

Kipitup

*Utilizing biosensory data for nuanced health and fitness recommendations*

Submitted to

THE SCHOOL OF INFORMATION

In Partial Fulfillment of the Requirements for the Degree

MASTERS in INFORMATION MANAGEMENT AND SYSTEMS

| Filipe Santos | Anudeep Chawla |
| --- | --- |
| Paul Kenniston | Nithin Ravindra |
| Dhanya Koottummel | Shreena Bhati |

Advisor: Professor John Chuang

## Table of Contents

[**Table of Contents 2**](#_a61tufgjdk64)

[**About Kipitup 3**](#_jdy93sss926l)

[Product Summary 3](#_z1xro3xl4pcc)

[Problem Being Solved 3](#_7nl99dqm8gzp)

[Benefit of Biosensory Data 3](#_rzudy9acvqg5)

[End Users 4](#_6ea8qvs4nuv7)

[Motivation 5](#_16v9u2z5l7ht)

[Market Position & Competitive Analysis 5](#_b4ajcvfensto)

[**Product Development Methodology 7**](#_vvf2eh2astts)

[Primary Use Case 7](#_fc4e445j7d9f)

[Defining Success 7](#_ktcjdnq10dvx)

[Bifurcating by Speciality 7](#_446m4jagowb)

[**Technical Implementation 9**](#_63tl5l5a0ol3)

[Data Architecture at a High Level 9](#_v3r6t3cxd1ok)

[Apple Watch 10](#_h3ocsmpb0ter)

[About Kipitup’s Apple Watch App 10](#_adhumca8apii)

[Flow & User Experience 10](#_of2790zfpbe4)

[Prompt Engineering 14](#_a4iz7x1756u)

[Understanding the behavior of custom GPT, in ChatGPT interface 14](#_dg8wkyww591d)

[Understanding the behavior of GPT-4-turbo API (via assistants) 15](#_64qw8o5eezva)

[Final Prompt Iteration 16](#_2mczl9kph7pf)

[Large Language Model (LLM) 18](#_84xg9l2yptxt)

[AWS Lambda 19](#_fw340o8l49bu)

[**How to Access Kipitup 20**](#_n4vyp0jj7u72)

[**Ongoing Development 22**](#_gznnqab7sarv)

[Ideas for Continued Development 22](#_vo9b16czb0cd)

[Lessons Learned in Hindsight 23](#_wbsjwpe7ibui)

[**References 25**](#_xz5jsj5so76d)

## 

## About Kipitup

##### Product Summary

Kipitup is an Apple Watch app that provides fitness advice for any athlete, no matter where they are on their journey. It delivers fitness advice that incorporates both the wearer's biosensory feedback and industry leading best practices, ensuring that advice is always specific to what would be best for the wearer. Kipitup is able to provide specific steps for improvement and feedback for 'cardio training' and ‘marathon preparation’.

Users will interact with Kipitup by asking natural-language questions to their Apple Watch. After Kipitup takes into account the user's biofeedback and trends, it will reply with customized user-specific results.

##### Problem Being Solved

The current ‘one-size-fits-all’ approach to fitness is ineffective. It ignores our body limitations, and pushes people away from fitness when general fitness advice doesn’t meet their needs. To give an example, let’s say that you wanted to start training for a marathon. One common way to start training is to Google “marathon training routine” and pick a routine that gradually ramps up difficulty. But, who’s to say that is the right routine for YOU? Perhaps you live in a very hilly city like San Francisco, and you frequently run out of breath trying to clamber the steep inclines. Or, you have a history of heart palpitations and need to stay within specific heart-rate bounds so as to not add any additional stress on your heart. Our product solves for this by uniting leading fitness advice with the wearer’s biosensory data. For our San Francisco runner, looking at the runner’s V02 max in combination with the average run time would enable us to create a new training regime that addresses the hills and changing altitudes. For a heart-sensitive runner, our product would adaptively modify their routine to ensure that they are never exceeding.

##### Benefit of Biosensory Data

Current health and fitness products are currently missing a critical dimension relevant to physical improvement - biosensory data. Software that doesn’t utilize biosensory data have models that typically only incorporate data that users manually input (e.g. ‘runs’, ‘workouts’), and use this information to create inferences and predictions regarding their user’s capabilities. But, our abilities to grow and perform effectively are dependent on our body’s response to the task at hand. We see health and fitness apps that do not utilize biosensory data as akin to personal trainers giving workout recommendations over Zoom without ever meeting their clients in person. They wouldn’t be able to see their client’s sweat, or see how flush their clients are in the middle of strenuous exercise. Sure, their clients would be able to vocally express how they are feeling to their personal trainer, but no amount of dialogue could express the complexity of the inner-workings of their body.

If we take our example of a marathon which is an endurance workout, heart rate (along with other biosensory data) becomes a very important metric to monitor. Marathon prep routines provided by competing apps are generic and ambivalent towards body type and physical characteristics. Surely, these guides cannot be equally relevant to all runners. What about the differential in leg-length? Shorter runners will undoubtedly have thousands more strides per race. What about runners with heart defects or pacemakers, who could benefit from maintaining more restrictive high and low bounds on their working heart rate? No two bodies are the same, so how could a health and fitness app that flattens the unique characteristics of users provide the best recommendation? Our utilization of biosensory data is the panacea that makes accurate health and fitness recommendations possible.

##### End Users

*Regarding the population of potential users:* one of the primary benefits of Kipitup, which makes it distinct from both traditional app-based fitness advisors and in-person personal trainers, is its flexibility and applicability to all fitness levels. Each question posed to Kipitup is accompanied by demographic information (‘Age’, ‘Height’, ‘Gender’, ‘Weight’) and biosensory data gathered by the apple watch (‘HR’, ‘VO2 Max’, etc.). Incorporating demographic data ensures that results are reflective of the unique characteristics of all sorts of users. Incorporating biosensory data ensures that results are reflective of the *specific user* who is querying KipitUp. This means that our possible population is very wide with only a few exceptions (mentioned below).

*Regarding hardware:* One such restriction on our possible population of users, and likely the biggest restriction, is a result of hardware. Currently, Kipitup is exclusively built for the Apple Watch. We chose to focus on the Apple Watch due to its saturation in the market (+229 million Apple Watches sold in the world before 2022)[[1]](#footnote-0), and dominance over other fitness-tracking wearable devices (most sold wearable devices, owning 22% of the market as of Q2 2023)[[2]](#footnote-1). We may consider expanding our software to other wrist-based wearables using different operating systems (e.g. Garmin, Amazfit Band, Samsung Galaxy Watch). Additionally, we may expand the scope of our primary data sources to include supplemental sensors and other wearable devices as well (e.g. Oura rings, smart treadmills). But, for the time being, our user base is restricted to only those who own an Apple Watch.

*Regarding use cases:* Currently, Kipitup is built with exclusively cardio in mind. While it is fairly versatile in giving generic running advice, it has been specifically designed and trained to provide advice for cardio training. This is another restriction to our total population, as it does not benefit athletes who are interested in other forms of non-cardio exercise (e.g. strength training, team sports); it also doesn’t benefit those who are interested in alternative forms of cardio (e.g. cycling, swimming). While this is a fairly limiting restriction to our potential population, this is also the easiest restriction to lift. To do this, we would only need to expand our corpus to include documents that speak to these alternate forms of exercise and train our model via prompt engineering to distinguish between these types of exercise. In future iterations we may expand the scope of our audience to include these other fitness types and grow our potential audience.

*Regarding language:* Kipitup is only built in English, which serves as another limiting agent that reduces our potential population. English is the language displayed in the watch app, as well as the language considered by our LLM when it analyzes the user’s query and generates a response. In future iterations, we may expand Kipitup to include multiple languages.

*Regarding disability:* All that said, Kipitup does not cater to those who are differently abled. The data types that we are gathering to derive fitness advice are generally applicable to able bodied users of all ages (within reason); however, we don’t capture data from our users that inform us of any physical disability which we could use to customize fitness results in a way that caters to the physical capabilities of our users. This omission is a result of time and resource limitations (e.g. lack of willing differently abled participants to test our product, lack of corpora providing fitness advice for differently abled athletes, limitations of the Apple Watch requiring traditional limbs), as well as technical limitations that would have complicated development (e.g. how would we quantify disability so that our LLM would provide accurate advice, how would we avoid providing results that further stress existing health issues). While we acknowledge the existence of differently abled athletes, and recognize that differently abled users would benefit from fitness advice as much as able bodied users, we regretfully had to exclude building for disability from our scope in order to get a working product out the door in time.

##### Motivation

We settled on a fitness app for the Apple Watch as a result of our personal goals to work with sensor-dependent hardware, our recognition of a sizable gap in the market for fitness and lifestyle applications, and our desire to build what we know and leverage our own experiences with personal training.

However, our primary motivating factor was to build something *real.* We wanted to leave this experience with something tactile - something with complicated components that could only have been created as a result of months of specialized training, knowledge, and labor. We wanted to leave this year having built something that we could have never accomplished prior to our university experience. No other product that we considered met our criteria of being tangible and complex like a *‘fitness coach on your wrist’* did.

##### Market Position & Competitive Analysis

Despite the restrictions on your potential population previously discussed, the potential customer base remains massive, even if you only include English speakers and Apple Watch users. Below are a few statistics that speak to the potential reach of this product.

* 621.2 million runners globally[[3]](#footnote-2)
* 49 million runners in the US alone[[4]](#footnote-3)
* 229 million Apple Watches sold in the world as of 2022[[5]](#footnote-4)
* Apple Watch owns 22% of the wearable device market (#1 consumer wearable)[[6]](#footnote-5)

As mentioned above, we have additional stretch goals to add new languages and to build for other wearable devices, which would expand our market even further.

Below is a brief analysis of several of the leading competitors in this space, and how Kipitup is uniquely positioned to disrupt their market position and claim their user base.

*Strava* - App used to catalog runs and compare to historical runs with a built-in social space. Strava boasts effective visualizations of runs using GPS data, and denotes changes in elevation and pace on the route. However, Strava doesn’t provide recommendations, nor does it acknowledge body-data.

*Nike Training Club (NTC)* - Unlike Strava, NTC does provide fitness recommendations to their users. However, NTC provides a one-size-fits-all approach to training, as their recommendations come from a handful of trainers in the form of virtual classes. Training recommendations are not provided in an accessible, instantaneous format, nor does it acknowledge the user’s body-data.

*Peloton* - Similar to NTC, Peloton offers trainer-led classes for running workouts. It suffers in the same ways that NTC suffers - a lack of accessible, instantaneous lessons, and that it doesn’t incorporate the user’s body data.

While there are plenty of fitness apps on the market, no one else is offering body-specific customized workouts.

## 

## Product Development Methodology

##### Primary Use Case

Building a product from the ground up, we recognized that excelling at a single use case for a smaller population is more useful than partially serving the needs of a wider population. To that end, we built Kipitup to provide excellent advice to runners. This use case was our ‘north star’ during development - as we considered new features and iterations of our product, we constantly reevaluated if these changes would provide a tangible benefit to runners.

##### Defining Success

In order to provide a product by our deadline that served runners effectively, we had to consistently reevaluate our Minimum Viable Product (MVP). Like any technical product, during the early stages of development we had lists of aspirational features that never made it into the final product. In an effort to avoid scope creep, we agreed on a collection of capabilities that were ‘must haves’, and considered every other capability beyond that as ‘nice-to-have’. Below is a summary of the core functionality that guided our development:

* Receive a query directly from the user through the Apple Watch
* Generate running advice dependent on the query
* Ensure that advice generation incorporates biosensory data
* Ensure that advice reflects long-term incremental progress
* Deliver the advice to the user

##### Bifurcating by Speciality

We realized early on that building this MVP required us to develop many different artifacts that all worked in concert with each other. To make our MVP work at its most foundational level necessitated a responsive LLM, a data warehouse and data delivery protocol, and a working Apple Watch app. But, to make our MVP truly effective, we also needed to create and iterate wireframes, design a means for delivering results in a readable and well-organized format, and design product personality artifacts (e.g. logos, color schemes). After documenting all these activities and outlining their dependencies, it became clear that we would need to divide this work within the team so that work can be completed in parallel.

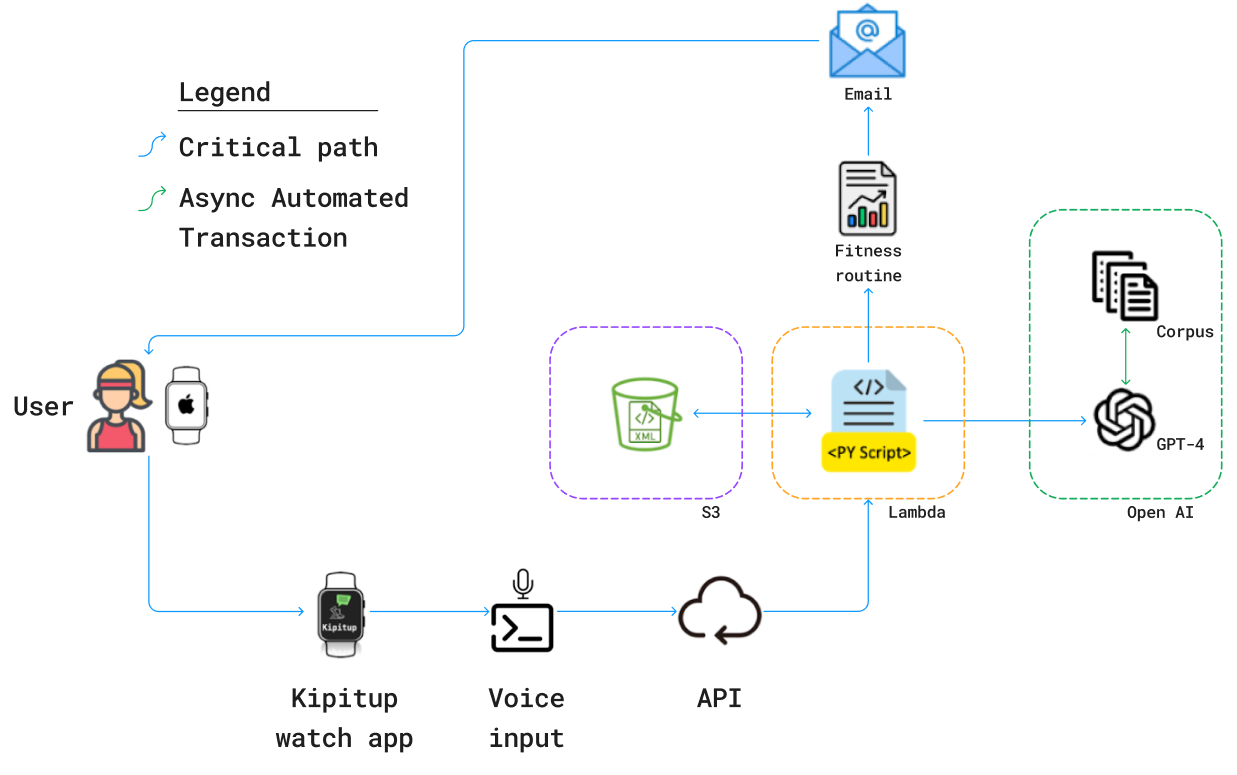
We bifurcated our team into two swimlanes, ‘LLM development’, and ‘Apple Watch development’, to parallelize our work. Whenever one swimlane encountered a blocker due to an unresolved dependency, members from that swimlane would tackle upcoming work that didn’t neatly fit into either of the two swimlanes (e.g. wireframes, website creation). This enabled us to make consistent progress, and mitigated the impact of technical dependencies.

Bifurcating into pillars also gave us breathing room to get messy and make mistakes. Our mentality throughout this project was that it is just as important to try things out, test interesting and disruptive ideas, and learn something new about product development as it is to build the final product. Because we divided work, we knew that even if one swimlane took some big swings that didn’t work out, the other swimlane was still pushing forward.

## 

## Technical Implementation

##### Data Architecture at a High Level



*Figure 1 - High Level Architecture (HLA) diagram*

Here is a high-level overview of how Kipitup will work. Users vocally ask a fitness-related question using the Kipitup app on their Apple Watch. Kipitup converts their question using speech-to-text. The user's question is directed to a first Large Language Model (LLM) assigned to perform intent classification[[7]](#footnote-6). Once the user's intent of prompting about a running routine is validated, a second Large Language Model, including a domain-specific corpus, is triggered, leveraging updated industry-leading fitness best practices. In parallel, a biosensory ''snapshot'' of the user is taken using the Apple Watch, and Kipitup sends this snapshot, along with the user's question to an API endpoint. This process automatically triggers an AWS lambda function to invoke the LLMs and handle the request, combining the user's prompt and pre-processed biosignals to add relevant context. Finally, Kipitup generates fitness advice based on existing best practices, which are tailored to the user's biodata, ensuring that recommendations are always unique and customized to the user. That generated fitness advice is then sent back to the user via email.

##### Apple Watch

###### About Kipitup’s Apple Watch App

Development of the ‘Kipitup’ Apple Watch was done entirely in Swift (*version 5.10*), the general-purpose programming language created by Apple. It has been developed as a ‘standalone’ Apple Watch app, meaning that users do not need to download a companion app onto their mobile phone in order to use Kipitup.

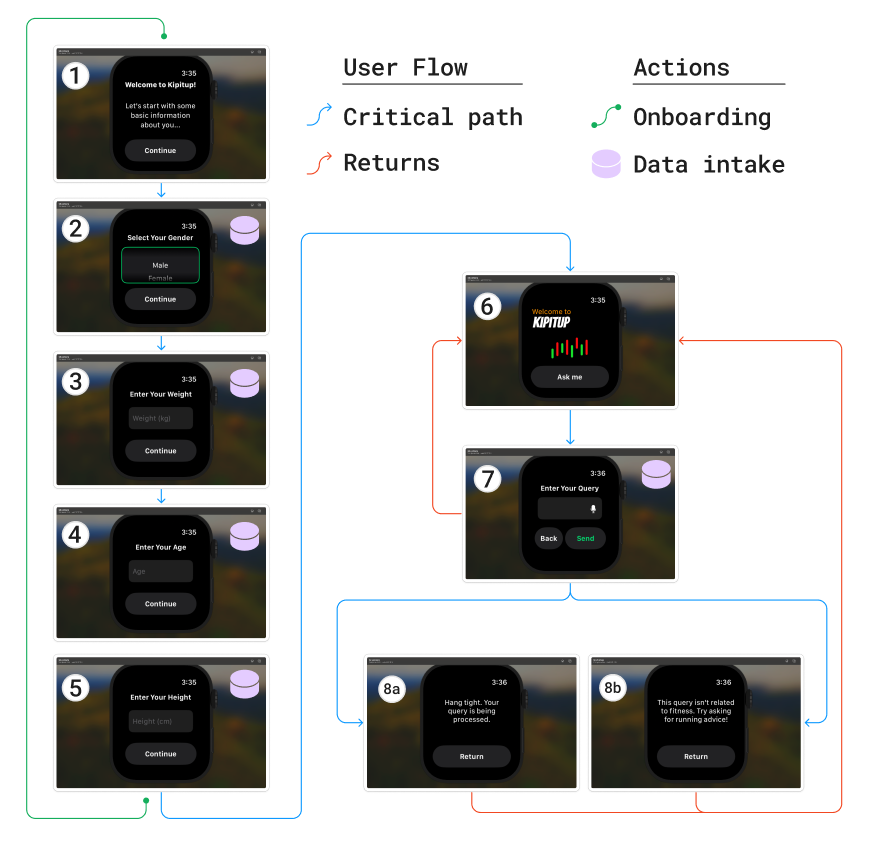
The Apple Watch app is the first stop in the user journey, and serves two critical purposes: 1) gathering biosensory data from our users, and 2) receiving the user’s fitness-related query. This data is then passed to the LLM via a AWS Lambda function, which processes the query and returns an appropriate answer to the query which is customized to the biodata of the user.

Packages Kipitup utilizes:

* ‘SwiftUI’ - This package tells Swift that we want to utilize tools offered by the SwiftUI framework. This enables the use of core functionality in Swift such as ‘struct’, ‘func’, ‘View’, and allows developers to see previews of their app without needing to export. This package, or its lesser-used alternative ‘UIKit’, is essentially necessary to build a project in Swift whether the app is being built for watchOS (Apple Watch), iOS (iPhone), or visionOS (Vision Pro).[[8]](#footnote-7)
* ‘URLSession’ - This package defines a class that can be used to download and upload data from an endpoint accessed via an URL. URLSession is being used to upload the user’s query and biodata to our AWS Lambda instance, and call the results generated from our LLM.[[9]](#footnote-8)

###### Flow & User Experience

Below is a diagram that outlines the flow and user experience of Kipitup’s Apple Watch app. The various ‘screens’ that are presented to users are numbered corresponding to the order in which users would see them.



*Figure 2 - User flow of of Kipitup’s watch app*

**Screen 1 - ‘Onboarding (Greeting)’**

* This screen introduces the app, and describes why we will need to collect user data in the following 4 screens.
* *Note Regarding Onboarding: This screen, and the rest of the screens titled ‘Onboarding’ are only presented the first time the user opens the app. Subsequent uses will take the user immediately to the ‘Main Screen’*.

**Screen 2 - ‘Onboarding (Gender)’:**

* The user is presented with a vertical scrolling screen that allows the user to use their finger to drag up and down to ‘spin’ the wheel. 3 options are presented to the user (‘Male’, ‘Female’, and ‘Other’). The user must select one option to move to the next screen. After selecting a value, the user would tap the ‘Continue’ button to move forward.
* *Note Regarding Onboarding: This screen, and the rest of the screens titled ‘Onboarding’ are only presented the first time the user opens the app. Subsequent uses will take the user immediately to the ‘Main Screen’*.
* *Note Regarding Inclusivity: We, as product designers, still have doubts regarding the way this screen is being presented. Sex is certainly not binary, nor is it ternary as we are presenting it here. While we could have included a multitude of additional choices that reflect other gender identities, we decided not to as ultimately our model (and the data in our corpus) didn’t give different advice for those gender distinctions like it does for ‘Male’ and ‘Female’. Additionally, the wording for the third option “Other” may reinforce the idea that non-binary users are ‘othered’ in society, inadvertently speaking to a claim that they don’t belong. We want to acknowledge that the list of options presented in this screen is not making a declarative claim regarding gender identities, and by no means are these options representative of the totality of gender identities.*

**Screen 3 - ‘Onboarding (Weight)’:**

* The user is presented with a free text field in which they may input their weight using Apple Watch ‘scribble’ commands or by using voice commands. Users **do not** have to input data into this field to progress. Regardless of whether they input data into this free text field, the user would tap the ‘Continue’ button to move forward.
* Currently, weight is only given in kilograms (kg). In reality, no unit of measurement is actually associated with the weight the user enters in the Swift code for this app. The unit of measurement is associated with the inputted number on the ‘back-end’ by the LLM after it receives data from the Apple Watch. In the future, we plan to add an additional dropdown menu alongside the free text field here that would allow the user to choose alternative units of measurement (e.g. ‘kg’, ‘lb’). This dropdown field would default to the unit of measurement common to the region the user is in.
* *Note Regarding Onboarding: This screen, and the rest of the screens titled ‘Onboarding’ are only presented the first time the user opens the app. Subsequent uses will take the user immediately to the ‘Main Screen’*.

**Screen 4 - ‘Onboarding (Age)’:**

* The user is presented with a free text field in which they may input their age using Apple Watch ‘scribble’ commands or by using voice commands. Users **do not** have to input data into this field to progress. Regardless of whether they input data into this free text field, the user would tap the ‘Continue’ button to move forward.
* *Note Regarding Onboarding: This screen, and the rest of the screens titled ‘Onboarding’ are only presented the first time the user opens the app. Subsequent uses will take the user immediately to the ‘Main Screen’*.

**Screen 5 - ‘Onboarding (Height)’:**

* The user is presented with a free text field in which they may input their height using Apple Watch ‘scribble’ commands or by using voice commands. Users **do not** have to input data into this field to progress. Regardless of whether they input data into this free text field, the user would tap the ‘Continue’ button to move forward.
* Currently, height is only given in centimeters (cm). In reality, no unit of measurement is actually associated with the weight the user enters in the Swift code for this app. The unit of measurement is associated with the inputted number on the ‘back-end’ by the LLM after it receives data from the Apple Watch. In the future, we plan to add an additional dropdown menu alongside the free text field here that would allow the user to choose alternative units of measurement (e.g. ‘cm’, ‘ft’). This dropdown field would default to the unit of measurement common to the region the user is in.
* *Note Regarding Onboarding: This screen, and the rest of the screens titled ‘Onboarding’ are only presented the first time the user opens the app. Subsequent uses will take the user immediately to the ‘Main Screen’*.

**Screen 6 - ‘Main Screen’:**

* The user is presented with a splash screen with our ‘Kipitup’ title and a single button to interact with. The user would tap the ‘Ask me’ button to move forward.
* This screen is meant to be the first screen in the flow once onboarding has been completed at least one time. Additionally, after a query is inputted (occurs later in the flow), users would return to this screen.
* At this point, the data that was gathered during the onboarding process is aggregated and associated with the wearer’s user profile.

**Screen 7 - ‘Query Input’:**

* The user is presented with a free text field in which they can input their fitness-related query using Apple Watch ‘scribble’ commands or by using voice commands. Of course, since a question is naturally going to be significantly more characters than one’s inputted age or height, it is recommended to use voice commands to input a query rather than Apple Watch’s sometimes cumbersome ‘scribble’ commands. Users **must** input data into the field to progress.
* At any point, the user could tap on the ‘Back’ button to return to **Screen 6 - ‘Main Screen’**.
* After adding characters to the text field, the user would tap on the “Send” button; after which, their query and the data that they provided during the onboarding process are concatenated and sent to a Lambda function in AWS. The ‘Send’ button then changes to a button labeled ‘Continue’ (replaces the position of the ‘Continue’ button). The user would then tap on ‘Continue’ to move forward to the next screen. After the user taps on ‘Continue’, their query and the data that they provided during the onboarding process are concatenated and sent to a Lambda function in AWS.
* Our LLM then takes the data housed in AWS and evaluates whether or not the query provided is relevant to fitness. If the query is relevant to fitness, the user is directed to **Screen 8a - ‘Query Confirmation’.** If the query is not relevant to fitness, the user is directed to **Screen 8b - ‘Query Rejection’**.

**Screen 8a - ‘Query Confirmation’:**

* The user is presented with a confirmation message that their query has been accepted, and that they **will receive a response** from our LLM.
* The user would then tap ‘Return’ to go back to **Screen 6 - ‘Main Screen’**.

**Screen 8b - ‘Query Rejection’:**

* The user is presented with a pithy message that their query is not related to fitness, and that they **will not receive a response** from our LLM. The message displayed is customized to the query that was given. For example, if the query was “Nutella”, the message might read “Sounds like you’re hungry. Grab a snack, then ask a question related to fitness!”.
* The user would then tap ‘Return’ to go back to **Screen 6 - ‘Main Screen’**.

##### Prompt Engineering

In the initial phase of the project, our goal was to evaluate how the GPT model could generate customized workout plans using basic user information such as height, weight, age, and gender. This was to understand the variations in workout recommendations when provided with minimal data compared to highly detailed biosensory data.

Gaining insights from this comparison allowed us to refine our prompts more effectively, leading to more accurate and optimized workout routines. Consequently, we structured the language model behavior training into two distinct segments:

###### Understanding the behavior of custom GPT, in ChatGPT interface

We began by developing a custom GPT model tailored for marathon-related topics. This was accomplished through ChatGPT, where we uploaded 15 documents pertinent to marathons and clearly defined the model's scope and intended role from a macro perspective. All interactions and instructions were conveyed via the ChatGPT conversation text box UI, without involving any programming.

After the model was established, the next step involved testing its capabilities. We initiated this phase by providing the GPT with basic prompts to gauge its responses and assess how well it understood and processed the marathon-specific information. We started with a general prompt, shown below:

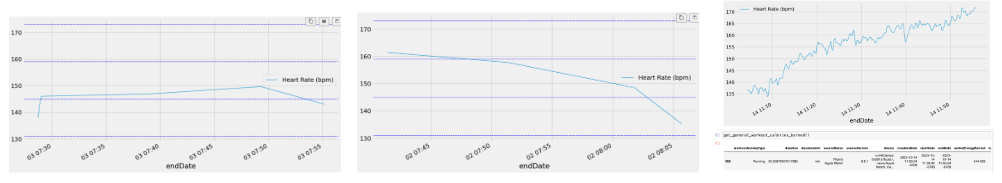
*For multiple runs, the image shows the data of heart rate, calories burned, and duration of the run (with duration and distance, you can calculate the average pace of the runner)*

*This runner is planning to run in a 10k marathon coming up in the next 12 weeks, generate a detailed marathon training routine. Based on the data provided, identify whether the user is working out at an easy, moderate, intense or high intense pase. Share a routine accordingly*

*Be sure to be as detailed as possible, Include*

*The exact exercises, reps, time, duration any laps (if needed), any inclination (if needed), rest times, specific weights for any weight training, Rest days, Intervals (in the case of interval running...), RPE (Rate of Perceived Exertion*

*Include any other details from your knowledge and docs you were trained on (Basically be as detailed as possible)*



*Figure 3 - Sample biosensory data used to train GPT*

With this our GPT model provided generic workout routines with instructions like 'Body weight exercises = 20 reps.' However, this guidance was too vague, as it did not specify which body weight exercises to perform, leading to ambiguities for the users.

To enhance the specificity and clarity of the responses, we delved into a series of educational materials, including videos and documents, to better understand the science of effective prompting. Applying these insights, we refined our approach iteratively, retaining successful modifications and discarding ineffective ones.

Subsequently, we began to incorporate more detailed user information, such as height, weight, age, and gender, into our prompts and expanded them to include basic nutritional macros. At this stage, we were still assessing the variety of response formats ChatGPT produced, which varied from paragraphs and bullet points to tables.

Recognizing the need for consistency in the output format, we experimented with different templates until we found one that yielded consistently detailed and structured responses. After enough instructions and prompting, we had achieved fairly detailed, but extremely consistent responses. Next step was to start working with the GPT API where we could pass the actual biosensory data and train it to improve accuracy of the training routine.

###### Understanding the behavior of GPT-4-turbo API (via assistants)

Having gained a clear understanding of how to structure our prompts and the kind of responses we could expect, we initiated the process by replicating the same prompt to see if it would consistently generate the same response. This step was crucial to ensure reliability in the model’s behavior.

Once consistency was confirmed, we shifted our focus to refining the specificity of our instructions to the LLM (Large Language Model). We recognized that LLMs perform optimally when prompts are clear and unambiguous. To enhance precision and reduce potential errors, we decided to eliminate graphs from our prompts and instead provide precise, direct data for analysis.

Consequently, we began to revise our prompts to be more straightforward and data-focused. An example of such a modified prompt is as follows:

*Can you provide a detailed workout plan for to run a full marathon in 6 weeks? Some of my important recent personal biosensory data are: 1) VO2Max = 55.13 mL/min.kg ; 2) Longest ran = 21km in 1 hour and 50 minutes; 3) HeartRateVariability = 121 ms; 4) Frequency of runs = 4 times a week.*

We began incorporating biosensory signals into our prompts, aiming to leverage the combined knowledge from the internet, the training documents we provided, and real-time biosensory data to generate more precise marathon training routines. The results were significantly improved compared to those from the custom GPT model used via the ChatGPT interface. This encouraged us to further refine our approach by including detailed instructions in the prompts, specifying a clear response template for the model to follow.

For instance, instead of a vague instruction like "Tuesday: 4 sets of squats," we required the model to provide detailed guidance such as "Tuesday: 4x15 squats, start with 35lbs, increment by 5lbs." This level of specificity allowed the model to utilize data like breathing patterns, heart rate, age, gender, height, and weight to tailor the exercise intensity appropriately.

With the detailed and consistently formatted training routines in place, we then shifted our focus to testing the model under various edge cases to identify potential limitations. One issue we encountered was when asking for a marathon training plan spanning a year. The GPT model provided detailed workouts for the initial 6-10 weeks but then generalized the remaining weeks with instructions to "increment by 10-15% every week." This behavior is likely a result of the token limit in GPT-4-turbo's response capability.

###### Final Prompt Iteration

Formatting instructions and response guidelines are hard coded in the prompt, and only need the user’s query and biosensory data (sent from the Apple Watch) to generate a meaningful response. Ultimately, this is our final prompt which has provided us with most accurate, customized routine in a consistent template:

*Can you provide a detailed workout plan for to run a full marathon in 10 weeks?*

*Some of my important recent personal biosensory data are:*

*Date VO2 Max (mL/min.kg )*

*3/5/2024 55.36*

*3/6/2024 45.36*

*….*

*Date Resting HR (BPM)*

*3/5/2024 140*

*3/6/2024 148*

*….*

*Date Time duration Average heart rate Exercise list*

*2024-03-05 00:00 - 8:00 77 Resting, sleeping, cycling*

*2024-03-05 8:01 - 16:00 94 resting, sleeping*

*….*

*Date Exercise Average pace (min/km) Distance (km) Duration (mins)*

*3/5/2024 Run 4.52 12.70 121.78*

*3/6/2024 Run 4.34 8.01 146.28*

*….*

*Number of exercises*

*20*

*Date Calories burned*

*3/5/2024 1831*

*3/6/2024 450*

*….*

*Height: 160 cm, weight: 80Kgs.*

*Based on the data provided, identify whether the user is working out at an easy, moderate, intense or high intense pace. Share a routine accordingly. Be sure to be as detailed as possible, Include The exact exercises, reps, time, duration any laps (if needed), any inclination (if needed), rest times, specific weights for any weight training, Rest days, Intervals (in the case of interval running...), RPE (Rate of Perceived Exertion. mention the intensity levels, give basic macros. Also give details of different zones and its values.*

*Include any other details from your knowledge and docs you were trained on (Basically be as detailed as possible).*

*Let the response be in this format:*

*Key Considerations:*

*(share the biosensory data attached and comment on its intensity for showing it back to the users)*

*Workout intensity level:*

*(based on analysis of their history of biosensory data)*

*Heart rate zones*

*Zone 1:*

*Zone 2:*

*Zone 3:*

*Zone 4:*

*Zone 5:*

*Max HR:*

*Marathon training plan:*

*Weeks x - y (based on the prompt)*

*Monday:*

*Tuesday:*

*Wednesday:*

*Thursday:*

*Friday:*

*Saturday:*

*Sunday:*

*make sure to add workout details for every weekly*

*For the above exercise list make sure to be extremely detailed and specific for the details and be accurate and be effective*

*Nutritional Guidance (Basic Macros):*

*Protein:*

*Carbohydrates:*

*Fats:*

*Additional Considerations:*

*Gear:*

*Warm-Up and Cool-Down:*

*Injury Prevention:*

*Monitoring Progress:*

*Weekly Check-Ins:*

*Flexibility:*

##### Large Language Model (LLM)

Kipitup utilizes the generative capabilities of two distinct Large Language Models (LLMs) to generate personalized training routines: GPT-4 Turbo and GPT-3.5 Turbo. These models differ in their training datasets and context window sizes, as summarized in the table below[[10]](#footnote-9).

| Model | Context window | Training data |
| --- | --- | --- |
| gpt-4-turbo | 128,000 tokens | Up to December 2023 |
| gpt-3.5-turbo | 16,385 tokens | Up to September 2021 |

*Table 1: Large-Language models deployed on Kipitup*

In the Kipitup pipeline, each LLM serves a specific purpose, based on their performance advantages and associated costs:

* **GPT-3.5 Turbo:** Functions as an intention classifier.
* **GPT-4 Turbo:** Generates complex recommendations.

At the initial stage of user interaction, Kipitup needs to discern the user's intent due to the open-ended nature of queries. For example, if a user issues a prompt unrelated to running, activating the recommendation system is unnecessary. In this role, GPT-3.5 Turbo assesses the intent and, as an agent, decides whether to initiate the recommendation pipeline. Optimized for Chat Completion tasks, GPT-3.5 Turbo operates efficiently without relying on additional files or data from OpenAI's infrastructure and with a shorter context window. Therefore, adopting GPT 3.5 Turbo results in faster computational performance and lower operational costs, making it ideal for the intention classification task.

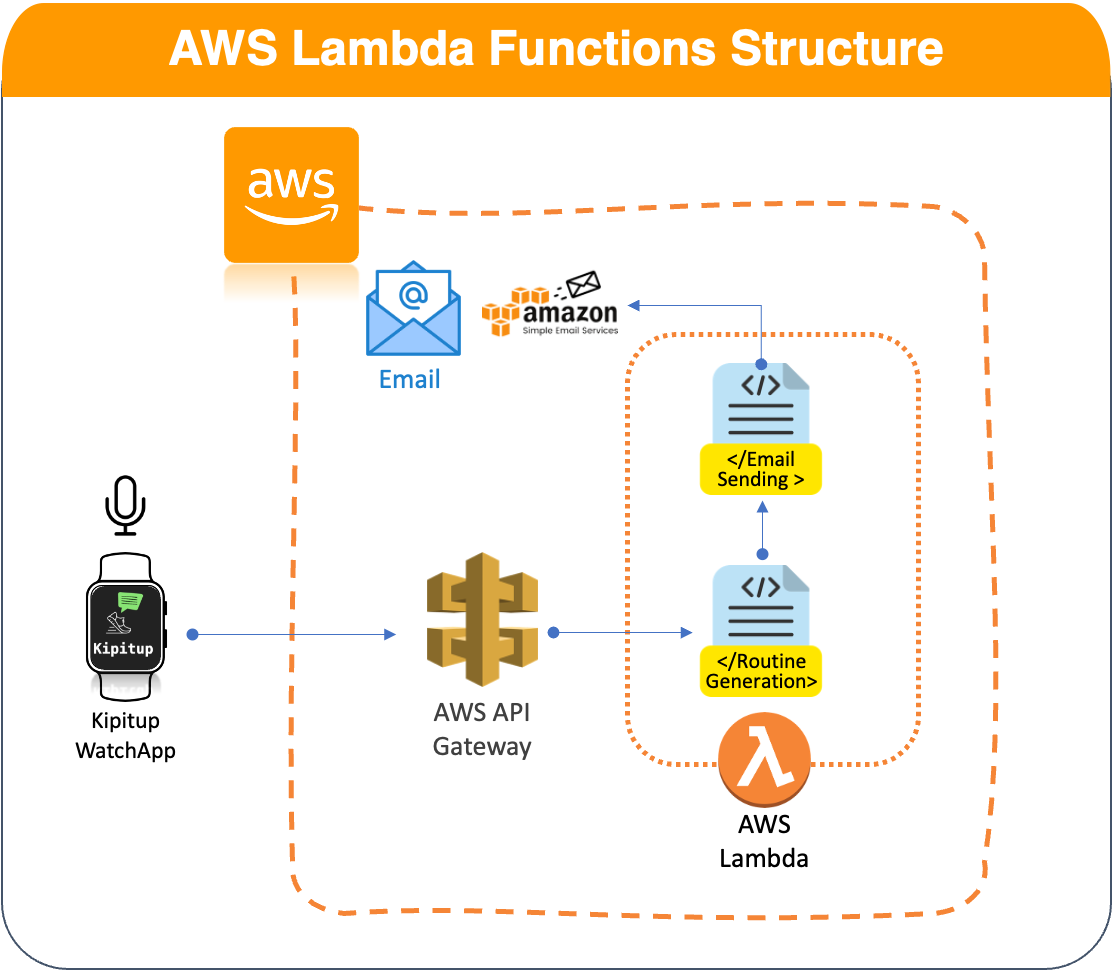
When the initial model identifies an intent aligned with the expectation to create a running preparation routine, a second LLM is triggered to generate the actual routine under specified parameters. With a much larger context window (128,000 tokens versus 16,386 from GPT-3.5 turbo) and more updated training data (including upwards to December 2023), GPT-4 turbo was the natural choice for generating the final recommendation plan. For this assignment, we implemented the Assistants API[[11]](#footnote-10) with previously defined instructions.

To enhance the quality of the model's recommendations, we also integrated a previously curated, domain-specific corpus. To preserve the capabilities of the original LLMs and enrich the quality of the model’s outputs with a corpus of marathon running best practices, while minimizing cost, we opted for the Retrieval-Augmented Generation (RAG) approach instead of model fine-tuning. As explained by Balaguer et al., the suitability of each alternative varies according to the specific application (Balaguer et al., 2024). Considering the tradeoffs and project constraints, RAG proved to be a more suitable solution for this project. The implemented method leverages OpenAI's infrastructure to segment uploaded documents, embed these segments, and store them in a vector database. The retrieval process uses OpenAI's standard method, employing cosine similarity as the distance metric to find the most relevant embedded document fragments.

##### AWS Lambda

We adopted AWS lambda functions as the computing service to host our Python code for scalability and to ensure a streamlined flow in the cloud. AWS lambda is a serverless service that can be triggered by different events and offers seamless integration with other AWS services, such as S3, DynamoDB, Simple Email Service, and API Gateway. Lambda functions are essentially designed to be cost-effective rapid solutions for multiple types of products, where specific environments can be easily set up, including Python, C#, and Node.js[[12]](#footnote-11). To guarantee affordability, AWS Lambda functions have a maximum time-out of 15 minutes and limited maximum processing capabilities. Therefore, the entire code must be considered, considering those constraints. However, one can also integrate a pipeline by invoking multiple sequential lambda functions to perform longer tasks.

For the project, we designed two primary AWS lambda functions: one to receive the prompt from the Apple Watch through an API Gateway and generate the response through the LLMs, and another that retrieves the LLM response, formats the string as a user-friendly report, and sends an email to the user with the formatted content using AWS Simple Email Service. The following figure briefly illustrates the flow of these two AWS lambda functions.



*Figure 4: Summarized structure of the AWS Lambda Functions part of Kipitup*

## How to Access Kipitup

Kipitup is currently publicly available to download as a local file, which can be run on any macOS device. Below are the steps required to download and start using the Kipitup Apple Watch app.

Step 1: Download the latest build of Kipitup from the Kipitup website. The download is a .zip file that contains everything necessary for the application.

* <https://kipitupmims.github.io/>

Step 2: Unzip the downloaded file. Doing so will unpack a folder titled *‘Kipitup Capstone Project (Apple Watch App)’*.

Step 3: Use a IDE capable of running Swift code to open the *‘Kipitup Capstone Project (Apple Watch App)’* folder as a project. We recommend using the IDE “Xcode”, which is an Apple-developed IDE, and is the most common IDE used for building and running iOS applications and watchOS applications. The following instructions will assume that Xcode is the IDE being used to run the project.

Step 4: Connect your Apple Watch to Xcode as a device, by navigating to ‘Devices and Simulators’, and adding the phone that is paired with your Apple Watch as a device. More information about how to pair devices and troubleshoot is contained in the link below:

* <https://developer.apple.com/documentation/xcode/running-your-app-in-simulator-or-on-a-device>

Step 5: Select your Apple Watch as the destination for the upcoming build.

Step 6: Press the play button near the top of the screen. This will create a build of the

Apple Watch application directly to your watch.

Step 7: Start asking questions, and continue your fitness journey!

## 

## Ongoing Development

##### Ideas for Continued Development

During the course of building Kipitup, we had several ideas for how to expand and improve upon our product that we were not able to implement by May. We’ve collected and tabled these ideas in the event that we decide to continue developing Kipitup post-graduation, which we have provided in a list ranked by priority below:

1. **Add ‘Weight Lifting’ as a use case** - Currently, Kipitup is limited to ‘running’ as its only use case. This is of course not comprehensive of all the activities that consumers do to stay healthy and in shape. Another common activity that is largely unrelated to the cardio-specific advice we provide is weight lifting or strength training. Weight lifting, much like long-distance running training, benefits from personalized advice that is unique to the user. It also benefits from long-term advice that gives incrementally more challenging tasks to the user. The parallels between weight lifting and long-distance running training make this a natural extension of Kipitup’s capabilities.
2. **Add alternative cardio use cases** - In a similar vein, we would like to expand Kipitup’s use cases to include alternative forms of cardio training such as cycling and swimming. This addition would dramatically expand Kipitup’s potential population and would only require a minimal amount of work (e.g. new corpus, prompt engineering).
3. **Add historical advice screen** - Our initial wireframes of the Kipitup app included screens that displayed previous queries asked by the user and the output Kipitup generated for those queries. This would allow the user to check back in on fitness advice that they’ve received directly from their Apple Watch. We tabled this due to the challenges involved in displaying often-lengthy advice plans on a fairly small Apple Watch screen. These are the same challenges that led us to the decision to output our fitness results to the user’s email, rather than displaying it directly on the Apple Watch. Still, we believe that a historical record of previous queries would be a useful addition to the product. It could be that instead of displaying the entire string of generated advice, we give a brief summarized version that would fit on a watch screen. Or, we integrate Apple’s ‘Mail’ app with Kipitup so that when a user taps on a previously generated response, it would automatically switch to the ‘Mail’ app and navigate to the email associated with that response. This feature improvement requires more solution engineering to truly be seamless, but we believe that some version of it would improve our product’s functionality.
4. **Building for disability** - As mentioned above, Kipitup is not built for those who are differently abled. It assumes some standardization regarding body capabilities when generating fitness advice. An iteration of Kipitup that better serves those who are differently abled might include additional parameters during onboarding to capture physical differences; for example, we could add optional text fields that ask the user to describe any physical disabilities that they identify as having. Of course, we would also need to change our corpus and our prompt engineering methodology to ensure that these parameters actually are reflected in the advice that we generate. Of course, even this addition wouldn’t appropriately address the needs of every type of physical disability. How would a visually impaired user respond to this new text field? While we don’t expect building for disability to be a simple task that is quickly solved, we would like to at least begin the process of bringing differently abled users into the fold.
5. **Add ‘Diet Planning’ as a use case** - Early on, we considered adding ‘diet planning’ as a use case and having Kipitup be a health and lifestyle project, rather than just a fitness product. Similar to ‘weight lifting’ as a use case, effective diet planning would require long-term advice and incremental progress check-ins, which directly aligns with how Kipitup structures its advice. Expanding to include diet planning would require some careful consideration for how we were to give advice and how we were to use the consumer’s biosensory data; but, we see this as a challenge worth tackling in the near future.
6. **Multiple languages** -Currently, Kipitup is presented only in English. Translating the text that is displayed on the watch’s interface and ensuring that the LLM that generates responses can respond to non-English languages would dramatically expand our potential user base. We would determine which languages should be prioritized by looking at the rate of Apple Watch adoption internationally. This prioritization would also partially be dependent on capabilities of natural language processing in non-English languages being considered.
7. **Multiple devices** - We may consider expanding our software to other wrist-based wearables using different operating systems (e.g. Garmin, Amazfit Band, Samsung Galaxy Watch). Additionally, we may expand the scope of our primary data sources to include supplemental sensors and other wearable devices as well (e.g. Oura rings, smart treadmills).

##### Lessons Learned in Hindsight

We encountered our share of roadblocks and challenges over the 9 months of development that ultimately led us to the product we have today. With the knowledge that we have now, there are certainly some things that we should have done differently in order to avoid these roadblocks.

**Lesson 1: Start early**

Like any good project manager, we assumed that we could be far more productive than we could in reality. Since everyone on this team is working on a space that is entirely new to them (e.g. LLM dev., Swift coding), we ran into multiple issues that we couldn’t have anticipated that led to delays. While curveballs are inevitable, we believe that some of them could have been mitigated were we to reduce the time spent on the *research and ideation* phase and increase the time spent on the *development* phase of this project. Essentially, the earlier we hit those inevitable roadblocks, the earlier we overcome them.

**Lesson 2: Iterate user flow more**

After sharing early builds of the Apple Watch app, we noticed several different tweaks that could be implemented to improve the user experience. Sometimes these tweaks were minor and easy to implement (such as formatting font and color scheme); sometimes these tweaks required significant modifications and took a lot of time to implement (like adding entirely new screens or changing how variables were passed to our AWS instance). We realize now that if we spent more time iterating the user experience through wireframes and low-fi mockups, we could have identified these requirements before actually building the app in Swift. This would have saved us significant development time and reduced the amount of prototype builds necessary to get to our final product.

**Lesson 3: Define and document success earlier on**

Several times during development we encountered ‘showstopper’ issues that seemed to prevent us from achieving the product that we had in mind. These showstopper issues were either a result of us planning interactions that were technically not feasible, or us planning interactions that required more time or resources than we had. Whenever we hit a showstopper issue, we had to connect as a group and discuss what our plans were to move forward and arrive at a product that still meets our requirements. We mentioned defining an MVP and the core requirements of Kipitup earlier in the ‘Defining Success’ section of this paper. These showstopper issues would have been less stressful and would have required less discussion if we defined several versions of what a successful product would look like in the early stages of development, instead of having to scramble to define success after we hit a showstopper roadblock. Eventually, we did define and document several different MVPs which made future roadblocks easier to circumvent; but, doing so earlier would have certainly saved us some trouble.

## 

## References

1. Runningwithgrit. “Statistics about Running; Facts about Runners.” *Running With Grit*, 7 Jan. 2022, runningwithgrit.com/statistics-about-running/.
2. *How Often Do You Run in Comparison to Other Runner’s World Readers? And How Does Your “easy Run” Pace Compare?*, www.runnersworld.com/uk/news/a45036050/runners-world-running-survey-2023/.
3. Sellers, Dennis, et al. “Apple Watch Dominates the Market with 50 Million Units Sold in 2022.” *Apple World Today*, 13 June 2023, appleworld.today/apple-watch-dominates-the-market-with-50-million-units-sold-in-2022/.
4. Laricchia, Federica. “Global Smartwatch Shipments Market Share 2023.” *Statista*, 28 Feb. 2024, www.statista.com/statistics/910862/worldwide-smartwatch-shipment-market-share/.
5. “Swiftui.” *Apple Developer Documentation*, developer.apple.com/documentation/swiftui/. Accessed 24 Apr. 2024.
6. “Urlsession.” *Apple Developer Documentation*, developer.apple.com/documentation/foundation/urlsession. Accessed 24 Apr. 2024.
7. “Using LLMS for Intent Classification.” *Conversational AI Platform*, 18 Apr. 2024, rasa.com/docs/rasa/next/llms/llm-intent/.
8. Balaguer, A., Benara, V., Cunha, R. L. de F., Estevão Filho, R. de M., Hendry, T., Holstein, D., Marsman, J., Mecklenburg, N., Malvar, S., Nunes, L. O., Padilha, R., Sharp, M., Silva, B., Sharma, S., Aski, V., & Chandra, R. (2024). RAG vs Fine-tuning: Pipelines, Tradeoffs, and a Case Study on Agriculture. arXiv preprint arXiv:2401.0840
9. *Models - Openai API*, platform.openai.com/docs/models. Accessed 30 Apr. 2024.
10. *Assistants Overview - OpenAI API*, platform.openai.com/docs/assistants/overview. Accessed 30 Apr. 2024.
11. *What Is Aws Lambda? - Aws Lambda*, docs.aws.amazon.com/lambda/latest/dg/welcome.html. Accessed 30 Apr. 2024.

1. https://appleworld.today/apple-watch-dominates-the-market-with-50-million-units-sold-in-2022/ [↑](#footnote-ref-0)
2. https://www.statista.com/statistics/910862/worldwide-smartwatch-shipment-market-share/ [↑](#footnote-ref-1)
3. https://runningwithgrit.com/statistics-about-running/ [↑](#footnote-ref-2)
4. https://www.runnersworld.com/uk/news/a45036050/runners-world-running-survey-2023/ [↑](#footnote-ref-3)
5. https://appleworld.today/apple-watch-dominates-the-market-with-50-million-units-sold-in-2022/ [↑](#footnote-ref-4)
6. https://www.statista.com/statistics/910862/worldwide-smartwatch-shipment-market-share/ [↑](#footnote-ref-5)
7. https://rasa.com/docs/rasa/next/llms/llm-intent/ [↑](#footnote-ref-6)
8. https://developer.apple.com/documentation/swiftui/ [↑](#footnote-ref-7)
9. https://developer.apple.com/documentation/foundation/urlsession [↑](#footnote-ref-8)
10. https://platform.openai.com/docs/models [↑](#footnote-ref-9)
11. https://platform.openai.com/docs/assistants/overview [↑](#footnote-ref-10)
12. https://docs.aws.amazon.com/lambda/latest/dg/welcome.html [↑](#footnote-ref-11)