Building Consumer Social Apps

James Heaney, https://jvheaney.com [james@heaney.ca]

Why??

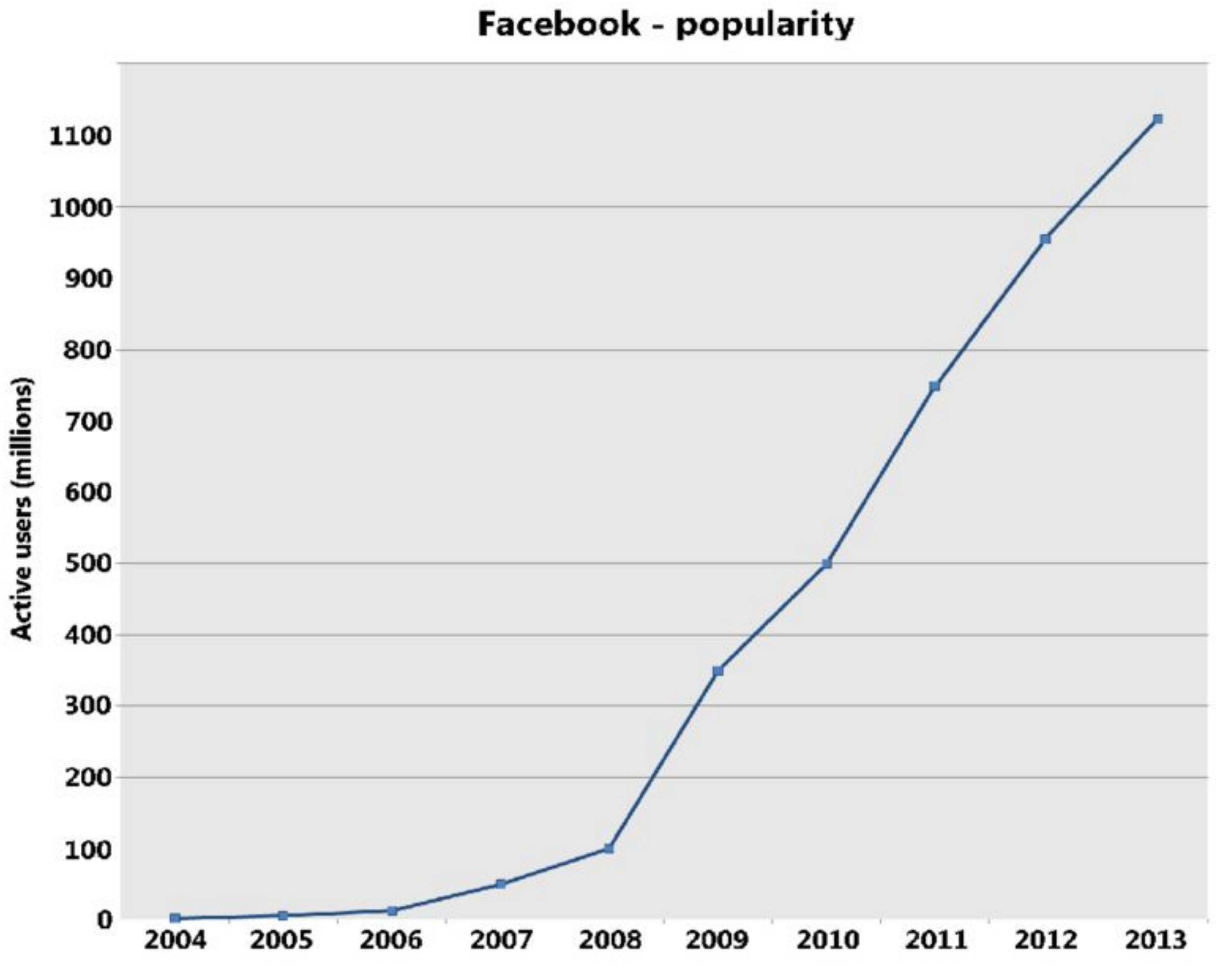
The Social Network

The bug

Let’s talk the facts...

Facebook launched in February of 2004

- Within 24h, **650 users** had signed up
- Within two weeks it grew to about **4300 users**
- By March it had it’s **10,000th user** on the platform after expanding to include Columbia, Stanford, and Yale
- It closed out 2004 with **1M users**
- 2005 closed with **6M users**


Facebook - popularity
Active users (millions)
1100
1000
900
800
700
600
500
400
300
200
100
0
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013

Why??

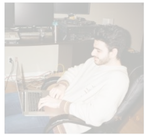
What indicated success?

- **Rapid growth:** over half the undergraduate population of Harvard was on Facebook within it's first month
- **Expanded well:** once other universities got access it spread like wildfire
- **Facebook was a destination:** it was some people's only destination on the internet
- **It stood out:** it emphasized real identities while competitors focused on usernames

Which boils down to...

- High growth rate
- High demand
- High engagement
- Novel features

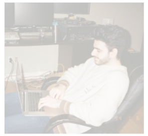
Metrics



jvheaney

User Growth Rate

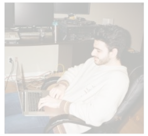
- Wildfire, especially when everyone is glued to their phone and computer
- Stems from the network effect
- Aim to have each new user bring **5 new users** with them



jvheaney

User Demand

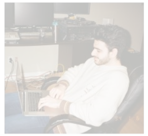
- Should be noticeably high, emails being sent to you, direct messages online, tweets, etc
- Should have **organic** social media popularity, **not paid for**
- People should be talking about you to their friends



jvheaney

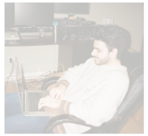
User Engagement

- Should be very high
- Once it slips, you begin to lose
- If your app feels empty and boring *it will be empty and boring*, it’s a self fulfilling prophecy



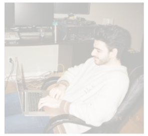
User Retention

- Extremely high, especially at the beginning
- If you’re not keeping users on the app in the first few days, how will you survive a month from now?



Friend Network

- We are creatures of tribalism
- We don’t want to stand out, we want to fit in with the crowd
- Maximize the amount of friends a user has on the platform, incentivize growing their friend network

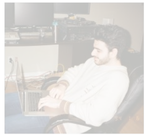


Daily/Monthly Active Users

- Important too
- Function of the other metrics, this just gives you feedback on how you’re handling the others

Understanding the Audience

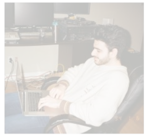
Why do people download apps?



jvheaney

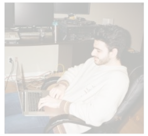
The three golden reasons

- To find a partner or love interest
- To make or save money
- To forget the world exists



Adults are terrible

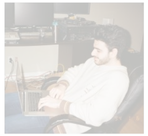
- **Friend groups are distant:** they don’t talk to each other as much, reducing your network effect
- **Creatures of habit:** they have their apps they use consistently, they are not changing for you



jvheaney

Younger audiences

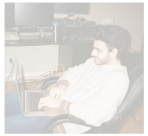
- More willing to try things
- More susceptible to psychological tricks
- More time to spare
- Don’t want to be left out
- They talk to their friends. *All the time.*



jvheaney

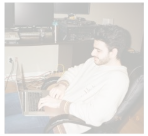
Who is your target audience?

- Who are they?
- What do they want?
- Which of the three categories do they fall into?
- Why do they fall into that category?
- How can you use psychology to appeal to their desires?



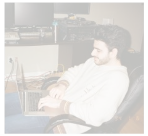
Geography

- What if Facebook launched to the entire US at once?
- Much harder to find and connect with friends (dead on arrival)
- Much harder to stalk your crush (find a love interest)
- Much harder to brag to your Facebook friends how great your life is (also in the pursuit of finding a love interest)



The Facebook formula

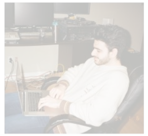
- They launched in tight communities
- Communities where people already talked to each other
- Could recognize profiles as people they’ve seen local to them



jvheaney

Saturating markets

1. Find a small market
2. Saturate it
3. Expand out
4. Keep going

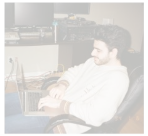


Copy the greats

Physical or collectives. The easier to meaningful saturate, the better.

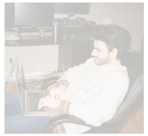
- Facebook launched to universities
- MySpace launched to music communities
- Gas, tbh apps launched to high schools
- PayPal launched to sellers on eBay

Creating the Idea



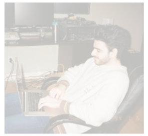
What is the app type?

- Messaging dominant?
- Posting dominant?
- Consumption dominant?
- Another type?



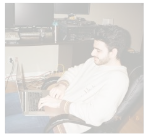
How involved is the user?

- Do they need to produce content for the app to work?
- Are they mostly consuming content? Where’s that content coming from?
- Are they responding to their friends?
- Are they having to do a repeated action consistently?



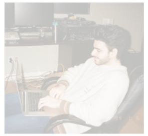
What is the medium?

- Short form video?
- Long form video?
- Pictures?
- Text?
- Polls?
- Pokes?
- Messages?



Where can we use psychology?

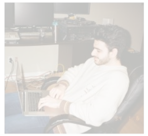
- The mystery of a crush is powerful and plays into finding love interests
- Habits and consistency are powerful triggers
- Can we show someone how much they are liked? How important they are?
- Can we let people live vicariously through others? Develop bonds with people over the internet?
- Can we make people feel like they’re missing out by not using our app?



What is the central action?

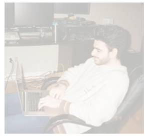
- Define one clear action that drives your entire app
- Make it as simple as possible
- TikTok, swipe
- Tinder, swipe
- Snapchat, take a photo
- Gas, answer polls

The Network Effect



The methods

- **Sync contacts:** who knows who, connect people, invites
- **External platform sharing:** easy to share, desirable to share, not an advertisement
- **Word of mouth:** might work, but less measurable and reliable

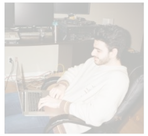


Be aggressive

- You can be aggressive, don’t be annoying
- People care less than you think and will share more than you realize
- You can entice people to invite their friends, they will probably do it

Designing the App

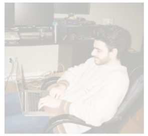
User Experience



jvheaney

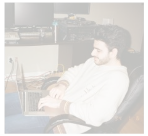
Users are dumb

- They are always dumber than you think, *always*
- Design your app to be as simple as possible
- Remove complication and replace with simplicity
- Confusion will lead to less use



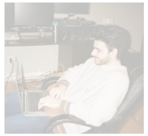
Make it a casino

- Break the stigma
- As addictive as crack
- Add colour, emojis, animations, elements that make it visually interesting and appealing to look at
- Keep them wanting to come back



Gamify

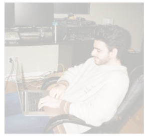
- Let users advance, earn something for their app usage
- Give them rewards that mean something to their motivations on the app
- Play with psychology, hints are more powerful and will keep people coming back



jvheaney

Simplify the flow

- Users should feel like they're already on the app *just by downloading it*
- Lean into exclusivity
- Remove unnecessary annoyances



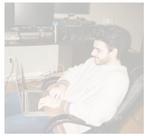
jvheaney

Leverage familiarity

- Use services they already have setup
- For mobile apps, make everything revolve around that phone
- Payment details needed? Apple Pay
- Signing up for an account? One time passcode texts
- Learn the device they are using and leverage it to it’s max
- Anything new is another point of friction

User Interface

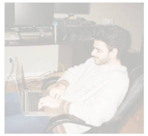
jvheaney



Visually appealing

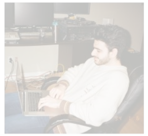
- Design it like a casino; smooth, clean, and uncluttered
- Lean into tasteful animations, make it snappy and quick
- Leverage your device; if you're on iPhone, make it feel like an extension of iOS
- Make everything simple, clear, and concise
- Instruction short and to the point, guided tours for anything mildly complicated
- Push them to complete your one central action

Philosophy



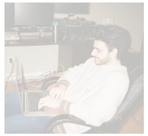
Components

- Make everything a component
- Build independently, bring together at the end
- Allows you to iterate easier
- Easy to swap out, and reuse when needed



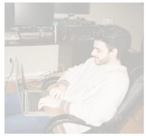
Subsystems

- Microservices philosophy without the headache
- Ease of maintenance and reduce points of failure
- Easy communication
- Monolithic architecture that can be separated is the best approach



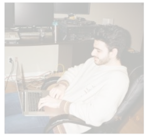
Front/backend independence

- Iterations are your friend, all-in-one frameworks are not
- Increases flexibility to swap technologies, platforms, hosting providers, etc
- Can scale them independently as required without burning a lot of cash



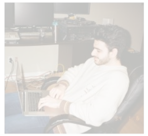
Simple tools

- Industry standard frameworks and tools
- Rust and Go are cool, but making money is cooler
- Is it worth 10x the amount of work to achieve the same thing as python?
- Keep it simple stupid



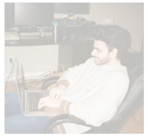
Everything in house

- Do not hire outside help
- Everyone working on the app should have a vested interest in its success, especially in the beginning
- Extreme familiarity and understanding with the code and tech stack is a must
- If something needs to be fixed *now*, who is doing it?



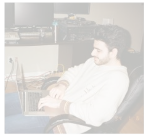
Iterative design

- Design with iterations in mind
- Experiments should be easy to do and easier to undo
- Little to no influence between subsystems
- Should be easy to pick a part and say “that didn’t work, now let me fix it”



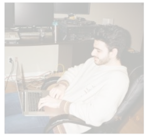
Native experiences

- Focus on the devices your demographic is most likely to use your app on
- In most cases, that's iPhone
- 74% of RU Mine's users used an iPhone
- It's hard to build and support both iOS and Android; so don't
- React Native and "code once deploy twice" frameworks miss the native feel
- Focus on the 80, forget the 20



Multiple providers

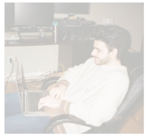
- Providers are sharks, they smell success in the water
- Have a backup plan, a second provider ready to go
- If providers start taking advantage, pit them against each other
- Use it to negotiate
- Reduce single points of failure, providers go down too



jvheaney

Don’t break

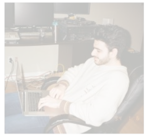
- Things are going to break, welcome to software
- Reduce it as much as possible, recover it when possible
- Hide it from the user unless absolutely necessary
- Safety nets in your backend
- Load balance and distribute as much as you can without it becoming a hassle



Ability to scale, not FOR scale

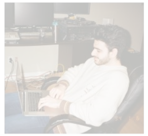
- What is not going to scale well? What's the backup plan?
- Pick technologies that can scale
- Do not optimize for it before it's needed, stay small and lean as long as you can
- Make sure your technologies can take advantage of auto-load balancing and auto-scaling tools

The Components



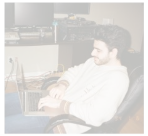
Friend/social graphs

- Who knows who, what connections can we suggest, invites, who likes who, who has influence in their circle
- Integral part of your app, dump interactions here
- Used in almost all social apps
- Clean data is key, everything flows downstream from this



Recommender systems

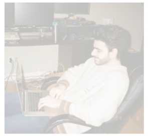
- Friends, posts, messages, content, gamifications steps, whatever is next
- Highly tunable, easy to change, social graph influences these decisions
- Tune for accuracy and speed
- Analytics are heuristics for this
- Machine learning is cool, but do not over engineer this



jvheaney

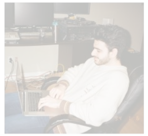
Messaging

- Real time private messages? Real time group chats? Asynchronous messaging?
- Maybe messaging outside of the app is best?
- Decide what is needed, complexity is bad



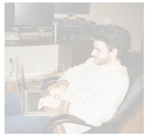
Timelines

- Based on recommender system, or chronological
- Can become complicated; do we generate on read or write?
- Implications of scale, is it worth it?
- Facebook and Twitter/X had and still have this issue



Searching

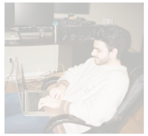
- Index of all your users, posts, etc, and their relationship
- Based on social graph
- Small recommendation system in there, stay relevant to the user
- More data, more indexes, more complexity



jvheaney

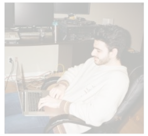
Sign up flow, OAuth, OTP

- Most important area of focus
- Simplicity sells, leverage existing and familiar services
- Reduce the amount of screens, keep auth simple
- Automate as much as possible, auto-fill in details if you can
- Social proofing works great, TikTok does a great job of this (become one with the crowd, don't be different!)



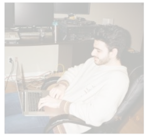
Posting

- How to deal with posting? Into groups, group chats?
- Private groups make apps look dead
- No user base related to the user, no content (Discord)
- How can I make it feel alive? Where's the FOMO?
- How can I encourage people to post?
- Animations, special interactions, public profiles/timelines



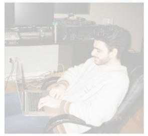
Real-time syncing

- Feels alive at the cost of complexity
- Constantly update likes/comments of posts as the user looks at it, makes it feel like people are using the app *right now*
- Downside, besides complexity: If your app is dead, it feels *really* ***really*** ***<u>really</u>*** *dead*
- Request load, it will now linearly increase the amount of requests to your backend, can you handle it?



Live streaming

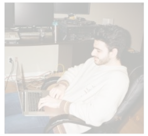
- Usually added later in the app’s lifecycle
- Very expensive, Twitch hasn’t returned a profit
- Bandwidth cost is immense
- Makes your app feel alive and present at the cost of a lot of money
- Big money sink, make sure it’s worth it



jvheaney

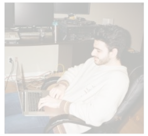
Posting and hosting photo/video

- Complexity and bandwidth cost
- Three factors: storing it properly, serving it well, paying for bandwidth
- Storage is not just more HDDs, it’s SSDs, RAM caches, and dealing with that, $/GB just went up
- Cheaper than live streaming, but still a money sink



Analytics

- Collect KPIs, service and performance metrics, other heuristics for recommendation system
- Nail this down, make it resilient
- Data from this component will be sole determining factor for decisions you make, keep it clean
- Build it once, build it right; or pay a service
- Service providers squeeze you here, so be aware and have backup plans

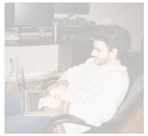


jvheaney

Safety

- Overlooked, but important
- ToS of app stores, requirement
- Content moderation plan if user generated content is allowed; how does that scale?
- Reporting and banning users is a requirement, build this into your recommendation and social graph systems

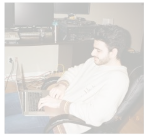
Launching



It’s go time

- Specific geographic area
- Keep your eye on the metrics
- When is it time to shut down and try again elsewhere?
- Identify what is working, what isn’t, and iterate
- Try, try, try again

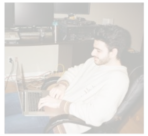
Scaling



jvheaney

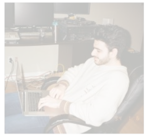
Do you have to scale?

- What metrics are telling you it’s time?
- Is there another reason those metrics could be returning that?
- Is there a simpler change you can make?



What needs to scale?

- Identify the components that need to scale, and why
- Databases? Do you have to optimize SQL queries? Build indexes?
- Servers? Do you have to optimize your code? Swap frameworks?
- Is it worth the time optimizing versus spending time on other parts of the business?

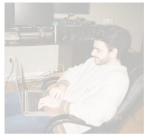


jvheaney

What will tank your business

- Will what you’re scaling tank your business?
- If it’s bandwidth/cost related, is it worth it?
- Do you have an exit strategy when it becomes too much?

Feedback



jvheaney

It can feel bad

- Actual people, actual names, maybe you recognize them
- You are an entity, not a person to them
- Find the constructive criticism, forget the bad
- Consumers don't actually know what they want, you should
- Rumours will start if you become big enough, it's normal
- Stay cool, calm, and collected

Are you sure?

Really??

James Heaney

Jvheaney.com
CTO @ CreatorCheck.io

Citations

Facebook user growth stats: https://www.thecrimson.com/article/2014/2/4/facebook-ten-years-feature-1/, https://finance.yahoo.com/news/number-active-users-facebook-over-years-214600186--finance.html?guccounter=1

Facebook popularity graph: VijayaChandra, J. & NarasimhamChalla, & SaiKiranPasupuleti, & Komati, Thirupathi Rao & Reddy, Vuyyuru. (2015). Numerical formulation and simulation of social networks using graph theory on social cloud platform. 11. 1253-1261.

General Facebook information: https://en.wikipedia.org/wiki/History_of_Facebook