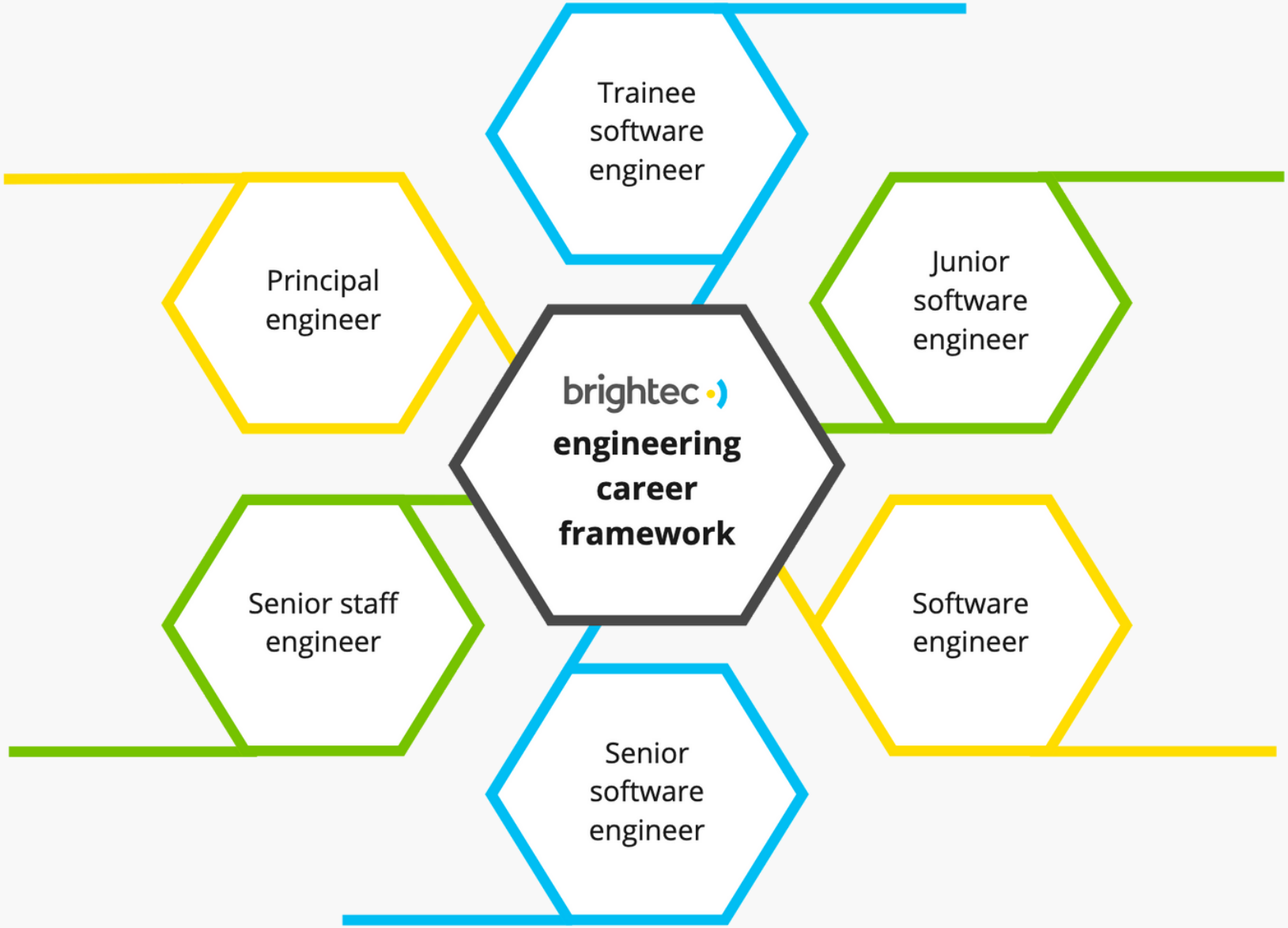
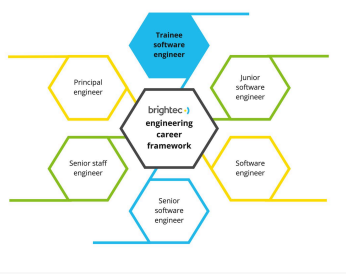

Trainee software engineer
Junior software engineer
Software engineer
Senior software engineer
Senior staff engineer
Principal engineer
brightec
engineering career framework

Trainee software engineer



A **trainee software engineer** is an aspiring engineer just starting to explore a professional career in software development. They will generally have some programming or software engineering experience, but it may be from purely academic or hobby contexts. They are still learning the trade, and are not expected to be a fully rounded engineer. They may have no previous commercial experience, or may have only been involved in internships or work placements, up to the point of being a viable contributor to a client project.

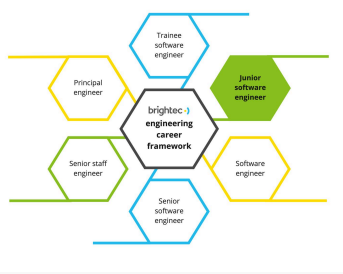
They are likely not entirely competent or performant in any software technology, but should have some foundation and a viable growth and learning path. There is a strong expectation that they will progress within a defined timeframe, with defined guidance and support to that end, and this will usually be within the context of focussing on a single technology area (e.g. Android, iOS, Web, Hybrid/React-Native).

A trainee's primary objective is to develop and demonstrate the skills required to become a permanent Brightec software engineer. This will involve a mix of self-study, formal training, team participation, shadowing and being mentored by other engineers. The journey should normally take 3-6 months depending on the starting technical foundations.

A trainee is not billable to clients, though they may take an active part in client projects as part of their training. In the context of client projects, they complete well scoped and defined tasks with hands-on guidance. Interns currently on temporary placement with Brightec will also be classed as trainee software engineers.

Anti-patterns: Lack of humility or eagerness to learn. Not a quick-study or keeps repeating the same mistakes. Unfocussed or unstructured. Lack of communication or accountability.

Junior software engineer



A **junior software engineer** is an entry-level professional software engineer. They may be a new graduate, or a graduating trainee. This is likely their first professional software development role.

They are capable of independently producing steady work output in a single software technology area. Their overall execution is expected to be slower, still being mixed with a lot of constant learning in their technology domain (example: learning more platform APIs), but consistent and within a reasonable timeframe. They should be seeking out guidance and eager to learn at every opportunity, and mentoring and guidance should be provided for them in response. They should be self-motivated, working without needing to be told what to do next.

They have fundamental language and syntax knowledge and can deliver basic code to meet specified requirements, usually working on tasks defined by others. They can read and understand the code around them, learning one or more client projects in increasing depth. They will not always come up with the right solution on the first try, but will independently attempt solutions using judgement and problem solving skills, and will participate in code review and take account of feedback. Some back and forth on code reviews is not only expected, but encouraged. They can investigate and fix bugs and issues. They should aim to be comfortable contributing one or more small code changes per day (e.g bug fixes, small changes), and also capable of crafting medium and large changes over the course of several days with some design guidance.

When given a task with unclear or incomplete requirements they should be able or learning to identify this and ask for clarification. They should become more proficient at identifying underlying assumptions in design and implementation that may need clarification or course-corrections.

They can pick up and follow team and project procedures and practices, and conduct themselves professionally both internally and with clients. They identify and proactively communicate status, progress and blockers to their team. They should learn and practice with guidance pushing code through the entire development life-cycle (from design and coding to production release).

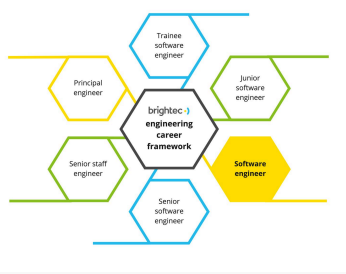
They will be actively working to develop a well rounded knowledge of software engineering tools and processes (source control, editors and IDEs, core CS concepts, software design and design/architecture patterns, testing). They will have developing strategies for personal productivity and work-life balance. They should take an active interest in the diversity of engineers around them.

Some of their work may end up being abortive or un-billable, but most will be billable (usually 80%+ depending on project/client circumstances), with a tapered ramp-up for new joiners (generally 20% → 80% over a defined time period).

Engineers at this level should be learning how to identify issues and learn from them to improve their skills. They should usually be aiming to progress to software engineer within 2 years.

Anti-patterns: Poor code quality. Disregards team process. Not self-motivated with their tasks or learning. Constantly off on tangents or lost in the weeds. Inclined to blame or complain. Helplessness when stuck.

Software engineer



A **software engineer** has a broad and rounded technical grounding, and is trusted to produce quality code for more sizeable tasks on a project. They are now very capable of work in a single technology or platform.

They can break down larger features into tasks with minimal guidance and execute on them independently. They can compose software of medium complexity using modern and collaborative development methods, and own small-to-medium features all the way from technical design to launch.

They can work both independently on smaller projects, and within teams in the context of larger more parallelised efforts. They are self-sufficient and make steady progress on tasks, knowing when to ask for help and how to proactively get unblocked.

They consistently deliver correct and clean high-quality code. Material feedback may still come up during code review, but usually they can find the solution on their own once a problem is highlighted. They observe and follow established patterns and approaches within existing code bases with ease.

They can investigate and address bugs and crashes of increasing complexity/obscurity, and write tests as well as fix the presenting issue.They design and write code that is well organised and maintainable. They are working on developing their technical design skills and engineering judgment, and actively participate in design discussions and seek feedback on their own designs.

When given a task with unclear or conflicting requirements they reliably push for clarification, and ensure that all assumptions are vetted before work starts to reduce the need for re-work.

They should be able to communicate confidently internally, and be increasingly comfortable communicating to and interfacing directly with clients both in meetings and over Slack/E-Mail. They are gaining confidence in sharing their knowledge to help others on the team.

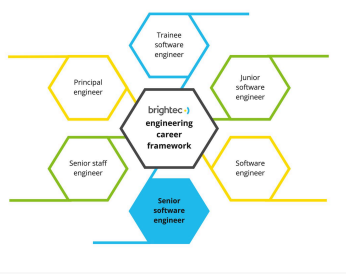
Their work is fully billable to clients.

Engineers at this level should usually be focussed on achieving mastery of their technology area and/or of one or more client projects. They *may* begin to take an interest in a second technology area, but should prioritise depth over breadth initially.

A software engineer enters this level capable of taking well-defined tasks and completing them to a high standard with some supervision from more senior team members. Progress through this level is focused on taking tasks of increasing complexity, scope and importance and completing them with very high quality and a lessening need for oversight. Engineers at this level should be focused on becoming great engineers, learning how to set high quality bars for their work without sacrificing productivity.

Anti-patterns: Dabbling in too many areas without depth / jack-of-all-trades-master-of-none. Unable to stay learning 'on the job'. Needs frequent supervision and oversight. Poor judgement without corresponding growth and learning. Perfectionism in place of sustained results delivery. Disregard for budget constraints.

Senior software engineer



A **senior software engineer** has a strong and comprehensive command of the languages they use and can deliver complex and/or large tasks end-to-end following good practice, with minimal oversight. They also likely have established end-to-end expertise with one or more client projects over a longer time horizon. They are competent in reactively assisting and guiding other engineers in the more advanced/obscure features of those languages and projects. They may be seen as a technical authority in some smaller technology or project areas.

They may have deep expertise in a single platform, or be making considerable contributions across multiple platforms. They can take on sizeable features single-handedly, and can start new projects from scratch with some oversight, guidance and assistance.

They have solid technical design abilities, and can make framework/dependency selections and evaluate the tradeoffs of different approaches. They have good judgement, and can articulate their decisions and arguments clearly both internally and externally with clients. Where an approach isn't obvious, they will build consensus / seek agreement up-front to avoid abortive work. They have a great awareness of programming best practices and apply them consistently.

They get a lot done, despite any roadblocks that arise, and drive constant progress. They can attack difficult problems and debug complex issues and crashes effectively without flailing. They can take on complex user stories and break them down into sub-tasks for one or more people to execute against. They can write clear tasks and requirements for others. They always have a high-level plan before jumping into implementation work. They have increasingly tuned estimation abilities, and budget awareness when working on client projects. They can identify and communicate risks. They make test plans for large work pieces.

They routinely support other less experienced team members through day-to-day activities such as code review, pair programming and group discussions. They are growing more experienced at offering material feedback at the right time in the right way.

They will usually begin to take on some project work where they are the most senior engineer on the team, and will carry out tech-lead responsibilities such as assisting less experienced engineers with effective task management and progress reporting, and collaborating with non-engineers on backlog curation and planning. They are likely starting to participate within a tech council.

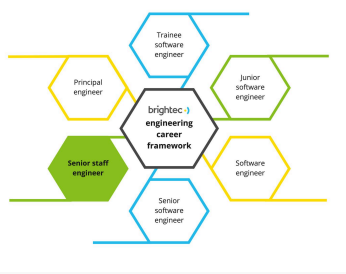
They care about the team as a whole, and will contribute to common code bases and team processes and standards. They have empathy for the users of their software and the commercial businesses of our clients, and use that understanding to influence task prioritisation and trade-offs.

They may participate in or shadow the hiring process for new engineers.

To keep growing, a senior software engineer will usually push their skills beyond one technology area (e.g. to Android and iOS, or iOS and Web), and may take on significant tasks and responsibilities in a secondary area such as research, documentation, prototyping, product roadmaps, visual design, etc. They set measurable self-development goals, and meet them.

Anti-patterns: Fails to identify or communicate big risks or roadblocks. Continually underestimates timelines. Solutions are overly complicated. Spends time on work items that don't materially matter to the users or our clients. Isolationist / can't collaborate and guide more junior engineers. Us-vs-them attitudes.

Senior staff engineer



A **senior staff engineer** exhibits deep, substantial expertise in multiple technologies or platforms, working across them as project needs require. They lead small teams to execute on multi-month features. They can start new projects from scratch with minimal guidance and assistance, involving and including less experienced engineers.

They are relentlessly productive. Given a nebulous project or an open-ended problem, they will appropriately define the requirements, scope it, find a solution, implement and launch that solution. They help define project roadmaps. They are business-aware, and great advocates both for our clients and for Brightec. They help groups of engineers to deliver complex projects with their project management ability. They take long projects or complex groups of user stories and break this work down into milestones to avoid large monolithic deliverables or big-bang releases, creating entire project plans/backlogs.

They are "a safe pair of hands", known for drama-free launches, and they own the technical quality and testing of their projects and releases. They anticipate technical issues at the product level and make architectural and design decisions to avoid them. They avoid accumulating unchecked technical debt. They consistently reduce the complexity of projects and processes in order to get more done with less work, articulating the benefits and winning buy-in and budget from clients to facilitate this.

They are proactive in assisting other engineers, through code review, technical mentoring, team knowledge building, getting unstuck, etc. Their breadth of knowledge makes them exceptional technical problem solvers and forward planners. They are proactively budget aware, and a skilled and self-aware estimator. They can also empathise with and estimate for more junior team members. They are trusted to generate budget estimates for large existing client projects and new business proposals, and provide other technical advice that the rest of the business will rely on.

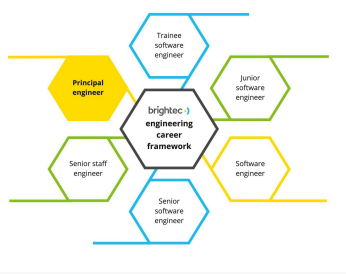
They show strong leadership either through team leadership (extensive tech leading, mentoring of other team members, etc) or exceptional individual contributions (championing strategic areas, defining new team standards, pioneering novel approaches, etc). They balance leadership with individual contribution. They are likely key participators on multiple tech councils.

They are viewed as the go-to expert for some client projects and/or entire technology areas, and are deeply involved in deciding the standards and processes for the entire engineering team. They keep up with and share industry news, research and propose new technologies, and have a deep and opinionated understanding of our chosen architectures and frameworks. They know how to not only identify technical problems and propose solutions, but are also able to get team buy-in for their solutions and oversee projects to make these solutions come to life.

They will participate in the hiring process for new engineers and provide recommendations and feedback from interviews.

Anti-patterns: Doesn't delegate / control freak. Cannot empathise with more junior engineers. Jumps into execution without careful consideration. Doesn’t follow new technologies or industry trends. Takes no interest in team and company strategy. Doesn't sweat the details. Fails to raise awareness of projects at risk or people-problems.

Principal engineer



A **principal engineer** plays a leading role in helping to set the direction and goals for the entire engineering team, and in defining and advocating for engineering excellence. They are a deeply trusted and relied upon strategic resource to the whole business, by virtue of their deep and transferable knowledge across multiple technologies, domains and client projects.

Over a long time horizon they have significantly contributed to the successful inception and launch of multiple client projects, and ultimately to the significant commercial advancement of many of our customers. They carry key responsibility for the technical and business output of our company. They are trusted to provide client-facing budget estimations for large projects and to take on commercial responsibility for the viability and success of entire client projects. They are deeply commercially aware, and coach other engineers in this mindset.

Their strategic oversight and directional input keeps the company stable and on-track, without letting it stagnate or fall behind. They balance the need to change, grow and improve with continuity, consistency and predictable results for the company and our clients. They are technical leaders with wisdom to judge which of the latest tools and industry trends are worth exploring and adopting. They exert technical influence across many of our projects, platforms and technology areas, including on the technology strategy with a multi-year timeframe. They push the whole organisation forwards technically, through both personal modelling and explicit team input and guidance. They carry key responsibility for the health, morale and output of our engineering team.

They continue to act in a very hands-on role, and to make personal individual contributions to our most novel and difficult opportunities. The problems and projects that the principal engineer is tasked with solving may be very open-ended even to the company leadership or the client who presented the problem. Most Brightec engineers would not be able to own and solve the problems that this person is attacking.

A principal engineer is sought-after for technical guidance, input and validation right across the team. They inspire other engineers and are seen as a role model and mentor to every technical member of the team. They have a track record of anticipating technical problems that will fall out of major products and designing solutions to overcome those problems. They are deeply knowledgeable across a majority of our client projects and are a technical owner of large parts of our active code bases. They help drive multiple tech councils.

A principal engineer can be relied upon not to make themselves indispensable on a project. They can consistently join a new engagement or project early on, define and scope the technical approach, architect an outline solution, communicate it to the client and to a project team, and help the team to execute while freeing themselves up to move on to another endeavour. They create architectures that enables many potential futures without knowing exactly what the future is.

They will be a key voice in the hiring decision for new engineers.

Anti-patterns: Frivolous or flippant with new opportunities and projects. Too eager to chase the newest "shiniest" technologies. Has a "pet" agenda. Focusses only on engineering / doesn't work closely with other business leaders. Doesn't have time for more junior team members any more / condescending or lacks empathy.