# **Morphir Jira Tasks**

## Data Constraints

* + Create JSON Schema Backend - Nunya
  + Add the initial set of data constraint to the Morphir IR - Nunya
  + Develop strategy for firm wide type system - Attila

## Spark Backend

* + Filter – Map SDK functions of column expressions
  + Join – Collect all Dick lookups within filter/map
  + Test – Add a test to the morphir-elm build that compiles the Spark code generated from the reference-model
  + Join – Add joins to relational expressions based on look-ups
  + Validate – Create an API to help modeling operations that might fail
  + Validate – Enrich the relational IR with error annotations
  + Aggregate – Extend backends to handle errors gracefully
  + Aggregate – Map aggregations in relational backend
  + Aggregate – Map aggregations in Spark backend
  + Test – Add a test to the morphir-elm build that runs the Spark test cases in reference model
  + Build AST for Spark Operations – Ebuka
  + Build public models that simulate LCR - Osborne
  + Update gulf file with Spark configuration and add Spark dependencies to morphir-elm – Nunya

## Build Performance

* + Add *removeUnreachableNodes* function to the DAG that returns a new DAG with unreachable nodes removed based on a set of included nodes.
  + Confirm if ‘morphir make’ and ‘morphir-elm make’ produces that same IR on the serenity models (Elorm)
  + Add the missing JS file to the morphir-elm NPM package to fix ‘morphir-make’ (Elorm)

## Maintenance

* + Document all the available components within the Morphir toolset – Attila
  + Fix NPM and Elm package publish automation
  + Update the morphir-jvm version in morphir-elm to refer to the most recent version and release the NPM package – Osborne
  + Ensure that the NPM package we publish is valid – Osborne
  + Fix the automated NPM publishing Github action
  + Verify Resolver only fails on usage of conflicting imports – Ebuka
  + Ensure the the unused modules are never processed in the incremental Frontend – Osborne
  + Check if the file content changed before updating a generated file – Elorm
  + Fix the morphir-make command to infer types more accurately without adding in fake types that could cause type inferencing within the repo to fail – Ebuka

## Scala JSON Serialization

* + Implement the Scala gen command – Elorm
  + Implement Elm decoders/encoders for the scala gen code generation – Elorm

## Cadl Support

* + Create a Cadl AST
  + Create a Codec from the Cadl AST to the JSON serialization format of Cadl
  + Define the mapping between Morphir and Cadl in a document
  + Implement the Morphir → Cadl mapping as a Backend
  + Implement the Cadl → Morphir mapping as a Frontend

## DataPass Pilot – Stepen

## Scala Frontend

* + Engage with Capital One

## Morphir Documentation

* + Create a walk-through for building Encoders – Kindson
  + Create a walk through for building Decoders – Kindson

## Data Processing Visualization

* + Visualize filter operations
  + Visualize map operations
  + Visualize lookup operations
  + Visualize validation
  + Visualize aggregation

## Smithy Support

* + Create a Smithy AST
  + Create a Codec from the Smithy AST to JSON serialization format of Smithy
  + Define the mapping between Morphir and Smithy in a document
  + Implement the Morphir → Smithy mapping as a Backend
  + Implement the Smithy → Morphir mapping as a Frontend

## Package Management

* + Define a new IR storage format that is more optimized for partial processing, detecting changes and small size – Ebuka
  + Add a function to expose the specification of the package to the Repo API
  + Add a function to expose the names of the packages that a repo depends on the Repo API
  + Add a function to include a subset of of a package in the repo as a dependency to the Repo API

## Modeler Experience

* + Engage with the Serenity team
  + Calculate the coverage metrics for the test cases
  + Make breadcrumbs clickable
  + Visualize series of function calls as data flow charts
  + Add Decimal support to the interpreter

## Decision Table Support