CodeMeal CLI — User Manual

Version 1.0.7 · codemeal.in

Solve Data Structures & Algorithms and SQL problems from your own terminal, in your own editor. Browse problems, pull down a starter file, run the tests, and submit - without opening a browser.

Contents

1 **Installing**

2 **Your first session**

3 **Command reference - Account - Problems - SQL problems - Revision queue - Notes - Patterns - Mock interviews - Updating the CLI**

4 **The revision loop**

5 **Mock interviews in depth**

6 **Using it in scripts**

7 **Where your files live**

8 **Configuration**

9 **Troubleshooting**

10 **Quick reference**

1 Installing

The CLI is a single executable. There is no runtime to install and no dependencies to manage — download it, put it somewhere on your `PATH`, and it runs.

Download the archive for your operating system from **codemeal.in/downloads**. The page detects your platform and offers the right build first.

macOS

```
tar -xzf ~/Downloads/codemeal_*_darwin_*.tar.gz
sudo mv codemeal /usr/local/bin/
codemeal --version
```

Macs from 2020 onward are almost always Apple Silicon. If you are not sure, check **Apple menu → About This Mac**.

Linux

```
tar -xzf ~/Downloads/codemeal_*_linux_*.tar.gz
sudo mv codemeal /usr/local/bin/
codemeal --version
```

Windows

Unzip the archive, put `codemeal.exe` somewhere permanent — for example `C:\Tools\codemeal` — and add that folder to your `PATH`.

```
cd C:\Tools\codemeal
codemeal.exe --version
```

First-run security warnings

The released binaries are not signed with an Apple Developer ID or a Windows Authenticode certificate, so both systems will say they cannot identify the publisher. **Neither warning is a malware detection.** They mean "unsigned", not "unsafe".

macOS — "Apple could not verify 'codemeal' is free of malware." macOS attaches a quarantine flag to anything downloaded through a browser. Remove it:

```
xattr -d com.apple.quarantine /usr/local/bin/codemeal
```

Or allow it once under **System Settings → Privacy & Security → Open Anyway**.

Windows — "Windows protected your PC." Click **More info**, then **Run anyway**. SmartScreen shows this for executables it has not yet seen signed by a known publisher.

Linux has no equivalent check.

Verifying your download

Every archive's SHA-256 is listed on the downloads page under **SHA-256 checksums**. Hash the file you downloaded and compare:

```
shasum -a 256 codemeal_<version>_<os>_<arch>.tar.gz    # macOS / Linux
certutil -hashfile codemeal_<version>_windows_amd64.zip SHA256   # Windows
```

If the two match, what you have is byte-for-byte what the build produced.

2 Your first session

A complete run through one problem.

Sign in. This opens your browser for Google or GitHub sign-in and stores the credential in your operating system's keychain.

```
codemeal login
codemeal whoami          # confirm who you are signed in as
```

Find something to solve. The list shows each problem's ID, title, difficulty and acceptance rate.

```
codemeal problem list
codemeal problem list --title "two sum" --difficulty Easy
codemeal problem 42                      # the details of one problem
```

Start it. This prints the problem statement and examples, writes a starter file to `~/.codemeal/solutions/42/`, and opens it in your editor.

```
codemeal problem solve 42
```

You will be asked to choose a language — **Java** or **Python**. The editor is `$VISUAL` or `$EDITOR` if either is set, then VS Code, then your system default. Pass `--no-open` to create the file without opening anything.

Run and submit. The public test cases run first and show you exactly what failed. Only if they all pass does it offer to submit against the private ones.

```
codemeal problem submit 42
```

When a submission is accepted you are asked how it felt — `struggled`, `okay` or `easy`. That single answer schedules when the problem comes back (section 4).

Keep a note on what the trick was, and check where you stand.

```
codemeal note edit 42
codemeal problem stats
codemeal submission
```

Prefer SQL? SQL problems have their own catalog and workspace. The flow is the same: find a problem, open its starter query, edit it, then submit it.

```
codemeal sql problem list
codemeal sql problem solve 17
codemeal sql problem submit 17
codemeal sql problem stats
codemeal sql submission
```

Run `codemeal --help`, or `codemeal <command> --help`, at any point.

3 Command reference

Every command is `codemeal <command>`. Arguments in `<angle brackets>` are required; those in `[square brackets]` are optional.

Account

`codemeal login`

Sign in via Google or GitHub. Opens your browser and waits for the callback. The credential is written to your OS keychain, never to a plain file.

`codemeal logout`

Remove the saved credential from the keychain.

`codemeal whoami`

Show the account you are currently signed in as.

Problems

`codemeal problem <id>`

Show one problem: its title, difficulty, section, acceptance rate, tags and tracks.

`codemeal problem list`

List problems.

FLAG	SHORT	DEFAULT	DESCRIPTION
`--skip`	`-s`	`0`	Number of problems to skip
`--limit`	`-l`	`30`	Number of problems to fetch
`--title`	`-t`	—	Search problems by title
`--difficulty`	`-d`	—	Filter by difficulty

```
codemeal problem list --difficulty Medium --limit 50
codemeal problem list --title "binary search"
codemeal problem list --skip 30 --limit 30      # the second page
```

`codemeal problem solve <id>`

Start — or resume — a solution. Prints the problem statement and examples, writes a starter file under `~/.codemeal/solutions/<id>/`, and opens it in your editor. Running it again on a problem you have already started reopens the file you were working on rather than overwriting it.

FLAG	DESCRIPTION
`--no-open`	Create the solution file without opening an editor

`codemeal problem submit <id>`

Run your solution against the public test cases and, on success, offer to submit it against the private ones.

FLAG	DESCRIPTION
`--confidence`	How it felt: `struggled`, `okay` or `easy`. Prompts if omitted
`--no-revision`	Don't schedule the problem for revision

```
codemeal problem submit 42
codemeal problem submit 42 --confidence okay     # skip the question
codemeal problem submit 42 --no-revision         # don't add it to the queue
```

`--confidence` also accepts a few synonyms: `struggle`, `hard` and `bad` all mean `struggled`; `ok`, `fine` and `medium` mean `okay`; `good` means `easy`.

`codemeal problem stats`

Your solved and total counts, broken down by difficulty, track, section and tag. The same numbers as the website.

`codemeal submission`

List your recent submissions.

SQL problems

SQL problems use a separate catalog and workspace, so they do not get mixed with Java and Python problems. The problem statement identifies the SQL dialect you should use; the current catalog uses PostgreSQL.

`codemeal sql problem <id>`

Show one SQL problem, including its description, difficulty, dialect, tables, examples and constraints.

`codemeal sql problem list`

List SQL problems. Each row shows the ID, title, difficulty, dialect and whether you have solved it.

FLAG	SHORT	DEFAULT	DESCRIPTION
`--skip`	`-s`	`0`	Number of SQL problems to skip
`--limit`	`-l`	`30`	Number to fetch, from `1` through `100`
`--search`	`-q`	-	Search titles and tags
`--difficulty`	`-d`	-	Filter by difficulty
`--solved`		-	Use `true` for solved or `false` for open

```
codemeal sql problem list --difficulty Easy --limit 25
codemeal sql problem list --search "group by"
codemeal sql problem list --solved=false
codemeal sql problem list --skip 30 --limit 30
```

`codemeal sql problem solve <id>`

Start or resume a query. The command prints the full statement, tables and examples, creates `~/.codemeal/sql/solutions/<id>/solution.sql`, and opens it in your editor. Running it again reopens your existing query without replacing it.

FLAG	DESCRIPTION
`--no-open`	Create the SQL file without opening an editor

`codemeal sql problem submit <id>`

Run the query against the public examples first. For each example, the CLI prints the expected and actual result tables. If every public example passes, it asks before running the private test cases.

FLAG	SHORT	DESCRIPTION
`--yes`	`-y`	Skip confirmation before the private test cases

```
codemeal sql problem submit 17
codemeal sql problem submit 17 --yes      # suitable for a script
```

The terminal displays at most 50 rows from a result table so a large result does not bury the useful output. All rows are still checked by the platform. An accepted submission removes the local query and metadata. A failed or cancelled submission keeps them so you can continue from the same file.

`codemeal sql problem stats`

Show your SQL solved and total counts, broken down by difficulty, dialect and tag.

`codemeal sql submission`

List your recent SQL submissions separately from coding submissions.

Revision queue

Solving a problem once is not learning it. The revision queue brings problems back at widening intervals based on how hard you found them — see section 4.

`codemeal revision`

The problems due to be solved again, most overdue first.

FLAG	DEFAULT	DESCRIPTION
`--limit`	`50`	Maximum number of due problems to list

`codemeal revision next`

The single problem most worth redoing right now.

FLAG	SHORT	DESCRIPTION
`--quiet`	`-q`	Print nothing but the problem ID

With `--quiet` this composes with other commands. When nothing is due it exits non-zero rather than printing an empty line, so a chained command stops instead of being handed an empty ID.

```
codemeal problem solve "$(codemeal revision next --quiet)"
```

`codemeal revision done <id>`

Record that you re-solved a problem, and reschedule it.

FLAG	DESCRIPTION
`--confidence`	How it felt: `struggled`, `okay` or `easy`. Prompts if omitted

`codemeal revision retire <id>`

Stop scheduling a problem. If a problem has stopped teaching you anything, retiring it takes it out of rotation. Recording a revision for it later puts it straight back.

Notes

Your own private notes on a problem. Notes are capped at 4,000 characters — the same limit the web editor shows — and live on the server, so they follow you between machines.

`codemeal note <id>`

Print your note on a problem. Writes nothing but the note to standard output, so it redirects cleanly.

```
codemeal note 42 > two-sum.md
```

`codemeal note edit <id>`

Write or revise the note in your editor.

`codemeal note set <id>`

Replace the note from a file or from standard input. No editor, no prompts — the scriptable half of `note edit`.

FLAG	SHORT	DESCRIPTION
`--file`	`-f`	Read the new note from this file
`--force`		Overwrite an existing note without asking

```
cat notes.md | codemeal note set 42
codemeal note set 42 --file notes.md
codemeal note 42 | sed 's/foo/bar/' | codemeal note set 42
```

Because it overwrites whatever was there, it asks for confirmation when a note already exists. `--force` skips that, and it is skipped automatically when there is no terminal to ask at — in a script or a pipeline.

`codemeal note delete <id>`

Delete your note on a problem.

FLAG	DESCRIPTION
`--force`	Delete without asking

Patterns

The recurring shapes behind the problems — sliding window, two pointers, monotonic stack, and the rest.

`codemeal pattern`

List the patterns.

FLAG	DEFAULT	DESCRIPTION
`--search`	—	Filter by name or summary
`--skip`	`0`	Number of patterns to skip
`--limit`	`100`	Maximum number to list

`codemeal pattern show <slug>`

One pattern in full: when to reach for it, and the problems that use it.

FLAG	DESCRIPTION
`--template`	Also print the starting template

```
codemeal pattern show sliding-window
codemeal pattern show sliding-window --template
```

`codemeal pattern template <slug>`

Print just the pattern's starting template, and nothing else — so it redirects straight into a file.

```
codemeal pattern template sliding-window > window.py
```

Exits non-zero when there is no template to print.

`codemeal pattern for <id>`

Which patterns a problem belongs to.

Mock interviews

`codemeal interview`

Your interview requests and where each one stands — waiting, scheduled, or needing a new window. Once a slot is booked it shows the time in your local zone and the Google Meet link on its own line.

`codemeal interview request`

Ask for a mock interview. Asks twelve short questions, most of which have sensible defaults — your email defaults to the account you are signed in as, and the time zone to this machine's. Every answer is also a flag, so the whole form can be filled from the command line.

FLAG	DESCRIPTION
`--company`	Company you are preparing for
`--role`	Role you are interviewing for
`--type`	Interview type: `technical` or `coding`
`--language`	Programming language (required for a coding interview)
`--email`	Contact email (defaults to your account email)
`--phone`	Contact phone number (optional)
`--from`	Earliest date you are free (`YYYY-MM-DD`)
`--to`	Latest date you are free (`YYYY-MM-DD`)
`--start-time`	Earliest time of day you are free (`HH:MM`)
`--end-time`	Latest time of day you are free (`HH:MM`)
`--timezone`	IANA time zone (defaults to this machine's)
`--duration`	Minutes: `30` , `45` , `60` or `90`
`--notes`	Anything the interviewer should know (optional)

`codemeal interview resubmit <id>`

Offer a new window on a request you were asked to reschedule.

Updating the CLI

`codemeal update [version]`

Replace this binary with another build of it. Needs no login — release downloads are public.

FLAG	SHORT	DESCRIPTION
`--check`		Report whether an update is available; exits non-zero if so
`--list`		List the versions available
`--force`		Install even when it is the version already running, or a local build
`--yes`	`-y`	Don't ask for confirmation
`--allow-unverified`		Install even when the API reports no checksum for the archive

```
codemeal update              # the newest release
codemeal update 1.0.2        # exactly that version, downgrade included
codemeal update --check      # is there an update?
codemeal update --list       # every version available
```

The archive is checked against the SHA-256 the API reports **before** anything is installed, so a truncated or altered download leaves your working binary exactly where it was. The build being replaced is kept beside it as `codemeal.old`, which is the quickest way back from a release that turns out badly — or pin the old version with `codemeal update <version>` and it comes back properly.

Four things it deliberately will not do:

- **Update a copy installed by a package manager.** If the binary lives under Homebrew, Scoop or Chocolatey, it says so and gives you their upgrade command instead. Replacing the file underneath them leaves their records wrong.
- **Replace a build you made yourself.** A locally built binary reports its version as `dev`, and overwriting it would throw away whatever you were working on. `--force` if you really mean it.
- **Install something it cannot verify.** If no checksum is published for an archive, it refuses until you pass `--allow-unverified`.
- **Ask for your password.** If the install directory belongs to root — `/usr/local/bin` usually does — it tells you to re-run under `sudo` rather than quietly elevating itself.

4 The revision loop

When a submission is accepted, `codemeal problem submit` asks how it felt. That one answer is the entire scheduler:

YOU SAID	IT COMES BACK IN	AND AFTER THAT
`struggled`	1 day	1 day again
`okay`	3 days	×2 each time
`easy`	7 days	×3 each time

So a practice session looks like this:

```
codemeal revision                                    # what is due
codemeal problem solve "$(codemeal revision next -q)"  # the most overdue one
codemeal problem submit 42                           # answer, and it reschedules
```

Answer honestly. Saying `easy` to something you half-remembered pushes it three weeks out, and nothing else will bring it back. Pass `--confidence easy` to skip the question when you already know, or `--no-revision` to keep a problem out of the queue entirely.

5 Mock interviews in depth

You say when you are free; a reviewer picks a slot inside that window, and the backend books a Google Meet and sends a calendar invite.

```
codemeal interview request      # the form, with defaults
codemeal interview              # waiting / scheduled / needs a new window
```

Fully scripted, it looks like this:

```
codemeal interview request \
  --company Zoho --role "Backend Developer" --type coding --language Java \
  --from 2026-11-10 --to 2026-11-17 \
  --start-time 19:00 --end-time 21:00 --duration 60
```

The rules, all checked before anything is sent:

- The date range starts today or later, and spans 60 days or fewer.
- The end time is after the start time.
- The duration is 30, 45, 60 or 90 minutes.
- The daily window is long enough to fit the duration.
- A **coding** interview needs a language; a **technical** one ignores it.

If a reviewer asks for a different window instead of approving, `codemeal interview` says so, shows their comment, and tells you to run `codemeal interview resubmit <id>`.

6 Using it in scripts

The commands that exist to be piped print nothing but their payload, so they behave properly in a shell:

```
codemeal note 42 > two-sum.md                                # keep a copy of a note
codemeal pattern template sliding-window > window.py         # start from a template
codemeal problem solve "$(codemeal revision next --quiet)"  # do what is due
cat draft.md | codemeal note set 42                          # write a note from a
file
codemeal sql problem submit 17 --yes                         # confirm private SQL
tests
```

`revision next --quiet` and `pattern template` exit **non-zero** when there is nothing to print, so a chained command stops rather than being handed an empty value.

Interactive confirmations are skipped automatically when there is no terminal to ask at, so `note set` and `note delete` work unattended without needing `--force` in a pipeline.

7 Where your files live

PATH	WHAT
`~/.codemeal/solutions/<id>/`	Your in-progress coding solution
`~/.codemeal/problems/<id>.json`	Language and file metadata for a coding problem
`~/.codemeal/sql/solutions/<id>/`	Your in-progress SQL query
`~/.codemeal/sql/problems/<id>.json`	File metadata for an SQL problem
`~/.codemeal/notes/<id>.md`	Scratch buffer while you edit a note
OS credential store	Your login tokens (never written to a plain file)

Notes themselves live on the server, not in that file — it is only what your editor opens. It is removed once the note is saved, and **kept** if saving fails or you decline, so an unsaved draft is never lost: the next `codemeal note edit` picks it back up.

Solution files are removed automatically once a submission is accepted, or if you decline to submit. A failed submission keeps them, so you can fix and retry.

SQL query files are removed only after an accepted submission. Cancelling the private test run or receiving a failed result keeps the query and its metadata.

8 Configuration

Where your credentials are kept

Login stores two credentials in your operating system's native credential store — Keychain on macOS, Credential Manager on Windows, Secret Service on Linux. Neither is ever written to a plain file.

- A short-lived **access token**, used for each request.
- A 30-day **refresh token**.

When the access token expires, the next command renews it transparently and retries. You should not notice. You will only be asked to sign in again if the refresh token expires or is revoked — and the CLI says so plainly rather than showing an API error.

Pointing at a different backend

The CLI talks to `https://codemeal.in/api`, baked in at build time. To point a locally built binary somewhere else:

```
export CODEMEAL_API_BASE_URL=http://localhost:5100/api
```

9 Troubleshooting

"You are not logged in." Run `codemeal login`.

"Your session has expired." Your refresh token has expired or been revoked. Run `codemeal login` again. This should be rare — sessions normally renew themselves.

`command not found: codemeal` The binary is not on your `PATH`. Check where you moved it, and that the containing folder is listed in `echo $PATH` (macOS/Linux) or your system environment variables (Windows).

macOS refuses to run it. See first-run security warnings — `xattr -d com.apple.quarantine /usr/local/bin/codemeal`.

`codemeal update` says it will not replace a local build. A binary you compiled yourself reports its version as `dev`. Pass `--force` if you really want to overwrite it.

`codemeal update` says to re-run under `sudo`. The install directory belongs to root. Re-run the same command with `sudo` rather than changing the directory's permissions.

"No SQL solution found." Run `codemeal sql problem solve <id>` first. The submit command intentionally uses the saved workspace file instead of accepting a query on the command line.

SQL submission cannot ask for confirmation. This happens when a script or CI job has no interactive terminal. Re-run it as `codemeal sql problem submit <id> --yes` to authorize the private test run.

A command exited non-zero but printed nothing. That is deliberate for `revision next --quiet` and `pattern template` — there was nothing to print. Every other failure prints a message to standard error prefixed with `Error:`.

Something else. Run the command with `--help` to check the flags, and `codemeal --version` to confirm which build you are on before reporting it.

10 Quick reference

Account

```
codemeal login                    sign in via Google or GitHub
codemeal logout                   remove the saved credential
codemeal whoami                   who am I signed in as
```

Problems

```
codemeal problem <id>              show one problem
codemeal problem list              list problems
    -s --skip N                      skip N
    -l --limit N                     fetch N (default 30)
    -t --title TEXT                  search by title
    -d --difficulty LEVEL            filter by difficulty
codemeal problem solve <id>        start or resume a solution
    --no-open                        don't open an editor
codemeal problem submit <id>       run the tests and submit
    --confidence LEVEL               struggled | okay | easy
    --no-revision                    don't schedule for revision
codemeal problem stats             solved/total by difficulty, track, tag
codemeal submission                recent submissions
```

SQL problems

```
codemeal sql problem <id>          show one SQL problem
codemeal sql problem list          list SQL problems
    -s --skip N                      skip N
    -l --limit N                     fetch N (default 30, maximum 100)
    -q --search TEXT                 search titles and tags
    -d --difficulty LEVEL            filter by difficulty
    --solved BOOL                    true for solved, false for open
codemeal sql problem solve <id>    start or resume solution.sql
    --no-open                        don't open an editor
codemeal sql problem submit <id>   run public and private tests
    -y --yes                         skip private-test confirmation
codemeal sql problem stats         totals by difficulty, dialect, tag
codemeal sql submission            recent SQL submissions
```

Revision

```
codemeal revision                  what is due, most overdue first
    --limit N                        at most N (default 50)
codemeal revision next             the one to do now
    -q --quiet                       print only the ID
codemeal revision done <id>        record a re-solve
    --confidence LEVEL               struggled | okay | easy
codemeal revision retire <id>      stop scheduling it
```

Notes

```
codemeal note <id>                 print the note
codemeal note edit <id>            write or revise it in your editor
codemeal note set <id>             replace from stdin
    -f --file PATH                   replace from a file
    --force                          don't ask before overwriting
codemeal note delete <id>          delete it
    --force                          don't ask
```

Patterns

```
codemeal pattern                    list the patterns
    --search TEXT                     filter by name or summary
    --skip N                          skip N
    --limit N                         at most N (default 100)
codemeal pattern show <slug>        when to use it, and its problems
    --template                        also print the template
codemeal pattern template <slug>    print only the template
codemeal pattern for <id>           which patterns a problem belongs to
```

Interviews

```
codemeal interview                  your requests and their status
codemeal interview request          ask for one
    --company --role --type --language
    --email --phone --notes
    --from --to --start-time --end-time
    --timezone --duration
codemeal interview resubmit <id>    offer a new window
```

The CLI itself

```
codemeal --version                  which build is this
codemeal --help                     every command
codemeal <command> --help           flags for one command
codemeal update [version]           replace this binary
    --check                           is there an update
    --list                            every version available
    --force                           overwrite a local or current build
    -y --yes                          don't ask
    --allow-unverified                install without a checksum
```

CodeMeal CLI 1.0.7 · codemeal.in/downloads