Engineering manager Interview

1. How do you set goals for your team?

- first taking input from senior leadership, engineering manager in my case.
  - OKRs
- secondly input is from my team senior software engineers and software engineers,
  - they are hands on where is the big technical debts that we have that we need to consider.
  - what is the system that is a legacy and very hard to maintain packages or fix security vulnerabilities.
- after gathering all these inputs, we start ranking those initiatives priority based on our intuition , and experience .
- once we rank them, then we decide how much if these initiatives are meaningful to be done by the team. and a little bit extra as a stretch goal (in order to motivate the team to push themselves).
  - we estimate the time and effort in this take.

2. How to motivate engineer when the Q is all about the technical debt

- It is important to speak the truth, and not sugar code things.
  - engineers are smart.
- to see how is interested in doing this.
- for RTB,

3. Tell me how you consider the impact of your work?

- Achieving the team project vision, and see it live in production. and how much impact we did as a team to the world.
  - the success of the project does not mean is live in production only, also the porccess of it.
  - as I lead I consider the project is successful when we estimate the correct effort and time for it, and won't lead us to being over loaded to catch the deadline.
- Another impact of my work is helping my team in their career growth (promotion case)
  - it bring me joy to see one of my team member get promoted.
  - I have previous experience in promotion cases.

what is this world that you mean to impact

- customer aka (candidate / hirer)
- organization.
- team (how much influence you're making to the team and inspire them in terms of career growth)

we impact the world every day with bug-fixes , feature &

4. How do you know upfront whether you and your team work will make an impact?

- it's important to communicate and explain how important the work my team is doing.
  - what impact we will achieve to the company (in terms number) if we successfully launched the app.
  - it's also important to celebrate the achievement afterwords as it's give us energy.

5. What Signs Do you need to know that your team's work is impactful ?

- making sure to ask the product/ SEO and analytics teams for graph data to see how our work is very impactful.
- [SEO] redirection and traffic.
- decommission legacy AWS services.

6. How to do measure the impact of technical debts

- there is common misconception that only delivering feature has greater impact than addressing technical debt.
- addressing technical debt has lots of benefit,
  - saving us lots of cost.
  - saving us from revamping and create a new application.
    - no high test coverage.
    - no maintenance to packages.
    - we reached to the point where we cannot update the footer and header.
- the problem with technical debt the developers are slowed down.
  - there is duplicate code.
  - slow build pipeline.
- test coverages with the new apps.

7. How to avoid bard impact on customer

- code review.
- test coverage.
- architecture review.
- staging env for testing.

8. one mistake you did

- Don't show your stress to your team.
  - I learned not to give direct orders
  - in code reviews I used the words : what do you think about this, suggestions ...etc.

- Communication skills are very crucial.
  - I'm blocked I cannot continue.
  - I kept quite and all the blame when on me.

9. The art of Delegation

- Try to align tasks with people's career goals. (you want to review their work and guide it, but they will do alot for you)
- lead an initiatives
  - if you have an engineer who wants to be a senior.
- Design a new system.
  - a senior who is looking to be staff.
- be a direct manager
  - a senior who is looking to be a lead.
- Don't take on every task yourself just because you don't trust anyone but yourself to get it done.
  - hey can you design this system for me.
    - you need to be more specific.
    - what are the requirements
    - when needs to be done.
    - if you're being blocked, let us know. seek help. you not supposed to know everything.

10. If you have to delivery 2 initiatives while your team can only delivery one.

`Delegation` is really the key to not be overwhelmed.

you don't want to approach:

- oh I'm very over-loaded I cannot do this.

here is how I do it:

- precent the facts of what is going to suffer if you do take this on.
- you can always take a new tasks, but the consequences everything will be delayed.
- While also being very stressful and burnout.
- I'm happy to take this on just to align expectations I won't be able to take this until Mar for example.

if you can be discipline, focused and delegation when appropriate, the result will be:

- you are reliable.
- you're responsive.
- you're productive.

11. Think strategically act tactically.

- Strategically (thinking of product from business prospective)
  - why my team is building something.
  - why it's very important to the organisation.
  - what is the purpose.
  - motivation.
  - understanding why this is important,
    - it helps you deliver the right thing.
    - you prioritise tasks and feature according if you understand what is more important to the company.
  - example how this is important:
    - unification indirect system of JobsDB → could not understand why we were doing this.
    - enable company profiles and reviews.
    - just ask senior managers why this is important.
- tactically (thinking of product from engineering prospective)
  - how my team is building something

12. Delivering Results

the value of simplicity:

- as an engineer, you will be attracted to new trendy technologies.
  - maybe overly complex for what you need to do.
  - By always opting for the simplest solution that meet your requirements
  - and have fewer things to maintain going forward.

13. Techniques to insure the team is productive and delivering on time.

Avoid overcommitment

- having good planning. having good time estimate.
  - do not have time estimate that are too accurate. have a buffer time, why:
    - plan for the unknown.
    - plan for writing tests.
    - be kind to yourself → not to the point where you're too relaxed.
    - they don't just code non stop
      - they have meetings → they need to rest → they need to eat.
      - they have to pair with each other and solve things out.
      - not every one have the same level of productivity.
  - involve the team while I can.
    - for each big initiatives we have two steps (discovery, and delivery)

        - lead engineer → will lead the discovery which is doing the architecture, doing the high level blueprint and estimation. creating the roadmap of the delivery.
        - senior engineer → will take over then start the delivery.
            - they can help with estimate because the typically they are move involved with the code.
- quickest way to fail is overcommitment and lead the time to burnout.

Tactics: how things get Done (how to estimate larger project)

- Receive the architecture from the staff engineer.
- break down the project into small pieces, normally I divide them based on the repo they belong.
    - do we have an existing repo project for this or do we need to build a new repo for it.
        - if new repo normally building an app from scratch will require more time that adding a feature.
            - building a pipeline.
            - building a monitoring and logging. ..etc
- need to understand the availability for the people that they want to do this task
    - when my team will be available to start this project.
- need to understand the efficiency of my team members.
    - Seniors typically more efficient and faster in delivering tasks.
    - allow room for career growth → building motivation in team members.
- knowing all this from 1-on-1 meetings.
- we need to consider dependencies of the tasks and plan right when comes to time estimation.
    - internal dependency with our control
    - external dependency (need a team to do something for us), can get very ugly and as a lead I need to resolve it right away as it will block the delivery.
        - really need to think upfront how to handle these situations and resolve them before starting the project.
- and eventually you turn those to scrum sprint and how many dev needed.

1-on-1 meetings

- what are they working on right now ? Does that surprise you ?
    - make sure you team member is creating a ticket for everything he is working on.
- How did they spend their time this week ?
- Are they blocked by anyone or anything ?
- planning for career growth.

14. Situational Awareness

- pay attention in the daily stand-ups
- you need to be aware of the situation all the time.

- talk to the business analysis and ask them about the overall progress.
    - what is running behind.
    - what issues we are facing.

15. If we cannot deliver on time

- I need to talk to my manger.
- better come up with better plan and good estimate.
- understand what went wrong in the first time and fix it.

16. The Importance of Simplicity

start simple and walk your way up.

- too simple, may not be very the same functionality that every one need.
- too complex to the point is hard to maintain and deliver.

Example: Building a recommendation for pro-hirer.

- do not jump directly to building deep learning machine. and fully control of customization.
- do for example graph database like the one is provided by Redis.
    - based on your selection, hirers tend to select those recruiters.
    - recruiters categorised based on organisation industry.
    - top rated recruiters categorised based on organisation industry.
- the deep learning solution can be tried but it needs to be justified the higher cost
    - why this complex solution is better than the simple one.
    - the justification can be done with demo POC compare the two solutions
    - need to understand your team capability: we are not specialised in deep learning - data science, can we get a consultant.
    - OR can we do a could AI solution like AWS Bedrock - or chatGPT ..etc.

17. The art of working backwards

Work backward from the customer experience you are delivering, and figure out the simplest technical solution to deliver that experience.

- focus on the customer result. and go backwards.
    - the experience I'm delivering.
    - meeting the requirements
    - developing the simplest solution that can scale.

It is the apposite of working FORWARD from whatever the latest trendy technology is

- cool technology that I want to use and grow my career plan.

Example: Udemy course completion certification

- students only goal to show off their certification in Linkedin.
- there is no need to blockchain education verification.

18. Press release: The art of working backwards

- Design the system from the customer backward
  - start with the design UI.
    - what needs to power it.
    - what framework and tech stack I need to use in order to full-fill the organisation version.
  - Then the services that power that UI.
  - Then that data that powers the service.

19. The benefit of working backwards

- focus on what you're delivering.
- meeting your requirement.
- deliver system that meets your needs and not overly complex.
- it will be delivered most likely on time. → simplicity
- easy to maintain → simplicity.
- and everyone will be happy in the long run.

20. Retention Driven Development

- engineers likes to work on complex things and learn complex things.
- learning sessions. R&D work. allocate time for learning and discover. not only for the benefit of the team dev stack. but in general.
- internal mobility.

21. Running useful 1:1 meetings

1:1 objectives

- learn about status deliverables, anything blocking you.
- work toward their career goals.
- Provide timely feedback, positive and negatives.
- Let them provide feedback for you.
  - what feedback you have for me.
  - how can I do better.

The art of giving feedback

- very Important to give positive feedback, and congrat their achievement.
- do not shy away from delivering areas for improvement

    - Nobody should hear negative feedback for the first time on performance review.
    - Don’t make it personal, talk about the behavior and the person.
        - Hey I think this task is talking longer than the time we estimate, is there something blocking.
        - how can I help you.

22. Sample 1:1 agenda

- Check-in on project status.
- Check-in on performance goals. S.M.A.R.T.
- Check-in on career goals.
- Feedback (Positive and negative).
- Feedback for you.

23. Handling difficult conversation

- in emotional situation
    - listen first and don’t jump into solution.
    - validation of what they are feeling.
        - I can see why you’re upset.
        - yes you’re right it’s not appropriate.
        - but don’t take sides.
- Stay cool
    - don’t make promise's.
- Involve the HR
    - personal situations, you’re not qualified to fix the issue.
        - issues at home.
        - harassment.
        - if you tried to be involved you might get sued.
- Write down what you heard and what you said someplace safe and secure.

24. Letting employees be their best

- Solicit and nurture new ideas
    - Find ways to let them explore these new idea without sacrificing current commitments.
- Align tasks with career goals.
- Guide technical decisions without being proscriptive.

25. Managing growth at different levels

- New hire
    - find them a buddy.

    - set very clear expectations.
    - ensure work is tightly defined.
    - check-in frequently.
- Software engineer
    - find opportunities for growth lead infinitives or lead speace of a large infinitives.
    - improve pipeline.
    - pair with other engineers
    - tech talks.
    - check in weekly
- Senior software engineers
    - collaborating with other teams on a large projects.
    - or work toward management. mentor people.
    - check in bi weekly or as needed.

26. Someone in my team crashed the whole website and ended up with loosing money, how to handle a situation like that?

- Be honest.
    - don't throw blame on someone.
    - talk to the problem the facts not the person.
- Come with solutions not with problem without a solution.
    - PIR session. needs to be priorities.
    - Hey this happened, and here is the solution.
    - Show you understand the root cause of why this happened.
    - the solution will involve:
        - not only fix the careless thing that happened.
        - but also the come up with a solution why this happened in the first place [understand the ROOT cause].
        - prevent this thing from happening again.

27. Meeting advices.

- say I don't know, I will get back to you on this. I will follow up with you.
    - don't make things up, just to make yourself look good.
- for estimate time:
    - don't make an estimate based on guess.
    - be kind for yourself.
    - plan for testing as well.
    - plan for something went wrong during development.

- don’t speak more than you need to.
    - don’t take time more than you have to.
- don’t interrupt others.
    - mute your microphone.
- Speak Clearly.
- Be prepared.
- Be on time.
- Leave on time.
- Don’t call unnecessary meetings.

28. Overcoming introversion

- Extroverts tend to dominate discussions.
- Finding openings to talk without interrupting can be hard.
- Find an ally, can give you an opportunity to speak.

29. Building Team Morale

how my make your team motivated.

- build your tribe against the goals, we want to crush these goals.
    - communicate very clearly these goals what are we trying to achieve.
    - why we need to achieve it.
    - provide the statistic after launch [happy news].
    - apply team’s career growth into org goals. always look for opportunities to help people with their career goals.
- Team T-shirt, mugs ..etc.
- Team building ..etc.
- if they are working late provide food. ..etc.
- generate identity
- keep the right people on the bus.
    - hey this is really dragging down the productivity of the team.
    - I need you to stop doing this, if you can.
    - if you have any issues, I’m happy to talk to you about them.
    - but don’t spread that negative energy.
    - because it does not benefit everybody.
- You want to make your team

30. Good Goals = Good Morale

31. Happy Team = Productive and Motivated team.

32. When a project is shut down, how to communicate this with your team?

- Understand from the senior management why is been cancelled, and I need to communicate this with my team.
  - hey this project is been shut down because there is another exciting things are going on.
  - tech them about the next big thing project: don't leave them wondering what will be their fate is going to be tomorrow.
- Deliver the news on person. Needs to be in a meeting.
- Lead by example, even if you're feeling down don't show it to them.
  - if you're seen you're upset about it, my team will be upset about it.
- Recognise the great work done on.
- Recognise what we have learned is very valuable.
- Take the work and knowledge from that cancelled project and apply it in the new one.
  - use it to accelerate future work.
- Keep the focus on the company vision, on the large strategic goals.

33. Growth as a Manager

- Find a mentor.
  - find someone who is more experience than you're.
  - my own manager is my mentor too.
- Understand the business.
  - why is been build.
  - think of these larger corporate goals.
- Learn from the decisions you don't understand.
  - They have boarder prospective that you lack.
  - Don't get frustrated from the decisions that you don't understand, Try to understand them and learn from them.
    - Just ask why.
- Observe upper management and learn from them.
- Build good relationships outside my team.

34. Lead Engineer is a team contributor not an individual contributor

- You will find that the larger results that I can deliver as a lead engineer for a team becomes addictive.
- You can get a lot more done and achieve larger goals with a team at your back than could as an individual.
- It's powerful tool/skill to inspire my team and build Great Morale to crush these goals.
- important not to micromanage people.

35. Strong Mentor VS Micromanagement

it's really fine line.

- few heads together will come up with better solutions, than just one person will.
- I need to rely on them and trust them with solutions that a lot of times I wouldn't have thought of myself.
- However at the same time I need to objectively measure and judge those ideas and those suggestions are they good or are they not. Are they obvious pitfalls here that I have seen play out in the past, if we go down this route will obviously happen ?
- Is it one of those things, hey let's give it some latitude here maybe it will go poorly maybe it won't but the learnings are worth it.
    - make calculated risk: which DB ORM to choose.
- if the team is afraid the first mistake is going to be fired they will never learn.
    - overly protect their image their reputation.
    - they will lie, hide it
    - never try new things
- What you reward as a leader gets repeated.
    - promote phycological safety.
- I lead the whole person.
    - how is week going? encourage it's okay to be human in the workplace.
    - the team will start helping each other, they will learn what some of the cues are that person's let me give them .
- if you make it, it's okay to fail and learn from it. and build a collaborative team that can help each other.
- if they feel safe and they have a courage to try new things and innovate, the team will take me places that I never thought I will go.

36. As a lead I'm responsible to provide feedback to the Engineering manager about my team.

- I don't only rely on the one-on-one to measure performance, I also ask senior Engineers about their team members.

37. How to help underperformance

- It shouldn't come as a surprise to the underperformance nor to the EM.
- I must provide this feedback regularly during one-on-one sessions.
- Why is this happening?
    - Are they working on something they don't like?
    - Are they not getting the necessary help that they need?
    - Is there something on the personal side that is impacting their professional performance at work?
    - Is there a technical knowledge gap?
    - Is the Engineer not asking for help when they need it ? (communication gap).

- You want to get to the root cause of the problem by asking the right questions.
- you break it down into small parts, have some milestones set.
    - goals to achieve in order to fix the issue.
    - As a lead engineer I need to provide the solution to fix the issue before the performance review.
    - you have one-on-one every week.

38. How to Keep these Top performers motivated?

- As a lead engineer I need Acknowledge and recognition the hard work the Engineer is putting.
- Give them better challenges.
- Promise to propose a promotion case to the senior management (in-role promotion)
- for senior to lead or staff → If you don't have capacity for lead and staff position take the initiatives to find them an opportunity outside the team but with-in SEEK (internal mobility).
    - why? because if they stayed to long in their comfort zone they will feel bored and leave the company seeking better opportunity.

39. How to retain an Engineer?

- I will ask Engineering Manager, do you want to keep this Engineer as all costs?
- if yes, you definitely need a plan of action if I'm going with this conversation
- I will try to find out what is causing this engineer to look for another job.
    - Could be the engineer not been compensated well?
    - Could be the engineer looking for new Challenge in a new domain.
    - Could be Work-Life Balance?

40. Set someone for promotion case?

- Build a good promotion case, it needs to be planned probably 6 months - 9 months or 1 year.
- you need to perform at the next level already.
- need to be very clear what is the expectation of the next level, and give them that opportunity to demonstrate these skills.
    - working on bigger projects.
    - leading projects.
    - better documentation.
    - being able to lead system design → senior to staff.
    - collaboration with other teams.
    - pair with other engineers and unblock them.
- The important thing here, you need to continuously need to meet that level. not only once.
- you guide them once, and see of the engineer is behaving like a senior by himself.

41. Owning the technology If there is something went wrong I need to own it and fix it

- Monitoring, browser and API test monitoring.
- we need to start scaling things horizontally.
- we have on call schedules
- communicate very clearly and set expectations and have empty.
- don’t assume they have all the answers.

42. [IMPORTANT] Software Development Life Cycle.

[discovery] Requirement gathering Analysis

- Define what is the problem, and how the product team wants to solve it.
- Work closely with the product team to understand the requirement.
    - Functional and non-Functional requirement.
- How many users will have → will define if horizontal scaling is needed or not.
- Ask about availability → will define the test coverage and monitoring.
- Ask about the budget.
- sketching user interface → work with UI/UX designers.
- All these decisions need to be documents using confluence page where they are available for both dev & product team.

[discovery] Design → system design.

- Working backward and understand the vision of the product.
- Always prefer the simplest solution.
- Work with product team to define the scope of each release.
- Based on how large the system is will go with either microservice or monolith.
    - If very large budget and very large project → microservice each service with be handled by a team.
    - If very tight budget → monolith.
- I will consult with staff and architecture team.
- This phase defines the elements of a software development, the components, the security level, modules, architecture and the different interfaces and type of data that goes through the system.
- Finally implement a POC to increase the confidence.

[discovery] Planning and Effort estimation

- Work with the product team to devide their vision into releases.
    - Will start with MVP then phase 2, ... all the way to the last phase.
- Based on the deadline given I need to estimate how many developers I need to finish with in the deadline.
- Always avoid bottlenecking in the dev team at least 2 dev in each domain if one is on-leave.
- Avoid overcommitment → be kind to yourself and team when estimation.

[implementation] Implementation and Deployment.

- at this stage you not suppose to have uncertainty.
- integrate with CI/CD.
- the actual code is been written here.
- set the standard of code architecture.
    - devide the project into repos,
    - implement CI/CD to insure reliable deployment.

[implementation] Set testing standards

- Bug bash stage.
- We want to focus on testing since it should be highly availability.
    - We cannot effort down time.
- Unit test, integration test, e2e test and syntattic monitoring, alerting from datadog to pagerduty.
- Have on-call schedule in the dev team.
- Have a QA team only when there is budget for it.

[implementation] set maintenance standards

- Integrate with Renovate and Snyk.
- Continuously evaluating system's performance.
- prevent from security threads.
- prevent from package support outdating.
- prevent revamping after a while. it will increase the age of the system.