Design Document – Smart Marathon running app

| Submission date: 02-14-2025  Student Name: Luka Brennan  Student Code: C00272285  Supervisor: Chris Meudec |
| --- |
| [**1. Technologies used 2**](#_tzjgcbbtbkht)  [1.1 Android Studio - IDE 2](#_og7s2kagb4q4)  [1.2 Frontend - XML 2](#_8wnc06klr16b)  [1.3 Backend - Java 2](#_ejjhadga9t5g)  [1.4 Postman 3](#_7pb1syjin1r0)  [1.5 JSON 3](#_allqwqm55t5f)  [1.6 FireBase database 3](#_3u7288rsgmuf)  [1.7 Strava API 3](#_q7r00kchb9l5)  [**2. Problems encountered 3**](#_ab26fqt4r338)  [**3. Algorithms 5**](#_n3soeudwn7tn)  [3.1 calculate pace per second Algorithm [3] 5](#_cw5l9mj7n3an)  [3.2 Format pace Algorithm 5](#_sn6ckysrm5l9)  [3.3 Pace adjustment algorithm 5](#_priz4m3hfest)  [3.4 Filter Runs 6](#_z65ywgfpf65q)  [3.5 Fetch Activity 6](#_cnsrk5ljo7sq)  [**4. Pseudocode for algorithms 6**](#_peggslc60g5h)  [4.1 calculate pace per second Algorithm 6](#_at968x52e7h5)  [4.2 Pace adjustment algorithm 6](#_t7p3mkvpsu4n)  [4.3 Filter Runs 7](#_iy72sglp9lcj)  [4.4 Fetch Activity 7](#_mbfq1juf018s)  [**5. UI 8**](#_23n46gp3510c)  [**6. What Platform will it be on 11**](#_cocjjyj249zn)  [**7. What Database will be used 11**](#_hjj583q0o3en)  [**8. Class Diagram 12**](#_z4oqieu0e1ty)  [**9. References 12**](#_ao3yeqax4as) Technologies used1.1 Android Studio - IDE The project spec specified creating an app. For this Android Studio will be used as the IDE as I have experience with working with it. Android Studio also provides both Frontend and Backend development (Frontend - XML, Backend - Java), this will help with developing the app. //Note maybe add pictures for the technologies. 1.2 Frontend - XML In Android Studio, XML files are built into the “layout” package. This is where XML files for the “Screens” go into. For instance the main screen for the first iteration is named “activity\_main.xml”. This screen will show the current finished run and the previous run before that, then will provide a new pace increase or decrease depending on the difference between the two runs data. 1.3 Backend - Java There are two different languages to use when creating a new activity in android studio, Kotlin or Java. Java was used for this as I have experience with working with this language.  Java is used to write the code to implement the necessary classes and interfaces for the project. For the interface classes code that is implemented here allow API calls from the app to STRAVA API. 1.4 Postman Postman[1] is a software testing tool which is used for testing api’s, as this project is mainly focused on the Strava api, it was necessary to use this tool. What was it used for?Postman gives the ability to refresh the access token of a runner without having to rely on the runner to authenticate the app continuously. All of the GET and POST commands were taken from Strava’s api doc[2]. 1.5 JSON The app from the previous iteration only used XML for displaying the information. Although this is good for a layout for displaying the data it is not a good choice. As the data would need to be hard code in java as well as xml it would cause too much headache and problems later down the line. With the implementation of JSON this problem has been resolved and allows for data to be better displayed and dynamically adjusted. 1.6 FireBase database As there has not been any implementation using a database in the app currently. During the next iteration this will be used as the STRAVA app will be put into production mode, allowing for more runners to use the app. The database will hold all unique access\_tokens from the runners. 1.7 Strava API The aim of this app is to use the Strava app to help runners perform their best while training. Strava API allows the app to acquire unique access tokens from runners that have authorized the app on their device. These will be stored in a Firebase database. Problems encountered There were many problems when it came to coding the app, these problems took many hours to fix and correct. Below are some of the errors that were encountered and how they were fixed.  The **first** of the problems was a “bad request message” - **{ "message": "Bad Request", "errors": [ { "resource": "Application", "field": "redirect\_uri", "code": "invalid" } ] }**  Once the first iteration began, there needed to be a runner to pull data from to begin coding the MVP. This was originally going to be done through the app by having a “Login” feature that would, once opened, bring the runner to Strava’s auth screen to authenticate the runner to the app, then would redirect them back to the app. This would not work as initially thought, as the redirect\_uri would not work correctly, also the access token and refresh token (which are needed to trade for access to the runners strava data) would not be provided this way.  To fix this issue and acquire the runners access token and refresh token, the use of Githubs “pages” was used, this allowed a html webpage to be made, allowing for html code to allow the runner access to the auth screen for strava to authenticate the app, then redirect them to a (“callback\_uir”) which would then show the runner there access token and refresh token. Currently the runner must send that data to the programmer in order to hardcode it into the app to retrieve that runner's Strava data.    The second problem was related to access tokens. The error message is the same as the 1 error, but in place for the “redirect\_uri” is accesstoken. This error was very simple in hindsight but when it first happened was quite hard to understand. To put it simply the app was trying to exchange the access token for access to the runners strava data. However this token was either expired or wrong. At the time when this happened the idea of access tokens being valid for only 6 hours was not something that was known. To fix this issue, making a POST to post man with the runner refresh token would provide a new access token to be used.  Another problem that was encountered was this error message - **Plugin [id: 'com.android.application'] was not found in any of the following sources:**. This error message occurred after I committed working code to github. Prior to the commit the code was working and there were no issues, it was only after the commit that this error message was popping up. The message was indicating errors with the gradel files in the apps Project build.gradel.kts file. No changes were made between the last commit and the commit that was made that broke this code, so there were no changes to this file. Many hours were spent trying to fix this issue, but to no avail. The only solution that could be made is to create a new android project and paste the code into it. This worked but now all the commits are being sent to the “master” branch in github.    Issues with the V02max → displayed wrong information.  There were a few problems that occurred when trying to use different models of v02max. As I'm using v02max to get a pace adjustment for the runner, I used different models such as the Cooper Test[4], and Jack daniles vdot[5]. During testing using the different models, problems occurred, such as the pace being unrealistic, the pace would show up as 0:10 for a recovery run. The only possible way this could be possible is if the runner was running at a pace of 0:20 - 0:15. This is just not possible. After a few hours of trying to figure out why this was happening another model was used. This one was slightly better but was also not very good. It was then decided that the assistance of AI would be used. I asked ChatGPT, that I was experiencing problems using the Coopers test for calculating V02max. And gave it my method that was used for the calculations. What it gave me in return was a better improvement to what I had originally, however I still felt like the data it was showing was not 100% accurate, this could be due to the fact that the AI did not actually give me a solid and official model/algorithm unlike Cooper or Daniels, it was a mixture o the two. Currently this is what is being used as of the end of iteration two. This is planned to be fixed during the last iteration.  During iteration two, what was being displayed to the android screen was mainly all XML code. This included the UI and the text that was being displayed, This was used for the majority of the iteration two, until a meeting with my project supervisor who provided me with information to use JSON instead of purely relying on XML, this proved to be a much better way compared to using just XML, as the problems that would be occurring for the app would be data not being properly displayed and the lack of ability to dynamically update and change the already existing hardcoded text int XML. The aim was to have the text be hardcoded for the runners 1st run so that they would have a rough understanding of what was needed in their run, then after the data would be changed (pace) to better fit their current fitness to better ease them into training. As what was in the app originally made this very difficult, the change to use JSON has proved to be a great help. 3. Algorithms3.1 calculate pace per second Algorithm[3] What is the aim for this algorithm? To be able to calculate the pace for the current finish and previous run. This algorithm uses this formula to calculate pace. Pace = MovingTime (seconds)/Distance(km).  As Strava or at least the get https, that was used during the first iteration - [https://www.strava.com/api/v3/activities/](https://www.strava.com/api/v3/activities/12788659484). This GET returns the latest activities of the runner, providing data for calculations but no data on pace. Due to this as the MVP for the app is to provide a dynamic increase or decrease to the runner's pace for their next run, this would need to be coded in on the app's end. This algorithm will get the pace in second. 3.2 Format pace Algorithm Why is this algorithm needed? In Strava the pace per split looks like this: 6:28 (6 minutes and 28 seconds). As currently the pace is just the pace in seconds, having it appear in a fashion that runners will know will help them read it more clearly. The algorithm used for this:  Minutes = pace per second/60  Seconds = pace per second mod 60  Example: Minuets = 290/60 = 4.833  Seconds = 290 mod 60 = 50 3.3 Pace adjustment algorithm The algorithm works as follows. If the runner is following the training plan that is selected, (currently only week 11) the app will check the date of the run taken from Strava to ensure that they are training that exact day on the training plan. The first run’s pace goal will be a static pace, this is to have a goal in mind for the runner, the reason for this is so the app will be able to get the necessary data from the runner to use in the Pace adjustment algorithm to dynamically update future runs. Once the runner completes their first run the app will provide a more suitable and achievable pace for the next run to start off the training. If the runner fails or over achieves in their training, the pace will either increase or decrease depending on their run data taken from Strava. The adjustments to the pace will be small to allow the runner to maintain improvements.  As for how it actually works, Strava keeps data of the runners heart rate (avg heart rate highest heart rate) and the distance. The algorithm will be using a version of V02max algorithm, as of now the only data that is being used is the heart rate and distance, but more data such as elevation and cadence can be used with v02max to give a more accurate pace. This is the aim for the next iteration and to finalise the algorithm. It currently works ok but needs more refinement.  The aim of this algorithm will be to allow for consistent growth of the runner to better them for their marathon, taking into account over excretion (making sure to take into account if their heart rate is way too high) and to ease recovery with realistic pace goals. 3.4 Filter Runs This aims to filter all the activities from the runners Strava so that only activities of type “Run” are accepted. 3.5 Fetch Activity This algorithm fetches the latest activities and displays relevant statistics.. 4. Pseudocode for algorithms4.1 calculate pace per second Algorithm distanceKm = distanceMeters / 1000  paceInSeconds = movingTime / distanceKm  return paceInSeconds 4.2 Pace adjustment algorithm paceArray = Split pace by " - "  Create an empty list adjustedPaces  baselineVO2Max = 40.0  adjustmentFactor = 1 - ((vo2Max - baselineVO2Max) / 100)  For each pace in paceArray:  parts = Split pace by ":"  If parts has at least 2 elements:  minutes = Convert parts[0] to integer  seconds = Convert parts[1] to integer  totalSeconds = (minutes \* 60) + seconds  totalSeconds = totalSeconds \* adjustmentFactor  adjustedMinutes = totalSeconds / 60  adjustedSeconds = totalSeconds % 60  adjustedPaces.Add formatted pace as string ("adjustedMinutes:adjustedSeconds")  Return Join adjustedPaces by " - " 4.3 Filter Runs runs = empty list  for each activity in activities:  if activity.type == "Run":  add activity to runs  return runs 4.4 Fetch Activity response = api.fetchActivities(accessToken, page=1, perPage=10)  if (response.success and response.data not empty){  runs = filterRuns(response.data)  if size of runs >= 2:  compareRuns(runs[0], runs[1])  else  displayRunStats(runs[0])  } 5. UI **Figure 1** - Iteration one UI    **Figure 2** - Iteration two UI    **Figure 3** - Iteration two UI  This image is taken from the app. For the first iteration the app will show the current completed run and the previous run, then compare them and provide a new pace for the next run. The pace provided will be the average pace min/km.  This new image is the current UI for the app for iteration 2.  Insert image here →  As seen the new UI incorporates a week training plan (currently only week 11 the first week of the training), this new version shows the day, distance and the pace. With the pace being changed dynamically based on whether or not the runner is following the training plan or not. 6. What Platform will it be on Because the app is being made with **Android Studio**, the published version of the app will be exclusive to **Android** apps. Perhaps later on once the app becomes successful an IOS version could be released. 7. What Database will be used As the app is still in **development** mode for **STRAVA**, this means that only 1 person can use the app. Due to this the only runner that is authorized on the app info is hardcoded into the code. Once the app goes into **production** mode in **STRAVA** this will allow more people to use it. Once this happens a database will be used to store the unique **access\_tokens** of the new runners that use the app to properly display their training plan. So when the time comes to use this database, the database that will be used will be **FireBase.** 8. Class Diagram  9. References [1] - www.cirruslabs.io. (n.d.). *6 Reasons to use Postman/Newman for API Integration Testing*. [online] Available at: <https://www.cirruslabs.io/blog1/modernized-technology/6-reasons-to-use-postman/newman-for-api-integration-testing>.  [2] - developers.strava.com. (n.d.). *Strava Developers*. [online] Available at: <https://developers.strava.com/docs/reference>. [Accessed on 05/12/2024].  [3] - Calculator.net. (2024). *Pace Calculator*. [online] Available at: <https://www.calculator.net/pace-calculator.html>.  [4] - Cooper Test [online]. Alvero-Cruz, J.R., Carnero, E.A., Giráldez García, M.A., Alacid, F., Rosemann, T., Nikolaidis, P.T. and Knechtle, B., 2019. Cooper test provides better half-marathon performance prediction in recreational runners than laboratory tests. *Frontiers in physiology*, *10*, p.1349.  [5] - Jack Daniels running formula [Book]. Daniels, J., 2013. *Daniels' running formula*. Human Kinetics |