# **Milestone 4 – API (route handler) Specification**

**#1 Route Handler – galleryOverview(req, res)**

**Description**: query and return for gallery’s summary statistics

**Request Path**: GET /home

**Route Parameter(s)**: n/a

**Query Parameter(s)**: n/a

**Return Type**: JSON

**Return Parameters**:

|  |
| --- |
| { **msg**: (string of welcoming message),  **results**: [  {classification: "painting", artworkCounts: (int) },  {classification: "drawing", artworkCounts: (int) },  {classification: "print", artworkCounts: (int) }  ]  } |

**Expected (Output) Behaviour**:

* This is a **static** query, with no parameter, output will always be a JSON array of Summary Statistics in the above format

**#2 Route Handler – artworkInfo(req, res)**

**Description**: given the objectID of an artwork, this function will query and return all the necessary/detailed information (results are broken down into 3 parts) about a given artwork

**Request Path**: GET /artwork

**Route Parameter(s)**: n/a

**Query Parameter(s)**: objectID (int)

**Return Type**: JSON

**Return Parameters**:

|  |
| --- |
| { **results\_P1**: [ (artwork cardinality: 1)  { title: (string),  attribution: (string),  medium: (string, nullable),  dimensions: (string, nullable),  classification: (string),  series: (string, nullable),  portfolio: (string, nullable),  volume: (string, nullable),  URL: (string) }  ],  **results\_P2**:[ (artist cardinality: 1 .. \*)  { preferredDisplayName: (string),  displayOrder: (int),  displayDate: (string),  visualBrowserNationality: (string)},  {element2},  . . . . . ,  (elememtN)  ],  **results\_P3**:[ (cardinality: **0** .. 6)  { termType: (string), term: (string) },  . . . . .,  {element6}  ]  } |

**Expected (Output) Behaviour**:

* CASE 1: if objectID query-parameter is specified
  + Case 1.1: Regular values 🡺 If the objectID is found
    - return the JSON array as specified above
  + Case 1.2: Faulty values 🡺 1) If the objectID is a number but is not found, OR 2) it is a non-nuemric (i.e. a string text)
    - return empty JSON array as the value for each of the 3-part-results without causing an error: {"results\_P1":[],"results\_P2":[],"results\_P3":[]}
* CASE 2: if objectID query-parameter is not specified
  + return the information of default artwork {title: “American Flamingo”, objectID = 32572}

**#3 Route Handler – similarArtwroks(req, res)**

**Description**: recommand similar artwork by primary (i.e. results\_P1) and secondary (i.e. results\_P2) similarities

**Request path**: GET /artwork/similarArtworks

**Route Parameter(s)**: n/a

**Query Parameter(s)**: objectID (int)

**Return Type**: JSON

**Return Parameters**:

|  |
| --- |
| { **results\_P1**: [ (artwork cardinality: 0..4)  { title: (string),  attribution: (string),  objectID: (int),  thumbURL: (string),  series: (string, nullable),  portfolio: (string, nullable),  volume: (string, nullable) },  . . . . . ,  {elememt4}  ],  **results\_P2**:[ (artist cardinality: 0 .. 4)  { title: (string),  attribution: (string),  objectID: (int),  thumbURL: (string),  termType: (string),  series: (string, nullable),  portfolio: (string, nullable),  volume: (string, nullable) },  . . . . . ,  {elememt4}  ]  } |

**Expected (Output) Behaviour**:

* **CASE 1:** if objectID query-parameter is specified
  + Regular Case: if found any similar artwork with the given objectID, return the 2-part-results JSON array as specified above
  + Edge Case: if there is no similar artwork found, will return message as JSON array:

{"results\_P1":"NOTHING","results\_P2":"NOTHING"}

* **CASE 2**: if objectID query-parameter is not specified
  + return similar artworks to the default artwork {title: “American Flamingo”, objectID = 32572}

**#4 Route Handler – filterSearch(req, res)**

**Description**: search relavent artworks by applying a variety of filtering conditions. Result will be returned by the following ordering: endYear >> title >> attribution.

**Request Path**: GET /search/byFilter

**Route Parameter(s)**: n/a

**Query Parameter(s)**: nationality (string), style (string), classification (string), beginYear (int), endYear(int), page (int, default: 1), pagesize (int, default: 6)

**Return Type**: JSON

**Return Parameters**:

|  |
| --- |
| { **results**: [ (artwork cardinality: 0..\*)  { title: (string),  attribution: (string),  endYear: (int),  objectID: (int),  thumbURL: (string) },  . . . . . ,  {elememtN}  ]  } |

**Expected (Output) Behavior**:

* Regular Case: Return an array with all artworks that match the constraints
* Edge Case: If no artwork satisfies the constraints, return an empty array as {"results":[]} , without causing an error

**#5 Route Handler – keywordSearch(req, res)**

**Description**: search relavent artworks by artwork's title OR/AND artist's name (**in case-insensitive manner**)

Request Path: GET /search/byKeyword

**Route Parameter(s)**: n/a

**Query Parameter(s)**: artworkTitle (string), artistName (string), page (int, default: 1), pagesize (int, default: 6)

**Return Type**: JSON

**Return Parameters**:

|  |
| --- |
| { **results**: [ (artwork cardinality: 0..\*)  { title: (string),  attribution: (string),  endYear: (int),  objectID: (int),  thumbURL: (string) },  . . . . . ,  {elememtN}  ]  } |

**Expected (Output) Behavior**:

* Regular Case: Return an array with all artworks that match the searching keywords.
* Edge Case: If no artwork satisfies the constraints, return an empty array as {"results":[]} , without causing an error

**#6 Route Handler – naughtySearchHeight(req, res)**

**Description**: naughty search "painting" artworks by matching user's height (cm) with artwork's height (cm), return a list of artworks in the order of least height-deviation to most height-deviation

**Request Path**: GET /search/naughtySearchByHeight

**Route Parameter(s)**: n/a

**Query Parameter(s)**: height (int or float, default: 170), page (int, default: 1), pagesize (int, default: 10)

**Return Type**: JSON

**Return Parameters**:

|  |
| --- |
| { **results**: [ (artwork cardinality: 0..10)  { title: (string),  attribution: (string),  objectID: (int),  thumbURL: (string),  dimension: (float),  deviation: (float) },  . . . . . ,  {elememtN}  ]  } |

**Expected (Output) Behavior**:

* Regular Case: Return an JSON array of artworks that are around the given height
* Edge Case: If any of the query parameters height, page, or pagesize is non-numeric, return an empty array as {"results":[]} , without causing an error

**#7 Route Handler – naughtySeachBirthYear(req, res)**

**Description**: naughty search artworks by matching with user's birthYear, return the artwork (of all kinds) produced in/around the birthYear, and then order the results in height descending order (tall --> short)

**Request Path**: GET /search/naughtySearchByBirthYear

**Route Parameter(s)**: n/a

**Query Parameter(s)**: birthYear (int), page (int, default: 1), pagesize (int, default: 10)

**Return Type**: JSON

**Return Parameters**:

|  |
| --- |
| { **results**: [ (artwork cardinality: 0..10)  { title: (string),  attribution: (string),  objectID: (int),  endYear: (int),  deviation: (int),  thumbURL: (string),  deviation: (float) },  . . . . . ,  {elememtN}  ]  } |

**Expected (Output) Behavior**:

* Regular Case: Return an JSON array of artworks that were completed around the given year
* Edge Case: If any of the query parameters birthYear, page, or pagesize is non-numeric, return an empty array as {"results":[]} , without causing an error

**#8 Route Handler – analysisOverview(req, res)**

**Description**: query and return for summary statistics of the analysis results

* Part 1) Showing how many term varieties each big analysis category contains
  + i.e. School (162), Style(82), Theme(467), Technique(163), Keyword(6320), Place Executed (1000)
* Part 2) Showing the top 5 popular terms (and the counts of associated artworks) for each category

**Request Path**: GET /analysis/analysisOverview

**Route Parameter(s)**: n/a

**Query Parameter(s)**: n/a

**Return Type:** JSON

**Return Parameters**:

|  |
| --- |
| {**Style**:[ ( Style terms cardinality: 80)  {term: (string), StyleCounts: (int)}, . . . . ., {element5}  ],  **School**:[ (School terms cardinality: 80)  {term: (string), SchoolCounts: (int)}, . . . . ., {element5}  ],  **Theme**:[ ( Theme terms cardinality: 80)  {term: (string), ThemeCounts: (int)}, . . . . ., {element5}  ],  **Technique**:[ ( Technique terms cardinality: 80)  {term: (string), TechniqueCounts: (int)}, . . . . ., {element5}  ],  **Keyword**:[ ( Keyword terms cardinality: 80)  {term: (string), KeywordCounts: (int)}, . . . . ., {element5}  ],  **PlaceExecuted**:[ ( PlaceExecuted terms cardinality: 80)  {term: (string), PlaceExecutedCounts: (int)}, . . . . ., {element5}  ]  } |

**Expected (Output) Behaviour**:

* This is a **static** query, with no parameter, output will always be a JSON array of Summary Statistics in the above format

**#9 Route Handler – analysisByType(req, res)**

**Description**:

* front-end will prompt user to specify whcih type of analysis he/she wants to check
* this function will return, in descending order, most popular terms under the analysis category
* analysis category: Style, School, Theme, Technique, Keyword, Place Executed

**Request Path**: GET /analysis/analysisByType/:analysisType

**Route Parameter(s)**: analysisType (string)

**Query Parameter(s)**: page (int), pagesize (int, default: 10)

**Return Type**: JSON

**Return Parameters**:

|  |
| --- |
| { **results** : [ ( term cardinality: 1 .. \*)  {term: (string), termCounts: (int)},  . . . . .,  {elementN}  ]  } |

**Expected (Output) Behaviour**:

* CASE 1: if analysisType route-parameter is specified
  + Case 1.1: if the page query parameter is specified (assuming in range), return a JSON array containing the analysis terms and counts on the corresponding page-number
  + Case 1.2: if the page query parameter is NOT specified, return a JSON array of all the terms and counts under this analysis category
* CASE 2: if analysisType route-parameter is NOT specified
  + return a JSON array of query result for defualt analysis typle “Style” without causing error. (page and pagesize query parameters are handled in the same way as in Case 1)

**#10 Route Handler – portraitsAcrossTime (req, res)**

**Description**:

* Within the given time range, find and return artworks that have their theme of contents been defined as portraits
* front-end will fetch for 5 different time-spans: 16th (1500~1599), 17th (1600~1699), 18th(1700~1799), 19th (1800~1899), 20th (1900~1999) centries

**Request Path**: GET /analysis/portraitsAcrossTime/:artworkClass

**Route Parameter(s)**: artworkClass (string)

**Query Parameter(s)**: beginYear (int, default: 1500), endYear(int, default: 1599), page (int), pagesize (int, default: 5)

**Return Type**: JSON

**Return Parameters**:

|  |
| --- |
| { **results**: [ (artwork cardinality: 1 .. \*)  { title: (string),  attribution: (string),  classification: (string),  objectID: (string),  thumbURL: (string a URL),  endYear (int) },  {element2},  . . . . . . .,  {elementN}  ]  } |

**Expected (Output) Behaviour**:

* CASE 1: If the page query-parameter is defined
  + return an a JSON array containing the artworks on the corresponding page-number, all other query-parameters have default values, so no error will be raised even if missing the specification of these parameters
* CASE 2: if the page query-parameter is NOT defined
  + return an a JSON array containing only the first 5 artworks, all other query-parameters have default values, so no error will be raised even if missing the specification of these parameters