# KEEP GOING

HOW TO CREATE AWARD-WINNING APPS IN 12 MONTHS

BARDI GOLRIZ

// Keep Going.PDF

**func draw() {**

**)**

# var producer, writer, developer = “Bardi Golriz”

```
var keepGoing = false
for(var age=33;;age++) {
   keepGoing = true
}
```

var bio = “Bardi Golriz had a jobby job. He only lives once, so he quit. He became an independent app developer. His apps have been awarded by Google Design and the Google Play Store. They’ve been downloaded more than 500,000 times (but he still doesn’t understand ASO). They have made more than $5,000 in a day, and $25,000 in a month (he now has an accountant). He lives in London, England with his wife Lana and sons, Kai and Nova. Lana describes him as “unreasonably reasonable”. His hobbies don’t include Paw Patrol. On weekends, he accumulates browser tabs and plays Inbox Zero. Writing this has made him realise he needs to find an actual hobby.”

var contact = “Bardi can be found on the web https://twitter.com/mtrostyle. He also writes a weekly newsletter at https://thirdculture.substack.com/ in the book’s spirit — a director’s commentary on indie app development. If you want to reach him directly, email bardi.golriz@gmail.com — he’d rather hear from you than make Inbox Zero.

var support = “You can download his apps Appy Weather and ruff from https://www.thirdculture.app. And/or check out his Patreon at https://www.patreon.com/barrrdi.”

# var executiveProducers =

# “Lana & Kai”

```
if bardi[“life”].contains(executiveProducers) {
   var ❤️ = 💯
}
executiveProducers.insert(“Nova”)
```

# const dedicatedTo =

# “Behrouz Golriz”

```
var bardi[“opportunity”] = null
while(dad) {
   bardi[“opportunity”] += 1
}
var remember = true
return
```

```
console.log(
```

”This story is based on actual events. In no cases were downloads, feelings and timelines changed for dramatic purposes. All apps, awards and developers are entirely non-fictitious.“

```
)
```

# return timeline = {



}

# print("Hello World")

It was mid-2018. Developing apps on the side, after several years, no longer quenched my creative thirst. This wasn't the first time I was overridden with this feeling. But there was a noteworthy variable edit since the last check-in. To be precise, there was a new constant: I had a two-year-old son now. Parenting changes the lens you see the world through. This has been said many times before, and yeah, they were right. But what they don't say is how it changes your perspective of time. I never took its finiteness seriously. Our personal circumstances remained stable, but, like every prior occasion, it wasn't the optimal time to take an unnecessary risk. I was in a comfortable job, and there were bills to pay. But I wasn't doing what I loved. I wasn't seeing who I loved.

Fortunately, my wife, over a plate of buffalo wings, understood inaction wasn't the necessary reaction. Not this time. Her support was my ticket. Fast forward six months, the first of January 2019 was my first day as a full-time independent developer. I had a 12-month runway to work with – succeed and continue, otherwise LinkedIn. That said, I didn't define what success at the end of the year actually is. Time will tell. Hopefully. And it did. More than once.

## // The First App

Appy Weather originally launched on Windows Phone in 2014. I described it as the most personal weather app – an alternative that didn't simply present the weather but conveniently summarised it for you in plain-speak. So, no, *not* just another weather app. In retrospect, it was launched on the wrong platform at the wrong time. However, for a platform on its third and final act, it did really well. So much so, I was convinced an Appy Weather comeback tour in 2019 to a (much) bigger and, more importantly, permanently settled audience would be popular. I initially targeted Android as it was my daily driver, betting its familiarity would accelerate development.

That was my next six months. It may have taken even less time were it not for my inability to resist a side-project.

## // The Other App

I've never enjoyed writing on a phone, whether it's an email, tweet etc. I wanted to build a barebones experience that's 1) designed for writing on-the-go and 2) makes it dead-simple to move your input elsewhere for the final push. Introducing ruff, your pocket-sized best friend. Work on ruff started in late February, launching only a few weeks after Appy Weather in late June.

# # #

That's two apps built in six months, one more than I had initially planned for my first (*last?)* year as an indie. I thought by not placing all my bets on a single app, I've mathematically *(theoretically?)* doubled my possibility of success, whatever the definition ends being. The problem was I had no baseline reference point, whether it related to downloads, sales or any other relevant metric, to measure both apps' performance on a popular platform (*sorry Windows Phone*). My inability to reliably estimate the amount an app developer can actually make on the store in 2019, when you're effectively in competition with millions, was discouraging. Admittedly, I didn't need pointers to determine whether this is going to be sustainable based on my personal situation's needs. I was optimistic that it would be at least enough to keep going, but the uncertainty was a noted concern. This knowledge would have help set, as well as continually refine, expectations & goals. Besides shaping the roadmap. Frustratingly, I was proceeding in the dark.

What follows is a chronological and unashamedly honest insight on how both apps performed in their first 12+ months of operation, after 350k downloads and multiple accolades between them. These wins didn't happen overnight. Patterns emerged that I became better (*quicker*) at identifying and optimising for. But the lows are equally

revealing. Many mistakes were made and plenty of lessons learnt. To be fair, I occasionally made the right calls too. Usually when it mattered, fortunately (*spoilers: trust your gut*). Hopefully you can benefit from the transparency I was originally after.

Last, and most importantly, there's a neglected *human* side to indie app development that deserves to be told. Getting to where I am today has been the most stress-coated fun. Doubt inconsiderately frequents the narrative alongside a strong supporting cast of emotions, many unwelcome. (*Disclaimer: This book is about feelings as much as numbers.*) The last six months have been an introspective rollercoaster. The writing of the book *was* for me. But the book itself is not. It's for Developers. Creators. Dreamers. This is their story as much as mine.

Keep reading.

var month = **1** // June 2019

var title =
**“In the Beginning”**

var revenue = **£388**

# appyWeather(month: 1)

Appy Weather launched as a free app on June 4, 2019 with an in-app subscription to Appy Weather+ for $3.99/year. The subscription enabled these benefits:

1. no ads
2. unlimited calls per day (as opposed to a maximum five)
3. widgets
4. notifications

You could also leave a tip of $0.99, $4.99 or $9.99. It shipped in English only.

```
/*
     9 updates
     4.5 average rating

     8,740 store listing visitors (1,628 organic)
     2,280 first-time installers (26.1%)
     67 buyers from new installs (2.9%)

     3,099 downloads
     106 subscriptions
        1. United States (49%)
        2. United Kingdom (9.7%)
        3. Canada (8.8%)
        4. Germany (7.6%)
        5. France (5.4%)
     12 cancellations

     £369 in subscriptions
     £8 in tips
*/
```

# ruff(month: 1)

ruff launched as a $0.99 paid-up-front app on June 28, 2019. It was available in English only.

```
/*
     1 update
     5.0 average rating

     42 downloads
     17 paid users
     4 refunds

     £11 in purchases
*/

// Note: when the Google Play Console for developers was upgraded
// in 2021, store analytics from before March 2020 became
// unavailable. Unfortunately, I didn't record them for ruff
// before the upgrade (because they never said data will be lost).
```

# getLessons(month: 1)

// Laziness and efficiency are the same thing

I tend to think of an app being composed of three layers: top, middle & bottom. The private testing period, combined with my own usage of the apps, caught most issues in the first two layers: at the surface level as well as any that were less visible but required minimal poking. They were given the time (*opportunity*) to be naturally encountered. This was especially beneficial for Appy Weather, as it allowed for different weather conditions to be reported/tested. The bottom layer however was mostly neglected — there was an insufficient breadth and depth of testers for it to be effectively targeted.

Keeping testing restricted shortened my reach. This is especially harmful on Android, where it's practically impossible to cover the potential device spectrum. (*I discovered BrowserStack several months later – it gives you remote access to the most popular Android phones. At a reasonable price too, especially if you compress testing so you can sign up to the cheaper monthly plan once.*) It would have been sensible to momentarily scale testing before the apps' beta status was dropped. By inviting testers in meaningful numbers from outside my inner trusted circle. Admittedly, this is a more necessary call-to-action when interfaces are complex and/or not responsively engineered. Although I wasn't in reckless pursuit of pixel perfection, I could have been more considerate with my designs.

The entirety of the experience needed to reflect my taste. It was fun and empowering to independently set the appearance and behaviour of every interface and interaction. But I didn't realise this unintentionally neglected the dependencies and/or setups of many users. This is why taking accessibility seriously is a net-win. Promoting it to be a design pillar facilitates interfaces that are less fragile in a testing environment because they were designed with an appreciation and understanding that users are *humans*. They're unique in different ways. But they did have one thing in common: they were *Android* users.

Accessibility makes you feel at home on the platform. I made an effort for the apps to reflect Material Design principles. But it was mostly superficial. I prioritised the apps

over the platform. They matched my taste, but it immediately introduced inefficiencies. For example, I didn't see the benefits of the ongoing improvements to the Android toolkit. Unnecessary reinvention meant changes that should have been free cost me valuable time to support and maintain. Responsiveness could have been baked in. But was a plumbing job instead. No pain is actually all gain. Take the initiative at a foundational level. Be the force of change. Not vulnerable to change. You can still make it feel like your own. I'm not building for my ego. Not anymore anyway.

# # #

## // Openness generates buzz

I'm generally a private person and, in the context of an app release, hype averse. This characteristic shaped the marketing strategy. That is, there was *zero* marketing carried out in the build-up to either app's launch. They were both quietly dropped onto the store. Maybe it's just me being nostalgic, but there's a refreshing charm when a product unexpectedly launches. No release date. No waiting list. Get it *now.* I like the work to speak for itself when it's ready. The benefit with this mindset is that I don't feel any external pressure to deliver as I've not set any expectations. Also, being developer-first, I subscribe to the notion that building the best possible product, and nothing else, will shape their reception. Marketing was considered almost counterproductive. I'm not good at it anyway. This was a mistake.

Not marketing in parallel to development, I wasn't newsworthy on the day of release. Releases are waves, and there's none theoretically bigger than the initial. But if nobody is aware, then it effectively never happened. (*On a practical level, an app's biggest recorded wave could possibly be when it 2.0's since it's now semi-established but still relatively new to the scene – I should be able to confirm this later in 2021.*) I was deep in my developer bunker. When I finally hit publish, it was a boring finish. No stress, but no fun either. This

should have been a memorable day. This is the result of development monopolising a developer's time. I didn't appreciate the app's reach is calculated as much by the messaging as the product. And I never budgeted for a messenger to spread the word. But there was another reason I was reluctant to commit to one.

## // Features are commodities

Not being established, I knew words on their own won't effectively generate signal. I had to embrace openness. Share as much as possible. But I was concerned it would position me in a competitive disadvantage. By giving away my ideas for others to copy before I actually shipped. This was naïve. And maybe even arrogant. The truth is it's unlikely others in the space will be paying any attention. Or placing any importance, if they did – their priorities are not set by a no-name developer of an unreleased app. But if they did, and this actually becomes a problem, then it's not necessarily a bad thing. This indicates you've run a stellar campaign. Launch day *will* be memorable.

Regardless, it's actually liberating when you accept features can easily be copied (*and bettered*) at any time. Experiences however can't. Nothing beats the original. Think of yourself. The best version of you is *you*. If you however try to be someone else, you're at best a runner-up. I wasn't confident then. Competition was a scary concept. Today the only competition I'm in is with myself. I'm not worried about losing users to other apps. I expect users to switch. Heck, I encourage it when we're not a match. I know better than to be peer pressured to stray from my vision. I'm not going to change for them, and I don't want them to have to adapt either. People have different tastes and preferences. The good news is there are a lot of people. And not many need to get what you're doing so you can keep on doing it. Build for a specific person and you'll have an audience. No matter the competition, you become an alternative.

## // I went to the press too early

There's another perspective that promotes caution. When I finally began rudimentary marketing, although the philosophy powering each apps' identity was clear, and

picked up by the press, there were too many functional shortcomings in the versions that were featured. And so, the positive press didn't really translate to optimal conversions, both in terms of downloads as well sales. After several months of development, marketing negligence aside, I wanted as many people as possible to get the app as soon as possible. But considering the lengthy development cycle, it's not as if speed was ever the priority. Completion was an incorrect trigger to escalate its importance. I didn't appreciate immediacy mattered to me, but timing to everyone else. Next time, I know better. I'll play the long(er) game by carefully considering the optimal moment for press coverage. I'll probably proceed with an official launch after an extended beta run. When testing inevitably confirms that the app is ready for the press to check out and share with their audiences.

## // Getting press isn't difficult

It's surprisingly easy to get press, irrespective of timing. I initially targeted Android fan sites. They need new things to write about, so it's a win-win. Minimal effort identified the appropriate person to contact. For example, with Appy Weather, I searched for recent posts on other weather apps, and reached out to the writer who most often or recently covered them. Alternatively, I'd contact whoever regularly highlights the best new apps (*a regular feature on these sites*). The best results were when I used email. However, when their address wasn't shared, I looked for them on Twitter. As a last resort, I'd leave a tip via their website's contact form and hope the message got through to someone (*it usually did*). With relationships established, when I had anything new to share, such as ruff's release later in the month, I was more likely to get through. Today, I'm a proud owner of a contacts directory. The next app can boast a coordinated marketing campaign to maximum its reach at launch.

## // Doing a 180 is disproportionate

Thinking ahead, I can't see myself being totally transparent during development. I'd still like to keep surprises for launch day. But I don't intend to keep progress private

either. There are creative ways to effectively tease, that don't involve desperate giveaways. Feature reveals are not essential. The process is more interesting. That – more than a screenshot – should hopefully generate signal. For example, one of the many reasons I'm writing this book to drum up interest in my future projects, i.e. content marketing. I could have written a Medium post or tweetstormed instead, but the seriousness of the effort is deliberate. A book represents non-ephemeral value. It takes several more months than an evening to compile (*if only I knew!*), but in my experience an input's returns are usually proportional – hopefully this time won't be an exception. Setting high expectations for the next one. See, I'm trying.

# # #

`// Earliest users = biggest fans`

Users are generally more tolerant and supportive when a product ships with a beta tag. It's okay for the app to be rough around the edges when used in this context. Or for there to be missing features. As long as you make the users feel as though their early investment in your app will pay off, because their feedback is legitimately shaping the roadmap. This can be achieved through a combination of actions and words. Regular updates within the beta period that blitz through low-hanging fruit (*i.e. the easy, or easier, items of feedback to act on*). Acknowledgement/transparency on the bigger, less immediately addressable points. These moves generate goodwill and trust in your efforts, potentially converting testers to fans. I'm lucky I began the testing cycle with a reliable base of supporters inherited from the Windows Phone era. Their ability to regularly contribute and validate – often on-demand – made me stay focused and be encouraged. I've managed to considerably increase the size of this group by being approachable, keeping honest and following through.

## // The first sale relieves a big weight

By the time Appy Weather was finally released, I questioned whether its effort was worthwhile. Prior, there were many moments when I felt really good and confident about its prospects. However, these were unevenly distributed within the first ~50% of its development. Ironically, as the app matured, doubts surfaced. The closer I was to the finish, the further I was from the excited position that kickstarted development. It's not that it wasn't meeting my expectations, but I had underestimated the effort involved to fulfil them. The greater the investment, the bigger the returns needed to justify the expense. I was uncertain that I'll be adequately compensated at the finish.

I made enough sales on the first day to feel a bit better: a total of two with a generous user leaving a $9.99 tip. It was practically nothing, but the symbolic value was disproportionately significant. I actually created something that people would pay for. Correction – I created something that people would *subscribe* to (*i.e. want to depend on*). The next few days saw four sales – these didn't feel as reassuring. However, after the app had its first bout of press (*it was mentioned in a new apps roundup*), things picked up in a (relatively) big way. Three consecutive days with 10+ new subscriptions. There *was* money to be made. My understanding of exactly how much evolved with time. I had no idea going into this. I thought I had a better idea after a week. But it was a year before I could differentiate between the potential of overnight success from the reality of the daily grind.

## // Nothing lasts forever

Shortly after the high of generating $100+ over a few days, there were consecutive days with zero sales. Back to reality. This became a pattern, regardless of the apps' popularity. You'll not be able to maintain the highs. They're temporary, so don't get carried away (*I need reminding to this day*). Although it's only a matter of time before the numbers settle, they'll potentially be reduced as opposed to reset. That is, entering a new normal rather than reverting to the previous. The speed of decline is usually proportional to the length of the peak period. If it's a few days, like above, then expect the numbers to decline to pre-peak numbers after a few days. Whereas if it lasts a few

weeks, for example, then the drop will be gradual over a similar timeframe. And the probability of a new normal (*i.e. more daily downloads/sales*) are better too. When this happens, the spike's trigger was presumably more popular. The analytics will have meaningfully improved. The app is better able to generate signal to the store's algorithms. The more popular the app was before a surge, the better its prospects of improving its store signal in the aftermath. Regardless of popularity, the next post-surge change in reality should be more pronounced than the last. Initial growth with both apps was probably the easiest. Going from zero to something is the product of basic effort. It's more complicated afterwards. It needs time. Keep showing up every day, and you're in the race. But, more than anything, you need breaks. I got mine after three, six and nine months. Each progressively raising the bar.

## // Never get complacent

I quickly realised complacency will be one of my biggest obstacles to succeeding – possibly the biggest. I should have strengthened marketing efforts in the middle of my first sales tick. Not withdraw back to development. The result of doing so was a dry spell. Unexpected yet predictable. A reaction was necessary. I hate *needing* to do something. The pressure of expectation is worrisome. The alternative is always better. Pre-emptive incremental steps. You'll always be a move ahead, never trailing in panic. Thinking of a way back, I reflected on what worked the last (*first*) time. Initial press immediately boosted sales, so I returned for more. Two days with $100+ combined sales immediately followed. Even better than last time!

The comeback made me consider a new day to be a competitive reset. Winning today doesn't ensure success tomorrow. But it elevates you to a position more likely to repeat it. Don't take advantage of the raised platform, and you'll be grounded in no time. That said, realistically, there will be setbacks. They're humbling. Opportunities to reflect and strengthen. The most effective counter to any loss, and to ensure it doesn't become a losing streak, is a determined response. Show up every day, and the overall win percentage is more likely to be above 50%. Relax and the number drops. Over

time, I got better at knowing when to switch focus to keep an indie ship in rocky waters stable.

# # #

## // Default to subscriptions

Every time a user opens Appy Weather, it costs me money. I'm fetching the latest forecast from an external provider. Although an individual request is inexpensive, if/when the app boasts a reasonable user base, it will add up to a tangible amount. Subscriptions enable these recurring expenses to be sustainable. Revenue won't be as high, but it's profits I'm after. Opting for short-term gains by rejecting the sub model would have been reckless and potentially terminal. Inevitably faced with a self-inflicted ultimatum to either switch to subscriptions or close shop. To keep messaging consistent, I decided against an alternative lifetime purchase option, usually available to appease the subscription haters. Although it could be sold at a premium, it didn't register that the app's continued success may in this context actually be counter-beneficial, as it increases the possibility that I'll make an eventual loss on a lifetime user. No thanks.

I sensed going all-in with the sub model may be a deeply unpopular and possibly controversial move. I didn't come across any weather app on Android that was subscription-only. I was up against many that were completely free, and any with a subscription offered a paid option too. There was a genuine possibility that the app would be dead-on-arrival. I briefly re-considered my decision, arguing that my maths had been theoretical, and the calculated worst-case scenario won't happen. Rationalising the improbability of every paid customer remaining an *active* user, and possibility that many will become lapsed users. Because that's just the way it is. I've paid for hundreds of apps over the years, but I actively depend on only a few. But I wanted Appy Weather to be one of the few apps that consistently makes the cut. I

couldn't proceed with a paid model that actively benefited from churn. That was a loser's mentality. So, doubts castaway, I set sail with the sub model.

Any opportunity available was used to communicate the merit of subscriptions. I did my best to convince users that it's actually in their best interest too. Assuming they intend to *actively depend* on the app, it's the only model with an alignment of priorities. I'm incentivised to keep improving their experience, as opposed to focusing on unpaid users in a non-sub model because they – and not already paid users – are technically able to contribute to the app's growth, i.e. pay next month's bills. I will spend more of my time developing and less marketing. Win-win. I was optimistic that any reasonable person will at least appreciate, if not agree, with the position. Repetition is a powerful communication tool. When apps began going free, the new normal was perceived as consumer-friendly but it was anti-developer. Subscriptions are an opportunity to re-calibrate expectations so that everyone actually benefits, because a happy developer is a happy user.

## `// Subscriptions are not essential`

ruff, on the other hand, didn't depend on any external APIs. Its only ongoing expense were my development hours. Subscriptions would incentivise its continued development. But they weren't necessary. Keeping dependencies local has an empowering effect. Time becomes the deciding currency, whereas monetisation determines the size of profits (*not revenue*). I placed multiple bets because I wasn't expecting a meaningful return on each app. Even in a worst-case scenario when ruff's development would be judged on meagre profits, there would have been valuable lessons to takeaway. At the end, there were a few reasons ruff didn't adopt subscriptions:

1. Given ruff intended to support the business – not sustain it – there would be appropriately less pressure to deliver regular updates. When you pay once for an app, you're presumably paying for what it can do today – not might in the future. As long as the purchase continues to work, and paid users don't need to

pay again if/when there are updates, you're mostly okay. That isn't to suggest there isn't any expectation for updates. But because there's technically no direct payments made for them, I'm not going to stress if it's been a while since the last update. I was right not to. ruff's update velocity has been slow, but it's never been a complaint. Even when it's been literally months since the last update. Appy Weather, on the other hand, was the daily driver. I embraced – encouraged even – the pressure to regularly update. A subscription business thrives when you regularly justify to its subscribers its value. It was being updated at least a few times every month, sometimes a week! When users' expectations are aligned with your priorities, work is not a burden.

2. There was a TBC feeling that people are more likely to pay up-front for a productivity app than to subscribe (*i.e. more revenue*). I check the weather regularly on my phone. I'm almost never productive on it (*hence ruff*). That made Appy Weather a more appropriate fit for subscriptions. The opposite would be true on a computer. The weather isn't checked as often, reducing cost therefore prices. In other words, subscriptions make more sense when (*where*) you expect users to actively depend on the app.

3. Thinking of the business, it made sense to experiment with monetisation so that I can benchmark each model's performance. ruff's monetary deficiencies can be offset by its learnings. Multiple bets are good. Different strategies are interesting. Combined, there's plenty to work with.

## `// Tips pay for a pizza`

It was immediately clear a sustainable income on the back of user tips won't be feasible. Not that I was expecting differently. To be fair, asking users to leave a tip before reasonable time to judge an app's development is presumptuous. I would have felt more comfortable asking after frequent meaningful updates. But I asked anyway, hoping users won't be offended. Fortunately, none were. Its screen was out of the way. Maybe that's why. When I did briefly present it more prominently on the default

screen (*when you scrolled all the way to the bottom*), there were a few negative vibes. In response, I reduced the frequency it showed up. There was no need to spoil the experience through pressure tactics. I'm content when the tip jar can cover an occasional steak dinner. It's managed this a few times. Users have more options to tip today. As much as $50. One user has. If you're reading this, thank you.

That said, I've noticed some apps clarify a tip won't leave you empty-handed. That your generosity will be reciprocated, usually with a cosmetic item(s). This move should substantially increase tips, but I expect the total to remain insignificant. Regardless, and maybe I'm just being cynical, but I decided against because I found the practice disingenuous. As far as I'm concerned, that's an in-app-purchase falsely promoted as a tip. A potential workaround is to keep quiet and surprise the tipper. That wouldn't increase tips but is at least a classy gesture that you could still benefit from, e.g. reviews, referrals, renewals etc.

# # #

## // Ads serve a purpose

Minimising Appy Weather's expenses was just as important as maximising its revenue – profits are what keep the lights on. I didn't want its experience to be peppered with ads. Definitely none I've no editorial control over. I also wasn't interested in silently collecting/passing on user data, whether to sell (*I get proposals regularly*) or to personalise what's served up. Red lines established, everything else was negotiable. At launch, Appy Weather delivered a single native ad promoting the premium upgrade, and later in the month rotated to highlight ruff too. You needed to scroll to the default screen's footer to see an ad. I rejected a permanent ad bar because I didn't want to cheapen the vibe. I was paranoid prevalent ads would lead to a negative first impression, so I effectively hid them. I was being considerate but that was a dumb move. I needed reminding that 1) I'm technically making a loss on every free user session and 2) I want

users to want to remove ads. Ads didn't need to be obtrusive, but at least visible so that it factors when a subscription is being evaluated.

## // Give and take

The discreet advertising facilitated a conservative-leaning position with the free version's feature set. Background weather requests through widgets and notifications were subscriber-only capabilities. And when the app was in active use, you were limited to no more than five checks a day. The limit was arbitrary. Based on my own usage habits, I thought that should be enough to get a free user through their day. For comparison, other weather apps were usually offering their free users at least basic background capabilities but would never restrict foreground use. I didn't take this inconsistency seriously. I was just relieved I managed to insert measures to effectively reduce load. A reality check was imminent.

## // Beware of angry users

Less than two weeks after Appy Weather's release, I doubled the quota of free weather requests. I dramatically underestimated how often people check the weather. The move intended to make the app competitive in active usage at least. I thought a 100% increase was a generous concession, but it wasn't perceived enough. It was not the number but that there was *a* number. This dissuaded a vocal segment of users from the app. They wildly speculated on the developer's priorities and intentions. There were better apps available in their opinion. Ones that didn't impose such a "ridiculous" constraint. That did more *and* cost less, sometimes nothing even. Stay the f___ away! That was the gist of their reviews. I didn't see this coming. But I had been on the Internet for more than a few minutes – I should have known better.

## // Not every problem needs a creative solution

People won't campaign against you when there's nothing to get angry about. Convention avoids controversy. Monetisation is a sensitive topic. I mishandled the free version. I misjudged its requirements. This was not a new problem that necessitated creative thinking. It was really simple. I should have included an ad bar. This would either encourage a subscription (*make money*) or discourage usage (*not lose money*). There shouldn't have been any foreground weather restrictions. The negativity it promoted was not worth it. The 1-star reviews were more damaging than the associated cost of unconstrained weather requests. Fear guided my position. This is what happens when you're defensive based on feeling and not data. I should have launched and assessed the actual financial reality. If necessary, make a change. That said, these choices are less obvious (*necessary*) when usage isn't a cost consideration, such as with ruff. You have less reason to deter usage. The presence of ads can still nudge users towards a purchase, but a less assertive position has benefits too. A clean experience increases engagement. An upgrade becomes more likely.

## // Give because you can

I however didn't budge with background capabilities, despite the criticism/pressure. My original positioning of the free version was that it must cover the essentials, and a subscription had to level up the experience with the niceties. These guidelines presented clarity on what went into both products (*and continue to this day*). The paywalled features usually cost me more. This made sense. You pay more for the convenience. I would have been more accommodating if cost wasn't a factor. With increased foreground calls, the hit was insignificant and not necessarily widespread (*some users would have made more than five requests, but unlikely every*). It would have been the opposite if free users could receive background weather updates too.

That said, I did consider making widgets available to free users but limiting their update frequency. Even so, I'd still be taking an undesirable hit, perhaps no longer exponential but still significant. More importantly, I questioned whether an infrequently updated widget has any utility. Users were less likely to upgrade if they

had a negative perception of their reliability, i.e. being subscription-only is possibly a more effective sales tactic. Also, it's a less complicated pitch, and arguably more attractive, to present widgets as an entire feature that's unlocked through a subscription, as opposed to parts unconstrained. This simplifies the messaging/choice: if you want widgets, you need to subscribe. But there was another way to provide limited *access* to the full Appy Weather experience: a x-day trial.

## // Risk it for the biscuit

I never included a trial originally due to an irrational fear of the potentially exponential hit if the conversion rates were inadequate, i.e. significantly more expenses. To be honest, I was equally concerned that it may result in less subscribers too, i.e. noticeably less revenue. The result: terminal prospects (*profits*). But there was a flip scenario where courage would be rewarded. I didn't consider this a possibility at the time though. I wasn't convinced that Appy Weather+ will be *perceived* as reasonable value during a trial. A subscription is a commitment, whereas a trial is momentary. I was worried that Appy Weather's alternative approach needed more than a few days to appreciate. It was more of an acquired taste. Extend the trial to a week, and expenses are potentially doubled. The first priority was to not lose money. This ruled trials out. Fast forward to today, although I still don't offer them, it's not down to a lack in confidence in the product. Appy Weather+ has made big strides with its value instantly recognisable. I'm not risk-averse anymore either. I've just been busy *and* lazy. There's more work involved than flipping a switch, so it's not been a priority. Times have not been desperate to justify a bump. And with a lengthy rewrite in development, the timing is inappropriate to tamper with income streams. But it should be baked in the 2.0.

## // Show and sell

What stung more than the absence of trials was the inability to at least *preview* widgets (as well as notifications). You actually needed a subscription to see what the widgets

looked like (*I later linked to screenshots*). There was a semblance of logic behind the decision: I was worried users won't realise they're not updating because they weren't subscribed. Cue the 1-star reviews. With the right design and/or messaging, this could have been an easy fix. But because I was so focused on the product, I developed a blind spot: I didn't see the need to convince users on an upgrade. I was laidback about it. Arguably arrogant.

Users who appreciated the app's point of view will want to upgrade to access the best version of the app. That was my perspective. And to thank them for their support, I wanted to spend every development minute making the best version be the best possible. To be fair, many did blindly upgrade. But I bet many more would have if I bothered to educate users on an upgrade's merit. And, even if they didn't, it would have been the right thing to do. I was being inconsiderate when asking users to upgrade based on a few bullet items. I should have defaulted to the position that every free user needs to be convinced. I could only gain from this mindset. It won't make a difference to the users who were going to upgrade regardless, but it will have to the many who entered the in-app purchase screen out of curiosity and not intent.

This lack of attention extended to both apps' store listings. They were considered a formality. Within six development months, I spent no more than a day on them. At the finale, when the apps were about to be published. I whizzed through them. Suddenly I was being impatient, prioritising speed. That was completely misguided. ruff's prospects in particular was severely compromised because it was a paid-up-front app. I needed reminding this was potential users' first and possibly most important impression of the apps. And they were unimpressive. Lacklustre. Careless.

This was a job that needed me at the top of my game, and instead I showed up late without my gear. I didn't realise that unlike most product work, getting this right will definitely make a meaningful difference. Quality is the product of effort over time. The apps greatly benefited from both. Except for this one area that wasn't allocated either. I let my standards slip. The best user experience is the most consistent. Every point of interaction, before you're in even. I understand why I didn't do any marketing. That was a misstep. But not selling was a mistake.

# # #

// Google Play is not evil

I don't know whether it's fair for Google to take 30% of every sale. Or necessary even, considering their wealth. Regardless, at this point, I considered the percentage to be insignificant compared to the 70% I began receiving. I appreciated how effortless they made it to start a new business on the store. That I didn't need to worry about payment processing, handling user authentication, integrating analytics etc. I accepted a premium for these conveniences. Especially when the biggest perk is undeniably what I needed the most: the marketing/sales boost as a result of a presence on a Google platform. *(That said, I observed the majority of store traffic for the month was, unsurprisingly, externally sourced – it would be fair if there was a distinction between these and organic installs.)* If I was publishing independently, I may be making more from every sale, but I doubt there would be as many. Maybe these feelings will change when I'm more established and am not benefiting as much from their APIs/reach. But for the time being at least, I was grateful.

# # #

I thought I could start to relax after both apps were released. I needed a break after the last six months. But the opposite happened – I doubled down. Ten updates went out within the first few weeks. And I already had enough planned for the next 100. When you have users, you want to keep them happy. And when you have happy users, you want more happy users. Welcome to independent app development.

// Of June's **£388** revenue, I received **£256** pounds.
// Appy Weather made **$5** in weather requests.

// Note: the amount I receive is reduced because of Google Play's
// commission as well as VAT deductions. The $5 above will be taken
// from the £256 pounds, and the final return may be reduced further
// due to income tax (assuming I meet the minimum threshold)

*Keep Going.*

Buy the full book to experience all the memorable ups, miserable downs, unexpected twists and valuable takeaways: http://gum.co/keepgoing.

If you liked it so far, you won't regret it!