SoD 2024 Skill Trees

These skill trees outline the skills developers need to pick up on their development journey at Oppia.

These trees are works in progress and subject to change. The Web tree is somewhat more developed than the Android tree. SoD applicants are welcome to use these draft trees to help write their SOIs, but they should acknowledge that the trees are not yet final.

Bolded skills or skill headings are tentative tutorials. Grayed-out skills are ones we expect developers to go elsewhere to learn (though we will point them to resources we think are good) or have already learned. Skills in red are advanced enough that they do not need tutorials. They will likely only be used by established Oppia devs who can learn what they need from our reference documentation. Writing and maintaining tutorials for these skills is likely more work than it's worth. Skills in **bold** are ones we plan to write or adapt tutorials for. Some skills are not bolded because we have pages for them already that we don't anticipate changing.

Oppia Web

The tree illustrates the possible roles, teams, and subteams at Oppia, where a user progresses down the table to more advanced roles over time and chooses teams and subteams to focus on.

The skills for each position are listed on the right. The skills are additive, so a collaborator needs all the skills of a new contributor, and a collaborator on the linter subteam also needs the skills listed for the more general roles "dev workflow" and "collaborator". A member will have the skills of one collaborator position (e.g. the dev workflow dependencies subteam) but likely not all collaborator positions.

Role	Team	Subteam	Skills

New Contributor			• Setup
			  ◦ **Getting started guide**
			  ▪ What wiki pages contributors should read and when
			  ◦ **Installing Oppia**
			  ▪ Command-line basics.
			  ▪ Setting up your IDE
			  ◦ **How to access Oppia webpages**
			  ▪ Include Glossary of terms page

			■ Move user docs to oppia.github.io ● **Get assigned your first issue** ○ Understand starter-level issues ○ Ask for help ○ Navigate the codebase ○ Trace how data and execution flow through code ○ Figure out what code to change ○ Clearly propose a solution to a starter-level issue ● **Solve your first starter issue** ○ Follow test-driven development ○ Write clear, robust code ○ Run code quality checks (e.g. linters, tests, type checking), understand output, and resolve failures ○ Debug, including how to efficiently and methodically identify the root cause of a bug. How to write a debugging doc. ○ Use git from the command line ● **Create a good PR** ○ Check and prove that a PR is correct (designing the user journey to show, showing log outputs, making a screen recording) ○ Respond to and address review comments ○ Debug CI failures
	Backend		● Python ● (sometimes) Bash ● (sometimes) Apache Beam
	Frontend		● Typescript ● Angular ● HTML ● CSS
Collaborator			● **Write and get feedback on TDDs** ● **How to test a feature thoroughly** ○ For example, release QA

			• Write, run, and fix E2E tests • **Write, run, and fix acceptance tests** • **Figure out how a feature works end-to-end**
	Dev Workflow		• **Write types for Python code** • **Read, write, and modify GitHub Actions workflows and actions** • **Read, write, and modify the pre-commit and pre-submit checks**
		Linter	• **Write and debug lint checks** • Modify the lint check runners
		Backend Tests	• **Write, run, and fix backend unit tests** • Modify the backend test runner • Modify the coverage check infrastructure
		Frontend Tests	• **Write, run, and fix frontend unit tests.** • Modify the frontend test runner • Modify the coverage check infrastructure
		E2E and Acceptance Tests	• **Debug CI flakes** • Modify the E2E test runner • Modify the acceptance test runner

		Dependencies	• Modify how Oppia's dependency management systems (dependencies.json, pip, package.json, miscellaneous) work
			• **Add, remove, and upgrade dependencies**
			  ○ Vet a third-party dependency for security and maintenance
			  ○ Read changelogs and check for breaking changes

		Release	• Modify how releases are built and deployed • Organize release testing • Serve as a release coordinator
	LACE and CD		
		Backend	• **Fix a backend bug**

		Work	• **Write, run, and fix backend unit tests** • **Write types for Python code** • **Fix a bug involving emails**

		Beam Jobs	• **Write your first beam job**
			  ○ Includes writing the testing doc
			• **Design good beam jobs**
			  ○ Includes robustness, safe and noisy failures, and adding sufficient logging for debugging

		Frontend Work	• **Fix a frontend bug** • **Write, run, and fix frontend unit tests.** • Make a simple UI change • **Write types for Typescript code**
	Server Errors		• **Figure out reproduction steps from server error logs** • **Fix server errors that cannot be reproduced**
	All Teams	Project Organizer	• **Triage issues** • **Keep GitHub Project board up to date** • **Answer GitHub Discussions**
Member			• When to merge PRs • When to rerun failing CI checks • Apply the revert and regression policy to PRs that introduce regressions (including flakes).

Codeowner			• **How to do a good code review**
			  ○ Review code for quality, correctness, and security
			  ○ Leave clear review comments

Grouping into tasks of approximately equal size (small is roughly half of medium, and large is roughly 50% larger than medium):

- **Task**
  - **[Large] Fix a backend bug**
  - **[Medium] Write types for Python code**
- **Task**
  - **[Large] Fix a frontend bug**
  - **[Medium] Write, run, and fix acceptance tests**
- **Task**

    - **[Large] Figure out how a feature works end-to-end.**
    - **[Medium] Write types for Typescript code**
- **Task**
    - **[Large] Write and debug lint checks**
    - **[Medium] Read, write, and modify the pre-commit and pre-submit checks**
- **Task**
    - **[Large] Fix a bug involving emails**
    - **[Small] Keep GitHub Project board up to date**
    - **[Small] Solve your first starter issue**
- **Task**
    - **[Medium] Write your first beam job**
    - **[Small] Answer GitHub Discussions**
    - **[Small] Create a good PR**
    - **[Small] Triage issues**
    - **[Small] How to test a feature thoroughly**
- **Task**
    - **[Medium] Design good beam jobs**
    - **[Medium] How to do a good code review**
    - **[Small] Get assigned your first issue**
    - **[Small] How to access Oppia webpages**
- **Task**
    - **[Large] Debug CI flakes**
    - **[Medium] Figure out reproduction steps from server error logs**
- **Task**
    - **[Medium] Write, run, and fix backend unit tests**
    - **[Medium] Fix server errors that cannot be reproduced**
    - **[Small] Write and get feedback on TDDs**
    - **[Small] Getting started guide**
- **Task**
    - **[Medium] Write, run, and fix frontend unit tests.**
    - **[Medium] Add, remove, and upgrade dependencies**
    - **[Small] Read, write, and modify GitHub Actions workflows and actions**
    - **[Small] Installing Oppia**
    - **[Medium] Re-organize Sidebar. A draft sidebar is in** Draft Wiki Sidebars
        - In “Guidelines for developers with write access to oppia/oppia” replace oppia/oppia with “the Oppia repository”
        - Fold https://github.com/oppia/oppia/wiki/Release-accessibility-checklist into “Coding and testing for accessibility”
        - Move https://github.com/oppia/oppia/wiki/Common-pull-request-workflows into “Make a Pull Request” and call it “Rules for making PRs”. The reference material from “Common pull request workflows” can move to a new page in the dev workflow section.
        - Fold the “Formatters” and “Bytes and string handling in Python 3” pages into the coding style guide.

        - Fold “Interpreting GitHubActions Results” into “If CI checks fail on your PR”

Oppia Android

Initially written by Ben Henning.

The tree illustrates the possible roles on the Android team, where a user progresses down the table to more advanced roles over time. The skills for each position are listed on the right. Some skills are required, while for other priority levels, contributors will pick up skills as needed.

Role	**Priority**	**Skills**

New Contributor	Required	• Setup
		  ○ Read and follow instructions in documentation.
		  ○ Create GitHub and Google accounts
		  ○ Command-line basics.
		  ○ Kotlin basics, from a Java development perspective
		• **Get assigned your first issue**
		  ○ Understand starter-level issues
		  ○ Ask for help
		  ○ Navigate the codebase
		  ○ Trace how data and execution flow through code
		  ○ Figure out what code to change
		  ○ Clearly propose a solution to a starter-level issue
		• **Solve your first starter issue**
		  ○ Follow test-focused development (e.g. all new and changed code should have tests)
		  ■ Write excellent unit tests
		  ○ Write clear, robust code
		  ○ Run code quality checks (e.g. linters, tests, type checking), understand output, and resolve failures
		  ○ Debug, including how to efficiently and methodically identify the root cause of a bug. How to write a debugging doc.
		  ○ Use git from the command line
		  ○ Use bazel from the command line
		• **Your first PR**

		○ Check and prove that a PR is correct (designing the user journey to show, showing log outputs, making a screen recording) ○ Filling out the PR template ○ Respond to and address review comments (particularly important) ○ Debug CI failures
Established Contributor	Required	● Basics of Android development ● How to use Android Studio effectively ● Effective debugging in Android ● **Write, run, and fix unit tests** ● **Understanding & using Dagger (basics)**
	Core Android domain-specific knowledge to pick up on over time	● Adding a new Android dependency ○ Prereqs: Android Studio, Kotlin, Bazel ● Kotlin coroutines & **Oppia DataProviders** ○ Prereqs: Android Basics, Kotlin ● **Creating a new controller in Oppia Android** ○ Prereq: coroutines & DataProviders ● **Read and translate mocks into code using view/data binding concepts** ○ Prereqs: Android Studio, Kotlin ● **Creating a new UI in Oppia Android (includes the boilerplate and high-level UI dataflow, mention livedata stuff)** ○ Prereqs: Controllers, Mocks
	Core general software development knowledge to pick up on over time	● Creating a new controller and a new UI ○ Prereqs: Android Studio, Kotlin, Navigate the codebase ● **How to root cause a bug (an end-to-end example)** ○ Prereq: navigating the codebase ● **How to root source a bug (e.g. find the offending commit)** ○ Prereq: Git ● **Advanced testing guide: how to detect, and avoid, flakes in Robolectric & Espresso** ○ Prereqs: writing excellent unit tests, data providers ● **How to communicate technical concepts effectively (e.g. how to ask better questions and give sufficient context with them)**

		• **Writing a TDD: an end-to-end example (video & sample TDD)** ○ Prereqs: navigating the codebase, effective technical communication
Advanced Contributor		• Creating a new static analysis check

Tasks of approximately equal size:

- Task
  - [Large] Creating a new UI in Oppia Android (includes the boilerplate and high-level UI dataflow, livedata, databinding)
  - [Small] How to communicate technical concepts effectively
- Task
  - [Large] Creating a new controller and a new UI
  - [Small] How to create an effective pull request that anyone on the team can easily understand
- Task
  - [Large] How to root cause a bug (an end-to-end example)
  - [Small] Writing a TDD: an end-to-end example (video & sample TDD)
- Task
  - [Large] Advanced testing guide: how to detect, and avoid, flakes in Robolectric & Espresso
- Task
  - [Medium] Basics of Android development
  - [Small] Get assigned your first issue
- Task
  - [Medium] Effective debugging in Android
  - [Small] Solve your first starter issue
- Task
  - [Medium] Understanding & using Dagger (basics)
  - [Small] Your first PR
- Task
  - [Medium] Adding a new Android dependency
  - [Small] How to use Android Studio effectively
- Task
  - [Medium] Kotlin coroutines & Oppia DataProviders
  - [Medium] Creating a new controller in Oppia Android
- Task
  - [Medium] Read and translate mocks into code using view/data binding concepts
  - [Medium] How to root source a bug (e.g. find the offending commit)