TECHNICAL DESIGN DOCUMENT

Project: DSS(1407)
Decision Support System - Phase 1

DEVELOPMENT AND IMPLEMENTATION COLLABORATION BETWEEN:
Punjab Municipal Corporation and eGovernments Foundation

TABLE OF CONTENTS

1. Preface

1.1. Purpose of this document

The purpose of this document is to provide technical details of software components that this project is made of.

1.2. Use of this document

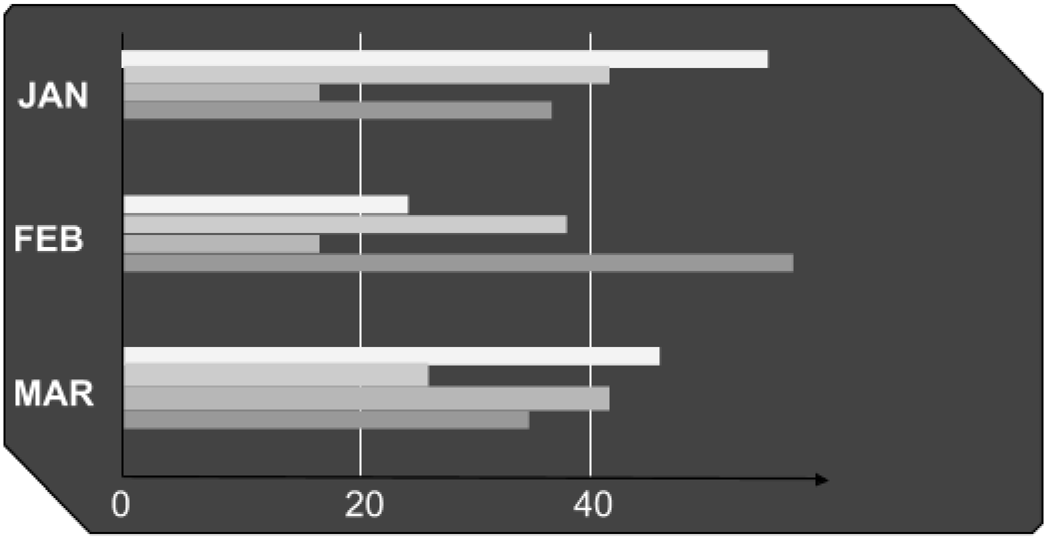
This document describes a solution to a given technical problem. It is a specification for a software program.

1.3. Overview

A decision support system (DSS) is a composite tool that collects, organizes and analyzes business data to facilitate quality decision-making for management, operations and planning. A well-designed DSS aids decision makers in compiling a variety of data from many sources: raw data, documents, personal knowledge from employees, management, executives and business models. DSS analysis helps organizations to identify and solve problems, and make decisions.

1.4. Objective

A dashboard to provide a comprehensive insight on the performance of all the ULBs across different domains on a timely basis.
Helps identifying hotspots and track performances.



1.5. A Reference Architecture for the DSS Programme

Ingest pipeline architecture comprises 4 modules ingest, validate, transform and enrich.
Each module works in sequences as shown in the Diagram 1 below. Each module receives the data from pipeline and pushes to pipeline.

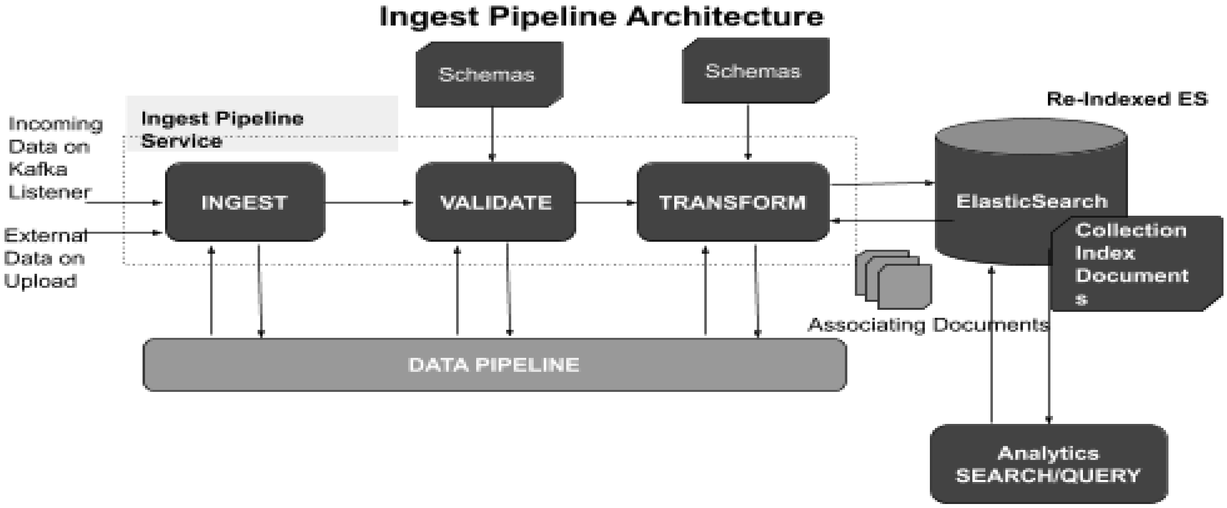
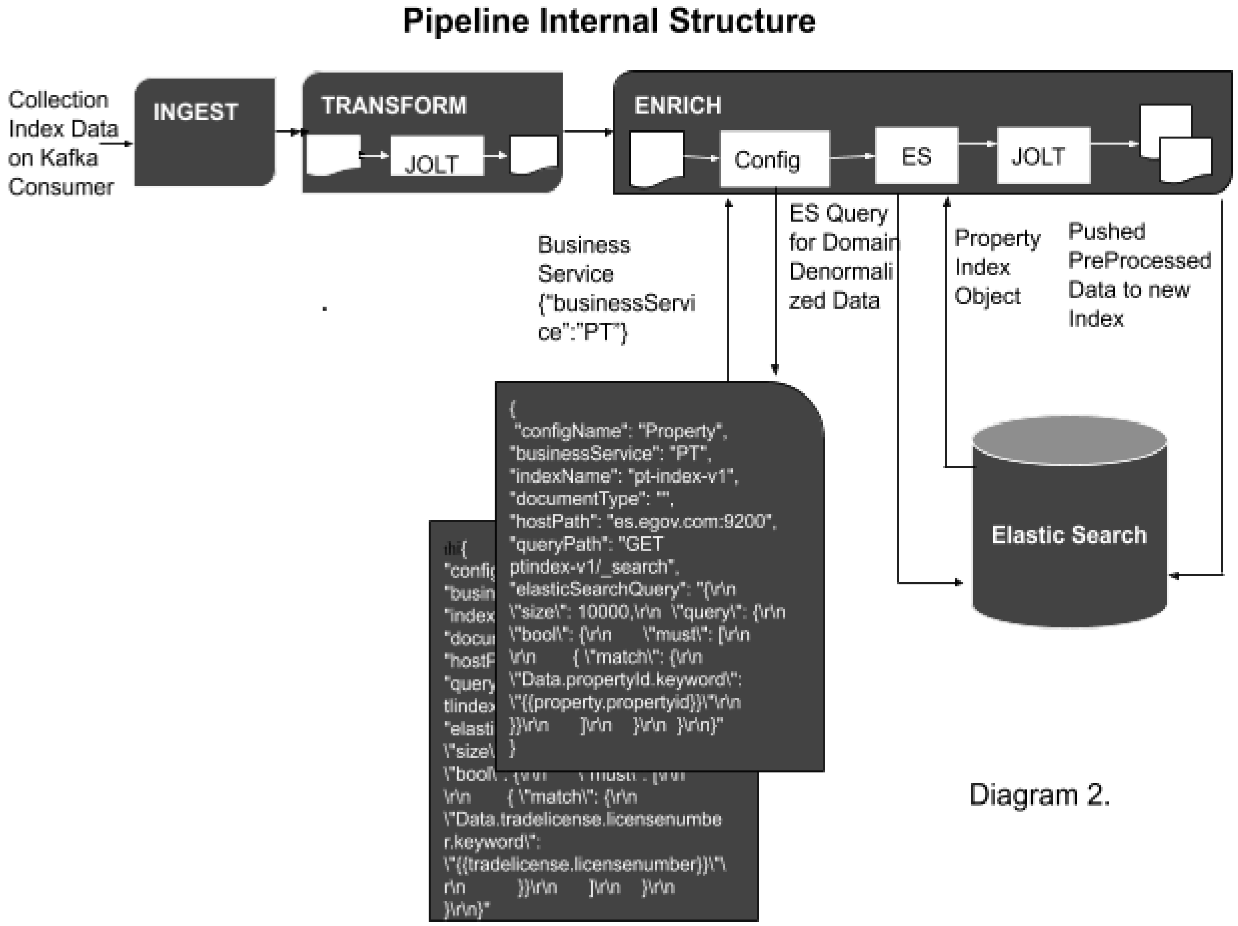


Diagram 1.

Pipeline internal structure as shown above consumes the collection/incoming data and transforms the data using jolt configurations. [Jolt is a json to json transformation library. You can find more information here]. After transformation, data enrichment is performed in order to hook the collection data to its business module data where the module data also gets transformed shown on the above enrichment box in Diagram 2. The final processed data is pushed to the new index.


Pipeline Internal Structure
Collection Index Data on Kafka Consumer
INGEST
TRANSFORM
JOLT
ENRICH
Config
ES
JOLT
Business Service {"businessService":"PT"}
ES Query for Domain Denormalized Data
Property Index Object
Pushed PreProcessed Data to new Index
Elastic Search

```
{
 "configName": "Property",
"businessService": "PT",
"indexName": "pt-index-v1",
"documentType": "",
"hostPath": "es.egov.com:9200",
"queryPath": "GET
ptindex-v1/_search",
"elasticSearchQuery": "{\r\n
\"size\": 10000,\r\n  \"query\": {\r\n
\"bool\": {\r\n      \"must\": [\r\n
\r\n       { \"match\": {\r\n
\"Data.propertyId.keyword\":
\"{{property.propertyid}}\"\r\n
}}\r\n      ]\r\n    }\r\n  }\r\n}"
}
```

```
ihi{
"config
"busin
"index
"docu
"hostF
"query
tlindex
"elasti
\"size\
\"bool\
\r\n       { \"match\": {\r\n
\"Data.tradelicense.licensenumbe
r.keyword\":
\"{{tradelicense.licensenumber}}\"\
r\n          }}\r\n      ]\r\n    }\r\n
}\r\n}"
```

Diagram 2.

Analysis pipeline approach is shown below in Diagram 3. This provides dashboard configurations for UI representation. The request for a configuration is received. The specific dashboard configuration is provided to the client/UI. Based on the configuration the UI paints the dashboard layouts. Once request for a specific visualisation code/chart is received, the application reads the chart configuration and identifies the index, aggregation queries and builds the final query to fire against the ES index. It fetches the data and translates it into a generalised response readable to client/UI.

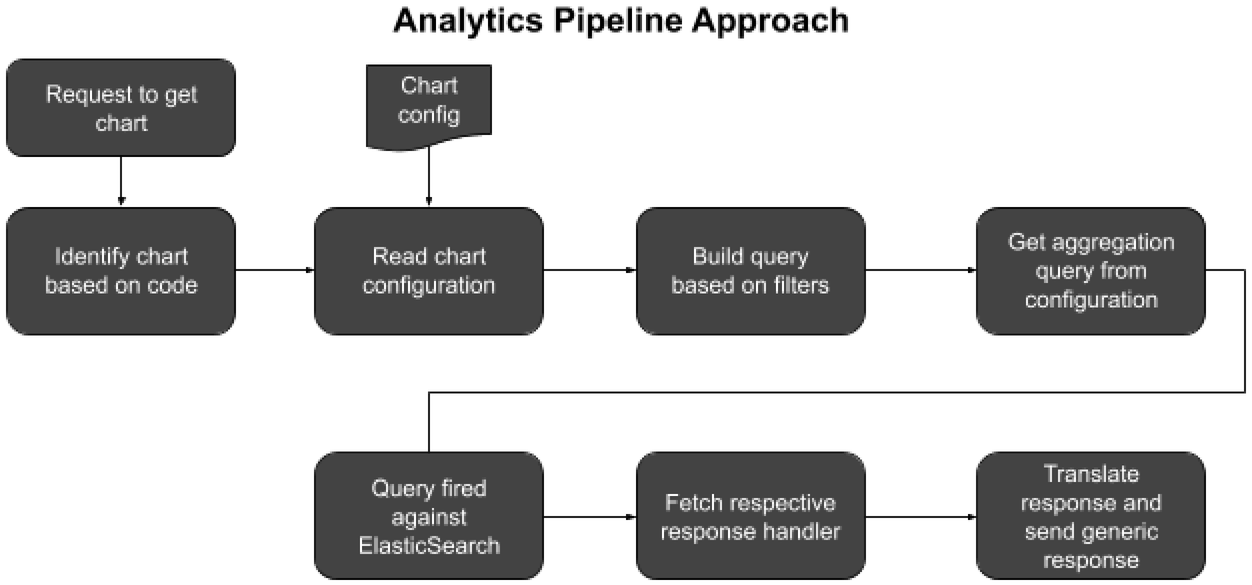
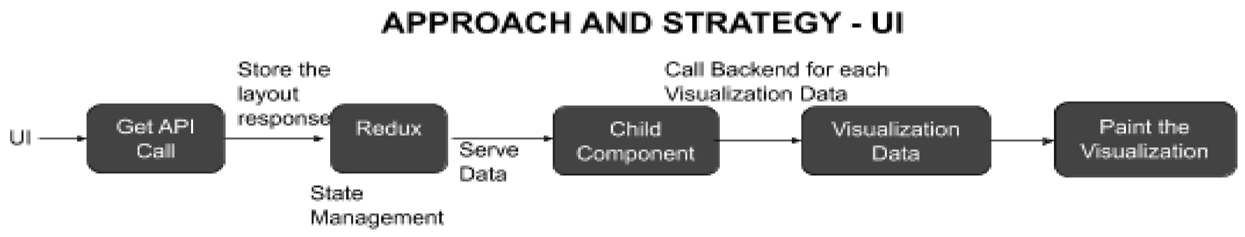
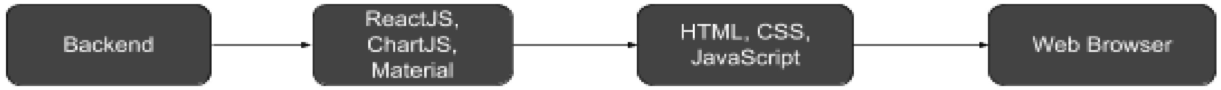

Analytics Pipeline Approach
Request to get chart
Chart config
Identify chart based on code
Read chart configuration
Build query based on filters
Get aggregation query from configuration
Query fired against ElasticSearch
Fetch respective response handler
Translate response and send generic response

Diagram 3

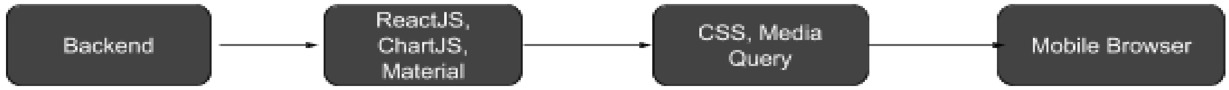


UI Functionality

Diagram 4



Web Flow Diagram



Mobile Flow Diagram

Diagram 5.

Above diagrams are of UI. DSS Dashboard - Technical Document for UI documentation contains more information.

1.6. Intended Audience

The intended audience is:

- Project customer
- Project supervisor
- Project team members
- Users

2. Introduction

2.1. Purpose

Dss system collects, organizes and analyzes business data to facilitate quality decision-making for management, operations and planning.

2.2. Scope

i. Personas

This are the type of users

- State Level Admin
- Commissioner
- Domain Level Employee

User Personas

Role	Key Decisions
State Administrator	- Monitor Performances - Examine Trends
Commissioner	- Operation Strategy - Root Cause Analysis
Employee	- Pending Tasks - Status of their domains

2.3. Definitions and acronyms

i. Definitions

Keyword	Definitions
ES	Elastic Search
POJO	Plain Old Java Objects
DAO	Data Access Objects
JDBC	JAVA Database Connectivity

CRUD	Create, Read, Update, Delete
DOM	Document Object Model
JSON	JavaScript Object Notation
REST	Representational State Transfer

ii. Acronyms and abbreviations

Acronyms or abbreviations	Definitions
IDE	Integrated development environment
API	Application Programming Interface, a set of protocols or standards for communicating with web-based applications
JSON format	Data-interchange format
RESTful API	An API that uses a standard set of HTTP requests
HTML 5	HyperText Markup Language; the fifth and current version of the HTML standard
CSS3	Cascading Style Sheets; language used to describe the presentation of a document written in markup language, e.g., HTML
JavaScript	Programming language used extensively in website development
Git version control	Free and open-source version control system
State Management	Management of the state of one or more user interface controls such as text fields, ok buttons, radio buttons, etc. in a graphical user interface.

2.4. References

SNo.	Description	Reference Link
1.	Product Requirement	https://docs.google.com/document/d/13QPJqisLw7BkPiZDnJntQKgTHEnActEiohYh-ngAxTY/edit#heading=h.2b9z52qifdh7
2.	Java specifics	https://www.oracle.com/java/
3.	Elastic search guide	https://www.elastic.co/guide/index.html
4.	Spring boot	https://spring.io/projects/spring-boot
5.	React Js	https://reactjs.org/
6.	Chart Js	https://www.chartjs.org/
7.	Redux	https://redux.js.org/

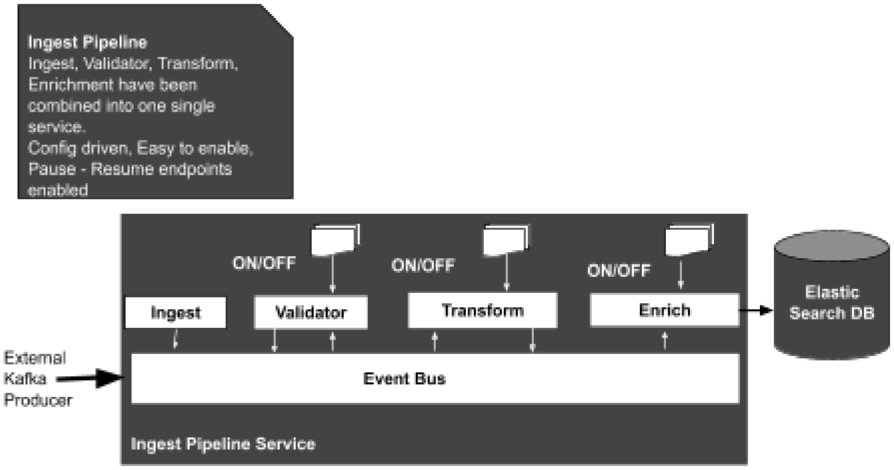
3. System Overview

3.1. Infrastructure Services

Infrastructure services have been provided to all applications with a view to reducing time, cost and risks of development through re-use. The requirements of all current applications and the anticipated requirements from future applications have been analysed through design of the services. The Infrastructure Services was built incrementally, implementing the most common requirements first, followed by more specialised services. Followed include:

i. Error Handling
ii. Performance monitoring
iii. Debugging
iv. Logging
v. Security

3.2. Ingest Pipeline Features



Ingest Pipeline
Ingest, Validator, Transform, Enrichment have been combined into one single service.
Config driven, Easy to enable, Pause - Resume endpoints enabled

4. System context

Two types of data are flowing into the system.

i. Get data from elasticsearch index, migration tool will re-index it to new index. As shown in Diagram 1. This a bulk one time operation. Based on specific conditions (direct push) it post the processed data to ES new index or to a topic in pipeline. Also for live data the system receives data in a topic and passes it to the same reindex pipeline.

ii. Excel data upload into System, store it in a new Index.
As shown in Diagram 8.

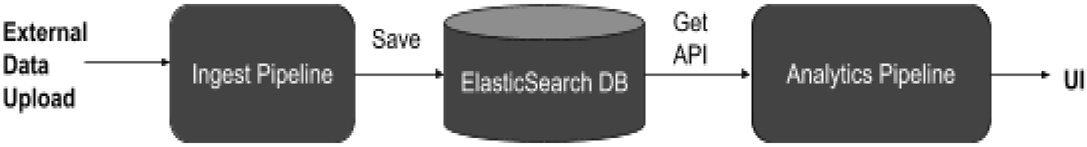


Diagram 8

5. System Design

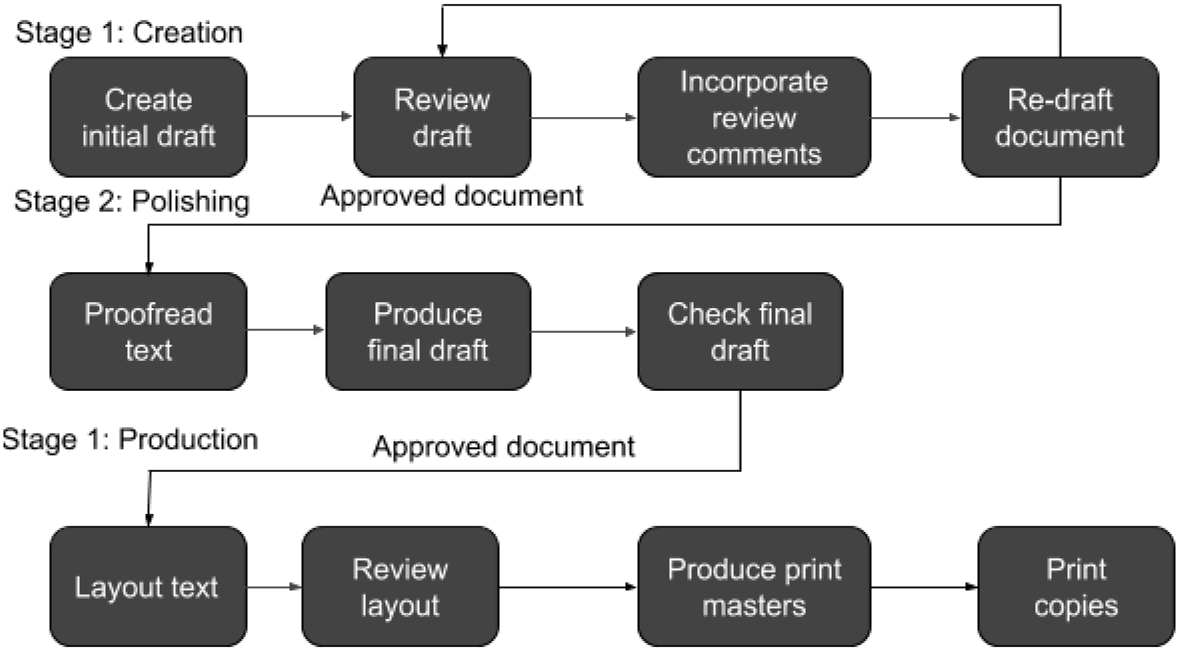
5.1. Design Method and Standards

Design and Analysis
Progress

- **User Stories**
    - User Stories template with basic user stories were recorded and shared across.
    - [Product requirement]

- **Data Source - Visualization Mapping**
    - At each visualization level, we have captured the data source indexes from where the data has to be consumed.
    - The logic which drives the data
    - The parameter which is responsible for getting the insight
    - Aggregation Factors on each of these parameters
    - [Data Source - Viz Mapping]

5.2. Documentation Standards



5.3. Naming conventions

The following format specifies the naming conventions of document:

Project Name_Technical Document_Document Stage Version.

5.4. Programming Standards

i. Functional and Structural
   - Write modular and reusable code so that it is easy to read and debug as well.
   - Avoid tightly coupled code.
   - Grouping of functional classes into meaningful packages.
   - Use Loggers and ResponseFactory to communicate to log files and responses

ii. Avoid Select * from SQL Queries.

iii. Remove unnecessary Try Catch Blocks.

iv. Move the repeated text or values to a Constant .

v. Not to assume anything which is coming from UI. Always validate. Sql Injection is a security threat.

vi. Always import specific classes from a package. Not the whole package.

vii. Use formatting tools to format the java code as per the standards.

5.5. Software development tools

i. Tech Stack Versions

Technologies	Version
Spring Boot	2.1.8
Elastic search	6.6.1
Apache Kafka	2.11
Reach JS	16.10.2
Chart Js	2.8.0
Redux	4.0.4
Material design	4.5.1

ii. UI Major Module

Major Module	Open Source React Module
Share	Mailto, Whatsapp API
PDF Generation	Dom-to-image
Chart	ChartJS
Storing Image	React S3
State Management	Redux
Fonts	Font Awesome
Export CSV	XLSX

5.6. System response

i. Response codes

API will respond with standard HTTP response codes appropriate to the result of the request. While the exact meaning of the code varies depending on the request, the general rules are:

SNo.	Code	Description
1.	200	A response code of 200 means the request was successful and details about the response can be found in the body of the response.
2.	201	The requested POST operation was successful and an object was created in the system.
3.	202	The requested operation has been accepted and the body contains information about an asynchronous job you can query to check on the progress of the request.
4.	401	The request sent by the client could not be authenticated.
5.	404	The requested resource does not exist.
6.	500	API failed to process the request because of an error inside the system.

ii. Response entities

- All GET methods respond with the JSON of the resource(s) being requested. HEAD methods have no response entity.
- POST methods may respond with a 201 CREATED or 202 ACCEPTED response code depending on whether the creation completed immediately or is an asynchronous operation. If the resource was created immediately, API should provide a JSON entity that includes the new resource’s unique ID.
- PUT and DELETE methods generally respond with 204 NO CONTENT unless the operation is a long lived operation. In those scenarios, the PUT will respond with a 202 ACCEPTED response code.

5.7. Data pipeline

Ingest/Data pipeline architecture comprises 4 modules ingest, validate, transform and enrich. Pipeline consumes the collection/incoming data and transforms the data using jolt configurations.

i. Transformation and enrichment

- Using the Transformation Schemas the incoming data gets transformed.
- Enrichment schemas for each of the modules gets loaded on application startup. Using this, Enrichment of data is taken care

ii. Preprocess and indexing

- Post Transformation and Enrichment, obtained objected is transformed to Domain level object
- Enriched in the Collection Object
- Preprocessed Data gets pushed to ElasticSearch in a new index.

5.8. Analytics

This provides dashboard configurations for UI representation. It fetches the data and translates it into a generalised response readable to client/UI.

i. Dashboard configuration

- This configuration is used to get group of visualisation for a specific dashboard
- On request from UI for a specific dashboard, this configuration is used for providing the group of visualisation required for the dashboard
- Then the response consumed by the UI to paint the UI layout

ii. Build generic response

- This provides dashboard configurations for UI representation. The request for a configuration is received the specific dashboard configuration is provide to the client/UI
- Once a request for a specific visualisation code/chart is received, the application reads the chart configuration and identifies the index, aggregation queries and build the final query to fire against the ES index. It fetches the data and translates it into a generalised response readable to client/UI.

6. Configurations

The configurations are broadly of two kinds. One is for data pipeline and the other one is for analytics. Below are listed for data pipeline and analytics.

6.1. Data pipeline configurations:

i. Topic Context Configurations
ii. Validator Schema
iii. JOLT Transformation Schema
iv. Enrichment Domain Configuration
v. JOLT Domain Transformation Schema

6.2. Analytics application configurations

i. Chart API Configuration
ii. Master Dashboard Configuration
iii. Role Dashboard Mappings Configuration

This configurations are explained in more detail in DSS backend configuration manual

7. Static Data Upload

Static data upload is used for uploading target data.

7.1. Excel data upload and parsing configurations:

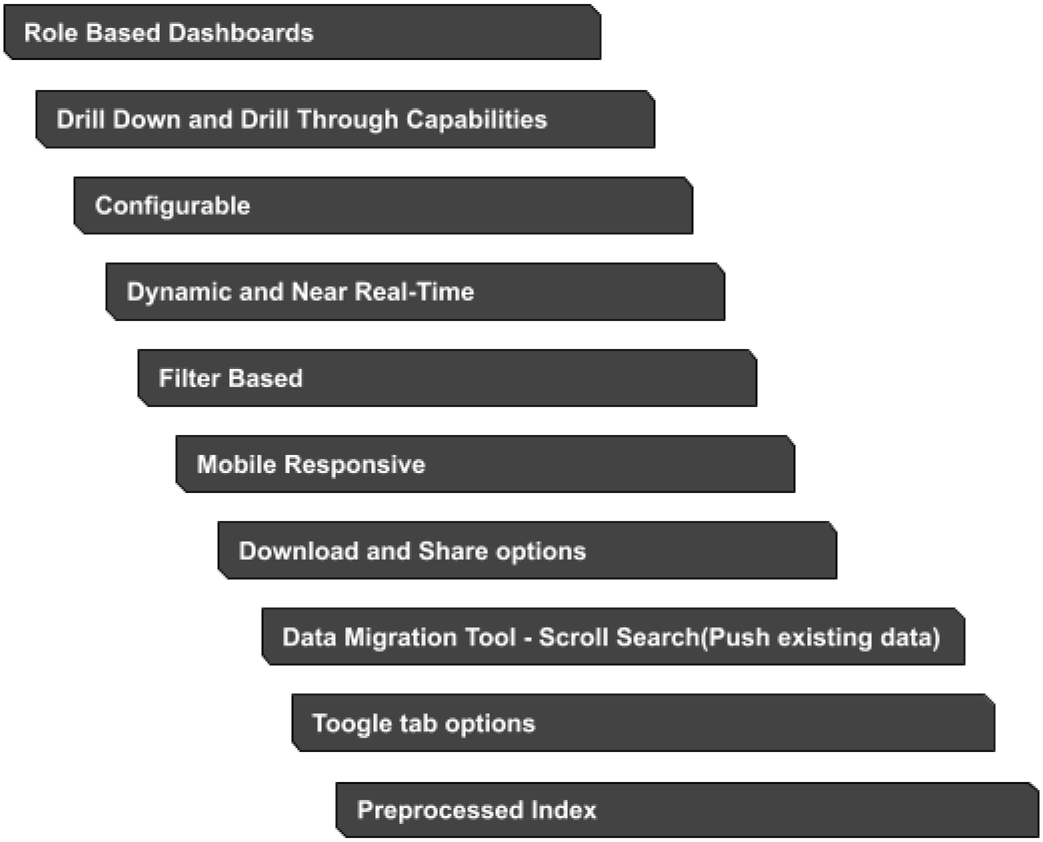
i. Excel data is uploaded.
ii. Based on configuration, the excel gets parsed and pushed to pipeline.
iii. Received on a kafka consumer.
iv. Data is saved in Elastic search new index.

7.2. APIs For Easy Retrieval

i. APIs are exposed to easily fetch the static data which has been uploaded.

8. Application Feature

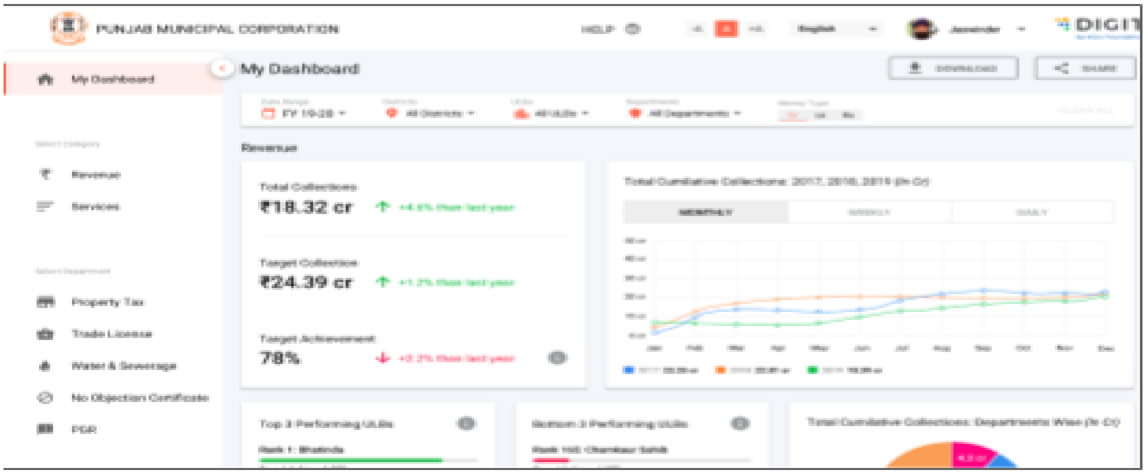
The application features for DSS ingest pipeline and dashboard analytics are as below.



- Role Based Dashboards
- Drill Down and Drill Through Capabilities
- Configurable
- Dynamic and Near Real-Time
- Filter Based
- Mobile Responsive
- Download and Share options
- Data Migration Tool - Scroll Search(Push existing data)
- Toogle tab options
- Preprocessed Index

9. Wireframes & Visual Designs

Wireframe and UI/UX designs and visuals are below



My Dashboard

Revenue

Total Collections ₹18.32 cr

Target Collection ₹24.39 cr

Target Achievement 78%

Total Cumulative Collections: 2017, 2018, 2019 (in Cr)

Top 3 Performing ULBs

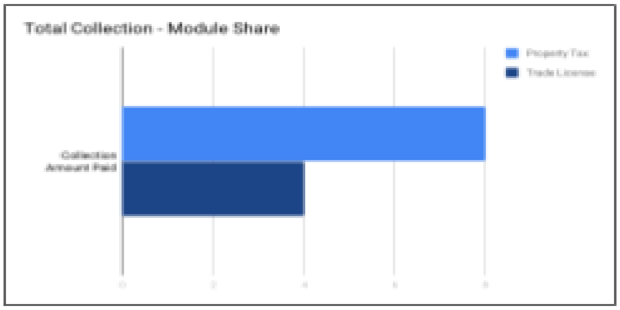
Rank 1: Bhatinda

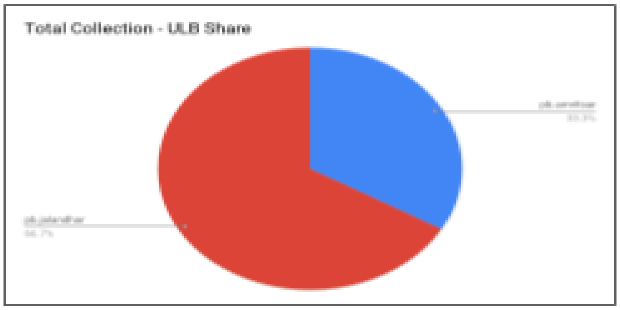
Bottom 3 Performing ULBs

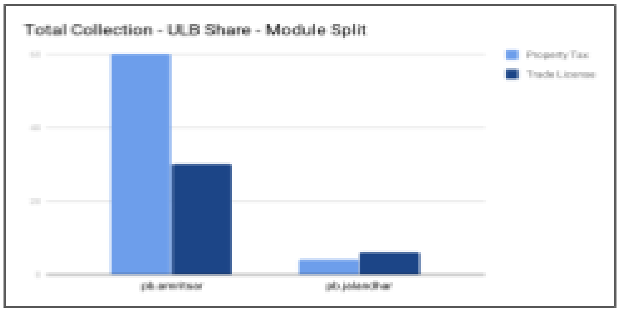
Total Cumulative Collections: Departments Wise (In Cr)

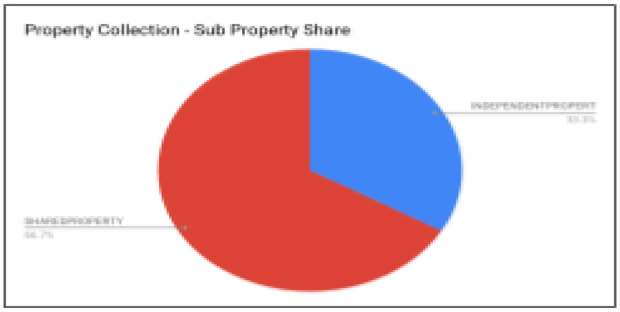
[Wireframes - Sketch Preview](#) contains information for wireframe

Flow and Visualizations


Total Collection - Module Share


Total Collection - ULB Share


Total Collection - ULB Share - Module Split


Property Collection - Sub Property Share

10. Document Control

Title:	DSS Technical Design Document
Issue:	Issue 1
Date:	
Author:	
Distribution:	
Reference:	
Filename:	DSS_Technical_Design_Document_Initial_Draft.docs

11. Document Sign Off

Nature of Signoff	Person	Signature	Date	Role
Authors				Project Member
Reviewers				

12. Document Change Record

Date	Version	Author	Change Details
25-11-2019	0.01	Rahul Sahu	Initial Draft
13-03-2020	0.02	Pritha C	Modified with internal review comments from Tarento.