Initial Document: A Multifunctional Fitness Application & Vertical Jump Height Calculator

Rowan Aldean
973765

Submitted to Swansea University in partial fulfilment
of the requirements for the Degree of BSc (Hons) Software Engineering w/ Year in Industry

Swansea University
Prifysgol Abertawe

Department of Computer Science
Swansea University

20th October 2021

Declaration

This work has not been previously accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed ..... (candidate)

Date .....

Statement

This work is the result of my own independent study/investigations, except where otherwise stated. Other sources are clearly acknowledged by giving explicit references. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure of this work and the degree examination as a whole.

Signed ..... (candidate)

Date .....

Contents

Chapter 1

Introduction

This document will outline the plan for development of a functional codebase for a fitness application. This process, herein referred to as "the project" is different to the application(s) produced as a result. The main difference is that the project is not concerned with the deployment, maintenance and future updates of the fitness application and no additional academic benefit is gained from creating the infrastructure or platform-specific functionality. This is to say that the creation of **both** a mobile and a web application is not needed for the project but makes up a core offering of the product for end users following deployment.

A **mobile** application will be produced during the project, which will have 2 main use cases:

1. A user will be able to pay (via monthly installments or upfront) for a fitness plan; where they can monitor, make notes and track their progress and routine via a calendar.

2. A user can create their own fitness plan by producing their own routine using available exercises on the app - the additional functions such as the calendar remain the same.

1.1 Motivations

The motivation for the project began following an initial consultation with a Californian NBA-level basketball athletic trainer who was looking for a bespoke application to reduce his business expenses. Initially, 4 days of planning and a 40-minute discussion took place before the business decided that the costs associated with building this platform are not a priority for the time being.

In the process of considering the development of said fitness platform, I spoke to developers working with smaller high-level sports trainers regarding the technology used and their clients future plans. At this time fitness influencers Krissy Cela and Jack Bullimore had proven the concept a year earlier with the rollout of "Tone and Sculpt" - grossing over GBP 1 million in the 8 months following its January 2019 release [1]. My initial desire to pursue a project in this space grew more after hearing the story and approach taken by "Tone and Sculpt" on the Diary of a CEO podcast [2].

In the world of athletic sports there is currently no viable, consistent and widely used means of measuring vertical leap besides the Vertec. Prices on these devices can be as high as £500 and cheaper alternatives commonly found at a youth/amateur level - that involve markings on walls - don't allow for an approach jump (without the risk of hitting the wall). This is being mentioned because a large motivation for the jump calculator functionality is to allow an easy and accessible means of measuring a users vertical leap progress. Naturally, the use of a smartphone for measurement is not completely accurate but will provide sufficient consistency (given the user's setup) and this can be used to monitor improvement.

From a trainers' perspective, using PDFs to release fitness plans is cumbersome and involves finding a suitable workflow and rollout. Further to the fact that they're easily copied and distributed. Existing white-label[1] mobile app solutions to these issues don't offer much customization beyond simple re-branding; many don't support video. Following discussion with fitness trainers in various countries (including the USA, UAE and UK) I concluded that this is something that needs to be addressed. Many robo-fitness apps exist for end users, but few include real trainers and even less give freedom to the administrator.

[1] A white-label product is a product or service produced by one company that other companies rebrand to make it appear as if they had made it.

In summary, the motivation for the project is to give a customizable fitness planning solution to those interested in monitoring their training. This solution should be usable by administrators (trainers/teams), average "gym go-ers" and highly trained athletes alike. The global fitness app market size was valued at USD 4.4 billion in 2020 and is expected to expand at a compound annual growth rate (CAGR) of 21.6% from 2021 to 2028 [3]. We can see that the total addressable market is huge and set to grow quickly, if this is any indicator of the volume of potential users looking for a fitness solution then the proposed project should prove tremendously useful at solving the aforementioned problems.

1.2 Overview & Aims

The projects aims include the following - some of these aims will be broken down into succinct functional and non-functional requirements in Section 3.2:

- The ability for a user to plan their workouts on a calendar.
- The ability for a user to track the progress of a workout.
- The ability for a user to measure their vertical leap using their phone.
- Allow users to keep track of their progress in relevant areas such as weight, strength measurements etc. more efficiently than traditional methods.
- Allow trainers to comfortably deliver fitness plans to clients without a steep technical learning curve.

The core overview of the project is that we're looking to develop a mobile application that allows a user to register an account before having the ability to follow/create a fitness plan and provide relevant tools for consistency and the successful reach of a users goals (irrespective of their background). The project aims to focus on the end user[2] however considerations will be made for the complete implementation of an administrator role (time permitting).

The project is concerned with the ability to solve both parties problems through the development of the application, but the primary focus is on the development of the core functionality and the vertical jump calculator. From a business perspective we can consider the jump calculator the unique selling point of *the product* and a complex component of *the project*.

[2]The user training using the app

Chapter 2

Project Research

We've previously seen that "Tone & Sculpt" is an existing similar application on the market. Below we'll consider other background research on existing systems and their functionality.

The bulk of fitness mobile applications fall into 2 categories:

1. Applications which allow fitness enthusiasts to monitor their training and nutrition - the market leaders in this space (by monthly active users) are "Fitbit" and "My Fitness Pal" (as of March 2018) [4].

2. Applications which allow trainers to facilitate the delivery of programmes either by partnering with the development company or white-labelling software. These are apps focused on trainers more than end users - the most notable player in this emerging market would be "TrueCoach", who raised USD 2.6 million in seed round funding before being acquired by TSG (Transaction Services Group) in April 2020 [5]. Estimated revenue over USD 1.2 million (from 20k users) for 2021 [6].

We can see the main difference in these groups are that the focus of the former is B2C (business-to-consumer) services whereas the latter focus on B2B (business-to-business; often acting as SaaS companies). This is an important distinction as *the project* is looking to combine elements of both and so the following research will reflect this.

We'll also look at vertical jump calculators and the mathematics involved in developing something of that nature.

2.1 Existing fitness application systems

We'll look at 4 systems, 2 from each of the aforementioned application groups. Here it is worth mentioning that some apps are on the borderline such as "Centr by Chris Hemsworth". The app follows a B2C approach but is marketed and part-owned by Chris Hemsworth, with his trainers being the facilitators of programmes and nutrition plans. A similar approach to the aforementioned "female-focused Tone & Sculpt". Because cases like these are not relevant to the application being built for the project, we will not look in any depth at these.

For each of these systems we'll be looking at:

- A typical user story for the given application.
- A high-level breakdown of the system(s) functionality.
- A look at the technology in use (where possible).

We'll then compare the applications (in pairs) to see any unique features and to understand which functions are seemingly core to a positive user experience in the fitness application space.

2.1.1 MyFitnessPal

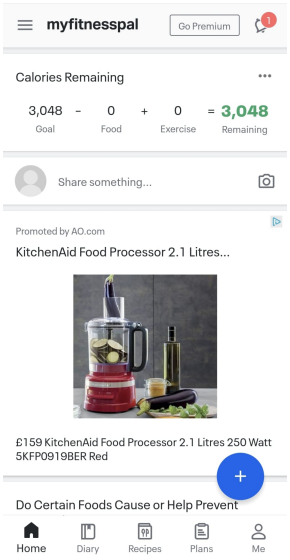


Figure 2.1: Homepage feed

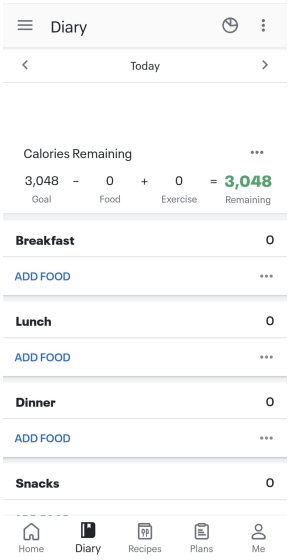


Figure 2.2: Food diary & calorie counter

User Story

"As a fitness enthusiast, I sign up to the app via email. I wake up every day and log my breakfast by scanning barcodes for my food. This is accounted for in my daily food tracking (Fig. 2.2). After breakfast I create a new workout based on advice from my community or homepage, I search for the exercises involved and log my workout later on in the day (Fig. 2.3)."

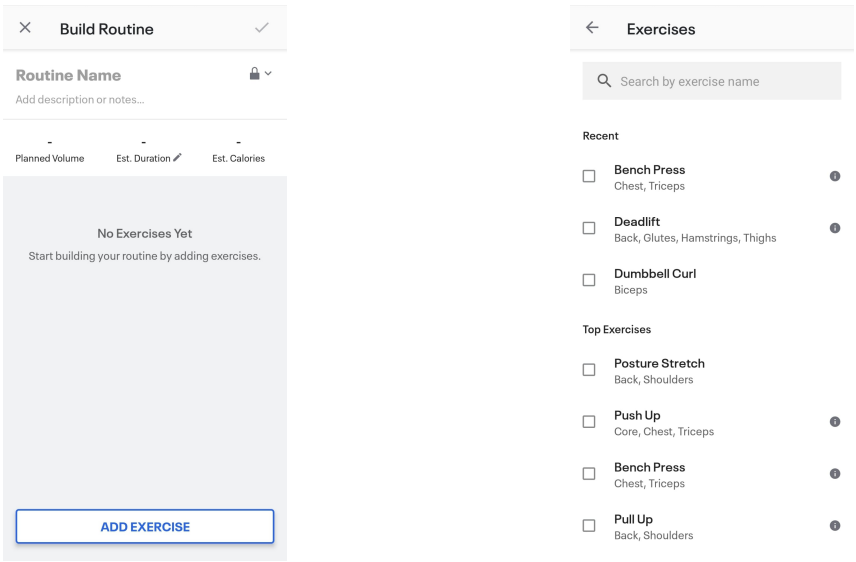


Figure 2.3: The custom workout routines in MFP

Features/Functionality

MyFitnessPal includes all of the following features:

- Estimate calories burned given a set of exercises/a routine.
- Track food intake via barcode scan or recipe input.
- Track water.
- Workout plans provided by MFP.
- Track weight.
- Create custom workout plans from existing exercises.
  - Exercises don't include video/images and no ability to add custom exercises.
  - These can be shared publically to dashboard.
- Count steps (using compatible devices).

- Add friends and send/receive messages.
- Community tab (requires separate signup).
- Export progress as CSV.
- Meal and weight tracking reminders (via push notifications).
- Set workout/week, minutes/workout and "calories burned" goals.
- Discover recipes/meal plans via community and MFP suggestions.

Technology Stack

MyFitnessPal uses the following technologies according to Stackshare [7]:

- React (a JavaScript framework for creating cross-platform applications)
- Cloudflare (for web infrastructure)
- NginX (for server side use)
- Vanilla JavaScript and CSS + libraries (jQuery, Bootstrap)

Various other technologies are being used on the server-side but for *the project* we are primarily concerned with the technology used for the application as opposed to the backend infrastructure.

2.1.2 Fitbit

Fitbit is the app that accompanies many of the Fitbit brands' hardware products, their first movers advantage was likely a large contributor towards their large market share (in 2018)[1] in the application space. The functionality of their hardware products (which had been the standard for fitness tracking wearables) is enhanced through use of the app. The research will consider the application alone and disregard any wearable-specific functionality.

[1] Q3 2020 wearables market share = 2.6%[8]

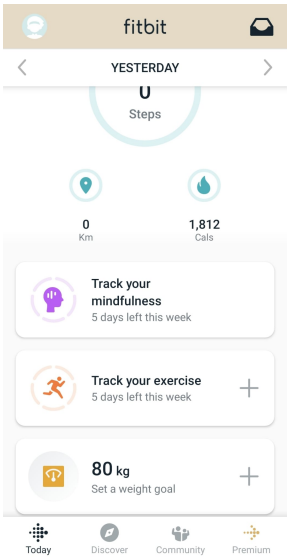


Figure 2.4: Dashboard

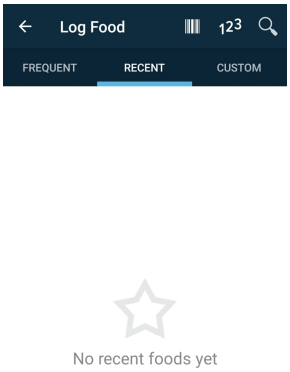


Figure 2.5: Food log screen & features

User Story

"As a professional athlete I wake up, go through guided meditation on my Fitbit calendar before weighing myself (Fig. 2.4). I then log my breakfast and eat (Fig. 2.5), followed by my commute to practice and logging my sport and duration (Fig. 2.6)."

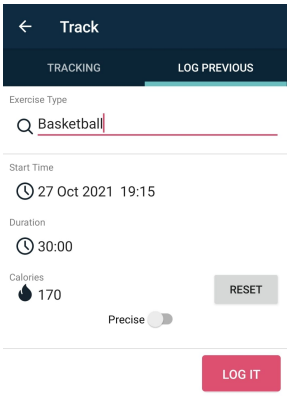


Figure 2.6: Using exercise search & log

Features/Functionality

Fibit includes all of the following features:

- Guided meditation (w/ simple week-calendar scheduling).
- Guided video workout plans (w/ simple week-calendar scheduling).
- Guided programmes (for meeting targets in health, not just fitness).
- Challenges (1-10 people, gamifying targets).
- Water tracking.
- Food tracking.
- Track exercise (only by sport/activity, no custom option and no individual exercises/routines).
- Step counter.
- Community feed and groups.
- Add friends and send/receive messages.

Technology Stack

Fitbit uses the following technologies according to Stackshare [9]:

- JavaScript(and Ember.js) for the front end.
- Fastly (for cloud content delivery)
- Redis (for middle layer and database caching)
- Node.js (for producing backend APIs)
- Java (and Spring) for the non-mobile applications
- MySQL (for the backend database(s))

Fitbit is cross-platform and multifunctional, thus there will be some technologies not in use by the mobile app we are researching. It's unlikely Spring is in use, but the use of JavaScript (w/ Node.js) for the front and backend is common in current world of software. It's highly likely the whole system is using a microservice architecture where components maybe used in several places.

2.1.3 Comparing MyFitnessPal and Fitbit

Firstly, we'll note that both applications make use of JavaScript for their front-end technologies. It's difficult to pinpoint what is being used without decompiling both applications (both of which have a huge codebase). It is apparent that MFP is using React[2]. Both apps are also using some form of CDN to speed up content delivery, which is needed given the abundance of data involved in the tracking of food and exercise. MFP offers developers access to their data set via their own API [10].

Whilst both apps calculate calories burned (an arguably useless metric for *our project*) during exercise, the scope and functionality of exercise is vastly different. Fitbit includes general sports and activities but no specific exercises (for example - "weights" is considered 1 exercise type and the calories burned are dependent on duration) and there is no ability to produce your own routine. On the other hand, MFP allows for the creation (and sharing) of your own routines using their database of exercises. There are no videos to follow for this component of the app. Both apps do however offer workout routines (and short-term programmes) via short videos published by the respective companies.

Both apps allow for the tracking of food and water intake, as well as the ability to scan product barcodes to log nutritional information. In regard to overall wellness, Fitbit seems to be covering more "bases" by providing guided meditation videos and challenges which can be taken alone or as a group. These challenges are a way to gamify the achievement of personal goals (such as using your phone less).

Finally, both apps include community features - very similar to traditional social media platforms. These and other nutrition focused features are beyond the desired scope of *our project* (Chapter 3) thus we won't be comparing and considering these.

Overall, the fitness aspect of MFP seems more useful, but there are elements of the app which are not yet on par with Fitbit - such as the goal setting and mindfulness measurements. That said, the UX/UI of MFP is more usable and the app has fewer features locked behind hardware devices (yet you can still pair compatible devices to MFP). Neither app has any major feature involving the camera besides the barcode scanning components. A combination of elements from both apps will be paramount to a successful project.

[2] an open-source UI software framework created by Meta, Inc (formerly Facebook)

The project is planned to have a side panel menu and similar minimal aesthetic to that of MFP (Section 4.2). It's not planned that it will include the calories burned estimates; however other metrics will be included in similar vain, such as total time trained and cumulative weight lifted/cumulative distance travelled. Calorie estimates will be an enhancement (additional feature) where time permits. The abilities to create custom routines and follow preset routines, which are featured in MFP and Fitbit respectively, are expected to be important features in *the project*. Albeit, the long term plan is to support the ability for trainers to set these programmes as opposed to presets alone.

In summary, the design and layout of these apps is what's useful for us - inspiration and additional features can also be drawn from these existing systems. With that said, *our product* is not a nutrition tracker and focusing on elements in this space would shift the scope of the *project* substantially (Section 5.4.2.3). An extensive breakdown of the feature set can be found in Section 3.1.

2.1.4 TrueCoach

TrueCoach provide a cross-platform system that includes a web application. We'll be considering the mobile app provided for end users and making reference to the functions available to administrators in the process. They provide a mobile app for each of these use cases - "**TrueCoach for Clients**" and "TrueCoach Connect" for trainers.

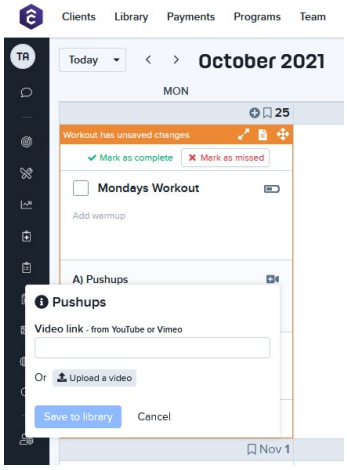


Figure 2.7: Trainer using web app to set clients workout(s).

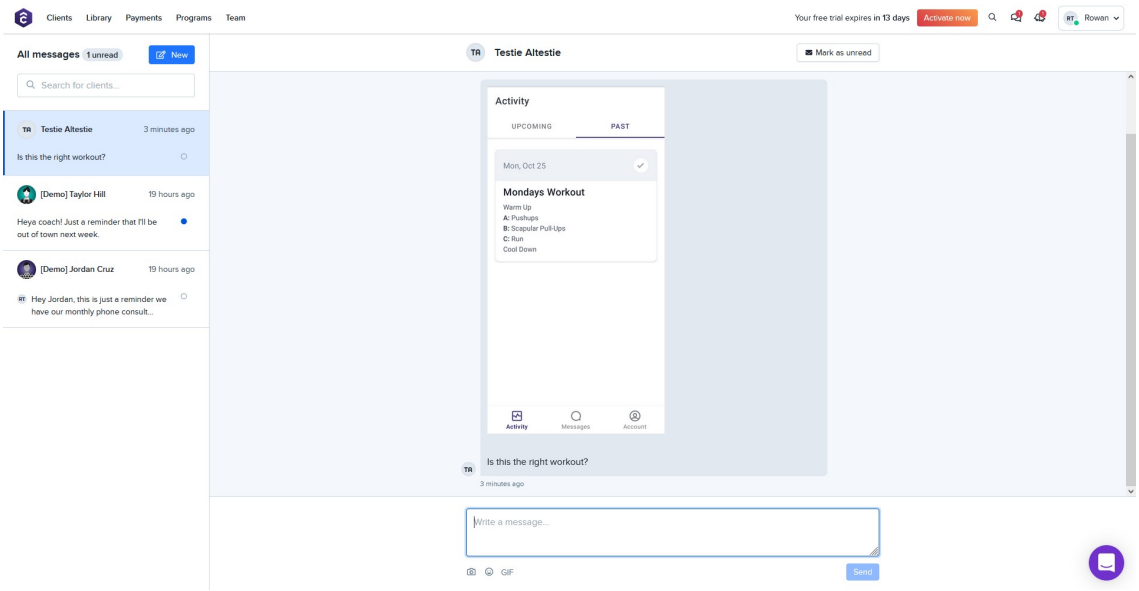


Figure 2.8: Trainer using web chat to manage clients.

User Story

"As a new gym go-er, I work remotely with my trainer (Figs. 2.8 and 2.10) to refine my training technique(s). I view my upcoming workouts (Fig. 2.9) and follow instructional videos before marking my workouts as complete and monitoring my progress via goals & challenges (Fig. 2.11). For advice I can view documents and progress photos in my settings page (Fig. 2.12)."

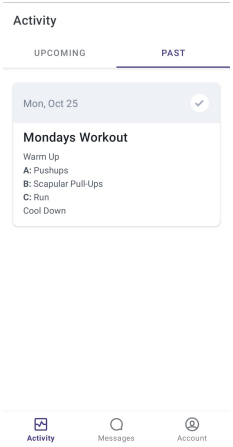


Figure 2.9: Viewing workouts (upcoming/past).

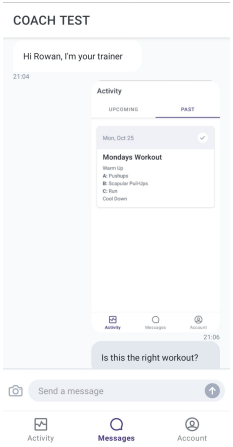


Figure 2.10: Instant messaging with a trainer.

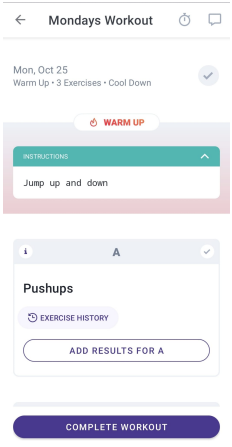


Figure 2.11: Logging a workout.

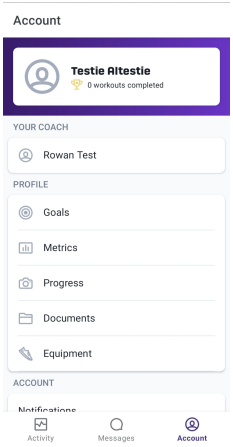


Figure 2.12: Viewing settings.

Above, "Testie Altestie" represents an end user and "Rowan Test" is the trainer. We can see the functions available to the user are dependent on the workouts uploaded/scheduled by the trainer using their interface (web application in this case).

Features/Functionality

"TrueCoach for Clients" includes all of the following features:

- Track coach's release of phased training (upcoming/past workouts).
- Log workouts (w/ notes).
- Timer & stopwatch function for tracking timed exercises.
- Video breakdowns of exercises (w/ rich text instructions).
- Instant messaging w/ trainer (including multimedia uploads).
- Tracking of progress via trainer-set goals & metrics.
- Progress and additional documents (outbound link to web app).
- Create account via invite link from trainer.
- View notifications.
- Edit settings (outbound link to web app).

TrueCoach covers a lot of needs for a user, and we can see in Fig. 2.8 the relevant tools for the trainer to facilitate these workouts. They are administrators of their clients and they're the means by which a user gets login credentials. Trainers can upload relevant videos (Fig. 2.7) and documents, as well as provide metrics to measure users progress.

Technology Stack

Stackshare (as used for the previous apps) is not useful in this case, so the following conclusions have been drawn following the decompilation of the TrueCoach APK [11]:

- Kotlin & Java (for Android development)
- Firebase (for push notifications and messaging)
- AWS Amplify (for backend deployment)
- Numerous open-source development packages such as glide.

After decompiling and opening the class files using dex2jar and JD-GUI, I found that the application is not using a cross-platform JavaScript framework as is standard in modern times. The app is using the recommended native language of Kotlin[3] and Java for all data objects and frontend display.

[3]Learn more: android development documentation

2.1.5 Exercise.com

Exercise.com are the company providing the white-labelled solution behind PJFPerformance. We'll be using the **PJFPerformance** app to analyse their application.

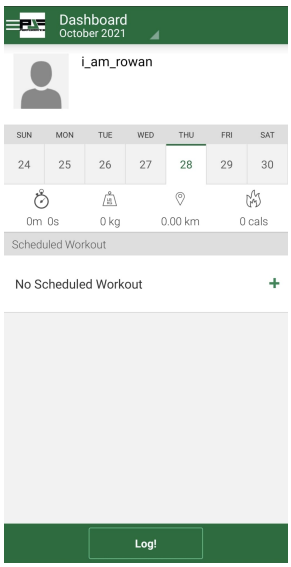


Figure 2.13: Home dashboard

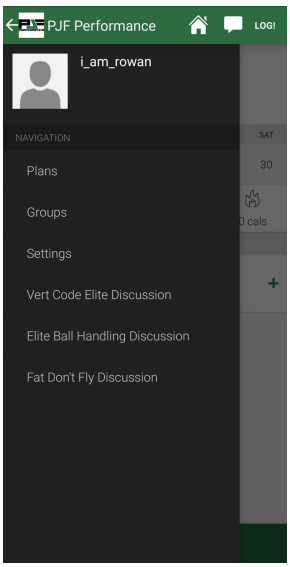


Figure 2.14: Sidebar menu options

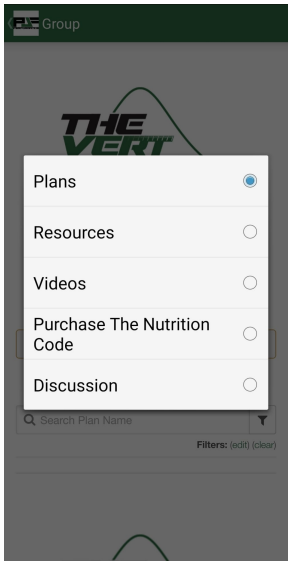


Figure 2.15: Viewing additional resources.

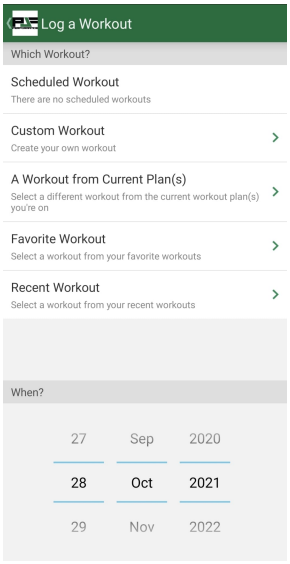


Figure 2.16: Scheduling/logging a workout.

User Story

"As a basketball player, I enroll to Paul Fabritz' workout plan using the web application before receiving credentials for the mobile app. Here I schedule the programme to begin at the start of the month and I allocate days of the week for my workouts (Fig. 2.13). I create my profile containing my weight and other metrics, so I can ask relevant questions to Paul and others on the same programme as me via a discussion room (Fig. 2.14). For guidance on training during the season and managing load, there is a resources section (Fig. 2.15) (outbound links). I log all my workouts and make notes for my own use. I sometimes create workouts for myself (Fig. 2.16) using some exercises available through the app. Whenever I train, I can see exercises explained via videos embedded from YouTube."

Features/Functionality

The PJFPerformance app includes all of the following features:

- Schedule workouts.
- Create custom workouts (using existing exercises).
- Log workouts (w/ notes).
- Enroll on phased programmes (made up of multiple workouts split over a given number of weeks & days).
- Edit profile.
- View settings.
- View notifications.
- Rest timer for timed sets in workouts.
- Videos (YouTube hosted) for exercises.
- Group discussions for programmes.
- View additional resources (hosted externally).

Exercise.com work on a 1-to-1 basis with trainers and white-label their solutions, this means any administrator view can only be presumed via the decompilation of the APK app file.

Technology Stack

Similarly to TrueCoach, finding the technology used by Exercise.com for the mobile application is only possible via decompilation using existing tools [11]. Based on my look into the source code, we can safely conclude that the app uses:

- Kotlin & Java (for Android development)
- Firebase (for push notifications and messaging)

- Amazon S3 (for data storage)
  - They expose data from the S3 instance via their web application API. This is where the mobile application is pulling data from.
  - The S3 instance is using some form of SQL database (we know this based on SQL migration scripts in the source code).
- Numerous open-source development packages such as FastJson.

Similarly to TrueCoach this app is written using Kotlin and not a cross-platform JavaScript framework. The MaterialDesign[4] influence is noticeable in this app and so it is less surprising. Performance to the User feels somewhere between average and slow for this type of app.

2.1.6 Comparing TrueCoach and PJFPerformance (Exercise.com)

TrueCoach has a larger focus on the client-trainer relationship and is more suited to the public (anybody can become an administrator and begin using their platform for their business). On the other hand, PJF provides a greater reliance on the community of people following the same programme as the User. Our project aims to blend both of these concepts; the core functionality will be developed first (focusing on the end user) before producing administrator views and allowing a many-to-many relationship between clients and trainers.

From a technical standpoint the implementation of the applications are similar (both using Kotlin and Java, with some use of open-source packages for specific needs - such as confetti in TrueCoach when completing a challenge). Both apps also make use of Firebase messaging for push notifications. This is easily implemented and simplifies the implementation of push notifications across platforms (Android/iOS). This is noteworthy for the development of our project and will prove useful if Firebase is used instead of or alongside the technologies outlined in Section 3.3. Our application frontend will be built using Flutter. The reasoning for this will be outlined further in the document, but performance is not expected to be adversely affected. Whilst the technology will be different the functionality and user experience aims to be similar to that of both apps.

TrueCoach has a simple 3 screens - logging and monitoring workouts (Fig. 2.11) has a much better UX/UI. The ability for a trainer to add new exercises and accompanying

[4]a design language invented by Google (Android parent company) in 2014.

video (both via embeds or uploading raw video) is unique (Fig. 2.7). These are features that are greatly beneficial to our project and reducing the number of app screens and keeping the interface minimal will both provide a much better final product and ease the time stress of development and testing. Only in the PJF app can a user create their own workout and log this (Fig. 2.16). This ability is core to our app. On the other hand, the TrueCoach app is entirely dependent on a trainer/administrator supporting you throughout your programme. Furthermore, TrueCoach allows for the measurement of set metrics (by the trainer) and goals to surpass. The PJF app lacks these features but does allow for further planning in the form of a calendar view and a "traffic-light system" for workouts on a daily basis. A red circle appears under missed workouts on the calendar, amber for pending workouts for the day and green for completed workouts. TrueCoach has personal goals but no extensive calendar view and there is a simple "mark as completed" feature for scheduled workouts. We plan for both of these elements to be included in our project; looking at the preliminary prototype (Fig. 4.2) and feature set (Section 3.1) should provide more clarity on how this will be achieved.

The final difference I consider notable between the apps is the use of timers. TrueCoach makes use of both a stopwatch and a timer for users to utilise during exercises. Alternatively, PJF includes a rest timer which Paul suggests to use for other functions but no stopwatch if needed (for example, when training for speed). We plan on following TrueCoach's example here and allowing users the option of both. Implementation is relatively easy and should enhance user experience.

Many of the differences outlined above come down to the business objectives of the relative applications. TrueCoach is building a SaaS company whereas Exercise.com is building a fitness application product that they can white-label to notable trainers in the fitness industry. This core difference means there are certain design decisions that differ between the apps. A common and core feature with both is the ability for users to view videos of exercises during workouts and to log there progress as they exercise. This is important to the success of *the product* and the approach we are looking at following involves a dashboard upon registration where a trainer can include their branding and thereby personalise the end user experience - similar to Stripe (Fig. 2.17).

This functionality falls beyond the scope of the outlined *project* but will help you form a long-term understanding of the proposed system.

There are many small differences both in the technologies used (such as AWS Amplify vs S3 storage and a self-hosted API) and in the feature sets. Ultimately, elements of both applications should be used in *the project* in order to produce a market-suitable *product*.

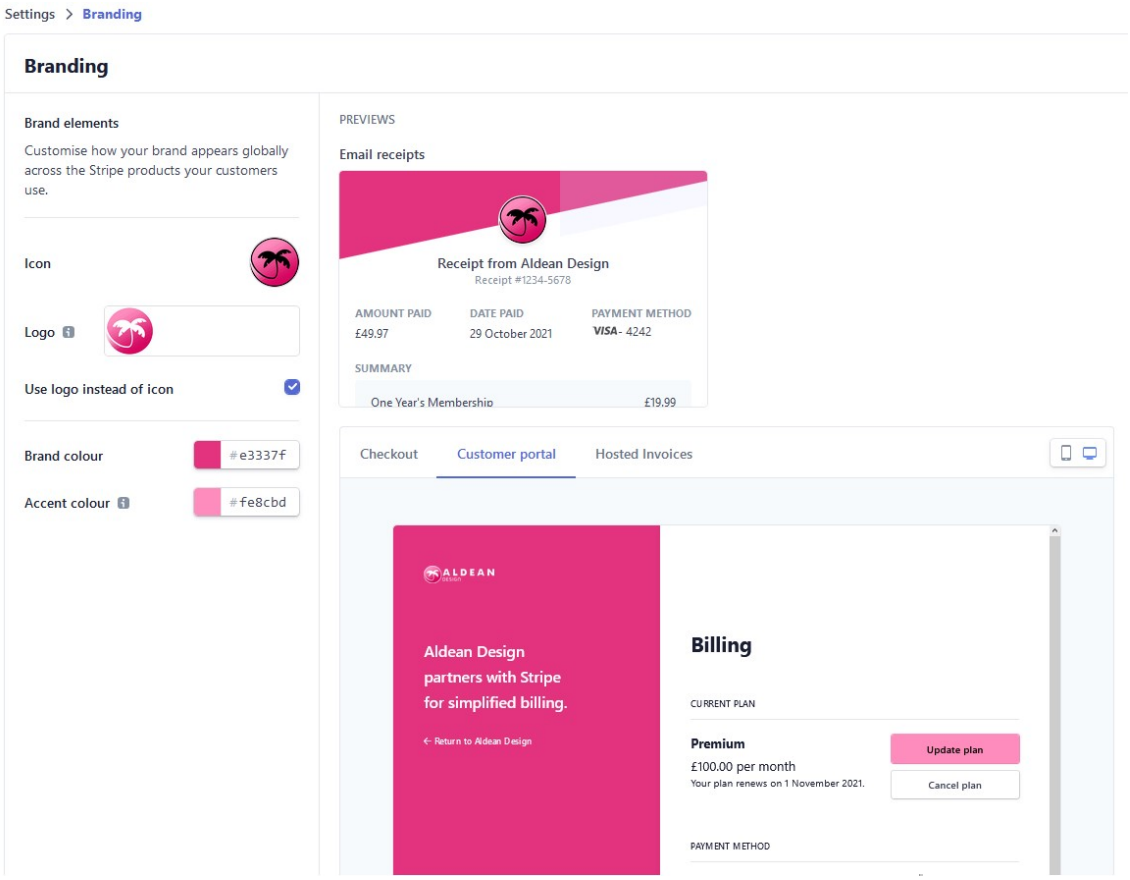


Figure 2.17: Customizing Stripe products using personal branding.

2.2 Existing vertical jump calculators

There are few *modern* jump calculators on the market right now. None are combined with existing fitness platforms - our research shows all of them being standalone mobile or web applications (2.2.2 to 2.2.4).

There are multiple methods of measuring jump height, often this is important to athletes where success is caused/correlated with jump ability i.e. Sprinting [12], American football, Basketball, Volleyball. We also know vertical jump is an important test to assess the explosive strength of the leg musculature of athletes [13, 14]. We'll be using the time-in-air (TIA) to estimate jump height. Whilst this is inevitably not perfectly accurate, when using a force plate [5] it is no worse than other methods (in terms of consistency) [15]. Thereby still proving useful to the target demographic.

To summarise, the vertical jump calculator created for the project is intended for use as a measurement of progress and not a concrete result. There will be some deviation from other measurements (such as using a Vertec).

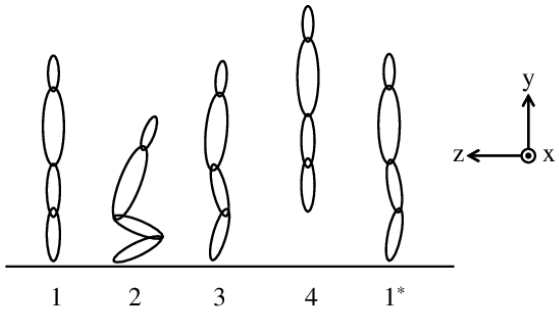
[5] A measuring instrument that measures the ground reaction forces generated by somebody.

2.2.1 Maths behind calculating vertical jump using video

Below we'll be considering the physics [16] involved in finding vertical jump height using **only smartphone video footage**.

To explain the forces demonstrated in a countermovement jump (CMJ)[6] we'll be considering the 5 phases of action involved (Fig. 2.18).

Figure 2.18: Motion phase of countermovement jump (CMJ). 1: Rest. Before jump.
2: Countermovement. 3: Push to ground.
4: Jump. Fly. 1 * : Rest. Land.
Vertical position of torso is y position [17].



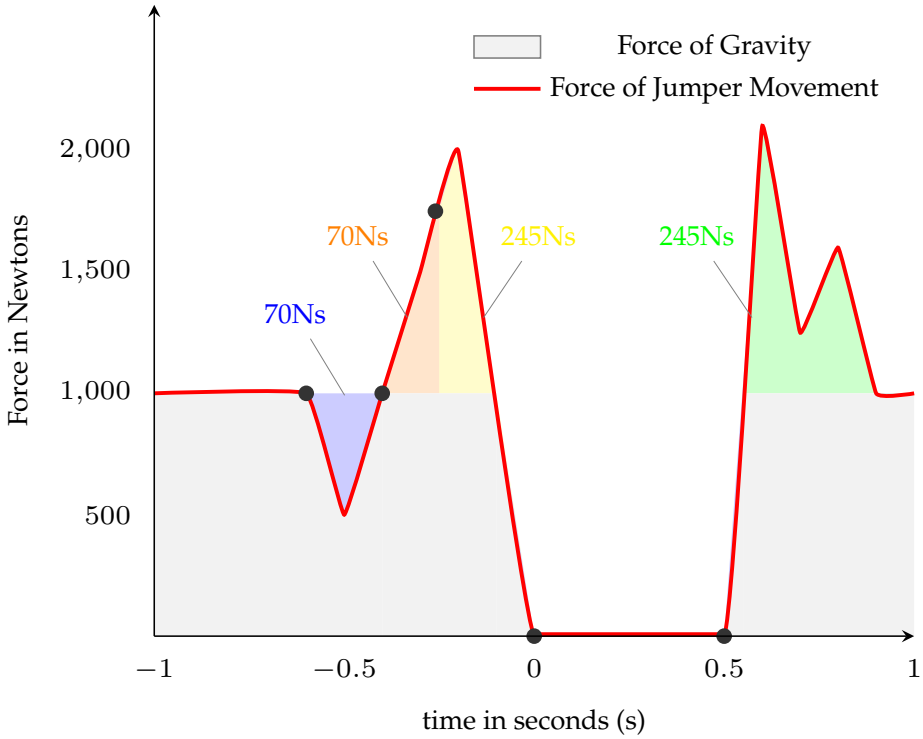


Figure 2.19: Ground reaction forces during vertical jump.

[6] A vertical jump test performed by having an athlete quickly squat to a self-selected depth and then jump as high as possible.

We'll be using the above, simplified force plate analysis (Fig. 2.19), to delve further into the phases of a jump and the mathematics. In reality the force curve wouldn't be so smooth but this should demonstrate the phases quite clearly.

Phase 1: Rest (Pre-jump)

Before jumping we can see a flat line at 981 Newton (Fig. 2.19). The athlete isn't jumping, but this is gravity doing it's work. It's easily explained as $F = m * g$, where m is the mass of the athlete and g is the acceleration of earth's gravity. F is the force needed to counter the effects of gravity. We know that $g = 9.81m/s^2$ on earth, thus:

$$981N = m * 9.81m/s^2$$
$$=> m = \frac{981N}{9.81m/s^2} = 100kg$$

Here you can see that the force plate is basically acting as a weighing scale, showing the force gravity exerts on the athlete.

Phase 2: Countermovement

As seen in Fig. 2.18, the countermovement involves the bending of the knees and typically swinging of the arms to lower the center of gravity before the jump. The force plate (Fig. 2.19) is registering forces lower than 981N, which means the athlete is accelerating in a downward movement.

Recall Newton's 3rd Law of motion; which states whenever two objects interact, they exert equal and opposite forces on each other. Knowing the above, we can define Ground Reaction Force (GRF) as "the force exerted by the ground on a body in contact with it". We can describe the forces acting at this time as follows:

$$F_{Jumper} = F_{GRF} - F_{Gravity} \leq 0$$

We can also use the analysis (Fig. 2.19) to find the speed the athlete is moving at before the jump. We know that $F = ma$ (mass * acceleration), and force is happening over time. Thus, $Ft = mat$. We also know force isn't a constant, it's a function of time and that we can define velocity as $v = at$; so our formula is:

$$\int_{t_1}^{t_2} F_{Jumper}(t)\mathrm{d}t = mv$$

where $F_{Jumper}(t)$ is the difference between the GRF and gravity. The integral can be

calculated from the force plate data. The downward impulse[7] made by the athlete is shown in the graph (Fig. 2.19) as the blue area below the line representing gravity.

We've assumed the integral (impulse) is -70Ns[8]. With this, we can conclude that:

$$\int_{t_1}^{t_2} F(t)\mathrm{d}t = mv$$
$$=> -70Ns = 100kg * v$$
$$=> v = -70Ns/100kg = -0.7m/s$$

Therefore, our athlete reaches a peak velocity of -0.7m/s during the countermovement phase.

Phase 2.5: Deceleration

Now we've seen where GRFs are lower than expected due to gravity - we must look for the period where our athlete has to decelerate the downward movement (pause) before launching up (Fig. 2.18). This should be an equally big impulse to that found above but in the opposite direction. We can describe this as:

$$\int_{t_2}^{t_3} F_{Jumper}(t)\mathrm{d}t = 70Ns$$

Because we know $F(t)$ and t_2, the algorithm of the force plate analysis now looks for t_3 so the impulse equals 70Ns.

On our graph (Fig. 2.19) we have pictured this - we are looking for the blue and orange areas to be the same size.

Phase 3: Push to ground

The athlete then generates force in preparation for takeoff before the GRFs register as 0 (when they are in flight). They reach some of the peak forces seen. To assess the velocity during takeoff we use the same technique as Phase 2:

$$\int_{t_3}^{t_4} F_{Jumper}(t)\mathrm{d}t = mv$$

Pictured in our graph (Fig. 2.19) as the yellow area.

[7]Impulse is the integral of a force.
[8]Newton seconds are the units of impulse.

The force plate graph detects 245Ns, so we can determine initial velocity as follows:

$$245Ns = 100kg * v$$
$$=> v = 245Ns/100kg = 2.45m/s$$

Phase 4: Jump. Flight! Here the athlete can't control their velocity any longer; their jump height is determined by the speed built before and during takeoff. Only gravity is acting on them, bringing them back to the ground.

We've determined initial velocity was $v(0) = 2.45m/s^2$ and the gravity of earth has an acceleration of $a = 9.81m/s^2$. Plus, the peak of the jump must have a vertical velocity of zero: $v(t_{peak}) = 0$.

Using the initial velocity and earths gravity we can find velocity at any given time using:

$$v(t) = v(0) - a * t$$

Thus we can find the velocity at the peak of the jump using:

$$v(t_{peak}) = v(0) - a * t_{peak}$$
$$=> 0 = v(0) - a * t_{peak}$$
$$=> t_{peak} = \frac{v(0)}{a} = \frac{2.45m/s}{9.81m/s^2} = 0.25s$$

With the knowledge of $v(t)$ during every moment and knowing the peak of the jump was 0.25s, we can calculate the jump height as the integral of velocity over the time it takes to reach the peak of the jump:

$$h_{jump} = \int_0^{\frac{v(o)}{a}} \left(v(o) - at\right) \mathrm{d}t =$$
$$= v(o)t \; - \; \frac{1}{2}at^2 \Big|_0^{\frac{v(0)}{a}}$$
$$= v(0)\left(\frac{v(0)}{a}\right) - \frac{1}{2}a\left(\frac{v(0)}{a}\right)^2$$
$$= \frac{v(0)^2}{a} \; - \; \frac{1}{2}\frac{v(0)^2}{a}$$
$$=> h_{jump} \; = \; \frac{1}{2}\frac{v(0)^2}{a}$$

This leaves us with a simple formula for finding jump height *when we know the initial velocity*. For our force plate analysis (Fig. 2.19) we get

$$h_{jump} = \frac{1}{2}\frac{(2.45m/s)^2}{9.81m/s^2} = 0.306m$$

Because we only need the initial velocity of an athlete to satisfy our formula, we don't have to rely on a force plate and expensive equipment. We simply have to find the initial velocity for a jump that takes 0.5s.

If someone jumps 0.5m high, they must also fall 0.5m after they peak. We have established velocity is a linear function ($v = at$), thus we can demonstrate the athletes peak is at the exact middle of the jump. Formally:

$$t_{peak} = 0.5 * t_{hangtime}$$

Thus to find the jump height of someone who jumps for 0.5s, we can calculate the distance of free fall in 0.25s!

$$\begin{aligned} S &= \int_0^{\frac{1}{2}t_{hangtime}} v(t)\ dt \\ &= \int_0^{\frac{1}{2}t_{hangtime}} a * t\ dt \\ &= \frac{1}{2}at^2 \Big|_0^{\frac{1}{2}t_{hangtime}} \\ &= \frac{1}{2}a\left(\frac{1}{2}t_{hangtime}\right)^2 \\ &= \frac{1}{8}a\ t_{hangtime}^2 \end{aligned}$$

In the case of a 0.5s hangtime, this gives us (as before with the force plate data):

$$S = \frac{1}{8}a\ t_{hangtime}^2 = \frac{1}{8} * 9.81m/s^2 * 0.5^2 = 0.306m$$

This formula is the crucial component to developing this feature. We won't be looking at the physics of "the landing" in this document as it is not directly relevant. However it is worth noting that to counteract the forces created during impact, the athlete braces themselves and absorbs force - this is notable as the green area in the graph has a value of 245Ns exactly like the yellow area.

2.2.2 FitnessMeter - Test & Measure

FitnessMeter is an iOS (Apple) exclusive mobile app, used for "fitness testing and athletic performance evaluation" [18].

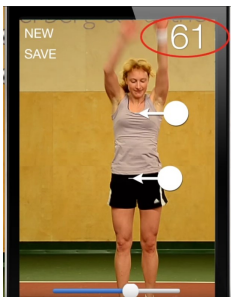


Figure 2.20: The FitnessMeter vertical jump calculator in use.

The app uses the change in position of the torso (Fig. 2.20) to determine jump height. Seemingly by comparing this to their position in the initial start. Our app aims to comfortably work even when only the feet are in frame. In traditional vertical jump tests the athlete maintains straight legs and so the issue of raising the legs/bending the knees to artificially increase hangtime is not a major concern. *Directions in our app will discourage this technique and establish how to best find accurate results.*

2.2.3 What's My Vertical?

Another iOS only application with poor UI. This application uses the same methods we plan on using; however, the app serves no purpose beyond the vertical jump measurement and "dunk calculator" math. It lacks further functionality and training support.

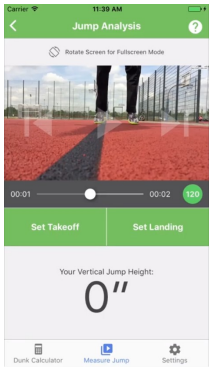


Figure 2.21: The jump calculator functionality in use.

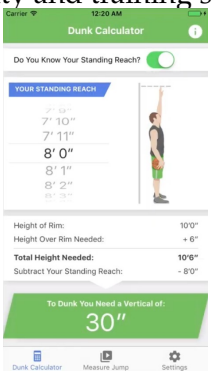


Figure 2.22: The dunk calculator math being used for estimation.

As mentioned, we intend on using a very similar system to what's seen here (Fig. 2.21). We will consider having a dunk math calculator (Fig. 2.22) as an enhancement and extra feature however this doesn't seem very useful for our project as the fitness application is not sport specific. Whilst the app will be usable by basketball players, having basketball specific functionality may limit user growth and alienate average enthusiasts.

2.2.4 My Jump 2

Finally we have looked at "MyJump 2" - a cross-platform sports science validated calculator app. It's been proven to be "valid, reliable and useful" [19] and provides the most clarity and insight.

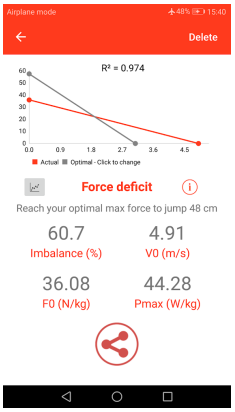


Figure 2.23: Analysis of a jump and data visualisation provided.

The level of precision and insights provided are beyond the scope of our project, as we are focused on providing consistency first and foremost - thereby allowing our users to track their progress. This app estimates and visualises forces produced (Fig. 2.23), which is useful in the field of sport science but may not provide sufficient benefit to users for us to dedicate time to development of a similar feature.

Chapter 3

Project Specification

3.1 Feature specification

The following is a summary of the core features we aim to provide; these will help in fulfilling the aims we set in Chapter 1.

- CRUD (Create, Read, Update, Delete) operations for a user account.
- A vertical jump calculator, operating via video upload (*not recording through the app*).
  - Core: Scrub through the video to set the takeoff/landing point.
  - Extra: Record video on the app (using the phone's camera).
- Create a workout.
  - Core: From existing exercises.
  - Extra: From custom uploaded exercises.
- Subscribe to a preset workout.
- A calendar for monitoring progress.
- A "traffic light system" for monitoring consistency.
- A means of communication with other users.
  - Potentially direct messaging or forum style messaging.
- Provide video playback of exercises
- Allow measurement via metric & imperial units (primarily for US users).
- Provide timer & stopwatch functionality.
- Provide push notification reminders for calendar scheduled workouts.
- Provide notes functions for each workout.
  - Extra: Notes per exercise (allowing specific notes).

3.2 Requirements

Using the above feature set (Section 3.1), we can place features into functional requirements. Other additional requirements will make up our non-functional requirements.

3.2.1 Functional requirements

- CRUD for a User account.
- Upload video from storage.
- Scrub through video and set takeoff/landing.
- Calculate vertical jump in given units (metric or imperial).
- Support video(s) at different frame rate(s).
- Track vertical jump progress.
- Add/remove a workout to or from a user's calendar.
- Mark workouts as complete (with some visual feedback).
- Make notes regarding a specific workout.
- Provide push notification reminders at set duration(s) before a workout.
- Video playback from a source (embedded or from our own data storage).
- Stopwatch & timer must work correctly.
- Simple chat/forum with other users.
- Calendar available on dashboard.
- Create a workout from existing exercises.

3.2.2 Non-functional requirements

- The app should be cross-platform (run on both iOS and Android).
- Have a consistent margin of error (under fixed conditions), when measuring vertical jump.
- Visualise workout consistency on user's dashboard.
- Operate smoothly and provide a close-to-native experience. Users must not feel delayed by the app if/when interviewed following usage.
- Aid users workout experience (aside from the vertical jump calculator).
- Be reliable (no unexpected data loss).

3.3 Technology Stack & Considerations

3.3.1 Frontend technologies

Given that the app will be cross-platform there are a few largely accepted technologies for achieving this with a single codebase. Namely, WebView frameworks (such as Apache Cordova, Ionic, and PhoneGap), "React Native" (created by Facebook) and "Flutter" (created by Google).

We'll be using **Flutter for the frontend** development for multiple reasons. Commonly, developers opt to use React Native due to its large community support - however in 2021, Flutter has a very similar ecosystem in my opinion. Furthermore, whilst React Native is not noticeably slower than Flutter (in many cases) there is a fundamental difference in how these software frameworks operate (and product native platform code from your high-level codebase). React Native uses a JavaScript bridge to communicate between native code modules; on the other hand, Flutter has native component availability by default - in practice the difference in performance is under a second (milliseconds) but it's a consideration that holds weight as complexity of a project grows.

Flutter is written using Dart (a rarely used language that is predominantly only used for Flutter development). This is a stark change from JavaScript (used for React Native and more) which I have had previous experience with. *The project* is an opportunity for me to expand my skill set and learn to work with Dart when using the Flutter framework - something which I have little experience with at present. Additionally, JavaScript will be used for *the product* during deployment.

The final reason worth mentioning is Flutter comes with out-of-the-box testing features as a framework - meaning there is less trouble finding a good workflow. All of this in addition to the plethora of default UI components and APIs[1] means that Flutter will be perfect for any form of iterative approach to software development.

Avoiding writing native code means we can use 1 single codebase and easily satisfies our aims of providing a hybrid mobile application. Avoiding WebView frameworks is primarily due to their functionality being noticeably weak and difficult to scale, given that they are rendering HTML, CSS and JavaScript in what is essentially a browser window. The application would function, but not smoothly as stress accumulates when dealing with increasingly large volumes of workout and application data.

[1] Application programming interface(s)

3.3.2 Backend technologies

For the backend data storage, **we'll be using MongoDB** - the most popular NoSQL database currently in use. In large part, this is being used as an opportunity for me to utilise NoSQL databases as part of a large project as well as learning to design NoSQL databases from scratch. Ordinarily developers are typically assigned to projects that have been set-up by other engineers thus they aren't typically involved in the architectural stages of system design - including the considerations made when creating database schema.

Personal motivation aside, NoSQL databases are hugely beneficial in a variety of ways. Firstly, we have the ability to handle large amounts of data at high speeds. This is achieved using a "scale-out architecture" - this simply means that instead of relying on more computing power (as is the case in traditional "scale-up architecture") we can scale by adding nodes. Related to this is the removal of SQL; not only are we now avoiding potentially complex and inefficient queries but by removing the "SQL magic" developers can focus on elegant application solutions instead of spending precious development hours optimizing SQL queries for the database schema that is in use.

From a data perspective, NoSQL means we can store all kinds of data structures (known formally as "models") instead of the SQL default - tables, rows, columns. MongoDB supports all forms of raw, semi-structured and structured data; this can be as simple as binary data, or creating "documents" that represent highly structured data such as "User accounts" with multiple references.

Overall MongoDB is being used primarily due to it being the most popular NoSQL database and the inherent community support that coincides with this will be greatly beneficial during development. It's important to note that poor implementation of NoSQL databases can mean operating at shockingly slow speeds. Due to the nature of the flexibility and scalability involved with these databases they are also more susceptible to duplicate data - it is much more difficult to remove this redundancy than traditional relational models where we can follow a normalization process. Often traditional normalization makes use of joins, which are not easily executed using MongoDB and can easily lead to inferior code.

To expose the backend data for the apps to consume we'll be using Express.js - a backend web application framework for **Node.js** used in building APIs. Routing is simple and a fully functional basic API can be written in under 100 lines of code. This level of detail is not essential at this stage but noting our use of Express will hopefully help in rationalizing the more high-level decisions outlined above.

Naturally, the technology stack used may include other middle layer technologies, but it's more than likely we focus our efforts on Flutter & MongoDB before making further decisions - for example, we may introduce a cache layer using Redis to optimize queries and make our product more efficient. This iterative behavior is natural in software engineering and so there's an abundance of changes that can occur in our stack which will ease development. This doesn't mean radical changes to our simple Flutter-Express-Node stack but hopefully explains why it's difficult to document any packages and frameworks we intend to use without a thorough system design stage (often taking months).

3.3.3 Security and deployment considerations

The application backend will be hosted on the cloud using AWS (Amazon Web Services). This is easier to set up than self-hosting on dedicated servers and avoids the configuration needed to create a secure system. The costs are predictable based on the usage we see, and we can easily scale without having to invest in additional server hardware. AWS is more configurable than other major cloud providers but provides greater support versus IAC[2] tools like Terraform.

We also plan on using fastlane as the CD component of the CI/CD pipeline for deployment. GitHub Actions can also be used during growth stages before moving to a traditional CI/CD tool like Jenkins - which can also be integrated with many AWS services.

A note must be made that many of the above technology decisions are made with HR in mind. To explain this we must think further into the future; avoiding IAC means we're less likely to need to employ cloud infrastructure engineers (thus reducing costs) and using popular technologies means we won't struggle to find talent if needed. These decisions are largely dependent on what happens with the product and whether the concept is proven as commercially viable, but that doesn't dismiss the need to account for factors such as employee churn as early as possible.

[2]Infrastructure-as-code

Chapter 4

Preliminary Work

4.1 Software architecture

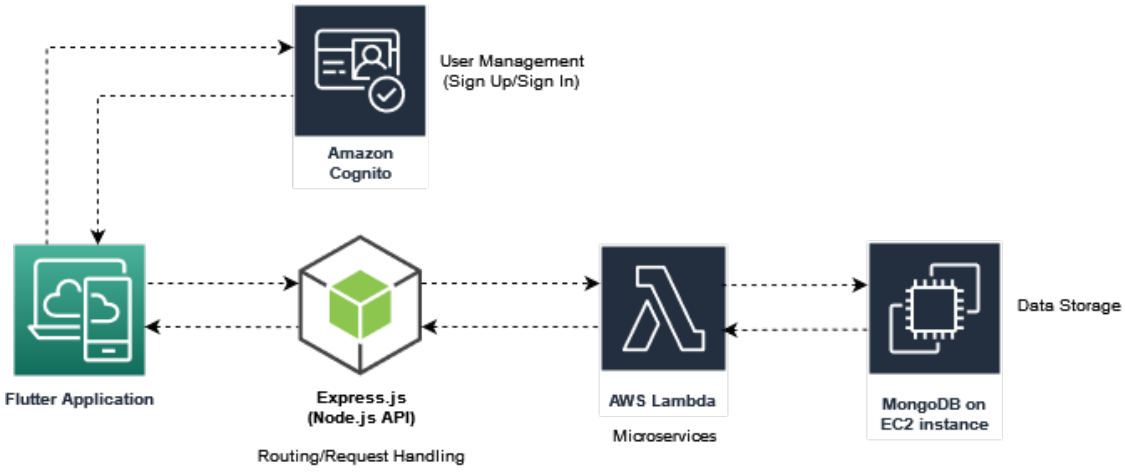
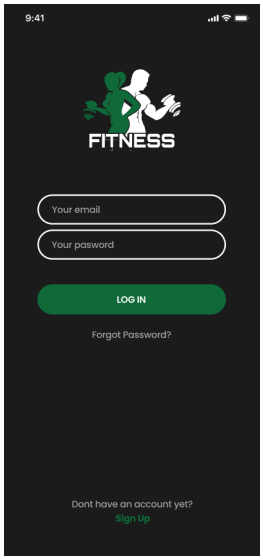


Figure 4.1: A high-level system architecture outlining basic interactions.

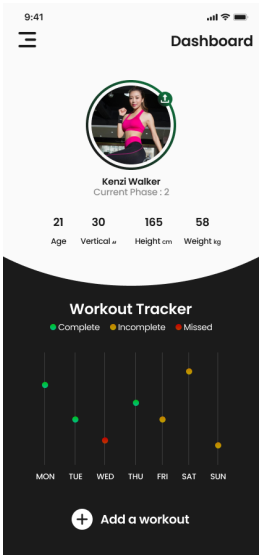
Using an Express API inside AWS Lambda is beneficial because AWS Lambda is an event-driven serverless platform. Instead of paying for cloud servers, you pay for compute power that is proportional to the complexity of events taking place. For example, we can process data on command instead of constantly having processes taking place. This implementation can cause "cold starts" where there is some time wasted if the system hasn't been accessed in a while, but the time spent is negligible for our project and the cost-benefit analysis would show that this is a rationale decision. Using Amazon Cognito also removes much of the development time needed to create simple user management. This doesn't require much skill and so making use of Amazon Cognito is saving us time to use where knowledge and focus is needed.

4.2 User interface design

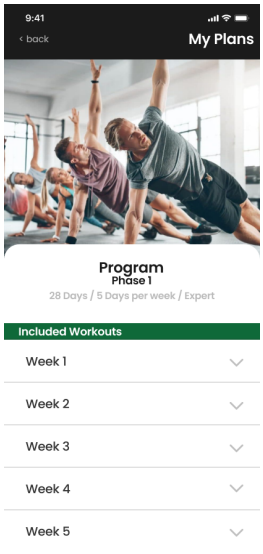
We have created some preliminary designs (Fig. 4.2) in the form of a functional prototype. Main screens have been created and the overall aesthetic is displayed. It's highly likely that the colour palette is changed to meet the brand we decide to create for the application. The functional prototype can be found here: Project Prototype Link.



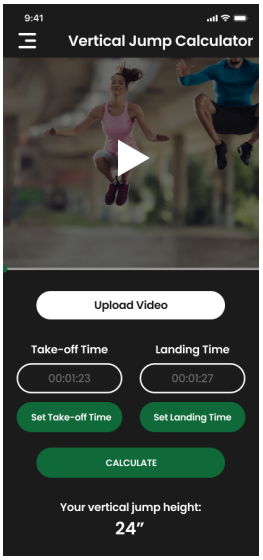
(a) Login screen



(b) Main dashboard



(c) Workout screen



(d) Jump calculator screen

Figure 4.2: A selection of prototype screens.

Chapter 5

Project Plan

5.1 Deliverables

Final UI design & brand guidelines

Due date: 13/12/21
Description: The finalized branding and user interface designs for all major screens of the application. These will be used during development for reference and to create reusable components ("widgets" in Flutter) for consistency across the application. The branding is also important for further marketing of *the product* in future.
Success criterion: Easy-to-follow guidance for developers. No further questions should be asked from developers as the guidance should be comprehensive enough for complete development.

Reusable UI Components ("Widgets") - and test cases

Due date: 24/12/21
Description: Custom Flutter widgets that complete the design system put forward in the guidelines. Any developer will use these widgets to produce consistent frontend code. Custom test cases must be written to verify these function as normal (buttons etc.).
Success criterion: No errors should be found when using custom widgets in place of default widgets during early stage development. All Flutter widget tests should pass.

Minimum Viable Product

Due date: 31/01/22
Description: A fully functional and ready-to-test application that features all the core features. The bare minimum product that could satisfy the aims we set in Section 1.2. The overall efficiency and elegance of the UI is not as crucial as the functionality at this stage.
Success criterion: Any stakeholder or external user could comfortably agree that the aims have been met when using the app. It should be demonstrable.

Complete Test Suite(s)

Due date: 14/02/22
Description: A comprehensive collection of tests for the Flutter application (and ideally for the Express API). This should be composed of existing widget tests, unit tests for specific functionality and integration tests for when the system is ready to be thoroughly tested. The "system under test" must be clearly defined where applicable.
Success criterion: Test coverage must cover all requirements set in Section 3.2 as well as architectural data paths visualised in Fig. 4.1 (thorough structural coverage). Tests must satisfy that the features desired (Section 3.1) are tested and function as expected.

MVP Test Results

Due date: 21/02/22
Description: Results and actions to take following the application of the test suite to the MVP.
Success criterion: All tests have actionable results - pass/fail must be accompanied by a relevant message describing the outcome(s) and next steps (if any).

Project completion & presentation created

Due date: 15/03/22
Description: All relevant testing changes made. A presentation document should accompany the final product ready for demonstration to stakeholders during the project fair. Usability and efficiency are not of utmost importance but should be prioritized.
Success criterion: A functional application that satisfies all aims and passes all test cases.

Dissertation

Due date: 03/05/22
Description: A documentation of the progress and development of the project through to completion. This forms a large portion of the basis for grading of the project and the deadline is final.
Success criterion: A thoroughly well written document that describes the development process as well as lessons learnt from the project.

5.2 Project schedule

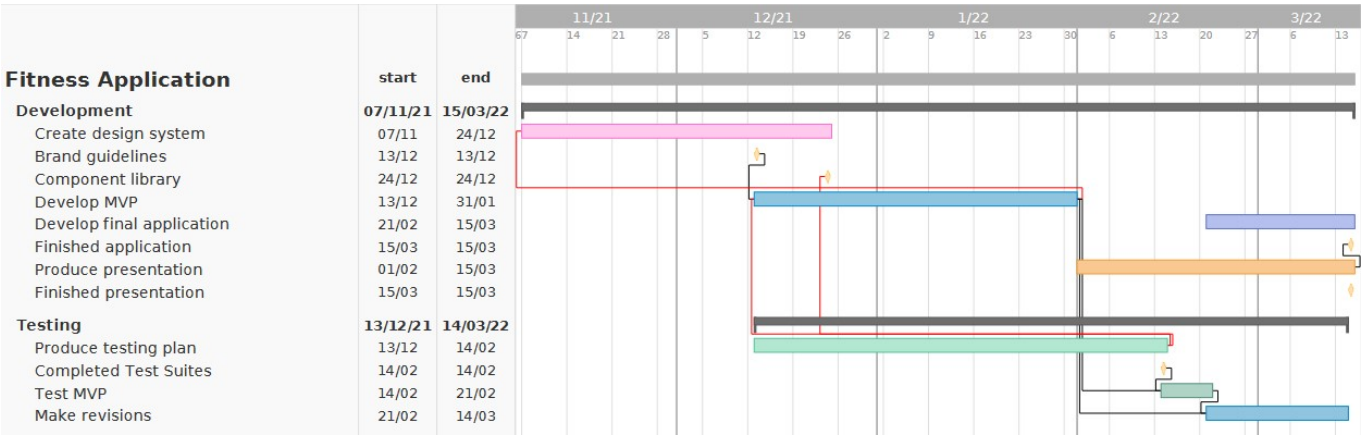


Figure 5.1: An overview Gantt chart, clearly outlining milestones and phases.

The project has begun already as seen in Chapter 4, however we are using 07/11/21 as an acting "start date". We expect the project to be completed by 15/03/22. There is some float as seen in the above Gantt chart (Fig. 5.1); mainly focused on the end of the MVP development phase and testing. Having some leverage here means we can adjust our schedule dependent on the volume of changes that need to be implemented.

5.2.1 Limitations

Our main limitations are time and scope. As established in Chapter 1, the project is different and has a more narrow scope than *the product*. Spending too much time refining code and not focusing on implementation of complex features will mean the grade achieved for the project is low(er) due to us not meeting the aims we have outlined in this document. Time is a factor in this problem as it must be used wisely and we must aim to stick to the project schedule in order to complete the core functionality (and by extension, the project) on time. These risks are assessed later (Section 5.4).

5.3 Software development strategy

We intend on following a "Feature-driven development" (FDD) model - an iterative and incremental software development process. The rationale here is that the bulk of our success criterion rely on the features (functionality) of our application. Focusing on establishing features first will mean we are more likely to fulfill our aims - albeit at the expense of thorough testing during development.

5.3.1 Choice of model

As mentioned prior, FDD is an iterative process; it comfortably falls under the "Agile" methodology umbrella. Agile models like FDD are more adaptable and mean that decisions can be made following short sprints of code without drastic further impact on our project schedule and lifecycle. Whilst it's true that we run the risk of making brash decisions, this can be mitigated by having regular reviews (stand-ups) and in our case, sanity checks and meetings with the project supervisor(s) and other stakeholders.

The decision to follow an Agile approach specifically is due to the tight deadlines and unclear final product. By reviewing development frequently and focusing on developing features of the application, we are less likely to encounter scope creep and more likely to satisfy the success criteria we have established. For the sake of argument, let's suppose we followed a traditional SDLC model (think Waterfall, Spiral, V-Shape). We'd have to spend significant time outlining clear phases of our lifecycle and focusing on said phases in sequence (with most models). This would mean less time is left for development before March and given the unfamiliarity with the proposed technology stack this would dramatically increase the risk of not completing the application (or worse the MVP). Adapting to roadblocks would also require pauses in development and alteration of our predetermined phases (such as testing). This (again) increases our risk of failure.

Ultimately, whilst traditional SDLC models are sufficient for large-scale well-documented projects with comprehensive feasibility reports and cost-benefit analyses - it is not ideal when we are completing a project single-handedly under strong time constraints. Deadlines are thoroughly fixed and without sacrificing performance in the degree it would be difficult to follow said models well.

5.3.2 Testing lifecycle

We'll be using aspects of "Behaviour-driven Development" (BDD) which is an extension of "Test-driven Development". Our use of BDD is only relevant to the testing approach we'll be following. BDD testing uses human-readable descriptions of user requirements (similar to those in Section 3.2) as the basis for software tests. The process involves defining entities, events and outputs with given names before using the names to encode system tests. Each test is based on a user story written using the aforementioned vocabulary. For example:

```
Story: User can create a new exercise

As an end user,
In order to add new exercises
I add video sources to a text description.

On the create workout page,
I press create new exercise,
describe the exercise and add a video source
before selecting "ok".
Then my new exercise should appear
as an option when creating a workout.
```

These tests are then translated into a programming language (in our case Dart code for Flutter) to produce unit tests. We'll be following a black-box testing method, meaning we aren't concerned with the internal workings behind the tests; we are only testing the functionality and not the implementation. This decouples our tests from our code and means there is less to refactor if/when our code is revised.

For this project it's not likely we add much automation to our testing beyond where it is natural such as widgets and other simple unit tests. Because deployment is outside of the scope of the project, there is no significant reason to form a strong CI/CD pipeline.

Our test results will be documented in a test summary report. This works similarly to this project proposal report and means that any stakeholder can read a brief report on how and what results were found following our test phase. Following revisions, we will repeat the testing and add tests if needed for future enhancements.

5.4 Risk management plan

We'll be using a risk register to document and describe risks below - this will form the single source where risks can be documented and added/altered (based on changes in resources).

Risk description: A description of the risk we're encountering.
Probability of occurrence: How likely the risk is to occur (using percentage).
Severity: The intensity of the risk on a 1-4 scale - low, medium, high, extremely high.
Status: View of the risk - potential, monitoring, occurring, or eliminated.
Loss size (days): Measuring the negative impact the occurrence of the risk would have.

5.4.1 Risk analysis

Risk description: File changes lost.
Probability of occurrence: 5%
Severity: 1
Status: potential
Loss size (days): 1

Mitigation strategy: Section 5.4.2.1

Risk description: Behind on project schedule.
Probability of occurrence: 50%
Severity: 2
Status: monitoring
Loss size (days): 7

Mitigation strategy: Section 5.4.2.2

Risk description: Scope creep.
Probability of occurrence: 10%
Severity: 2
Status: monitoring
Loss size (days): 3

Mitigation strategy: Section 5.4.2.3

Risk description: Contraction of COVID-19 or other illness.
Probability of occurrence: 5%
Severity: 3
Status: potential
Loss size (days): 10
Mitigation strategy: Section 5.4.2.4

Risk description: Travel delays without access to codebase.
Probability of occurrence: 30%
Severity: 4
Status: potential
Loss size (days): 3
Mitigation strategy: Section 5.4.2.5

Risk description: Technical knowledge roadblocks
Probability of occurrence: 90%
Severity: 3
Status: monitoring
Loss size (days): 14
Mitigation strategy: Section 5.4.2.6

5.4.2 Risk mitigation

5.4.2.1 File changes lost:

This risk will be mitigated if encountered by using git version control and GitHub remote repositories. The developer(s) will commit code periodically during development of features and at minimum 1x per day (at COB [1]).

5.4.2.2 Behind on project schedule:

The risk here is low given the float we have in our project schedule. If this occurs then the bottleneck will be identified before considering which other processes can continue in tandem with finding a solution. This risk is prevalent during all stages of the project and avoidance is better than mitigation. Mitigation strategies are largely dependent on identifying the bottleneck delaying the project. In general this will involve trading personal self-study for crisis resolution - the worst case is 1 complete week is lost getting back on schedule.

[1] Close of business = 5pm

5.4.2.3 Scope creep:

To avoid scope creep in the first instance, the project requirements will be tracked on a Kanban-style board during feature-driven development. To mitigate any scope creep I will schedule regular sanity checks with my supervisor and make use of "rubber duck debugging" to consciously hear what is being worked on. Code reviews will take place via GitHub pull requests and the labelling of features as "enhancements" will be used to mitigate the creeping in of new functionality and aims.

5.4.2.4 Contraction of COVID-19 or other illness:

I will actively avoid large gatherings and wear a mask where possible. I have received my 2nd vaccine and will stay updated on the status of a booster vaccine if it provides sufficient efficacy for avoiding covid-19. To mitigate the risk once contracted I will rest and self isolate until recovered. This risk is one of the most difficult to mitigate as it involves personal health - without which I cannot actively work on the project.

5.4.2.5 Travel delays without access to codebase:

Being delayed during flights and trains without stable internet access is more than likely given the global situation. To mitigate this risk I will use GitHub and carry a "MiFi" 4G router when travelling. This means access to the code should be possible with just my GitHub credentials. In extreme circumstances, I will clone the repository and work locally on a temporary machine before reaching home to comfortably push changes back. I don't expect delays surpassing 3 days and there has been sufficient time allocated in the project schedule to limit the effects of any delays.

5.4.2.6 Technical knowledge roadblocks:

This risk is inevitable given the project scope and development experience. To mitigate this risk I will make extensive use of the Flutter forum(s) including StackOverflow. This risk could mean the failure of the project - in which case mitigation would involve drastically reframing the projects aims to be more achievable and avoid the roadblock being encountered.

Chapter 6

Summary

We have looked at the project in adequate detail for stakeholders to thoroughly understand the system being built. We've made a clear distinction between future plans for *the product* and the current plans for *the project* (Section 1.1).

To more broadly summarise, we'll be using Flutter and Express.js to implement the project described in Chapter 1. We'll be using a Kanban-style project board with a feature-driven development model to actively implement the features set out in Section 3.1. The core focus will be on the jump calculator as it's the unique aspect of the application being developed and involves substantial difficulty when being created using frameworks for hybrid apps (as seen by the lack of these frameworks and lack of cross-platform calculators) (Chapter 2). If time allows, we'll be implementing an "administrator" role for the application as well as other extra features.

This document has described at a high-level the entirety of the project plan, however it's worth mentioning that a thorough cost-benefit analysis and feasibility report as well as database and system design (in detail) would add tremendous value to the development process. Whilst these are not explicitly required from this document, they would be near essential in a corporate environment and would form the basis for the project proposal being accepted/rejected.

Given enough time (months, not weeks) to write a full project plan, we could collate a fully comprehensive document detailing the technology choices, strategies and deployment methods for developing a fitness application for training (with a vertical jump calculator).

Bibliography

[1] S. Finley, "Woman, 24, who turned to exercise to relax while training as a lawyer and working as a waitress has made over £1m in eight months since launching her own fitness app," *Daily Mail Online*, 2018. [Online]. Available: https://www.dailymail.co.uk/femail/article-7328129/Lawyer-24-turned-Instagram-fitness-star-turnover.html

[2] Steven Bartlett, "Ep57: How she built her confidence, and then an empire with krissy cela," Nov. 2020. [Online]. Available: https://www.youtube.com/watch?v=j1i4WkJ4qFo

[3] Grand View Research. (2021) Fitness app market size, share & growth report, 2021-2028. [Online]. Available: https://www.grandviewresearch.com/industry-analysis/fitness-app-market

[4] Statista Research Department. (2018) Most popular health and fitness apps in the united states as of may 2018, by monthly active users. [Online]. Available: https://www.statista.com/statistics/650748/health-fitness-app-usage-usa/

[5] Crunchbase. Truecoach summary. [Online]. Available: https://www.crunchbase.com/organization/fitbot

[6] LATKA SaaS Database. How truecoach hit $1.3m in revenue with 20k customers in 2021. [Online]. Available: https://getlatka.com/companies/truecoach

[7] Stackshare. Myfitnesspal technology stack. [Online]. Available: https://stackshare.io/myfitnesspal/myfitnesspal

[8] L. S. Vailshery. (2021, Aug.) Market share of wearables unit shipments worldwide from 1st quarter 2014 to 2nd quarter 2021, by vendor. [Online]. Available:

https://www.statista.com/statistics/435944/
quarterly-wearables-shipments-worldwide-market-share-by-vendor/

[9] Stackshare. Fitbit technology stack. [Online]. Available:
https://stackshare.io/fitbit/fitbit

[10] MyFitnessPal Inc. Myfitnesspal api. [Online]. Available:
https://myfitnesspalapi.com/

[11] Decompiler.com. Apk online decompiler. [Online]. Available:
https://www.decompiler.com/

[12] K. Davis, S. Rossi, J. Langdon, and J. McMillan, "The relationship between jumping and sprinting performance in collegiate ultimate athletes," *Journal of Coaching Education*, vol. 5, no. 2, pp. 24–37, 2012.

[13] G. Tidow, "Aspects of strength training in athletics," *New Studies in Athletics*, vol. 1, no. 93-110, p. 504, 1990.

[14] W. Young, "Laboratory strength assessment of athletes," *New Studies in Athletics*, vol. 10, pp. 89–96, 1995.

[15] G. L. Moir, "Three different methods of calculating vertical jump height from force platform data in men and women," *Measurement in Physical Education and Exercise Science*, vol. 12, no. 4, pp. 207–218, 2008. [Online]. Available:
https://doi.org/10.1080/10913670802349766

[16] A. Rauh. (2021) The physics of the vertical jump. [Online]. Available:
https://www.thehoopsgeek.com/the-physics-of-the-vertical-jump

[17] S. Park, I. H. Suh, and W. K. Chung, "Dynamic motion phase segmentation using semg during countermovement jump based on hidden semi-markov model," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1461–1467.

[18] Appmaker.se. (2021) Fitnessmeter. [Online]. Available:
https://appmaker.se/home/fitnessmeter/

[19] Š. Bogataj, M. Pajek, V. Hadžić, S. Andrašić, J. Padulo, and N. Trajković, "Validity, reliability, and usefulness of my jump 2 app for measuring vertical jump in primary school children," *International Journal of Environmental Research and Public Health*, vol. 17, no. 10, p. 3708, 2020.