:3000"])
```

Key configurations:

supports_credentials=True: Allows cookies to be included in cross-origin requests
Specific origin instead of wildcard (*): Required for credentials to work
2. Session Configuration
We needed to configure session cookies to work cross-origin:

```
app.config.update(
   SECRET_KEY='your-secret-key',
   SESSION_COOKIE_SECURE=True,
   SESSION_COOKIE_SAMESITE='None',
   SESSION_COOKIE_HTTPONLY=True
)
```

Important settings:

SESSION_COOKIE_SECURE: Ensures cookies are only sent over HTTPS
SESSION_COOKIE_SAMESITE='None': Allows cross-site cookie setting
SESSION_COOKIE_HTTPONLY: Prevents JavaScript access to cookies
3. Frontend Axios Configuration
On the React frontend, we needed to configure Axios to include credentials:

```
axios.defaults.withCredentials = true;
axios.defaults.baseURL = 'http://localhost:5000';
```

Lessons Learned
Security vs. Development Experience: We learned to balance security requirements with development convenience. While allowing specific origins and configuring SameSite cookies as 'None' made development easier, we documented that these settings should be reviewed and tightened for production.

Testing Different Environments: We discovered the importance of testing authentication flows in various environments early. What worked in Postman didn't necessarily work in the browser, and what worked in development might not work in production.

Documentation Importance: Clear documentation of configuration requirements became crucial. We created setup guides for both development and production environments to ensure consistent configuration across the team.

Recommendations for Future Teams
Start with Authentication Early: Implement and test authentication mechanisms early in the development cycle. This gives time to discover and resolve cross-origin issues before building features that depend on authentication.

Use Environment Variables: Store configuration values like origins, cookie settings, and security keys in environment variables. This makes it easier to maintain different configurations for development and production.

Implement Comprehensive Testing: Create tests that specifically verify authentication behavior across different origins and requests types. Include tests for:

Session persistence
CORS preflight requests
Authentication state maintenance
Cookie handling
Security Considerations: Remember to:

Use HTTPS in production
Implement proper session expiration
Consider rate limiting for authentication endpoints
Regularly rotate session keys
Impact and Results
After implementing these solutions:

Session authentication worked consistently across all environments
Development workflow became smoother with proper CORS configuration
Security was maintained while allowing necessary cross-origin requests
Team productivity improved with clear documentation and configuration guides
This challenge highlighted the importance of understanding web security fundamentals and the interaction between modern frontend frameworks and backend services. The solutions we implemented not only resolved our immediate issues but also provided a solid foundation for adding more features that required authentication.

Are there other things that changed comparing the final application with the original proposal?

Describe future work that you think, other than the interface, that the application can improve on.

The application can benefit significantly from improvements in session management, data handling, and integration with external services. Transitioning to token-based authentication would enhance scalability and

security, while features like automatic session expiration and secure reauthentication can further protect user accounts. Optimizing backend queries and adopting scalable database solutions like PostgreSQL or MongoDB can ensure the application remains efficient as user data grows. Additionally, integrating with APIs such as Fitbit or Google Fit would enrich the user experience by providing personalized workout data, while adding email or push notifications would improve engagement through timely reminders.

Advanced analytics and recommendations present another avenue for enhancement. Machine learning algorithms could analyze user workout histories to suggest personalized fitness plans, combining exercise and diet recommendations. Progress tracking through data visualization tools, such as graphs for calorie burn or workout frequencies, would empower users to monitor their achievements effectively. Offline functionality, allowing users to record workouts without internet connectivity, could further enhance usability, syncing data once they reconnect.

Introducing features for multi-user support and customizable fitness goals would expand the app's appeal. Family or group accounts could foster collaborative goal-setting, while tools for creating and sharing workout routines would allow users to tailor the app to their specific needs. Gamification, such as badges, leaderboards, and streak counters, could boost user motivation and retention by turning fitness tracking into an engaging, interactive experience. These improvements would elevate the application into a comprehensive platform for health and fitness management.

Describe the final division of labor and how well you managed teamwork.

The team divided the work based on each member's strengths and expertise. Nianze Guo and Chenhan Luo focused on building the frontend, designing a responsive and user-friendly interface, while also assisting with some backend tasks. Jiaxin Liu worked on the backend, creating essential features like authentication, session management, and API endpoints. Meanwhile, Ze Yang did some of the advanced database features.

Connecting the frontend and backend was a joint effort by Nianze Guo, Chenhan Luo, and Jiaxin Liu. Together, they ensured smooth communication between the client and server, making the application work seamlessly. The workload was divided evenly, and everyone played an important role in making the project a success.

Teamwork was one of our strongest points. We often helped each other, especially during tricky integration and debugging tasks. For example, when we ran into problems with API communication, the frontend and backend teams worked closely to figure out and fix the issues. This collaborative approach kept things moving forward and allowed us to tackle challenges as a team. Overall, we managed to balance our individual responsibilities with group problem-solving, which made the project a rewarding experience.