Scope

You will need to work with:

- Google Forms add-ons https://developers.google.com/apps-script/reference/forms
- Mongo Atlas https://docs.atlas.mongodb.com/api

Task Description

The task is to:

1. Listen to form submission event (you will be provided with a sample add-on code that already have the listener function);
2. On form submission, construct an object that represents the form structure;
3. Attempt to find an already existing revision to the form in Mongo by comparing structures and:
    a. If such a structure is not yet stored, then create a new revision and use it as a reference to the submitted responses. A new revision can be indicated as a number starting from 0;
    b. If such a structure already exists, then use it as a reference to the submitted responses and do not create a new revision;
4. After form structure checking (and saving if needed), construct the responses object. It must include all responses and must be referenced to the correct form structure revision. Files, images and other non-input fields can be discarded from responses object.
5. Save the responses object to Mongo.

Note: please use JS ES5.

Acceptance Criteria

- It must be possible to fully reconstruct forms and responses, basing on just the data from Mongo.
- The script should support all possible scenarios and form structures, including:
    - Multi-page forms
    - All possible field types (inc. multi-checkboxes, matrix/grid etc)
    - All possible form filling scenarios
- If something fails we should know about it, for example, see it in Google Script logs.
- The code must not be dumped into a single function and reasonably split into multiple functions.

Annex

The “Get started” code

```
setOnSubmitTrigger();

/**
 * Sets the onFormSubmit trigger.
 */
function setOnSubmitTrigger () {
  var form = FormApp.getActiveForm();
  var functionName = 'onFormSubmit';

  if (!checkIfTriggerExists(functionName)) {
    ScriptApp.newTrigger(functionName)
      .forForm(form)
      .onFormSubmit()
      .create();
  }
}

/**
 * Checks if trigger exists for a given function name in the current form
 *
 * @param  {String} functionName
 * @return {Boolean}
 */
function checkIfTriggerExists (functionName) {
  var form = FormApp.getActiveForm();
  var allTriggers = ScriptApp.getUserTriggers(form);
  var triggerExistsForForm = false;

  for (var i = 0; i < allTriggers.length; i++) {
    if (
      allTriggers[i].getHandlerFunction() === functionName &&
      allTriggers[i].getTriggerSourceId() === form.getId()) {
      triggerExistsForForm = true;
      break;
    }
  }

  return triggerExistsForForm;
}

/**
 * Responds to a form submission event if an
 * onFormSubmit trigger has been enabled.
 *
 * @param {Object} e The event parameter created by a form
 */
function onFormSubmit (e) {
  var form = FormApp.getActiveForm();

  // ... the structure and responds saving ...
}
```