Package 'LLMR.shiny'

October 29, 2026

Title Shared 'Shiny' Components for 'LLMR' Family Applications

Version 0.1.3

Description Reusable 'Shiny' user interface and server components from which the graphical applications in the 'LLMR' package family are assembled.

License MIT + file LICENSE

URL https://github.com/asanaei/LLMR.shiny

BugReports https://github.com/asanaei/LLMR.shiny/issues

Encoding UTF-8

RoxygenNote 7.3.3

Imports shiny (>= 1.5.0), bslib (>= 0.5.1), stats, utils

Suggests LLMR (>= 0.8.8), DT, testthat (>= 3.0.0), withr

Config/testthat/edition 3

NeedsCompilation no

Author Ali Sanaei [aut, cre]

Maintainer Ali Sanaei <sanaei@uchicago.edu>

Repository CRAN

Date/Publication 2026-10-29 00:10:08 UTC

Contents

annotate_demo_result 2
as_display_table 3
build_llm_config 4
build_runner 4
column_names_for_mapping 5
condition_category 5
demo_banner_ui 6
demo_notice 6
demo_runner 7

diagnostics_table . . . 7
extract_token_counts . . . 8
guess_column . . . 9
help_tip . . . 10
install_guidance_ui . . . 10
is_demo_result . . . 11
key_state . . . 11
key_state_tile . . . 12
live_run_blocker_ui . . . 12
llmr_theme . . . 13
map_columns . . . 13
persona_selector_server . . . 14
persona_selector_ui . . . 15
provider_choices . . . 15
provider_default_model . . . 16
provider_display_name . . . 16
provider_env_vars . . . 17
provider_registry . . . 17
read_csv_path . . . 18
read_csv_upload . . . 18
report_text . . . 19
safe_llmr_call . . . 19
shell_context . . . 20
shell_sidebar . . . 20
text_block_output . . . 21
usage_add . . . 21
usage_empty . . . 22
usage_set_plan . . . 23
usage_tile . . . 23
validate_column_mapping . . . 24

Index **25**

`annotate_demo_result` *Mark a result frame as demonstration output*

Description

Marks a result produced without a live model: a `run_mode` column set to `"demo"`, a `demo_notice` column, and the `llmrshiny_demo_result` class that `is_demo_result()` tests and `demo_banner_ui()` announces. `demo_runner()` applies it automatically; call it directly for demonstration results built some other way.

Usage

```
annotate_demo_result(x)
```

Arguments

x	A data frame of results.

Value

x with the two source columns and the demonstration class added.

Examples

```
annotate_demo_result(data.frame(response_text = "stub"))
```

---

as_display_table *Coerce an arbitrary result to a display table*

Description

Data frames and matrices pass through (head-limited). For a list, name the component to display. Anything else becomes a one-column capture of its structure.

Usage

```
as_display_table(x, max_rows = 500L, component = NULL, digits = 3L)
```

Arguments

x	Any object.
max_rows	Row cap for the display.
component	For a list, the name of the component to display.
digits	Number of significant digits used for ordinary double columns. Integer, identifier, and nonnumeric columns are not changed.

Value

A data frame.

Examples

```
x <- data.frame(
  item_id = c(1000001, 1000002),
  share = c(0.123456, 0.987654),
  count = c(3L, 7L)
)
as_display_table(x)
```

build_llm_config *Build an LLM config*

Description

Build an LLM config

Usage

```
build_llm_config(
  provider,
  model,
  ...,
  temperature = NULL,
  max_tokens = NULL,
  reasoning_effort = NULL
)
```

Arguments

provider, model	Provider and model ids.
...	Other arguments passed to LLMR::llm_config().
temperature	Sampling temperature, or NULL to use the model default.
max_tokens	Maximum output tokens, or NULL, NA, or "" to use the model default.
reasoning_effort	Reasoning effort ("low", "medium", or "high"), or NULL or "" to use the model default.

Value

An LLMR::llm_config.

build_runner *Build an execution function for a mode*

Description

Build an execution function for a mode

Usage

```
build_runner(mode, responder = NULL)
```

Arguments

mode	"demo" or "live".
responder	Optional demonstration responder (see demo_runner()).

Value

A callable function that accepts a request data frame.

---

column_names_for_mapping *Column names of a data frame, for a mapping select input*

Description

Column names of a data frame, for a mapping select input

Usage

```
column_names_for_mapping(data)
```

Arguments

data	A data frame.

Value

A character vector of column names.

---

condition_category *Error category of a caught condition*

Description

LLMR's classed errors carry their category in the condition class (llmr_api_auth_error, llmr_api_rate_limit_error, and so on); the class is read first. A plain category field or attribute is honored as a fallback for conditions built by other tools.

Usage

```
condition_category(e)
```

Arguments

e	A condition.

Value

A length-1 character category (e.g. `"auth"`, `"rate_limit"`, `"param"`, `"server"`, `"unknown"`), or `NA`.

---

demo_banner_ui *A banner announcing a demonstration result*

Description

A banner announcing a demonstration result

Usage

```
demo_banner_ui()
```

Value

A warning card.

---

demo_notice *Demo-result notice string*

Description

Demo-result notice string

Usage

```
demo_notice()
```

Value

The marker text stamped on every demo result.

demo_runner *An offline demonstration response function*

Description

Returns a function accepted by `.runner` arguments. It adds LLMR response columns to a request data frame. The per-row response text is decided by `responder`, a function `(text) -> character`. Results are marked as demonstrations.

Usage

```
demo_runner(
  responder = NULL,
  text_cols = c("text", "unit", "document", "prompt", "input")
)

## S3 method for class 'llmrshiny_demo_result'
print(x, ...)
```

Arguments

`responder`	A function mapping a single input text to a response string. Defaults to echoing a short stub.
`text_cols`	Candidate column names to read the input text from.
`x`	A demonstration result.
`...`	Passed to the next print method.

Value

A response function of class `llmrshiny_demo_runner`.

diagnostics_table *Render an object's* `diagnostics()` *as a display table*

Description

Render an object's `diagnostics()` as a display table

Usage

```
diagnostics_table(x, ...)
```

Arguments

`x`	A result object with an `LLMR::diagnostics()` method.
`...`	Passed to `diagnostics()`.

Value

A data frame, or NULL when no method applies.

`extract_token_counts` *Summarize call counts and token usage*

Description

Reads token counts from a result data frame for use by `usage_add()`.

Usage

```
extract_token_counts(x, fallback_calls = 0L)
```

Arguments

x	A result data frame, or a list containing data frames. A data frame supplied directly is used as given. List components are ranked by their columns in the following order. 1. Both call identifiers and tokens. 2. Call identifiers only. 3. Tokens only. 4. Neither. Ties use the first frame and produce a warning naming the candidates.
fallback_calls	A nonnegative whole-number call count used when x contains no data frame. A positive count has unknown usage. NA is retained with a warning. Counts must fit in an R integer.

Details

A row's total is its reported `total_tokens` value. If it is missing, the total is computed from `sent_tokens` and `rec_tokens` when both are known. Each component is summed over its reported values, including failed calls. An absent column supplies missing values. Nonnumeric, negative, fractional, and non-finite values are treated as missing with one warning per column. A logical column containing only NA is accepted as missing without a warning.

The integer unknown count includes live rows missing any count after computing the total. `sent_unknown` and `received_unknown` count rows missing the respective component. A row with `rowpack_row > 1` is excluded from these counts only when a live row with the same `rowpack_id` and `rowpack_row == 1` is present.

Call identifiers are considered in the order `call_id`, `request_id`, then `response_id`. The first column with complete identifiers on at least one live row and on every nonfailed live row is used. Distinct identifiers count as calls. A list-column may contain several identifiers per row; missing or empty entries make that row incomplete. Failed rows (`success = FALSE`) with incomplete identifiers add one call each. If an identifier column exists but every live row is failed and unidentified, each row counts as one call.

If no identifier column qualifies, live rows are counted as result rows. Incomplete identifiers on nonfailed rows produce a warning. Rows with `run_mode = "demo"` add result rows and no tokens. If `run_mode` is absent, the `llmrshiny_demo_result` class marks the whole frame as demonstration output.

Value

A named list with double token subtotals `sent`, `received`, and `total`, and integer counts `unknown`, `sent_unknown`, and `received_unknown`. A subtotal is `NA` when no value is reported and `unknown` is positive; otherwise an empty subtotal is zero.

The list contains `calls` when an identifier column qualifies or the fallback is used. It also contains `calls` when an identifier column exists and every live row is failed and unidentified. Otherwise it contains `result_rows`. Demonstration rows add `result_rows`, including when identifiable live calls are present.

Examples

```
calls <- data.frame(response_id = c("r1", "r2", "r3"),
                    sent_tokens = c(100L, 110L, NA),
                    rec_tokens = c(20L, 21L, NA))
extract_token_counts(calls)
```

---

guess_column *Guess a column for a data-mapping role*

Description

Column names are matched without regard to case or common separators. When no recognized name remains, the first available column is returned. Text names include `text`, `body`, `content`, `response`, `message`, and `utterance`; label names include `label`, `labels`, `code`, `category`, and `class`; identifier names include `id`, `doc_id`, and `unit_id`.

Usage

```
guess_column(cols, kind = c("text", "label", "id"), exclude = character())
```

Arguments

`cols`	A character vector of column names.
`kind`	The mapping role to identify: `"text"`, `"label"`, or `"id"`.
`exclude`	Column names that are already assigned to another role.

Value

One column name as a character vector of length one, or `NULL` when no column remains.

Examples

```
cols <- c("id", "text", "label")
text_col <- guess_column(cols, "text")
guess_column(cols, "label", exclude = text_col)
```

---

help_tip *Inline help tooltip*

Description

Inline help tooltip

Usage

```
help_tip(text, placement = "right")
```

Arguments

text	Help text shown by the tooltip and exposed to assistive technology.
placement	Tooltip placement relative to the icon.

Value

A focusable information icon wrapped in a bslib tooltip.

Examples

```
shiny::tags$label(
  "Turn-taking flow",
  help_tip("How the next speaker is chosen.")
)
```

---

install_guidance_ui *Install-guidance card for a missing package*

Description

Install-guidance card for a missing package

Usage

```
install_guidance_ui(package, title = package)
```

Arguments

package	Package name.
title	Card title (defaults to the package name).

Value

A `bslib::card` with an installation command.

`is_demo_result` *Is a result a demonstration result?*

Description

Is a result a demonstration result?

Usage

```
is_demo_result(x)
```

Arguments

`x`	A result object.

Value

`TRUE` when the result has the demonstration class or source fields.

`key_state` *Key state for a provider, read from the environment only*

Description

Never returns the key value; only whether one was found and in which variable.

Usage

```
key_state(provider)
```

Arguments

`provider`	Provider id.

Value

A list: `provider`, `display`, `env_vars`, `found`, `env_var`.

key_state_tile *A tile reporting key state (never the key value)*

Description

A tile reporting key state (never the key value)

Usage

```
key_state_tile(state)
```

Arguments

state	A key_state() list.

Value

A bslib value box or warning card.

live_run_blocker_ui *A banner shown when a live run is blocked for want of a key*

Description

A banner shown when a live run is blocked for want of a key

Usage

```
live_run_blocker_ui(state)
```

Arguments

state	A key_state() list.

Value

A warning card.

llmr_theme *Shared theme for an LLMR studio*

Description

The family theme uses a common Bootstrap palette and a studio accent only for the active navigation underline.

Usage

```
llmr_theme(studio = c("content", "panel", "focus"), ...)
```

Arguments

studio	One of "content", "panel", or "focus".
...	Additional arguments passed to bslib::bs_theme(). The Bootstrap version, Bootswatch base, and family colors cannot be overridden.

Value

A Bootstrap 5 bslib theme.

Examples

```
theme <- llmr_theme("content")
if (interactive()) {
  shiny::shinyApp(
    bslib::page_navbar(
      bslib::nav_panel("Home", "Demo"),
      title = "LLMRcontent",
      theme = theme
    ),
    function(input, output, session) {}
  )
}
```

map_columns *Map user columns to* text *(and optionally* labels*)*

Description

Map user columns to text (and optionally labels)

Usage

```
map_columns(data, text_col, label_col = NULL, keep_original = TRUE)
```

Arguments

data	A data frame.
text_col	Name of the column to become text.
label_col	Optional name of the column to become labels.
keep_original	Keep the original columns alongside the mapped ones. A pre-existing text (or labels) column that is not itself the mapped source is preserved under a .original suffix rather than overwritten.

Value

A data frame with a text column (and labels when requested), always with one row per input row.

---

persona_selector_server

Server for the persona selector module

Description

Renders a compact overview of data as a multi-select table, reports the live selected-row count, and returns the selected row indices (relative to data) as a reactive. When data carries the persona contract (see LLMR::llm_persona_overview()), the overview columns are chosen automatically; otherwise the first few columns are shown.

Usage

```
persona_selector_server(
  id,
  data,
  overview = NULL,
  page_length = 8L,
  height = "260px",
  empty_selection_note = "A seeded sample of the requested size will be drawn."
)
```

Arguments

id	Module id.
data	A persona data frame (or a reactive returning one).
overview	Optional overview data frame, or a function function(df) building one. Defaults to LLMR::llm_persona_overview() when LLMR is available, else the first columns of data.
page_length	Rows per page in the table. Default 8.
height	CSS height for the scrollable table body. Default "260px".
empty_selection_note	Optional text appended to the count when no rows are selected. Set NULL to show the count alone.

Value

A `reactive` returning an integer vector of selected row indices into `data` (`integer(0)` when nothing is selected). When `DT` is not installed the module renders nothing and the reactive is always `integer(0)`, matching the install guidance shown by `persona_selector_ui()`.

See Also

`persona_selector_ui()`.

`persona_selector_ui` *UI for the persona selector module*

Description

Renders the selectable persona table. Pair with `persona_selector_server()`. The table's height is set by the `height` argument of `persona_selector_server()`, which controls the scrollable body.

Usage

```
persona_selector_ui(id)
```

Arguments

`id`	Module id.

Value

Shiny UI elements for a live selection count and a `DT` output.

See Also

`persona_selector_server()`.

`provider_choices` *Provider choices for a select input*

Description

Provider choices for a select input

Usage

```
provider_choices()
```

Value

A named character vector (`display` -> `provider`).

provider_default_model *Default model for a provider*

Description

Default model for a provider

Usage

```
provider_default_model(provider)
```

Arguments

provider	Provider id.

Value

The default model string, or "" for an unknown provider.

provider_display_name *Display name for a provider*

Description

Display name for a provider

Usage

```
provider_display_name(provider)
```

Arguments

provider	Provider id.

Value

The display name, or the provider id.

provider_env_vars *Environment-variable names a provider's key may live in*

Description

Environment-variable names a provider's key may live in

Usage

```
provider_env_vars(provider)
```

Arguments

provider	Provider id.

Value

A length-2 character vector `c("<P>_API_KEY", "<P>_KEY")`.

provider_registry *Provider registry*

Description

The registry supplies a working default model for each provider. Set the `LLMR.shiny.default_models` option to a named character vector to replace selected defaults without changing the others.

Usage

```
provider_registry(
  default_models = getOption("LLMR.shiny.default_models", character())
)
```

Arguments

default_models	A named character vector of provider model defaults.

Details

Built-in defaults are `openai/gpt-oss-20b` for Groq and Together, `gpt-4o-mini` for OpenAI, `claude-haiku-4-5` for Anthropic, `deepseek-v4-flash` for DeepSeek, and `gemini-3.1-flash-lite` for Gemini.

Value

A data frame of `provider`, `display`, and `default_model`.

read_csv_path *Read a CSV from a path*

Description

Read a CSV from a path

Usage

```
read_csv_path(path, col_classes = NA, na_strings = "NA")
```

Arguments

path	File path.
col_classes	Passed to utils::read.csv() as colClasses. The default, NA, infers each column's type.
na_strings	Passed to utils::read.csv() as na.strings. The default reads the text "NA" as missing.

Value

A data frame.

read_csv_upload *Read an uploaded CSV (a Shiny fileInput value)*

Description

By default every column's type is inferred, as utils::read.csv() does, so an identifier such as 001 becomes the number 1 and two identifiers 001 and 01 become the same value. To keep identifiers and text exactly as written, read with col_classes = "character" and na_strings = character(0), and convert the other columns afterwards with utils::type.convert(x, as.is = TRUE, na.strings = "NA"), which reproduces the default inference.

Usage

```
read_csv_upload(file, col_classes = NA, na_strings = "NA")
```

Arguments

file	A fileInput value (a list with datapath).
col_classes	Passed to utils::read.csv() as colClasses. The default, NA, infers each column's type.
na_strings	Passed to utils::read.csv() as na.strings. The default reads the text "NA" as missing.

Value

A data frame.

---

report_text *Render an object's* report() *prose, falling back to print output*

Description

Render an object's report() prose, falling back to print output

Usage

```
report_text(x, ...)
```

Arguments

x	A result object with an LLMR::report() method, or any object.
...	Passed to report().

Value

A character scalar of report text.

---

safe_llmr_call *Run an expression, capturing any error as a banner*

Description

Evaluates expr; on error returns a list with ok = FALSE and a ready-made ui banner (auth-aware) instead of stopping. On success returns ok = TRUE and the value.

Usage

```
safe_llmr_call(expr, provider = NULL)
```

Arguments

expr	An expression to evaluate.
provider	Optional provider id, for an auth banner.

Value

list(ok, value, error, ui).

shell_context *Wire the standard sidebar and return the shared reactive context*

Description

Call once at the top of a GUI's server with the top-level `input`, `output`, `session`. It keeps the model field in sync with the provider, renders the key and usage tiles, tracks usage, and returns a list of reactives and mutators (`provider`, `model`, `mode`, `temperature`, `max_tokens`, `reasoning_effort`, `key`, `can_run`, `set_plan`, `add_usage`) for the per-package modules to consume.

Usage

```
shell_context(input, output, session)
```

Arguments

`input`, `output`, `session`
: The top-level Shiny server arguments.

Value

A shared-context list.

shell_sidebar *The standard GUI sidebar: mode, live configuration, and usage*

Description

The standard GUI sidebar: mode, live configuration, and usage

Usage

```
shell_sidebar(
  id = NULL,
  default_provider = "groq",
  max_tokens_help = "Leave blank to use the model default.",
  max_tokens_placeholder = "Model default"
)
```

Arguments

`id`
: The module namespace (or `NULL` for top-level inputs).

`default_provider`
: Provider selected initially.

`max_tokens_help`
: Help text for a blank maximum-output-token field.

`max_tokens_placeholder`
: Placeholder for the maximum-output-token field.

Value

A `bslib::sidebar`.

`text_block_output` *Output container for long text results*

Description

Output container for long text results

Usage

```
text_block_output(id, height = "18rem")
```

Arguments

`id`	Output id passed to `shiny::verbatimTextOutput()`.
`height`	CSS height of the scrollable container.

Value

A bordered, scrollable Shiny output container.

Examples

```
text_block_output("report")
```

`usage_add` *Add usage to a session record*

Description

Adds call counts and token subtotals to a usage record and clears its pending run.

Usage

```
usage_add(state, tokens)
```

Arguments

`state`	A usage record, usually from `usage_empty()`, or `NULL`.
`tokens`	A named list, such as the result of `extract_token_counts()`. It may contain `calls`, `result_rows`, `sent`, `received`, `total`, `unknown`, `sent_unknown`, and `received_unknown`.

Details

Missing or NA token values contribute nothing to the stored subtotals. A total absent from tokens is computed from sent plus received when both are known and unknown is absent or zero. An explicit NA total remains unknown. Token values must be finite nonnegative whole numbers; an invalid value is treated as unknown with one warning naming the field.

A supplied unknown count is added to the record. If absent, it defaults to the call count when any token component is unknown, and zero otherwise. The same rule applies separately to sent_unknown and received_unknown using their respective components. Result rows alone do not supply these unknown counts.

Missing call and result-row counts contribute zero. A scalar NA count is retained with a warning. Other counts must be finite nonnegative whole numbers that fit in an R integer. Fields absent from an older record start at zero; missingness not recorded there cannot be recovered.

Value

The updated record with double token subtotals and integer counts. planned_calls is zero and planned_label is "No pending run". Other fields are retained.

Examples

```
state <- usage_add(usage_empty(), list(calls = 2, sent = 120, received = 40))
state <- usage_add(state, list(calls = 1))
state[c("calls", "sent", "received", "total", "unknown")]
```

---

usage_empty *Create an empty session usage record*

Description

Initializes the counters used by usage_add() and usage_tile().

Usage

```
usage_empty()
```

Value

A named list with zero numeric counters calls, result_rows, sent, received, total, unknown, sent_unknown, received_unknown, and planned_calls. Token subtotals are doubles; the other counters are integers. planned_label is initialized to "No pending run".

usage_set_plan *Record a planned run on a usage record*

Description

Record a planned run on a usage record

Usage

```
usage_set_plan(state, calls, label = "Next run")
```

Arguments

state	A usage record.
calls	Number of calls the next run will make.
label	A short label for the planned run.

Value

The updated usage record.

usage_tile *Display session usage*

Description

Creates a value box showing recorded calls, result rows, and sent and received token subtotals. A pending run is shown when its planned call count is positive.

Usage

```
usage_tile(state)
```

Arguments

state	A usage record, usually updated by usage_add(), or NULL.

Details

Each component uses its own unknown count. A zero unknown count gives the exact subtotal. Otherwise a zero subtotal is labeled "unknown" and a positive subtotal is prefixed by "at least". An NA subtotal is labeled "unknown". If both components are unknown, the box displays "Token usage unknown". The unknown field gives the number of replies with incomplete usage in a separate line.

Unknown counts absent from older records are treated as zero. An NA call or result-row count is displayed as unknown.

Value

A value box created by `bslib::value_box()`.

`validate_column_mapping` *Validate a column mapping*

Description

Columns are selected by name, so the data must have unique, non-empty column names.

Usage

```
validate_column_mapping(data, text_col, label_col = NULL)
```

Arguments

`data`	A data frame.
`text_col`	Required text column name.
`label_col`	Optional label column name; `NULL` or `""` for none.

Value

`TRUE`, or an error.

Index

annotate_demo_result, 2
as_display_table, 3

bslib::bs_theme(), *13*
bslib::value_box(), *24*
build_llm_config, 4
build_runner, 4

column_names_for_mapping, 5
condition_category, 5

demo_banner_ui, 6
demo_banner_ui(), *2*
demo_notice, 6
demo_runner, 7
demo_runner(), *2*, *5*
diagnostics_table, 7

extract_token_counts, 8
extract_token_counts(), *21*

guess_column, 9

help_tip, 10

install_guidance_ui, 10
is_demo_result, 11
is_demo_result(), *2*

key_state, 11
key_state(), *12*
key_state_tile, 12

live_run_blocker_ui, 12
LLMR::llm_config(), *4*
LLMR::llm_persona_overview(), *14*
llmr_theme, 13

map_columns, 13

persona_selector_server, 14
persona_selector_server(), *15*
persona_selector_ui, 15
persona_selector_ui(), *15*
print.llmrshiny_demo_result
*(*demo_runner*)*, 7
provider_choices, 15
provider_default_model, 16
provider_display_name, 16
provider_env_vars, 17
provider_registry, 17

read_csv_path, 18
read_csv_upload, 18
report_text, 19

safe_llmr_call, 19
shell_context, 20
shell_sidebar, 20
shiny::verbatimTextOutput(), *21*

text_block_output, 21

usage_add, 21
usage_add(), *8*, *22*, *23*
usage_empty, 22
usage_empty(), *21*
usage_set_plan, 23
usage_tile, 23
usage_tile(), *22*
utils::read.csv(), *18*

validate_column_mapping, 24