Paypal

- emphasis on customer-first mindset

Tell me about yourself

- I'm a year 3 CS student at NUS.
- A little fun fact: in my first year I studied pharmaceutical science, got bored of it, realized I much preferred fast-paced problem solving, so I made the jump to CS, and so far I'm really enjoying it.
- So far I've been fortunate to have had 2 internships, the first at MailShield AI, a startup that uses AI to detect scams in emails. And the second at GovTech where I worked on an API portal, a platform that allows government API publishers and consumers to manage their APIs.
- Outside of internships, I also enjoy building personal projects to sharpen my technical skills and to learn new things (eg. technologies that are new to me).
  - Eg. I am not broke, very first web app, in which I learned the basics of full-stack development
  - CVWO web forum, a full-stack web forum, in which I picked up web technologies that were new to me.
  - ~~TinyKV, a key-value store implemented in C++, I did it to learn the language and because I wanted to dabble in a bit of systems programming.~~ (not rly relevant here)
  - Concurrent web server in C++, to experience working on concurrent systems.
- Through these projects, I’ve realized that I love the process of SWE, of breaking down some abstract requirements into smaller, managageable steps and bringing an idea to life through code.
- I also realized I enjoy building **reliable systems** that people actually use —> seeing the tangible impact excites me and makes me feel fulfilled.
- And that's why this internship with Paypal excites me, because that's exactly what I'll get to do
- Excited to contribute both my technical and soft skills to the projects at Paypal, which directly impacts milliions of users. I find this very exciting and fulfilling, to know that I'm helping to build software that makes an impact.
- I also see this as a valuable opportunity to learn how large scale systems are maintained and optimized in the real world. I'm **excited for the technical challenges I'll face**, and to **learn from these challenges** so that I can grow into a stronger and more well-rounded SWE who can create products of value in the future.
- And that's why I'm really excited for this internship oppt with Paypal.

Why Paypal?

1. I’m excited about this oppt with PayPal because it gives me the chance to work on **large-scale systems** where **reliability** and **performance** are essential. ~~I'm excited for the technical challenges that these will bring, and I'm eager to face it head on to learn and grow as a SWE.~~ **I see this as a valuable opportunity** to learn and understand how large-scale, production-grade systems are maintained and optimized IRL, and **I find it really exciting that I'll be helping to build products used by thousands to even millions**.
2. I'm drawn to the culture of **innovation and engineering excellence** at Paypal. **Paypal is known to be an innovative leader in financial technology, allowing it to build products that have a global impact**. As someone who enjoys finding innovative solutions to problems I face, and who always strives to do my best in every pursuit I take on, this culture resonates with me and I and excited to be a part of it.

3. Lastly, I'm excited for this opportunity with Paypal because I'm eager to work with and learn from the experienced engineers at Paypal. I hope to be able to learn from their **code discipline**, **best practices**, and **how they approach complex problems** such as scalability. I hope to be able to take all that I've learned with me to become a stronger engineer, and build products of value in the future.

Why should we hire you? / Explain why your background and experience would be a good fit for this job.

- Believe I'll be a strong fit because both my technical and soft skills allow me to contribute meaningfully to the projects at Paypal.
- Through my coursework and personal projects, **I've built a strong foundation in backend development**, with proficiency in languages such as C++ and Python, as well as **exposure to** cloud technologies like Docker, and AWS -> enable me to contribute to the codebase
- **I love a challenge** - Challenging problems excite me because even though it's painful to think about, but I feel myself getting stronger as an SWE after that. It usually signals some patterns that I have yet to pick up and that excites me. I believe this puts me in a good place to tackle the various challenging problems that I'll face at PayPal.
- (Eagerness to learn and improve) I am someone who has a **strong drive to always learn and improve**, to do things better than the previous time around, and I believe this will enable me to pick up new technologies fast, and adapt to the pace at Autodesk, and hence contribute meaningfully to the projects at Autodesk.
- I believe that my **technical skills**, love for challenges, and **strong drive to learn and improve** would allow me to contribute meaningfully to Autodesk's projects.

Strengths

- I think my biggest strength is my resourcefulness.
- I think my biggest strength is my drive to always learn and improve myself. In every project I take on and in every internship I go to, I always try to get something out of it, to learn something from it.
- Eg. in my full-stack web projects I always try to learn new things each project I go to - from new technologies or languages that were new to me, or finding a better way to do things, learning how to design a system from scratch, etc.
- And in GovTech, I realized that technical-wise the skills weren't very new to me, but I tried looking for other areas through which I could grow;
- Asked myself what can I learn here that I can't in my personal projects?
- Learned how to communicate my ideas to my team members, how to reason about my decisions; how to visualize my ideas eg. in the form of diagrams; and the importance of documentation
- And now I take this skill with me in every project that I work on - I lay out the problem, the different solutions and can weigh them out; I visualize my design through diagrams and this has helped me greatly when I build my personal projects.
- Across all these projects, the common theme is that I'm always looking for ways to improve and to pick up new concepts, while doing my best to deliver something that works."
- This strong drive to learn and improve as a SWE ensures that I'm always learning and growing, and doing things better than the previous time around, shaping me into a stronger SWE.

Weaknesses

- One of my weaknesses is that I can sometimes be too perfectionistic. In the past, I would spend hours thinking about the most optimal solution and how to do something instead of just starting to implement it.
- I realized this was slowing me down, so I've been working on p**rioritizing progress over perfection.** Now, when I catch myself overthinking, I make a conscious choice to pick one reasonable approach and try it. Even if it isn't perfect, I usually learn much faster by testing and iterating than by staying stuck in planning.

- This shift has made me more productive, and it's helped me balance quality with efficiency."

Where do you see yourself in 3-5 years time?

- (design and maintain **reliable**, **high perf systems**) In 3–5 years, I hope to become a **well-rounded** software engineer who can **design and maintain reliable, high-performance systems** end-to-end.
- (what field?) I want to **deepen my expertise** in **systems programming** and **scalable architecture**, and become the kind of engineer who can work confidently across different layers of abstraction — from web applications to infrastructure.
- (maintain tech blog) I also maintain a technical blog where I write about my projects and the things I learn; **I hope to keep growing it** as a way to reflect, consolidate my knowledge, and share it with others.
- (mentor juniors) And as I gain more experience, I want to mentor junior developers — because I've benefited so much from guidance myself, and I'd like to pay that forward and help others find their footing too.

Resume Deep Dive

PROJECTS:

- **Purpose & motivation** – what problem or curiosity drove you to build it?
- **Tech stack** – show what you picked and *why.*
- **Architecture or technical decision** – one interesting design choice (e.g. Docker setup, concurrency model, caching, etc.).
- **Challenge faced & how you solved it** – debugging, design, performance, new tech, etc.
- **What you learned / takeaway** – ideally phrased as a growth reflection.

INTERNSHIPS OVERVIEW:

1. **P**roject
2. **A**ction
3. **R**esult
4. **L**earning

💬 **Example structure**
1️⃣ **Project / Context** (what the team does & your role)

> "I interned at GovTech on the API Portal team. The API Portal is a platform that helps government agencies publish, manage, and consume APIs securely and consistently. My role was to help improve the backend and build new features to make the portal more useful for both publishers and consumers."

✅ *Purpose:* Sets context clearly. The interviewer now knows what the product does and what your scope was.

2️⃣ **Actions / Responsibilities (what you actually worked on)**

> “I worked on several backend features and enhancements. One example was a feature that allows users to manually define backend API specifications instead of uploading configuration files — this was important for users who had company restrictions preventing them from uploading files.
> I also contributed to improvements in the search functionality and helped refine some frontend components.”

✅ *Purpose:* Shows breadth, but still gives a clear example of ownership.

3️⃣ Result / Outcome (impact or completion)

> “These features were integrated into the development environment and tested internally, and the feedback from the UX lead was positive — the manual API creation feature was seen as intuitive and user-friendly.”

✅ *Purpose:* Even if small, show closure — that you finished and delivered something valuable.

4️⃣ Learning / Takeaways (your reflection)

> “Beyond the technical side, the internship taught me a lot about communicating my ideas clearly — using diagrams, reasoning through trade-offs, and aligning with my team before implementation.
> I also learned the importance of taking initiative to clarify requirements early rather than waiting for detailed instructions.”

✅ *Purpose:* Ends on a reflective, growth-oriented note — always leaves a good impression.

1) I Am Not Broke

2) CVWO Web Forum

- **Purpose & motivation** – I built it for a
- **Tech stack** – Built with Golang, containerized with Docker, deployed with AWS EC2, because I wanted to learn a new language, and wanted to learn containerization and deployment.
- **Architecture or technical decision** – controller-service-repository layer setup.
- **Challenge faced & how you solved it** – the challenge was learning how to structure my BE code. Another challenge was deployment; I didn't really understand what I was doing, so I had to consult online documentation + talk to chatGPT to understand the concepts for Dockerization.
- **What you learned / takeaway** – ideally phrased as a growth reflection.
- I built a **full-stack web forum** as part of the CVWO assignment. It allows users to create, edit, and comment on posts, with authentication and tags for posts. The backend was a REST API built with **Go**, and the frontend with **React**.
- DB: PostgreSQL, which my server interacted with via an ORM (Gorm).
- I containerized the app using **Docker** for consistency across environments, and deployed it to **AWS EC2**. For authentication, I implemented **JWT tokens**, so each request is securely validated.
- The project helped me strengthen my understanding of **web architecture, deployment, and API design**, and taught me **how to structure a project cleanly so it’s maintainable** as it grows.

Project structure

- controller service repo model
- **controller:** the first layer, interacts directly with the request - handles request validation, parsing, etc.
- **service:** the next layer that handles business logic. Communicates with DB via repo layer
- **repo:** layer that communicates with DB.

JWT Authentication

LOGIN FLOW: (Login controller handles steps 1-4)

1. **Client sends credentials** (`email` + `password`) in a `POST /login` request.
2. **Backend verifies credentials** against the database.
3. If valid → backend **generates a JWT token** representing the logged-in user.
4. The server **returns the token** + and optionally sets it as a HTTP-only cookie (m,eanng only the browser can see it)
5. On subsequent requests, the client includes the token — either via `Authorization` header or cookie — and middleware verifies it.

Why JWT for auth?

- In order to keep the servers **stateless and scalable**.
- JWT allows backend server to verify the user's identity w/o needingna DB call (it just needs to use its own private key to decrypt the secret, then check if header + body == secret (or smt like that butn i dont really rmb)

What Docker containers you used and how they talk to each other.

- Backend, frontend, and postgres
- In my setup, Docker Compose automatically created a **shared internal network** where containers can communicate by service name.
- The backend connects to Postgres internally via `postgres:5432`, while the frontend and users access the backend through the EC2 host's public IP (`13.237.169.162:8080`), which maps to the backend container's port.
- This setup keeps internal traffic secure within Docker while exposing only the necessary ports to the outside world.

How deployment worked (EC2 setup, .envs, etc.).

- Ran the containers on an elastic cloud instance.

One technical challenge + how you solved it.

-

What you learned (deployment, ownership, testing).

- Picked up new language (Go), and technologies that were new to me like Docker, AWS; also first time deploying
- I learned how to structure my backend code to make it scalable

- I learned how to develop iterateively - eg. using mocks, then refactoring with API calls

2) TinyKV

- LSM-tree based key value store; flush to disk when full; writes go to WAL (on disk); WAL is deleted when memtable is flushed, OR WAL is deleted the moment it is written to disk.
- **Purpose & motivation** – I wanted to learn C++ and also wanted to learn systems programming. I thought a key-value store would be a good place to start because I was curious how they worked.
- **Tech stack** – show what you picked and *why.*
- **Architecture or technical decision** – I chose to implement LSM-tree based key value store because it was how many noSQL DBs are implemented today, like CassandraDB, RocksDB etc. And I thoguht it interesting to build one too.
- **Challenge faced & how you solved it** – The challenge was transforming the idea into implementation; how to represent SSTables (which were a bunch of files grouped into different levels); and how to implement compaction
  - Looked at different implementations to get ideas, and I built off of that
  - For example, online implementation only showed SSTable with one level, so I translated that into multiple levels
- **What you learned / takeaway** – I learned how to break down and model a system using OOP; and I learned how to utilize existing implementations to get ideas for my own implementation as well

3) Concurrent Web Server

- **Purpose & motivation** – I built because I wanted to gain exposure to concurrent programming in C++.
- **Tech stack** – Built with C++ and CMake
- **Architecture or technical decision** – thread pool worker model; one thread per worker, fixed pool of threads to take job from shared queue
- **Challenge faced & how you solved it** – writing a concurrent queue from scratch; I had never done one before. So I started with writing a normal queue, then worte a test to test if it worked as expected for single thread; then implemented with mutex, by identifying which resources were shared and stuff; then tested with multiple threads to check if the behavior was as expected.
- **What you learned / takeaway** – This project deepened my curiosity about how real servers handle concurrency at scale. While researching different concurrency models, I realized there's rarely a single "best" solution — each design involves trade-offs between performance and simplicity. That inspired me to keep refining my implementation and document possible improvements on my blog.

DateIdeas

-

4) GovTech

TEMPLATE: (use for answering qns)

- **Context:** API Portal for govt APIs.

- **Feature:** Manual backend API spec creation (your STAR story).
- **Key tech:** REST APIs, Next.js frontend.
- **Challenge:** unclear requirements → you proposed two approaches.
- **Learning:** proactive communication, reasoning about trade-offs, documentation, feedback loops.
- **Backup story:** global search feature → teamwork with another intern.

Tell me about your experience working in a team

- **SITUATION:** Previous internship with GovTech, I worked with teh APEX Cloud team, a team of developers + UX lead + product lead working on APEX Cloud, a portal that allows producers and consumers of govt APIs to manage their APIs. Every 2 weeks we would have a **sprint planning session**, where we would come together and plan for the next sprint - what we wished to accomplish, backlog alignment, etc.
- **TASK:** To make the platform intuitive for users so they feel **confident** using it. Part of this was the global search feature me and another intern were tasked with.
- **ACTION:** The other intern and I had a productive meeting where we defined our responsibilities; then we worked in parallel to complete our individual portions. Afterwards we ...
- **RESULT:** Demo was well-received, and the feature was eventually released for use. And this taught me that working in a team is more than dividing tasks; it's about constant communication, giving and receiving feedback, and iterating together toward a shared goal.

What were some challenges you faced working at GovTech?

- One of the main challenges I faced at GovTech was **working with a large, existing codebase** for the API Portal.
- Working on my first endpoint, not sure how to structure the code based on the team's conventions.
- So I took an example EP as reference and tireed tracing throught it - saw that ut lowed from controller -> composite -> service -> rejpo layer
- not sure what the composiite l;ayer was so I asked the tech lead who exmplained tone
- **tracing through the code** and
- **reading API logs** to understand the data flow. **And also consulted online documentation**.
- I **analyzed the code**, and **clarified with full-timers** if my understanding was correct.
- Through that process, I learned how to navigate complex systems more independently, and how to adapt to different coding conventions and codig styles encountered in different eams.

What were your biggest achievements from working at GovTech?

One achievement I'm proud of is learning to **propose solutions confidently** and **reason through trade-offs**.

Situation: I was tasked with building a feature that allowed users to manually define backend API specifications on our API portal, which they could then use to create frontend APIs. **I faced a problem:** deciding how to handle the relationship between backend and frontend APIs — whether to reuse existing backend specs in the database, or create a new copy for each frontend API.

Action: I analyzed both options: reusing backend specs would save storage but risk unintended coupling if a shared spec changed, while copying would take up more space but ensure isolation. I proposed both scenarios to my team lead and reasoned that reusing backend specs would be more intuitive and convenient for users, even at the cost of slightly more database space.

Result: My lead agreed with the reasoning, and I implemented and tested the feature successfully. When we demoed it, both my supervisor and the UX lead found it intuitive and easy to use. The experience really boosted my confidence in making design decisions and communicating them clearly.

- **Taught me to make my own decisions instead of waiting for prescriptions**.

GovTech tech stack?

What did you learn from your internship at GovTech?

1. Technical skills-wise, I learned how to write unit tests (lol)
2. Communication & Teamwork - how to communicate my ideas to the people I'm working with.
    - How to communicate an idea with the use of diagrams
    - How to give and receive feedback
3. Taking initiative to find solutions to problems instead of waiting for prescriptions
4. Importance of documentation
5. My internship at GovTech taught me a lot, both technically and personally. On the technical side, I learned how to write proper unit tests and understand the importance of testing in maintaining reliable systems.
6. More importantly, I learned how to communicate effectively within a team — how to present ideas clearly using diagrams, how to give and receive feedback, and how collaboration shapes better solutions.
    - Eg. our team was talking about how we should store some ID thingy
7. I also learned to take initiative instead of waiting for instructions — to reason through trade-offs, propose solutions confidently, and ask the right questions when I'm stuck.
8. Lastly, I gained a real appreciation for good documentation. Working in a large codebase made me realize how much time clear, up-to-date documentation saves for everyone on the team.

5) MailShield AI

- **Context:** startup using AI to detect email scams.
- **Feature:** dashboard design or backend endpoint.
- **Challenge:** PDT lead vs UX lead disagreement.
- **Action:** scheduled short sync, summarized both perspectives, facilitated compromise.
- **Learning:** mediation, communication, aligning stakeholders, teamwork in small startup setting.

ADDITIONAL STAR TYPE QNS

1) Received feedback & worked on it

- GovTech (perfectionism -> iteration)

2) Unclear requirements

- GovTech (backend API spec; unclear which way to approach; reasoned about different approaches and communicated to team lead)

3) Conflict/disagreement (eg. conflicting feedback)

- MailShield AI (product vs UX)

4) Initiative

- GovTech -> after feature done, check with UX lead if satisfied

5) Describe a time you contributed to a team's success.

- Global search feature led to a **well-received demo** and it was eventually pushed to production?
- **SITUATION:** At GovTech, I was tasked with building a global search feature for our API portal, which was core to improving the consumer experience.
- **ACTION:** Another intern and I worked together to develop the feature end-to-end, from designing the backend APIs to implementing the frontend. After we had a first version, we actively gathered feedback from our teammates, refined the UI and UX, and then pushed it to the development environment. The feature was well received — both internally and by early users — and it became an important part of how people navigated the platform.
- **RESULT:** I learned that a big part of contributing to a team's success is not just delivering code, but collaborating effectively, incorporating feedback, and ensuring what we build truly serves the users."

Tell me about a time you learned a new skill quickly.

- **SITUATION:** This past summer, I wanted to learn C++ and challenge myself to build a project in it, but I could only started in mid-July and had 2 weeks to learn + build project alongside GovTech internship.
- **TASK:** At first, my goal was simply to pick up the fundamentals quickly enough to begin coding.
- **ACTION:** I focused on the basics — syntax, memory management, and OOP — and started building right away instead of waiting until I felt 'fully ready.' As I worked on my project, I encountered concepts like templates, Standard Template Library (STL) and RAII, which I learned on the fly through documentation and practice.
- **RESULT:** By the end of summer, I had a working key–value store implemented in C++, and while I know I still have a lot more to learn, I came out of the experience much more confident in my ability to pick up new languages and apply them to real projects under time pressure."

Tell me about a mistake you made and what you learned.

-

6) Tell me about a technical project you're proud of.

Situation:
One of the projects I'm proudest of is a concurrent web server I built in C++.
It started as a personal challenge — I wanted to understand how real web servers handle multiple client requests efficiently, and to push myself beyond the higher-level projects I'd done before.

Task:
My goal was to implement concurrency from scratch — build a system that could handle multiple connections at once without sacrificing correctness or reliability.

Action:
I began with a simple, single-threaded server to make sure I understood the basic flow.
Then, I designed a thread pool with a shared job queue, where each worker thread picked up tasks to process.
To ensure correctness, I first implemented a normal queue and tested it in isolation, then gradually added locking mechanisms and verified it under multi-threaded conditions.
Throughout the process, I benchmarked performance after every iteration, documented what I learned, and researched alternative designs like event loops for future improvement.

Result:
In the end, I saw a visible performance improvement compared to my initial version, and I came away with a much deeper appreciation for synchronization, testing, and reliability in systems design.
Even though there's still a lot I'd like to improve — like optimizing the queue or experimenting with non-blocking I/O — I'm proud of how I approached it: starting simple, adding on complexity, and continuously thinking of ways in which I can improve it.

Example of a time you automated something

- Talk about your blog and how you automated the process to
- upload to supabase so the flow is super smooth, just edit on obsidian + run script
- Or the Telegram bot we made so we don't have to manually upload photos to Google Drive

Describe a difficult task you were faced with and how you addressed it.

- **TASK:** Implementing durability (even when system crashes, data is not lost) for tinyKV.
- **SITUATION:** It was difficult because I wasn't sure how to ensure durability - I was unfamiliar with the idea of writing to files on disk and how it would work.
- **ACTION:** To tackle it, I broke the problem into smaller steps. I first implemented a simple in-memory key–value store to understand the core operations. Then, I added a basic persistence layer — writing entries directly to disk to see how data could be stored and reloaded. Once I understood that flow, I iterated and introduced more robust components: a **Write-Ahead Log** to ensure changes were recorded before being applied, and **SSTables** to store data on disk in a sorted, immutable structure. Each step gave me a working checkpoint and built up toward the full solution.
- **RESULT:** Implemented a durable and persistent key-value store with SSTables, WAL, and compaction.
- **Takeaway:** Shows persistence, problem decomposition, and learning on the fly.

Describe a time you had to make a difficult decision and what you did, what choice you made, and why.

- fluff something like make up a story lol
- eg. a time you had to decide between shipping fast or making it efficient

Teamwork and collaboration

- **Tell me about a time you worked well within a team.**
- **Describe a situation where you had a disagreement with a team member. How did you handle it?**
- Tell me about a time you had to work with a difficult coworker.
- How do you handle conflict within a team?

Problem-solving and challenges

- Tell me about a time you faced a really hard problem or a challenging project.
- **Describe a situation where you had to work on a tight deadline. How did you prioritize?**
- Can you give an example of a project that didn't go as planned? What did you learn from it?
- Tell me about a time you had to learn a new technology or programming language.What was that process like?

Questions

- “How do you usually define success for interns on your team?”
- “What kind of tech stack or tools would an intern typically be working with on this team?”
- “What’s an interesting technical or non-technical challenge your team has faced recently?”
- Is there anything / any tech stack that I should familiarize myself with before starting the internship?