Package 'engager'

July 30, 2026

Type Package

Title Analyze Student Engagement from 'WebVTT' Transcripts

Version 0.1.0

Description Analyzes participation in course-session transcripts stored in the 'WebVTT' format <https://www.w3.org/TR/webvtt1/>, including transcripts exported by 'Zoom' and similar videoconferencing platforms. Provides tools to load and process transcripts, calculate speaker-level engagement metrics, create privacy-supporting plots and exports, and perform reviewable exact name matching against course rosters. Structured-field masking and technical privacy-review helpers support local review but do not determine legal or institutional compliance.

License MIT + file LICENSE

Depends R (>= 4.1.0)

Language en-US

Encoding UTF-8

RoxygenNote 7.3.3

Imports digest, dplyr, ggplot2, hms, openssl, jsonlite, lubridate, magrittr, tidyr, readr, rlang, stringi, stringr, tibble

Suggests testthat (>= 3.0.0), withr, covr, knitr, rmarkdown, purrr, microbenchmark, gridExtra

VignetteBuilder knitr

Config/testthat/edition 3

URL https://github.com/revgizmo/engager, https://revgizmo.github.io/engager/

BugReports https://github.com/revgizmo/engager/issues

NeedsCompilation no

Author Conor Healy [aut, cre]

Maintainer Conor Healy <conorhealy@berkeley.edu>

Repository CRAN

Date/Publication 2026-07-30 17:20:16 UTC

Contents

engager-package . . . 3
abort_zse . . . 3
analyze_transcripts . . . 4
anonymize_educational_data . . . 5
apply_name_matching . . . 6
apply_privacy_aware_matching . . . 7
basic_transcript_analysis . . . 7
batch_basic_analysis . . . 8
calculate_content_similarity . . . 9
consolidate_transcript . . . 10
convert_error_to_user_friendly . . . 10
data . . . 11
detect_unmatched_names . . . 11
ensure_privacy . . . 12
ethical_compliance . . . 13
find_function_for_task . . . 13
generate_privacy_review_report . . . 14
get_next_level . . . 15
get_smart_recommendations . . . 15
get_ux_level . . . 16
get_visible_functions . . . 16
load_roster . . . 17
load_zoom_transcript . . . 17
lookup_merge_utils . . . 18
make_blank_cancelled_classes_df . . . 19
make_blank_section_names_lookup_csv . . . 19
match_names_workflow . . . 19
plot_users . . . 20
privacy_audit . . . 21
privacy_review . . . 21
process_zoom_transcript . . . 22
quick_analysis . . . 23
review_privacy_risks . . . 24
safe_name_matching_workflow . . . 25
schema . . . 26
set_ux_level . . . 26
show_available_functions . . . 27
show_error_recovery . . . 27
show_function_categories . . . 28
show_function_help . . . 28
show_getting_started . . . 29
show_privacy_guidance . . . 29
show_troubleshooting . . . 30
show_workflow_help . . . 30
summarize_transcript_files . . . 31
summarize_transcript_metrics . . . 32

user_friendly_error 33
validate_directory_argument 34
validate_file_argument 34
validate_privacy_compliance 35
write_metrics 36
write_unresolved 37
zse_schema 37
%>% 38

Index **39**

`engager-package` *engager: Student Engagement Analysis Package*

Description

Tools for processing Zoom transcripts, matching names with rosters, computing engagement metrics, and creating privacy-aware visualizations and reports. The package emphasizes privacy-supporting defaults, privacy review helpers, and masked outputs by default.

Author(s)

Maintainer: Conor Healy <conorhealy@berkeley.edu>

See Also

Useful links:

- https://github.com/revgizmo/engager
- https://revgizmo.github.io/engager/
- Report bugs at https://github.com/revgizmo/engager/issues

`abort_zse` *Error handling helpers*

Description

Provides a small wrapper around `rlang::abort()` to standardize error classes within the package for precise testing and user handling.

Usage

```
abort_zse(message, class = character())
```

Arguments

message	Error message to display
class	Additional error class to add to the error

Value

No return value, throws an error

---

analyze_transcripts *Analyze Transcripts (High-level orchestration)*

Description

Convenience wrapper to process a set of `.transcript.vtt` files from a folder, compute engagement metrics, and optionally write outputs.

Usage

```
analyze_transcripts(
  transcripts_folder = NULL,
  names_to_exclude = c("dead_air"),
  write = FALSE,
  output_path = NULL
)
```

Arguments

transcripts_folder	Path to folder containing transcript files
names_to_exclude	Vector of names to exclude from analysis (default: "dead_air")
write	Whether to write results to file (default: FALSE)
output_path	Explicit path for the output file when `write = TRUE`.

Value

A tibble with speaker-level engagement metrics for the discovered transcript files. When `write = TRUE`, the same metrics are also written to `output_path` after export privacy policies are applied.

anonymize_educational_data
Anonymize Educational Data

Description

Advanced anonymization for educational data that preserves data utility while supporting privacy review.

Usage

```
anonymize_educational_data(
  data = NULL,
  method = c("mask", "hash", "pseudonymize"),
  preserve_columns = NULL,
  hash_salt = NULL,
  aggregation_level = c("individual", "section", "course", "institution")
)
```

Arguments

`data`	Data frame or tibble containing educational data
`method`	Identifier transformation method: "mask", "hash", or "pseudonymize". The previously advertised "aggregate" method is not supported in version 0.1.0 because it did not reliably remove row-level identifiers.
`preserve_columns`	Vector of column names to preserve unchanged
`hash_salt`	Required non-empty salt for hash-based transformation.
`aggregation_level`	Reserved for a future aggregation workflow. It has no effect for supported version 0.1.0 methods.

Value

A data frame of the same row shape as `data`, with recognized structured identifier columns transformed by the selected method. Missing and blank identifiers remain missing or blank. These transformations do not inspect free text and do not establish that a data set is anonymous or compliant with legal or institutional requirements.

Anonymize sample data

sample_data <- tibble::tibble(student_id = c("12345", "67890"), preferred_name = c("Alice Johnson", "Bob Smith"), section = c("A", "B"), participation_score = c(85, 92))

Mask method (default)

anonymized <- anonymize_educational_data(sample_data, method = "mask")

Hash method requires a caller-provided salt

hashed <- anonymize_educational_data(sample_data, method = "hash", hash_salt = "my_salt")

Examples

```
sample_data <- tibble::tibble(
  student_id = c("12345", "67890"),
  preferred_name = c("Alice Johnson", "Bob Smith"),
  section = c("A", "B"),
  participation_score = c(85, 92)
)
anonymize_educational_data(sample_data)
```

---

apply_name_matching *Apply Name Matching*

Description

Internal function to apply name matching to transcript data.

Usage

```
apply_name_matching(transcript_data, name_lookup, roster_data)
```

Arguments

transcript_data	Transcript data to match
name_lookup	Name lookup table
roster_data	Roster data for matching

Value

Matched transcript data

apply_privacy_aware_matching *Apply Privacy-Aware Name Matching*

Description

Internal function to apply privacy-aware name matching using consistent hashing. This function enhances the existing matching logic with privacy-supporting design.

Usage

```
apply_privacy_aware_matching(result, section_names_lookup, privacy_level)
```

Arguments

result	Data frame containing matching results
section_names_lookup	Lookup table for section names
privacy_level	Privacy level for data processing

Value

Enhanced matching results with privacy-aware processing

basic_transcript_analysis *Basic Analysis Workflow for New Users*

Description

Simplified workflow that guides new users through basic analysis of Zoom transcripts with minimal complexity. Complete basic transcript analysis workflow

This is the main function for new users. It performs a complete analysis workflow in five simple steps: load, process, analyze, visualize, and either return results in memory or export to an explicit directory.

Usage

```
basic_transcript_analysis(
  transcript_file,
  output_dir = NULL,
  privacy_level = "high"
)
```

Arguments

transcript_file	Path to a WebVTT transcript file
output_dir	Optional output directory. When NULL (the default), the analysis is returned in memory and no files or directories are created.
privacy_level	Privacy protection level: "high" (default), "medium", or "low". These map to "privacy_strict", "privacy_standard", and "mask", respectively. All three levels mask common identifiers.

Value

A named list with analysis (a tibble of speaker metrics), plots (a ggplot object), output_dir (the explicit output directory or NULL), transcript_file, and privacy_level.

Examples

```
transcript_file <- system.file(
  "extdata/test_transcripts/ideal_course_session1.vtt",
  package = "engager"
)
results <- basic_transcript_analysis(transcript_file)
```

---

batch_basic_analysis *Batch analysis for multiple transcript files*

Description

Process multiple transcript files in one workflow

Usage

```
batch_basic_analysis(
  transcript_files,
  output_dir = NULL,
  privacy_level = "high"
)
```

Arguments

transcript_files	Vector of transcript file paths
output_dir	Optional parent output directory. When NULL (the default), all results are returned in memory without creating directories.
privacy_level	Privacy protection level

Value

A named list keyed by transcript basename. Each element is the analysis list returned by `basic_transcript_analysis()` or a list with an `error` message when that file could not be processed.

Examples

```
transcript_dir <- system.file("extdata/test_transcripts", package = "engager")
files <- file.path(transcript_dir, c(
  "ideal_course_session1.vtt", "ideal_course_session2.vtt"
))
results <- batch_basic_analysis(files)
```

---

calculate_content_similarity

Calculate Content Similarity Between Two Transcripts

Description

This function calculates the similarity between two transcript data frames based on various metrics including speaker overlap, duration, word count, comment count, and content similarity.

Usage

```
calculate_content_similarity(
  transcript1 = NULL,
  transcript2 = NULL,
  names_to_exclude = c("dead_air")
)
```

Arguments

`transcript1`	First transcript data frame
`transcript2`	Second transcript data frame
`names_to_exclude`	Vector of names to exclude from comparison (default: "dead_air")

Value

Similarity score between 0 and 1

consolidate_transcript

Consolidate Transcript

Description

Take a tibble containing the comments from a Zoom recording transcript and return a tibble that consolidates all consecutive comments from the same speaker where the time between the end of the first comment and start of the second comment is less than `max_pause_sec` seconds. This function addresses an issue with the Zoom transcript where the speaker is speaking a continuous sentence, but the Zoom transcript will cut the comment into two lines. For example, a comment of "This should be a single sentence." is often split into "This should be" and "a single sentence". This function stitches those together into "This should be a single sentence." where the `start` time of the consolidated comment will be the beginning of the first row and the `end` time of the consolidated comment will be the ending of the last row.

Usage

```
consolidate_transcript(df = NULL, max_pause_sec = 1)
```

Arguments

`df`	A tibble containing transcript comments with columns: name, start, end, comment
`max_pause_sec`	Maximum pause in seconds between comments to consolidate (default: 1)

Value

A tibble with consecutive comments consolidated by speaker and pause threshold, including `name`, `comment`, `start`, `end`, `duration`, and `wordcount` columns. Returns `NULL` when `df` is not a tibble.

convert_error_to_user_friendly

Convert technical error to user-friendly message

Description

Converts technical error messages to helpful, actionable guidance

Usage

```
convert_error_to_user_friendly(error_message, context)
```

Arguments

error_message	Technical error message
context	Operation context

Value

User-friendly error message

---

data *Package Data Documentation*

Description

User-provided data and bundled synthetic fixtures.

Details

The package is designed to analyze user-provided WebVTT transcripts and related institutional data. It does not bundle real course or student data. Synthetic transcript and roster fixtures are included under `inst/extdata/test_transcripts` for runnable examples and package validation. They must not be interpreted as institutional records.

See also: load_zoom_transcript() for loading transcript data and write_metrics() for exporting analysis results.

---

detect_unmatched_names *Detect unmatched names from transcripts against a roster*

Description

Detect unmatched names from transcripts against a roster

Usage

```
detect_unmatched_names(transcripts_df, roster_df, options = list())
```

Arguments

transcripts_df	Tibble/data.frame with at least a `speaker` column.
roster_df	Tibble/data.frame returned by `load_roster()`.
options	List of options; supports `match_strategy` (default `'exact'`) and `include_name_hash` (default `FALSE`).

Value

Tibble with columns: `name_hash`, `occurrence_n`, `first_seen_at`, `reason`, `guidance`. `name_hash` omitted unless `include_name_hash = TRUE`.

See Also

Other name-matching: `match_names_workflow()`, `write_unresolved()`

`ensure_privacy` *Ensure Privacy for Outputs*

Description

Applies privacy rules to objects before they are returned, written, or plotted. By default, masks personally identifiable information in tabular data to privacy-supporting placeholders.

Usage

```
ensure_privacy(
  x = NULL,
  privacy_level = getOption("engager.privacy_level", "mask"),
 id_columns = c("preferred_name", "name", "first_last", "name_raw", "student_id",
    "email", "transcript_name", "formal_name", "user_name", "speaker"),
  audit_log = TRUE
)
```

Arguments

`x`	Data object to apply privacy rules to (typically a `tibble`)
`privacy_level`	Privacy level: "mask", "privacy_strict", "privacy_standard", or "none". Deprecated aliases "ferpa_strict" and "ferpa_standard" are accepted with a warning.
`id_columns`	Vector of column names to treat as identifiers (default: common name columns)
`audit_log`	Whether to log privacy operations (default: TRUE)

Details

CRITICAL ETHICAL USE: This function is designed to promote participation equity and educational improvement, NOT surveillance. Outputs are masked by default to support student privacy review; institutional compliance review remains the user's responsibility.

The default behavior is controlled by the global option `engager.privacy_level`, which is set to "mask" on package load. Pass `privacy_level` explicitly for one call, or set the `engager.privacy_level` option for the current R session.

Value

For tabular input, an object with the same data-frame class as `x` and configured identifier columns transformed according to `privacy_level`. Non-tabular input is returned unchanged. This is a technical masking result, not a legal or institutional compliance determination.

Examples

```
df <- tibble::tibble(
  section = c("A", "A", "B"),
  preferred_name = c("Alice Johnson", "Bob Lee", "Cara Diaz"),
  session_ct = c(3, 5, 2)
)
ensure_privacy(df)
```

---

ethical_compliance *Ethical Review Functions*

Description

Internal technical prompts for reviewing educational data-analysis use. These helpers do not determine whether a workflow is ethically approved or compliant with institutional requirements.

---

find_function_for_task *Interactive Help System*

Description

Provides interactive help and function discovery to help users find the right functions for their specific tasks. Interactive function discovery

Helps users find the right function based on what they want to do

Usage

```
find_function_for_task(task)
```

Arguments

`task`	Description of what user wants to do

Value

Invisibly `NULL`; called for its function suggestions printed to the console.

Examples

```
find_function_for_task("load transcript file")
find_function_for_task("create visualizations")
find_function_for_task("export results")
```

`generate_privacy_review_report`
Generate a Privacy Review Report

Description

Generates a lightweight report from `review_privacy_risks()`. The report is intended to support local review and documentation; it does not certify legal compliance.

Usage

```
generate_privacy_review_report(
  data = NULL,
  output_file = NULL,
  report_format = c("text", "html", "json"),
  include_audit_trail = TRUE,
  institution_info = NULL,
  institution_type = c("educational", "research", "mixed")
)
```

Arguments

`data`	Data frame or tibble containing educational data.
`output_file`	Optional file path for writing the report.
`report_format`	Output format: "text", "html", or "json".
`include_audit_trail`	Whether to include basic report metadata.
`institution_info`	Optional institution-provided metadata to include.
`institution_type`	Review context: "educational", "research", or "mixed".

Value

A named list containing report metadata, a summary, the underlying technical review results, and recommendations. When `output_file` is supplied, the same report is also serialized there.

get_next_level *Get next UX level*

Description

Get next UX level

Usage

```
get_next_level(current_level)
```

Arguments

current_level	Current UX level

Value

Next UX level

get_smart_recommendations *Smart function recommendations*

Description

Provides intelligent function recommendations based on user context

Usage

```
get_smart_recommendations(context)
```

Arguments

context	User's current context or situation

Value

Invisibly NULL; called for its recommendations printed to the console.

Examples

```
get_smart_recommendations("new user")
get_smart_recommendations("batch processing")
get_smart_recommendations("privacy concerns")
```

get_ux_level *Get current user experience level*

Description

Get current user experience level

Usage

```
get_ux_level()
```

Value

A character scalar containing the configured experience level, or "basic" when no level has been configured. This function takes no arguments.

Examples

```
current_level <- get_ux_level()
```

get_visible_functions *Function Visibility Management System*

Description

Manages which functions are visible to users based on experience level to implement progressive disclosure and simplify the user interface. Get visible functions for user experience level

Usage

```
get_visible_functions(level = "basic")
```

Arguments

level	User experience level: "basic", "intermediate", "advanced", "expert"

Value

A character vector of exported function names visible at level.

Examples

```
# Get functions visible to basic users
basic_functions <- get_visible_functions("basic")

# Get functions visible to intermediate users
intermediate_functions <- get_visible_functions("intermediate")
```

load_roster *Load a roster file*

Description

Load a roster file

Usage

```
load_roster(
  data_folder = ".",
  roster_file = "roster.csv",
  strict_errors = FALSE
)
```

Arguments

data_folder	Directory containing the roster file
roster_file	Name of the roster CSV file
strict_errors	Logical; if TRUE, throw error when file doesn't exist

Value

A tibble containing the roster rows. When an `enrolled` column is present, only rows where it is `TRUE` are returned. A missing file returns an empty tibble unless `strict_errors = TRUE`.

Examples

```
roster_path <- tempfile(fileext = ".csv")
readr::write_csv(
  tibble::tibble(student_id = "S001", preferred_name = "Student One"),
  roster_path
)
roster <- load_roster(roster_path)
unlink(roster_path)
```

load_zoom_transcript *Load Zoom Transcript*

Description

Load a Zoom recording transcript and return tibble containing the comments from a Zoom recording transcript

Usage

```
load_zoom_transcript(transcript_file_path = NULL)
```

Arguments

transcript_file_path	Path to the transcript file to load

Details

Original code posted by Conor Healy: https://ucbischool.slack.com/archives/C02A36407K9/p1631855705002000
Addition of `wordcount` by Brooks Ambrose: https://gist.github.com/brooksambrose/1a8a673eb3bf884c1868ad4d80f08246

Value

A tibble with one row per parsed WebVTT cue and columns describing speaker name, cue text, start and end times, duration, word count, and source transcript. Returns `NULL` when the file is empty.

Examples

```
# Load a sample transcript from the package's extdata directory
transcript_file <- system.file("extdata/test_transcripts/intro_statistics_week1.vtt",
  package = "engager"
)
load_zoom_transcript(transcript_file_path = transcript_file)
```

`lookup_merge_utils` *Lookup Merge Utilities (Safe, Transactional, UTF-8)*

Description

Utilities to safely read, merge, and write the participant lookup configuration (`section_names_lookup.csv`). All operations are read-then-merge in memory; writes are opt-in and transactional with timestamped backups to prevent accidental data loss.

Details

Expected columns in lookup data frame:

- transcript_name
- preferred_name
- formal_name
- participant_type (e.g., instructor, enrolled_student, guest, unknown)
- student_id
- notes (optional)

make_blank_cancelled_classes_df

Make Cancelled Classes Tibble This function creates an empty tibble for recording of cancelled class sessions for scheduled classes where a Zoom recording is not expected.

Description

where a zoom recording is not expected.

Usage

```
make_blank_cancelled_classes_df()
```

make_blank_section_names_lookup_csv

Create Blank Section Names Lookup Template

Description

Creates an empty tibble template for customizing student names by section. This function generates a properly structured data frame that can be filled in to map between different name formats (preferred names, formal names, transcript names) for students across different course sections.

Usage

```
make_blank_section_names_lookup_csv()
```

match_names_workflow *Match names workflow with privacy-supporting defaults*

Description

Returns an object of class 'engager_match' with redacted print/summary.

Usage

```
match_names_workflow(transcripts_df, roster_df, options = list())
```

Arguments

transcripts_df	Tibble/data.frame with at least a speaker column.
roster_df	Tibble/data.frame returned by load_roster().
options	List of options; supports match_strategy (default 'exact') and include_name_hash (default FALSE).

Value

A list with elements `transcripts_with_ids`, `unresolved`, and `audit`. Returns an object of class `'engager_match'` with redacted `print()` and `summary()` methods.

See Also

Other name-matching: `detect_unmatched_names()`, `write_unresolved()`

`plot_users` *Plot Users*

Description

Unified plotting function for engagement metrics with privacy-aware options.

Usage

```
plot_users(
  data = NULL,
  metric = "session_ct",
  student_col = "name",
  facet_by = c("section", "transcript_file", "none"),
  mask_by = c("name", "rank"),
  privacy_level = getOption("engager.privacy_level", "mask"),
  metrics_lookup_df = NULL
)
```

Arguments

`data`	A tibble containing the data to plot
`metric`	Column name for the metric to plot (default: "session_ct")
`student_col`	Column name for student identification (default: "name")
`facet_by`	Faceting option: "section", "transcript_file", or "none" (default: "section")
`mask_by`	Masking option: "name" or "rank" (default: "name")
`privacy_level`	Privacy level for data visualization (default: from global option)
`metrics_lookup_df`	Optional lookup table for metric names (default: NULL)

Value

A `ggplot` object whose data contains the selected metric and the configured privacy-processed participant labels, ready to print or modify.

privacy_audit *Privacy Audit*

Description

Summarize which identifier columns were present and how many values were masked.

Usage

```
privacy_audit(
  data = NULL,
 id_columns = c("preferred_name", "name", "first_last", "name_raw", "student_id",
    "email")
)
```

Arguments

data	A tibble containing the data to audit
id_columns	Vector of column names to check for identifiers (default: common name columns)

Value

A tibble with one row per detected identifier column and the fields `column`, `values`, `non_empty`, and `masked_estimate`, summarizing the technical masking state of the supplied data.

privacy_review *Privacy Review Functions*

Description

Functions to support institutional privacy review for educational data. These helpers do not guarantee legal compliance; institutions remain responsible for policy review and authorization.

process_zoom_transcript *Process Zoom Transcript*

Description

Process a Zoom recording transcript with given parameters and return tibble containing the consolidated and annotated comments.

Usage

```
process_zoom_transcript(
  transcript_file_path = "",
  consolidate_comments = TRUE,
  max_pause_sec = 1,
  add_dead_air = TRUE,
  dead_air_name = "dead_air",
  na_name = "unknown",
  transcript_df = NULL
)
```

Arguments

transcript_file_path	Path to the transcript file to process
consolidate_comments	Whether to consolidate consecutive comments from the same speaker (default: TRUE)
max_pause_sec	Maximum pause in seconds between comments to consolidate (default: 1)
add_dead_air	Whether to add dead air rows for gaps in transcript (default: TRUE)
dead_air_name	Name to use for dead air periods (default: 'dead_air')
na_name	Name to use for unknown speakers (default: 'unknown')
transcript_df	Pre-loaded transcript data frame (alternative to transcript_file_path)

Details

Original code posted by Conor Healy: https://ucbischool.slack.com/archives/C02A36407K9/p1631855705002000
Addition of `wordcount`, `wordcount_perc`, and `wpm` by Brooks Ambrose: https://gist.github.com/brooksambrose/1a8a673eb3

Value

A tibble of processed transcript cues with normalized speaker, comment, timing, duration, wordcount, and source-file fields. Depending on the arguments, adjacent cues may be consolidated and dead-air rows added.

Examples

```
# Load a sample transcript from the package's extdata directory
transcript_file <- system.file("extdata/test_transcripts/intro_statistics_week1.vtt",
  package = "engager"
)
process_zoom_transcript(transcript_file_path = transcript_file)
```

`quick_analysis` *Quick analysis for single transcript file*

Description

Simplified version for users who just want quick results

Usage

```
quick_analysis(transcript_file, output_dir = NULL)
```

Arguments

`transcript_file`	Path to transcript file
`output_dir`	Optional output directory. When `NULL`, no files or directories are created.

Value

The named analysis list returned by `basic_transcript_analysis()`.

Examples

```
transcript_file <- system.file(
  "extdata/test_transcripts/ideal_course_session1.vtt",
  package = "engager"
)
results <- quick_analysis(transcript_file)
```

`review_privacy_risks` *Review Data for Privacy Risks*

Description

Performs a technical privacy review of a data frame by looking for common identifier columns, optional retention concerns, and institution-specific review prompts. This is a screening helper, not a legal compliance determination.

Usage

```
review_privacy_risks(
  data = NULL,
  institution_type = c("educational", "research", "mixed"),
  check_retention = TRUE,
  retention_period = c("academic_year", "semester", "quarter", "custom"),
  custom_retention_days = NULL,
  audit_log = TRUE
)
```

Arguments

`data`	Data frame or tibble containing educational data.
`institution_type`	Review context: "educational", "research", or "mixed".
`check_retention`	Whether to run the retention-policy helper.
`retention_period`	Retention period: "academic_year", "semester", "quarter", or "custom".
`custom_retention_days`	Custom retention period in days, used when `retention_period = "custom"`.
`audit_log`	Whether to record an in-memory review audit event.

Value

A named list containing the logical technical review status, detected identifier fields, recommendations, retention-review details, and institutional review prompts.

safe_name_matching_workflow *Safe Name Matching Workflow*

Description

Main workflow function for privacy-supporting name matching. Implements two-stage processing: Stage 1 (unmasked matching in memory) and Stage 2 (privacy masking for outputs). Provides configuration-driven behavior for unmatched names.

Usage

```
safe_name_matching_workflow(
  transcript_file_path = NULL,
  roster_data = NULL,
  privacy_level = getOption("engager.privacy_level", "mask"),
  unmatched_names_action = getOption("engager.unmatched_names_action", "stop"),
  data_folder = NULL,
  section_names_lookup_file = "section_names_lookup.csv",
  write_lookup = FALSE
)
```

Arguments

transcript_file_path	Path to the transcript file to process
roster_data	Data frame containing roster information
privacy_level	Privacy level to apply. One of `c("privacy_strict", "privacy_standard", "mask", "none")`. Defaults to `getOption("engager.privacy_level", "mask")`.
unmatched_names_action	Action to take for unmatched names. One of `c("stop", "warn")`. Defaults to `getOption("engager.unmatched_names_action", "stop")`.
data_folder	Directory containing data files
section_names_lookup_file	Name of the section names lookup file
write_lookup	Logical; whether the unmatched-name flow may write a lookup template to the explicitly selected `data_folder`.

Value

Processed transcript data with privacy-aware name matching applied

schema *Schema validators and contracts*

Description

Defines simple schema validation helpers and canonical schemas used throughout the package. Keep this intentionally lightweight to avoid dependency bloat.

set_ux_level *Set user experience level*

Description

Set user experience level

Usage

```
set_ux_level(level = "basic")
```

Arguments

level	User experience level: "basic", "intermediate", "advanced", "expert"

Value

Invisibly, the selected experience level as a character scalar.

Examples

```
old_ux_level <- getOption("zoomstudentengagement.ux_level")
# Set to basic level (5 essential functions)
set_ux_level("basic")

# Set to intermediate level (15 functions)
set_ux_level("intermediate")
options(zoomstudentengagement.ux_level = old_ux_level)
```

show_available_functions *Show functions available at current UX level*

Description

Show functions available at current UX level

Usage

```
show_available_functions(level = NULL)
```

Arguments

level	User experience level (optional, uses current level if not specified)

Value

Invisibly NULL; called for its formatted console output.

Examples

```
# Show functions at current level
show_available_functions()

# Show functions at specific level
show_available_functions("intermediate")
```

show_error_recovery *Show error recovery suggestions*

Description

Provides specific recovery suggestions based on error type

Usage

```
show_error_recovery(error_type)
```

Arguments

error_type	Type of error encountered

Value

Invisibly NULL; called for its recovery guidance printed to the console.

Examples

```
show_error_recovery("file_not_found")
show_error_recovery("permission_denied")
```

`show_function_categories` *Show function categories and counts*

Description

Show function categories and counts

Usage

```
show_function_categories()
```

Value

Invisibly `NULL`; called for its category summary printed to the console.

Examples

```
show_function_categories()
```

`show_function_help` *Show help for specific function*

Description

Show help for specific function

Usage

```
show_function_help(function_name)
```

Arguments

`function_name`	Name of function to get help for

Value

Invisibly `NULL`; called for its formatted console help.

Examples

```
show_function_help("load_zoom_transcript")
show_function_help("basic_transcript_analysis")
```

show_getting_started *User Guidance and Help System*

Description

Provides contextual help and guidance for users to navigate the package effectively and find the right functions for their tasks. Show getting started guide

Displays a comprehensive getting started guide for new users

Usage

```
show_getting_started()
```

Value

Invisibly `NULL`; called for its formatted console guide.

Examples

```
show_getting_started()
```

show_privacy_guidance *Show privacy and ethics guidance*

Description

Show privacy and ethics guidance

Usage

```
show_privacy_guidance()
```

Value

Invisibly `NULL`; called for its formatted console guide.

Examples

```
show_privacy_guidance()
```

show_troubleshooting *Show troubleshooting guide*

Description

Show troubleshooting guide

Usage

```
show_troubleshooting()
```

Value

Invisibly `NULL`; called for its formatted console guide.

Examples

```
show_troubleshooting()
```

show_workflow_help *Show workflow help and templates*

Description

Show workflow help and templates

Usage

```
show_workflow_help()
```

Value

Invisibly `NULL`; called for its formatted console guide.

Examples

```
show_workflow_help()
```

summarize_transcript_files *Summarize Transcript Files*

Description

Summarize multiple transcript files and return aggregated metrics. If a tibble with additional columns beyond 'transcript_file' is provided, all metadata columns will be preserved in the output.

Usage

```
summarize_transcript_files(
  transcript_file_names = NULL,
  data_folder = ".",
  transcripts_folder = "transcripts",
  names_to_exclude = NULL,
  deduplicate_content = FALSE,
  similarity_threshold = 0.95,
  duplicate_method = c("hybrid", "content", "metadata")
)
```

Arguments

transcript_file_names	Vector of transcript file names or tibble with transcript_file column
data_folder	Base folder containing data files (default: ".")
transcripts_folder	Subfolder where transcript files are stored (default: "transcripts")
names_to_exclude	Vector of names to exclude from analysis (default: NULL)
deduplicate_content	Whether to detect and handle duplicate content (default: FALSE)
similarity_threshold	Similarity threshold for duplicate detection (default: 0.95)
duplicate_method	Method for duplicate detection: "hybrid", "content", or "metadata" (default: "hybrid")

Value

A tibble of speaker-level metrics combined across the requested transcript files. Metadata columns supplied with `transcript_file_names` are retained in the result.

Examples

```
transcript_dir <- system.file("extdata/test_transcripts", package = "engager")
transcript_files <- c(
  "ideal_course_session1.vtt",
  "ideal_course_session2.vtt"
)
summary <- summarize_transcript_files(
  transcript_file_names = transcript_files,
  data_folder = transcript_dir,
  transcripts_folder = "."
)
```

`summarize_transcript_metrics` *Summarize Transcript Metrics*

Description

Process a Zoom recording transcript and return summary metrics by speaker

Usage

```
summarize_transcript_metrics(
  transcript_file_path = "",
  names_exclude = c("dead_air"),
  consolidate_comments = TRUE,
  max_pause_sec = 1,
  add_dead_air = TRUE,
  dead_air_name = "dead_air",
  na_name = "unknown",
  transcript_df = NULL,
  comments_format = c("list", "text", "count")
)
```

Arguments

`transcript_file_path`	Path to the transcript file to process
`names_exclude`	Vector of names to exclude from analysis (default: c("dead_air"))
`consolidate_comments`	Whether to consolidate consecutive comments (default: TRUE)
`max_pause_sec`	Maximum pause in seconds between comments to consolidate (default: 1)
`add_dead_air`	Whether to add dead air rows for gaps in transcript (default: TRUE)
`dead_air_name`	Name to use for dead air periods (default: 'dead_air')
`na_name`	Name to use for unknown speakers (default: 'unknown')

`transcript_df` Pre-loaded transcript data frame (alternative to transcript_file_path)

`comments_format`
Format for comments: "list", "text", or "count" (default: "list")

Details

Original code posted by Conor Healy: https://ucbischool.slack.com/archives/C02A36407K9/p1631855705002000
Addition of `wordcount`, `wordcount_perc`, and `wpm` by Brooks Ambrose: https://gist.github.com/brooksambrose/1a8a673eb3

This function preserves in-memory comment data for analysis compatibility. Use `write_metrics()` for privacy-safe CSV exports. The privacy helpers mask structured identifier columns, but they do not redact identifiers embedded in free transcript text.

Value

A tibble with one row per speaker (and source transcript when present), containing participation totals such as duration, word count, session count, and the configured comment representation.

Examples

```
# Load a sample transcript from the package's extdata directory
transcript_file <- system.file("extdata/test_transcripts/intro_statistics_week1.vtt",
  package = "engager"
)
summarize_transcript_metrics(transcript_file_path = transcript_file)
```

`user_friendly_error` *User-Friendly Error Handling System*

Description

Provides helpful error messages and recovery guidance to help users understand and resolve issues quickly. User-friendly error wrapper

Wraps expressions with user-friendly error handling

Usage

```
user_friendly_error(expr, context = "operation")
```

Arguments

`expr`	Expression to evaluate
`context`	Context for error message

Value

The value of `expr` unchanged when evaluation succeeds. On failure, the function raises a rewritten error with recovery guidance.

Examples

```
result <- user_friendly_error(1 + 1, "adding values")
```

`validate_directory_argument` *Validate directory argument with user-friendly errors*

Description

Validates directory arguments and provides helpful error messages

Usage

```
validate_directory_argument(
  dir_path,
  context = "directory operation",
  create_if_missing = TRUE
)
```

Arguments

`dir_path`	Path to directory
`context`	Context for error message
`create_if_missing`	Whether to create directory if it doesn't exist

Value

TRUE if valid, stops with user-friendly error if invalid

`validate_file_argument` *Validate function arguments with user-friendly errors*

Description

Validates function arguments and provides helpful error messages

Usage

```
validate_file_argument(file_path, context = "file operation")
```

Arguments

`file_path`	Path to file
`context`	Context for error message

Value

TRUE if valid, stops with user-friendly error if invalid

`validate_privacy_compliance` *Validate Privacy Compliance*

Description

Scans data objects for possible real names when package masking is enabled. This is an exact-match technical check, not a legal or institutional compliance determination. It stops processing when configured names are found.

Usage

```
validate_privacy_compliance(
  data = NULL,
  privacy_level = getOption("engager.privacy_level", "mask"),
  real_names = NULL,
  stop_on_violation = TRUE
)
```

Arguments

`data`	Data frame or object to inspect for possible identifier exposure
`privacy_level`	Privacy level to validate against. One of `c("privacy_strict", "privacy_standard", "mask", "none")`. Defaults to `getOption("engager.privacy_level", "mask")`. Deprecated aliases "ferpa_strict" and "ferpa_standard" are accepted with a warning.
`real_names`	Vector of real names to check against. If NULL, uses common name patterns to detect potential violations.
`stop_on_violation`	Whether to stop processing if violations are found. Defaults to TRUE for maximum privacy protection.

Value

The logical scalar `TRUE` when the technical check finds no configured real names. The function raises an error instead when a blocking exposure is found.

Examples

```
# Validate privacy compliance
df <- tibble::tibble(
  name = c("Student_01", "Student_02"),
  score = c(85, 92)
)
```

```
validate_privacy_compliance(df)

# Check with specific real names
real_names <- c("John Smith", "Jane Doe")
validate_privacy_compliance(df, real_names = real_names)
```

`write_metrics` *Write Metrics*

Description

Unified writer for engagement-related outputs with privacy enforcement. Parent directories are created if they do not exist.

Usage

```
write_metrics(
  data = NULL,
  what = c("engagement", "summary", "session_summary"),
  path,
  comments_format = NULL,
  privacy_level = getOption("engager.privacy_level", "mask"),
  comments_policy = c("auto", "omit", "count", "text")
)
```

Arguments

`data`	A tibble containing the data to write
`what`	Type of output: "engagement", "summary", or "session_summary" (default: "engagement")
`path`	Explicit file path where the output will be written. No default path is used.
`comments_format`	Deprecated alias for `comments_policy`. Use `comments_policy = "count"` or `comments_policy = "text"` instead. Retained before `privacy_level` to preserve positional compatibility.
`privacy_level`	Privacy level for data export (default: from global option)
`comments_policy`	Export policy for the `comments` column: "auto", "omit", "count", or "text" (default: "auto"). "auto" resolves to "omit" so raw transcript text is never exported accidentally. "text" is allowed only with `privacy_level = "none"` and emits a warning.

Details

`write_metrics()` is the privacy-safe CSV export path. It applies structured identifier masking with `ensure_privacy()` and then omits raw free-text comments by default. `ensure_privacy()` does not redact identifiers embedded inside comment text.

Value

Invisibly returns the exported tibble after privacy and export policies are applied.

`write_unresolved` *Write unresolved matches to disk with privacy guardrails*

Description

By default, writes only hashed columns (no raw names). To include raw names, both conditions must be met: `include_raw = TRUE` and `options(engager.allow_raw_name_exports = TRUE)` is set. Otherwise, the function aborts with class `engager_privacy_error`.

Usage

```
write_unresolved(unresolved_tbl, path, include_raw = FALSE, overwrite = FALSE)
```

Arguments

`unresolved_tbl`	Tibble returned by `detect_unmatched_names()`.
`path`	Output file path.
`include_raw`	Logical; include raw names if `TRUE` and allowed.
`overwrite`	Logical; overwrite existing file if `TRUE`.

Value

Invisibly, a character scalar containing the path that was written.

See Also

Other name-matching: `detect_unmatched_names()`, `match_names_workflow()`

`zse_schema` *Canonical schemas used in pipelines*

Description

These are documented here for reference and to be used by callers/tests. Keep them minimal; adjust as the package evolves.

Usage

```
zse_schema
```

Format

An object of class `list` of length 2.

`%>%` *Pipe operator*

Description

See `magrittr::%>%` for details.

Index

∗ **datasets**
    zse_schema, 37
∗ **name-matching**
    detect_unmatched_names, 11
    match_names_workflow, 19
    write_unresolved, 37
%>%, *38*, 38

abort_zse, 3
analyze_transcripts, 4
anonymize_educational_data, 5
apply_name_matching, 6
apply_privacy_aware_matching, 7

basic_transcript_analysis, 7
batch_basic_analysis, 8

calculate_content_similarity, 9
consolidate_transcript, 10
convert_error_to_user_friendly, 10

data, 11
detect_unmatched_names, 11, *20*, *37*

engager *(*engager-package*)*, 3
engager-package, 3
ensure_privacy, 12
ethical_compliance, 13

find_function_for_task, 13

generate_privacy_review_report, 14
get_next_level, 15
get_smart_recommendations, 15
get_ux_level, 16
get_visible_functions, 16

load_roster, 17
load_zoom_transcript, 17
lookup_merge_utils, 18

make_blank_cancelled_classes_df, 19
make_blank_section_names_lookup_csv, 19
match_names_workflow, *12*, 19, *37*

plot_users, 20
privacy_audit, 21
privacy_review, 21
process_zoom_transcript, 22

quick_analysis, 23

review_privacy_risks, 24

safe_name_matching_workflow, 25
schema, 26
set_ux_level, 26
show_available_functions, 27
show_error_recovery, 27
show_function_categories, 28
show_function_help, 28
show_getting_started, 29
show_privacy_guidance, 29
show_troubleshooting, 30
show_workflow_help, 30
summarize_transcript_files, 31
summarize_transcript_metrics, 32

user_friendly_error, 33

validate_directory_argument, 34
validate_file_argument, 34
validate_privacy_compliance, 35

write_metrics, 36
write_unresolved, *12*, *20*, 37

zse_schema, 37