Objective #1:

## Evaluate Skeleton project setup following the setup documentation.

**Installing Jhipster:**

**Pre-requisites**: Java 8, Node.js, Yarn

* Install Java 8 [from Oracle](http://www.oracle.com/technetwork/java/javase/downloads/index.html)
* Install Node.js from [http://nodejs.org](http://nodejs.org/)
* Install Yarn from <https://yarnpkg.com/en/docs/install>

To install run:

1. yarn global add bower
2. yarn global add gulp-cli
3. yarn global add generator-jhipster

**Creating the project using Jhipster:**

1. From command line: mkdir **hackathon**, cd **hackathon**, **jhipster**
2. Q&A:
   1. Which type of application would you like to create?
      * Monolithic application
   2. What is the base name of your application?
      * hackathon
   3. What is your default Java package name?
      * com.armedia.hackathon
   4. Do you want to use the JHipster Registry to configure, monitor and scale your application?
      * Y
   5. Which type of database would you like to use?
      * An SQL database
   6. Which production database would you like to use?
      * MySQL
   7. Which development database would you like to use?
      * H2 with its data stored on disk
   8. Do you want to use Hibernate 2nd level cache?
      * No
   9. Would you like to use Maven or Gradle?
      * Maven
   10. Which other technologies would you like to use?
       * None (skip all) - Press Enter to skip all
   11. Which Framework would you like to use for the client?
       * Pick your choice
   12. Would you like to use the LibSass stylesheet preprocessor for your CSS?
       * N
   13. Would you like to enable internationalization support?
       * N
   14. Which testing frameworks would you like to use?
       * Select all using <a> key.
   15. Would you like to install other generators from the JHipster Marketplace?
       * N
3. ./mvnw to build the snapshot of the configured Spring Boot Application
4. From your favorite web browser open <http://localhost:8080> to access admin panel for Jhipster after successful build.

Objective #2:

## Authentication with ArkCase

ArkCase provides several REST endpoints to retrieve data in JSON format. These data can be related to a case file, task, file or folder object among many others.

ArkCase provides various authentication methods like basic authentication, token-based authentication, multi-factor authentication etc. We will be using a combination of the first two. In order to get authenticated to ArkCase, we will be using one of the APIs to generate an authentication token. We will be passing a basic authentication header to this REST end-point that will generate a token which can be used for any subsequent REST calls. Once the user has received the token, it is not necessary to pass the basic authentication header anymore when making REST calls to ArkCase. The token will expire after 30 minutes of inactivity and will have to renewed by following the process mentioned above.

We will use Swagger to look up the required API. A barebone structure of the swagger response in JSON format looks like below:

{"paths": {

" https://core.arkcase.dev.armedia.com/arkcase/api/latest/authenticationtoken": {

"get": {

…

"consumes": ["application/json"],

"produces": ["text/plain"],

…

"responses": {

"200": {

"description": "OK",

"schema": {

"type": "string"

}

},

"401": {

"description": "Unauthorized"

},

"403": {

"description": "Forbidden"

},

"404": {

"description": "Not Found"

}

}

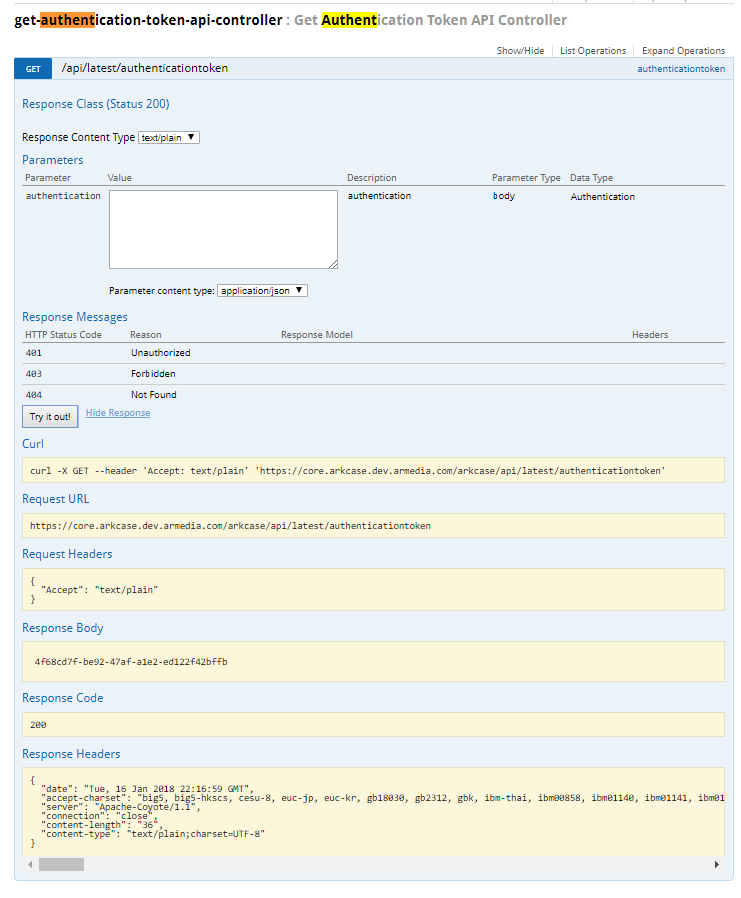
}

}}}

A better way to visualize the information is to log in to the Swagger UI. To get to ArkCase Swagger, copy and paste the following link to your browser :

<https://core.arkcase.dev.armedia.com/arkcase/swagger-ui.html>

Enter the ArkCase login credentials and then look for **“get-authentication-token-api-controller : Get Authentication Token API Controller”**. You can explore this interface to get an idea about the API end-point. There is no need to provide authentication and session fields values in ArkCase Swagger pages as they are set by Spring by default. Enter the rest of the parameter combinations and click **“Try it out!”**.



Once you have a token, you will need to append the token to the end of the request URLs e.g.

https://core.arkcase.dev.armedia.com/arkcase/api/latest/plugin/search/advancedSearch**?acm\_ticket=<$token>**

## Using Advanced Search API

In the Swagger UI, find **“advanced-search-api-controller : Advanced Search API Controller”**.

ArkCase Advanced Search API is a wrapper around standard SOLR APIs. You can formulate SOLR queries to retrieve objects as below:

*q=object\_type\_s:CASE\_FILE*

*q=object\_type\_s:TASK*

*q=object\_type\_s:FILE*

Child objects like FIlE and TASK will have a reference to the parent object which is maintained in the “parent\_ref\_s” field in the format “ID-OBJECT\_TYPE” where object types could be **CASE\_FILE**, **TASK** or **FILE**. In order to retrieve child objects for a parent object, you can formulate SOLR queries as below:

*q=object\_type\_s:FILE AND parent\_ref\_s:101-CASE\_FILE*



*You can explore various combinations of queries by using the fields in the response JSON.*

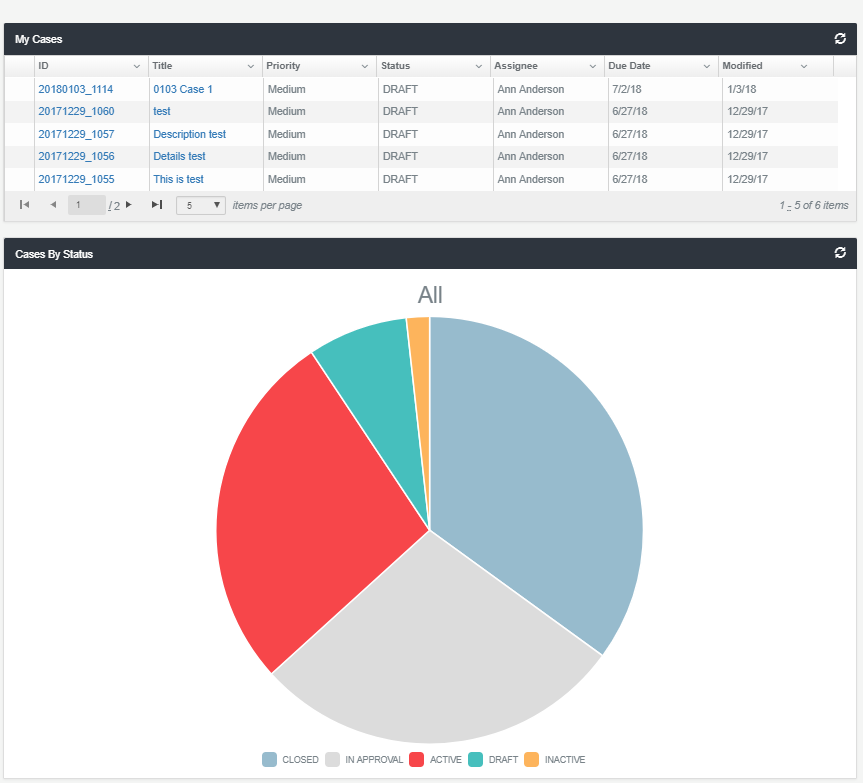
[Task Sample JSON](https://core.arkcase.dev.armedia.com/solr/acmAdvancedSearch/select?q=object_type_s%3ATASK&rows=1&wt=json&indent=true)

[File Sample JSON](https://core.arkcase.dev.armedia.com/solr/acmAdvancedSearch/select?q=object_type_s%3AFILE&rows=1&wt=json&indent=true)

[Case File Sample JSON](https://core.arkcase.dev.armedia.com/solr/acmAdvancedSearch/select?q=object_type_s%3ACASE_FILE&rows=1&wt=json&indent=true)

Objective #3:

## Utilize the data



Once you get the API calls working, you’re free to show your creativity and utilize the data in any manner you deem suitable. Think about charts, tables, graphs, your own HTML page with custom CSS, JS etc.