

# Synthetic Evaluation Schemas & Prompting

This document contains details of prompting resources used in the Synthetic Evaluation process discussed in Chapter 8.

## Contents

- S8.1 – Schemas (Evaluation)
- S8.2 – Schemas (Data Service)
- S8.3 – Prompt Elements
- S8.4 – Examples
- S8.5 – Processor

## S8.1 Evaluation Schemas

These schemas are used for evaluation actions - i.e. requests to the evaluation LLM and the responses back. These schemas are also available at: [doi:10.21954/ou.rd.28044944](https://doi.org/10.21954/ou.rd.28044944) [path: /eval/schemas].

### S8.1.1 Persona Schema

Listing S8.1 shows the Persona Schema, which is used as part of all evaluation requests. This schema is derived from our work on Persona development, as discussed in Section 5.2.

```
{
  "$id": "https://parse.net/awag/0.1/persona.schema.json",
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "type": "object",
  "title": "Awareness Agent Evaluation Persona",
  "description": "Object representing a request to evaluate one or more items in the role of the specified persona from the supplied perspectives",
  "properties": {
    "name": {
      "title": "Persona Name",
      "description": "The name given to this persona",
      "type": "string"
    },
    "id": {
      "title": "Persona ID",
      "description": "The internal ID for this persona (ignore)",
      "type": "string"
    }
  }
}
```

## S8 Synthetic Evaluation Schemas & Prompting

```

    },
    "definition":{
      "title":"Evaluation Persona Definition",
      "description":"The definition of this persona",
      "type":"object",
      "properties":{
        "type":{
          "title":"Persona Type",
          "description":"Internal identifier of the persona type (ignore)",
          "type":"string"
        },
        "age":{
          "title":"Age",
          "description":"The age of the persona",
          "type":"integer"
        },
        "gender":{
          "title":"Gender",
          "description":"What gender does this persona identify as",
          "type":"string"
        },
        "does":{
          "title":"Does",
          "description":"What does this persona do in their work and personal life",
          "type":"string"
        },
        "feelThinkBelieve":{
          "title":"Feel/Think/Believe",
          "description":"What does this persona feel, think and believe; what is their outlook on
            life, priorities etc",
          "type":"string"
        },
        "technologyExperience":{
          "title":"Technology Experience",
          "description":"What experience does this persona have with technology",
          "type":"string"
        },
        "problems":{
          "title":"Problems",
          "description":"What problems does this persona encounter",
          "type":"string"
        },
        "needs":{
          "title":"Needs",
          "description":"What needs does this persona have",
          "type":"string"
        },
        "existingSolutions":{
          "title":"Existing Solutions",
          "description":"What solutions does this persona currently employ to address their
            problems and meet their needs",
          "type":"string"
        }
      }
    }
  },
  "required":[
    "id",
    "name",
    "definition"
  ]
}

```

Listing S8.1: Evaluation Persona Schema

## S8 Synthetic Evaluation Schemas & Prompting

### S8.1.2 Perspective Schema

Listing S8.2 shows the Perspective Schema, which is used as part of Mode 1 evaluation requests.

```
{
  "$id": "https://parse.net/awag/0.1/perspective.schema.json",
  "$schema": "http://json-schema.org/2020-12/schema#",
  "title": "Awareness Agent Evaluation Perspective",
  "description": "Object representing an evaluation perspective.",
  "type": "object",
  "properties": {
    "perspectiveId": {
      "title": "Perspective ID",
      "description": "A string identifying the perspective",
      "type": "string"
    },
    "perspectiveName": {
      "title": "Perspective Name",
      "description": "The name of the perspective",
      "type": "string"
    },
    "perspectiveText": {
      "title": "Perspective Text",
      "description": "Textual description of the perspective to apply",
      "type": "string"
    }
  },
  "required": [
    "perspectiveId",
    "perspectiveName",
    "perspectiveText"
  ]
}
```

Listing S8.2: Evaluation Perspective Schema

### S8.1.3 Evaluation Request Schema - Mode 1

Listing S8.3 shows the Classification Evaluation Request Schema; this schema is only used with **Mode 1** evaluation requests as discussed in Section 8.2.3.

```
{
  "$id": "https://parse.net/awag/0.1/classification-evaluation-request.schema.json",
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "type": "object",
  "title": "Evaluation Request",
  "description": "Object representing a request to evaluate one or more items in the role of the specified persona from the supplied perspectives",
  "properties": {
    "persona": {
      "title": "Evaluation Persona",
      "description": "The persona to adopt when performing the evaluation",
      "type": "object",
      "properties": {
        "name": {
          "title": "Persona Name",

```

**S8 Synthetic Evaluation Schemas & Prompting**

```

        "description": "The name given to this persona",
        "type": "string"
    },
    "id": {
        "title": "Persona ID",
        "description": "The internal ID for this persona (ignore)",
        "type": "string"
    },
    "definition": {
        "title": "Evaluation Persona Definition",
        "description": "The definition of this persona",
        "type": "object",
        "properties": {
            "type": {
                "title": "Persona Type",
                "description": "Internal identifier of the persona type (ignore)",
                "type": "string"
            },
            "age": {
                "title": "Age",
                "description": "The age of the persona",
                "type": "integer"
            },
            "gender": {
                "title": "Gender",
                "description": "What gender does this persona identify as",
                "type": "string"
            },
            "does": {
                "title": "Does",
                "description": "What does this persona do in their work and personal life",
                "type": "string"
            },
            "feelThinkBelieve": {
                "title": "Feel/Think/Believe",
                "description": "What does this persona feel, think and believe; what is their outlook on life, priorities etc",
                "type": "string"
            },
            "technologyExperience": {
                "title": "Technology Experience",
                "description": "What experience does this persona have with technology",
                "type": "string"
            },
            "problems": {
                "title": "Problems",
                "description": "What problems does this persona encounter",
                "type": "string"
            },
            "needs": {
                "title": "Needs",
                "description": "What needs does this persona have",
                "type": "string"
            },
            "existingSolutions": {
                "title": "Existing Solutions",
                "description": "What solutions does this persona currently employ to address their problems and meet their needs",
                "type": "string"
            }
        }
    }
},
"required": [
    "name", "id"
]

```

**S8 Synthetic Evaluation Schemas & Prompting**

```

    ]
  },
  "perspectives":{
    "title":"Perspectives",
    "description":"An array of perspectives to use when evaluating each item. Each perspective defines a way that the item should be evaluated. For each perspective, the persona should consider the evaluation from the angle defined by the perspective",
    "type":"array",
    "items":[
      {
        "title":"Perspective",
        "description":"An individual perspective form which to consider the evaluation",
        "type":"object",
        "properties":{
          "id":{
            "title":"Perspective ID",
            "description":"A string identifying the perspective",
            "type":"string"
          },
          "name":{
            "title":"Perspective Name",
            "description":"The name of the perspective",
            "type":"string"
          },
          "text":{
            "title":"Perspective Text",
            "description":"Textual description of the perspective to apply",
            "type":"string"
          }
        },
        "required":[
          "id", "name", "text"
        ]
      }
    ]
  },
  "items":{
    "title":"Evaluation Items",
    "description":"An array of items to evaluate. Each of these represents a set of classifications made by the software system that is being evaluated that must be assessed",
    "type":"array",
    "items":[
      {
        "title":"Evaluation Item",
        "description":"An item to evaluate.",
        "type":"object",
        "properties":{
          "itemId":{
            "title":"Item ID",
            "description":"The ID of the item being evaluated",
            "type":"string"
          },
          "sent":{
            "title":"Sent Date Time",
            "description":"The date and time that the item was sent or delivered. This should be used in any time-based perspective evaluations",
            "type":"string",
            "format":"date-time"
          },
          "title":{
            "title":"Item Title",
            "description":"The title or short description of the item being evalated. Use this as part of the evaluation of whether the software system correctly classified the item",
            "type":[

```

## S8 Synthetic Evaluation Schemas & Prompting

```

        "string",
        "null"
    ]
},
"summary":{
    "title":"Item Summary",
    "description":"Summarised text of the item being evaluated. Use this as part of the
        evaluation of whether the software system correctly classified the item",
    "type":"string"
},
"source":{
    "title":"Item Source",
    "description":"Description of where the item came from, which will vary according
        to the type of thing that the item is",
    "type":"object",
    "properties":{
        "type":{
            "title":"Item Type",
            "description":"Short description of the type of the item being evaluated,
                such as a Slack message or RSS item.",
            "type":[
                "string",
                "null"
            ]
        }
    },
    "originator":{
        "title":"Item Originator",
        "description":"Information on who/what the item came from, which will vary by
            item type. This could be the sender's email address, an RSS publisher,
            Slack user etc.",
        "type":[
            "string",
            "null"
        ]
    },
    "channel":{
        "title":"Item Channel ",
        "description":"Information on how/where the item was distributed, which will
            vary by item type. For example, this could be a Slack channel, a mailing
            list name, or an RSS feed etc.",
        "type":[
            "string",
            "null"
        ]
    }
},
"required":[
    "type",
    "originator",
    "channel"
]
},
"classifications":{
    "title":"Classifications",
    "description":"An array of classifications that the software system has applied to
        the item. These are the classifications that should be evaluated for each
        perspective using the supplied persona",
    "type":"array",
    "items":[
        {
            "type":"object",
            "properties":{
                "name":{
                    "title":"Classification Name",
                    "description":"The name of the classification. Do not use this in your

```

**S8 Synthetic Evaluation Schemas & Prompting**

```

        evaluation as it is only an identifier",
        "type":"string"
    },
    "desc":{
        "title":"Classification Description",
        "description":"The description of the classification. Use this text to
            understand what type of classification this is and the criteria for
            selecting different values",
        "type":"string"
    },
    "selected":{
        "title":"Classification Selection",
        "description":"This is the value that you must evaluate. It is the
            classification value that the source system has chosen for the item
            from the list of available classifications. Note that a value of '
            none', 'not' or null is a special case that does not need to be in
            the list of available classifications; these would mean that the
            system decided to classify the item as not being relevant for this
            classification, which should also be taken into consideration in the
            evaluation",
        "type":"string"
    },
    "available":{
        "title":"Available Classifications",
        "description":"The classification values that were available to the
            system to apply to this classification for this item. Use this to
            understand the context of the chosen classification by examining the
            possible alternatives",
        "type":"array",
        "items":[
            {
                "type":"string"
            }
        ]
    },
    "required":[
        "name",
        "selected",
        "available"
    ]
},
    "required":[
        "itemId",
        "sent",
        "title",
        "summary",
        "source",
        "classifications"
    ]
},
    "required":[
        "persona",
        "perspectives",
        "items"
    ]
}

```

Listing S8.3: Classification Evaluation Request Schema (Mode 1)

### S8.1.4 Evaluation Request Schema - Modes 2 & 3

Listing S8.4 shows the Alternative Classification Evaluation Request Schema; **Mode 2** and **Mode 3** evaluation requests conform to this schema, although it is never used directly in the evaluation process.

```
{
  "$id": "https://parse.net/awag/0.1/classification-evaluation-request.alt.schema.json",
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "type": "object",
  "title": "Evaluation Request",
  "description": "Object representing a request to evaluate one or more items in the role of the specified persona from the supplied perspectives",
  "properties": {
    "persona": {
      "title": "Evaluation Persona",
      "description": "The persona to adopt when performing the evaluation",
      "type": "object",
      "properties": {
        "name": {
          "title": "Persona Name",
          "description": "The name given to this persona",
          "type": "string"
        },
        "id": {
          "title": "Persona ID",
          "description": "The internal ID for this persona (ignore)",
          "type": "string"
        },
        "definition": {
          "title": "Evaluation Persona Definition",
          "description": "The definition of this persona",
          "type": "object",
          "properties": {
            "type": {
              "title": "Persona Type",
              "description": "Internal identifier of the persona type (ignore)",
              "type": "string"
            },
            "age": {
              "title": "Age",
              "description": "The age of the persona",
              "type": "integer"
            },
            "gender": {
              "title": "Gender",
              "description": "What gender does this persona identify as",
              "type": "string"
            },
            "does": {
              "title": "Does",
              "description": "What does this persona do in their work and personal life",
              "type": "string"
            },
            "feelThinkBelieve": {
              "title": "Feel/Think/Believe",
              "description": "What does this persona feel, think and believe; what is their outlook on life, priorities etc",
              "type": "string"
            },
            "technologyExperience": {
              "title": "Technology Experience",

```



**S8 Synthetic Evaluation Schemas & Prompting**

```

        "description": "What experience does this persona have with technology",
        "type": "string"
    },
    "problems": {
        "title": "Problems",
        "description": "What problems does this persona encounter",
        "type": "string"
    },
    "needs": {
        "title": "Needs",
        "description": "What needs does this persona have",
        "type": "string"
    },
    "existingSolutions": {
        "title": "Existing Solutions",
        "description": "What solutions does this persona currently employ to address their
            problems and meet their needs",
        "type": "string"
    }
    }
}
},
"required": [
    "id",
    "name",
    "definition"
]
},
"perspective": {
    "title": "Perspective",
    "description": "The perspective defines a way that the item should be evaluated - the persona
        should consider the evaluation from the angle defined by the perspective",
    "type": "string"
},
"items": {
    "title": "Items",
    "description": "An array of items to be evaluated",
    "type": "array",
    "items": {
        "type": "object",
        "properties": {
            "id": {
                "title": "Item Identification",
                "description": "Unique identifiers for the item and classification",
                "type": "object",
                "properties": {
                    "item_id": {
                        "title": "Item ID",
                        "description": "The ID of the item being evaluated",
                        "type": "string"
                    },
                    "perspective_id": {
                        "title": "Perspective ID",
                        "description": "A string identifying the perspective",
                        "type": "string"
                    },
                    "classification_id": {
                        "title": "Classification ID",
                        "description": "Identifier for the specific classification being evaluated",
                        "type": "string"
                    }
                }
            },
            "required": [
                "item_id",
                "perspective_id",

```

**S8 Synthetic Evaluation Schemas & Prompting**

```

        "classification_id"
    ],
    },
    "content":{
        "title":"Content",
        "description":"The content of the item being evaluated",
        "type":"string"
    },
    },
    "classification":{
        "title":"Classification",
        "description":"Details of the classification that was applied to the item",
        "type":"object",
        "properties":{
            "description":{
                "title":"Description",
                "description":"Explanation of the classification's criteria",
                "type":"string"
            },
        },
        "classified_as":{
            "title":"Classified As",
            "description":"This is the value that must be evaluated. It is the
                classification value that the source system has chosen for the item from the
                list of available classifications. Note that a value of 'none', 'not' or
                null is a special case that does not need to be in the list of available
                classifications; these would mean that the system decided to classify the
                item as not being relevant for this classification, which should also be
                taken into consideration in the evaluation",
            "type":"string"
        },
        "from_available_classifications":{
            "title":"Available Classifications",
            "description":"The classification values that were available to the system to
                apply to this classification for this item",
            "type":"array",
            "items":{
                "type":"string"
            }
        }
    },
    },
    "required":[
        "description",
        "classified_as",
        "from_available_classifications"
    ]
}
},
"required":[
    "id",
    "content",
    "classification"
]
}
},
},
"required":[
    "persona",
    "perspective",
    "items"
]
}
}

```

Listing S8.4: Classification Evaluation Request Schema (Mode 2/3)

### S8.1.5 Evaluation Result Schema - Mode 1

Listing S8.5 shows the Classification Evaluation Result Schema for **Mode 1** evaluations. This schema defines the `get_evaluations` function passed to the OpenAI chat completions call using the Function Calling technique<sup>1</sup>.

Note that `classificationName` in the schema is actually what we refer to as the Classification ID.

```
{
  "$id": "https://parse.net/awag/0.1/classification-evaluation-result.model.schema.json",
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "name": "get_evaluations",
  "description": "Return a set of evaluations of the provided input data",
  "parameters": {
    "type": "object",
    "properties": {
      "resultItems": {
        "type": "array",
        "items": {
          "type": "object",
          "title": "Evaluation results",
          "description": "Array containing all of the evaluation results for the provided data",
          "properties": {
            "itemId": {
              "title": "Item ID",
              "description": "The ID of the item being evaluated",
              "type": "string"
            },
            "evaluations": {
              "type": "array",
              "items": {
                "type": "object",
                "properties": {
                  "classificationName": {
                    "title": "Evaluated Classification Name",
                    "description": "The name of the classification that this evaluation is for",
                    "type": "string"
                  },
                  "perspectives": {
                    "title": "Evaluated Perspectives",
                    "description": "Array containing all of the evaluation results for this item, with one entry per perspective evaluated",
                    "type": "array",
                    "items": {
                      "type": "object",
                      "title": "Evaluated Perspective",
                      "description": "The evaluation result for the evaluated classification from a specific perspective",
                      "properties": {
                        "perspectiveId": {
                          "title": "Evaluated Perspective ID",
                          "description": "The ID of the perspective that this evaluation is for",
                          "type": "string"
                        },
                        "evaluatedSelection": {
                          "title": "Evaluated Classification Selection",

```

<sup>1</sup>[https://cookbook.openai.com/examples/how\\_to\\_call\\_functions\\_with\\_chat\\_models](https://cookbook.openai.com/examples/how_to_call_functions_with_chat_models) [<https://perma.cc/FY7T-WUNA>]

Listing S8.5: Classification Evaluation Result Schema - Mode 1

### S8.1.6 Evaluation Result Schema - Mode 2

Listing S8.6 shows the Classification Evaluation Result Schema for **Mode 2** evaluations. This schema defines the Mode 2 version of the `get_evaluations` function.

The Mode 2 result reflects the simpler nature of the Mode 2 and 3 request types compared to Mode 1. While Mode 1 has nested result arrays for the multiple Perspectives and Classifications passed in the request, Modes 2 & 3 results are a flatter array of items only, reflecting that only one Perspective & Classification are evaluated per item in those modes.

```
{
  "$id": "https://parse.net/awag/0.1/classification-evaluation-result.mode2.schema.json",
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "name": "get_evaluations",
  "description": "Return a set of evaluations of the provided input data",
  "parameters": {
    "type": "object",
    "properties": {
      "resultItems": {
        "type": "array",
        "items": {
          "type": "object",
          "title": "Evaluation Result",
          "description": "Object containing the evaluation result for a given item",
          "properties": {
            "id": {
              "type": "object",
              "properties": {
                "itemId": {
                  "type": "string",
                  "description": "The ID of the item being evaluated"
                },
                "perspectiveId": {
                  "type": "string",
                  "description": "The ID of the perspective that this evaluation is for"
                },
                "classificationId": {
                  "type": "string",
                  "description": "The ID of the classification that this evaluation is for"
                }
              }
            },
            "required": [
              "itemId",
              "perspectiveId",
              "classificationId"
            ]
          },
          "evaluatedSelection": {
            "title": "Evaluated Classification Selection",
            "description": "The classification selection chosen by the system, that the evaluation has been applied to",
            "type": "string"
          },
          "evaluationLikert": {
            "type": "integer",
            "minimum": 1,
            "maximum": 5,
            "title": "Evaluation Likert Scale Value",
            "description": "A likert scale ranging from 1 (not at all in agreement with the item"
          }
        }
      }
    }
  }
}
```

**S8 Synthetic Evaluation Schemas & Prompting**

```

        classification) to 5 (absolutely in agreement). Values must only be in the
        range 1-5"
    },
    "evaluationText":{
        "type":"string",
        "title":"Evaluation Text",
        "description":"Textual description elaborating on why the evaluationLikert value
        was chosen for this item"
    }
},
"required":[
    "id",
    "evaluatedSelection",
    "evaluationLikert",
    "evaluationText"
]
}
},
"required":[
    "resultItems"
]
}
}

```

Listing S8.6: Classification Evaluation Result Schema - Mode 2

**S8.1.7 Evaluation Result Schema - Mode 3**

Listing S8.7 shows the Classification Evaluation Result Schema for **Mode 3** evaluations. This schema defines the Mode 3 version of the `get_evaluations` function.

Mode 3 differs from mode 2 only in that it contains a binary `evaluationAgreement` value rather than the `evaluationLikert` rating returned by modes 1 & 2.

```

{
  "$id":"https://parse.net/awag/0.1/classification-evaluation-result.mode3.schema.json",
  "$schema":"https://json-schema.org/draft/2020-12/schema",
  "name":"get_evaluations",
  "description":"Return a set of evaluations of the provided input data",
  "parameters":{
    "type":"object",
    "properties":{
      "resultItems":{
        "type":"array",
        "items":{
          "type":"object",
          "title":"Evaluation Result",
          "description":"Object containing the evaluation result for a given item",
          "properties":{
            "id":{
              "type":"object",
              "properties":{
                "itemId":{
                  "type":"string",
                  "description":"The ID of the item being evaluated"
                },
                "perspectiveId":{
                  "type":"string",

```

**S8 Synthetic Evaluation Schemas & Prompting**

```

        "description": "The ID of the perspective that this evaluation is for"
      },
      "classificationId": {
        "type": "string",
        "description": "The ID of the classification that this evaluation is for"
      }
    },
    "required": [
      "itemId",
      "perspectiveId",
      "classificationId"
    ]
  },
  "evaluatedSelection": {
    "title": "Evaluated Classification Selection",
    "description": "The classification selection chosen by the system, that the evaluation has been applied to",
    "type": "string"
  },
  "evaluationAgreement": {
    "type": "string",
    "enum": [
      "AGREE",
      "DISAGREE"
    ],
    "title": "Evaluation Agreement",
    "description": "Binary evaluation agreement value indicating whether the evaluator agrees or disagrees with the item classification."
  },
  "evaluationText": {
    "type": "string",
    "title": "Evaluation Text",
    "description": "Textual description elaborating on the evaluationAgreement value chosen for this item"
  }
},
"required": [
  "id",
  "evaluatedSelection",
  "evaluationAgreement",
  "evaluationText"
]
}
}
},
"required": [
  "resultItems"
]
}
}

```

Listing S8.7: Classification Evaluation Result Schema - Mode 3

**S8.2 Evaluation Schemas - Data Service**

These schemas are used for communication with awagdata - for example to store/retrieve evaluation results.

**S8 Synthetic Evaluation Schemas & Prompting****S8.2.1 awagdata Record Evaluation Data**

Listing S8.8 shows the Record Evaluation Data Schema, which describes the content of the POST request to awagdata (/data/eval) used to store the results of an evaluation query.

```
{
  "$id": "https://parse.net/awag/0.1/record-evaluation-data.schema.json",
  "$schema": "http://json-schema.org/2020-12/schema#",
  "title": "AwAg Evaluation Data Recording",
  "description": "Contains the results of the evaluation of one or more Content Items for a given
    evaluationPersona. This data is intended to be submitted to the awagdata /eval/record-evaluation-
    data service",
  "type": "object",
  "properties": {
    "agent": {
      "title": "Agent ID",
      "description": "Agent ID for the agent performing the operation",
      "type": "string"
    },
    "evaluationPersona": {
      "$ref": "https://parse.net/awag/0.1/persona.schema.json"
    },
    "evaluationPerspectives": {
      "title": "Evaluation Perspectives",
      "description": "The list of perspectives used for evaluating items",
      "type": "array",
      "items": {
        "$ref": "https://parse.net/awag/0.1/perspective.schema.json"
      }
    },
    "items": {
      "title": "Evaluated Items",
      "description": "An array of completed evaluations for Content Items",
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "itemId": {
            "title": "Content Item ID",
            "type": "string"
          },
          "tags": {
            "title": "Tags",
            "description": "The Output Tags to apply to this Content Item evaluation",
            "type": "array",
            "items": {
              "type": "string"
            }
          }
        }
      },
      "classifications": {
        "title": "Classifications",
        "description": "The classifications that are associated with this Content Item - these are
          what was evaluated",
        "type": "array",
        "items": {
          "type": "object",
          "properties": {
            "title": "Classification Name",
            "description": "The name of the classification",
            "classificationName": {
              "type": "string"
            }
          }
        }
      }
    }
  }
}
```



## S8 Synthetic Evaluation Schemas & Prompting

```

    },
    "classificationValue":{
      "title":"Classification Value",
      "description":"The selected value of the classification that was evaluated",
      "type":"string"
    },
    "classificationOptions":{
      "title":"Classification Options",
      "description":"List of the available values that this classification could have had",
      "type":"array",
      "items":{
        "type":"string"
      }
    }
  },
  "required":[
    "classificationName",
    "classificationValue",
    "classificationOptions"
  ]
}
},
"evaluateSourceType":{
  "title":"Evaluate Item Source Type",
  "description":"The source type for this evaluated item - such as 'RSS Item'",
  "type":"string"
},
"evaluateSourceOriginator":{
  "title":"Evaluate Item Source Originator",
  "description":"The originator of this evaluated item - such as an RSS provider",
  "type":"string"
},
"evaluateSourceChannel":{
  "title":"Evaluate Item Source Channel",
  "description":"The channel or other subdivision that this evaluated item was published to",
  "type":"string"
},
"evaluateTimeText":{
  "title":"Evaluate Time Text",
  "description":"An ISO 8601 compliant textual representation of the datetime of publication of the evaluated item",
  "type":"string"
},
"evaluateTimeMs":{
  "title":"Evaluate Time MS",
  "description":"The datetime of publication of the evaluated item expressed as milliseconds since the epoch",
  "type":"number"
},
"evaluateTitle":{
  "title":"Evaluate Title",
  "description":"The Title property of the evaluated item - this is content used for classification that was part of the evaluation",
  "type":"string"
},
"evaluateText":{
  "title":"Evaluate Text",
  "description":"The Text (body/content) property of the evaluated item - this is content used for classification that was part of the evaluation",
  "type":"string"
},
"evaluationResponses":{
  "title":"Evaluation Responses",
  "description":"The evaluation results for this item, organised by Perspective nested within Classification",

```

## S8 Synthetic Evaluation Schemas & Prompting

```

"type": "array",
"items": {
  "type": "object",
  "properties": {
    "classificationName": {
      "title": "Evaluated Classification Name",
      "description": "The name of the classification",
      "type": "string"
    },
    "perspectives": {
      "type": "array",
      "items": {
        "title": "Evaluated Perspective",
        "description": "The evaluation results by Perspective for this Classification",
        "type": "object",
        "properties": {
          "perspectiveId": {
            "title": "Perspective ID",
            "description": "The ID of the Perspective used for this evaluation",
            "type": "string"
          },
          "evaluationLikertVal": {
            "title": "Evaluation Likert Value",
            "description": "The evaluation result as a Likert value (modes 1 & 2)",
            "minimum": 1,
            "maximum": 5,
            "type": "integer"
          },
          "evaluationLikertText": {
            "title": "Evaluation Likert Text",
            "description": "The textual version of the evaluation Likert value",
            "type": "string"
          },
          "evaluationAgreement": {
            "title": "Evaluation Agreement",
            "description": "The evaluation result as an AGREE/DISAGREE string (mode 3)",
            "enum": [
              "AGREE",
              "DISAGREE"
            ],
            "type": "string"
          },
          "evaluationText": {
            "title": "Evaluation Text",
            "description": "A textual version of the evaluation",
            "type": "string"
          }
        }
      },
      "required": [
        "perspectiveId",
        "evaluationText"
      ],
      "oneOf": [
        {
          "required": [
            "evaluationAgreement"
          ]
        },
        {
          "required": [
            "evaluationLikertVal",
            "evaluationLikertText"
          ]
        }
      ]
    }
  }
}

```

## S8 Synthetic Evaluation Schemas & Prompting

```

        }
      },
      "required": [
        "classificationName",
        "perspectives"
      ]
    }
  },
  "required": [
    "itemId",
    "tags",
    "classifications",
    "evaluateSourceType",
    "evaluateSourceOriginator",
    "evaluateSourceChannel",
    "evaluateTimeText",
    "evaluateTimeMs",
    "evaluateText",
    "evaluationResponses"
  ]
}
},
"context": {
  "title": "Context",
  "description": "Additional contextual information to be recorded alongside the evaluation",
  "type": "object",
  "properties": {
    "contextId": {
      "title": "Context ID",
      "description": "A unique identifier for this evaluation job",
      "type": "string"
    },
    "evaluationItems": {
      "title": "Evaluation Items",
      "description": "An array of the evaluated items in their fuller native format containing full information, provided for general information or to allow the evaluation to be re-run with a differently constructed request",
      "type": "array",
      "items": {
        "type": "object"
      }
    },
    "openaiApiCompletionsQueryInfo": {
      "title": "OpenAI Completions Query Information",
      "description": "Details of the OpenAI Completions API query, including parameters and full prompt information, as well as the response state",
      "type": "object",
      "properties": {
        "maxTokens": {
          "title": "Max Tokens",
          "description": "Parameter: maximum tokens limit for query",
          "type": "integer"
        },
        "model": {
          "title": "Model",
          "description": "Parameter: model to use for query",
          "type": "string"
        },
        "temperature": {
          "title": "Temperature",
          "description": "Parameter: temperature",
          "type": "number"
        }
      }
    }
  }
},

```

**S8 Synthetic Evaluation Schemas & Prompting**

```

    "promptText":{
      "title":"Prompt Text",
      "description":"The full text part of the prompt supplied to the API",
      "type":"string"
    },
    "promptQueryJson":{
      "title":"Prompt Query JSON",
      "description":"The full prompt supplied to the API in JSON format",
      "type":"object"
    },
    "queryState":{
      "title":"Query State",
      "description":"The query state taken from the OpenAI response, including response code,
        actual token usage etc.",
      "type":"object"
    }
  }
},
"required":[
  "contextId",
  "evaluationItems",
  "openaiApiCompletionsQueryInfo"
]
}
},
"required":[
  "agent",
  "evaluationPersona",
  "evaluationPerspectives",
  "items",
  "context"
]
}
}

```

Listing S8.8: Record Evaluation Data Schema

**S8.2.2 awagdata Record Evaluation Failure**

Listing S8.9 shows the Record Evaluation Failure Schema, which describes the content of the POST request to awagdata (/data/eval) used to store the results of a **failed** evaluation query. This contains the response in raw, unparsed text format, in case invalid JSON was returned from the API, and the IDs of Content Items in the evaluation query (bearing in mind that multiple CI's could be in any one query).

```

{
  "$id":"https://parse.net/awag/0.1/record-evaluation-failure.schema.json",
  "$schema":"http://json-schema.org/2020-12/schema#",
  "title":"AwAg Evaluation Failure Recording",
  "description":"Contains the details of an evaluation failure, including the query state, context, and
    relevant persona information.",
  "type":"object",
  "properties":{
    "agent":{
      "title":"Agent ID",
      "type":"string",
      "description":"Agent ID for the agent performing the operation"
    },
  },
}

```

## S8 Synthetic Evaluation Schemas & Prompting

```

"queryState":{
  "title":"Query State",
  "description":"The query state taken from the OpenAI response, including response code, finish
    reason, actual token usage etc.",
  "type":"object"
},
"maxTokens":{
  "title":"Max Tokens",
  "type":"integer"
},
"model":{
  "title":"Model",
  "type":"string",
  "description":"The OpenAI model used for the query"
},
"promptQueryJson":{
  "title":"Prompt Query JSON",
  "description":"The full prompt supplied to the API in JSON format",
  "type":"object"
},
"contextIdentifier":{
  "title":"Context ID",
  "type":"string",
  "description":"Unique Context ID for the query"
},
"evaluationPersona":{
  "$ref":"https://parse.net/awag/0.1/persona.schema.json"
},
"evaluationPerspectives":{
  "title":"Evaluation Perspectives",
  "description":"The list of perspectives used for evaluating items",
  "type":"array",
  "items":{
    "$ref":"https://parse.net/awag/0.1/perspective.schema.json"
  }
},
"responseText":{
  "title":"Response Text",
  "type":"string",
  "description":"The raw response text received back from OpenAI - this may or may not be parseable
    JSON"
},
"items":{
  "title":"Evaluated Item IDs",
  "type":"array",
  "items":{
    "type":"string"
  },
  "description":"List of Content Item IDs involved in the query"
},
"itemCount":{
  "title":"Item Count",
  "type":"integer",
  "description":"Count of Content Items in the query"
},
"statusCode":{
  "title":"Status Code",
  "type":"integer",
  "description":"HTTP status code of the response"
}
},
"required":[
  "agent",
  "queryState",
  "maxTokens",

```

## S8 Synthetic Evaluation Schemas & Prompting

---

```
"model",
"promptQueryJson",
"contextIdentifier",
"evaluationPersona",
"evaluationPerspectives",
"responseText",
"rawReponseText",
"items",
"itemCount",
"statusCode"
]
}
```

Listing S8.9: Record Evaluation Failure Schema

### S8.3 Static Prompt Elements

This section contains the static text elements that make up evaluation prompts for modes 1, 2 & 3. The prompt text is made up of common and mode-specific structures which are assembled into System and User messages for OpenAI as described in Section 8.3.3.

#### S8.3.1 System Message – Common

This **system** message element is used for all modes as the common system message for all Awareness Agent evaluation calls. It is worded in such a way to be compatible with the different modes, by avoiding mode-specific phrasing.

##### **AWAG\_SYSTEM\_MESSAGE\_COMMON**

You are the AwAg Evaluator. Your job is to role play a persona, for the purpose of evaluating a software system.

The software system that you are evaluating is intended to manage a user's incoming information from multiple sources - including work and personal - so that the user is not overwhelmed or distracted. Consider it a personal information triage service. Your evaluation should focus on how well this system classifies the content that it processes and you will be asked to indicate your agreement or disagreement with the decisions.

An AwAg Evaluation Request is presented as a structured JSON document, and its main elements are:

1. Persona - the definition or ID of the persona that you should adopt when evaluating the request;
2. Perspective - the way in which you should consider the items in your evaluation;
3. Items - the actual items that you should evaluate.

## S8 Synthetic Evaluation Schemas & Prompting

---

The exact structure of the AwAg Evaluation Request may vary, but you will be given guidance on how to interpret it. You will also be told how to respond for each request. Possible response types include binary agree/disagree, or using a likert scale from 1 (completely disagree) to 5 (completely agree) as well as text describing your evaluation.

You will be asked to respond with your evaluation(s) of item(s) in a structured way. Your evaluation result should include your own evaluation, and also the value of the evaluated classification value or selection.

### S8.3.2 System Message – Extra for Mode 1

This **system** message is added to Mode 1 requests and is used to reference the Evaluation Request Schema [S8.1.3] that Mode 1 requests conform to.

#### **AWAG\_SYSTEM\_MESSAGE\_EXTRA\_MODE1**

Information about the persona that you should adopt and the items that you must evaluate is provided as a JSON document - called the AwAg Evaluation Request JSON Schema (this schema has ID 'https://parse.net/awag/0.1/classification-evaluation-request.schema.json'). Sometimes you may be given only a subset of this document to evaluate or asked to respond in a different way. Evaluate every item and classification combination, returning a result for each one.

### S8.3.3 System Message – Extra for Modes 2 & 3

This **system** message is added to Mode 2 and Mode 3 requests and is used to describe the request structure in a narrative textual rather than schema-based manner. While it is named 'AWAG\_SYSTEM\_MESSAGE\_EXTRA\_MODE2', the same text is used for both modes 2 and 3 requests.

#### **AWAG\_SYSTEM\_MESSAGE\_EXTRA\_MODE2**

Information about the persona that you should adopt is provided as a JSON document. The 'definition' part of the document tells you about the age and gender of the persona, as well as other information about them. The 'does' property tells you what they do, both for work and social/personal activities. The 'feelThinkBelieve' parameter tells you what the persona feels, thinks and believes, telling you about their motivations and opinions. The 'technologyExperience' property tells you about the experience this persona has with technology. The 'problems' property tells you what sort of problems the persona might encounter, specifically in relation to managing information overload. The 'needs' and 'existingSolutions' properties tell you about the solution needs this persona has, and what existing solutions they have to try and address these.

## S8 Synthetic Evaluation Schemas & Prompting

---

Use the ‘perspective’ element of the request to tell you what approach to take to evaluating the items. For example, the perspective might ask you to determine if an item has been correctly classified.

The items to evaluate are provided to you as an array in the ‘items’ property of the request. Each item contains the following important information: ‘id’ contains the identifiers needed to uniquely identify this item, which you should use in your response for identification only. The ‘content’ property contains the actual textual content of the item that you should evaluate. The ‘classification’ property contains a JSON object that describes how the system that you are evaluating has classified the item - this contains the description of the classification, the available options that the system had to choose from (fromAvailableClassifications), and the option that it selected (classifiedAs). When evaluating an item, you should consider how well the selected classification matches the content of the item, and whether one of the alternative available options would have been a better fit.

### S8.3.4 User Message – Common Likert

This **user** message is added to all evaluation calls that require a Likert type response: specifically Mode 1 and Mode 2.

#### **EVALUATION\_USER\_MESSAGE\_COMMON\_LIKERT**

In this case, your evaluation should take the form of a Likert scale with a value range of 1-5. This describes how much your persona agrees with the classification that you are evaluating. If you strongly agree with how the system classified the item, you should use a likert value of 5. If you strongly disagree with the system’s classification, you should use a likert value of 1. You should use values between this when you can’t be as sure; for example a likert value of 4 indicates that you mostly agree with the classification chosen by the system you are evaluating. On the other hand, a value of 2 indicates that you would probably choose a different classification. If you don’t have enough information to decide either way or are otherwise unsure, you should choose a value of 3. Your evaluation must also include explanatory text written as your persona. It is important that you use the Likert scale consistently and use the full range as needed, and that the scale value you choose is consistent with the text of your evaluation. You must only consider whether the selected classification is the most appropriate of the listed available classifications - do not suggest any alternative classifications that are not listed.

### S8.3.5 User Message Prefix – Mode 1

This is the first **user** message added to all Mode 1 evaluation calls. Because Mode 1 uses Likert responses, it is followed by [EVALUATION\\_USER\\_MESSAGE\\_COMMON\\_LIKERT](#).



## S8 Synthetic Evaluation Schemas & Prompting

---

### **EVALUATION\_USER\_MESSAGE\_MODE1**

Please evaluate the following AwAg Evaluation Request. You should evaluate each item from each specified perspective, and return the results in the required result schema. For each item classification, consider the content, persona and perspectives. Be sure to evaluate the item for every classification and perspective combination.

### **S8.3.6 User Message Prefix – Mode 2**

This is the first **user** message added to all Mode 2 evaluation calls. Because Mode 2 uses Likert responses, it is followed by [EVALUATION\\_USER\\_MESSAGE\\_COMMON\\_LIKERT](#).

### **EVALUATION\_USER\_MESSAGE\_MODE2**

Please evaluate the following AwAg Evaluation Request. You should evaluate each item for the specified persona and perspective, and return the results in the required result schema. Consider how item has been classified in the context of the persona and perspective.

### **S8.3.7 User Message Prefix – Mode 3**

This is the first **user** message added to all Mode 3 evaluation calls (which do not expect a Likert response).

### **EVALUATION\_USER\_MESSAGE\_MODE3**

Please evaluate the following AwAg Evaluation Request. You should evaluate each item for the specified persona and perspective, and return the results in the required result schema. Consider how item has been classified in the context of the persona and perspective.

In this case, rate whether you as your persona AGREE or DISAGREE with the selected evaluation based on the list of available classifications, returning this in the evaluationAgreement property.

### **S8.3.8 User Message Example – Mode 2**

This **user** message gives examples of Mode 2 requests and responses to guide the LLM how to interpret these and construct responses. Only modes 2 and 3 have this type of example provided, with Mode 1 relying on a more schema-based approach.

### **EVALUATION\_USER\_MESSAGE\_EXAMPLE\_MODE2**

Here are some examples to help you understand how to respond. This is an example request:

**S8 Synthetic Evaluation Schemas & Prompting**

---

```
{
  "id":{
    "itemId":"214ea6fa-874d-44ae-a931-6561e69cedff",
    "perspectiveId":"classification",
    "classificationId":"work-pers"
  },
  "content":"2024-04-06T08:49:20.791998\nWho's going to the pub after
work today?",
  "classification":{
    "description":"Does this item relate to my work or personal life?",
    "classifiedAs":"work",
    "fromAvailableClassifications":[
      "work",
      "personal"
    ]
  }
}
```

In this case, the content that the classification is for is: “Who’s going to the pub after work today?”. The classification that this request relates to is “Does this item relate to my work or personal life?”, which has two available classifications: ‘work’ and ‘personal’. The system that you are evaluating has classified this as ‘work’. In this case, even though the content mentions work, the topic is socialising outside of work, so the better classification would be ‘personal’. The appropriate response to this request would therefore look like:

```
{
  "id":{
    "itemId":"214ea6fa-874d-44ae-a931-6561e69cedff",
    "perspectiveId":"classification",
    "classificationId":"work-pers"
  },
  "evaluatedSelection":"work",
  "evaluationLikert":2,
  "evaluationText":"This message is about going to the pub after work,
which mentions work but is generally a personal topic. Therefore,
I mostly disagree with the classification of 'work' - I would have
chosen 'personal' instead on balance. I have awarded an
evaluationLikert value of 2 to indicate disagreement."
}
```

Here is a second example:

```
{
  "id":{
    "itemId":"2e308a00-36b0-44f5-ab47-b2af74e87dd5",
    "perspectiveId":"classification",
    "classificationId":"work-pers"
  },
  "content":"2024-04-06T10:51:32.783943\nCan someone help me with this
```

**S8 Synthetic Evaluation Schemas & Prompting**

---

```

report? I'm really struggling.",
"classification":{
  "description":"Does this item relate to my work or personal life?",
  "classifiedAs":"personal",
  "fromAvailableClassifications":[
    "work",
    "personal"
  ]
}
}

```

This request is for the same work-personal classification, but in this case the content is more clearly work related. However, the classifiedAs that you are evaluating is 'personal' - so the system clearly got it wrong. Therefore the appropriate response is:

```

{
  "id":{
    "itemId":"2e308a00-36b0-44f5-ab47-b2af74e87dd5",
    "perspectiveId":"classification",
    "classificationId":"work-pers"
  },
  "evaluatedSelection":"personal",
  "evaluationLikert":1,
  "evaluationText":"I strongly disagree with the selected classification of 'personal' for this item. The request for help with a report seems very much a work-specific task so I would instead have selected 'work'. I have given an evaluationLikert value of 1 to indicate strong disagreement."
}

```

Consider the same content, but imagine that the system had classified it differently:

```

{
  "id":{
    "itemId":"2e308a00-36b0-44f5-ab47-b2af74e87dd5",
    "perspectiveId":"classification",
    "classificationId":"work-pers"
  },
  "content":"2024-04-06T10:51:32.783943\nCan someone help me with this report? I'm really struggling.",
  "classification":{
    "description":"Does this item relate to my work or personal life?",
    "classifiedAs":"work",
    "fromAvailableClassifications":[
      "work",
      "personal"
    ]
  }
}

```

**S8 Synthetic Evaluation Schemas & Prompting**

---

In this case the system got it right, so your response should be:

```
{
  "id":{
    "itemId":"2e308a00-36b0-44f5-ab47-b2af74e87dd5",
    "perspectiveId":"classification",
    "classificationId":"work-pers"
  },
  "evaluatedSelection":"work",
  "evaluationLikert":5,
  "evaluationText":"I strongly agree with the selected classification of
'work' for this item - the request for help with a report seems very much
a work-specific task. I have given an evaluationLikert value of 5 to
indicate strong agreement."
}
```

**S8.3.9 User Message Example – Mode 3**

This **user** message gives examples of Mode 3 requests and responses to guide the LLM how to interpret and respond. The Mode 3 example text is identical to the Mode 2 version, except that it replaces `evaluationLikert` with `evaluationAgreement` and has accordingly different `evaluationText` responses that do not mention a likert value.

**EVALUATION\_USER\_MESSAGE\_EXAMPLE\_MODE3**

Here are some examples to help you understand how to respond. This is an example request:

```
{
  "id":{
    "itemId":"214ea6fa-874d-44ae-a931-6561e69cedff",
    "perspectiveId":"classification",
    "classificationId":"work-pers"
  },
  "content":"2024-04-06T08:49:20.791998\nWho's going to the pub after
work today?",
  "classification":{
    "description":"Does this item relate to my work or personal life?",
    "classifiedAs":"work",
    "fromAvailableClassifications":[
      "work",
      "personal"
    ]
  }
}
```

In this case, the content that the classification is for is: “Who’s going to the pub after work today?”. The classification that this request relates to is “Does this item relate to my work or personal life?”, which has two available classifications:

**S8 Synthetic Evaluation Schemas & Prompting**

---

‘work’ and ‘personal’. The system that you are evaluating has classified this as ‘work’. In this case, even though the content mentions work, the topic is socialising outside of work, so the better classification would be ‘personal’. The appropriate response to this request would therefore look like:

```
{
  "id":{
    "itemId":"214ea6fa-874d-44ae-a931-6561e69cedff",
    "perspectiveId":"classification",
    "classificationId":"work-pers"
  },
  "evaluatedSelection":"work",
  "evaluationAgreement":"DISAGREE",
  "evaluationText":"This message is about going to the pub after work,
  which mentions work but is generally a personal topic. Therefore, I
  mostly disagree with the classification of 'work' - I would have
  chosen 'personal' instead on balance."
}
```

Here is a second example:

```
{
  "id":{
    "itemId":"2e308a00-36b0-44f5-ab47-b2af74e87dd5",
    "perspectiveId":"classification",
    "classificationId":"work-pers"
  },
  "content":"2024-04-06T10:51:32.783943\nCan someone help me with this
  report? I'm really struggling.",
  "classification":{
    "description":"Does this item relate to my work or personal life?",
    "classifiedAs":"personal",
    "fromAvailableClassifications":[
      "work",
      "personal"
    ]
  }
}
```

This request is for the same work-personal classification, but in this case the content is more clearly work related. However, the classifiedAs that you are evaluating is ‘personal’ - so the system clearly got it wrong. Therefore the appropriate response is:

**S8 Synthetic Evaluation Schemas & Prompting**

---

```
{
  "id":{
    "itemId":"2e308a00-36b0-44f5-ab47-b2af74e87dd5",
    "perspectiveId":"classification",
    "classificationId":"work-pers"
  },
  "evaluatedSelection":"personal",
  "evaluationAgreement":"DISAGREE",
  "evaluationText":"I strongly disagree with the selected classification
of 'personal' for this item. The request for help with a report seems
very much a work-specific task so I would instead have selected 'work'."
}
```

Consider the same content, but imagine that the system had classified it differently:

```
{
  "id":{
    "itemId":"2e308a00-36b0-44f5-ab47-b2af74e87dd5",
    "perspectiveId":"classification",
    "classificationId":"work-pers"
  },
  "content":"2024-04-06T10:51:32.783943\nCan someone help me with this
report? I'm really struggling.",
  "classification":{
    "description":"Does this item relate to my work or personal life?",
    "classifiedAs":"work",
    "fromAvailableClassifications":[
      "work",
      "personal"
    ]
  }
}
```

In this case the system got it right, so your response should be:

```
{
  "id":{
    "itemId":"2e308a00-36b0-44f5-ab47-b2af74e87dd5",
    "perspectiveId":"classification",
    "classificationId":"work-pers"
  },
  "evaluatedSelection":"work",
  "evaluationAgreement":"AGREE",
  "evaluationText":"I strongly agree with the selected classification of
'work' for this item - the request for help with a report seems very much
a work-specific task."
}
```

## S8.4 Evaluation JSON Examples

This section contains example JSON requests and responses modes 1, 2 & 3.

### S8.4.1 Mode 1

Listing S8.10 shows an example **Mode 1** evaluation **request** [S8.1.3].

```
{
  "persona":{
    "definition":{
      "age":49,
      "gender":"Male",
      "does":"I work as a client manager for a global consumer products company. You will have heard of us. My role is essentially to drive sales, but looking after existing customers and bringing new ones in. As many of my accounts are themselves global corporates, I end up doing a fair bit of international travel.\nI never went to university - it didn't interest me at the time - so I started my career at 18 just after A-levels.\nI'm divorced, with one son who lives with his mum but I see often.\nOutside work, I watch football and play golf. I used to do it more the other way around when I was younger.",
      "feelThinkBelieve":"I have a lot to stay on top of at work. I. sometimes miss things, but someone usually reminds me if it's urgent. I prefer not to miss things from important customers though, even if they are not urgent I like to give them a prompt reply. It can be difficult though, when you get things flying at you from all directions at all times. I'm on Facebook, Twitter and all that, like most people are. Not sure how I'd organise my social life without Facebook ( how did I manage before?) I hear all the usual scare stories about social media giants, but they're a bit overblown by the media. I don't overshare, but I'm happy to put information out there - I think you get more out of it that way. Of course in an ideal world I'd be happy to have more control.",
      "technologyExperience":"I don't think I'm held back by not having a degree. A lot of graduates joining my company haven't got a clue about the real business world anyway. Experience counts for a lot and I've always done well for getting stuck in with new things. I really believe that you get out what you're prepared to put in with technology. I'll put in the time when I can to get things set up right, tell it my preferences or whatever helps. In mainly use office tools like Word and Excel and communications tools such as Webex day to day. We also have a new CRM system that I have to use, as well as the old CRM system that we haven't completely migrated off yet, and the other CRM system that we also use, for reasons I don't fully understand. I travel a lot, so I'm pretty good at managing with a notebook and mobile.",
      "problems":"Has to use multiple different systems that are not integrated with each other. Receives many interruptions from different sources that need to be prioritised differently.",
      "needs":"Methods to handle and prioritise incoming messages.",
      "existingSolutions":"Email rules. Social media functionality. Features of bespoke applications that control notifications.",
      "type":"PathyPersona"
    },
    "id":"kenton",
    "name":"Kenton"
  },
  "perspectives":[
    {
      "id":"classification",
      "name":"Classification",
      "text":"Has this item been correctly classified - put into the appropriate channel?"
    }
  ],
  "items":[
    {
      "itemId":"8MwrDYJrtpQJJAYNGR4T_1",
      "sent":1710777356.1113026,

```

## S8 Synthetic Evaluation Schemas & Prompting

```

    "title": "",
    "summary": "Please can you get back to me as soon as possible about the customer meeting this
        afternoon",
    "source": {
        "type": "Simulated Item",
        "originator": "David Williams <davidwilliams78>",
        "channel": "t"
    },
    "classifications": [
        {
            "name": "work-pers",
            "desc": "Does this item relate to my work or personal life?",
            "available": [
                "work",
                "personal"
            ],
            "selected": "personal"
        },
        {
            "name": "urgency",
            "desc": "Is this item in relation to something that I think is urgent?",
            "available": [
                "urgent",
                "not"
            ],
            "selected": "not"
        },
        {
            "name": "work-logistics",
            "desc": "Is this item related to logistics at work?",
            "available": [
                "logistics",
                "not"
            ],
            "selected": "logistics"
        },
        {
            "name": "football",
            "desc": "Is this item related to my hobby/interest of Football?",
            "available": [
                "football",
                "not"
            ],
            "selected": "football"
        },
        {
            "name": "interested",
            "desc": "Am I interested in this item, from a work or a personal perspective?",
            "available": [
                "work",
                "personal",
                "not"
            ],
            "selected": "not"
        }
    ]
}

```

Listing S8.10: Example Evaluation Request - Mode 1



**S8 Synthetic Evaluation Schemas & Prompting**

Listing S8.11 shows an example **Mode 1** evaluation **response** [S8.1.5].

```
[
  {
    "itemId": "8MwrDYJrtpQJJAYNGR4T_1",
    "evaluations": [
      {
        "classificationName": "work-pers",
        "perspectives": [
          {
            "perspectiveId": "classification",
            "evaluatedSelection": "personal",
            "evaluationLikert": 1,
            "evaluationText": "I strongly disagree with the classification of 'personal' for this item. The content indicates that it is work-related, as it mentions a customer meeting this afternoon. Therefore, the most appropriate classification is 'work'."
          }
        ]
      },
      {
        "classificationName": "urgency",
        "perspectives": [
          {
            "perspectiveId": "classification",
            "evaluatedSelection": "not",
            "evaluationLikert": 1,
            "evaluationText": "I strongly disagree with the classification of 'not' for this item. The content suggests that the item is related to a customer meeting this afternoon, which indicates urgency. Therefore, the most appropriate classification is 'urgent'."
          }
        ]
      },
      {
        "classificationName": "work-logistics",
        "perspectives": [
          {
            "perspectiveId": "classification",
            "evaluatedSelection": "logistics",
            "evaluationLikert": 1,
            "evaluationText": "I strongly disagree with the classification of 'logistics' for this item. The content does not indicate any logistical information but rather refers to a customer meeting. Therefore, the most appropriate classification is 'not'."
          }
        ]
      },
      {
        "classificationName": "football",
        "perspectives": [
          {
            "perspectiveId": "classification",
            "evaluatedSelection": "football",
            "evaluationLikert": 1,
            "evaluationText": "I strongly disagree with the classification of 'football' for this item. The content does not mention anything related to football, so the most appropriate classification is 'not'."
          }
        ]
      },
      {
        "classificationName": "interested",
        "perspectives": [
          {
            "perspectiveId": "classification",
            "evaluatedSelection": "not",

```

## S8 Synthetic Evaluation Schemas &amp; Prompting

```

        "evaluationLikert":5,
        "evaluationText":"I strongly agree with the classification of 'not' for this item. The
            content indicates that the item is neither related to work nor of personal interest, so '
            not' is the most appropriate classification."
    }
}
}
}
}
]

```

Listing S8.11: Example Evaluation Response - Mode 1

## S8.4.2 Mode 2

Listing S8.12 shows an example **Mode 2** evaluation **request** [S8.1.4].

```

{
  "persona":{
    "definition":{
      "age":49,
      "gender":"Male",
      "does":"I work as a client manager for a global consumer products company. You will have heard of
        us. My role is essentially to drive sales, but looking after existing customers and bringing
        new ones in. As many of my accounts are themselves global corporates, I end up doing a fair bit
        of international travel.\nI never went to university - it didn't interest me at the time - so
        I started my career at 18 just after A-levels.\nI'm divorced, with one son who lives with his
        mum but I see often.\nOutside work, I watch football and play golf. I used to do it more the
        other way around when I was younger.",
      "feelThinkBelieve":"I have a lot to stay on top of at work. I. sometimes miss things, but someone
        usually reminds me if it's urgent. I prefer not to miss things from important customers though,
        even if they are not urgent I like to give them a prompt reply. It can be difficult though,
        when you get things flying at you from all directions at all times. I'm on Facebook, Twitter
        and all that, like most people are. Not sure how I'd organise my social life without Facebook (
        how did I manage before?) I hear all the usual scare stories about social media giants, but
        they're a bit overblown by the media. I don't overshare, but I'm happy to put information out
        there - I think you get more out of it that way. Of course in an ideal world I'd be happy to
        have more control.",
      "technologyExperience":"I don't think I'm held back by not having a degree. A lot of graduates
        joining my company haven't got a clue about the real business world anyway. Experience counts
        for a lot and I've always done well for getting stuck in with new things. I really believe that
        you get out what you're prepared to put in with technology. I'll put in the time when I can to
        get things set up right, tell it my preferences or whatever helps. In mainly use office tools
        like Word and Excel and communications tools such as Webex day to day. We also have a new CRM
        system that I have to use, as well as the old CRM system that we haven't completely migrated
        off yet, and the other CRM system that we also use, for reasons I don't fully understand. I
        travel a lot, so I'm pretty good at managing with a notebook and mobile.",
      "problems":"Has to use multiple different systems that are not integrated with each other.
        Receives many interruptions from different sources that need to be prioritised differently.",
      "needs":"Methods to handle and prioritise incoming messages.",
      "existingSolutions":"Email rules. Social media functionality. Features of bespoke applications
        that control notifications.",
      "type":"PathyPersona"
    },
    "id":"kenton",
    "name":"Kenton"
  },
  "perspective":"Has this item been correctly classified?",
  "items":[
    {

```

**S8 Synthetic Evaluation Schemas & Prompting**

```

    "id":{
      "item_id":"hJeWF47TAdKfMtGSHF7T_1",
      "perspective_id":"classification",
      "classification_id":"work-pers"
    },
    "content":"1710777356.1113026\nPlease can you get back to me as soon as possible about the
      customer meeting this afternoon",
    "classification":{
      "description":"Does this item relate to my work or personal life?",
      "classified_as":"work",
      "from_available_classifications":[
        "work",
        "personal"
      ]
    }
  },
  {
    "id":{
      "item_id":"hJeWF47TAdKfMtGSHF7T_1",
      "perspective_id":"classification",
      "classification_id":"urgency"
    },
    "content":"1710777356.1113026\nPlease can you get back to me as soon as possible about the
      customer meeting this afternoon",
    "classification":{
      "description":"Is this item in relation to something that I think is urgent?",
      "classified_as":"urgent",
      "from_available_classifications":[
        "urgent",
        "not"
      ]
    }
  }
]
}

```

Listing S8.12: Example Evaluation Request - Mode 2

Listing S8.13 shows an example **Mode 2** evaluation **response** [S8.1.6].

```

[
  {
    "id":{
      "itemId":"lpULZ8bF7Bf5QfgivmMo_1",
      "perspectiveId":"classification",
      "classificationId":"work-pers"
    },
    "evaluatedSelection":"work",
    "evaluationAgreement":"AGREE",
    "evaluationText":"This item is related to work, so I agree with the selected classification of 'work'
      for this content."
  },
  {
    "id":{
      "itemId":"lpULZ8bF7Bf5QfgivmMo_1",
      "perspectiveId":"classification",
      "classificationId":"urgency"
    },
    "evaluatedSelection":"urgent",
    "evaluationAgreement":"AGREE",
    "evaluationText":"This item is related to something that I would think is urgent, so I agree with
      the selected classification of 'urgent' for this content."
  }
]

```

1

## Listing S8.13: Example Evaluation Response - Mode 2

## S8.4.3 Mode 3

Listing S8.14 shows an example **Mode 3** evaluation **request** [S8.1.4].

```
{
  "persona":{
    "definition":{
      "age":49,
      "gender":"Male",
      "does":"I work as a client manager for a global consumer products company. You will have heard of us. My role is essentially to drive sales, but looking after existing customers and bringing new ones in. As many of my accounts are themselves global corporates, I end up doing a fair bit of international travel.\nI never went to university - it didn't interest me at the time - so I started my career at 18 just after A-levels.\nI'm divorced, with one son who lives with his mum but I see often.\nOutside work, I watch football and play golf. I used to do it more the other way around when I was younger.",
      "feelThinkBelieve":"I have a lot to stay on top of at work. I. sometimes miss things, but someone usually reminds me if it's urgent. I prefer not to miss things from important customers though, even if they are not urgent I like to give them a prompt reply. It can be difficult though, when you get things flying at you from all directions at all times. I'm on Facebook, Twitter and all that, like most people are. Not sure how I'd organise my social life without Facebook ( how did I manage before?) I hear all the usual scare stories about social media giants, but they're a bit overblown by the media. I don't overshare, but I'm happy to put information out there - I think you get more out of it that way. Of course in an ideal world I'd be happy to have more control.",
      "technologyExperience":"I don't think I'm held back by not having a degree. A lot of graduates joining my company haven't got a clue about the real business world anyway. Experience counts for a lot and I've always done well for getting stuck in with new things. I really believe that you get out what you're prepared to put in with technology. I'll put in the time when I can to get things set up right, tell it my preferences or whatever helps. In mainly use office tools like Word and Excel and communications tools such as Webex day to day. We also have a new CRM system that I have to use, as well as the old CRM system that we haven't completely migrated off yet, and the other CRM system that we also use, for reasons I don't fully understand. I travel a lot, so I'm pretty good at managing with a notebook and mobile.",
      "problems":"Has to use multiple different systems that are not integrated with each other. Receives many interruptions from different sources that need to be prioritised differently.",
      "needs":"Methods to handle and prioritise incoming messages.",
      "existingSolutions":"Email rules. Social media functionality. Features of bespoke applications that control notifications.",
      "type":"PathyPersona"
    },
    "id":"kenton",
    "name":"Kenton"
  },
  "perspective":"Has this item been correctly classified?",
  "items":[
    {
      "id":{
        "item_id":"lpULZ8bF7Bf5QfgivmMo_1",
        "perspective_id":"classification",
        "classification_id":"work-pers"
      },
      "content":"1710777356.1113026\nPlease can you get back to me as soon as possible about the customer meeting this afternoon",
      "classification":{
        "description":"Does this item relate to my work or personal life?",

```

## S8 Synthetic Evaluation Schemas & Prompting

```

        "classified_as": "work",
        "from_available_classifications": [
            "work",
            "personal"
        ]
    },
    {
        "id": {
            "item_id": "lpULZ8bF7Bf5QfgivmMo_1",
            "perspective_id": "classification",
            "classification_id": "urgency"
        },
        "content": "1710777356.1113026\nPlease can you get back to me as soon as possible about the customer meeting this afternoon",
        "classification": {
            "description": "Is this item in relation to something that I think is urgent?",
            "classified_as": "urgent",
            "from_available_classifications": [
                "urgent",
                "not"
            ]
        }
    }
]

```

Listing S8.14: Example Evaluation Request - Mode 3

Listing S8.15 shows an example **Mode 3** evaluation **response** [S8.1.7].

```

[
  {
    "id": {
      "itemId": "lpULZ8bF7Bf5QfgivmMo_1",
      "perspectiveId": "classification",
      "classificationId": "work-pers"
    },
    "evaluatedSelection": "work",
    "evaluationAgreement": "AGREE",
    "evaluationText": "This item is related to work, so I agree with the selected classification of 'work' for this content."
  },
  {
    "id": {
      "itemId": "lpULZ8bF7Bf5QfgivmMo_1",
      "perspectiveId": "classification",
      "classificationId": "urgency"
    },
    "evaluatedSelection": "urgent",
    "evaluationAgreement": "AGREE",
    "evaluationText": "This item is related to something that I would think is urgent, so I agree with the selected classification of 'urgent' for this content."
  }
]

```

Listing S8.15: Example Evaluation Response - Mode 3

## S8.5 Evaluation Processor

### S8.5.1 Evaluation Processor on Python

Listing S8.16 shows slightly abridged Python code of the awagdata implementation of the Evaluation Processor. This code, which is intended to be run within a Flask environment, runs evaluations individually or in batch using evaluation items stored in awagdata at runtime by the Awareness Agent.

This functionality was originally implemented within the Java Awareness Agent application, and was later ported to Python to give an alternative batch processing environment with more flexibility. One consequence of this is that the Python code is relatively complex as it attempts to replicate structures and processes that exist in the Java original.

```
import os, json, time, random, threading

from datetime import datetime
from dateutil import parser
from enum import Enum
from zoneinfo import ZoneInfo

from flask import current_app

from domestique.datetime import get_current_time_ms
from domestique.validation import Validator, NoiseLevel
from domestique.json import get_dict_from_dict_or_json_str
from domestique.text import truncate_string
from domestique.identifiers import generate_id
from domestique.logging import log_exception

from .shared_resources import logger, get_dataset_namespace, get_dataset_meta_namespace,
    get_dataset_namespace_base_for_type
from .shared_resources import validate_persona, validate_subset_percent, get_likert_label
from .shared_resources import validate_model_evaluation_result, validate_mode2_mode3_evaluation_result

from .awag_evaluation_request_generator import AwAgEvaluationRequestGenerator

class AwAgEvaluationProcessor:

    def __init__(self,
        client_id,
        openai_client,
        evaluation_system_message_common,
        evaluation_system_message_extra,
        evaluation_result_schema,
        evaluation_user_messages,
        mode="model1",
        evaluation_request_schema=None,
        default_model=None,
        awagdata_client=None,
        flask_app=None):

        if mode == "model1":
            if not evaluation_request_schema:
                raise ValueError("evaluation_request_schema is required for model1")
            logger.debug("Init for model1")
        elif mode == "model2":
```

**S8 Synthetic Evaluation Schemas & Prompting**

```

        logger.debug("Init for mode2")
    elif mode == "mode3":
        logger.debug("Init for mode3")
    else:
        raise ValueError(f"Invalid mode: {mode}")

    self.client_id = client_id
    self.mode = mode
    self.openai_client = openai_client
    self.awagdata_client = awagdata_client
    self.evaluation_system_message_common = evaluation_system_message_common
    self.evaluation_system_message_extra = evaluation_system_message_extra
    self.evaluation_request_schema = evaluation_request_schema
    self.evaluation_result_schema = evaluation_result_schema
    self.evaluation_user_messages = evaluation_user_messages
    self.default_model = default_model
    self.flask_app = flask_app

    self.job_statuses = {}

def get_namespace_base(self):

    return self.namespace_base

def get_objectstore_client(self):

    return self.objectstore_client

def get_openai_client_wrapper(self):

    return self.openai_client_wrapper

def _perform_evaluation(self, messages, model, tools, tool_choice, openai_params={}):

    get_evaluations_response, info_json = self.openai_client.run_chat_completions(
        messages=messages,
        model=model,
        tools=tools,
        tool_choice=tool_choice,
        openai_params=openai_params)

    resp_message = get_evaluations_response.message
    tool_calls = resp_message.tool_calls

    finish_reason = get_evaluations_response.finish_reason

    if not tool_calls:
        ex_msg = f"Did not get tool_calls from OpenAI API response: {get_evaluations_response}"
        logger.error(ex_msg)
        logger.error(f"info_json: {info_json}")
        raise Exception(ex_msg)

    function = tool_calls[0].function
    evaluations_raw = function.arguments
    evaluations_obj = get_dict_from_dict_or_json_str(evaluations_raw)
    result_items = evaluations_obj.get("resultItems")
    logger.debug(f"Got result_items: {result_items}")

    info_json["additional_query_info"] = {
        "tools": tools,
        "messages": messages,

```

**S8 Synthetic Evaluation Schemas & Prompting**

```

        "finish_reason": finish_reason
    }

    return result_items, info_json

def get_evaluation(self, eval_request, model=None, openai_params={}):

    if model is None:
        model=self.default_model

    try:

        evaluation_system_message = f"{self.evaluation_system_message_common}{self.
            evaluation_system_message_extra}"

        messages = self.openai_client.append_system_message(evaluation_system_message)
        evaluation_user_messages = self.evaluation_user_messages.copy()
        for evaluation_user_message in evaluation_user_messages:
            if evaluation_user_message:
                self.openai_client.append_user_message(evaluation_user_message, messages)
        self.openai_client.append_user_message(f"{eval_request}", messages)

        tools = [self.openai_client.get_tools_json_from_function_schema(self.
            evaluation_result_schema)]
        tool_choice = self.openai_client.get_tool_choice_json("get_evaluations")

        result_items, info_json = self._perform_evaluation(messages, model, tools, tool_choice,
            openai_params)

        return result_items, info_json

    except Exception as err:

        logger.error(f"Got Exception in get_evaluation: {err}")
        logger.error(f"Was processing eval_request: {eval_request}")
        return None, None

def run_evaluations(self, tag_source, tags_dest, openai_model, persona, perspectives,
    items_to_process, openai_params={}, is_exclude_existing=True, subset_tag=None, subset_percent=
    None, is_most_recent=False, last_n_hours=None, by_uuid=None, is_async=False, is_dry_run=False):
    """
    This code is a replication of evaluation processing code form the AwAg Java application;
    unfortunately this means it is quite complex. The Java version started with the original
    model implementation, which is able to create a compound evaluation request that combines
    multiple perspectives, classifications and items into one OpenAI query (with the original
    intent being efficiency of prompt tokens). Modes 2 and 3 followed, to present OpenAI with
    a more cleanly understood query structure with each item being presented in a more flat
    manner - also using different names for the benefit of OpenAI's processing. As a result we
    have to take stored evaluation items of a specific structure and restructure them according
    to mode. Additionally, the features that record evaluation results are different from the
    original Java, because not all information is available in the same format. On the other
    hand, by porting the functionality to this Python app, we are able to add more information,
    better track jobs and make easier test changes.
    """

    mode = self.mode

    if subset_tag is not None:
        if subset_percent is None:
            raise ValueError("Missing subset_percent parameter")
        subset_percent = validate_subset_percent(subset_percent)

```



**S8 Synthetic Evaluation Schemas & Prompting**

```

persona_id, persona_name = validate_persona(persona)

job_id = generate_id()

# Only model conforms to the Evaluation Request Schema
if mode == "model":
    logger.info(f"Running evaluations in {mode} - will use Evaluation Request Schema in OpenAI calls")
    is_use_request_schema=True
else:
    logger.info(f"Running evaluations in {mode} - will NOT use Evaluation Request Schema in OpenAI calls")
    is_use_request_schema=False

if not isinstance(tags_dest, list) or not tags_dest:
    raise ValueError("tags_dest must be a non-empty list")

if is_exclude_existing:
    # Exclude existing based on the FIRST item in tags_dest
    exclude_tag = tags_dest[0]
else:
    exclude_tag = None

job_status = {
    "job_id": job_id,
    "status": "INITIATED",
    "message": f"Initialised processing for '{persona_name}' with job_id: {job_id}",
    "processed_items": 0,
    "remaining": -1,
    "init_remaining": -1,
    "items_to_process": items_to_process,
    "usage": {
        "completion_tokens": 0,
        "prompt_tokens": 0,
        "total_tokens": 0
    },
    "mode": mode,
    "tag_source": tag_source,
    "tags_dest": tags_dest,
    "is_exclude_existing": is_exclude_existing,
    "exclude_tag": exclude_tag,
    "openai_model": openai_model,
    "eval_info": [],
    "persona": persona,
    "subset_tag": subset_tag,
    "subset_percent": subset_percent,
    "is_most_recent": is_most_recent,
    "last_n_hours": last_n_hours,
    "persona_id": persona_id,
    "persona_name": persona_name,
    "is_async": is_async,
    "is_error": False,
    "evaluations": []
}

if is_dry_run:
    job_status["is_dry_run"] = is_dry_run
    job_status["evaluation_success_data"] = []
    job_status["evaluation_failure_data"] = []
    job_status["eval_info_base"] = None
self.job_statuses[job_id] = job_status

def get_resp_perspective_from_far(formatted_evaluation_response):

    return {
        "perspectiveId": formatted_evaluation_response["perspectiveId"],

```

**S8 Synthetic Evaluation Schemas & Prompting**

```

        "mode": self.mode,
        "evaluatedSelection": formatted_evaluation_response["evaluatedSelection"],
        "evaluationText": formatted_evaluation_response["evaluationText"]
    }

def get_eval_info_base(mode, generated_request):

    # eval_info exists to generate an easily human readable summary
    # for each evaluation - this has been added for the Python version

    eval_info_base = {
        "mode": mode,
        "persona": persona["id"],
        "evaluate_content": {},
        "evaluate_classifications": {}
    }

    for item in generated_request["items"]:

        if mode == "model":
            item_id = item["itemId"]
            evaluate_content = item["sent"]
            if item["title"]:
                evaluate_content += "\n" + item["title"]
            if item["summary"]:
                evaluate_content += "\n" + item["summary"]
            eval_info_base["evaluate_content"][item_id] = evaluate_content
            for evaluate_classification in item["classifications"]:
                classification_id = evaluate_classification["name"]
                if not item_id in eval_info_base["evaluate_classifications"]:
                    eval_info_base["evaluate_classifications"][item_id] = {}
                eval_info_base["evaluate_classifications"][item_id][classification_id] = {
                    "classification_id": classification_id,
                    "classification_desc": evaluate_classification["desc"],
                    "classification_options": evaluate_classification["available"],
                    "classification_value": evaluate_classification["selected"]
                }

        else:
            item_id = item["id"]["itemId"]
            classification_id = item["id"]["classificationId"]
            # Expect eval_info_base["evaluate_content"] to be written to multiple times,
            # but content will actually be same for each item due to mode2/3 structure
            eval_info_base["evaluate_content"][item_id] = item["content"]
            classification_contents = item["classification"]
            if not item_id in eval_info_base["evaluate_classifications"]:
                eval_info_base["evaluate_classifications"][item_id] = {}
            eval_info_base["evaluate_classifications"][item_id][classification_id] = {
                "classification_id": classification_id,
                "classification_desc": classification_contents["description"],
                "classification_options": classification_contents["fromAvailableClassifications"],
                "classification_value": classification_contents["classifiedAs"]
            }

    return eval_info_base

def get_eval_info(eval_info_base, evaluation_result, classification_id=None):

    mode = eval_info_base["mode"]

    try:

        if mode == "model":

```

## S8 Synthetic Evaluation Schemas &amp; Prompting

```

    if not classification_id:
        raise ValueError("get_eval_info requires classification_id for mode1")
    item_id = evaluation_result["itemId"]
    for evaluation in evaluation_result["evaluations"]:
        if classification_id == evaluation["classificationName"]:
            first_perspective = evaluation["perspectives"][0] # Only supporting first!
            evaluation_text = first_perspective["evaluationText"]
            evaluation_result = first_perspective["evaluationLikert"]
            break

    else:

        if classification_id is not None:
            # classification_id not needed due to flat structure
            # of mode2/3 evaluation_result
            logger.warn(f"get_eval_info will ignore classification_id for mode: {mode}")

            ids = evaluation_result["ids"]
            item_id = ids["itemId"]
            classification_id = ids["classificationId"]
            evaluation_text = evaluation_result["evaluationText"]
            if mode == "mode2":
                evaluation_result = evaluation_result["evaluationLikert"]
            else:
                evaluation_result = evaluation_result["evaluationAgreement"]

        if not item_id in eval_info_base["evaluate_classifications"]:
            logger.warn(f"eval_info_base does not contain item with id '{item_id}': {eval_info_base}")
            eval_info_base["evaluate_classifications"][item_id] = {}
        if not item_id in eval_info_base["evaluate_classifications"]:
            logger.warn(f"eval_info_base does not contain classification with id '{classification_id}': {eval_info_base}")
        classification = eval_info_base["evaluate_classifications"][item_id].get(classification_id)
        classification_text = f"{classification['classification_value'].upper()} - {classification['classification_id']}"

        eval_info = {
            "item": f"{item_id} - {mode} - {eval_info_base['persona']}",
            "content": eval_info_base["evaluate_content"][item_id],
            "classification": classification_text,
            "evaluationText": evaluation_text,
            "evaluationResult": evaluation_result
        }

    return eval_info

except Exception as err:

    logger.error(f"Got Exception in get_eval_info: {err}")
    logger.error(f"Was processing evaluation_result: {evaluation_result}")
    return None

def process_evaluations():

    nonlocal job_status

    with self.flask_app.app_context():

        try:

            time.sleep(0.1) # Allow short time for calling function to return initial job_status
            job_status["status"] = "PROCESSING"

```

## S8 Synthetic Evaluation Schemas & Prompting

```

request_generator = AwAgEvaluationRequestGenerator(flask_app=self.flask_app)

page = 1
count = 20

limit_reached = False

while True:

    if not is_dry_run:
        record_evaluation_job_response = self.awagdata_client.record_evaluation_job(
            job_status)

    if limit_reached:
        logger.debug("Limit reached")
        break

    data, remaining = self.awagdata_client.fetch_evaluation_items(tag_source,
        exclude_tag, last_n_hours, subset_tag, subset_percent, is_most_recent,
        by_uuid, page, count)

    if job_status["init_remaining"] < 0:
        job_status["init_remaining"] = remaining
        job_status["remaining"] = remaining

    if not data:
        if page == 1:
            logger.warn(f"No data to process for tag: {tag_source}")
            break
        else:
            logger.debug(f"No more data at page {page} (count: {count})")
            break

    for evaluation_item in data:

        generated_requests = request_generator.
            generate_requests_from_evaluation_items([evaluation_item], persona,
            perspectives, mode)

        if not is_dry_run:
            record_evaluation_job_response = self.awagdata_client.
                record_evaluation_job(job_status)

        for generated_request in generated_requests:

            evaluation_results, info_json = self.get_evaluation(
                eval_request=generated_request,
                model=openai_model,
                openai_params=openai_params)

            if evaluation_results == None:
                # This should be rare. We don't have enough information here
                # to log the failure so just skip it so we can continue
                is_error = True
                logger.debug(f"Unable to process evaluation - skipping")
                break

            # Put cut-down version of info_json in the job_entry
            info_for_job_entry = {
                "query_info": info_json["query_info"],
                "query_state": info_json["query_state"],
                "usage": info_json["usage"]
            }

```

**S8 Synthetic Evaluation Schemas & Prompting**

```

eval_info_base = get_eval_info_base(mode, generated_request)
if is_dry_run:
    job_status["eval_info_base"] = eval_info_base

request_items = generated_request["items"]

job_entry = {
    "request_items": request_items,
    "info": info_for_job_entry,
}

usage = info_json["usage"]
job_status["usage"]["completion_tokens"] += usage["completion_tokens"]
job_status["usage"]["prompt_tokens"] += usage["prompt_tokens"]
job_status["usage"]["total_tokens"] += usage["total_tokens"]

# query_info is similar but not identical to
#   APIChatCompletionsQueryInfoEvaluate from Java
query_info = info_json["query_info"]
query_info["mode"] = mode
query_info["eval_request"] = generated_request

# Put the engine value into query_info so it is sent to the recorder.
# We do it this way for compatability with existing code but it's not
# the best place for it
query_info["engine"] = info_json["engine"]

query_state = info_json["query_state"]
additional_query_info = info_json["additional_query_info"]

finish_reason = additional_query_info["finish_reason"]
if finish_reason == "stop" or finish_reason == function_call:
    logger.debug(f"Got non-error finish_reason: {finish_reason}")
    is_error = False
else:
    logger.error(f"Got error finish_reason: {finish_reason}")
    is_error = True
job_status["finish_reason"] = finish_reason

# Ref IEvaluationResponse from Java
formatted_evaluation_responses = []

job_status["is_error"] = is_error

if not is_error:
    for evaluation_result in evaluation_results:
        if mode == "model":
            is_evaluation_result_valid =
                validate_model_evaluation_result(evaluation_result)
            if not is_evaluation_result_valid:
                logger.error(f"Got bad evaluation_result: {
                    evaluation_result}")
                is_error = True
                continue;

            item_id = evaluation_result["itemId"]

            for evaluation in evaluation_result["evaluations"]:
                logger.debug(f"Processing {mode} evaluation: {evaluation
                    }")

```

**S8 Synthetic Evaluation Schemas & Prompting**

```

# EvaluationResponseModel implements IEvaluationResponse
evaluation_reponse_model = {
    "itemId": item_id, # Not actually in
                        # EvaluationResponseModel!
    "classificationName": evaluation["classificationName"],
    "perspectives": evaluation["perspectives"]
}

formatted_evaluation_responses.append(
    evaluation_reponse_model)

eval_info = get_eval_info(eval_info_base,
    evaluation_result, classification_id=evaluation["
classificationName"])

if eval_info is None:
    # Ditch this whole evaluation_results set as None
    # eval_info indicates bad response from OpenAI
    is_error = True
    formatted_evaluation_responses = []
    break
else:
    formatted_evaluation_responses.append(
        evaluation_reponse_model)
    job_status["eval_info"].append(eval_info)

else:

    is_evaluation_result_valid =
        validate_mode2_mode3_evaluation_result(evaluation_result,
            mode)
    if not is_evaluation_result_valid:
        logger.error(f"Got bad evaluation_result: {
            evaluation_result}")
        is_error = True
        continue;

    logger.debug(f"Processing {mode} evaluation: {
        evaluation_result}")

    ids = evaluation_result["id"]
    item_id = ids["itemId"]

    evaluation_reponse = {
        "itemId": item_id,
        "perspectiveId": ids["perspectiveId"],
        "classificationName": ids["classificationId"],
        "evaluationText": evaluation_result["evaluationText"],
        "evaluatedSelection": evaluation_result["
            evaluatedSelection"],
    }

    if mode == "mode2":
        # EvaluationResponseMode2 implements IEvaluationResponse
        evaluation_reponse["evaluationLikert"] =
            evaluation_result["evaluationLikert"]
    elif mode == "mode3":
        # EvaluationResponseMode3 implements IEvaluationResponse
        evaluation_reponse["evaluationAgreement"] =
            evaluation_result["evaluationAgreement"]
    else:
        raise ValueError(f"Invalid mode: {mode}")

    eval_info = get_eval_info(eval_info_base, evaluation_result)

```

**S8 Synthetic Evaluation Schemas & Prompting**

```

        if eval_info is None:
            # Ditch this whole evaluation_results set as None
            # eval_info indicates bad response from OpenAI
            is_error = True
            formatted_evaluation_responses = []
            break
        else:
            formatted_evaluation_responses.append(evaluation_reponse
            )
            job_status["eval_info"].append(eval_info)

    job_entry["results"] = formatted_evaluation_responses

    evaluation_perspectives_modified = []
    # record_evaluation_failure/record_evaluation_data use different naming
    for perspectives
    for evaluation_perspective in perspectives:
        evaluation_perspectives_modified.append({
            "perspectiveId": evaluation_perspective["id"],
            "perspectiveName": evaluation_perspective["name"],
            "perspectiveText": evaluation_perspective["text"],
        })

    if is_error:

        logger.error(f"Got one or more bad evaluation_results")

        evaluation_failure_data = {
            "agent": self.client_id,
            "contextIdentifier": generated_request["context_identifier"],
            "queryInfo": query_info,
            "additionalQueryInfo": additional_query_info,
            "finishReason": finish_reason,
            "queryState": query_state,
            "mode": mode,
            "model": openai_model,
            "evaluationItem": evaluation_item,
            "evaluationPersona": persona,
            "evaluationPerspectives": evaluation_perspectives_modified,
            "evaluationRequest": generated_request
        }

        if is_dry_run:
            logger.info("DRY RUN - not storing evaluation failure data")
            job_status["evaluation_failure_data"].append(
                evaluation_failure_data)
        else:
            record_evaluation_data_response = self.awagdata_client.
                record_evaluation_failure(evaluation_failure_data)

    else:

        # Ref EvaluateRecorderRestClient class in original Java

        # For legacy compatability
        expanded_query_state = query_state.copy()
        expanded_query_state["finish_reason"] = finish_reason
        expanded_query_state["usage"] = info_json["usage"]

        # contextId is a consequence of original model design, where
        # multiple
        # itemIds can be associated with a single query. We give the query
        # a
        # contextId so that query-level data can be recorded. However, it
        # adds

```

**S8 Synthetic Evaluation Schemas & Prompting**

```

# a level of complexity to each awagdata database entry

context = {
    "contextId": generated_request["context_identifier"],
    "evaluationItems": [evaluation_item],
    "openaiApiCompletionsQueryInfo": query_info,
    "additionalQueryInfo": additional_query_info,
    "queryState": expanded_query_state
}

evaluate_time_text = evaluation_item["evaluateTime"]
evaluate_time_obj = parser.parse(evaluate_time_text)
evaluate_time_ms = int(evaluate_time_obj.timestamp() * 1000)

if tags_dest:
    item_tags = tags_dest
else:
    item_tags = evaluation_item["tags"]

om_item = {
    "itemId": evaluation_item["contentItemSummary"]["itemId"],
    "tags": item_tags,
    "contentItemSummary": evaluation_item["contentItemSummary"],
    "classifications": evaluation_item["evaluateClassifications"],
    "evaluateSourceType": evaluation_item["evaluateSourceType"],
    "evaluateSourceOriginator": evaluation_item["evaluateSourceOriginator"],
    "evaluateSourceChannel": evaluation_item["evaluateSourceChannel"],
    "evaluateTimeText": evaluate_time_text,
    "evaluateTimeMs": evaluate_time_ms,
    "evaluateTitle": evaluation_item.get("evaluateTitle", None),
    "evaluateText": evaluation_item.get("evaluateText", None),
    "evaluationResponses": []
}

for far in formatted_evaluation_responses:

    # Equivalent to EvaluationResponseModel, EvaluationResponseMode2
    # , EvaluationResponseMode3

    # Name is slightly confusing; keeping for consistency with Java
    # original
    # Each one of these is an item in the evaluationResponses list
    # Perspectives is a list because it can have multiple values for
    # model
    om_classification = {
        "classificationName": far["classificationName"],
        "perspectives": [] # This is the classification *result*
    }

    if mode == "model1":

        for persp in far["perspectives"]:
            # In this case, pass persp to
            # get_resp_perspective_from_far()
            # - should have same structure as
            # formatted_evaluation_response
            this_perspective = get_resp_perspective_from_far(persp)
            this_perspective["evaluationLikertVal"] = persp["evaluationLikert"]
            this_perspective["evaluationLikertText"] =
            get_likert_label(persp["evaluationLikert"])
            om_classification["perspectives"].append(
                this_perspective)

```



## S8 Synthetic Evaluation Schemas &amp; Prompting

```

        elif mode == "mode2":

            this_perspective = get_resp_perspective_from_far(far)
            this_perspective["evaluationLikertVal"] = far["
                evaluationLikert"]
            this_perspective["evaluationLikertText"] = get_likert_label(
                far["evaluationLikert"])
            om_classification["perspectives"].append(this_perspective)

        elif mode == "mode3":

            this_perspective = get_resp_perspective_from_far(far)
            this_perspective["evaluationAgreement"] = far["
                evaluationAgreement"]
            om_classification["perspectives"].append(this_perspective)

        else:

            raise ValueError(f"Invalid mode: {mode}")

        om_item["evaluationResponses"].append(om_classification)

    evaluation_success_data = {
        "agent": self.client_id,
        "evaluationPersona": persona,
        "evaluationPerspectives": evaluation_perspectives_modified,
        "context": context,
        "evaluationRequest": generated_request,
        "items": [om_item]
    }

    if is_dry_run:
        logger.info("DRY RUN - not storing evaluation success data")
        job_status["evaluation_success_data"].append(
            evaluation_success_data)
    else:
        record_evaluation_data_response = self.awagdata_client.
            record_evaluation_data(evaluation_success_data)

    job_status["evaluations"].append(job_entry)

    job_status["processed_items"] += 1

    if items_to_process is not None and items_to_process > 0:
        if job_status["processed_items"] >= items_to_process:
            logger.info(f"Stopping at passed limit of {items_to_process} items")
            limit_reached = True
            break

    if remaining < 1:
        logger.debug(f"No more data remaining after page {page} (count: {count})")
        break

    page += 1

    job_status["message"] = f"Completed processing of {len(job_status['evaluations'])}
        items."
    job_status["status"] = "COMPLETE"

    if not is_dry_run:
        record_evaluation_job_response = self.awagdata_client.record_evaluation_job(
            job_status)

except Exception as e:

```

## S8 Synthetic Evaluation Schemas & Prompting

---

```
        msg = f"Error in process_evaluations for tag_source: '{tag_source}': {e}"
        job_status["status"] = "ERROR"
        job_status["message"] = msg
        logger.error(msg)
        log_exception(self.client_id, e, include_traceback=True)

        self.job_statuses[job_id] = job_status

    if is_async:

        logger.debug(f"Started process_evaluations...")
        thread = threading.Thread(target=process_evaluations)
        thread.start()

    else:

        process_evaluations()

    return self.job_statuses.get(job_id)
```

Listing S8.16: AwAg Evaluation Processor