Package: LLMR.shiny (via r-universe)

July 22, 2026

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

Version 0.1.2

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

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports shiny, bslib, stats, utils

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

Config/testthat/edition 3

Config/pak/sysreqs cmake make libuv1-dev zlib1g-dev

Repository https://asanaei.r-universe.dev

Date/Publication 2026-07-22 20:57:49 UTC

RemoteUrl https://github.com/asanaei/llmr.shiny

RemoteRef HEAD

RemoteSha dfd30725badd9941d9a393c2efe162b9e3dbcf41

Contents

`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)
```

Arguments

x	Any object.
max_rows	Row cap for the display.
component	For a list, the name of the component to display.

Value

A data frame.

---

build_llm_config *Build an LLM config*

Description

Build an LLM config

Usage

```
build_llm_config(provider, model, ...)
```

Arguments

provider, model	Provider and model ids.
...	Passed to LLMR::llm_config() (e.g. temperature).

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 *Extract call/token counts from a result frame*

Description

Extract call/token counts from a result frame

Usage

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

Arguments

x	A result data frame (or list containing one).
fallback_calls	Call count to use when x has no result frame.

Value

A list with token totals and either calls, when explicit call provenance is available, or result_rows.

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.

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 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"
)
```

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"`.

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

A Shiny UI element (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 "".

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

Model defaults are optional because provider aliases can change between package releases. Set the `LLMR.shiny.default_models` option to a named character vector to supply local defaults.

Usage

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

Arguments

default_models	A named character vector of provider model defaults.

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)
```

Arguments

path	File path.

Value

A data frame.

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

Description

Read an uploaded CSV (a Shiny fileInput value)

Usage

```
read_csv_upload(file)
```

Arguments

file	A `fileInput` value (a list with `datapath`).

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`, `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: provider, model, mode, key tile, usage tile*

Description

The standard GUI sidebar: provider, model, mode, key tile, usage tile

Usage

```
shell_sidebar(id = NULL, default_provider = "groq")
```

Arguments

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

`default_provider`
: Provider selected initially.

Value

A `bslib::sidebar`.

usage_add *Add realized usage to a usage record*

Description

Add realized usage to a usage record

Usage

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

Arguments

state	A usage record.
tokens	A token-count list (see extract_token_counts()).

Value

The updated usage record.

usage_empty *An empty usage record*

Description

An empty usage record

Usage

```
usage_empty()
```

Value

A list with zeroed counters.

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 *A value box reporting session usage*

Description

A value box reporting session usage

Usage

```
usage_tile(state)
```

Arguments

state	A usage record.

Value

A `bslib::value_box`.

validate_column_mapping *Validate a column mapping*

Description

Validate a column mapping

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.

Value

TRUE, or an error.

Index