Module Interface Specification for Software Engineering

Team #18, Gouda Engineers
Aidan Goodyer
Jeremy Orr
Leo Vugert
Nathan Perry
Tim Pokanai

January 21, 2026

1 Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at https://github.com/OCD-Rats-Capstone/OCD-Rat-Infrastructure/blob/main/docs/SRS-Volere/SRS.pdf

Contents

3 Introduction

The following document details the Module Interface Specifications for **RatBat2**, a data analysis web application designed to visualize, query, and process behavioural data from experiments involving rats with Obsessive-Compulsive Disorder (OCD). The system enables researchers to upload experimental trial data, perform natural language–based searches, and generate dynamic visualizations for behavioural comparisons and trend analysis.

Complementary documents include the *System Requirements Specification (SRS)* and *Module Guide (MG)*. The full documentation and implementation can be found at https://github.com/OCD-Rats-Capstone/OCD-Rat-Infrastructure.

4 Notation

This document adopts a formal notation and structural convention to describe the architecture and module interface specifications (MIS) for the Behavioral Data Analysis Platform for Animal Models of OCD.

The structure of each MIS follows the framework of Hoffman and Strooper (1995), extended to incorporate template modules as described by Ghezzi et al. (2003). The mathematical and logical notation is consistent with Chapter 3 of Hoffman and Strooper (1995), with domain-specific adaptations for data processing and behavioral event analysis.

Mathematical and Logical Conventions

- The symbol `:=` denotes an assignment or multiple assignment statement. For example, `session.avg_latency := sum(latency) / count(latency)`.
- Conditional expressions follow the form:

$$(c_1 \Rightarrow r_1 \mid c_2 \Rightarrow r_2 \mid \ \ldots \ \mid c_n \Rightarrow r_n)$$

  For instance:

  (event.type = ”drug 1” ⇒ increment(count) | event.type = ”drug 2” ⇒ record(duration))

  where the rule corresponding to the first true condition is applied.

- Logical connectives are used as follows:
  - $\wedge$ — logical AND
  - $\vee$ — logical OR
  - ‘ — logical NOT
  - $\Rightarrow$ — implication

- Set notation follows standard mathematical conventions: $\{x \mid P(x)\}$ represents the set of all x satisfying predicate $P(x)$. Example: $\{e \in Events \mid e.duration > 10s\}$ denotes the set of long-duration events.
- Ranges are denoted as $[a..b]$, representing all integer time indices t such that $a \leq t \leq b$.
- Function definitions are expressed as mappings:

$$f : Input \rightarrow Output$$

  Example:

$$\texttt{computeMetrics} : SessionData \rightarrow BehavioralSummary$$

Data and Type Notation

- *Session* — a structured dataset representing a single experimental trial.
- *Event* — a tuple of attributes describing a behavioral observation, e.g. $(type, timestamp, duration)$.
- *Metric* — a computed quantitative value derived from one or more events.
- *AnimalID* — a unique identifier for a subject.
- $TrialSet := \{s_1, s_2, ..., s_n\}$ — the set of all sessions recorded for a given animal.

Generally python notation or something similar will be used for data types which is outlined below. Additionally, some additional data types are outlined in this section. For simplicity, some data types will be desribed in simple language.

Data Type	Notation
Integer	int
String	String
Float	float
JSON Object	JSON Object
HTTP Response/Request	HTTP Response/Request
File	File
[]	Array of the prepended data type (e.g. int[])
{}	Map of prepended data type (e.g. String{})
SQL Query	SQL Query
Boolean	bool

Units and Measurement Conventions

- Time values are expressed in seconds (s).
- Counts and frequencies are represented as integers.
- Statistical metrics (e.g., mean, standard deviation, z-score) are expressed as real numbers ($\mathbb{R}$).

These conventions are used consistently throughout the system specification to ensure mathematical clarity and facilitate unambiguous interpretation of the behavioral data models, algorithms, and transformations.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	Front-End Interface Module API Layer Module Data Schema and Storage Module Data Visualization Module
Software Decision Module	NLP Query Processor Data Processing Pipeline Module Authentication and Access Control Module Fault and Error Management Module

Table 1: Module Hierarchy

6 MIS of M1: Hardware-Hiding Module

6.1 Module

The hardware hiding module will be implemented by the native OS of the machine running our system. This will not be designed by us. For detailed design implementation please see the breakdown of the OS of the local machine running our system's web page (client-side) or the Linux-based OS of the server hosting our system's backend and database.

7 MIS of M2: Query Processor Module

7.1 Module

NLPModule

7.2 Uses

- M9: Fault and Error Management

7.3 Syntax

7.3.1 Exported Constants

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
NLP Processor	String	String	incoherent prompt, empty input

7.4 Semantics

7.4.1 State Variables

None

7.4.2 Environment Variables

- LLM_API_KEY: String
- $System\_Prompt$: String
- $Schema\_Attributes$: String
- $Valid\_Operators$: String

7.4.3 Assumptions

- The system prompt environment variable will provide enough information to the LLM to ensure that the returned response string conforms exactly to the structure of the needed input format for a database query.
- The large language model used to process the natural language input into database commands will require the use of an API key and will not be locally hosted, thus the need for an API key environment variable.

7.4.4 Access Routine Semantics

$NLP\_Processor(String : natural\_language) \rightarrow String$:

- output: out := if $(is\_valid\_output(natural\_language))$ *then* $\{LLM\_API\_Response(natural\_language)\}$ *else* $\{raise\ incoherent\_prompt\}$
- exceptions: $(natural\_language$ IS NULL $=> empty\_input)$, $(!is\_valid\_output(natural\_language) => incoherent\_prompt)$

7.4.5 Local Functions

$is\_valid\_output(String : LLM\_Response) \rightarrow bool$

- output: out := (b such that b $\in$ $([x \in Schema\_Attributes]; [y \in Valid\_Operators]; [< Value >]))$

$LLM\_API\_Call(String : natural\_language, String : System\_Prompt) \rightarrow String$

- $output : LLM\_Response := (Call\ LLM\ API\ where :$ $Promptinput = natural\_language\ prepended\ by\ System\_Prompt)$

8 MIS of M3: Front-End Interface Module

8.1 Module

FrontEnd

8.2 Uses

- M7: Data Processing Pipeline Module
- M4: API Layer Module
- M6: Data Visualization Module
- M9: Fault and Error Management Module
- M8: Authentication and Access Control Module

8.3 Syntax

8.3.1 Exported Constants

N/A

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
Submit Search Query	QueryParameters	DisplayedResults	invalid query
Apply Filter	FilterCriteria	UpdatedResults	invalid filter
Render Visualization	VisualizationConfig, DataSet	RenderedChart	unsupported chart type
Download Data	FormatType, DataSet	File	unsupported format, empty dataset

8.4 Semantics

8.4.1 State Variables

- NavPage: String (URL)
- HTTPStatusCode: int
- UserInterfaceComponents: bool, String, int, char (Variables which provide values to UI components such as a checkbox being checked or the value in an input box. Aggregated as it would be a waste of time to list each individual component and it's state variable(s))
- DataSet: JSON Object

8.4.2 State Invariant

- $NavPage \in \{FrontEndURLs\}$
- $HTTPStatusCode \in \{$Standard HTTP Status Codes$\}$

8.4.3 Environment Variables

- BackEndLocation: String (URL for HTTP request destinations)
- FrontEndURLs: String

8.4.4 Assumptions

- Only one backend location will be necessary. HTTP request body will contain sufficient information to call the correct corresponding module and serve the proper response.
- Front end module will be free of explicit business logic and will simply package user input and serve output responded. Aggregating user input, formatting HTTP requests and serving responses are the extent of the front end requirements.

8.4.5 Access Routine Semantics

N/A

8.4.6 Local Functions

$HTTP\_Send\_Request(HTTP\ Request : HTTP\_request) \rightarrow HTTP\_Response$:

- output: $out := HTTP\_response\ from\ backend\ modules$
- exception: None

$Nav\_Page(String : Nav\_URL)$:

- $transition : NavPage := Nav\_URL$
- $exception : exc := (NavPage = bad\ URL => page\_not\_found)$

$HTTP\_Error\_Handle(HTTP\ Response : HTTP\_response)$:

- $transition : HTTPStatusCode := HTTP\_response.StatusCode, NavPage := 'Initial\ Landing\ Page', UserInterfaceComponents :=$ {component initial values}
- exception: None

$HTTP\_Serve\_Data$(JSON Object: DataSet):

- $transition : DataSet := HTTP\_Send\_Request(HTTP\_request).body$
- $exception : exc := (HTTP\_Send\_Request(HTTP\_request).body == NULL =>$ Empty HTTP Response)

9 MIS of M4: API Layer Module

9.1 Module

APILayer

9.2 Uses

- M5: Data Schema and Storage Module
- M2: NLP Query Processor Module
- M8: Authentication and Access Control Module

9.3 Syntax

9.3.1 Exported Constants

N/A

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
Process Structured Query	StructuredQuery	JSON Object	malformed query, empty result
Process NLP Query	NaturalLanguageQuery	JSON Object	unrecognized intent, NLP failure
Get Dataset	QueryResultID	DataSet (JSON)	invalid ID, empty dataset
Request RawFiles	DataSet{"FileLocations"}	File[]	bad URL, missing files
Authenticate Request	AuthToken	AccessDecision	invalid token

9.4 Semantics

9.4.1 State Variables

- $QueryType : String \in \{'NaturalLanguage', 'Structured'\}$
- ModuleHasData: bool
- Dataset: JSON Object
- RequestedQuery: SQL Query

9.4.2 Environment Variables

- HostedDatabaseEndPoint: String (URL for Database Connection)
- DatabaseAccessKey: String
- FRDRDataSetEndPoint: String (URL EndPoint for Dr.Szechtman's Data Sets in the FRDR repository)
- BaseQuery: SQL Query (To be augmented by requested filters)

9.4.3 Assumptions

- Only one instance of our database will be hosted and thus, one hardcoded endpoint will be sufficient for handling our database connection. Additionally, a single access key will be sufficient to provide all users access to the database connection.
- The Endpoint for accessing Dr.Szechtman's datasets on the FRDR website will not change.
- The disk space available on the application server will be sufficient to temporarily download several .CSV or .MPG files files from FRDR for processing purposes.
- A boilerplate query for the database will be sufficient for every unique query, only needing to be augmented algorithmically through the requested filters.

9.4.4 Access Routine Semantics

$Get\_Dataset() \rightarrow File[]$:

- output: out := DataSet
- exceptions: exc:= (HasData := False => empty_dataset), exc := ($HTTP\_Response \in \{4++\}$)

$Request\_RawFiles(JSON\ Object : DataSet) :\rightarrow File[]$

- *transition* : *out* :=Array of files provided via automatic downloads faciliatated by HTTP requests to the DataSet{"FileLocations"} Attribute
- *exception* : $exc := (HasData := False => Empty\_Result)$,
  $exc := (Database\ Connection\ Unsuccessful => Connection\_Refused)$

9.4.5 Local Functions

$Choose\_Query\_Process(String : QueryType) \rightarrow JSON\ Object$:

- transition: $DataSet := if(QueryType ==' NaturalLanguage'\{NLP\_Query(HTTP\_Request)\}$
  $elseif(QueryType ==' Structured'\{Structured\_Query(HTTP\_Request)\}$
- exception: $exc := (\not(QueryType \in \{'NaturalLanguage','Structured'\}) => Bad\_Query\_Type)$

$NLP\_Query(HTTP\ Request : HTTP\_Request) :\rightarrow SQL\ Query$

- *transition* : *RequestedQuery* := Augmented Base Query Based on Output of NLP-Module.NLP_Processor(HTTP_Request.body)
- *exception* : $exc := (HTTP\_Request.body\ IS\ NULL => empty\_filters)$

$Structured\_Query(HTTP\ Response : HTTP\_response) :\rightarrow SQL\ Query$

- $transition$: $RequestedQuery$:= Augmented Base Query Based on filters in HTTP_Request.body
- $exception$: $exc := (HTTP\_Request.body\ IS\ NULL => empty\_filters)$

$Request\_DataSet(SQL\ Query : RequestedQuery, String : HostedDatabaseEndPoint) :\rightarrow JSON\ Object$

- $transition$: $DataSet$:=JSON Object generated from result of RequestedQuery API request at URL of HostedDatabaseEndPoint
- $transition : if(DataSet\ IS\ NOT\ NULL)\{HasData := True\}$
- $exception : exc := (HasData := False => Empty\_Result), exc :=$ $(DatabaseConnectionUnsuccessful => Connection\_Refused)$

10 MIS of M5: Data Schema and Storage Module

10.1 Module

DataBaseSchema

Note: It does not make sense to outline this as a typical module with functions, variables etc... since this module consists of our database schema for querying the experimental data relevant to this project. As a result, this module will be outlined via the schema definition of our database. Also note that our team and Dr.Szechtman have already worked to develop a schema that makes sense based on Dr.Szechtman's understanding of the datasets. Thus, this will be very detailed and specific, more so than is likely necessary but since the work has already been done, the full design will be outlined.

10.2 Schema Overview

10.2.1 Core Entity Tables

- 1. apparatuses - Testing Equipment
- 2. rats - Subject Animals
- 3. brain_mainpulations - Surgical interventions
- 4. drug_rx - Drug regimen combinations
- 5. drug_rx_details - Individual compound specifications

10.2.2 Reference Tables

- 6. light_cycles - Colony room light/dark cycles
- 7. session_types - Session classification codes
- 8. testers - Experimenter identifiers
- 9. apparatus_patterns - Object arrangement configurations
- 10. drugs - Pharmacological agents
- 11. brain_regions - Brain region codes
- 12. testing_rooms - Physical testing locations

10.2.3 Experimental Session Tables

- 13. experimental_sessions - All 20,491 experimental sessions
- 14. session_drug_details - Drug administration for experimental sessions

10.2.4 Histology and Video Tables

- 15. histology_results – Post-mortem lesion verification
- 16. movie_files – Video file metadata

10.2.5 Data File Locations Table

- 17. data_file_locations – Physical storage locations (336,507 records)

10.2.6 Session Data Files and Observations Tables

- 18. session_data_files – Session-to-data-file linking
- 19. session_observations – Quality control and notes

10.3 Schema Definition

- 1. apparatuses - Testing Equipment
  - **Schema Definition:** CREATE TABLE apparatuses (apparatus_id SERIAL PRIMARY KEY, apparatus_name VARCHAR(100), apparatus_code VARCHAR(10), testing_room_id INTEGER REFERENCES testing_rooms(room_id), apparatus_location_in_room VARCHAR(50), apparatus_notes TEXT);
  - **Referenced By:** experimental_sessions **Foreign Key:** apparatus_id (Many-to-One: many sessions use same apparatus)

- 2. rats - Subject Animals
  - **Schema Definition:** CREATE TABLE rats (rat_id SERIAL PRIMARY KEY, legacy_rat_id VARCHAR(20) UNIQUE, sex VARCHAR(10), light_cycle_id INTEGER REFERENCES light_cycles(light_cycle_id), strain VARCHAR(50), birth_date DATE);
  - **Referenced By:** experimental_sessions **Foreign Key:** rat_id (One-to-Many: one rat, many sessions)
  - **Referenced By:** brain_manipulations **Foreign Key:** rat_id (One-to-Many: one rat can have multiple manipulations)
- 3. brain_mainpulations - Surgical interventions
  - **Schema Definition:** CREATE TABLE brain_manipulations (manipulation_id SERIAL PRIMARY KEY, rat_id INTEGER REFERENCES rats(rat_id), surgery_type VARCHAR(50), target_region_id INTEGER REFERENCES brain_regions(region_id), surgery_date DATE);
  - **Referenced By:** histology_results table **Foreign Key:** manipulation_id (One-to-One: each lesion has histology assessment)
- 4. drug_rx - Drug regimen combinations
  - **Schema Definition:** CREATE TABLE drug_rx (drug_rx_id SERIAL PRIMARY KEY, rx_label VARCHAR(200), num_drugs INTEGER, notes_drug_rx TEXT);
  - **Referenced By:** experimental_sessions **Foreign Key:** drug_rx_id (Many-to-One: many sessions use same regimen)
  - **Referenced By:** drug_rx_details **Foreign Key:** drug_rx_id (One-to-Many: each regimen has multiple detail records)
- 5. drug_rx_details - Individual compound specifications
  - **Schema Definition:** CREATE TABLE drug_rx_details (drug_rx_detail_id SERIAL PRIMARY KEY, drug_rx_id INTEGER REFERENCES drug_rx(drug_rx_id), drug_id INTEGER REFERENCES drugs(drug_id), prescribed_dose NUMERIC(10,6), dose_unit VARCHAR(50), route VARCHAR(50), time_before_session_hours NUMERIC(10,2));
- light_cycles
  - **Schema Definition:** CREATE TABLE light_cycles (light_cycle_id SERIAL PRIMARY KEY, cycle_name VARCHAR(50) NOT NULL, lights_on_time TIME, lights_off_time TIME);

  - **Referenced By:** rats **Foreign Key:** light_cycle_id (Many-to-One: many rats share same light cycle)

- session_types - Session classification codes
  - **Schema Definition:** CREATE TABLE session_types (session_type_id SERIAL PRIMARY KEY, type_name VARCHAR(100) NOT NULL, session_types_notes TEXT);
  - **Referenced By:** experimental_sessions **Foreign Key:** session_type_id (Many-to-One: many sessions can be the same type)

- 8. testers - Experimenter identifiers
  - **Schema Definition:** CREATE TABLE testers (tester_id SERIAL PRIMARY KEY, first_last_name VARCHAR(100), initials VARCHAR(10));
  - **Referenced By:** experimental_sessions **Foreign Key:** tester_id (Many-to-One: each tester conducted many sessions)

- 9. apparatus_patterns - Object arrangement configurations
  - **Schema Definition:** CREATE TABLE apparatus_patterns (pattern_id SERIAL PRIMARY KEY, pattern_description TEXT);
  - **Referenced By:** experimental_sessions **Foreign Key:** pattern_id (Many-to-One: many sessions use the same pattern)

- 10. drugs - Pharmacological agents
  - **Schema Definition:** CREATE TABLE drugs (drug_id SERIAL PRIMARY KEY, drug_abbreviation VARCHAR(20), drug_name VARCHAR(200), drug_is_active BOOLEAN, dose_unit VARCHAR(20), drugs_notes TEXT);
  - **Referenced By:** drug_rx_details **Foreign Key:** drug_id (Many-to-One: specifies drugs in prescription regimens)
  - **Referenced By:** session_drug_details **Foreign Key:** drug_id (Many-to-One: records actual drugs administered)

- 11. brain_regions - Brain region codes
  - **Schema Definition:** CREATE TABLE brain_regions (region_id SERIAL PRIMARY KEY, region_name VARCHAR(100), region_abbreviation VARCHAR(20));
  - **Referenced By:** brain_manipulations **Foreign Key:** target_region_id (Many-to-One: many rats received surgery targeting the same brain region)

- 12. testing_rooms - Physical testing locations

- **Schema Definition:** CREATE TABLE testing_rooms (room_id SERIAL PRIMARY KEY, room_name VARCHAR(50), room_notes TEXT);
- **Referenced By:** apparatuses **Foreign Key:** testing_room_id (Many-to-One: multiple apparatus can be in the same room)

- 13. experimental_sessions - All 20,491 experimental sessions
  - **Schema Definition:** CREATE TABLE experimental_sessions (session_id SERIAL PRIMARY KEY, legacy_session_id VARCHAR(7) UNIQUE, session_type_id INTEGER REFERENCES session_types(session_type_id), rat_id INTEGER REFERENCES rats(rat_id), body_weight_grams NUMERIC(6,2), drug_rx_id INTEGER REFERENCES drug_rx(drug_rx_id), tester_id INTEGER REFERENCES testers(tester_id), effective_manipulation_id INTEGER REFERENCES brain_manipulations(manipulation_id), apparatus_id INTEGER REFERENCES apparatuses(apparatus_id), pattern_id INTEGER REFERENCES apparatus_patterns(pattern_id), locale_in_room VARCHAR(20), room_id INTEGER REFERENCES testing_rooms(room_id), testing_lights_on SMALLINT, session_timestamp TIMESTAMP, data_trial_id VARCHAR(28), cumulative_drug_injection_number INTEGER);
  - **Referenced By:** session_drug_details **Foreign Key:** session_id (One-to-Many. Each session has 1-4 drug detail records specifying actual drugs and doses administered)
  - **Referenced By:** session_data_files **Foreign Key:** session_id (One-to-Many. Each session links to multiple data files (videos, tracks, plots))
  - **Referenced By:** session_observations **Foreign Key:** session_id (One-to-Many. Quality control notes and behavioral observations for sessions)

- 14. session_drug_details - Drug administration for experimental sessions
  - **Schema Definition:** CREATE TABLE session_drug_details (detail_id SERIAL PRIMARY KEY, session_id INTEGER REFERENCES experimental_sessions(session_id), drug_id INTEGER REFERENCES drugs(drug_id), dose_given NUMERIC(12,5), dose_unit VARCHAR(20), route VARCHAR(50), time_before_session_hours NUMERIC(10,2), cumulative_injections_count_for_regimen INTEGER, cumulative_apparatus_exposure_number INTEGER);

- 15. histology_results – Post-mortem lesion verification
  - **Schema Definition:** CREATE TABLE histology_results (histology_id SERIAL PRIMARY KEY, manipulation_id INTEGER REFERENCES brain_manipulations(manipulation_id), histology_status_id INTEGER REFERENCES lkp_histology_status(histology_status_id), intact_status_id INTEGER REFERENCES

functional_intact_status(intact_status_id), left_percent_damage DOUBLE PRECISION, right_percent_damage DOUBLE PRECISION, notes_histology_results TEXT);

- 16. movie_files – Video file metadata
  - **Schema Definition:** CREATE TABLE movie_files (movie_file_id SERIAL PRIMARY KEY, movie_file_name VARCHAR(255), legacy_movie_file_name VARCHAR(255), file_size_mb DOUBLE PRECISION, is_file_available BOOLEAN, notes_movie_files TEXT);
  - **Referenced By:** session_data_files **Foreign Key:** .movie_file_id (Many-to-One: Multiple sessions can reference the same movie file (video multiplexing))
- 17. data_file_locations – Physical storage locations (336,507 records)
  - **Schema Definition:** CREATE TABLE data_file_locations (data_file_location_id SERIAL PRIMARY KEY, data_file_id INTEGER REFERENCES session_data_files(data_file_id) repository_type_id INTEGER REFERENCES lkp_repository_type(repository_type_id), repo_dataset_doi VARCHAR(100), repo_dataset_name VARCHAR(255), repo_dataset_version VARCHAR(24), repo_file_url TEXT, is_primary_copy BOOLEAN, date_linked TIMESTAMP, api_endpoint VARCHAR(255), notes_data_file_locations TEXT);
- 18. session_data_files – Session-to-data-file linking
  - **Schema Definition:** CREATE TABLE session_data_files (data_file_id SERIAL PRIMARY KEY, session_id INTEGER REFERENCES experimental_sessions(session_id), movie_file_id INTEGER REFERENCES movie_files(movie_file_id), start_frame INTEGER, object_type_id INTEGER REFERENCES lkp_data_file_object_type(object_type_id), file_extension VARCHAR(10), file_name VARCHAR(255), file_size_mb DOUBLE PRECISION, notes_session_data_files TEXT);
  - **Referenced By:** data_file_locations **Foreign Key:** ..data_file_id (One-to-Many. Each file record has multiple location records (local, FRDR v1, FRDR v2, backups))
- 19. session_observations – Quality control and notes
  - **Schema Definition:** CREATE TABLE session_observations (observation_id SERIAL PRIMARY KEY, session_id INTEGER REFERENCES experimental_sessions(session_id), observation_text TEXT, num_falls_during_test INTEGER, falls_during_test_time_when_fell_str VARCHAR(255));

11 MIS of M6: Data Visualization Module

11.1 Module

Visualization Engine

11.2 Uses

- M4: API Layer Module
- M1: Hardware-Hiding Module

11.3 Syntax

11.3.1 Exported Constants

This module exports a set of predefined visualization/chart type constants.

- **VTYPE_BAR** String constant representing bar chart.
- **VTYPE_PLOT** String constant representing a general plot area (points, curves).
- **VTYPE_PIE** String constant representing a pie chart.
- **VTYPE_HEATMAP** String constant representing an (x,y) heatmap.
- **VTYPE_TIMESERIES** String constant representing time series (x,y,t) position data.

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
GenerateViz	(VType, DataBlob, ChartConfig)	Viz_SVG	InvalidDataSchema

11.4 Semantics

11.4.1 State Variables

The Data Visualization Model acts as a stateless data transformation stage and therefore has no associated state variables. It is stateless because all visualization information (data, type, and configuration) is provided through parameters to GenerateViz, so no persistent internal state is required.

11.4.2 Environment Variables

- **D3_Library**: External Dependency

11.4.3 Assumptions

- All input data provided via DataBlob is in valid JSON form.
- There will be continued D3 support and existence of all visualization types exported.
- The client environment supports modern SVG standards.

11.4.4 Access Routine Semantics

GenerateViz(VType, DataBlob, ChartConfig):

- **Transition: None**
- **Output: Viz_SVG** (String) representing the rendered D3.js SVG visualization.

  Note that the image is represented by the XML path string for the resulting SVG.
- exception: **InvalidDataSchema**: Error raised if the JSON DataBlob is incompatible with the specified visualization type.

  For example, most charts are constructed from a tabular form. If this is violated the exception will be raised.

11.4.5 Local Functions

VerifyDataSchema(Data,VType):

- **Purpose:** to check if the JSON data blob conforms to the input specification of the requested visualization. For example, a scatter plot requires 2 vectors for x and y data, while a bar chart requires categorical (key,value) pairs.
- **Output: Boolean** True if valid, False otherwise.

ApplyD3Template(Data,Type,Config):

- **Purpose:** Convert the validated input data into the specified D3 template, and generate the SVG output.
- **Output:** SVG (String).

12 MIS of M7: Data Processing Pipeline Module

12.1 Module

DataProcessingPipeline

12.2 Uses

This module interfaces with:

- M4: API Layer Module
- M9: Fault and Error Management Module

12.3 Syntax

12.3.1 Exported Constants

None.

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
ProcessData	Dataset (CSV / JSON), Parameters (dict)	Processed DataFrame / JSON	invalid format, missing fields
ComputeMetrics	Processed DataFrame / JSON	Metrics (dict)	empty input, computation error
GenerateSummary	Processed DataFrame / JSON	Summary Statistics (dict)	invalid input

12.4 Semantics

12.4.1 State Variables

None. All computations are stateless and performed on data inputs provided per request.

12.4.2 Environment Variables

- **DATA_CACHE_PATH:** String – temporary directory for caching intermediate results.
- **MAX_WORKERS:** Integer – controls the number of concurrent processing threads.

12.4.3 Assumptions

- Input datasets conform to schema definitions from Data Schema and Storage Module.
- Parameters for filtering, grouping, or aggregation are valid and correspond to known attributes.
- The environment has sufficient memory and CPU resources for parallel processing.

12.4.4 Access Routine Semantics

ProcessData(Dataset, Parameters) → Processed DataFrame:

- **output:** out := cleaned and structured dataset prepared for computation.
- **exception:** invalid format if dataset fails schema validation; missing fields if required attributes are absent.

ComputeMetrics(Processed DataFrame) → Metrics:

- **output:** out := computed behavioral metrics (e.g., grooming duration, trial success rate).
- **exception:** empty input if dataset is empty; computation error if numerical operation fails.

GenerateSummary(Processed DataFrame) → Summary Statistics:

- **output:** out := dictionary of aggregated statistics (mean, median, standard deviation).
- **exception:** invalid input if DataFrame structure mismatches expected schema.

12.4.5 Local Functions

validateInput(Dataset) → bool

- **output:** true if dataset matches schema attributes and expected datatypes.

computeAggregateMetrics(DataFrame) → dict

- **output:** dictionary of computed metrics (e.g., total grooming time, movement variance).

cleanData(Dataset) → DataFrame

- **output:** dataset with missing values handled and outliers removed.

13 MIS of M8: Authentication and Access Control Module

13.1 Module

The Authentication and Access Control Module is responsible for verifying user identities, assigning access roles, and ensuring scalable entry points for users that access the software through the web application or through our API clients.

13.2 Uses

None

Additionally, this module will use external libraries, including FastAPI's OAuth2PasswordBearer, and JWT libraries for token generation, encryption, and validation.

13.3 Syntax

13.3.1 Exported Constants

N/A

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
Login User	String Username, String Password	String JWT Token	InvalidCredentialsError
Verify Token	String JWT Token	User Object	TokenExpiredError, InvalidTokenError
Get User	String JWT Token	User Object	Authentication Error
Authorize Role	User Object, String JWT Token	Boolean	AccessDeniedError
Logout User	String JWT Token	Boolean	InvalidTokenError
Refresh Token	String JWT Token	String JWT Token	InvalidTokenError, TokenExpiredError

13.4 Semantics

13.4.1 State Variables

- User Credentials: A HashMap of usernames to password hashes.
- Active Tokens: Dictionary of JWT Token Strings mapped to Expiry Timestamps.
- Permissions Mapping: Dictionary of Role Strings mapped to Permission Strings.

13.4.2 Environment Variables

- JWT SECRET KEY: User key used for token signing and verification.
- MAX CONCURRENT USERS: Maximum allowed concurrent users.
- NUM CONCURRENT USERS: Number of current concurrent users.
- USER DATABASE URL: Connection URL to the credential and role management database.
- TOKEN EXPIRY: Token lifetime configuration for user session.

13.4.3 Assumptions

- Authentication requests may orginiate concurrently from both UI or API clients.
- Passwords are stored securely using a one-way hash.
- HTTPS is enforced for all login and token exchange operations.

13.4.4 Access Routine Semantics

loginUser():

- transition: Verifies credentials from userCredentials.
- output: JWT Token as a String.
- exception: InvalidCredentialsError.

verifyToken():

- transition: Checks JWT Token signature, expiry time, and status in activeTokens.
- output: User Object if token is valid.
- exception: InvalidTokenError or TokenExpiredError.

getUser():

- transition: None.
- output: User Object.
- exception: AuthenticationError.

authorizeRole():

- transition: None.

- output: True if authorized, False otherwise.
- exception: AccessDeniedError.

logoutUser():

- transition: Removes identifier from activeTokens, logging out the user.
- output: True if successful, False otherwise.
- exception: InvalidTokenError.

refreshToken():

- transition: Validates token and reissues a fresh JWT Token with renewed expiry time.
- output: JWT Token as a String
- exception: InvalidTokenError or TokenExpiredError.

13.4.5 Local Functions

- hashPassword(String password): Generates a secure hash of a user password.
- createJWTToken(User Object, int expiry): Generates a JWT Token containing user data and expiry period.
- decodeJWTToken(String JWTToken): Decodes and validates a JWT Token, extracting user data and expiry period.
- validateUser(String username, String password): Authenticates user via stored credentials.

References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.

14 MIS of M9: Fault and Error Management Module

14.1 Module

This module is responsible for tracking, logging, alerting, and automatically recovering from errors and faults in the system. It maintains both transient and persistent records of errors, interfaces with dependent modules to capture exceptions, and ensures system reliability and observability.

14.2 Uses

None

14.3 Syntax

14.3.1 Exported Constants

- `LOG_LEVELS` – Enumerated type {DEBUG, INFO, WARNING, ERROR, CRITICAL} specifying verbosity levels for logging.

14.3.2 Exported Access Programs

Name	In	Out	Exceptions
`logEvent`	message: string, level: LOG_LEVELS	success: string	ValidationError
`isTrue`	two variables of the same data type	success: boolean	ValidationError
`getFaultHistory`	moduleID: string, timeRange: tuple	faultList: list	DatabaseError
`raiseError`	message: type of error, severity: LOG_LEVELS	success: boolean	IOError

14.4 Semantics

14.4.1 State Variables

- `errorBuffer`: Temporary queue of recent error events awaiting persistence or notification.

14.4.2 Environment Variables

- Nothing to be listed that fits the criteria.

14.4.3 Assumptions

- All dependent modules provide standardized error codes and messages.
- Persistent storage is available for fault logging.
- System has sufficient resources (disk space, memory) to maintain logs and state variables.

14.4.4 Access Routine Semantics

`purgeOldLogs()`:

- description: Removes outdated log entries from the system based on the retention policy.

15 Appendix

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Problem Analysis and Design.

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?

   The team was much more focused this deliverable than we had been for the previous deliverable (VnV Plan). Now that midterms are over, we had more time to really spend a lot of time trying to do this deliverable well. The previous deliverable was less of a priority due to the midterms dwarfing it in terms of grade weight. We think this resulted in a more well done, coherent and complete deliverable that puts us in a good spot as we begin the implementation of our system. Additionally, the teamwork aspect of this deliverable was also quite good this deliverable. This likely also ties back into having more time in general but group members were more willing to take on more parts and help others that needed it. Two members even focused largely on the PoC rather than writing the design document because the other three members had the capacity to shoulder a larger documentation load.

2. What pain points did you experience during this deliverable, and how did you resolve them?

   There were not too many major pain points experienced during this deliverable, it generally went quite well. The first pain point of note would be ensuring consistency across all parts and between documents. Although we work as a team, we mainly complete the section individually and thus we need to ensure that two members don't write things in different sections that are not aligned with each other. This deliverable's sections were the most interconnected yet and thus it will require a pretty thorough document review to ensure everything is consistent. The second pain point for this deliverable was the MIS design of each module. This was the final section to be completed as we were not quite sure how to split it up or exactly what was required. As it turns out this is not just one section but rather a section for each module (so essentially 10 additional sections of the document) which was quite a large workload to complete, especially since it was left right until the end. This was simply resolved

by splitting up the modules based on who had the capacity to take them on and to commit the needed time to completing it. As mentioned earlier, members generally had time to do this so it wasn't a terribly painful process.

3. Which of your design decisions stemmed from speaking to your client(s) or a proxy (e.g. your peers, stakeholders, potential users)? For those that were not, why, and where did they come from?

   First and foremost, essentially all of module 5 was developed by speaking with our client, Dr. Henry Szechtman. He is the definitive expert on these datasets and thus he is most equipped to develop a schema for our system. Luckily, he has enough technical knowledge to develop this for us to look over and decide if it is workable, which we believe it is. Additionally, the data-processing module, the visualization module and the front-end module were all largely influenced by our discussions with Dr.Szechtman, specficially through him describing to us what specific functions he wants the system to have (e.g. data processing abilities, front-end design, available visualizations). Outside of these design aspects, most of the rest of the design was created outside of much influence from our client. This includes things like the natural language processing module, the fault and error management module, the API layer module etc... The reason that these decisions were not made based on client interactions is because they are more purely part of what could be called the 'intermediate technical design' rather than user specifications. For example, the natural language processing module requires it to be designed such that natural language can be accurately converted into a SQL query. Alternatively, the error management module simply regulates UI, API and query inputs to prevent any undesired behaviour. Both of these design outlines do not require much input from the client/users as they are basically purely technical implementation (there are not really many different ways to specify natural language into a SQL query). As such, they are left to the technical experts which are the members of our group.

4. While creating the design doc, what parts of your other documents (e.g. requirements, hazard analysis, etc), it any, needed to be changed, and why?

   (a) Our functional requirement related to serving pre-defined datasets will need to be revised. After further discussion with our client, we elicited this requirement in more detail and he is looking more so for a 'web store' outline in the sense that you can sort through the available records based on various available categories rather than displaying specific data sets as 'products' which is how we initially understood it.

   (b) Our database schema containing all of the data related to the session trials will now be implemented specifically using a PostGres SQL server. This will need to be specified in our SRS.

   (c) We are not technically using the FRDR API directly because it only serves the metadata for the datasets. Accessing the data objects themselves are done via an

HTTP request to a specified URL that the FRDR organization provides to us. This is not technically using the API and is rather just making HTTP requests to publicly available URLs that exist within the FRDR domain. This requirement should be updated accordingly.

(d) Likewise, our hazard related to the API layer should be slightly augmented since we are going to be dealing with HTTP requests. This hazard should now focus on hazards related specifically to HTTP such as what could happen due to various error codes and othe related issues that could come up.

(e) Initially, the idea for our database was that all the data is publicly available to all users at all times. We are beginning to develop a new requirement which would both be an addition (as it's a new requirement) and a revision to this publicly available clause. The idea is that users can essentially create a materialized view for their experiments and save them on the database so that other users can view them. However, in the case that they are not allowed to make the details of their experiment public, we would like to restrict access to this view for a set time period (maybe one year after the experiment is created).

5. What are the limitations of your solution? Put another way, given unlimited resources, what could you do to make the project better? (LO_ProbSolutions)

   A major limitation of our solution is the resources available for running our server and backend. The total amount of disk space taken up by all of the data objects that are relevant to our project is about 11 terabytes. This amount of space is not feasible for our server, not to mention the amount of computational resources required to do data processing on all of that data. If this were not an issue, hosting all of the data objects on our server without the need to interact with the FRDR API to access the files would greatly streamline the workflow, likely improve performance of our system and also give us more control over our implementation without the need to conform and work around the specifications of FRDR.

   A secondary limitation of our system is our access to LLMs and other machine learning models. Natural language processing and behavioural cateogorization are important parts of our system. However, we realistically do not have the resources or expertise to develop our own model for these specific purposes and will have to rely on Off-The-Shelf software or external systems for these purposes. We can make the best of this by using system prompts to guide the model in the exact direction we need it to go, add error checking for coherent results or even train the model with our available data if needed. None of these would be quite as effective as developing a model specifically for the intended purpose of natural language input into query output or behavioral categorizations of rats.

   Finally, maybe the biggest limitation for us is our lack of time. The lack of time is twofold. First, we only have about 6 more months to build the project and secondly, the entire team will likely have 5 other courses happening the entire time we are working

on it. It is very common for capstone teams to not build anything close to a finalized project, sometimes it doesn't even work at all really. We envision that we will land somewhere in the middle but the time limit imposed on us almost guarantees that we will not be able to implement the plethora of integrations desired for this system. From natural language input to extensive behavioural categorizations to countless possible visualizations of the datasets and statistical information we could potentially provide that would be useful for future researchers, the chances we are able to implement a complete toolbox of these functionalities is slim to none.

6. Give a brief overview of other design solutions you considered. What are the benefits and tradeoffs of those other designs compared with the chosen design? From all the potential options, why did you select the documented design? (LO_Explores)

   We actually did not consider too many different design options for two reasons. The first is that a previous iteration of this project already exists so our supervisor has a general idea of what he wants it to look like and thus we are expected to mostly conform to the general outline that was devised by him and last year's team. So, they most broad high level conceptual design was already mostly predetermined when the project started. Secondly, when talking about the system architecture, since this will essentially be a full-stack web application with many integrations, it is kind of automatically assumed that an MVC architecture or something that approximates that will be used as that is the bog standard for this sort of project. Of course, there are many more aspects to the design (especially at slightly higher granularity) that had a couple options for us to decide on. Below are a few of possible design alternatives and our reasoning for avoiding them.

   (a) Exclusively using the FRDR API to query the relevant data rather than producing a seperate database schema which then points us to the relevant endpoints in the FRDR domain. This would allow us to avoid needing to create a database and load all 20,000 session details into the schema. This might have been useful if there were an order of magnitude or two more sessions and storing them ourselves would become an issue. In the end, making a schema streamlines the query process and gives us more control over data processing. Furthermore, after investigating the FRDR API, accessing this data would have actually been a lot more difficult than we initially had expected.

   (b) Buying our own personal LLM model (likely would be a mostly blank model to be trained) that could run on our application server rather than using an LLM API (such as the OpenAI API). This would actually likely improve the accuracy of our prompts if we could train it on the relevant trial data. However, the main concerns with this would be the potential lack of data to train it enough to be beneficial (we only have about 20,000 trials, the model may need millions of data points) and that it may be quite expensive to purchase a full model for our personal use. In the end, we figured that using an LLM API with a sufficient system prompt to

guide it's response would be the most cost-effective route that we are confident will provide good enough results for our purposes.

(c) This is more of a subtle design alternative but one of our modules handles error detection and fault tolerance. This will provide the means to track, log and provide details on the various errors that can be raised by the various different modules (such as bad HTTP requests, invalid NLP responses etc...). We heavily considered removing this module and leaving this up to each individual module as we thought it would simplify the implementation (one less module after all). However, in the end we decided that having a unified error management module would it make it easier to track and manage errors across modules as well as ensure they are reported consistently.

(d) Using a different front-end development kit other than React. We considered building the front-end from the ground up using HTML/CSS/JavaScript as we felt it would give us more control over the interface and would be more purely our work rather than relying on components created by other people. However, in the end we decided that using a React component library to build the front-end user interface would both result in a better looking final product (we are not graphic designers and thus a React component library will certainly look better than what we could make) and offloading our effort to a component library will open up more time for us to implement more backend functionality which is what truly drives the value of this project.