

# Data Science Capstone Project

Generated by Doxygen 1.9.8



---

<b>1 Namespace Index</b>	<b>1</b>
1.1 Package List . . . . .	1
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy . . . . .	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List . . . . .	5
<b>4 File Index</b>	<b>7</b>
4.1 File List . . . . .	7
<b>5 Namespace Documentation</b>	<b>9</b>
5.1 BlazorApp Namespace Reference . . . . .	9
5.2 BlazorApp.Controllers Namespace Reference . . . . .	9
5.3 BlazorApp.Hubs Namespace Reference . . . . .	9
5.4 BlazorApp.Models Namespace Reference . . . . .	9
5.5 components Namespace Reference . . . . .	10
5.6 components.book_conversion Namespace Reference . . . . .	10
5.6.1 Variable Documentation . . . . .	10
5.6.1.1 nlp . . . . .	10
5.6.1.2 sentencizer . . . . .	10
5.7 components.connectors Namespace Reference . . . . .	11
5.8 components.document_storage Namespace Reference . . . . .	11
5.8.1 Function Documentation . . . . .	11
5.8.1.1 _docs_to_df() . . . . .	11
5.8.1.2 _find_compatible_nested_key() . . . . .	12
5.8.1.3 _flatten_recursive() . . . . .	12
5.8.1.4 _sanitize_document() . . . . .	13
5.8.1.5 _sanitize_json() . . . . .	13
5.8.1.6 mongo_handle() . . . . .	14
5.9 components.fact_storage Namespace Reference . . . . .	14
5.10 components.metrics Namespace Reference . . . . .	14
5.10.1 Function Documentation . . . . .	15
5.10.1.1 create_summary_payload() . . . . .	15
5.10.1.2 generate_default_metrics() . . . . .	15
5.10.1.3 post_basic_output() . . . . .	15
5.10.1.4 post_example_results() . . . . .	16
5.10.1.5 post_payload() . . . . .	16
5.10.2 Variable Documentation . . . . .	16
5.10.2.1 HOST . . . . .	16
5.10.2.2 url . . . . .	16
5.11 components.text_processing Namespace Reference . . . . .	16
5.11.1 Variable Documentation . . . . .	17

5.11.1.1 nlp . . . . .	17
5.11.1.2 sentencizer . . . . .	17
5.12 src Namespace Reference . . . . .	17
5.13 src.main Namespace Reference . . . . .	17
5.13.1 Function Documentation . . . . .	18
5.13.1.1 chunk_single() . . . . .	18
5.13.1.2 convert_from_csv() . . . . .	18
5.13.1.3 convert_single() . . . . .	18
5.13.1.4 full_pipeline() . . . . .	19
5.13.1.5 graph_triple_files() . . . . .	19
5.13.1.6 output_single() . . . . .	19
5.13.1.7 process_single() . . . . .	19
5.13.1.8 test_relation_extraction() . . . . .	19
5.13.2 Variable Documentation . . . . .	20
5.13.2.1 book_chapters . . . . .	20
5.13.2.2 book_id . . . . .	20
5.13.2.3 book_title . . . . .	20
5.13.2.4 chapters . . . . .	20
5.13.2.5 end . . . . .	20
5.13.2.6 end_str . . . . .	20
5.13.2.7 epub_path . . . . .	20
5.13.2.8 response_files . . . . .	20
5.13.2.9 session . . . . .	21
5.13.2.10 start . . . . .	21
5.13.2.11 start_str . . . . .	21
5.13.2.12 story_id . . . . .	21
5.13.2.13 tei . . . . .	21
5.13.2.14 triple_files . . . . .	21
5.14 src.setup Namespace Reference . . . . .	21
5.14.1 Variable Documentation . . . . .	22
5.14.1.1 session . . . . .	22
5.15 src.util Namespace Reference . . . . .	22
5.15.1 Function Documentation . . . . .	22
5.15.1.1 all_none() . . . . .	22
5.15.1.2 check_values() . . . . .	22
5.15.1.3 df_natural_sorted() . . . . .	23
5.16 tests Namespace Reference . . . . .	23
5.17 tests.conftest Namespace Reference . . . . .	23
5.17.1 Function Documentation . . . . .	23
5.17.1.1 pytest_addoption() . . . . .	23
5.17.1.2 session() . . . . .	24
5.18 tests.test_components Namespace Reference . . . . .	24

---

5.18.1 Function Documentation . . . . .	24
5.18.1.1 <code>_test_query_file()</code> . . . . .	24
5.18.1.2 <code>docs_db()</code> . . . . .	25
5.18.1.3 <code>graph_db()</code> . . . . .	25
5.18.1.4 <code>load_examples_relational()</code> . . . . .	25
5.18.1.5 <code>relational_db()</code> . . . . .	25
5.18.1.6 <code>test_db_docs_comprehensive()</code> . . . . .	25
5.18.1.7 <code>test_db_docs_minimal()</code> . . . . .	25
5.18.1.8 <code>test_db_graph_comprehensive()</code> . . . . .	26
5.18.1.9 <code>test_db_graph_minimal()</code> . . . . .	26
5.18.1.10 <code>test_db_relational_comprehensive()</code> . . . . .	26
5.18.1.11 <code>test_db_relational_minimal()</code> . . . . .	26
5.18.1.12 <code>test_mongo_example_1()</code> . . . . .	26
5.18.1.13 <code>test_mongo_example_2()</code> . . . . .	26
5.18.1.14 <code>test_mongo_example_3()</code> . . . . .	27
5.18.1.15 <code>test_sql_example_1()</code> . . . . .	27
5.18.1.16 <code>test_sql_example_2()</code> . . . . .	27
<b>6 Class Documentation</b>	<b>29</b>
<b>6.1 Book Class Reference</b> . . . . .	<b>29</b>
<b>6.1.1 Constructor &amp; Destructor Documentation</b> . . . . .	<b>29</b>
6.1.1.1 <code>__init__()</code> . . . . .	29
<b>6.1.2 Member Data Documentation</b> . . . . .	29
6.1.2.1 <code>author_key</code> . . . . .	29
6.1.2.2 <code>date_key</code> . . . . .	30
6.1.2.3 <code>language_key</code> . . . . .	30
6.1.2.4 <code>title_key</code> . . . . .	30
<b>6.2 BookFactory Class Reference</b> . . . . .	30
<b>6.2.1 Member Function Documentation</b> . . . . .	31
6.2.1.1 <code>create_book()</code> . . . . .	31
<b>6.3 BookStream Class Reference</b> . . . . .	31
<b>6.3.1 Constructor &amp; Destructor Documentation</b> . . . . .	32
6.3.1.1 <code>__init__()</code> . . . . .	32
<b>6.3.2 Member Function Documentation</b> . . . . .	33
6.3.2.1 <code>stream_segments()</code> . . . . .	33
<b>6.3.3 Member Data Documentation</b> . . . . .	33
6.3.3.1 <code>book</code> . . . . .	33
<b>6.4 Chunk Class Reference</b> . . . . .	33
<b>6.4.1 Detailed Description</b> . . . . .	34
<b>6.4.2 Constructor &amp; Destructor Documentation</b> . . . . .	34
6.4.2.1 <code>__init__()</code> . . . . .	34
<b>6.4.3 Member Function Documentation</b> . . . . .	35

6.4.3.1 <code>__repr__()</code>	35
6.4.3.2 <code>char_count()</code>	35
6.4.4 Member Data Documentation	35
6.4.4.1 <code>text</code>	35
6.5 Connector Class Reference	35
6.5.1 Detailed Description	37
6.5.2 Member Function Documentation	37
6.5.2.1 <code>configure()</code>	37
6.5.2.2 <code>execute_file()</code>	37
6.5.2.3 <code>execute_query()</code>	38
6.5.2.4 <code>test_connection()</code>	38
6.6 DatabaseConnector Class Reference	39
6.6.1 Detailed Description	41
6.6.2 Constructor & Destructor Documentation	41
6.6.2.1 <code>__init__()</code>	41
6.6.3 Member Function Documentation	41
6.6.3.1 <code>_is_single_query()</code>	41
6.6.3.2 <code>_split_combined()</code>	42
6.6.3.3 <code>change_database()</code>	42
6.6.3.4 <code>configure()</code>	42
6.6.3.5 <code>create_database()</code>	43
6.6.3.6 <code>database_exists()</code>	43
6.6.3.7 <code>drop_database()</code>	43
6.6.3.8 <code>execute_combined()</code>	45
6.6.3.9 <code>execute_file()</code>	45
6.6.3.10 <code>execute_query()</code>	46
6.6.3.11 <code>get_dataframe()</code>	46
6.6.4 Member Data Documentation	47
6.6.4.1 <code>connection_string</code>	47
6.6.4.2 <code>db_engine</code>	47
6.6.4.3 <code>db_type</code>	47
6.6.4.4 <code>host</code>	47
6.6.4.5 <code>password</code>	47
6.6.4.6 <code>port</code>	47
6.6.4.7 <code>username</code>	47
6.6.4.8 <code>verbose</code>	47
6.7 DocumentConnector Class Reference	48
6.7.1 Detailed Description	50
6.7.2 Constructor & Destructor Documentation	50
6.7.2.1 <code>__init__()</code>	50
6.7.3 Member Function Documentation	50
6.7.3.1 <code>_split_combined()</code>	50

---

6.7.3.2 change_database()	51
6.7.3.3 check_connection()	51
6.7.3.4 create_database()	52
6.7.3.5 database_exists()	52
6.7.3.6 delete_dummy()	53
6.7.3.7 drop_database()	53
6.7.3.8 execute_query()	53
6.7.3.9 get_dataframe()	54
6.7.3.10 test_connection()	54
6.7.4 Member Data Documentation	55
6.7.4.1 _auth_suffix	55
6.7.4.2 connection_string	55
6.7.4.3 database_name	55
6.7.4.4 verbose	55
6.8 EPUBToTEI Class Reference	55
6.8.1 Detailed Description	56
6.8.2 Constructor & Destructor Documentation	56
6.8.2.1 __init__()	56
6.8.3 Member Function Documentation	56
6.8.3.1 _prune_bad_tags()	56
6.8.3.2 _sanitize_ids()	57
6.8.3.3 clean_teip()	57
6.8.3.4 convert_to_teip()	57
6.8.4 Member Data Documentation	57
6.8.4.1 clean_teip_content	57
6.8.4.2 encoding	57
6.8.4.3 epub_path	57
6.8.4.4 pandoc_xml_path	58
6.8.4.5 raw_teip_content	58
6.8.4.6 save_pandoc	58
6.8.4.7 save_teip	58
6.8.4.8 teip_path	58
6.8.4.9 xml_namespace	58
6.9 Log.Failure Class Reference	58
6.9.1 Constructor & Destructor Documentation	59
6.9.1.1 __init__()	59
6.9.2 Member Function Documentation	59
6.9.2.1 __str__()	59
6.9.3 Member Data Documentation	59
6.9.3.1 msg	59
6.9.3.2 prefix	60
6.10 GraphConnector Class Reference	60

6.10.1 Detailed Description . . . . .	63
6.10.2 Constructor & Destructor Documentation . . . . .	63
6.10.2.1 <code>__init__()</code> . . . . .	63
6.10.3 Member Function Documentation . . . . .	63
6.10.3.1 <code>_split_combined()</code> . . . . .	63
6.10.3.2 <code>add_triple()</code> . . . . .	64
6.10.3.3 <code>change_database()</code> . . . . .	64
6.10.3.4 <code>change_graph()</code> . . . . .	65
6.10.3.5 <code>check_connection()</code> . . . . .	65
6.10.3.6 <code>create_database()</code> . . . . .	65
6.10.3.7 <code>database_exists()</code> . . . . .	66
6.10.3.8 <code>delete_dummy()</code> . . . . .	66
6.10.3.9 <code>drop_database()</code> . . . . .	67
6.10.3.10 <code>execute_query()</code> . . . . .	67
6.10.3.11 <code>get_all_triples()</code> . . . . .	67
6.10.3.12 <code>get_dataframe()</code> . . . . .	68
6.10.3.13 <code>get_edge_counts()</code> . . . . .	68
6.10.3.14 <code>get_unique()</code> . . . . .	69
6.10.3.15 <code>IS_DUMMY_()</code> . . . . .	69
6.10.3.16 <code>NOT_DUMMY_()</code> . . . . .	70
6.10.3.17 <code>print_nodes()</code> . . . . .	70
6.10.3.18 <code>print_triples()</code> . . . . .	70
6.10.3.19 <code>SAME_DB_KG_()</code> . . . . .	70
6.10.3.20 <code>test_connection()</code> . . . . .	70
6.10.4 Member Data Documentation . . . . .	71
6.10.4.1 <code>_created_dummy</code> . . . . .	71
6.10.4.2 <code>connection_string</code> . . . . .	71
6.10.4.3 <code>database_name</code> . . . . .	71
6.10.4.4 <code>graph_name</code> . . . . .	71
6.10.4.5 <code>verbose</code> . . . . .	71
6.11 LLMConnector Class Reference . . . . .	72
6.11.1 Detailed Description . . . . .	73
6.11.2 Constructor & Destructor Documentation . . . . .	73
6.11.2.1 <code>__init__()</code> . . . . .	73
6.11.3 Member Function Documentation . . . . .	73
6.11.3.1 <code>configure()</code> . . . . .	73
6.11.3.2 <code>execute_file()</code> . . . . .	74
6.11.3.3 <code>execute_full_query()</code> . . . . .	74
6.11.3.4 <code>execute_query()</code> . . . . .	74
6.11.3.5 <code>test_connection()</code> . . . . .	75
6.11.4 Member Data Documentation . . . . .	75
6.11.4.1 <code>llm</code> . . . . .	75

---

6.11.4.2 model_name . . . . .	75
6.11.4.3 system_prompt . . . . .	75
6.11.4.4 temperature . . . . .	75
6.12 Log Class Reference . . . . .	75
6.12.1 Detailed Description . . . . .	77
6.12.2 Member Function Documentation . . . . .	78
6.12.2.1 fail() . . . . .	78
6.12.2.2 fail_legacy() . . . . .	79
6.12.2.3 success() . . . . .	79
6.12.2.4 success_legacy() . . . . .	79
6.12.2.5 warn() . . . . .	80
6.12.3 Member Data Documentation . . . . .	80
6.12.3.1 bad_addr . . . . .	80
6.12.3.2 bad_path . . . . .	80
6.12.3.3 bad_val . . . . .	80
6.12.3.4 BRIGHT . . . . .	80
6.12.3.5 conn_abc . . . . .	80
6.12.3.6 create_db . . . . .	81
6.12.3.7 db_conn_abc . . . . .	81
6.12.3.8 db_exists . . . . .	81
6.12.3.9 doc_db . . . . .	81
6.12.3.10 drop_db . . . . .	81
6.12.3.11 FAILURE_COLOR . . . . .	81
6.12.3.12 FULL_DF . . . . .	81
6.12.3.13 get_df . . . . .	81
6.12.3.14 get_unique . . . . .	81
6.12.3.15 good_val . . . . .	82
6.12.3.16 gr_db . . . . .	82
6.12.3.17 GREEN . . . . .	82
6.12.3.18 kg . . . . .	82
6.12.3.19 msg_bad_addr . . . . .	82
6.12.3.20 msg_bad_coll . . . . .	82
6.12.3.21 msg_bad_exec_f . . . . .	82
6.12.3.22 msg_bad_exec_q . . . . .	82
6.12.3.23 msg_bad_graph . . . . .	82
6.12.3.24 msg_bad_path . . . . .	82
6.12.3.25 msg_bad_table . . . . .	83
6.12.3.26 MSG_COLOR . . . . .	83
6.12.3.27 msg_compare . . . . .	83
6.12.3.28 msg_db_connect . . . . .	83
6.12.3.29 msg_db_current . . . . .	83
6.12.3.30 msg_db_exists . . . . .	83

6.12.3.31 msg_db_not_found . . . . .	83
6.12.3.32 msg_fail_manage_db . . . . .	83
6.12.3.33 msg_fail_parse . . . . .	84
6.12.3.34 msg_good_coll . . . . .	84
6.12.3.35 msg_good_exec_f . . . . .	84
6.12.3.36 msg_good_exec_q . . . . .	84
6.12.3.37 msg_good_exec_qr . . . . .	84
6.12.3.38 msg_good_graph . . . . .	84
6.12.3.39 msg_good_path . . . . .	84
6.12.3.40 msg_good_table . . . . .	84
6.12.3.41 msg_multiple_query . . . . .	85
6.12.3.42 msg_result . . . . .	85
6.12.3.43 msg_success_managed_db . . . . .	85
6.12.3.44 msg_swap_db . . . . .	85
6.12.3.45 msg_swap_kg . . . . .	85
6.12.3.46 msg_unknown_error . . . . .	85
6.12.3.47 pytest_db . . . . .	85
6.12.3.48 RED . . . . .	85
6.12.3.49 rel_db . . . . .	86
6.12.3.50 run_f . . . . .	86
6.12.3.51 run_q . . . . .	86
6.12.3.52 SUCCESS_COLOR . . . . .	86
6.12.3.53 swap_db . . . . .	86
6.12.3.54 swap_kg . . . . .	86
6.12.3.55 test_basic . . . . .	86
6.12.3.56 test_conn . . . . .	86
6.12.3.57 test_df . . . . .	86
6.12.3.58 test_info . . . . .	86
6.12.3.59 test_tmp_db . . . . .	87
6.12.3.60 USE_COLORS . . . . .	87
6.12.3.61 WARNING_COLOR . . . . .	87
6.12.3.62 WHITE . . . . .	87
6.12.3.63 YELLOW . . . . .	87
6.13 MetricsController Class Reference . . . . .	88
6.13.1 Constructor & Destructor Documentation . . . . .	89
6.13.1.1 MetricsController() . . . . .	89
6.13.2 Member Function Documentation . . . . .	89
6.13.2.1 GetAll() . . . . .	89
6.13.2.2 GetIndex() . . . . .	89
6.13.2.3 Post() . . . . .	89
6.13.3 Member Data Documentation . . . . .	89
6.13.3.1 _hubContext . . . . .	89

---

6.13.3.2 <code>_logger</code>	89
6.13.3.3 <code>Summaries</code>	89
6.14 MetricsHub Class Reference	90
6.14.1 Constructor & Destructor Documentation	90
6.14.1.1 <code>MetricsHub()</code>	90
6.14.2 Member Function Documentation	91
6.14.2.1 <code>OnConnectedAsync()</code>	91
6.14.2.2 <code>OnDisconnectedAsync()</code>	91
6.14.3 Member Data Documentation	91
6.14.3.1 <code>_logger</code>	91
6.15 mysqlConnector Class Reference	91
6.15.1 Detailed Description	94
6.15.2 Constructor & Destructor Documentation	94
6.15.2.1 <code>__init__()</code>	94
6.15.3 Member Data Documentation	94
6.15.3.1 <code>specific_queries</code>	94
6.16 ParagraphStreamTEI Class Reference	94
6.16.1 Detailed Description	96
6.16.2 Constructor & Destructor Documentation	96
6.16.2.1 <code>__init__()</code>	96
6.16.3 Member Function Documentation	97
6.16.3.1 <code>pre_compute_segments()</code>	97
6.16.3.2 <code>stream_segments()</code>	97
6.16.4 Member Data Documentation	97
6.16.4.1 <code>allowed_chapters</code>	97
6.16.4.2 <code>book_id</code>	97
6.16.4.3 <code>chunks</code>	97
6.16.4.4 <code>encoding</code>	98
6.16.4.5 <code>end_inclusive</code>	98
6.16.4.6 <code>lines</code>	98
6.16.4.7 <code>root</code>	98
6.16.4.8 <code>start_inclusive</code>	98
6.16.4.9 <code>story_id</code>	98
6.16.4.10 <code>tei_path</code>	98
6.16.4.11 <code>xml_namespace [1/2]</code>	98
6.16.4.12 <code>xml_namespace [2/2]</code>	98
6.17 postgresConnector Class Reference	99
6.17.1 Detailed Description	102
6.17.2 Constructor & Destructor Documentation	102
6.17.2.1 <code>__init__()</code>	102
6.17.3 Member Data Documentation	102
6.17.3.1 <code>specific_queries</code>	102

---

6.18 PRF1Metric Class Reference . . . . .	102
6.18.1 Property Documentation . . . . .	103
6.18.1.1 F1Score . . . . .	103
6.18.1.2 Name . . . . .	103
6.18.1.3 Precision . . . . .	103
6.18.1.4 Recall . . . . .	103
6.19 QAItem Class Reference . . . . .	103
6.19.1 Property Documentation . . . . .	103
6.19.1.1 Accuracy . . . . .	103
6.19.1.2 GeneratedAnswer . . . . .	103
6.19.1.3 GoldAnswer . . . . .	104
6.19.1.4 IsCorrect . . . . .	104
6.19.1.5 Question . . . . .	104
6.20 QAMetric Class Reference . . . . .	104
6.20.1 Property Documentation . . . . .	104
6.20.1.1 AverageAccuracy . . . . .	104
6.20.1.2 QAItems . . . . .	104
6.21 RelationalConnector Class Reference . . . . .	105
6.21.1 Detailed Description . . . . .	107
6.21.2 Constructor & Destructor Documentation . . . . .	107
6.21.2.1 __init__(). . . . .	107
6.21.3 Member Function Documentation . . . . .	108
6.21.3.1 _split_combined(). . . . .	108
6.21.3.2 change_database(). . . . .	108
6.21.3.3 check_connection(). . . . .	108
6.21.3.4 create_database(). . . . .	109
6.21.3.5 database_exists(). . . . .	109
6.21.3.6 drop_database(). . . . .	110
6.21.3.7 execute_query(). . . . .	110
6.21.3.8 from_env(). . . . .	111
6.21.3.9 get_dataframe(). . . . .	111
6.21.3.10 test_connection(). . . . .	111
6.21.4 Member Data Documentation . . . . .	112
6.21.4.1 connection_string . . . . .	112
6.21.4.2 database_name . . . . .	112
6.21.4.3 db_type . . . . .	112
6.21.4.4 verbose . . . . .	112
6.22 RelationExtractor Class Reference . . . . .	112
6.22.1 Constructor & Destructor Documentation . . . . .	113
6.22.1.1 __init__(). . . . .	113
6.22.2 Member Function Documentation . . . . .	113
6.22.2.1 extract() . . . . .	113

---

6.22.3 Member Data Documentation . . . . .	113
6.22.3.1 max_tokens . . . . .	113
6.22.3.2 model . . . . .	113
6.22.3.3 tokenizer . . . . .	113
6.22.3.4 tuple_delim . . . . .	113
6.23 ScalarMetric Class Reference . . . . .	114
6.23.1 Property Documentation . . . . .	114
6.23.1.1 Name . . . . .	114
6.23.1.2 Value . . . . .	114
6.24 Session Class Reference . . . . .	114
6.24.1 Detailed Description . . . . .	115
6.24.2 Constructor & Destructor Documentation . . . . .	115
6.24.2.1 __init__(). . . . .	115
6.24.3 Member Function Documentation . . . . .	115
6.24.3.1 __new__(). . . . .	115
6.24.3.2 reset(). . . . .	115
6.24.3.3 test_database_connections(). . . . .	116
6.24.4 Member Data Documentation . . . . .	116
6.24.4.1 _instance . . . . .	116
6.24.4.2 docs_db . . . . .	116
6.24.4.3 graph_db . . . . .	116
6.24.4.4 relational_db . . . . .	116
6.24.4.5 verbose . . . . .	116
6.25 Story Class Reference . . . . .	117
6.25.1 Constructor & Destructor Documentation . . . . .	117
6.25.1.1 __init__(). . . . .	117
6.25.2 Member Function Documentation . . . . .	117
6.25.2.1 _make_single(). . . . .	117
6.25.2.2 _merge_chunks(). . . . .	117
6.25.2.3 pre_split_chunks(). . . . .	118
6.25.2.4 stream_chunks(). . . . .	118
6.25.3 Member Data Documentation . . . . .	118
6.25.3.1 reader . . . . .	118
6.26 StoryStreamAdapter Class Reference . . . . .	118
6.26.1 Member Function Documentation . . . . .	119
6.26.1.1 stream_paragraphs(). . . . .	119
6.26.1.2 stream_segments(). . . . .	119
6.26.1.3 stream_sentences(). . . . .	120
6.27 SummaryData Class Reference . . . . .	120
6.27.1 Property Documentation . . . . .	120
6.27.1.1 BookID . . . . .	120
6.27.1.2 BookTitle . . . . .	120

6.27.1.3 Metrics . . . . .	120
6.27.1.4 QAResults . . . . .	120
6.27.1.5 SummaryText . . . . .	121
6.28 SummaryMetrics Class Reference . . . . .	121
6.28.1 Member Function Documentation . . . . .	121
6.28.1.1 GetDefault() . . . . .	121
6.28.2 Property Documentation . . . . .	121
6.28.2.1 PRF1Metrics . . . . .	121
6.28.2.2 QA . . . . .	121
6.28.2.3 ScalarMetrics . . . . .	121
<b>7 File Documentation</b> . . . . .	<b>123</b>
7.1 /home/runner/work/dsci-capstone/dsci-capstone/components/book_conversion.py File Reference . . . . .	123
7.2 /home/runner/work/dsci-capstone/dsci-capstone/components/connectors.py File Reference . . . . .	124
7.3 /home/runner/work/dsci-capstone/dsci-capstone/components/document_storage.py File Reference . . . . .	124
7.4 /home/runner/work/dsci-capstone/dsci-capstone/components/fact_storage.py File Reference . . . . .	125
7.5 /home/runner/work/dsci-capstone/dsci-capstone/components/metrics.py File Reference . . . . .	125
7.6 /home/runner/work/dsci-capstone/dsci-capstone/components/semantic_web.py File Reference . . . . .	126
7.7 /home/runner/work/dsci-capstone/dsci-capstone/components/text_processing.py File Reference . . . . .	126
7.8 /home/runner/work/dsci-capstone/dsci-capstone/components/__init__.py File Reference . . . . .	126
7.9 /home/runner/work/dsci-capstone/dsci-capstone/src/__init__.py File Reference . . . . .	126
7.10 /home/runner/work/dsci-capstone/dsci-capstone/tests/__init__.py File Reference . . . . .	126
7.11 /home/runner/work/dsci-capstone/dsci-capstone/src/main.py File Reference . . . . .	127
7.12 /home/runner/work/dsci-capstone/dsci-capstone/src/setup.py File Reference . . . . .	127
7.13 /home/runner/work/dsci-capstone/dsci-capstone/src/util.py File Reference . . . . .	128
7.14 /home/runner/work/dsci-capstone/dsci-capstone/tests/conftest.py File Reference . . . . .	128
7.15 /home/runner/work/dsci-capstone/dsci-capstone/tests/test_components.py File Reference . . . . .	129
7.16 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/_Imports.razor File Reference . . . . .	130
7.17 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/App.razor File Reference . . . . .	130
7.18 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Layout/MainLayout.razor File Reference . . . . .	130
7.19 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Layout/NavMenu.razor File Reference . . . . .	130
7.20 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Pages/Error.razor File Reference . . . . .	130
7.21 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Pages/Graph.razor File Reference . . . . .	130
7.22 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Pages/Home.razor File Reference . . . . .	130
7.23 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Pages/Metrics.razor File Reference . . . . .	130
7.24 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Routes.razor File Reference . . . . .	130

---

7.25 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Controllers/MetricsController.cs File Reference . . . . .	130
7.26 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Hubs/MetricsHub.cs File Reference . . . . .	131
7.27 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/PRF1Metric.cs File Reference . . . . .	131
7.28 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/QAItem.cs File Reference . . . . .	131
7.29 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/QAMetric.cs File Reference . . . . .	131
7.30 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/ScalarMetric.cs File Reference . . . . .	132
7.31 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/SummaryData.cs File Reference . . . . .	132
7.32 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/SummaryMetrics.cs File Reference . . . . .	132
<b>Index</b>	<b>133</b>



# Chapter 1

## Namespace Index

### 1.1 Package List

Here are the packages with brief descriptions (if available):

BlazorApp . . . . .	9
BlazorApp.Controllers . . . . .	9
BlazorApp.Hubs . . . . .	9
BlazorApp.Models . . . . .	9
components . . . . .	10
components.book_conversion . . . . .	10
components.connectors . . . . .	11
components.document_storage . . . . .	11
components.fact_storage . . . . .	14
components.metrics . . . . .	14
components.text_processing . . . . .	16
src . . . . .	17
src.main . . . . .	17
src.setup . . . . .	21
src.util . . . . .	22
tests . . . . .	23
tests.conftest . . . . .	23
tests.test_components . . . . .	24



# Chapter 2

## Hierarchical Index

### 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Book . . . . .	29
Chunk . . . . .	33
ControllerBase	
MetricsController . . . . .	88
EPUBToTEI . . . . .	55
Hub	
MetricsHub . . . . .	90
Log . . . . .	75
PRF1Metric . . . . .	102
QAItem . . . . .	103
QAMetric . . . . .	104
RelationExtractor . . . . .	112
RuntimeError	
Log.Failure . . . . .	58
ScalarMetric . . . . .	114
Session . . . . .	114
Story . . . . .	117
SummaryData . . . . .	120
SummaryMetrics . . . . .	121
ABC	
BookFactory . . . . .	30
StoryStreamAdapter	
BookStream . . . . .	31
ParagraphStreamTEI . . . . .	94
Connector . . . . .	35
DatabaseConnector . . . . .	39
RelationalConnector . . . . .	105
mysqlConnector . . . . .	91
postgresConnector . . . . .	99
DocumentConnector . . . . .	48
GraphConnector . . . . .	60
LLMConnector . . . . .	72



# Chapter 3

## Class Index

### 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">Book</a>	.....	29
<a href="#">BookFactory</a>	.....	30
<a href="#">BookStream</a>	.....	31
<a href="#">Chunk</a>	Lightweight container for a span of story text	33
<a href="#">Connector</a>	Abstract base class for external connectors	35
<a href="#">DatabaseConnector</a>	Abstract base class for database engine connectors	39
<a href="#">DocumentConnector</a>	Connector for MongoDB (document database)	48
<a href="#">EPUBToTEI</a>	Converts EPUB files to XML format (TEI specification)	55
<a href="#">Log.Failure</a>	.....	58
<a href="#">GraphConnector</a>	Connector for Neo4j (graph database)	60
<a href="#">LLMConnector</a>	Connector for prompting and returning LLM output (raw text/JSON) via LangChain	72
<a href="#">Log</a>	Standardizes console output	75
<a href="#">MetricsController</a>	.....	88
<a href="#">MetricsHub</a>	.....	90
<a href="#">mysqlConnector</a>	A relational database connector configured for MySQL	91
<a href="#">ParagraphStreamTEI</a>	Streams paragraphs from a TEI file as Chunk objects	94
<a href="#">postgresConnector</a>	A relational database connector configured for PostgreSQL	99
<a href="#">PRF1Metric</a>	.....	102
<a href="#">QALtem</a>	.....	103
<a href="#">QAMetric</a>	.....	104
<a href="#">RelationalConnector</a>	Connector for relational databases (MySQL, PostgreSQL)	105
<a href="#">RelationExtractor</a>	.....	112
<a href="#">ScalarMetric</a>	.....	114

<a href="#">Session</a>	
Stores active database connections and configuration settings	114
<a href="#">Story</a>	117
<a href="#">StoryStreamAdapter</a>	118
<a href="#">SummaryData</a>	120
<a href="#">SummaryMetrics</a>	121

# Chapter 4

## File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

/home/runner/work/dsci-capstone/dsci-capstone/components/__init__.py . . . . .	126
/home/runner/work/dsci-capstone/dsci-capstone/components/book_conversion.py . . . . .	123
/home/runner/work/dsci-capstone/dsci-capstone/components/connectors.py . . . . .	124
/home/runner/work/dsci-capstone/dsci-capstone/components/document_storage.py . . . . .	124
/home/runner/work/dsci-capstone/dsci-capstone/components/fact_storage.py . . . . .	125
/home/runner/work/dsci-capstone/dsci-capstone/components/metrics.py . . . . .	125
/home/runner/work/dsci-capstone/dsci-capstone/components/semantic_web.py . . . . .	126
/home/runner/work/dsci-capstone/dsci-capstone/components/text_processing.py . . . . .	126
/home/runner/work/dsci-capstone/dsci-capstone/src/__init__.py . . . . .	126
/home/runner/work/dsci-capstone/dsci-capstone/src/main.py . . . . .	127
/home/runner/work/dsci-capstone/dsci-capstone/src/setup.py . . . . .	127
/home/runner/work/dsci-capstone/dsci-capstone/src/util.py . . . . .	128
/home/runner/work/dsci-capstone/dsci-capstone/tests/__init__.py . . . . .	126
/home/runner/work/dsci-capstone/dsci-capstone/tests/conftest.py . . . . .	128
/home/runner/work/dsci-capstone/dsci-capstone/tests/test_components.py . . . . .	129
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/_Imports.razor . . . . .	130
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/App.razor . . . . .	130
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Routes.razor . . . . .	130
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Layout/MainLayout.razor 130	
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Layout/NavMenu.razor 130	
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Pages/Error.razor . . . . .	130
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Pages/Graph.razor . . . . .	130
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Pages/Home.razor . . . . .	130
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Components/Pages/Metrics.razor . . . . .	130
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Controllers/MetricsController.cs . . . . .	130
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Hubs/MetricsHub.cs . . . . .	131
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/PRF1Metric.cs . . . . .	131
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/QAItem.cs . . . . .	131
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/QAMetric.cs . . . . .	131
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/ScalarMetric.cs . . . . .	132
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/SummaryData.cs . . . . .	132
/home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/SummaryMetrics.cs . . . . .	132



# Chapter 5

## Namespace Documentation

### 5.1 BlazorApp Namespace Reference

#### Namespaces

- namespace [Controllers](#)
- namespace [Hubs](#)
- namespace [Models](#)

### 5.2 BlazorApp.Controllers Namespace Reference

#### Classes

- class [MetricsController](#)

### 5.3 BlazorApp.Hubs Namespace Reference

#### Classes

- class [MetricsHub](#)

### 5.4 BlazorApp.Models Namespace Reference

#### Classes

- class [PRF1Metric](#)
- class [QALtem](#)
- class [QAMetric](#)
- class [ScalarMetric](#)
- class [SummaryData](#)
- class [SummaryMetrics](#)

## 5.5 components Namespace Reference

### Namespaces

- namespace [book\\_conversion](#)
- namespace [connectors](#)
- namespace [document\\_storage](#)
- namespace [fact\\_storage](#)
- namespace [metrics](#)
- namespace [text\\_processing](#)

## 5.6 components.book\_conversion Namespace Reference

### Classes

- class [Book](#)  
*Lightweight container for a span of story text.*
- class [BookFactory](#)
- class [BookStream](#)
- class [Chunk](#)  
*Converts EPUB files to XML format (TEI specification).*
- class [EPUBToTEI](#)  
*Streams paragraphs from a TEI file as Chunk objects.*
- class [ParagraphStreamTEI](#)
- class [Story](#)
- class [StoryStreamAdapter](#)

### Variables

- `nlp = spacy.blank("en")`
- `sentencizer = nlp.add_pipe("sentencizer")`

### 5.6.1 Variable Documentation

#### 5.6.1.1 `nlp`

```
nlp = spacy.blank("en")
```

#### 5.6.1.2 `sentencizer`

```
sentencizer = nlp.add_pipe("sentencizer")
```

## 5.7 components.connectors Namespace Reference

### Classes

- class [Connector](#)  
*Abstract base class for external connectors.*
- class [DatabaseConnector](#)  
*Abstract base class for database engine connectors.*
- class [mysqlConnector](#)  
*A relational database connector configured for MySQL.*
- class [postgresConnector](#)  
*A relational database connector configured for PostgreSQL.*
- class [RelationalConnector](#)  
*Connector for relational databases (MySQL, PostgreSQL).*

## 5.8 components.document\_storage Namespace Reference

### Classes

- class [DocumentConnector](#)  
*Connector for MongoDB (document database)*

### Functions

- Generator[Database[Any], None, None] [mongo\\_handle](#) (str host, str alias)  
*Establish a temporary connection to MongoDB.*
- DataFrame [\\_flatten\\_recursive](#) (DataFrame df)  
*Explode all list columns and flatten dict columns until only scalars remain.*
- str [\\_sanitize\\_json](#) (str text)  
*Remove comments and other non-JSON content from a MongoDB query string.*
- Dict[str, Any] [\\_sanitize\\_document](#) (Dict[str, Any] doc, Dict[str, Set[Type[Any]]] type\_registry)  
*Normalize document fields to consistent types for DataFrame construction.*
- DataFrame [\\_docs\\_to\\_df](#) (List[Dict[str, Any]] docs, bool merge\_unspecified=True)  
*Convert raw MongoDB documents to a Pandas DataFrame.*
- str [\\_find\\_compatible\\_nested\\_key](#) (Type[Any] value\_type, Dict[str, Set[Type[Any]]] nested\_schema, bool merge\_unspecified)  
*Find a nested column compatible with the given primitive type.*

### 5.8.1 Function Documentation

#### 5.8.1.1 [\\_docs\\_to\\_df\(\)](#)

```
DataFrame _docs_to_df (
    List[Dict[str, Any]] docs,
    bool merge_unspecified = True ) [protected]
```

Convert raw MongoDB documents to a Pandas DataFrame.

Handles schema inconsistencies by:

1. First pass: identify all nested column names and their types
2. Second pass: sanitize and wrap primitives using type-compatible nested columns
3. Flatten structures into final DataFrame

**Parameters**

<i>docs</i>	List of MongoDB documents to convert.
<i>merge_unspecified</i>	If True, merge primitives into type-compatible nested columns using aggressive type casting (int→float, bool→int→float). If False, keep as <code>_unspecified_type</code> columns.

**Exceptions**

<i>Log.Failure</i>	If parsing query results to JSON fails.
--------------------	---

**5.8.1.2 `_find_compatible_nested_key()`**

```
str _find_compatible_nested_key (
    Type[Any] value_type,
    Dict[str, Set[Type[Any]]] nested_schema,
    bool merge_unspecified ) [protected]
```

Find a nested column compatible with the given primitive type.

Uses type compatibility hierarchy for aggressive merging: bool → int → float (numeric types) str (isolated, only matches str) Searches for exact match first, then compatible types.

**Parameters**

<i>value_type</i>	The type of the primitive value to map (e.g., str, int, float).
<i>nested_schema</i>	Dict mapping nested keys to sets of observed types.
<i>merge_unspecified</i>	Whether to attempt type-compatible merging.

**Returns**

The nested key name to use for wrapping the primitive.

**5.8.1.3 `_flatten_recursive()`**

```
DataFrame _flatten_recursive (
    DataFrame df ) [protected]
```

Explode all list columns and flatten dict columns until only scalars remain.

Recursive Process:

1. Find columns containing lists → explode to create new rows
2. Find columns containing dicts → normalize to create new columns
3. Repeat until no lists or dicts remain

**Parameters**

<i>df</i>	DataFrame with potentially nested structures.
-----------	---

**Returns**

Fully flattened DataFrame with only scalar values.

**5.8.1.4 `_sanitize_document()`**

```
Dict[str, Any] _sanitize_document (
    Dict[str, Any] doc,
    Dict[str, Set[Type[Any]]] type_registry ) [protected]
```

Normalize document fields to consistent types for DataFrame construction.

Converts all field values to lists and tracks type patterns.

- Objectid → string
- Single value → [value]
- Mixed types tracked in type\_registry for conflict resolution

**Parameters**

<i>doc</i>	MongoDB document to sanitize.
<i>type_registry</i>	Tracks observed types per field path (e.g., {"effects": {str, list}}).

**Returns**

Document with all fields as lists.

**5.8.1.5 `_sanitize_json()`**

```
str _sanitize_json (
    str text ) [protected]
```

Remove comments and other non-JSON content from a MongoDB query string.

Removes the following elements:

- Block comments /\* ... \*/
- Single-line comments //
- Half-line comments ... //
- Trailing commas before closing braces
- Newlines and whitespace Preserves bad text inside JSON string values.

**Parameters**

<i>text</i>	Raw text that may contain comments.
-------------	-------------------------------------

**Returns**

Cleaned text suitable for JSON parsing.

**5.8.1.6 mongo\_handle()**

```
Generator[Database[Any], None, None] mongo_handle (
    str host,
    str alias )
```

Establish a temporary connection to MongoDB.

**Parameters**

<i>host</i>	A valid MongoDB connection string.
<i>alias</i>	A unique name for the usage of this connection.

Allows scoped access to the low-level PyMongo handle from MongoEngine. Usage: `with mongo_handle(host=self.connection_string, alias="create_db") as db: (your code here...)` This will disconnect all connections on the alias once finished. Helpful when `test_connection` wants to call `execute_query`, but continue using its existing db handle after `execute_query` disconnects.

**5.9 components.fact\_storage Namespace Reference****Classes**

- class [GraphConnector](#)  
*Connector for Neo4j (graph database).*

**5.10 components.metrics Namespace Reference****Functions**

- [generate\\_default\\_metrics](#) (`rouge_precision=0.0, rouge_recall=0.0, rouge_f1=0.0, bert_precision=0.0, bert_recall=0.0, bert_f1=0.0, boook_score=0.0, questeval_score=0.0, qa_question1="UNKNOWN", qa_gold1="UNKNOWN", qa_generated1="UNKNOWN", qa_correct1=False, qa_accuracy1=0.0, qa_question2="UNKNOWN", qa_gold2="UNKNOWN", qa_generated2="UNKNOWN", qa_correct2=False, qa_accuracy2=0.0`)  
*Generate metrics payload with customizable default values.*
- [create\\_summary\\_payload](#) (`book_id, book_title, summary, metrics=None`)  
*Create the full summary payload for the API.*
- [post\\_payload](#) (`payload`)  
*Verify and post any given payload using the requests API.*
- [post\\_example\\_results](#) ()  
*Send placeholder values to the web app.*
- [post\\_basic\\_output](#) (`book_id, book_title, summary, **kwargs`)  
*Send book information and a summary to the web app.*

## Variables

- `HOST` = `os.getenv("BLAZOR_HOST")`
- str `url` = `f"http://{{HOST}}:5055/api/metrics"`

### 5.10.1 Function Documentation

#### 5.10.1.1 `create_summary_payload()`

```
create_summary_payload (
    book_id,
    book_title,
    summary,
    metrics = None )
```

Create the full summary payload for the API.

#### 5.10.1.2 `generate_default_metrics()`

```
generate_default_metrics (
    rouge_precision = 0.0,
    rouge_recall = 0.0,
    rouge_f1 = 0.0,
    bert_precision = 0.0,
    bert_recall = 0.0,
    bert_f1 = 0.0,
    boooook_score = 0.0,
    questeval_score = 0.0,
    qa_question1 = "UNKNOWN",
    qa_gold1 = "UNKNOWN",
    qa_generated1 = "UNKNOWN",
    qa_correct1 = False,
    qa_accuracy1 = 0.0,
    qa_question2 = "UNKNOWN",
    qa_gold2 = "UNKNOWN",
    qa_generated2 = "UNKNOWN",
    qa_correct2 = False,
    qa_accuracy2 = 0.0 )
```

Generate metrics payload with customizable default values.

#### 5.10.1.3 `post_basic_output()`

```
post_basic_output (
    book_id,
    book_title,
    summary,
    ** kwargs )
```

Send book information and a summary to the web app.

#### Parameters

<code>book_id</code>	Integer book identifier
<code>book_title</code>	Book title
<code>summary</code>	Summary text string
<code>**kwargs</code>	Any other metric parameters to override defaults (e.g., rouge_f1=0.75)

#### 5.10.1.4 `post_example_results()`

```
post_example_results ( )
```

Send placeholder values to the web app.

#### 5.10.1.5 `post_payload()`

```
post_payload (
    payload )
```

Verify and post any given payload using the requests API.

### 5.10.2 Variable Documentation

#### 5.10.2.1 `HOST`

```
HOST = os.getenv(f"BLAZOR_HOST")
```

#### 5.10.2.2 `url`

```
str url = f"http://{{HOST}}:5055/api/metrics"
```

## 5.11 `components.text_processing` Namespace Reference

### Classes

- class [LLMConnector](#)  
*Connector for prompting and returning LLM output (raw text/JSON) via LangChain.*
- class [RelationExtractor](#)

### Variables

- `nlp` = spacy.blank("en")
- `sentencizer` = nlp.add\_pipe("sentencizer")

## 5.11.1 Variable Documentation

### 5.11.1.1 nlp

```
nlp = spacy.blank("en")
```

### 5.11.1.2 sentencizer

```
sentencizer = nlp.add_pipe("sentencizer")
```

## 5.12 src Namespace Reference

### Namespaces

- namespace [main](#)
- namespace [setup](#)
- namespace [util](#)

## 5.13 src.main Namespace Reference

### Functions

- [convert\\_single \(\)](#)  
*Converts one EPUB file to TEI format.*
- [convert\\_from\\_csv \(\)](#)  
*Converts several EPUB files to TEI format.*
- [chunk\\_single \(\)](#)  
*Creates a Story and many Chunks from a TEI file.*
- [test\\_relation\\_extraction \(\)](#)  
*Runs REBEL on a basic example; used for debugging.*
- [process\\_single \(\)](#)  
*Uses NLP and LLM to process an existing TEI file.*
- [graph\\_triple\\_files \(\)](#)  
*Loads JSON into Neo4j to test the Blazor graph page.*
- [output\\_single \(\)](#)  
*Generates a summary from triples stored in JSON, and posts data to Blazor.*
- [full\\_pipeline \(epub\\_path, book\\_chapters, start\\_str, end\\_str, book\\_id, story\\_id, book\\_title\)](#)  
*Connects all components to convert an EPUB file to a book summary.*

## Variables

- `session = Session(verbose=False)`
- `str tei = "./datasets/examples/trilogy-wishes-1.tei"`  
*Will revisit later - Book classes need refactoring ####.*
- `str chapters`
- `str start = ""`
- `str end = "But I must say no more."`
- `list triple_files`
- `list response_files = ["./datasets/triples/chunk-160_story-1.txt"]`
- `epub_path`
- `book_chapters`
- `start_str`
- `end_str`
- `book_id`
- `story_id`
- `book_title`

## 5.13.1 Function Documentation

### 5.13.1.1 chunk\_single()

`chunk_single ( )`

Creates a Story and many Chunks from a TEI file.

Requires hard-coded specificaitons

- List of all chapter names.
- Optional start / end strings.

### 5.13.1.2 convert\_from\_csv()

`convert_from_csv ( )`

Converts several EPUB files to TEI format.

#### Note

Files are specified as rows in a CSV which contains parsing instructions.

### 5.13.1.3 convert\_single()

`convert_single ( )`

Converts one EPUB file to TEI format.

#### 5.13.1.4 full\_pipeline()

```
full_pipeline (
    epub_path,
    book_chapters,
    start_str,
    end_str,
    book_id,
    story_id,
    book_title )
```

Connects all components to convert an EPUB file to a book summary.

Data conversions

- EPUB file
- XML (TEI)
- JSON triples (NLP & LLM)
- Neo4j graph database
- Output summary
- Blazor graph and metrics pages

#### 5.13.1.5 graph\_triple\_files()

```
graph_triple_files ( )
```

Loads JSON into Neo4j to test the Blazor graph page.

#### 5.13.1.6 output\_single()

```
output_single ( )
```

Generates a summary from triples stored in JSON, and posts data to Blazor.

#### 5.13.1.7 process\_single()

```
process_single ( )
```

Uses NLP and LLM to process an existing TEI file.

#### 5.13.1.8 test\_relation\_extraction()

```
test_relation_extraction ( )
```

Runs REBEL on a basic example; used for debugging.

## 5.13.2 Variable Documentation

### 5.13.2.1 book\_chapters

```
book_chapters
```

### 5.13.2.2 book\_id

```
book_id
```

### 5.13.2.3 book\_title

```
book_title
```

### 5.13.2.4 chapters

```
str chapters
```

#### Initial value:

```
00001 = """
00002 CHAPTER 1 BEAUTIFUL AS THE DAY\n
00003 CHAPTER 2 GOLDEN GUINEAS\n
00004 CHAPTER 3 BEING WANTED\n
00005 CHAPTER 4 WINGS\n
00006 CHAPTER 5 NO WINGS\n
00007 CHAPTER 6 A CASTLE AND NO DINNER\n
00008 CHAPTER 7 A SIEGE AND BED\n
00009 CHAPTER 8 BIGGER THAN THE BAKER'S BOY\n
00010 CHAPTER 9 GROWN UP\n
00011 CHAPTER 10 SCALPS\n
00012 CHAPTER 11 THE LAST WISH\n
00013 """
```

### 5.13.2.5 end

```
str end = "But I must say no more."
```

### 5.13.2.6 end\_str

```
end_str
```

### 5.13.2.7 epub\_path

```
epub_path
```

### 5.13.2.8 response\_files

```
list response_files = ["./datasets/triples/chunk-160_story-1.txt"]
```

### 5.13.2.9 session

```
session = Session(verbose=False)
```

### 5.13.2.10 start

```
str start = ""
```

### 5.13.2.11 start\_str

```
start_str
```

### 5.13.2.12 story\_id

```
story_id
```

### 5.13.2.13 tei

```
str tei = "./datasets/examples/trilogy-wishes-1.tei"
```

Will revisit later - Book classes need refactoring ###.

### 5.13.2.14 triple\_files

```
list triple_files
```

#### Initial value:

```
00001 = [
00002     "./datasets/triples/chunk-160_story-1.json",
00003     "./datasets/triples/chunk-70_story-1.json",
00004 ]
```

## 5.14 src.setup Namespace Reference

### Classes

- class [Session](#)  
*Stores active database connections and configuration settings.*

### Variables

- [session = Session\(\)](#)

## 5.14.1 Variable Documentation

### 5.14.1.1 session

```
session = Session()
```

## 5.15 src.util Namespace Reference

### Classes

- class [Log](#)

*The Log class standardizes console output.*

### Functions

- [all\\_none](#) (\*args)  
*Checks if all provided args are None.*
- DataFrame [df\\_natural\\_sorted](#) (DataFrame df, List[str] ignored\_columns=[])  
*Sort a DataFrame in natural order using only certain columns.*
- bool [check\\_values](#) (List[Any] results, List[Any] expected, bool verbose, str log\_source, bool raise\_error)  
*Safely compare two lists of values.*

## 5.15.1 Function Documentation

### 5.15.1.1 all\_none()

```
all_none (
    * args )
```

Checks if all provided args are None.

### 5.15.1.2 check\_values()

```
bool check_values (
    List[Any] results,
    List[Any] expected,
    bool verbose,
    str log_source,
    bool raise_error )
```

Safely compare two lists of values.

Helper for [components.connectors.RelationalConnector.test\\_connection](#)

#### Parameters

<i>results</i>	A list of observed values from the database.
<i>expected</i>	A list of correct values to compare against.
<i>verbose</i>	Whether to print success messages.
<i>log_source</i>	The Log class prefix indicating which method is performing the check.
<i>raise_error</i>	Whether to raise an error on connection failure.

### Exceptions

<i>LogFailure</i>	If any result does not match what was expected.
-------------------	---

#### 5.15.1.3 df\_natural\_sorted()

```
DataFrame df_natural_sorted (
    DataFrame df,
    List[str] ignored_columns = [] )
```

Sort a DataFrame in natural order using only certain columns.

- The provided DataFrame will not be modified, since `inplace=False` by default.
- Existing row numbers will be deleted and regenerated to match the sorted order.

### Parameters

<i>df</i>	The DataFrame containing unsorted rows.
<i>ignored_columns</i>	A list of column names to NOT sort by.

## 5.16 tests Namespace Reference

### Namespaces

- namespace [conftest](#)
- namespace [test\\_components](#)

## 5.17 tests.conftest Namespace Reference

### Functions

- [pytest\\_adoption](#) (`parser`)
- [session](#) (`request`)

*Fixture to create session.*

#### 5.17.1 Function Documentation

##### 5.17.1.1 pytest\_adoption()

```
pytest_adoption (
    parser )
```

### 5.17.1.2 session()

```
session (
    request )
```

Fixture to create session.

## 5.18 tests.test\_components Namespace Reference

### Functions

- [relational\\_db](#) (session)
 

*Fixture to get relational database connection.*
- [docs\\_db](#) (session)
 

*Fixture to get document database connection.*
- [graph\\_db](#) (session)
 

*Fixture to get document database connection.*
- [test\\_db\\_relational\\_minimal](#) ([relational\\_db](#))
 

*Tests if the RelationalConnector has a valid connection string.*
- [test\\_db\\_docs\\_minimal](#) ([docs\\_db](#))
 

*Tests if the DocumentConnector has a valid connection string.*
- [test\\_db\\_graph\\_minimal](#) ([graph\\_db](#))
 

*Tests if the GraphConnector has a valid connection string.*
- [test\\_db\\_relational\\_comprehensive](#) ([relational\\_db](#))
 

*Tests if the GraphConnector is working as intended.*
- [test\\_db\\_docs\\_comprehensive](#) ([docs\\_db](#))
 

*Tests if the GraphConnector is working as intended.*
- [test\\_db\\_graph\\_comprehensive](#) ([graph\\_db](#))
 

*Tests if the GraphConnector is working as intended.*
- [load\\_examples\\_relational](#) ([relational\\_db](#))
 

*Fixture to create relational tables using engine-specific syntax.*
- [test\\_sql\\_example\\_1](#) ([relational\\_db](#), [load\\_examples\\_relational](#))
 

*Run queries contained within test files.*
- [test\\_sql\\_example\\_2](#) ([relational\\_db](#), [load\\_examples\\_relational](#))
 

*Run queries contained within test files.*
- [test\\_mongo\\_example\\_1](#) ([docs\\_db](#))
 

*Run queries contained within test files.*
- [test\\_mongo\\_example\\_2](#) ([docs\\_db](#))
 

*Run queries contained within test files.*
- [test\\_mongo\\_example\\_3](#) ([docs\\_db](#))
 

*Run queries contained within test files.*
- [\\_test\\_query\\_file](#) ([db\\_fixture](#), str filename, List valid\_files)
 

*Run queries from a local file through the database.*

### 5.18.1 Function Documentation

#### 5.18.1.1 \_test\_query\_file()

```
_test_query_file (
    db_fixture,
    str filename,
    List valid_files ) [protected]
```

Run queries from a local file through the database.

**Parameters**

<i>db_fixture</i>	Fixture corresponding to the current session's database.
<i>filename</i>	The name of a query file (for example ./tests/example1.sql).
<i>valid_files</i>	A list of file extensions valid for this database type.

**5.18.1.2 docs\_db()**

```
docs_db (
    session )
```

Fixture to get document database connection.

**5.18.1.3 graph\_db()**

```
graph_db (
    session )
```

Fixture to get document database connection.

**5.18.1.4 load\_examples\_relational()**

```
load_examples_relational (
    relational_db )
```

Fixture to create relational tables using engine-specific syntax.

**5.18.1.5 relational\_db()**

```
relational_db (
    session )
```

Fixture to get relational database connection.

**5.18.1.6 test\_db\_docs\_comprehensive()**

```
test_db_docs_comprehensive (
    docs_db )
```

Tests if the GraphConnector is working as intended.

**5.18.1.7 test\_db\_docs\_minimal()**

```
test_db_docs_minimal (
    docs_db )
```

Tests if the DocumentConnector has a valid connection string.

#### 5.18.1.8 `test_db_graph_comprehensive()`

```
test_db_graph_comprehensive (
    graph_db )
```

Tests if the GraphConnector is working as intended.

#### 5.18.1.9 `test_db_graph_minimal()`

```
test_db_graph_minimal (
    graph_db )
```

Tests if the GraphConnector has a valid connection string.

#### 5.18.1.10 `test_db_relational_comprehensive()`

```
test_db_relational_comprehensive (
    relational_db )
```

Tests if the GraphConnector is working as intended.

#### 5.18.1.11 `test_db_relational_minimal()`

```
test_db_relational_minimal (
    relational_db )
```

Tests if the RelationalConnector has a valid connection string.

#### 5.18.1.12 `test_mongo_example_1()`

```
test_mongo_example_1 (
    docs_db )
```

Run queries contained within test files.

Internal errors are handled by the class itself, and ruled out earlier. Here we just assert that the received results DataFrame matches what we expected.

#### 5.18.1.13 `test_mongo_example_2()`

```
test_mongo_example_2 (
    docs_db )
```

Run queries contained within test files.

Internal errors are handled by the class itself, and ruled out earlier. Here we just assert that the received results DataFrame matches what we expected.

### 5.18.1.14 test\_mongo\_example\_3()

```
test_mongo_example_3 (
    docs_db )
```

Run queries contained within test files.

Internal errors are handled by the class itself, and ruled out earlier. Here we just assert that the received results DataFrame matches what we expected.

### 5.18.1.15 test\_sql\_example\_1()

```
test_sql_example_1 (
    relational_db,
    load_examples_relational )
```

Run queries contained within test files.

Internal errors are handled by the class itself, and ruled out earlier. Here we just assert that the received results DataFrame matches what we expected.

#### Note

Uses a table-creation fixture to load / unload schema.

### 5.18.1.16 test\_sql\_example\_2()

```
test_sql_example_2 (
    relational_db,
    load_examples_relational )
```

Run queries contained within test files.

Internal errors are handled by the class itself, and ruled out earlier. Here we just assert that the received results DataFrame matches what we expected.

#### Note

Uses a table-creation fixture to load / unload schema.



# Chapter 6

## Class Documentation

### 6.1 Book Class Reference

#### Public Member Functions

- `__init__ (self, str title_key="Title:", str author_key="Author:", str language_key="Language:", str date_key="Release date:")`

#### Public Attributes

- `title_key`
- `author_key`
- `language_key`
- `date_key`

#### 6.1.1 Constructor & Destructor Documentation

##### 6.1.1.1 `__init__()`

```
__init__ (
    self,
    str title_key = "Title:",
    str author_key = "Author:",
    str language_key = "Language:",
    str date_key = "Release date:" )
```

#### 6.1.2 Member Data Documentation

##### 6.1.2.1 `author_key`

```
author_key
```

### 6.1.2.2 date\_key

date\_key

### 6.1.2.3 language\_key

language\_key

### 6.1.2.4 title\_key

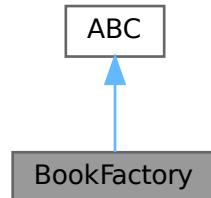
title\_key

The documentation for this class was generated from the following file:

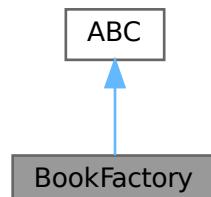
- /home/runner/work/dsci-capstone/dsci-capstone/components/[book\\_conversion.py](#)

## 6.2 BookFactory Class Reference

Inheritance diagram for BookFactory:



Collaboration diagram for BookFactory:



## Public Member Functions

- [Book create\\_book \(self\)](#)

### 6.2.1 Member Function Documentation

#### 6.2.1.1 `create_book()`

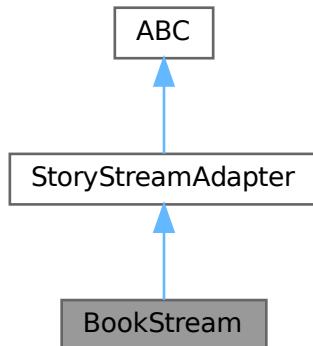
```
Book create_book (
    self )
```

The documentation for this class was generated from the following file:

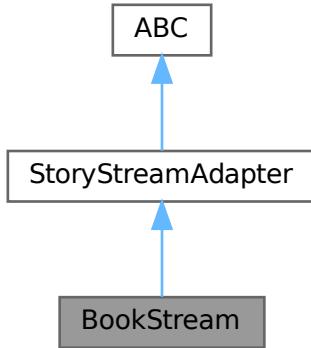
- [/home/runner/work/dsci-capstone/dsci-capstone/components/book\\_conversion.py](#)

## 6.3 BookStream Class Reference

Inheritance diagram for BookStream:



Collaboration diagram for BookStream:



## Public Member Functions

- [\\_\\_init\\_\\_ \(self, Book book\)](#)
- [Chunk stream\\_segments \(self\)](#)  
*Yields sanitized parts of a book.*

## Public Member Functions inherited from [StoryStreamAdapter](#)

- [Chunk stream\\_paragraphs \(self\)](#)  
*Concrete helper method to split segments into paragraphs.*
- str [stream\\_sentences \(self\)](#)  
*Concrete helper method to split paragraphs into sentences.*

## Public Attributes

- [book](#)

### 6.3.1 Constructor & Destructor Documentation

#### 6.3.1.1 \_\_init\_\_()

```
__init__ (
    self,
    Book book )
```

### 6.3.2 Member Function Documentation

#### 6.3.2.1 stream\_segments()

```
Chunk stream_segments (
    self )
```

Yields sanitized parts of a book.

- Story segments usually correspond to chapters.
- They serve as borders between chunking operations, ensuring chunks do not span multiple chapters. Implementation is handled by child classes BookStream, etc.
- Segments should be pre-cleaned and must contain 1 paragraph per line with all other newlines removed.

Reimplemented from [StoryStreamAdapter](#).

### 6.3.3 Member Data Documentation

#### 6.3.3.1 book

book

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/components/[book\\_conversion.py](#)

## 6.4 Chunk Class Reference

Lightweight container for a span of story text.

### Public Member Functions

- `__init__` (self, str `text`, int `book_id`, int `chapter_number`, int `line_start`, int `line_end`, int `story_id`, float `story_percent`, float `chapter_percent`, int `max_chunk_length=-1`)  
*Construct a Chunk.*
- int `char_count` (self, bool `prune_newlines=False`)  
*Computes the character count.*
- str `__repr__` (self)

### Public Attributes

- `text`

### 6.4.1 Detailed Description

Lightweight container for a span of story text.

- Carries positional metadata so downstream consumers can reconstruct context.
- Filter by story\_id to fetch all chunks for a particular story.
- Use story\_percent and chapter\_percent to quickly sort chunks by intended order.
- Use book\_id, chapter\_number, line\_start, and line\_end to locate this chunk within source material.

### 6.4.2 Constructor & Destructor Documentation

#### 6.4.2.1 \_\_init\_\_()

```
__init__ (
    self,
    str text,
    int book_id,
    int chapter_number,
    int line_start,
    int line_end,
    int story_id,
    float story_percent,
    float chapter_percent,
    int max_chunk_length = -1 )
```

Construct a Chunk.

#### Parameters

<i>text</i>	The text content for this span.
<i>book_id</i>	Corresponds to a single book file in the dataset.
<i>chapter_number</i>	The chapter containing this chunk in the book file, 1-based.
<i>line_start</i>	The starting line within the TEI file, 1-based.
<i>line_end</i>	The inclusive ending line index within the TEI file ( $\geq$ line_start).
<i>story_id</i>	A stable id for the overall story. May be identical to book_id
<i>story_percent</i>	Approximate progress through the whole story [0.0, 100.0].
<i>chapter_percent</i>	Approximate progress through the current segment [0.0, 100.0].
<i>max_chunk_length</i>	Max allowed characters ( $\leq 0$ means "no limit").

#### Exceptions

<i>ValueError</i>	if text exceeds max_chunk_length when max_chunk_length > 0.
-------------------	---

### 6.4.3 Member Function Documentation

#### 6.4.3.1 \_\_repr\_\_()

```
str __repr__ (
    self )
```

#### 6.4.3.2 char\_count()

```
int char_count (
    self,
    bool prune_newlines = False )
```

Computes the character count.

##### Parameters

<code>prune_newlines</code>	Whether to remove newlines for the count.
-----------------------------	---

##### Returns

The number of characters in the chunk text.

### 6.4.4 Member Data Documentation

#### 6.4.4.1 text

`text`

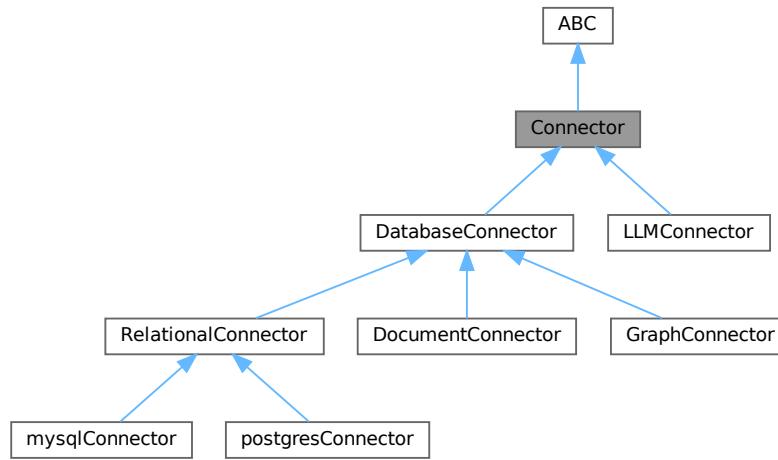
The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/components/[book\\_conversion.py](#)

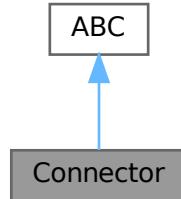
## 6.5 Connector Class Reference

Abstract base class for external connectors.

Inheritance diagram for Connector:



Collaboration diagram for Connector:



## Public Member Functions

- None `configure` (self, str DB, str database\_name)  
*Read connection settings from the .env file.*
- bool `test_connection` (self, bool raise\_error=True)  
*Establish a basic connection to the database.*
- Optional[DataFrame] `execute_query` (self, str query)  
*Send a single command through the connection.*
- List[Optional[DataFrame]] `execute_file` (self, str filename)  
*Run several commands from a file.*

### 6.5.1 Detailed Description

Abstract base class for external connectors.

#### Note

Credentials are specified in the .env file.

Derived classes should implement:

- `init`
- `components.connectors.Connector.configure`
- `components.connectors.Connector.test_connection`
- `components.connectors.Connector.execute_query`
- `components.connectors.Connector.execute_file`

### 6.5.2 Member Function Documentation

#### 6.5.2.1 `configure()`

```
None configure (
    self,
    str DB,
    str database_name )
```

Read connection settings from the .env file.

##### Parameters

<code>DB</code>	The prefix of fetched credentials.
<code>database_name</code>	The specific service to connect to.

Reimplemented in [LLMConnector](#), and [DatabaseConnector](#).

#### 6.5.2.2 `execute_file()`

```
List[Optional[DataFrame]] execute_file (
    self,
    str filename )
```

Run several commands from a file.

##### Parameters

<code>filename</code>	The path to a specified query or prompt file (.sql, .txt).
-----------------------	--

**Returns**

Whether the query was performed successfully.

Reimplemented in [DatabaseConnector](#), and [LLMConnector](#).

### 6.5.2.3 execute\_query()

```
Optional[DataFrame] execute_query (
    self,
    str query )
```

Send a single command through the connection.

**Parameters**

<i>query</i>	A single query to perform on the database.
--------------	--

**Returns**

The result of the query, or None

Reimplemented in [DatabaseConnector](#), [RelationalConnector](#), [DocumentConnector](#), [GraphConnector](#), and [LLMConnector](#).

### 6.5.2.4 test\_connection()

```
bool test_connection (
    self,
    bool raise_error = True )
```

Establish a basic connection to the database.

Can be configured to fail silently, which enables retries or external handling.

**Parameters**

<i>raise_error</i>	Whether to raise an error on connection failure.
--------------------	--

**Returns**

Whether the connection test was successful.

**Exceptions**

<i>RuntimeError</i>	If <i>raise_error</i> is True and the connection test fails to complete.
---------------------	--

Reimplemented in [LLMConnector](#), [RelationalConnector](#), [DocumentConnector](#), and [GraphConnector](#).

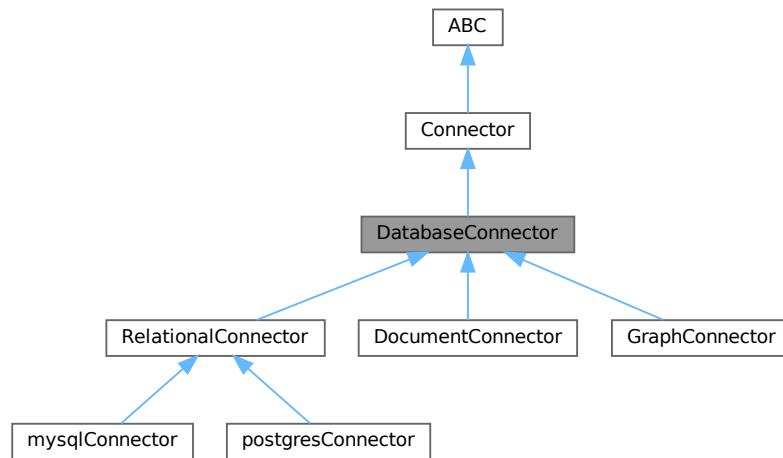
The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/components/[connectors.py](#)

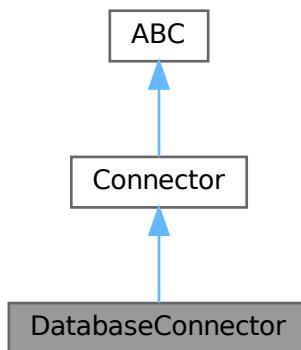
## 6.6 DatabaseConnector Class Reference

Abstract base class for database engine connectors.

Inheritance diagram for DatabaseConnector:



Collaboration diagram for DatabaseConnector:



## Public Member Functions

- None `__init__` (self, bool `verbose=False`)  
*Initialize the connector.*
- None `configure` (self, str DB, str database\_name)  
*Read connection settings from the .env file.*
- None `change_database` (self, str new\_database)  
*Update the connection URI to reference a different database in the same engine.*
- Optional[DataFrame] `execute_query` (self, str query)  
*Send a single command through the connection.*
- List[Optional[DataFrame]] `execute_combined` (self, str multi\_query)  
*Run several database commands in sequence.*
- List[Optional[DataFrame]] `execute_file` (self, str filename)  
*Run several database commands from a file.*
- Optional[DataFrame] `get_dataframe` (self, str name)  
*Automatically generate and run a query for the specified resource.*
- None `create_database` (self, str database\_name)  
*Use the current database connection to create a sibling database in this engine.*
- None `drop_database` (self, str database\_name)  
*Delete all data stored in a particular database.*
- bool `database_exists` (self, str database\_name)  
*Search for an existing database using the provided name.*

## Public Member Functions inherited from `Connector`

- bool `test_connection` (self, bool raise\_error=True)  
*Establish a basic connection to the database.*

## Public Attributes

- `verbose`  
*Whether to print debug messages.*
- `db_type`
- `db_engine`
- `username`
- `password`
- `host`
- `port`
- `connection_string`

## Protected Member Functions

- bool `_is_single_query` (self, str query)  
*Checks if a string contains multiple queries.*
- List[str] `_split_combined` (self, str multi\_query)  
*Checks if a string contains multiple queries.*

### 6.6.1 Detailed Description

Abstract base class for database engine connectors.

Derived classes should implement:

- `components.connectors.DatabaseConnector.__init__`
- `components.connectors.DatabaseConnector.test_connection`
- `components.connectors.DatabaseConnector.execute_query`
- `components.connectors.DatabaseConnector._split_combined`
- `components.connectors.DatabaseConnector.get_dataframe`
- `components.connectors.DatabaseConnector.create_database`
- `components.connectors.DatabaseConnector.drop_database`
- `components.connectors.DatabaseConnector.change_database`
- `components.connectors.DatabaseConnector.database_exists`

### 6.6.2 Constructor & Destructor Documentation

#### 6.6.2.1 \_\_init\_\_()

```
None __init__ (
    self,
    bool verbose = False )
```

Initialize the connector.

##### Parameters

<code>verbose</code>	Whether to print debug messages.
----------------------	----------------------------------

##### Note

Attributes will be set to None until `components.connectors.DatabaseConnector.configure()` is called.

Reimplemented in `RelationalConnector`, `mysqlConnector`, `postgresConnector`, `DocumentConnector`, and `GraphConnector`.

### 6.6.3 Member Function Documentation

#### 6.6.3.1 \_is\_single\_query()

```
bool _is_single_query (
    self,
    str query ) [protected]
```

Checks if a string contains multiple queries.

**Parameters**

<i>query</i>	A single or combined query string.
--------------	------------------------------------

**Returns**

Whether the query is single (true) or combined (false).

**6.6.3.2 `_split_combined()`**

```
List[str] _split_combined (
    self,
    str multi_query ) [protected]
```

Checks if a string contains multiple queries.

**Parameters**

<i>multi_query</i>	A string containing multiple queries.
--------------------	---------------------------------------

**Returns**

A list of single-query strings.

Reimplemented in [RelationalConnector](#), [DocumentConnector](#), and [GraphConnector](#).

**6.6.3.3 `change_database()`**

```
None change_database (
    self,
    str new_database )
```

Update the connection URI to reference a different database in the same engine.

**Parameters**

<i>new_database</i>	The name of the database to connect to.
---------------------	---

Reimplemented in [RelationalConnector](#), [DocumentConnector](#), and [GraphConnector](#).

**6.6.3.4 `configure()`**

```
None configure (
    self,
    str DB,
    str database_name )
```

Read connection settings from the .env file.

**Parameters**

<i>DB</i>	The prefix of fetched database credentials.
<i>database_name</i>	The name of the database to connect to.

Reimplemented from [Connector](#).

### 6.6.3.5 `create_database()`

```
None create_database (
    self,
    str database_name )
```

Use the current database connection to create a sibling database in this engine.

**Parameters**

<i>database_name</i>	The name of the new database to create.
----------------------	---

**Exceptions**

<i>Log.Failure</i>	If the database already exists.
--------------------	---------------------------------

Reimplemented in [RelationalConnector](#), [DocumentConnector](#), and [GraphConnector](#).

### 6.6.3.6 `database_exists()`

```
bool database_exists (
    self,
    str database_name )
```

Search for an existing database using the provided name.

**Parameters**

<i>database_name</i>	The name of a database to search for.
----------------------	---------------------------------------

**Returns**

Whether the database is visible to this connector.

Reimplemented in [RelationalConnector](#), [DocumentConnector](#), and [GraphConnector](#).

### 6.6.3.7 `drop_database()`

```
None drop_database (
    self,
    str database_name )
```

Delete all data stored in a particular database.

**Parameters**

<i>database_name</i>	The name of an existing database.
----------------------	-----------------------------------

**Exceptions**

<i>Log.Failure</i>	If the database does not exist.
--------------------	---------------------------------

Reimplemented in [DocumentConnector](#), [GraphConnector](#), and [RelationalConnector](#).

### 6.6.3.8 execute\_combined()

```
List[Optional[DataFrame]] execute_combined (
    self,
    str multi_query )
```

Run several database commands in sequence.

**Parameters**

<i>multi_query</i>	A string containing multiple queries.
--------------------	---------------------------------------

**Returns**

A list of query results converted to DataFrames.

### 6.6.3.9 execute\_file()

```
List[Optional[DataFrame]] execute_file (
    self,
    str filename )
```

Run several database commands from a file.

**Note**

Loads the entire file into memory at once.

**Parameters**

<i>filename</i>	The path to a specified query file (.sql, .cql, .json).
-----------------	---

**Returns**

Whether the query was performed successfully.

**Exceptions**

<i>Log.Failure</i>	If any query in the file fails to execute.
--------------------	--

Reimplemented from [Connector](#).

**6.6.3.10 execute\_query()**

```
Optional[DataFrame] execute_query (
    self,
    str query )
```

Send a single command through the connection.

**Note**

If a result is returned, it will be converted to a DataFrame.

**Parameters**

<i>query</i>	A single query to perform on the database.
--------------	--

**Returns**

DataFrame containing the result of the query, or None

**Exceptions**

<i>Log.Failure</i>	If the query fails to execute.
--------------------	--------------------------------

Reimplemented from [Connector](#).

Reimplemented in [RelationalConnector](#), [DocumentConnector](#), and [GraphConnector](#).

**6.6.3.11 get\_dataframe()**

```
Optional[DataFrame] get_dataframe (
    self,
    str name )
```

Automatically generate and run a query for the specified resource.

**Parameters**

<i>name</i>	The name of an existing table or collection in the database.
-------------	--

**Returns**

DataFrame containing the requested data, or None

Reimplemented in [RelationalConnector](#), [DocumentConnector](#), and [GraphConnector](#).

## 6.6.4 Member Data Documentation

### 6.6.4.1 connection\_string

connection\_string

### 6.6.4.2 db\_engine

db\_engine

### 6.6.4.3 db\_type

db\_type

### 6.6.4.4 host

host

### 6.6.4.5 password

password

### 6.6.4.6 port

port

### 6.6.4.7 username

username

### 6.6.4.8 verbose

verbose

Whether to print debug messages.

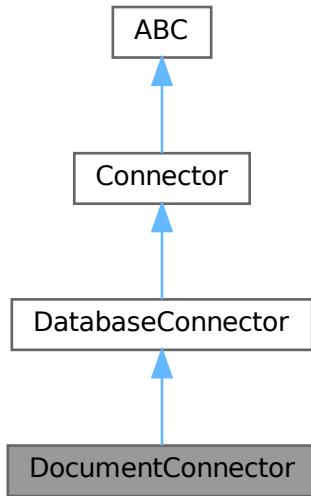
The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/components/[connectors.py](#)

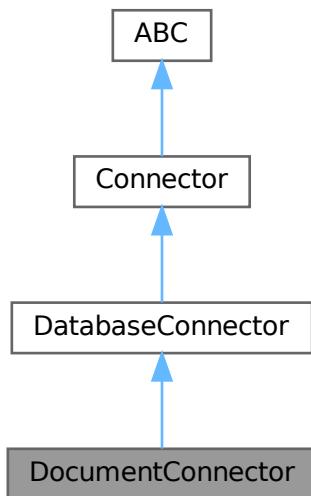
## 6.7 DocumentConnector Class Reference

Connector for MongoDB (document database)

Inheritance diagram for DocumentConnector:



Collaboration diagram for DocumentConnector:



## Public Member Functions

- None `__init__` (self, bool `verbose=False`)  
*Creates a new MongoDB connector.*
- None `change_database` (self, str `new_database`)  
*Update the connection URI to reference a different database in the same engine.*
- bool `test_connection` (self, bool `raise_error=True`)  
*Establish a basic connection to the MongoDB database.*
- bool `check_connection` (self, str `log_source`, bool `raise_error`)  
*Minimal connection test to determine if our connection string is valid.*
- Optional[DataFrame] `execute_query` (self, str `query`)  
*Send a single MongoDB command using PyMongo.*
- Optional[DataFrame] `get_dataframe` (self, str `name`)  
*Automatically generate and run a query for the specified collection.*
- None `create_database` (self, str `database_name`)  
*Use the current database connection to create a sibling database in this engine.*
- None `drop_database` (self, str `database_name`)  
*Delete all data stored in a particular database.*
- bool `database_exists` (self, str `database_name`)  
*Search for an existing database using the provided name.*
- None `delete_dummy` (self)  
*Delete the initial dummy collection from the database.*

## Public Member Functions inherited from DatabaseConnector

- None `configure` (self, str `DB`, str `database_name`)  
*Read connection settings from the .env file.*
- List[Optional[DataFrame]] `execute_combined` (self, str `multi_query`)  
*Run several database commands in sequence.*
- List[Optional[DataFrame]] `execute_file` (self, str `filename`)  
*Run several database commands from a file.*

## Public Attributes

- `database_name`
- `verbose`
- `connection_string`

## Public Attributes inherited from DatabaseConnector

- `verbose`  
*Whether to print debug messages.*
- `db_type`
- `db_engine`
- `username`
- `password`
- `host`
- `port`
- `connection_string`

## Protected Member Functions

- list[str] [\\_split\\_combined](#) (self, str multi\_query)

*Divides a string into non-divisible MongoDB commands by splitting on semicolons at depth 0.*

## Protected Member Functions inherited from [DatabaseConnector](#)

- bool [\\_is\\_single\\_query](#) (self, str query)

*Checks if a string contains multiple queries.*

## Protected Attributes

- [\\_auth\\_suffix](#)

### 6.7.1 Detailed Description

Connector for MongoDB (document database)

- Uses mongoengine.connect(...) on-demand for connections.
- Low-level operations use pymongo via mongoengine.get\_db().
- create\_database uses an init collection insertion (MongoDB is lazy).

### 6.7.2 Constructor & Destructor Documentation

#### 6.7.2.1 [\\_\\_init\\_\\_\(\)](#)

```
None __init__ (
    self,
    bool verbose = False )
```

Creates a new MongoDB connector.

##### Parameters

<code>verbose</code>	Whether to print debug messages.
----------------------	----------------------------------

Reimplemented from [DatabaseConnector](#).

### 6.7.3 Member Function Documentation

#### 6.7.3.1 [\\_split\\_combined\(\)](#)

```
list[str] _split_combined (
    self,
    str multi_query ) [protected]
```

Divides a string into non-divisible MongoDB commands by splitting on semicolons at depth 0.

Handles nested brackets and semicolons inside JSON strings.

#### Parameters

<i>multi_query</i>	A string containing multiple queries with possible comments.
--------------------	--

#### Returns

A list of single-query strings (cleaned, ready for JSON parsing).

Reimplemented from [DatabaseConnector](#).

### 6.7.3.2 change\_database()

```
None change_database (
    self,
    str new_database )
```

Update the connection URI to reference a different database in the same engine.

#### Note

Additional settings are appended as a suffix to the MongoDB connection string.

#### Parameters

<i>new_database</i>	The name of the database to connect to.
---------------------	---

Reimplemented from [DatabaseConnector](#).

### 6.7.3.3 check\_connection()

```
bool check_connection (
    self,
    str log_source,
    bool raise_error )
```

Minimal connection test to determine if our connection string is valid.

Connect to MongoDB using `MongoEngine.connect()`

#### Parameters

<i>log_source</i>	The Log class prefix indicating which method is performing the check.
<i>raise_error</i>	Whether to raise an error on connection failure.

**Returns**

Whether the connection test was successful.

**Exceptions**

<i>RuntimeError</i>	If <code>raise_error</code> is True and the connection test fails to complete.
---------------------	--

#### 6.7.3.4 `create_database()`

```
None create_database (
    self,
    str database_name )
```

Use the current database connection to create a sibling database in this engine.

**Note**

Forces MongoDB to actually create it by inserting a small init document.

**Parameters**

<code>database_name</code>	The name of the new database to create.
----------------------------	---

**Exceptions**

<i>Log.Failure</i>	If we fail to create the requested database for any reason.
--------------------	---

Reimplemented from [DatabaseConnector](#).

#### 6.7.3.5 `database_exists()`

```
bool database_exists (
    self,
    str database_name )
```

Search for an existing database using the provided name.

**Parameters**

<code>database_name</code>	The name of a database to search for.
----------------------------	---------------------------------------

**Returns**

Whether the database is visible to this connector.

Reimplemented from [DatabaseConnector](#).

### 6.7.3.6 delete\_dummy()

```
None delete_dummy (
    self )
```

Delete the initial dummy collection from the database.

#### Note

Call this method whenever real data is being added to avoid pollution.

### 6.7.3.7 drop\_database()

```
None drop_database (
    self,
    str database_name )
```

Delete all data stored in a particular database.

#### Parameters

<i>database_name</i>	The name of an existing database.
----------------------	-----------------------------------

#### Exceptions

<i>Log.Failure</i>	If we fail to drop the target database for any reason.
--------------------	--

Reimplemented from [DatabaseConnector](#).

### 6.7.3.8 execute\_query()

```
Optional[DataFrame] execute_query (
    self,
    str query )
```

Send a single MongoDB command using PyMongo.

- The query must be a valid JSON command object (e.g. `{"find": "users", "filter": {...}}`).
- Mongo shell syntax such as `db.users.find({ ... })` or `.js` files will NOT work.
- If a result is returned, it will be converted to a DataFrame.

#### Exceptions

<i>Log.Failure</i>	If the query fails to execute.
--------------------	--------------------------------

Reimplemented from [DatabaseConnector](#).

### 6.7.3.9 `get_dataframe()`

```
Optional[DataFrame] get_dataframe (
    self,
    str name )
```

Automatically generate and run a query for the specified collection.

#### Parameters

<code>name</code>	The name of an existing table or collection in the database.
-------------------	--

#### Returns

DataFrame containing the requested data, or None

#### Exceptions

<code>Log.Failure</code>	If we fail to create the requested DataFrame for any reason.
--------------------------	--

Reimplemented from [DatabaseConnector](#).

### 6.7.3.10 `test_connection()`

```
bool test_connection (
    self,
    bool raise_error = True )
```

Establish a basic connection to the MongoDB database.

Can be configured to fail silently, which enables retries or external handling.

#### Parameters

<code>raise_error</code>	Whether to raise an error on connection failure.
--------------------------	--

#### Returns

Whether the connection test was successful.

#### Exceptions

<code>Log.Failure</code>	If <code>raise_error</code> is True and the connection test fails to complete.
--------------------------	--

Reimplemented from [Connector](#).

## 6.7.4 Member Data Documentation

### 6.7.4.1 `_auth_suffix`

```
_auth_suffix [protected]
```

### 6.7.4.2 `connection_string`

```
connection_string
```

### 6.7.4.3 `database_name`

```
database_name
```

### 6.7.4.4 `verbose`

```
verbose
```

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/components/[document\\_storage.py](#)

## 6.8 EPUBToTEI Class Reference

Converts EPUB files to XML format (TEI specification).

### Public Member Functions

- [`\_\_init\_\_`](#) (`self`, `epub_path`, `save_pandoc=False`, `save_tei=True`)  
*Initialize the converter.*
- [`convert\_to\_tei`](#) (`self`)  
*Uses Pandoc to draft a TEI string from EPUB.*
- [`clean\_tei`](#) (`self`)  
*Wrap root if missing, sanitize ids, and save cleaned TEI.*

### Public Attributes

- [`epub\_path`](#)
- [`save\_pandoc`](#)
- [`pandoc\_xml\_path`](#)
- [`save\_tei`](#)
- [`tei\_path`](#)
- [`raw\_tei\_content`](#)
- [`clean\_tei\_content`](#)

## Static Public Attributes

- dict `xml_namespace` = {"tei": "http://www.tei-c.org/ns/1.0"}
- str `encoding` = "utf-8"

## Protected Member Functions

- str `_sanitize_ids` (self, str content)  
*Sanitize XML IDs in the TEI content to ensure they are valid and consistent.*
- str `_prune_bad_tags` (self, str content)  
*Replace all `<lb` tags with newline characters in TEI.*

### 6.8.1 Detailed Description

Converts EPUB files to XML format (TEI specification).

Takes an EPUB book file and converts it to TEI in order to represent its chapter hierarchy.

### 6.8.2 Constructor & Destructor Documentation

#### 6.8.2.1 `__init__()`

```
__init__ (
    self,
    epub_path,
    save_pandoc = False,
    save_teis = True )
```

Initialize the converter.

#### Parameters

<code>epub_path</code>	String containing the relative path to an EPUB file.
<code>save_pandoc</code>	Flag to save the intermediate Pandoc output to .tei.xml
<code>save_teis</code>	Flag to save the final TEI file as .tei

### 6.8.3 Member Function Documentation

#### 6.8.3.1 `_prune_bad_tags()`

```
str _prune_bad_tags (
    self,
    str content ) [protected]
```

Replace all `<lb` tags with newline characters in TEI.

### 6.8.3.2 `_sanitize_ids()`

```
str _sanitize_ids (
    self,
    str content ) [protected]
```

Sanitize XML IDs in the TEI content to ensure they are valid and consistent.

Pandoc sometimes generates invalid or non-unique `xml:id` attributes (e.g., containing spaces, punctuation, or mixed casing). Since we rely on these IDs as dictionary keys / anchors, we sanitize them using a regex to enforce alphanumeric/underscore/dash format.

#### Parameters

<code>content</code>	The raw TEI XML string possibly containing invalid <code>xml:id</code> attributes.
----------------------	--

#### Returns

A TEI XML string with valid NCNames, prefixed with 'id\_'.

### 6.8.3.3 `clean_teis()`

```
clean_teis (
    self )
```

Wrap root if missing, sanitize ids, and save cleaned TEI.

### 6.8.3.4 `convert_to_teis()`

```
convert_to_teis (
    self )
```

Uses Pandoc to draft a TEI string from EPUB.

## 6.8.4 Member Data Documentation

### 6.8.4.1 `clean_teis_content`

```
clean_teis_content
```

### 6.8.4.2 `encoding`

```
str encoding = "utf-8" [static]
```

### 6.8.4.3 `epub_path`

```
epub_path
```

#### 6.8.4.4 pandoc\_xml\_path

```
pandoc_xml_path
```

#### 6.8.4.5 raw\_teis\_content

```
raw_teis_content
```

#### 6.8.4.6 save\_pandoc

```
save_pandoc
```

#### 6.8.4.7 save\_teis

```
save_teis
```

#### 6.8.4.8 tei\_path

```
tei_path
```

#### 6.8.4.9 xml\_namespace

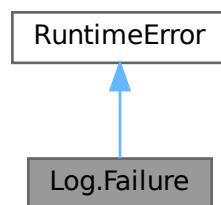
```
dict xml_namespace = {"tei": "http://www.tei-c.org/ns/1.0"} [static]
```

The documentation for this class was generated from the following file:

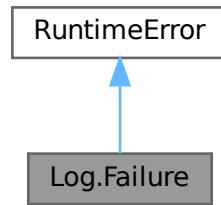
- /home/runner/work/dsci-capstone/dsci-capstone/components/[book\\_conversion.py](#)

## 6.9 Log.Failure Class Reference

Inheritance diagram for Log.Failure:



Collaboration diagram for Log.Failure:



## Public Member Functions

- `__init__` (self, str `prefix`="ERROR", str `msg`=")
- `__str__` (self)

## Public Attributes

- `prefix`
- `msg`

### 6.9.1 Constructor & Destructor Documentation

#### 6.9.1.1 `__init__()`

```
__init__ (
    self,
    str prefix = "ERROR",
    str msg = "" )
```

### 6.9.2 Member Function Documentation

#### 6.9.2.1 `__str__()`

```
__str__ (
    self )
```

### 6.9.3 Member Data Documentation

#### 6.9.3.1 `msg`

```
msg
```

### 6.9.3.2 prefix

prefix

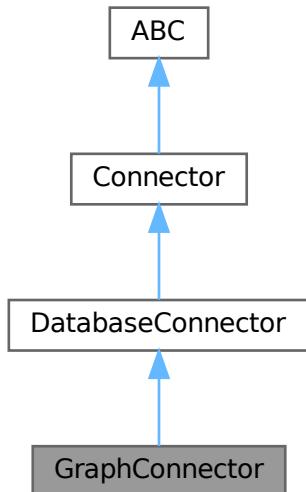
The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/src/[util.py](#)

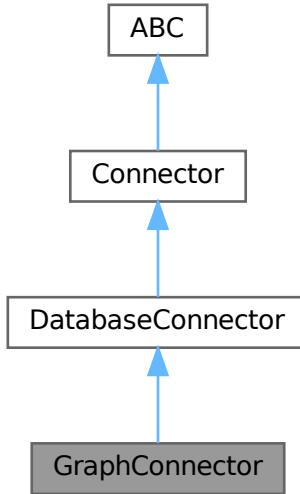
## 6.10 GraphConnector Class Reference

Connector for Neo4j (graph database).

Inheritance diagram for GraphConnector:



Collaboration diagram for GraphConnector:



## Public Member Functions

- None `__init__(self, bool verbose=False)`  
*Creates a new Neo4j connector.*
- None `change_database(self, str new_database)`  
*Update the connection URI to reference a different database in the same engine.*
- None `change_graph(self, str graph_name)`  
*Sets graph\_name to create new a Knowledge Graph (collection of triples).*
- bool `test_connection(self, bool raise_error=True)`  
*Establish a basic connection to the Neo4j database.*
- bool `check_connection(self, str log_source, bool raise_error)`  
*Minimal connection test to determine if our connection string is valid.*
- Optional[DataFrame] `execute_query(self, str query)`  
*Send a single Cypher query to Neo4j.*
- Optional[DataFrame] `get_dataframe(self, str name)`  
*Automatically generate and run a query for the specified Knowledge Graph collection.*
- List[str] `get_unique(self, str key)`  
*Retrieve all unique values for a specified node property.*
- None `create_database(self, str database_name)`  
*Create a fresh pseudo-database if it does not already exist.*
- None `drop_database(self, str database_name)`  
*Delete all nodes stored under a particular database name.*
- bool `database_exists(self, str database_name)`  
*Search for an existing database using the provided name.*
- None `delete_dummy(self)`  
*Delete the initial dummy node from the database.*
- None `add_triple(self, str subject, str relation, str object_, bool _delete_init=True)`

- Add a semantic triple to the graph using raw Cypher.
- DataFrame [get\\_edge\\_counts](#) (self, int top\_n=10)
 

*Return node names and their edge counts, ordered by edge count descending.*
- DataFrame [get\\_all\\_triples](#) (self)
 

*Return all triples in the current pseudo-database as a pandas DataFrame.*
- None [print\\_nodes](#) (self, int max\_rows=20, int max\_col\_width=50)
 

*Print all nodes and edges in the current pseudo-database with row/column formatting.*
- None [print\\_triples](#) (self, int max\_rows=20, int max\_col\_width=50)
 

*Print all nodes and edges in the current pseudo-database with row/column formatting.*
- str [IS\\_DUMMY\\_](#) (self, str alias='n')
 

*Generates Cypher code to select dummy nodes inside a WHERE clause.*
- str [NOT\\_DUMMY\\_](#) (self, str alias='n')
 

*Generates Cypher code to select non-dummy nodes inside a WHERE clause.*
- str [SAME\\_DB\\_KG\\_](#) (self)
 

*Generates a Cypher pattern dictionary to match nodes by current database and graph name.*

## Public Member Functions inherited from [DatabaseConnector](#)

- None [configure](#) (self, str DB, str database\_name)
 

*Read connection settings from the .env file.*
- List[Optional[DataFrame]] [execute\\_combined](#) (self, str multi\_query)
 

*Run several database commands in sequence.*
- List[Optional[DataFrame]] [execute\\_file](#) (self, str filename)
 

*Run several database commands from a file.*

## Public Attributes

- [database\\_name](#)
- [verbose](#)
- [graph\\_name](#)
- [connection\\_string](#)

## Public Attributes inherited from [DatabaseConnector](#)

- [verbose](#)

*Whether to print debug messages.*
- [db\\_type](#)
- [db\\_engine](#)
- [username](#)
- [password](#)
- [host](#)
- [port](#)
- [connection\\_string](#)

## Protected Member Functions

- List[str] [\\_split\\_combined](#) (self, str multi\_query)
 

*Divides a string into non-divisible CQL queries, ignoring comments.*

## Protected Member Functions inherited from [DatabaseConnector](#)

- bool [\\_is\\_single\\_query](#) (self, str query)  
*Checks if a string contains multiple queries.*

## Protected Attributes

- [\\_created\\_dummy](#)

### 6.10.1 Detailed Description

Connector for Neo4j (graph database).

- Uses neomodel to abstract some operations, but raw CQL is required for many tasks.
- Neo4j does not support multiple logical databases in community edition, so we emulate them.
- This is achieved by using a 'db' property (database name) and 'kg' property (graph name) on nodes.

### 6.10.2 Constructor & Destructor Documentation

#### 6.10.2.1 [\\_\\_init\\_\\_\(\)](#)

```
None __init__ (
    self,
    bool verbose = False )
```

Creates a new Neo4j connector.

#### Parameters

<i>verbose</i>	Whether to print success and failure messages.
----------------	--

Reimplemented from [DatabaseConnector](#).

### 6.10.3 Member Function Documentation

#### 6.10.3.1 [\\_split\\_combined\(\)](#)

```
List[str] _split_combined (
    self,
    str multi_query ) [protected]
```

Divides a string into non-divisible CQL queries, ignoring comments.

#### Parameters

<i>multi_query</i>	A string containing multiple queries.
--------------------	---------------------------------------

**Returns**

A list of single-query strings.

Reimplemented from [DatabaseConnector](#).

**6.10.3.2 add\_triple()**

```
None add_triple (
    self,
    str subject,
    str relation,
    str object_,
    bool _delete_init = True )
```

Add a semantic triple to the graph using raw Cypher.

1. Finds nodes by exact match on `name` attribute.
2. Creates a relationship between them with the given label.

**Parameters**

<code>subject</code>	A string representing the entity performing an action.
<code>relation</code>	A string describing the action.
<code>object_</code>	A string representing the entity being acted upon.
<code>_delete_init</code>	Whether to delete the dummy node added during database creation.

**Exceptions**

<code>Log.Failure</code>	If the triple cannot be added to our graph database.
--------------------------	--

**6.10.3.3 change\_database()**

```
None change_database (
    self,
    str new_database )
```

Update the connection URI to reference a different database in the same engine.

**Note**

Neo4j does not accept database names routed through the connection string.

**Parameters**

<code>new_database</code>	The name of the database to connect to.
---------------------------	---

Reimplemented from [DatabaseConnector](#).

#### 6.10.3.4 change\_graph()

```
None change_graph (
    self,
    str graph_name )
```

Sets graph\_name to create new a Knowledge Graph (collection of triples).

Similar to creating tables in SQL and collections in Mongo.

##### Note

This change will apply to any new nodes created.

##### Parameters

<i>graph_name</i>	A string corresponding to the 'kg' node attribute.
-------------------	--

#### 6.10.3.5 check\_connection()

```
bool check_connection (
    self,
    str log_source,
    bool raise_error )
```

Minimal connection test to determine if our connection string is valid.

Connect to Neo4j using #####

##### Parameters

<i>log_source</i>	The Log class prefix indicating which method is performing the check.
<i>raise_error</i>	Whether to raise an error on connection failure.

##### Returns

Whether the connection test was successful.

##### Exceptions

<i>RuntimeError</i>	If raise_error is True and the connection test fails to complete.
---------------------	---

#### 6.10.3.6 create\_database()

```
None create_database (
```

```
    self,  
    str database_name )
```

Create a fresh pseudo-database if it does not already exist.

#### Note

This change will apply to any new nodes created after [components.connectors.DatabaseConnector.change\\_database](#) is called.

#### Parameters

<i>database_name</i>	A database ID specifying the pseudo-database.
----------------------	---

#### Exceptions

<i>Log.Failure</i>	If we fail to create the requested database for any reason.
--------------------	---

Reimplemented from [DatabaseConnector](#).

### 6.10.3.7 database\_exists()

```
bool database_exists (  
    self,  
    str database_name )
```

Search for an existing database using the provided name.

#### Parameters

<i>database_name</i>	The name of a database to search for.
----------------------	---------------------------------------

#### Returns

Whether the database is visible to this connector.

Reimplemented from [DatabaseConnector](#).

### 6.10.3.8 delete\_dummy()

```
None delete_dummy (  
    self )
```

Delete the initial dummy node from the database.

#### Note

Call this method whenever real data is being added to avoid pollution.

### 6.10.3.9 drop\_database()

```
None drop_database (
    self,
    str database_name )
```

Delete all nodes stored under a particular database name.

#### Parameters

<i>database_name</i>	A database ID specifying the pseudo-database.
----------------------	---

#### Exceptions

<i>Log.Failure</i>	If we fail to drop the target database for any reason.
--------------------	--

Reimplemented from [DatabaseConnector](#).

### 6.10.3.10 execute\_query()

```
Optional[DataFrame] execute_query (
    self,
    str query )
```

Send a single Cypher query to Neo4j.

#### Note

If a result is returned, it will be converted to a DataFrame.

#### Parameters

<i>query</i>	A single query to perform on the database.
--------------	--

#### Returns

DataFrame containing the result of the query, or None

#### Exceptions

<i>Log.Failure</i>	If the query fails to execute.
--------------------	--------------------------------

Reimplemented from [DatabaseConnector](#).

### 6.10.3.11 get\_all\_triples()

```
DataFrame get_all_triples (
    self )
```

Return all triples in the current pseudo-database as a pandas DataFrame.

#### Exceptions

<i>Log.Failure</i>	If the query fails to retrieve the requested DataFrame.
--------------------	---

#### 6.10.3.12 get\_dataframe()

```
Optional[DataFrame] get_dataframe (
    self,
    str name )
```

Automatically generate and run a query for the specified Knowledge Graph collection.

- Fetches all public node attributes, the internal ID, and all labels (e.g. :Person :Character)
- Does not explode lists or nested values
- Different approach than DocumentConnector because our node attributes are usually flat key:value already.

#### Parameters

<i>name</i>	The name of an existing table or collection in the database.
-------------	--

#### Returns

DataFrame containing the requested data, or None

#### Exceptions

<i>Log.Failure</i>	If we fail to create the requested DataFrame for any reason.
--------------------	--

Reimplemented from [DatabaseConnector](#).

#### 6.10.3.13 get\_edge\_counts()

```
DataFrame get_edge_counts (
    self,
    int top_n = 10 )
```

Return node names and their edge counts, ordered by edge count descending.

#### Parameters

<i>top_n</i>	Number of top nodes to return (by edge count). Default is 10.
--------------	---

**Returns**

DataFrame with columns: node\_name, edge\_count

**Exceptions**

<i>Log.Failure</i>	If the query fails to retrieve the requested DataFrame.
--------------------	---

**6.10.3.14 get\_unique()**

```
List[str] get_unique (
    self,
    str key )
```

Retrieve all unique values for a specified node property.

Queries all nodes in the database and extracts distinct values for the given key.

**Parameters**

<i>key</i>	The node property name to extract unique values from (e.g. 'db' or 'kg').
------------	---

**Returns**

A list of unique values for the specified key, or an empty list if none exist.

**Exceptions**

<i>Log.Failure</i>	If the query fails to execute.
--------------------	--------------------------------

**6.10.3.15 IS\_DUMMY\_()**

```
str IS_DUMMY_ (
    self,
    str alias = 'n' )
```

Generates Cypher code to select dummy nodes inside a WHERE clause.

Usage: MATCH (n) WHERE {self.IS\_DUMMY\_('n')};

**Returns**

A string containing Cypher code.

### 6.10.3.16 NOT\_DUMMY\_()

```
str NOT_DUMMY_ (
    self,
    str alias = 'n' )
```

Generates Cypher code to select non-dummy nodes inside a WHERE clause.

Usage: MATCH (n) WHERE {self.NOT\_DUMMY\_('n')};

#### Returns

A string containing Cypher code.

### 6.10.3.17 print\_nodes()

```
None print_nodes (
    self,
    int max_rows = 20,
    int max_col_width = 50 )
```

Print all nodes and edges in the current pseudo-database with row/column formatting.

### 6.10.3.18 print\_triples()

```
None print_triples (
    self,
    int max_rows = 20,
    int max_col_width = 50 )
```

Print all nodes and edges in the current pseudo-database with row/column formatting.

### 6.10.3.19 SAME\_DB\_KG\_()

```
str SAME_DB_KG_ (
    self )
```

Generates a Cypher pattern dictionary to match nodes by current database and graph name.

Usage: MATCH (n {self.SAME\_DB\_KG\_()})

#### Returns

A string containing Cypher code.

### 6.10.3.20 test\_connection()

```
bool test_connection (
    self,
    bool raise_error = True )
```

Establish a basic connection to the Neo4j database.

Can be configured to fail silently, which enables retries or external handling.

**Parameters**

<i>raise_error</i>	Whether to raise an error on connection failure.
--------------------	--

**Returns**

Whether the connection test was successful.

**Exceptions**

<i>Log.Failure</i>	If <i>raise_error</i> is True and the connection test fails to complete.
--------------------	--

Reimplemented from [Connector](#).

## 6.10.4 Member Data Documentation

### 6.10.4.1 `_created_dummy`

`_created_dummy` [protected]

### 6.10.4.2 `connection_string`

`connection_string`

### 6.10.4.3 `database_name`

`database_name`

### 6.10.4.4 `graph_name`

`graph_name`

### 6.10.4.5 `verbose`

`verbose`

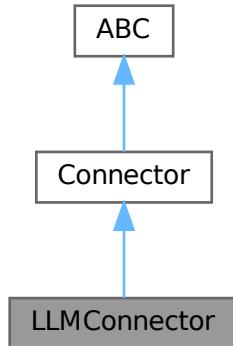
The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/components/[fact\\_storage.py](#)

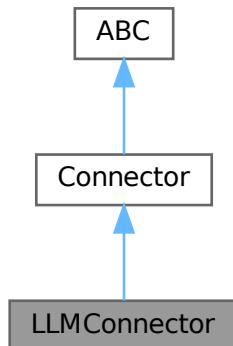
## 6.11 LLMConnector Class Reference

Connector for prompting and returning LLM output (raw text/JSON) via LangChain.

Inheritance diagram for LLMConnector:



Collaboration diagram for LLMConnector:



### Public Member Functions

- `__init__(self, float temperature=0, str system_prompt="You are a helpful assistant.")`  
*Initialize the connector.*
- `configure(self)`  
*Initialize the LangChain LLM using environment credentials.*
- `test_connection(self)`  
*Send a trivial prompt to verify LLM connectivity.*

- str [execute\\_full\\_query](#) (self, str [system\\_prompt](#), str [human\\_prompt](#))  
*Send a single prompt to the LLM with separate system and human instructions.*
- str [execute\\_query](#) (self, str [query](#))  
*Send a single prompt through the connection and return raw LLM output.*
- str [execute\\_file](#) (self, str [filename](#))  
*Run a single prompt from a file.*

## Public Attributes

- [model\\_name](#)
- [temperature](#)
- [system\\_prompt](#)
- [llm](#)

### 6.11.1 Detailed Description

Connector for prompting and returning LLM output (raw text/JSON) via LangChain.

#### Note

The method [components.text\\_processing.LLMConnector.execute\\_query](#) simplifies the prompt process.

### 6.11.2 Constructor & Destructor Documentation

#### 6.11.2.1 [\\_\\_init\\_\\_\(\)](#)

```
__init__ (
    self,
    float temperature = 0,
    str system_prompt = "You are a helpful assistant." )
```

Initialize the connector.

#### Note

Model name is specified in the .env file.

### 6.11.3 Member Function Documentation

#### 6.11.3.1 [configure\(\)](#)

```
configure (
    self )
```

Initialize the LangChain LLM using environment credentials.

Reads:

- OPENAI\_API\_KEY from .env for authentication
- LLM\_MODEL and LLM\_TEMPERATURE to override defaults

Reimplemented from [Connector](#).

### 6.11.3.2 execute\_file()

```
str execute_file (
    self,
    str filename )
```

Run a single prompt from a file.

Reads the entire file as a single string and sends it to execute\_query.

#### Parameters

<i>filename</i>	Path to the prompt file (.txt)
-----------------	--------------------------------

#### Returns

Raw LLM response as a string.

Reimplemented from [Connector](#).

### 6.11.3.3 execute\_full\_query()

```
str execute_full_query (
    self,
    str system_prompt,
    str human_prompt )
```

Send a single prompt to the LLM with separate system and human instructions.

### 6.11.3.4 execute\_query()

```
str execute_query (
    self,
    str query )
```

Send a single prompt through the connection and return raw LLM output.

#### Parameters

<i>query</i>	A single string prompt to send to the LLM.
--------------	--

#### Returns

Raw LLM response as a string.

Reimplemented from [Connector](#).

### 6.11.3.5 test\_connection()

```
test_connection (
    self )
```

Send a trivial prompt to verify LLM connectivity.

#### Returns

Whether the prompt executed successfully.

Reimplemented from [Connector](#).

## 6.11.4 Member Data Documentation

### 6.11.4.1 llm

```
llm
```

### 6.11.4.2 model\_name

```
model_name
```

### 6.11.4.3 system\_prompt

```
system_prompt
```

### 6.11.4.4 temperature

```
temperature
```

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/components/[text\\_processing.py](#)

## 6.12 Log Class Reference

The Log class standardizes console output.

### Classes

- class [Failure](#)

## Static Public Member Functions

- None `success` (str prefix="PASS", str msg="", bool verbose=True)  
*A success message begins with a green prefix.*
- None `warn` (str prefix="PASS", str msg="", bool verbose=True)  
*A warning message begins with a yellow prefix.*
- None `fail` (str prefix="ERROR", str msg="", bool raise\_error=True, Optional[Exception] other\_error=None)  
*A failure message begins with a red prefix.*
- None `success_legacy` (str msg "")  
*A legacy success message begins with a Green Plus.*
- None `fail_legacy` (str msg "")  
*A legacy failure message begins with a Red X.*

## Static Public Attributes

- bool `USE_COLORS` = True  
*Enable ANSI colors in output.*
- str `GREEN` = "\033[32m"  
*ANSI code for green text.*
- str `RED` = "\033[31m"  
*ANSI code for red text.*
- str `YELLOW` = "\033[33m"  
*ANSI code for yellow text.*
- str `BRIGHT` = "\033[93m"  
*ANSI code for bright yellow / cream.*
- str `WHITE` = "\033[0m"  
*ANSI code to reset color.*
- str `SUCCESS_COLOR` = `GREEN`  
*ANSI color applied to the prefix of success messages.*
- str `WARNING_COLOR` = `YELLOW`  
*ANSI color applied to the prefix of ignored fail messages.*
- str `FAILURE_COLOR` = `RED`  
*ANSI color applied to the prefix of critical fail messages.*
- str `MSG_COLOR` = `BRIGHT`  
*ANSI color applied to the body of every Log message.*
- bool `FULL_DF` = False  
*When printing the results of a query.*
- str `conn_abc` = "BASE CONNECTOR: "
- str `db_conn_abc` = "CONNECTOR: "
- str `rel_db` = "REL DB: "
- str `gr_db` = "GRAPH DB: "
- str `doc_db` = "DOCS DB: "
- str `bad_addr` = "BAD ADDRESS: "
- f `msg_bad_addr` = lambda connection\_string"Failed to connect on {connection\_string}"
- str `bad_path` = "FILE NOT FOUND: "
- f `msg_bad_path` = lambda file\_path"Failed to open file '{file\_path}'"
- f `msg_good_path` = lambda file\_path"Reading contents of file '{file\_path}'"
- f `msg_good_exec_f` = lambda file\_path"Finished executing queries from '{file\_path}'"
- f `msg_bad_exec_f` = lambda file\_path"Error occurred while executing queries from '{file\_path}'"
- f `msg_db_connect` = lambda database\_name"Successfully connected to database: {database\_name}"
- str `good_val` = "VALID RESULT: "

- str `bad_val` = "INCORRECT RESULT: "
- f `msg_compare` = lambda observed, expected"Expected {expected}, got {observed}"
- tuple `msg_result`
- tuple `msg_good_table`
- tuple `msg_good_coll`
- tuple `msg_good_graph`
- f `msg_bad_table` = lambda name"Table '{name}' not found"
- f `msg_bad_coll` = lambda name"Collection '{name}' not found"
- f `msg_bad_graph` = lambda name"Graph '{name}' not found"
- str `test_conn` = "CONNECTION TEST: "
- str `test_basic` = "BASIC: "
- str `test_info` = "DB INFO: "
- str `test_df` = "GET DF: "
- str `test_tmp_db` = "CREATE DB: "
- str `msg_unknown_error` = "An unhandled error occurred."
- str `get_df` = "GET\_DF: "
- str `create_db` = "CREATE\_DB: "
- str `drop_db` = "DROP\_DB: "
- str `run_q` = "QUERY: "
- str `run_f` = "FILE EXEC: "
- f `msg_success_managed_db` = lambda managed, database\_name"Successfully {managed} database '{database\_name}'"
- tuple `msg_fail_manage_db`
- f `msg_fail_parse` = lambda alias, bad\_value, expected\_type"Could not convert {alias} with value {bad\_value} to type {expected\_type}"
- tuple `msg_multiple_query`
- f `msg_good_exec_q` = lambda query"Executed successfully:\n'{query}'"
- f `msg_good_exec_qr` = lambda query, results"Executed successfully:\n'{query}'\n{Log.msg\_result(results)}"
- f `msg_bad_exec_q` = lambda query"Failed to execute query:\n'{query}'"
- str `kg` = "KG: "
- str `pytest_db` = "PYTEST (DB): "
- str `db_exists` = "DB\_EXIST: "
- f `msg_db_exists` = lambda database\_name"Database '{database\_name}' already exists."
- f `msg_db_not_found` = lambda database\_name, connection\_string"Could not find database '{database\_name}' using connection '{connection\_string}'"
- f `msg_db_current` = lambda database\_name"Cannot drop database '{database\_name}' while connected to it!"
- str `swap_db` = "SWAP\_DB: "
- str `swap_kg` = "SWAP\_GRAPH: "
- f `msg_swap_db` = lambda old\_db, new\_db"Switched from database '{old\_db}' to database '{new\_db}'"
- f `msg_swap_kg` = lambda old\_kg, new\_kg"Switched from graph '{old\_kg}' to graph '{new\_kg}'"
- str `get_unique` = "UNIQUE: "

### 6.12.1 Detailed Description

The Log class standardizes console output.

## 6.12.2 Member Function Documentation

### 6.12.2.1 fail()

```
None fail (
    str  prefix = "ERROR",
    str  msg = "",
    bool raise_error = True,
    Optional[Exception] other_error = None ) [static]
```

A failure message begins with a red prefix.

**Parameters**

<i>prefix</i>	The context of the message.
<i>msg</i>	The message to print.
<i>raise_error</i>	Whether to raise an error.
<i>other_error</i>	Another Exception resulting from this failure.

**Exceptions**

<i>Log.Failure</i>	If <i>raise_error</i> is True
--------------------	-------------------------------

**6.12.2.2 fail\_legacy()**

```
None fail_legacy (
    str msg = "" ) [static]
```

A legacy failure message begins with a Red X.

**Parameters**

<i>msg</i>	The message to print.
------------	-----------------------

**6.12.2.3 success()**

```
None success (
    str prefix = "PASS",
    str msg = "",
    bool verbose = True ) [static]
```

A success message begins with a green prefix.

**Parameters**

<i>prefix</i>	The context of the message.
<i>msg</i>	The message to print.
<i>verbose</i>	Whether to actually print. Saves space and reduces nested if statements.

**6.12.2.4 success\_legacy()**

```
None success_legacy (
    str msg = "" ) [static]
```

A legacy success message begins with a Green Plus.

**Parameters**

<i>msg</i>	The message to print.
------------	-----------------------

**6.12.2.5 warn()**

```
None warn (
    str prefix = "PASS",
    str msg = "",
    bool verbose = True ) [static]
```

A warning message begins with a yellow prefix.

**Parameters**

<i>prefix</i>	The context of the message.
<i>msg</i>	The message to print.
<i>verbose</i>	Whether to actually print. Saves space and reduces nested if statements.

**6.12.3 Member Data Documentation****6.12.3.1 bad\_addr**

```
str bad_addr = "BAD ADDRESS: " [static]
```

**6.12.3.2 bad\_path**

```
str bad_path = "FILE NOT FOUND: " [static]
```

**6.12.3.3 bad\_val**

```
str bad_val = "INCORRECT RESULT: " [static]
```

**6.12.3.4 BRIGHT**

```
str BRIGHT = "\033[93m" [static]
```

ANSI code for bright yellow / cream.

**6.12.3.5 conn\_abc**

```
str conn_abc = "BASE CONNECTOR: " [static]
```

**6.12.3.6 create\_db**

```
str create_db = "CREATE_DB: " [static]
```

**6.12.3.7 db\_conn\_abc**

```
str db_conn_abc = "CONNECTOR: " [static]
```

**6.12.3.8 db\_exists**

```
str db_exists = "DB_EXIST: " [static]
```

**6.12.3.9 doc\_db**

```
str doc_db = "DOCS DB: " [static]
```

**6.12.3.10 drop\_db**

```
str drop_db = "DROP_DB: " [static]
```

**6.12.3.11 FAILURE\_COLOR**

```
str FAILURE_COLOR = RED [static]
```

ANSI color applied to the prefix of critical fail messages.

**6.12.3.12 FULL\_DF**

```
bool FULL_DF = False [static]
```

When printing the results of a query.

**6.12.3.13 get\_df**

```
str get_df = "GET_DF: " [static]
```

**6.12.3.14 get\_unique**

```
str get_unique = "UNIQUE: " [static]
```

### 6.12.3.15 **good\_val**

```
str good_val = "VALID RESULT: " [static]
```

### 6.12.3.16 **gr\_db**

```
str gr_db = "GRAPH DB: " [static]
```

### 6.12.3.17 **GREEN**

```
str GREEN = "\033[32m" [static]
```

ANSI code for green text.

### 6.12.3.18 **kg**

```
str kg = "KG: " [static]
```

### 6.12.3.19 **msg\_bad\_addr**

```
f msg_bad_addr = lambda connection_string"Failed to connect on {connection_string}" [static]
```

### 6.12.3.20 **msg\_bad\_coll**

```
f msg_bad_coll = lambda name"Collection '{name}' not found" [static]
```

### 6.12.3.21 **msg\_bad\_exec\_f**

```
f msg_bad_exec_f = lambda file_path"Error occurred while executing queries from '{file_path}'" [static]
```

### 6.12.3.22 **msg\_bad\_exec\_q**

```
f msg_bad_exec_q = lambda query"Failed to execute query:\n'{query}'" [static]
```

### 6.12.3.23 **msg\_bad\_graph**

```
f msg_bad_graph = lambda name"Graph '{name}' not found" [static]
```

### 6.12.3.24 **msg\_bad\_path**

```
f msg_bad_path = lambda file_path"Failed to open file '{file_path}'" [static]
```

**6.12.3.25 msg\_bad\_table**

```
f msg_bad_table = lambda name"Table '{name}' not found" [static]
```

**6.12.3.26 MSG\_COLOR**

```
str MSG_COLOR = BRIGHT [static]
```

ANSI color applied to the body of every Log message.

**6.12.3.27 msg\_compare**

```
f msg_compare = lambda observed, expected"Expected {expected}, got {observed}" [static]
```

**6.12.3.28 msg\_db\_connect**

```
f msg_db_connect = lambda database_name"Successfully connected to database: {database_name}" [static]
```

**6.12.3.29 msg\_db\_current**

```
f msg_db_current = lambda database_name"Cannot drop database '{database_name}' while connected to it!" [static]
```

**6.12.3.30 msg\_db\_exists**

```
f msg_db_exists = lambda database_name"Database '{database_name}' already exists." [static]
```

**6.12.3.31 msg\_db\_not\_found**

```
f msg_db_not_found = lambda database_name, connection_string"Could not find database '{database_name}' using connection '{connection_string}'" [static]
```

**6.12.3.32 msg\_fail\_manage\_db**

```
tuple msg_fail_manage_db [static]
```

**Initial value:**

```
= (
    lambda manage, database_name, connection_string: f"Failed to {manage} database '{database_name}' on connection {connection_string}"
)
```

### 6.12.3.33 msg\_fail\_parse

```
f msg_fail_parse = lambda alias, bad_value, expected_type"Could not convert {alias} with value {bad_value} to type {expected_type}" [static]
```

### 6.12.3.34 msg\_good\_coll

```
tuple msg_good_coll [static]
```

**Initial value:**

```
= (
    lambda name, df: f
)
```

### 6.12.3.35 msg\_good\_exec\_f

```
f msg_good_exec_f = lambda file_path"Finished executing queries from '{file_path}'" [static]
```

### 6.12.3.36 msg\_good\_exec\_q

```
f msg_good_exec_q = lambda query"Executed successfully:\n'{query}'" [static]
```

### 6.12.3.37 msg\_good\_exec\_qr

```
f msg_good_exec_qr = lambda query, results"Executed successfully:\n'{query}'\n{Log.msg_result(results)}" [static]
```

### 6.12.3.38 msg\_good\_graph

```
tuple msg_good_graph [static]
```

**Initial value:**

```
= (
    lambda name, df: f
)
```

### 6.12.3.39 msg\_good\_path

```
f msg_good_path = lambda file_path"Reading contents of file '{file_path}'" [static]
```

### 6.12.3.40 msg\_good\_table

```
tuple msg_good_table [static]
```

**Initial value:**

```
= (
    lambda name, df: f
)
```

#### 6.12.3.41 msg\_multiple\_query

```
tuple msg_multiple_query [static]
```

**Initial value:**

```
= (
    lambda n_queries, query: f"A combined query ({n_queries} results) was executed as a single query.
    Extra results were discarded. Query:{query}"
)
```

#### 6.12.3.42 msg\_result

```
tuple msg_result [static]
```

**Initial value:**

```
= (
    lambda results: f
)
```

#### 6.12.3.43 msg\_success\_managed\_db

```
f msg_success_managed_db = lambda managed, database_name"Successfully {managed} database '{database←
_name}'" [static]
```

#### 6.12.3.44 msg\_swap\_db

```
f msg_swap_db = lambda old_db, new_db"Switched from database '{old_db}' to database '{new←
_db}'" [static]
```

#### 6.12.3.45 msg\_swap\_kg

```
f msg_swap_kg = lambda old_kg, new_kg"Switched from graph '{old_kg}' to graph '{new_kg}'"
[static]
```

#### 6.12.3.46 msg\_unknown\_error

```
str msg_unknown_error = "An unhandled error occurred." [static]
```

#### 6.12.3.47 pytest\_db

```
str pytest_db = "PYTEST (DB): " [static]
```

#### 6.12.3.48 RED

```
str RED = "\033[31m" [static]
```

ANSI code for red text.

**6.12.3.49 rel\_db**

```
str rel_db = "REL DB: " [static]
```

**6.12.3.50 run\_f**

```
str run_f = "FILE EXEC: " [static]
```

**6.12.3.51 run\_q**

```
str run_q = "QUERY: " [static]
```

**6.12.3.52 SUCCESS\_COLOR**

```
str SUCCESS_COLOR = GREEN [static]
```

ANSI color applied to the prefix of success messages.

**6.12.3.53 swap\_db**

```
str swap_db = "SWAP_DB: " [static]
```

**6.12.3.54 swap\_kg**

```
str swap_kg = "SWAP_GRAPH: " [static]
```

**6.12.3.55 test\_basic**

```
str test_basic = "BASIC: " [static]
```

**6.12.3.56 test\_conn**

```
str test_conn = "CONNECTION TEST: " [static]
```

**6.12.3.57 test\_df**

```
str test_df = "GET DF: " [static]
```

**6.12.3.58 test\_info**

```
str test_info = "DB INFO: " [static]
```

**6.12.3.59 test\_tmp\_db**

```
str test_tmp_db = "CREATE DB: " [static]
```

**6.12.3.60 USE\_COLORS**

```
bool USE_COLORS = True [static]
```

Enable ANSI colors in output.

**6.12.3.61 WARNING\_COLOR**

```
str WARNING_COLOR = YELLOW [static]
```

ANSI color applied to the prefix of ignored fail messages.

**6.12.3.62 WHITE**

```
str WHITE = "\033[0m" [static]
```

ANSI code to reset color.

**6.12.3.63 YELLOW**

```
str YELLOW = "\033[33m" [static]
```

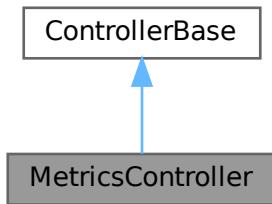
ANSI code for yellow text.

The documentation for this class was generated from the following file:

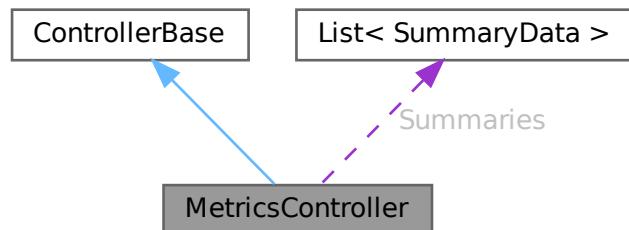
- /home/runner/work/dsci-capstone/dsci-capstone/src/[util.py](#)

## 6.13 MetricsController Class Reference

Inheritance diagram for MetricsController:



Collaboration diagram for MetricsController:



### Public Member Functions

- `MetricsController (ILogger< MetricsController > logger, IHubContext< MetricsHub > hubContext)`
- `async Task< IActionResult > Post ([FromBody] SummaryData summary)`
- `IActionResult GetIndex (int id)`
- `IActionResult GetAll ()`

### Private Attributes

- `readonly ILogger< MetricsController > _logger`
- `readonly IHubContext< MetricsHub > _hubContext`

### Static Private Attributes

- `static readonly List< SummaryData > Summaries = new()`

## 6.13.1 Constructor & Destructor Documentation

### 6.13.1.1 MetricsController()

```
MetricsController (
    ILogger< MetricsController > logger,
    IHubContext< MetricsHub > hubContext )
```

## 6.13.2 Member Function Documentation

### 6.13.2.1 GetAll()

```
IActionResult GetAll ( )
```

### 6.13.2.2 GetIndex()

```
IActionResult GetIndex (
    int id )
```

### 6.13.2.3 Post()

```
async Task< IActionResult > Post (
    [FromBody] SummaryData summary )
```

## 6.13.3 Member Data Documentation

### 6.13.3.1 \_hubContext

```
readonly IHubContext<MetricsHub> _hubContext [private]
```

### 6.13.3.2 \_logger

```
readonly ILogger<MetricsController> _logger [private]
```

### 6.13.3.3 Summaries

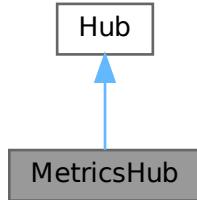
```
readonly List<SummaryData> Summaries = new() [static], [private]
```

The documentation for this class was generated from the following file:

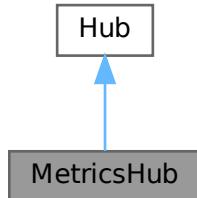
- /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Controllers/MetricsController.cs

## 6.14 MetricsHub Class Reference

Inheritance diagram for MetricsHub:



Collaboration diagram for MetricsHub:



### Public Member Functions

- `MetricsHub (ILogger< MetricsHub >? logger=null)`
- override async Task `OnConnectedAsync ()`
- override async Task `OnDisconnectedAsync (Exception? exception)`

### Private Attributes

- `readonly? ILogger< MetricsHub > _logger`

### 6.14.1 Constructor & Destructor Documentation

#### 6.14.1.1 MetricsHub()

```
MetricsHub (
    ILogger< MetricsHub >? logger = null )
```

## 6.14.2 Member Function Documentation

### 6.14.2.1 OnConnectedAsync()

```
override async Task OnConnectedAsync ( )
```

### 6.14.2.2 OnDisconnectedAsync()

```
override async Task OnDisconnectedAsync (
    Exception? exception )
```

## 6.14.3 Member Data Documentation

### 6.14.3.1 \_logger

```
readonly? ILogger<MetricsHub> _logger [private]
```

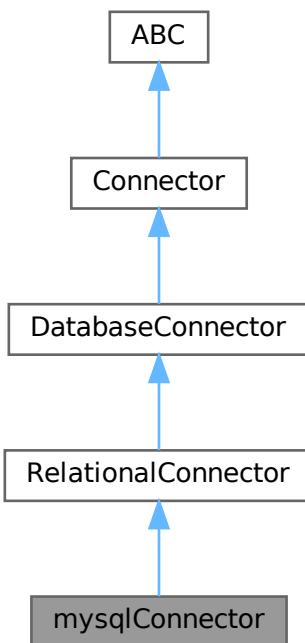
The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Hubs/MetricsHub.cs

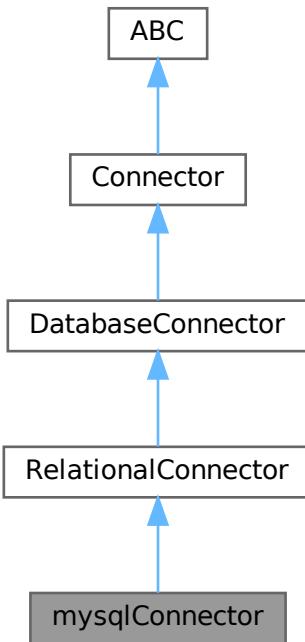
## 6.15 mysqlConnector Class Reference

A relational database connector configured for MySQL.

Inheritance diagram for mysqlConnector:



Collaboration diagram for mysqlConnector:



### Public Member Functions

- None `__init__` (self, bool `verbose=False`)

*Configures the relational connector.*

### Public Member Functions inherited from `RelationalConnector`

- "RelationalConnector" `from_env` (cls, bool `verbose=False`)
 

*Decides what type of relational connector to create using the .env file.*
- None `change_database` (self, str `new_database`)
 

*Update the connection URI to reference a different database in the same engine.*
- bool `test_connection` (self, bool `raise_error=True`)
 

*Establish a basic connection to the database.*
- bool `check_connection` (self, str `log_source`, bool `raise_error`)
 

*Minimal connection test to determine if our connection string is valid.*
- Optional[DataFrame] `execute_query` (self, str `query`)
 

*Send a single command to the database connection.*
- Optional[DataFrame] `get_dataframe` (self, str `name`)
 

*Automatically generate and run a query for the specified table using SQLAlchemy.*
- None `create_database` (self, str `database_name`)
 

*Use the current database connection to create a sibling database in this engine.*
- None `drop_database` (self, str `database_name=""`)
 

*Delete all data stored in a particular database.*
- bool `database_exists` (self, str `database_name`)
 

*Search for an existing database using the provided name.*

## Public Member Functions inherited from DatabaseConnector

- None `configure` (self, str DB, str database\_name)  
*Read connection settings from the .env file.*
- List[Optional[DataFrame]] `execute_combined` (self, str multi\_query)  
*Run several database commands in sequence.*
- List[Optional[DataFrame]] `execute_file` (self, str filename)  
*Run several database commands from a file.*

## Static Public Attributes

- dict `specific_queries`

## Additional Inherited Members

### Public Attributes inherited from RelationalConnector

- `database_name`
- `verbose`
- `connection_string`
- `db_type`

### Public Attributes inherited from DatabaseConnector

- `verbose`  
*Whether to print debug messages.*
- `db_type`
- `db_engine`
- `username`
- `password`
- `host`
- `port`
- `connection_string`

### Protected Member Functions inherited from RelationalConnector

- List[str] `_split_combined` (self, str multi\_query)  
*Divides a string into non-divisible SQL queries using sqlparse.*

### Protected Member Functions inherited from DatabaseConnector

- bool `_is_single_query` (self, str query)  
*Checks if a string contains multiple queries.*

### 6.15.1 Detailed Description

A relational database connector configured for MySQL.

#### Note

Should be hidden from the user using a factory method.

### 6.15.2 Constructor & Destructor Documentation

#### 6.15.2.1 `__init__()`

```
None __init__ (
    self,
    bool verbose = False )
```

Configures the relational connector.

#### Parameters

<code>verbose</code>	Whether to print success and failure messages.
----------------------	--

Reimplemented from [RelationalConnector](#).

### 6.15.3 Member Data Documentation

#### 6.15.3.1 `specific_queries`

```
dict specific_queries [static]
```

#### Initial value:

```
= {
    "MYSQL": [
        "SELECT DATABASE();", # Single value, name of the current database.
        "SHOW DATABASES;", # List of databases the secondary user can access.
    ] # List of all databases in the database engine.
}
```

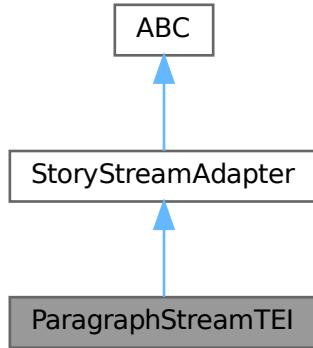
The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/components/[connectors.py](#)

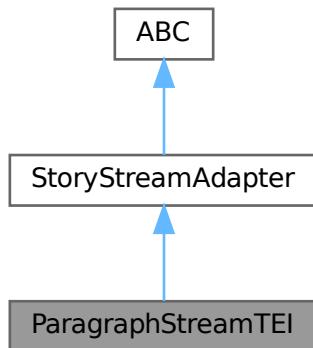
## 6.16 ParagraphStreamTEI Class Reference

Streams paragraphs from a TEI file as Chunk objects.

Inheritance diagram for ParagraphStreamTEI:



Collaboration diagram for ParagraphStreamTEI:



## Public Member Functions

- `__init__ (self, str tei_path, int book_id, int story_id, list[str] allowed_chapters=None, str start_inclusive="", str end_inclusive="")`  
*Create a ParagraphStreamTEI object.*
- `Iterator[Chunk] stream_segments (self)`  
*Yields sanitized parts of a book.*
- `List[Chunk] pre_compute_segments (self)`  
*Splits the target book into paragraphs.*

## Public Member Functions inherited from StoryStreamAdapter

- `Chunk stream_paragraphs (self)`  
*Concrete helper method to split segments into paragraphs.*
- `str stream_sentences (self)`  
*Concrete helper method to split paragraphs into sentences.*

## Public Attributes

- `tei_path`
- `book_id`
- `story_id`
- `allowed_chapters`
- `start_inclusive`
- `end_inclusive`
- `lines`
- `root`
- `chunks`
- `xml_namespace`

## Static Public Attributes

- dict `xml_namespace` = {"tei": "http://www.tei-c.org/ns/1.0"}
- str `encoding` = "utf-8"

### 6.16.1 Detailed Description

Streams paragraphs from a TEI file as Chunk objects.

### 6.16.2 Constructor & Destructor Documentation

#### 6.16.2.1 \_\_init\_\_()

```
__init__ (
    self,
    str tei_path,
    int book_id,
    int story_id,
    list[str] allowed_chapters = None,
    str start_inclusive = "",
    str end_inclusive = "" )
```

Create a ParagraphStreamTEI object.

#### Parameters

<code>tei_path</code>	Path to an existing TEI XML file.
<code>book_id</code>	ID for this book.
<code>story_id</code>	ID for this story (may be same as book_id).
<code>allowed_chapters</code>	A list of valid chapter titles. Must exactly match the contents of head.
<code>start_inclusive</code>	(Optional) Unique string representing the start of the book.
<code>end_inclusive</code>	(Optional) Unique string representing the end of the book.

### 6.16.3 Member Function Documentation

#### 6.16.3.1 pre\_compute\_segments()

```
List[Chunk] pre_compute_segments (
    self )
```

Splits the target book into paragraphs.

Yields Chunk objects for each paragraph (

) in the TEI file. Uses etree Element.sourceline to approximate start/end line in TEI. Supports optional start\_inclusive / end\_inclusive boundaries to slice text and stop iteration. Computes progress percentages using character counts:

- story\_percent: progress through the entire story
- chapter\_percent: progress through the current chapter Populates self.chunks so they can be streamed as requested by interface

#### 6.16.3.2 stream\_segments()

```
Iterator[Chunk] stream_segments (
    self )
```

Yields sanitized parts of a book.

- Story segments usually correspond to chapters.
- They serve as borders between chunking operations, ensuring chunks do not span multiple chapters. Implementation is handled by child classes BookStream, etc.
- Segments should be pre-cleaned and must contain 1 paragraph per line with all other newlines removed.

Reimplemented from [StoryStreamAdapter](#).

### 6.16.4 Member Data Documentation

#### 6.16.4.1 allowed\_chapters

allowed\_chapters

#### 6.16.4.2 book\_id

book\_id

#### 6.16.4.3 chunks

chunks

#### 6.16.4.4 encoding

```
str encoding = "utf-8" [static]
```

#### 6.16.4.5 end\_inclusive

```
end_inclusive
```

#### 6.16.4.6 lines

```
lines
```

#### 6.16.4.7 root

```
root
```

#### 6.16.4.8 start\_inclusive

```
start_inclusive
```

#### 6.16.4.9 story\_id

```
story_id
```

#### 6.16.4.10 tei\_path

```
tei_path
```

#### 6.16.4.11 xml\_namespace [1/2]

```
dict xml_namespace = {"tei": "http://www.tei-c.org/ns/1.0"} [static]
```

#### 6.16.4.12 xml\_namespace [2/2]

```
xml_namespace
```

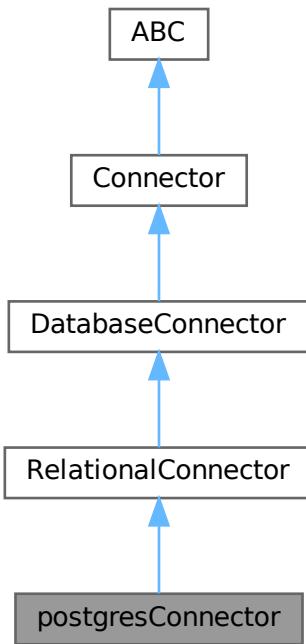
The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/components/[book\\_conversion.py](#)

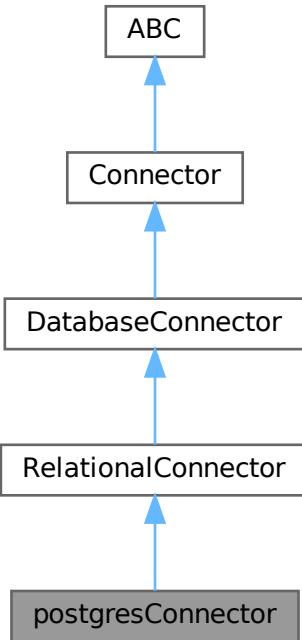
## 6.17 postgresConnector Class Reference

A relational database connector configured for PostgreSQL.

Inheritance diagram for postgresConnector:



Collaboration diagram for postgresConnector:



### Public Member Functions

- None `__init__` (self, bool `verbose=False`)

*Configures the relational connector.*

### Public Member Functions inherited from [RelationalConnector](#)

- "RelationalConnector" `from_env` (cls, bool `verbose=False`)  
*Decides what type of relational connector to create using the .env file.*
- None `change_database` (self, str `new_database`)  
*Update the connection URI to reference a different database in the same engine.*
- bool `test_connection` (self, bool `raise_error=True`)  
*Establish a basic connection to the database.*
- bool `check_connection` (self, str `log_source`, bool `raise_error`)  
*Minimal connection test to determine if our connection string is valid.*
- Optional[DataFrame] `execute_query` (self, str `query`)  
*Send a single command to the database connection.*
- Optional[DataFrame] `get_dataframe` (self, str `name`)  
*Automatically generate and run a query for the specified table using SQLAlchemy.*
- None `create_database` (self, str `database_name`)  
*Use the current database connection to create a sibling database in this engine.*
- None `drop_database` (self, str `database_name=""`)  
*Delete all data stored in a particular database.*
- bool `database_exists` (self, str `database_name`)  
*Search for an existing database using the provided name.*

## Public Member Functions inherited from DatabaseConnector

- None `configure` (self, str DB, str database\_name)  
*Read connection settings from the .env file.*
- List[Optional[DataFrame]] `execute_combined` (self, str multi\_query)  
*Run several database commands in sequence.*
- List[Optional[DataFrame]] `execute_file` (self, str filename)  
*Run several database commands from a file.*

## Static Public Attributes

- dict `specific_queries`

## Additional Inherited Members

### Public Attributes inherited from RelationalConnector

- `database_name`
- `verbose`
- `connection_string`
- `db_type`

### Public Attributes inherited from DatabaseConnector

- `verbose`  
*Whether to print debug messages.*
- `db_type`
- `db_engine`
- `username`
- `password`
- `host`
- `port`
- `connection_string`

### Protected Member Functions inherited from RelationalConnector

- List[str] `_split_combined` (self, str multi\_query)  
*Divides a string into non-divisible SQL queries using sqlparse.*

### Protected Member Functions inherited from DatabaseConnector

- bool `_is_single_query` (self, str query)  
*Checks if a string contains multiple queries.*

### 6.17.1 Detailed Description

A relational database connector configured for PostgreSQL.

#### Note

Should be hidden from the user using a factory method.

### 6.17.2 Constructor & Destructor Documentation

#### 6.17.2.1 `__init__()`

```
None __init__ (
    self,
    bool verbose = False )
```

Configures the relational connector.

#### Parameters

<code>verbose</code>	Whether to print success and failure messages.
----------------------	--

Reimplemented from [RelationalConnector](#).

### 6.17.3 Member Data Documentation

#### 6.17.3.1 `specific_queries`

```
dict specific_queries [static]
```

#### Initial value:

```
= {
    "POSTGRES": [
        "SELECT current_database();", # Single value, name of the current database.
        "SELECT datname FROM pg_database;", # List of ALL databases, even ones we cannot access.
    ] # List of all databases in the database engine.
}
```

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/components/[connectors.py](#)

## 6.18 PRF1Metric Class Reference

#### Properties

- string `Name` [get, set]
- double `Precision` [get, set]
- double `Recall` [get, set]
- double `F1Score` [get, set]

## 6.18.1 Property Documentation

### 6.18.1.1 F1Score

```
double F1Score [get], [set]
```

### 6.18.1.2 Name

```
string Name [get], [set]
```

### 6.18.1.3 Precision

```
double Precision [get], [set]
```

### 6.18.1.4 Recall

```
double Recall [get], [set]
```

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/PRF1Metric.cs

## 6.19 QAItem Class Reference

### Properties

- string [Question](#) [get, set]
- string [GoldAnswer](#) [get, set]
- string [GeneratedAnswer](#) [get, set]
- bool? [IsCorrect](#) [get, set]
- double? [Accuracy](#) [get, set]

## 6.19.1 Property Documentation

### 6.19.1.1 Accuracy

```
double? Accuracy [get], [set]
```

### 6.19.1.2 GeneratedAnswer

```
string GeneratedAnswer [get], [set]
```

### 6.19.1.3 GoldAnswer

```
string GoldAnswer [get], [set]
```

### 6.19.1.4 IsCorrect

```
bool? IsCorrect [get], [set]
```

### 6.19.1.5 Question

```
string Question [get], [set]
```

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/QAItem.cs

## 6.20 QAMetric Class Reference

### Properties

- List<[QAItem](#)> QAItems = new() [get, set]
- double AverageAccuracy [get]

### 6.20.1 Property Documentation

#### 6.20.1.1 AverageAccuracy

```
double AverageAccuracy [get]
```

#### 6.20.1.2 QAItems

```
List<QAItem> QAItems = new() [get], [set]
```

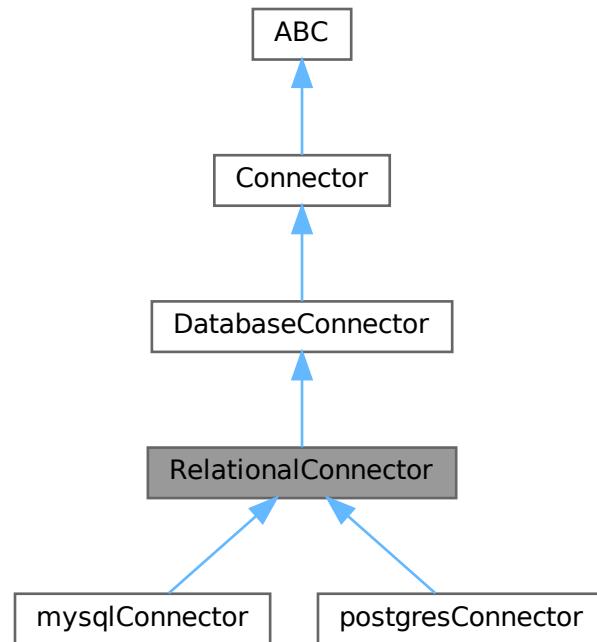
The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/QAMetric.cs

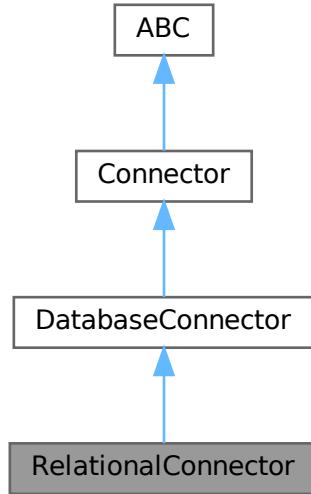
## 6.21 RelationalConnector Class Reference

Connector for relational databases (MySQL, PostgreSQL).

Inheritance diagram for RelationalConnector:



Collaboration diagram for RelationalConnector:



## Public Member Functions

- None `__init__` (self, bool `verbose`, List[str] `specific_queries`)  
*Creates a new database connector.*
- "RelationalConnector" `from_env` (cls, bool `verbose=False`)  
*Decides what type of relational connector to create using the .env file.*
- None `change_database` (self, str `new_database`)  
*Update the connection URI to reference a different database in the same engine.*
- bool `test_connection` (self, bool `raise_error=True`)  
*Establish a basic connection to the database.*
- bool `check_connection` (self, str `log_source`, bool `raise_error`)  
*Minimal connection test to determine if our connection string is valid.*
- Optional[DataFrame] `execute_query` (self, str `query`)  
*Send a single command to the database connection.*
- Optional[DataFrame] `get_dataframe` (self, str `name`)  
*Automatically generate and run a query for the specified table using SQLAlchemy.*
- None `create_database` (self, str `database_name`)  
*Use the current database connection to create a sibling database in this engine.*
- None `drop_database` (self, str `database_name=""`)  
*Delete all data stored in a particular database.*
- bool `database_exists` (self, str `database_name`)  
*Search for an existing database using the provided name.*

## Public Member Functions inherited from DatabaseConnector

- None `configure` (self, str DB, str database\_name)  
*Read connection settings from the .env file.*
- List[Optional[DataFrame]] `execute_combined` (self, str multi\_query)  
*Run several database commands in sequence.*
- List[Optional[DataFrame]] `execute_file` (self, str filename)  
*Run several database commands from a file.*

## Public Attributes

- `database_name`
- `verbose`
- `connection_string`
- `db_type`

## Public Attributes inherited from DatabaseConnector

- `verbose`  
*Whether to print debug messages.*
- `db_type`
- `db_engine`
- `username`
- `password`
- `host`
- `port`
- `connection_string`

## Protected Member Functions

- List[str] `_split_combined` (self, str multi\_query)  
*Divides a string into non-divisible SQL queries using sqlparse.*

## Protected Member Functions inherited from DatabaseConnector

- bool `_is_single_query` (self, str query)  
*Checks if a string contains multiple queries.*

## 6.21.1 Detailed Description

Connector for relational databases (MySQL, PostgreSQL).

Uses SQLAlchemy to abstract complex database operations. Hard-coded queries are used for testing purposes, and depend on the specific engine.

## 6.21.2 Constructor & Destructor Documentation

### 6.21.2.1 \_\_init\_\_()

```
None __init__ (
    self,
    bool verbose,
    List[str] specific_queries )
```

Creates a new database connector.

Use `components.connectors.RelationalConnector.from_env` instead (this is called by derived classes).

**Parameters**

<i>verbose</i>	Whether to print success and failure messages.
<i>specific_queries</i>	A list of helpful SQL queries.

Reimplemented from [DatabaseConnector](#).

Reimplemented in [mysqlConnector](#), and [postgresConnector](#).

### 6.21.3 Member Function Documentation

#### 6.21.3.1 `_split_combined()`

```
List[str] _split_combined (
    self,
    str multi_query ) [protected]
```

Divides a string into non-divisible SQL queries using `sqlparse`.

**Parameters**

<i>multi_query</i>	A string containing multiple queries.
--------------------	---------------------------------------

**Returns**

A list of single-query strings.

Reimplemented from [DatabaseConnector](#).

#### 6.21.3.2 `change_database()`

```
None change_database (
    self,
    str new_database )
```

Update the connection URI to reference a different database in the same engine.

**Parameters**

<i>new_database</i>	The name of the database to connect to.
---------------------	---

Reimplemented from [DatabaseConnector](#).

#### 6.21.3.3 `check_connection()`

```
bool check_connection (
    self,
```

```
    str log_source,
    bool raise_error )
```

Minimal connection test to determine if our connection string is valid.

Connect to our relational database using SQLAlchemy's engine.begin()

#### Parameters

<i>log_source</i>	The Log class prefix indicating which method is performing the check.
<i>raise_error</i>	Whether to raise an error on connection failure.

#### Returns

Whether the connection test was successful.

#### Exceptions

<i>RuntimeError</i>	If <i>raise_error</i> is True and the connection test fails to complete.
---------------------	--

### 6.21.3.4 `create_database()`

```
None create_database (
    self,
    str database_name )
```

Use the current database connection to create a sibling database in this engine.

#### Parameters

<i>database_name</i>	The name of the new database to create.
----------------------	---

#### Exceptions

<i>Log.Failure</i>	If we fail to create the requested database for any reason.
--------------------	---

Reimplemented from [DatabaseConnector](#).

### 6.21.3.5 `database_exists()`

```
bool database_exists (
    self,
    str database_name )
```

Search for an existing database using the provided name.

#### Parameters

<i>database_name</i>	The name of a database to search for.
----------------------	---------------------------------------

**Returns**

Whether the database is visible to this connector.

Reimplemented from [DatabaseConnector](#).

### 6.21.3.6 drop\_database()

```
None drop_database (
    self,
    str database_name = "" )
```

Delete all data stored in a particular database.

**Parameters**

<i>database_name</i>	The name of an existing database.
----------------------	-----------------------------------

**Exceptions**

<i>Log.Failure</i>	If we fail to drop the target database for any reason.
--------------------	--

Reimplemented from [DatabaseConnector](#).

### 6.21.3.7 execute\_query()

```
Optional[DataFrame] execute_query (
    self,
    str query )
```

Send a single command to the database connection.

**Note**

If a result is returned, it will be converted to a DataFrame.

**Parameters**

<i>query</i>	A single query to perform on the database.
--------------	--

**Returns**

DataFrame containing the result of the query, or None

**Exceptions**

<i>Log.Failure</i>	If the query fails to execute.
--------------------	--------------------------------

Reimplemented from [DatabaseConnector](#).

#### 6.21.3.8 from\_env()

```
"RelationalConnector" from_env (
    cls,
    bool verbose = False )
```

Decides what type of relational connector to create using the .env file.

##### Parameters

<i>verbose</i>	Whether to print success and failure messages.
----------------	--

##### Exceptions

<i>Log.Failure</i>	If the .env file contains an invalid DB_ENGINE value.
--------------------	---

#### 6.21.3.9 get\_dataframe()

```
Optional[DataFrame] get_dataframe (
    self,
    str name )
```

Automatically generate and run a query for the specified table using SQLAlchemy.

##### Parameters

<i>name</i>	The name of an existing table or collection in the database.
-------------	--

##### Returns

Sorted DataFrame containing the requested data, or None

##### Exceptions

<i>Log.Failure</i>	If we fail to create the requested DataFrame for any reason.
--------------------	--

Reimplemented from [DatabaseConnector](#).

#### 6.21.3.10 test\_connection()

```
bool test_connection (
    self,
    bool raise_error = True )
```

Establish a basic connection to the database.

Can be configured to fail silently, which enables retries or external handling.

**Parameters**

<code>raise_error</code>	Whether to raise an error on connection failure.
--------------------------	--

**Returns**

Whether the connection test was successful.

**Exceptions**

<code>LogFailure</code>	If <code>raise_error</code> is True and the connection test fails to complete.
-------------------------	--

Reimplemented from [Connector](#).

## 6.21.4 Member Data Documentation

### 6.21.4.1 `connection_string`

`connection_string`

### 6.21.4.2 `database_name`

`database_name`

### 6.21.4.3 `db_type`

`db_type`

### 6.21.4.4 `verbose`

`verbose`

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/components/[connectors.py](#)

## 6.22 RelationExtractor Class Reference

### Public Member Functions

- `__init__` (self, model\_name="Babelscape/rebel-large", `max_tokens`=1024)
- `extract` (self, str text, bool parse\_tuples=False)

## Public Attributes

- `tokenizer`
- `model`
- `max_tokens`
- `tuple_delim`

## 6.22.1 Constructor & Destructor Documentation

### 6.22.1.1 `__init__()`

```
__init__ (
    self,
    model_name = "Babelscape/rebel-large",
    max_tokens = 1024 )
```

## 6.22.2 Member Function Documentation

### 6.22.2.1 `extract()`

```
extract (
    self,
    str text,
    bool parse_tuples = False )
```

## 6.22.3 Member Data Documentation

### 6.22.3.1 `max_tokens`

`max_tokens`

### 6.22.3.2 `model`

`model`

### 6.22.3.3 `tokenizer`

`tokenizer`

### 6.22.3.4 `tuple_delim`

`tuple_delim`

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/components/[text\\_processing.py](#)

## 6.23 ScalarMetric Class Reference

### Properties

- string `Name` [get, set]
- double `Value` [get, set]

#### 6.23.1 Property Documentation

##### 6.23.1.1 Name

`string Name [get], [set]`

##### 6.23.1.2 Value

`double Value [get], [set]`

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/[ScalarMetric.cs](#)

## 6.24 Session Class Reference

Stores active database connections and configuration settings.

### Public Member Functions

- `__new__(cls, *args, **kwargs)`

*Creates a new session at first access, otherwise uses the existing session.*
- `__init__(self, verbose=False)`

*Initializes the session using the .env file.*
- `test_database_connections(self)`

*Configure the databases and verify they are working correctly.*
- `reset(self)`

*Deletes all created databases and tables.*

### Public Attributes

- `verbose`

*Initializes the session using the .env file.*
- `relational_db`

*Stores RDF-compliant semantic triples.*
- `docs_db`

*Stores input text, pre-processed chunks, JSON intermediates, and final output.*
- `graph_db`

*Main storage for entities (nodes) and relations (edges).*

## Static Protected Attributes

- `_instance` = None

*Creates a new session at first access, otherwise uses the existing session.*

### 6.24.1 Detailed Description

Stores active database connections and configuration settings.

- This class implements Singleton design, so only one session can be created.
- However, the session config can still be updated using the normal constructor.

### 6.24.2 Constructor & Destructor Documentation

#### 6.24.2.1 `__init__()`

```
__init__ (
    self,
    verbose = False )
```

Initializes the session using the .env file.

- The relational database connector is created using a Factory Method, choosing mysql or postgres based on the .env file.
- The document database connector is created normally since mongo is the only supported option.
- The graph database connector is created normally since neo4j is the only supported option.

### 6.24.3 Member Function Documentation

#### 6.24.3.1 `__new__()`

```
__new__ (
    cls,
    * args,
    ** kwargs )
```

Creates a new session at first access, otherwise uses the existing session.

#### 6.24.3.2 `reset()`

```
reset (
    self )
```

Deletes all created databases and tables.

### 6.24.3.3 test\_database\_connections()

```
test_database_connections (
    self )
```

Configure the databases and verify they are working correctly.

## 6.24.4 Member Data Documentation

### 6.24.4.1 \_instance

```
_instance = None [static], [protected]
```

Creates a new session at first access, otherwise uses the existing session.

### 6.24.4.2 docs\_db

```
docs_db
```

Stores input text, pre-processed chunks, JSON intermediates, and final output.

### 6.24.4.3 graph\_db

```
graph_db
```

Main storage for entities (nodes) and relations (edges).

### 6.24.4.4 relational\_db

```
relational_db
```

Stores RDF-compliant semantic triples.

### 6.24.4.5 verbose

```
verbose
```

Initializes the session using the .env file.

- The relational database connector is created using a Factory Method, choosing mysql or postgres based on the .env file.
  - The document database connector is created normally since mongo is the only supported option.
  - The graph database connector is created normally since neo4j is the only supported option.

Enables or disables the components from printing debug info.

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/src/[setup.py](#)

## 6.25 Story Class Reference

### Public Member Functions

- `_init_(self, StoryStreamAdapter reader)`
- `Chunk stream_chunks (self)`
- `pre_split_chunks (self, int max_chunk_length)`

*Splits paragraphs into chunks.*

### Public Attributes

- `reader`

### Protected Member Functions

- `_merge_chunks (self, segs, max_len)`
- `_make_single (self, seg, text, max_len, start=None)`

#### 6.25.1 Constructor & Destructor Documentation

##### 6.25.1.1 `_init_()`

```
_init_ (
    self,
    StoryStreamAdapter reader )
```

#### 6.25.2 Member Function Documentation

##### 6.25.2.1 `_make_single()`

```
_make_single (
    self,
    seg,
    text,
    max_len,
    start = None ) [protected]
```

##### 6.25.2.2 `_merge_chunks()`

```
_merge_chunks (
    self,
    segs,
    max_len ) [protected]
```

### 6.25.2.3 pre\_split\_chunks()

```
pre_split_chunks (
    self,
    int max_chunk_length )
```

Splits paragraphs into chunks.

- Populates self.chunks with Chunk objects that obey max\_chunk\_length.
- Combines adjacent paragraphs when possible.
- Falls back to splitting by sentences if one paragraph is too long.

### 6.25.2.4 stream\_chunks()

```
Chunk stream_chunks (
    self )
```

## 6.25.3 Member Data Documentation

### 6.25.3.1 reader

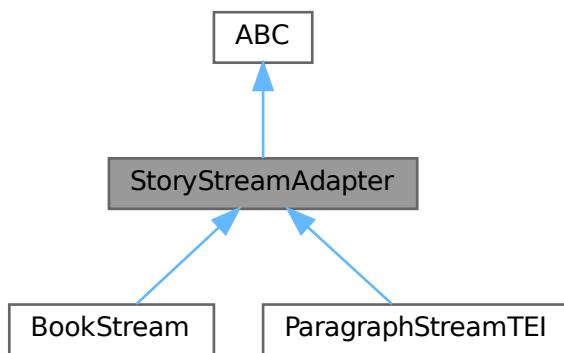
```
reader
```

The documentation for this class was generated from the following file:

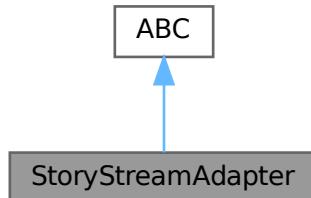
- /home/runner/work/dsci-capstone/dsci-capstone/components/book\_conversion.py

## 6.26 StoryStreamAdapter Class Reference

Inheritance diagram for StoryStreamAdapter:



Collaboration diagram for StoryStreamAdapter:



## Public Member Functions

- [Chunk stream\\_segments \(self\)](#)  
*Yields sanitized parts of a book.*
- [Chunk stream\\_paragraphs \(self\)](#)  
*Concrete helper method to split segments into paragraphs.*
- str [stream\\_sentences \(self\)](#)  
*Concrete helper method to split paragraphs into sentences.*

### 6.26.1 Member Function Documentation

#### 6.26.1.1 [stream\\_paragraphs\(\)](#)

```
Chunk stream_paragraphs (
    self )
```

Concrete helper method to split segments into paragraphs.

The Chunk class is repurposed here so we pass location info. Depending on the Story.pre\_split\_chunks implementation, this might be unnecessary.

#### 6.26.1.2 [stream\\_segments\(\)](#)

```
Chunk stream_segments (
    self )
```

Yields sanitized parts of a book.

- Story segments usually correspond to chapters.
- They serve as borders between chunking operations, ensuring chunks do not span multiple chapters. Implementation is handled by child classes BookStream, etc.
- Segments should be pre-cleaned and must contain 1 paragraph per line with all other newlines removed.

Reimplemented in [ParagraphStreamTEI](#), and [BookStream](#).

### 6.26.1.3 stream\_sentences()

```
str stream_sentences (
    self )
```

Concrete helper method to split paragraphs into sentences.

Mostly for debugging.

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/components/[book\\_conversion.py](#)

## 6.27 SummaryData Class Reference

### Properties

- string [BookID](#) [get, set]
- string [BookTitle](#) [get, set]
- string [SummaryText](#) [get, set]
- [SummaryMetrics Metrics](#) = new() [get, set]
- List<[QAMetric](#)> [QAResults](#) = new() [get, set]

### 6.27.1 Property Documentation

#### 6.27.1.1 BookID

```
string BookID [get], [set]
```

#### 6.27.1.2 BookTitle

```
string BookTitle [get], [set]
```

#### 6.27.1.3 Metrics

```
SummaryMetrics Metrics = new() [get], [set]
```

#### 6.27.1.4 QAResults

```
List<QAMetric> QAResults = new() [get], [set]
```

### 6.27.1.5 SummaryText

```
string SummaryText [get], [set]
```

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/SummaryData.cs

## 6.28 SummaryMetrics Class Reference

### Static Public Member Functions

- static [SummaryMetrics GetDefault \(\)](#)

### Properties

- List< [PRF1Metric](#) > [PRF1Metrics](#) = new() [get, set]
- [QAMetric QA](#) = new() [get, set]
- List< [ScalarMetric](#) > [ScalarMetrics](#) = new() [get, set]

### 6.28.1 Member Function Documentation

#### 6.28.1.1 GetDefault()

```
static SummaryMetrics GetDefault ( ) [static]
```

### 6.28.2 Property Documentation

#### 6.28.2.1 PRF1Metrics

```
List<PRF1Metric> PRF1Metrics = new() [get], [set]
```

#### 6.28.2.2 QA

```
QAMetric QA = new() [get], [set]
```

#### 6.28.2.3 ScalarMetrics

```
List<ScalarMetric> ScalarMetrics = new() [get], [set]
```

The documentation for this class was generated from the following file:

- /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/SummaryMetrics.cs



# Chapter 7

## File Documentation

### 7.1 /home/runner/work/dsci-capstone/dsci-capstone/components/book←\_conversion.py File Reference

#### Classes

- class [Chunk](#)  
*Lightweight container for a span of story text.*
- class [StoryStreamAdapter](#)
- class [Story](#)
- class [ParagraphStreamTEI](#)  
*Streams paragraphs from a TEI file as Chunk objects.*
- class [Book](#)
- class [BookStream](#)
- class [BookFactory](#)
- class [EPUBToTEI](#)  
*Converts EPUB files to XML format (TEI specification).*

#### Namespaces

- namespace [components](#)
- namespace [components.book\\_conversion](#)

#### Variables

- [nlp](#) = spacy.blank("en")
- [sentencizer](#) = nlp.add\_pipe("sentencizer")

## 7.2 /home/runner/work/dsci-capstone/dsci-capstone/components/connectors.py File Reference

### Classes

- class [Connector](#)  
*Abstract base class for external connectors.*
- class [DatabaseConnector](#)  
*Abstract base class for database engine connectors.*
- class [RelationalConnector](#)  
*Connector for relational databases (MySQL, PostgreSQL).*
- class [mysqlConnector](#)  
*A relational database connector configured for MySQL.*
- class [postgresConnector](#)  
*A relational database connector configured for PostgreSQL.*

### Namespaces

- namespace [components](#)
- namespace [components.connectors](#)

## 7.3 /home/runner/work/dsci-capstone/dsci-capstone/components/document\_storage.py File Reference

### Classes

- class [DocumentConnector](#)  
*Connector for MongoDB (document database)*

### Namespaces

- namespace [components](#)
- namespace [components.document\\_storage](#)

### Functions

- Generator[Database[Any], None, None] [mongo\\_handle](#) (str host, str alias)  
*Establish a temporary connection to MongoDB.*
- DataFrame [\\_flatten\\_recursive](#) (DataFrame df)  
*Explode all list columns and flatten dict columns until only scalars remain.*
- str [\\_sanitize\\_json](#) (str text)  
*Remove comments and other non-JSON content from a MongoDB query string.*
- Dict[str, Any] [\\_sanitize\\_document](#) (Dict[str, Any] doc, Dict[str, Set[Type[Any]]] type\_registry)  
*Normalize document fields to consistent types for DataFrame construction.*
- DataFrame [\\_docs\\_to\\_df](#) (List[Dict[str, Any]] docs, bool merge\_unspecified=True)  
*Convert raw MongoDB documents to a Pandas DataFrame.*
- str [\\_find\\_compatible\\_nested\\_key](#) (Type[Any] value\_type, Dict[str, Set[Type[Any]]] nested\_schema, bool merge\_unspecified)  
*Find a nested column compatible with the given primitive type.*

## 7.4 /home/runner/work/dsci-capstone/dsci-capstone/components/fact\_← storage.py File Reference

### Classes

- class [GraphConnector](#)  
*Connector for Neo4j (graph database).*

### Namespaces

- namespace [components](#)
- namespace [components.fact\\_storage](#)

## 7.5 /home/runner/work/dsci-capstone/dsci- capstone/components/metrics.py File Reference

### Namespaces

- namespace [components](#)
- namespace [components.metrics](#)

### Functions

- [generate\\_default\\_metrics](#) (`rouge_precision=0.0, rouge_recall=0.0, rouge_f1=0.0, bert_precision=0.0, bert_recall=0.0, bert_f1=0.0, boook_score=0.0, questeval_score=0.0, qa_question1="UNKNOWN", qa_gold1="UNKNOWN", qa_generated1="UNKNOWN", qa_correct1=False, qa_accuracy1=0.0, qa_question2="UNKNOWN", qa_gold2="UNKNOWN", qa_generated2="UNKNOWN", qa_correct2=False, qa_accuracy2=0.0`)  
*Generate metrics payload with customizable default values.*
- [create\\_summary\\_payload](#) (`book_id, book_title, summary, metrics=None`)  
*Create the full summary payload for the API.*
- [post\\_payload](#) (`payload`)  
*Verify and post any given payload using the requests API.*
- [post\\_example\\_results](#) ()  
*Send placeholder values to the web app.*
- [post\\_basic\\_output](#) (`book_id, book_title, summary, **kwargs`)  
*Send book information and a summary to the web app.*

### Variables

- `HOST = os.getenv("BLAZOR_HOST")`
- `str url = f"http://{{HOST}}:5055/api/metrics"`

- 7.6 /home/runner/work/dsci-capstone/dsci-capstone/components/semantic\_web.py File Reference**
- 7.7 /home/runner/work/dsci-capstone/dsci-capstone/components/text\_processing.py File Reference**
- Classes**
- class [RelationExtractor](#)
  - class [LLMConnector](#)
- Connector for prompting and returning LLM output (raw text/JSON) via LangChain.*
- Namespaces**
- namespace [components](#)
  - namespace [components.text\\_processing](#)
- Variables**
- [nlp](#) = spacy.blank("en")
  - [sentencizer](#) = nlp.add\_pipe("sentencizer")
- 7.8 /home/runner/work/dsci-capstone/dsci-capstone/components/\_\_init\_\_.py File Reference**
- Namespaces**
- namespace [components](#)
- 7.9 /home/runner/work/dsci-capstone/dsci-capstone/src/\_\_init\_\_.py File Reference**
- Namespaces**
- namespace [src](#)
- 7.10 /home/runner/work/dsci-capstone/dsci-capstone/tests/\_\_init\_\_.py File Reference**
- Namespaces**
- namespace [tests](#)

## 7.11 /home/runner/work/dsci-capstone/dsci-capstone/src/main.py File Reference

### Namespaces

- namespace `src`
- namespace `src.main`

### Functions

- `convert_single ()`  
*Converts one EPUB file to TEI format.*
- `convert_from_csv ()`  
*Converts several EPUB files to TEI format.*
- `chunk_single ()`  
*Creates a Story and many Chunks from a TEI file.*
- `test_relation_extraction ()`  
*Runs REBEL on a basic example; used for debugging.*
- `process_single ()`  
*Uses NLP and LLM to process an existing TEI file.*
- `graph_triple_files ()`  
*Loads JSON into Neo4j to test the Blazor graph page.*
- `output_single ()`  
*Generates a summary from triples stored in JSON, and posts data to Blazor.*
- `full_pipeline (epub_path, book_chapters, start_str, end_str, book_id, story_id, book_title)`  
*Connects all components to convert an EPUB file to a book summary.*

### Variables

- `session = Session(verbose=False)`
- `str tei = "./datasets/examples/trilogy-wishes-1.tei"`  
*Will revisit later - Book classes need refactoring ###.*
- `str chapters`
- `str start = ""`
- `str end = "But I must say no more."`
- `list triple_files`
- `list response_files = ["./datasets/triples/chunk-160_story-1.txt"]`
- `epub_path`
- `book_chapters`
- `start_str`
- `end_str`
- `book_id`
- `story_id`
- `book_title`

## 7.12 /home/runner/work/dsci-capstone/dsci-capstone/src/setup.py File Reference

### Classes

- class `Session`  
*Stores active database connections and configuration settings.*

## Namespaces

- namespace [src](#)
- namespace [src.setup](#)

## Variables

- `session = Session()`

# 7.13 /home/runner/work/dsci-capstone/dsci-capstone/src/util.py File Reference

## Classes

- class [Log](#)  
*The Log class standardizes console output.*
- class [Log.Failure](#)

## Namespaces

- namespace [src](#)
- namespace [src.util](#)

## Functions

- `all_none (*args)`  
*Checks if all provided args are None.*
- DataFrame `df_natural_sorted` (DataFrame df, List[str] ignored\_columns=[ ])  
*Sort a DataFrame in natural order using only certain columns.*
- bool `check_values` (List[Any] results, List[Any] expected, bool verbose, str log\_source, bool raise\_error)  
*Safely compare two lists of values.*

# 7.14 /home/runner/work/dsci-capstone/dsci-capstone/tests/conftest.py File Reference

## Namespaces

- namespace [tests](#)
- namespace [tests.conftest](#)

## Functions

- `pytest_adoption` (parser)
- `session` (request)  
*Fixture to create session.*

## 7.15 /home/runner/work/dsci-capstone/dsci-capstone/tests/test\_components.py File Reference

### Namespaces

- namespace [tests](#)
- namespace [tests.test\\_components](#)

### Functions

- [relational\\_db](#) (session)  
*Fixture to get relational database connection.*
- [docs\\_db](#) (session)  
*Fixture to get document database connection.*
- [graph\\_db](#) (session)  
*Fixture to get document database connection.*
- [test\\_db\\_relational\\_minimal](#) ([relational\\_db](#))  
*Tests if the RelationalConnector has a valid connection string.*
- [test\\_db\\_docs\\_minimal](#) ([docs\\_db](#))  
*Tests if the DocumentConnector has a valid connection string.*
- [test\\_db\\_graph\\_minimal](#) ([graph\\_db](#))  
*Tests if the GraphConnector has a valid connection string.*
- [test\\_db\\_relational\\_comprehensive](#) ([relational\\_db](#))  
*Tests if the GraphConnector is working as intended.*
- [test\\_db\\_docs\\_comprehensive](#) ([docs\\_db](#))  
*Tests if the GraphConnector is working as intended.*
- [test\\_db\\_graph\\_comprehensive](#) ([graph\\_db](#))  
*Tests if the GraphConnector is working as intended.*
- [load\\_examples\\_relational](#) ([relational\\_db](#))  
*Fixture to create relational tables using engine-specific syntax.*
- [test\\_sql\\_example\\_1](#) ([relational\\_db](#), [load\\_examples\\_relational](#))  
*Run queries contained within test files.*
- [test\\_sql\\_example\\_2](#) ([relational\\_db](#), [load\\_examples\\_relational](#))  
*Run queries contained within test files.*
- [test\\_mongo\\_example\\_1](#) ([docs\\_db](#))  
*Run queries contained within test files.*
- [test\\_mongo\\_example\\_2](#) ([docs\\_db](#))  
*Run queries contained within test files.*
- [test\\_mongo\\_example\\_3](#) ([docs\\_db](#))  
*Run queries contained within test files.*
- [\\_test\\_query\\_file](#) (db\_fixture, str filename, List valid\_files)  
*Run queries from a local file through the database.*

7.16 /home/runner/work/dsci-capstone/dsci-capstone/web-app/Blazor ←  
App/Components/\_Imports.razor File Reference

7.17 /home/runner/work/dsci-capstone/dsci-capstone/web-app/Blazor ←  
App/Components/App.razor File Reference

7.18 /home/runner/work/dsci-capstone/dsci-capstone/web-app/Blazor ←  
App/Components/Layout/MainLayout.razor File Reference

7.19 /home/runner/work/dsci-capstone/dsci-capstone/web-app/Blazor ←  
App/Components/Layout/NavMenu.razor File Reference

7.20 /home/runner/work/dsci-capstone/dsci-capstone/web-app/Blazor ←  
App/Components/Pages/Error.razor File Reference

7.21 /home/runner/work/dsci-capstone/dsci-capstone/web-app/Blazor ←  
App/Components/Pages/Graph.razor File Reference

7.22 /home/runner/work/dsci-capstone/dsci-capstone/web-app/Blazor ←  
App/Components/Pages/Home.razor File Reference

7.23 /home/runner/work/dsci-capstone/dsci-capstone/web-app/Blazor ←  
App/Components/Pages/Metrics.razor File Reference

7.24 /home/runner/work/dsci-capstone/dsci-capstone/web-app/Blazor ←  
App/Components/Routes.razor File Reference

7.25 /home/runner/work/dsci-capstone/dsci-capstone/web-app/Blazor ←  
App/Controllers/MetricsController.cs File Reference

## Classes

- class [MetricsController](#)

## Namespaces

- namespace [BlazorApp](#)
- namespace [BlazorApp.Controllers](#)

## 7.26 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Hubs/MetricsHub.cs File Reference

### Classes

- class [MetricsHub](#)

### Namespaces

- namespace [BlazorApp](#)
- namespace [BlazorApp.Hubs](#)

## 7.27 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/PRF1Metric.cs File Reference

### Classes

- class [PRF1Metric](#)

### Namespaces

- namespace [BlazorApp](#)
- namespace [BlazorApp.Models](#)

## 7.28 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/QAItem.cs File Reference

### Classes

- class [QAItem](#)

### Namespaces

- namespace [BlazorApp](#)
- namespace [BlazorApp.Models](#)

## 7.29 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/QAMetric.cs File Reference

### Classes

- class [QAMetric](#)

## Namespaces

- namespace [BlazorApp](#)
- namespace [BlazorApp.Models](#)

## 7.30 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/ScalarMetric.cs File Reference

### Classes

- class [ScalarMetric](#)

## Namespaces

- namespace [BlazorApp](#)
- namespace [BlazorApp.Models](#)

## 7.31 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/SummaryData.cs File Reference

### Classes

- class [SummaryData](#)

## Namespaces

- namespace [BlazorApp](#)
- namespace [BlazorApp.Models](#)

## 7.32 /home/runner/work/dsci-capstone/dsci-capstone/web-app/BlazorApp/Models/SummaryMetrics.cs File Reference

### Classes

- class [SummaryMetrics](#)

## Namespaces

- namespace [BlazorApp](#)
- namespace [BlazorApp.Models](#)

# Index

/home/runner/work/dsci-capstone/dsci-capstone/components/\_init\_.py, 126 /home/runner/work/dsci-capstone/dsci-capstone/web-  
/home/runner/work/dsci-capstone/dsci-capstone/components/book\_capstone/app/Components/BookApp/Components/Routes.razor, 123 130  
/home/runner/work/dsci-capstone/dsci-capstone/components/docu\_100\_storage.py, 124 /home/runner/work/dsci-capstone/dsci-capstone/web-  
/home/runner/work/dsci-capstone/dsci-capstone/components/fact\_storag\_100\_app/BlazorApp/Controllers/MetricsController.cs, 125 130  
/home/runner/work/dsci-capstone/dsci-capstone/components/metric\_100\_hub.py, 126 /home/runner/work/dsci-capstone/dsci-capstone/web-  
app/BlazorApp/Hubs/MetricsHub.cs, 131  
/home/runner/work/dsci-capstone/dsci-capstone/components/prf1metric\_100\_webapp/BlazorApp/Models/PRF1Metric.cs, 131  
/home/runner/work/dsci-capstone/dsci-capstone/components/qaltem\_100\_processing.py, 126 /home/runner/work/dsci-capstone/dsci-capstone/web-  
app/BlazorApp/Models/QAItem.cs, 131  
/home/runner/work/dsci-capstone/dsci-capstone/src/\_init\_.py, 126 /home/runner/work/dsci-capstone/dsci-capstone/web-  
app/BlazorApp/Models/QAMetric.cs, 131  
/home/runner/work/dsci-capstone/dsci-capstone/src/main.py, 127 /home/runner/work/dsci-capstone/dsci-capstone/web-  
app/BlazorApp/Models/ScalarMetric.cs, 132  
/home/runner/work/dsci-capstone/dsci-capstone/src/setup.py, 127 /home/runner/work/dsci-capstone/dsci-capstone/web-  
app/BlazorApp/Models/SummaryData.cs, 132  
/home/runner/work/dsci-capstone/dsci-capstone/src/util.py, 128 /home/runner/work/dsci-capstone/dsci-capstone/web-  
app/BlazorApp/Models/SummaryMetrics.cs, 132  
/home/runner/work/dsci-capstone/dsci-capstone/tests/\_init\_\_.py, 126 \_\_init\_\_  
/home/runner/work/dsci-capstone/dsci-capstone/tests/conftest.py, 128 Book, 29  
BookStream, 32  
/home/runner/work/dsci-capstone/dsci-capstone/tests/test\_components.py, 129 Components  
DatabaseConnector, 41  
DocumentConnector, 50  
EPUBToTEI, 56  
GraphConnector, 63  
LLMConnector, 73  
Log.Failure, 59  
mysqlConnector, 94  
ParagraphStreamTEI, 96  
postgresConnector, 102  
RelationalConnector, 107  
RelationExtractor, 113  
Session, 115  
Story, 117  
Session, 115  
Session, 130 \_\_new\_\_  
Session, 130 \_\_repr\_\_  
Chunk, 35  
Session, 130 \_\_str\_\_  
Log.Failure, 59  
auth suffix

DocumentConnector, 55  
`_created_dummy`  
     GraphConnector, 71  
`_docs_to_df`  
     components.document\_storage, 11  
`_find_compatible_nested_key`  
     components.document\_storage, 12  
`_flatten_recursive`  
     components.document\_storage, 12  
`_hubContext`  
     MetricsController, 89  
`_instance`  
     Session, 116  
`_is_single_query`  
     DatabaseConnector, 41  
`_logger`  
     MetricsController, 89  
     MetricsHub, 91  
`_make_single`  
     Story, 117  
`_merge_chunks`  
     Story, 117  
`_prune_bad_tags`  
     EPUBToTEI, 56  
`_sanitize_document`  
     components.document\_storage, 13  
`_sanitize_ids`  
     EPUBToTEI, 56  
`_sanitize_json`  
     components.document\_storage, 13  
`_split_combined`  
     DatabaseConnector, 42  
     DocumentConnector, 50  
     GraphConnector, 63  
     RelationalConnector, 108  
`_test_query_file`  
     tests.test\_components, 24

**Accuracy**  
     QALitem, 103

**add\_triple**  
     GraphConnector, 64

**all\_none**  
     src.util, 22

**allowed\_chapters**  
     ParagraphStreamTEI, 97

**author\_key**  
     Book, 29

**AverageAccuracy**  
     QAMetric, 104

**bad\_addr**  
     Log, 80

**bad\_path**  
     Log, 80

**bad\_val**  
     Log, 80

**BlazorApp**, 9

**BlazorApp.Controllers**, 9

BlazorApp.Hubs, 9  
 BlazorApp.Models, 9

**Book**, 29  
     `__init__`, 29  
     author\_key, 29  
     date\_key, 29  
     language\_key, 30  
     title\_key, 30  
**book**  
     BookStream, 33

**book\_chapters**  
     src.main, 20

**book\_id**  
     ParagraphStreamTEI, 97  
     src.main, 20

**book\_title**  
     src.main, 20

**BookFactory**, 30  
     create\_book, 31

**BookID**  
     SummaryData, 120

**BookStream**, 31  
     `__init__`, 32  
     book, 33  
     stream\_segments, 33

**BookTitle**  
     SummaryData, 120

**BRIGHT**  
     Log, 80

**change\_database**  
     DatabaseConnector, 42  
     DocumentConnector, 51  
     GraphConnector, 64  
     RelationalConnector, 108

**change\_graph**  
     GraphConnector, 65

**chapters**  
     src.main, 20

**char\_count**  
     Chunk, 35

**check\_connection**  
     DocumentConnector, 51  
     GraphConnector, 65  
     RelationalConnector, 108

**check\_values**  
     src.util, 22

**Chunk**, 33  
     `__init__`, 34  
     `__repr__`, 35  
     char\_count, 35  
     text, 35

**chunk\_single**  
     src.main, 18

**chunks**  
     ParagraphStreamTEI, 97

**clean\_tei**  
     EPUBToTEI, 57

**clean\_tei\_content**

EPUBToTEI, 57  
components, 10  
components.book\_conversion, 10  
  nlp, 10  
  sentencizer, 10  
components.connectors, 11  
components.document\_storage, 11  
  \_docs\_to\_df, 11  
  \_find\_compatible\_nested\_key, 12  
  \_flatten\_recursive, 12  
  \_sanitize\_document, 13  
  \_sanitize\_json, 13  
  mongo\_handle, 14  
components.fact\_storage, 14  
components.metrics, 14  
  create\_summary\_payload, 15  
  generate\_default\_metrics, 15  
  HOST, 16  
  post\_basic\_output, 15  
  post\_example\_results, 16  
  post\_payload, 16  
  url, 16  
components.text\_processing, 16  
  nlp, 17  
  sentencizer, 17  
configure  
  Connector, 37  
  DatabaseConnector, 42  
  LLMConnector, 73  
conn\_abc  
  Log, 80  
connection\_string  
  DatabaseConnector, 47  
  DocumentConnector, 55  
  GraphConnector, 71  
  RelationalConnector, 112  
Connector, 35  
  configure, 37  
  execute\_file, 37  
  execute\_query, 38  
  test\_connection, 38  
convert\_from\_csv  
  src.main, 18  
convert\_single  
  src.main, 18  
convert\_to\_tei  
  EPUBToTEI, 57  
create\_book  
  BookFactory, 31  
create\_database  
  DatabaseConnector, 43  
  DocumentConnector, 52  
  GraphConnector, 65  
  RelationalConnector, 109  
create\_db  
  Log, 80  
create\_summary\_payload  
  components.metrics, 15  
database\_exists  
  DatabaseConnector, 43  
  DocumentConnector, 52  
  GraphConnector, 66  
  RelationalConnector, 109  
database\_name  
  DocumentConnector, 55  
  GraphConnector, 71  
  RelationalConnector, 112  
DatabaseConnector, 39  
  \_\_init\_\_, 41  
  \_is\_single\_query, 41  
  \_split\_combined, 42  
  change\_database, 42  
  configure, 42  
  connection\_string, 47  
  create\_database, 43  
  database\_exists, 43  
  db\_engine, 47  
  db\_type, 47  
  drop\_database, 43  
  execute\_combined, 45  
  execute\_file, 45  
  execute\_query, 46  
  get\_dataframe, 46  
  host, 47  
  password, 47  
  port, 47  
  username, 47  
  verbose, 47  
date\_key  
  Book, 29  
db\_conn\_abc  
  Log, 81  
db\_engine  
  DatabaseConnector, 47  
db\_exists  
  Log, 81  
db\_type  
  DatabaseConnector, 47  
  RelationalConnector, 112  
delete\_dummy  
  DocumentConnector, 52  
  GraphConnector, 66  
df\_natural\_sorted  
  src.util, 23  
doc\_db  
  Log, 81  
docs\_db  
  Session, 116  
  tests.test\_components, 25  
DocumentConnector, 48  
  \_\_init\_\_, 50  
  \_auth\_suffix, 55  
  \_split\_combined, 50  
  change\_database, 51  
  check\_connection, 51  
  connection\_string, 55

create\_database, 52  
 database\_exists, 52  
 database\_name, 55  
 delete\_dummy, 52  
 drop\_database, 53  
 execute\_query, 53  
 get\_dataframe, 53  
 test\_connection, 54  
 verbose, 55  
**drop\_database**  
     DatabaseConnector, 43  
     DocumentConnector, 53  
     GraphConnector, 66  
     RelationalConnector, 110  
**drop\_db**  
     Log, 81  
**encoding**  
     EPUBToTEI, 57  
     ParagraphStreamTEI, 97  
**end**  
     src.main, 20  
**end\_inclusive**  
     ParagraphStreamTEI, 98  
**end\_str**  
     src.main, 20  
**epub\_path**  
     EPUBToTEI, 57  
     src.main, 20  
**EPUBToTEI**, 55  
     \_\_init\_\_, 56  
     \_prune\_bad\_tags, 56  
     \_sanitize\_ids, 56  
     clean\_tei, 57  
     clean\_tei\_content, 57  
     convert\_to\_teい, 57  
     encoding, 57  
     epub\_path, 57  
     pandoc\_xml\_path, 57  
     raw\_teи\_content, 58  
     save\_pandoc, 58  
     save\_teи, 58  
     teи\_path, 58  
     xml\_namespace, 58  
**execute\_combined**  
     DatabaseConnector, 45  
**execute\_file**  
     Connector, 37  
     DatabaseConnector, 45  
     LLMConnector, 73  
**execute\_full\_query**  
     LLMConnector, 74  
**execute\_query**  
     Connector, 38  
     DatabaseConnector, 46  
     DocumentConnector, 53  
     GraphConnector, 67  
     LLMConnector, 74  
     RelationalConnector, 110  
**extract**  
     RelationExtractor, 113  
**F1Score**  
     PRF1Metric, 103  
**fail**  
     Log, 78  
**fail\_legacy**  
     Log, 79  
**FAILURE\_COLOR**  
     Log, 81  
**from\_env**  
     RelationalConnector, 111  
**FULL\_DF**  
     Log, 81  
**full\_pipeline**  
     src.main, 18  
**generate\_default\_metrics**  
     components.metrics, 15  
**GeneratedAnswer**  
     QAItem, 103  
**get\_all\_triples**  
     GraphConnector, 67  
**get\_dataframe**  
     DatabaseConnector, 46  
     DocumentConnector, 53  
     GraphConnector, 68  
     RelationalConnector, 111  
**get\_df**  
     Log, 81  
**get\_edge\_counts**  
     GraphConnector, 68  
**get\_unique**  
     GraphConnector, 69  
     Log, 81  
**GetAll**  
     MetricsController, 89  
**GetDefault**  
     SummaryMetrics, 121  
**GetIndex**  
     MetricsController, 89  
**GoldAnswer**  
     QAItem, 103  
**good\_val**  
     Log, 81  
**gr\_db**  
     Log, 82  
**graph\_db**  
     Session, 116  
     tests.test\_components, 25  
**graph\_name**  
     GraphConnector, 71  
**graph\_triple\_files**  
     src.main, 19  
**GraphConnector**, 60  
     \_\_init\_\_, 63  
     \_created\_dummy, 71  
     \_split\_combined, 63

add\_triple, 64  
change\_database, 64  
change\_graph, 65  
check\_connection, 65  
connection\_string, 71  
create\_database, 65  
database\_exists, 66  
database\_name, 71  
delete\_dummy, 66  
drop\_database, 66  
execute\_query, 67  
get\_all\_triples, 67  
get\_dataframe, 68  
get\_edge\_counts, 68  
get\_unique, 69  
graph\_name, 71  
IS\_DUMMY\_, 69  
NOT\_DUMMY\_, 69  
print\_nodes, 70  
print\_triples, 70  
SAME\_DB\_KG\_, 70  
test\_connection, 70  
verbose, 71  
GREEN  
  Log, 82  
HOST  
  components.metrics, 16  
host  
  DatabaseConnector, 47  
IS\_DUMMY\_  
  GraphConnector, 69  
IsCorrect  
  QAltem, 104  
kg  
  Log, 82  
language\_key  
  Book, 30  
lines  
  ParagraphStreamTEI, 98  
llm  
  LLMConnector, 75  
LLMConnector, 72  
  \_\_init\_\_, 73  
  configure, 73  
  execute\_file, 73  
  execute\_full\_query, 74  
  execute\_query, 74  
  llm, 75  
  model\_name, 75  
  system\_prompt, 75  
  temperature, 75  
  test\_connection, 74  
load\_examples\_relational  
  tests.test\_components, 25  
Log, 75  
bad\_addr, 80  
bad\_path, 80  
bad\_val, 80  
BRIGHT, 80  
conn\_abc, 80  
create\_db, 80  
db\_conn\_abc, 81  
db\_exists, 81  
doc\_db, 81  
drop\_db, 81  
fail, 78  
fail\_legacy, 79  
FAILURE\_COLOR, 81  
FULL\_DF, 81  
get\_df, 81  
get\_unique, 81  
good\_val, 81  
gr\_db, 82  
GREEN, 82  
kg, 82  
msg\_bad\_addr, 82  
msg\_bad\_coll, 82  
msg\_bad\_exec\_f, 82  
msg\_bad\_exec\_q, 82  
msg\_bad\_graph, 82  
msg\_bad\_path, 82  
msg\_bad\_table, 82  
MSG\_COLOR, 83  
msg\_compare, 83  
msg\_db\_connect, 83  
msg\_db\_current, 83  
msg\_db\_exists, 83  
msg\_db\_not\_found, 83  
msg\_fail\_manage\_db, 83  
msg\_fail\_parse, 83  
msg\_good\_coll, 84  
msg\_good\_exec\_f, 84  
msg\_good\_exec\_q, 84  
msg\_good\_exec\_qr, 84  
msg\_good\_graph, 84  
msg\_good\_path, 84  
msg\_good\_table, 84  
msg\_multiple\_query, 84  
msg\_result, 85  
msg\_success\_managed\_db, 85  
msg\_swap\_db, 85  
msg\_swap\_kg, 85  
msg\_unknown\_error, 85  
pytest\_db, 85  
RED, 85  
rel\_db, 85  
run\_f, 86  
run\_q, 86  
success, 79  
SUCCESS\_COLOR, 86  
success\_legacy, 79  
swap\_db, 86  
swap\_kg, 86

test\_basic, 86  
 test\_conn, 86  
 test\_df, 86  
 test\_info, 86  
 test\_tmp\_db, 86  
 USE\_COLORS, 87  
 warn, 80  
 WARNING\_COLOR, 87  
 WHITE, 87  
 YELLOW, 87  
 Log.Failure, 58  
 \_\_init\_\_, 59  
 \_\_str\_\_, 59  
 msg, 59  
 prefix, 59  
 max\_tokens  
     RelationExtractor, 113  
 Metrics  
     SummaryData, 120  
 MetricsController, 88  
     \_hubContext, 89  
     \_logger, 89  
     GetAll, 89  
     GetIndex, 89  
     MetricsController, 89  
     Post, 89  
     Summaries, 89  
 MetricsHub, 90  
     \_logger, 91  
     MetricsHub, 90  
     OnConnectedAsync, 91  
     OnDisconnectedAsync, 91  
 model  
     RelationExtractor, 113  
 model\_name  
     LLMConnector, 75  
 mongo\_handle  
     components.document\_storage, 14  
 msg  
     Log.Failure, 59  
 msg\_bad\_addr  
     Log, 82  
 msg\_bad\_coll  
     Log, 82  
 msg\_bad\_exec\_f  
     Log, 82  
 msg\_bad\_exec\_q  
     Log, 82  
 msg\_bad\_graph  
     Log, 82  
 msg\_bad\_path  
     Log, 82  
 msg\_bad\_table  
     Log, 82  
 MSG\_COLOR  
     Log, 83  
 msg\_compare  
     Log, 83  
 msg\_db\_connect  
     Log, 83  
 msg\_db\_current  
     Log, 83  
 msg\_db\_exists  
     Log, 83  
 msg\_db\_not\_found  
     Log, 83  
 msg\_fail\_manage\_db  
     Log, 83  
 msg\_fail\_parse  
     Log, 83  
 msg\_good\_coll  
     Log, 84  
 msg\_good\_exec\_f  
     Log, 84  
 msg\_good\_exec\_q  
     Log, 84  
 msg\_good\_exec\_qr  
     Log, 84  
 msg\_good\_graph  
     Log, 84  
 msg\_good\_path  
     Log, 84  
 msg\_good\_table  
     Log, 84  
 msg\_multiple\_query  
     Log, 84  
 msg\_result  
     Log, 85  
 msg\_success\_managed\_db  
     Log, 85  
 msg\_swap\_db  
     Log, 85  
 msg\_swap\_kg  
     Log, 85  
 msg\_unknown\_error  
     Log, 85  
 mysqlConnector, 91  
     \_\_init\_\_, 94  
     specific\_queries, 94  
 Name  
     PRF1Metric, 103  
     ScalarMetric, 114  
 nlp  
     components.book\_conversion, 10  
     components.text\_processing, 17  
 NOT\_DUMMY\_  
     GraphConnector, 69  
 OnConnectedAsync  
     MetricsHub, 91  
 OnDisconnectedAsync  
     MetricsHub, 91  
 output\_single  
     src.main, 19  
 pandoc\_xml\_path

EPUBToTEI, 57  
ParagraphStreamTEI, 94  
    \_\_init\_\_, 96  
    allowed\_chapters, 97  
    book\_id, 97  
    chunks, 97  
    encoding, 97  
    end\_inclusive, 98  
    lines, 98  
    pre\_compute\_segments, 97  
    root, 98  
    start\_inclusive, 98  
    story\_id, 98  
    stream\_segments, 97  
    tei\_path, 98  
    xml\_namespace, 98  
password  
    DatabaseConnector, 47  
port  
    DatabaseConnector, 47  
Post  
    MetricsController, 89  
post\_basic\_output  
    components.metrics, 15  
post\_example\_results  
    components.metrics, 16  
post\_payload  
    components.metrics, 16  
postgresConnector, 99  
    \_\_init\_\_, 102  
    specific\_queries, 102  
pre\_compute\_segments  
    ParagraphStreamTEI, 97  
pre\_split\_chunks  
    Story, 117  
Precision  
    PRF1Metric, 103  
prefix  
    Log.Failure, 59  
PRF1Metric, 102  
    F1Score, 103  
    Name, 103  
    Precision, 103  
    Recall, 103  
PRF1Metrics  
    SummaryMetrics, 121  
print\_nodes  
    GraphConnector, 70  
print\_triples  
    GraphConnector, 70  
process\_single  
    src.main, 19  
pytest\_addoption  
    tests.conftest, 23  
pytest\_db  
    Log, 85  
QA  
    SummaryMetrics, 121  
QAItem, 103  
    Accuracy, 103  
    GeneratedAnswer, 103  
    GoldAnswer, 103  
    IsCorrect, 104  
    Question, 104  
QAItems  
    QAMetric, 104  
    QAMetric, 104  
        AverageAccuracy, 104  
        QAItems, 104  
QAResults  
    SummaryData, 120  
Question  
    QAItem, 104  
raw\_tei\_content  
    EPUBToTEI, 58  
reader  
    Story, 118  
Recall  
    PRF1Metric, 103  
RED  
    Log, 85  
rel\_db  
    Log, 85  
relational\_db  
    Session, 116  
    tests.test\_components, 25  
RelationalConnector, 105  
    \_\_init\_\_, 107  
    \_split\_combined, 108  
    change\_database, 108  
    check\_connection, 108  
    connection\_string, 112  
    create\_database, 109  
    database\_exists, 109  
    database\_name, 112  
    db\_type, 112  
    drop\_database, 110  
    execute\_query, 110  
    from\_env, 111  
    get\_dataframe, 111  
    test\_connection, 111  
    verbose, 112  
RelationExtractor, 112  
    \_\_init\_\_, 113  
    extract, 113  
    max\_tokens, 113  
    model, 113  
    tokenizer, 113  
    tuple\_delim, 113  
reset  
    Session, 115  
response\_files  
    src.main, 20  
root  
    ParagraphStreamTEI, 98  
run\_f

Log, 86  
 run\_q  
     Log, 86  
 SAME\_DB\_KG\_  
     GraphConnector, 70  
 save\_pandoc  
     EPUBToTEI, 58  
 save\_tei  
     EPUBToTEI, 58  
 ScalarMetric, 114  
     Name, 114  
     Value, 114  
 ScalarMetrics  
     SummaryMetrics, 121  
 sentencizer  
     components.book\_conversion, 10  
     components.text\_processing, 17  
 Session, 114  
     \_\_init\_\_, 115  
     \_\_new\_\_, 115  
     \_instance, 116  
     docs\_db, 116  
     graph\_db, 116  
     relational\_db, 116  
     reset, 115  
     test\_database\_connections, 115  
     verbose, 116  
 session  
     src.main, 20  
     src.setup, 22  
     tests.confest, 23  
 specific\_queries  
     mysqlConnector, 94  
     postgresConnector, 102  
 src, 17  
 src.main, 17  
     book\_chapters, 20  
     book\_id, 20  
     book\_title, 20  
     chapters, 20  
     chunk\_single, 18  
     convert\_from\_csv, 18  
     convert\_single, 18  
     end, 20  
     end\_str, 20  
     epub\_path, 20  
     full\_pipeline, 18  
     graph\_triple\_files, 19  
     output\_single, 19  
     process\_single, 19  
     response\_files, 20  
     session, 20  
     start, 21  
     start\_str, 21  
     story\_id, 21  
     tei, 21  
     test\_relation\_extraction, 19  
     triple\_files, 21  
     src.setup, 21  
         session, 22  
     src.util, 22  
         all\_none, 22  
         check\_values, 22  
         df\_natural\_sorted, 23  
     start  
         src.main, 21  
     start\_inclusive  
         ParagraphStreamTEI, 98  
     start\_str  
         src.main, 21  
 Story, 117  
     \_\_init\_\_, 117  
     \_\_make\_single, 117  
     \_\_merge\_chunks, 117  
     pre\_split\_chunks, 117  
     reader, 118  
     stream\_chunks, 118  
 story\_id  
     ParagraphStreamTEI, 98  
     src.main, 21  
 StoryStreamAdapter, 118  
     stream\_paragraphs, 119  
     stream\_segments, 119  
     stream\_sentences, 119  
 stream\_chunks  
     Story, 118  
 stream\_paragraphs  
     StoryStreamAdapter, 119  
 stream\_segments  
     BookStream, 33  
     ParagraphStreamTEI, 97  
     StoryStreamAdapter, 119  
 stream\_sentences  
     StoryStreamAdapter, 119  
 success  
     Log, 79  
 SUCCESS\_COLOR  
     Log, 86  
 success\_legacy  
     Log, 79  
 Summaries  
     MetricsController, 89  
 SummaryData, 120  
     BookID, 120  
     BookTitle, 120  
     Metrics, 120  
     QAResults, 120  
     SummaryText, 120  
 SummaryMetrics, 121  
     GetDefault, 121  
     PRF1Metrics, 121  
     QA, 121  
     ScalarMetrics, 121  
 SummaryText  
     SummaryData, 120  
 swap\_db

Log, 86  
swap\_kg  
    Log, 86  
system\_prompt  
    LLMConnector, 75  
  
tei  
    src.main, 21  
tei\_path  
    EPUBToTEI, 58  
    ParagraphStreamTEI, 98  
temperature  
    LLMConnector, 75  
test\_basic  
    Log, 86  
test\_conn  
    Log, 86  
test\_connection  
    Connector, 38  
    DocumentConnector, 54  
    GraphConnector, 70  
    LLMConnector, 74  
    RelationalConnector, 111  
test\_database\_connections  
    Session, 115  
test\_db\_docs\_comprehensive  
    tests.test\_components, 25  
test\_db\_docs\_minimal  
    tests.test\_components, 25  
test\_db\_graph\_comprehensive  
    tests.test\_components, 25  
test\_db\_graph\_minimal  
    tests.test\_components, 26  
test\_db\_relational\_comprehensive  
    tests.test\_components, 26  
test\_db\_relational\_minimal  
    tests.test\_components, 26  
test\_df  
    Log, 86  
test\_info  
    Log, 86  
test\_mongo\_example\_1  
    tests.test\_components, 26  
test\_mongo\_example\_2  
    tests.test\_components, 26  
test\_mongo\_example\_3  
    tests.test\_components, 26  
test\_relation\_extraction  
    src.main, 19  
test\_sql\_example\_1  
    tests.test\_components, 27  
test\_sql\_example\_2  
    tests.test\_components, 27  
test\_tmp\_db  
    Log, 86  
tests, 23  
tests.conftest, 23  
    pytest\_addoption, 23  
    session, 23  
  
tests.test\_components, 24  
    \_test\_query\_file, 24  
docs\_db, 25  
graph\_db, 25  
load\_examples\_relational, 25  
relational\_db, 25  
test\_db\_docs\_comprehensive, 25  
test\_db\_docs\_minimal, 25  
test\_db\_graph\_comprehensive, 25  
test\_db\_graph\_minimal, 26  
test\_db\_relational\_comprehensive, 26  
test\_db\_relational\_minimal, 26  
test\_mongo\_example\_1, 26  
test\_mongo\_example\_2, 26  
test\_mongo\_example\_3, 26  
test\_sql\_example\_1, 27  
test\_sql\_example\_2, 27  
  
text  
    Chunk, 35  
title\_key  
    Book, 30  
tokenizer  
    RelationExtractor, 113  
triple\_files  
    src.main, 21  
tuple\_delim  
    RelationExtractor, 113  
  
url  
    components.metrics, 16  
USE\_COLORS  
    Log, 87  
username  
    DatabaseConnector, 47  
  
Value  
    ScalarMetric, 114  
verbose  
    DatabaseConnector, 47  
    DocumentConnector, 55  
    GraphConnector, 71  
    RelationalConnector, 112  
    Session, 116  
  
warn  
    Log, 80  
WARNING\_COLOR  
    Log, 87  
WHITE  
    Log, 87  
  
xml\_namespace  
    EPUBToTEI, 58  
    ParagraphStreamTEI, 98  
  
YELLOW  
    Log, 87