Milestone 2 – Report

HealthMate

Your Smart Health Tracker

Group 22 Vines
Kiratpal Singh Johal, Mohid Khan, Justin Borgeson
Summer 2025

Repository Link
Deployed Website

Analysis of User Needs and Problem

To ensure our application would be useful and intuitive, we engaged in multiple rounds of informal user feedback throughout development. In the early stages, we shared mockups and described the idea to peers and family, even if they had an interest in fitness/nutrition tracking or not. This helped us validate the need for a centralized app that combines nutrition and exercise functions in one place. We decided for each page to have a three-column layout:

- The left column would contain input boxes and buttons for the user to use the function of the page.
- The middle column would include the results according to what the user wrote on the left.
- The right column would include additional information relevant to the middle column.

However, feedback from our peers for the mid-fidelity prototype for Milestone 1 suggested that there was too much information on the screen, so we decided to remove any columns on the right. In the event that extra information had to be displayed, they were organized under tabs that the user can click to access that information.

Peer testing was very positive with the overall look/aesthetics of the page, so were not able to make any changes regarding that. The only change change we made was to the navbar in an attempt to reduce clutter on the screen and organize functions better. Initially, the navbar would directly include buttons to functions such as “Analyze Meal” and “Log Exercise”, but we used two drop downs (“Nutrition” and “Exercise”) to categorize the functions (i.e., “Log Exercise”, “Search Exercises”, and “Generate Plan” are under “Exercise”).

Functionality wise for peer testing, we ran into two issues. The most significant issue was when users would log a meal. Sometimes the API would take too long, and the page wouldn’t update, so the user thought their inputs didn’t register, which would result in users clicking the button to submit a meal multiple times, resulting in multiple logs the user originally didn’t want in the first place. We circumvented this issue by disabling the submit button until the meal has been logged and clearing the form after the meal is logged.
The other issue was when a user tried to hit backspace in the nutrition goal entry. Users are able to backspace on numbers greater than zero but cannot backspace when the entry is 0. A user can either click the up arrow to increment the goal by one and then add digits to the end or double click the input box to highlight the 0 and then enter the number they want. This is an unresolved bug.

These iterative feedback loops helped us tailor the application toward beginner users with limited technical or fitness tracking experience, ensuring the core features remained streamlined and purposeful.

SDLC Model Analysis

Our team utilized an Agile framework for our software development lifecycle. This allowed us to break down development into smaller sprints. Each sprint included feature planning, development, testing, and retrospective discussions. Each week our team met twice (Wednesdays and Fridays) for roughly an hour to go over the current sprints, discuss the current state of the project, and determine requirements for future projects. Agile helped our team to stay on track for all tasks that needed to be completed through our various phases: Design, development and testing.

During each scrum meeting, we would discuss each other's work. This gave us a platform to easily provide feedback, and to allow all members to know what stage other members are at.

Furthermore, this helped us keep in track of any problems that team member faced, and the issue would be dealt ASAP.

For future improvements, we plan to improve sprint backlog tracking and implement better issue estimation metrics.

Feature Implementation Details

API	Feature Name	Description	Changes	Real World Usage
Edamam	Meal Analyzer	Users input full meals and receive a full nutritional breakdown. Helps users understand homemade or multi-item meals.	Instead of a search bar, a dynamic form was used to allow the user to enter just a single item or their own custom meal to add each individual ingredient included in it.	A user with dietary restrictions (e.g., diabetes) wants to analyze a homemade meal for sugar and carb content before eating. Instead of guessing, they enter each ingredient and get a full breakdown, helping them make informed choices. This supports the goal of promoting health-conscious decisions.
Edamam	Diet Filters	Users select diet preferences and system checks meal compatibility	Originally intended to allow the user to save their diet preferences to their profile by changing their settings but instead moved to the "Analyze Meal" page where the user interacts with a drop-down menu (with options such as "Low-Fat", "Low-Carb, etc) and performs a system check if it meets the requirements for the selected option or not.	A user on a low-carb diet selects "Low-Carb" from the dropdown and inputs their meal. The app alerts them that their meal exceeds the recommended carb limit, prompting them to replace high-carb items with healthier options. This shows how the app adapts to personal goals.
Edamam	Nutrition Tips	App offers suggestions to improve meals based on analysis.	None	After analyzing a meal high in sodium, the app suggests using herbs instead of salt. For example, a user planning spaghetti might be

				told to replace canned sauce with homemade alternatives. These tips educate users without forcing behavior changes.
Wger	Exercise Search	Search for exercises based on criteria.	Originally intended to include a search bar but uses three category filters to search for exercises instead: Category (arms, legs, etc), Equipment (barbell, dumbell, etc), and Muscle (anterior deltoid, biceps brachii, etc)	A user recovering from a leg injury uses the filters to find upper-body exercises that don't require gym equipment. The muscle and equipment filters help them safe and quick workouts.
Wger	Workout Logging	Users log sets, reps, and weights.	None	A gym user logs sets and reps for each workout session to track progress. Over time, they notice increases in strength and adjust their reps. This feature supports consistent tracking and motivation, aligning with fitness progression goals.
Wger	Routine Suggestions	Fetch or generate weekly plans based on goals.	Originally intended to allow the user to edit and save their exercise goals to their profile by changing their settings but instead moved to the "Generate Plan" page where the user interacts with buttons on the sidebar (such as "Muscle Building", "Cardio/Endurance", etc) and returns a plan based on the button pressed.	A beginner interested in building muscle clicks "Muscle Building" and receives a structured plan for the week. The plan includes exercises with images and descriptions. This real-world guidance helps new users start exercising

				without confusion or needing a personal trainer.

CI/CD Pipeline Overview

HealthMate was integrated and tested through the use of Github Actions. Our project was written in React.js, and so we used the Jest testing system to provide both unit and integration tests. As changes are being made, Github Actions runs the premade tests we have provided to determine if these changes prevent our project from working as intended. If these tests fail, Github actions will prevent deploying those additions into the overall project. If the tests are successful, then the system will determine them as safe for our overall project and will deploy.

Deployment is handled through Netlify. After determining that the implemented changes should run, Github actions notifies Netlify to deploy the adjusted project. Netlify handles the rest of our deployment.

Testing Strategy and Implementation

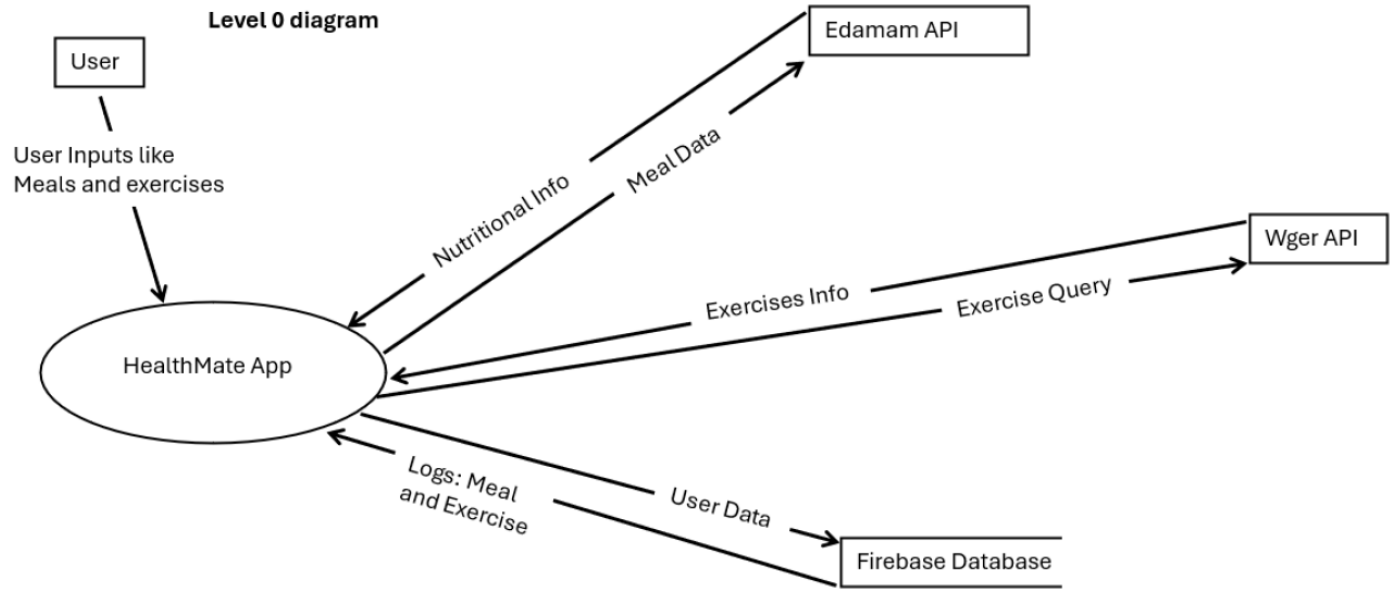
Our team prioritized testing backend functions, frontend integration, and effective API calls and responses. Our tests primarily covered the scope of unit testing and integration testing. We looked for proper functionality, and adjusted our functions as needed. The majority of our testing involved generating known data, and providing it into our functions, to emulate a user's input. Our tests hold on to an expected output and compare this to the actual output received from the function. If the received output does not match the expected output, the test will fail. Our integration tests work in a similar manner. They use the provided routes and determine if the results derived are correct.

Our testing was primary written using Jest. Mocks were created to simulate data and responses, so that databases were not directly accessed. The expect keyword was regularly used to provide the expected output. This function would throw an error in case of failure.

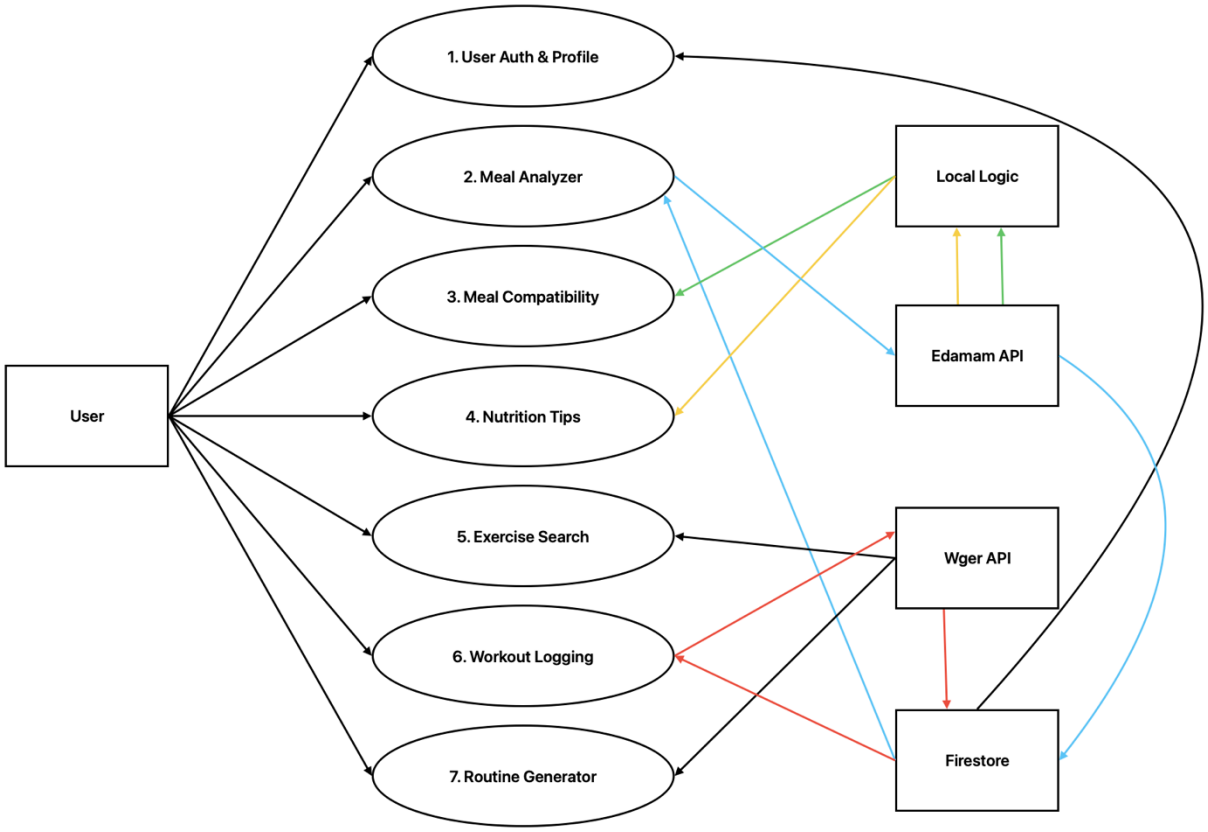
Most basic tests were automated, to guarantee that new additions would not break base functionality. More specific actions were done through manual testing. Some larger scale integration testing was also done manually, but due to the structuring of our code, identifying where any of these issues were sourced was not a major concern. Any determined bugs would be noted and issued out to whichever member was most equipped to deal with that issue. These were notified in our private slack chats and are meant to be listed in our Github Issues.

Project Architecture Diagram

Level 0 DFD

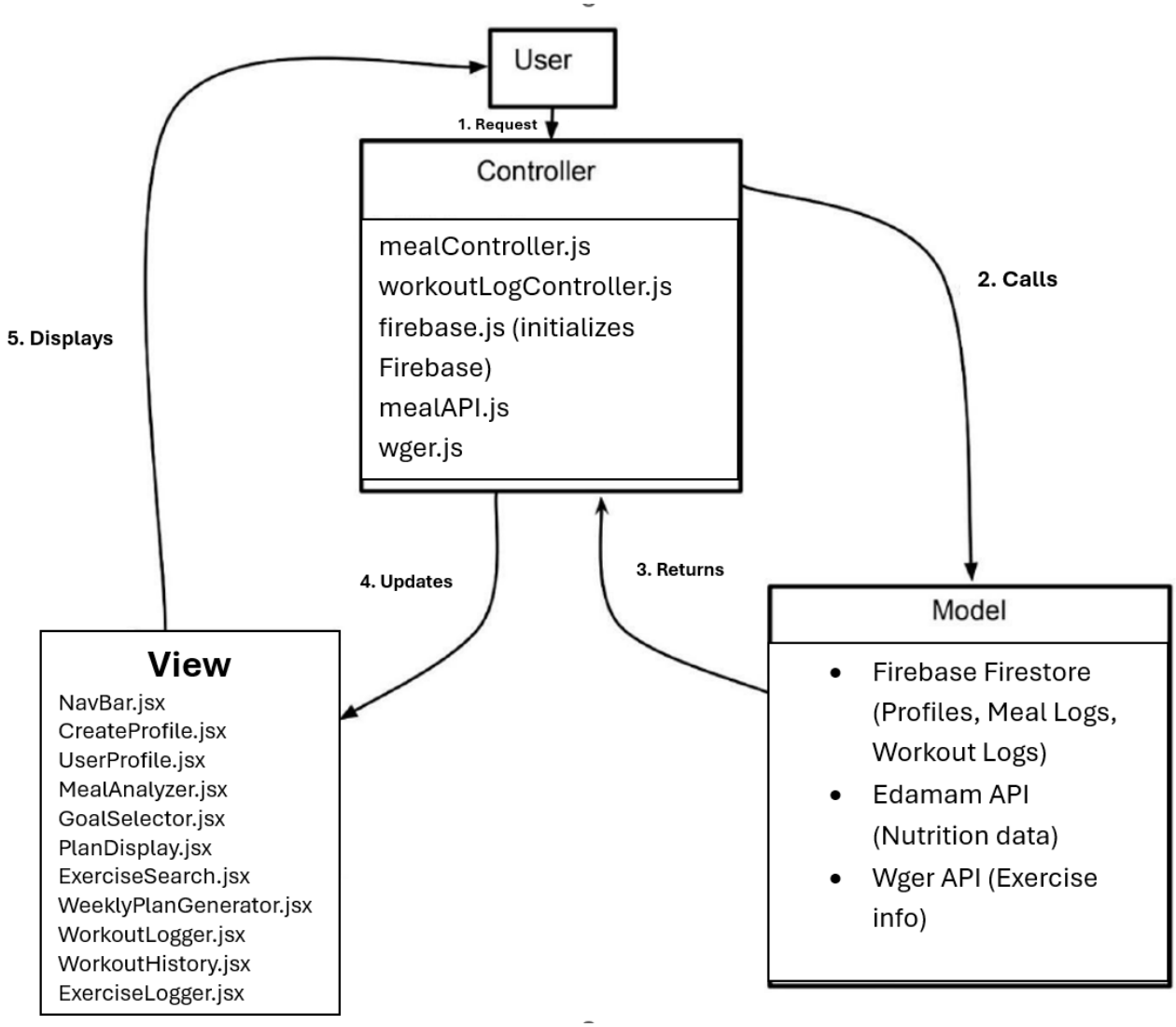


Level 1 DFD



The Level 1 Data Flow Diagram breaks down the internal structure of the HealthMate App into seven core sub-processes, each representing a major user facing feature. The first process, User Authentication & Profile, handles user login and stores profile information in Firebase Database. The Meal Analyzer allows users to input meal details, which are sent to the Edamam API to retrieve nutritional information. Diet Filters compare this nutritional data against selected dietary preferences to determine if it matches the criteria that is hardcoded in to nutritiontips.js file. Nutrition Tips analyzes the meal's nutritional values and suggests possible improvements. For physical activity, Exercise Search sends body-part or goal-specific queries to the Wger API and displays relevant exercises. Users can log these activities through Workout Logging, which stores the entries in Firestore for future reference. Lastly, Routine Suggestions generate a beginner friendly weekly workout plan based on a chosen fitness goal, using Wger API data and custom logic, and save it to Firestore. External systems like the Edamam API, Wger API, and Firestore are key components, enabling real-time analysis, data storage, and personalized feedback. This diagram illustrates how each feature connects to these services and supports the app's overall functionality.

MVC Diagram



HealthMate follows the MVC pattern to ensure separation of concerns, scalability, and maintainability.

- **Model**: This layer handles data and business logic. In our app, Firebase Firestore acts as the main database for storing user profiles, meal logs, and workout logs. It ensures data consistency across features and is accessed via server-side logic in Node.js. The Wger and Edamam APIs also act as external data sources treated as models when fetching dynamic content like exercise data or nutrition info.
- **Controller**: The controllers manage interactions between the frontend and backend. On the backend, files such as mealController.js and workoutLogController.js process API requests and route data to/from Firestore and third-party APIs. They ensure data is properly formatted, stored, or returned to the client. On the frontend, intermediate logic in files like mealAPI.js and wger.js also serves a controller role by handling fetch requests to our backend endpoints.

- **View**: This is the React frontend, which includes all components users interact with — such as CreateProfile, MealAnalyzer, WorkoutLogger, ExerciseSearch, GoalSelector, and PlanDisplay. These components render data and enable user actions like logging meals or generating a workout plan.

This separation ensures that any change in how data is stored (Model), how logic is applied (Controller), or how the UI looks (View) can be made independently without breaking the rest of the system.

Bug and Issue Tracking

GitHub Issue	Priority	Description	Reproduction Steps
#74	Medium: Its Medium priority because there are number of other exercises that trains same muscle and have English description, so its easy for user to switch to another one.	Some exercises are listed in different languages. We tried to filter the exercise by language but when we used it, it stopped displaying all the exercise even the ones with English name and description. A user might get an exercise that's not in English and will have to find another in English.	Go to "Generate Plan" under "Exercise". Set the Category to "Abs", and leave the other options as "Any". Click search. Scroll until you find the exercise named "Estiramiento de brazos y cuello "
#75	High	When a user wants to set their diet goals by going to settings the user can enter a negative value which should not happen in real life.	Go to "Diet Goals" under "Settings" and try to clear an input field only using backspace
#76	Low	Workout plan print page does not include styling as it prints in pdf and everything is in Times New Roman. Does not look as pleasant.	Go to "Generate Plan" under "Exercsie", click on a plan and click "Print This Plan". The preview page should have no styling
#77	High	Entering invalid input (typos, gibberish, etc) just returns the last valid meal you inputted and adds into the current meal. That is why user has to add the foods with proper syntax.	Go to "Analyze Meal" or "Log Food" under "Nutrition". Enter and submit a valid input (i.e., "1 egg"). Record the nutrition details. Clear the form. Enter and submit an invalid input (i.e., "ajkghfda"). The nutrition details

			should have the same output as the previous valid input.
#78	Medium	Pressing enter inside a meal entry creates a new entry below it, but does not auto focus to it. Slows down the user, making the user have to click more.	Go to “Analyze Meal” or “Log Food” under “Nutrition”. Enter and add an entry (i.e., “test”). Press enter. The new entry will be made, but does not auto focus to allow text to be inputted.

Future Work and Improvements

Feature	Description	Impact	Est. Time
Accounts	Users will need to sign up with 2FA as right now anyone can access other people accounts because there is no 2FA authentication with phone numbers right now.	Very high. Currently there is no way for the user to store their logged meals or nutrition goals, and the current method for storing exercises with phone numbers is not secure.	3 days
Filters for workout logs	Right now user see all their workout logs grouped together but in future we would want to make new filters so user can filter out the workouts that they did on a certain day instead of going through all the logs one by one.	Medium. User can still find the their workouts for a certain day as they are arranged according the to the time they were logged but filters will make it easier for user to quickly find out what they did on a certain day.	4 days
Search Bar for “Search Exercises” page	In addition to the three filters in “Search Exercises”, the user can narrow down their search by using a search bar to input exercise names	High. A user may just want to find more information about a specific exercise they heard about, and might be completely unable to do so if they aren’t aware the categories the exercise falls under to find it.	4 days

Lessons Learned and Takeaways

- Having a very clear understanding of what information the API can deliver and creating features based off of that is significantly more efficient than creating features you would ideally want and hoping you find an API that fulfills all those needs. For example, we had the “Settings” page populated with a variety of options for the user such as “Activity Preferences” and “Diet Type”, which were instead implemented in the “Generate Plan” page and the “Meal Compatibility” feature, respectively. Going forward, we will do more research on the APIs we will use for future projects to fully understand what it is capable of delivering.

- One of the most important and challenging lessons we learned during this project was how to work effectively as a team, especially when using Git and GitHub for collaboration. The first time we tried to merge our branches into the main branch, we ran into many merge conflicts. Resolving each conflict manually was time-consuming and confusing because we weren’t aware of the changes others had made. After that experience, we improved our communication. We started updating each other on Slack whenever we made changes to our branches or created a pull request. This way, everyone knew when to pull the latest changes from the main branch into their local repo and could avoid editing the same files at the same time. As a result, we had fewer conflicts and smoother mergers in the later stages of the project. In future we now know how important it is to communicate on time and update our team members of new changes being pushed to main branch.