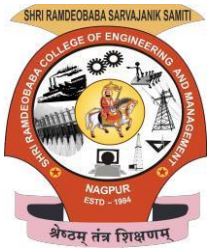
DAG Designer

Internship Report
submitted to
Shri Ramdeobaba College of Engineering & Management, Nagpur
in partial fulfilment of requirement for the award of
degree of

Bachelor of Engineering
In
Computer Science and Engineering

By
Ruchit Bhardwaj

Guide
Prof. D. A. Borikar



Computer Science and Engineering
Shri Ramdeobaba College of Engineering & Management,
Nagpur 440 013
(An Autonomous Institute affiliated to Rashtrasant Tukdoji Maharaj Nagpur University Nagpur)

April 2019

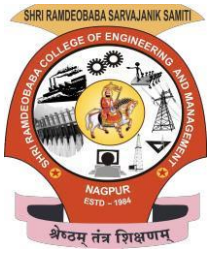
DAG Designer

Internship Report
submitted to
Shri Ramdeobaba College of Engineering & Management, Nagpur
in partial fulfillment of requirement for the award of
degree of

Bachelor of Engineering
In
Computer Science and Engineering

By
Ruchit Bhardwaj

Guide
Prof. D. A. Borikar



Computer Science and Engineering
Shri Ramdeobaba College of Engineering & Management,
Nagpur 440 013
(An Autonomous Institute affiliated to Rashtrasant Tukdoji Maharaj Nagpur University Nagpur)

April 2019

SHRI RAMDEOBABA COLLEGE OF ENGINEERING & MANAGEMENT, NAGPUR

(An Autonomous Institute Affiliated to Rashtrasant Tukdoji Maharaj Nagpur University Nagpur)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Internship Report on **"DAG Designer"** is a bonafide work of **Ruchit Bhardwaj** submitted to the Rashtrasant Tukdoji Maharaj Nagpur University, Nagpur in partial fulfilment of the award of a Bachelor of Engineering , in Computer Science and Engineering has been carried out at **ZS Associates India Pvt. Ltd., Pune** during the academic year 2018-2019.

Date:
Place:

Mr. Gaurav Chopra
SD Manager
ZS Associates

Dr. M. B. Chandak
H.O.D
Department of Computer Science and Engineering

Prof. D. A. Borikar
Internal Guide
Department of Computer Science and Engineering

Dr. Anupam Kher
Dean
Training & Placement

DECLARATION

I, hereby declare that the project titled **"DAG Designer"** submitted herein, has been carried out in ZS Associates India Pvt. Ltd, Pune. The work is original and has not been submitted earlier as a whole or part for the award of any degree / diploma at this or any other institution / University.

Date:

Place:

Ruchit Bhardwaj

Roll No. 55

ACKNOWLEDGEMENTS

I take this opportunity to acknowledge our profound indebtedness and extend our deep sense of gratitude to my coach and mentor **Mr. Gaurav Chopra** and the Head of Department **Dr. M. B. Chandak** for his support and guidance for our project. We would also like to express our gratitude to our project internal guide **Prof. D. A. Borikar**, for his valuable guidance and advice during the internship and throughout the course of the project that has helped us immensely for the completion of this project.

I would specially thank my team members at ZS Associates India Pvt. Ltd. for guiding me during my internship period in helping me to understand and contribute towards the building and development of the project.

My thanks and appreciations also go to our colleagues in developing the project and people who have willingly helped us out with their abilities.

Project By:

Ruchit Bhardwaj [55]

ABSTRACT

As we all know, medical representatives (or Sales Representatives, as they are referred to in the United States) are an integral part of the healthcare profession. They provide valuable insights as to which new medicines have arrived in the market and thus serve as a key point of contact between the pharmaceuticals and medical companies and healthcare professionals. Hence, these reps are expected to increase the business for their companies and thus intelligent suggestions need to be provided to these Sales Reps. These suggestions are based on certain triggers and are coded in Excels. However, the coding mechanism in Excels is quite tedious and time consuming, not to mention error prone all the same.

This is where DAG Designer comes into picture. DAG Designer allows the user to create logical flows specific to the need of the user to generate suggestions and visualize the flow to get a better perception about it. This reduces the risk of errors and helps to give the users a better insight to the complete flow as such. It basically serves as a replacement for the mammoth excels thereby saving time and reducing the risks of errors by providing a visual representation of what the user wishes to portray. This report covers the basic architecture design of the project and dives into how a basic 'flow' is created, saved and executed from scratch. The same mechanism can then be used to create flows for suggestion generation.

Finally, the report discusses the internship work accomplished during the internship period and the various technologies that I got to work on and the technologies that the project was built upon.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

As we all know, medical representatives (or Sales Representatives, as they are referred to in the United States) are an integral part of the healthcare profession. They provide valuable insights as to which new medicines have arrived in the market and thus serve as a key point of contact between the pharmaceuticals and medical companies and healthcare professionals. Hence, these reps are expected to increase the business for their companies and thus intelligent suggestions need to be provided to these Sales Reps. These suggestions are based on certain triggers and are coded in Excels. However, the coding mechanism in Excels is quite tedious and time consuming, not to mention error prone all the same. This is where DAG Designer comes into picture. DAG Designer allows the user to create logical flows specific to the need of the user to generate suggestions and visualize the flow to get a better perception about it. This reduces the risk of errors and helps to give the users a better insight to the complete flow as such. It basically serves as a replacement for the mammoth excels thereby saving time and reducing the risks of errors by providing a visual representation of what the user wishes to portray. This report covers the basic architecture design of the project and dives into how a basic 'flow' is created, saved and executed from scratch. The same mechanism can then be used to create flows for suggestion generation.

1.1 OBJECTIVE OF THE PRODUCT

- To design a tool that can provide users with a visual representation of the flow that they are creating thereby reducing errors.
- To provide a tool that would allow the users to save time by not coding the suggestion mechanism in lengthy excels but rather using specialized nodes in the DAG Designer for the same.
- Provide easy to handle mechanism to improve the overall efficiency of the application.

1.2 NEED OF YOUR PROJECT

In the current scenario, the basis on which suggestions are generated to be then given to the sales reps are coded inside excels. These excels are mammoth in nature and are thus prone to error. Tracing down those errors in an excel is quite time consuming and inefficient all the same. Hence, the DAG Designer was thought of. The idea of DAG Designer was to provide visual aid to the users by entrusting them with specialised nodes to perform certain tasks. This allowed the users to be more efficient in their work since the errors were reduced and so was the time required to identify those errors, if any.

1.3 FEATURES

- ***Visual Representation***: Various nodes are provided in the application to perform specified tasks which fall into categories such as Source Nodes, SQL Nodes, Action Nodes etc.
- ***Easier execution mechanism:*** Once the flow is created, saved and committed, a single button click allows the complete flow to execute. Highest level of abstraction for the end user.
- ***Customizable***: Based upon the user preferences, the application allows the user to customize the way in which the flow execution takes place.

1.4 BENEFITS

- ***Less error prone****:* A visual representation of the nodes allows the complete application to be error free as compared to the scenario wherein suggestion generation was coded inside lengthy excels
- ***Time Saving****:* Less errors would mean less time spent in finding those errors. Thus, this would allow for less time spent on error hunting and more time utilization for constructing and executing logical flows.
- ***User Friendly***: With state of the art UI, the DAG Designer application is user friendly by providing the users a rich experience in terms of interaction with the application.
- ***Two-step authentication****:* The details of the configuration used by an agent to execute flows is tokenized providing the user with an extra level of security.

CHAPTER 2. REVIEW OF LITERATURE

2.1 PROJECT DOMAIN

DAG Designer, at its core, can be referred to as an ETL (Extract-Transform-Load) tool. It simplifies the ETL process to such an extent that much complex tasks of suggestion generation can be modularised into small components each of which can be then executed in an efficient manner.

However, the domain of technologies that DAG Designer uses is discussed briefly in the following section.

2.2 TECHNOLOGIES/TOOLS USED

1) ***Java***

   Java is a general-purpose programming language that is class-based, object-oriented, and designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to "bytecode" that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture. The syntax of Java is similar to C and C++, but it has fewer low-level facilities than either of them.

2) ***Scala***

   Scala is a general-purpose programming language providing support for functional programming and a strong static type system. Designed to be concise, many of Scala's design decisions aimed to address criticisms of Java. Scala source code is intended to be compiled to Java bytecode, so that the resulting executable code runs on a Java virtual machine. Scala provides language interoperability with Java, so that libraries written in either language may be referenced directly in Scala or Java code. Like Java, Scala is object-oriented, and uses a curly-brace syntax reminiscent of the C programming language.

3) ***JavaScript***

   JavaScript often abbreviated as JS, is a high-level, interpreted programming language that conforms to the ECMAScript specification. JavaScript has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions. Alongside HTML

and CSS, JavaScript is one of the core technologies of the World Wide Web. JavaScript enables interactive web pages and is an essential part of web applications. The vast majority of websites use it, and major web browsers have a dedicated JavaScript engine to execute it. As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has APIs for working with text, arrays, dates, regular expressions, and the DOM, but the language itself does not include any I/O, such as networking, storage, or graphics facilities. It relies upon the host environment in which it is embedded to provide these features.

4) Angular JS

AngularJS is a JavaScript-based open-source front-end web framework mainly maintained by Google and by a community of individuals and corporations to address many of the challenges encountered in developing single-page applications. It aims to simplify both the development and the testing of such applications by providing a framework for client-side model–view–controller (MVC) and model–view–view-model (MVVM) architectures, along with components commonly used in rich Internet applications. (This flexibility has led to the acronym MVW, which stands for "model-view-whatever" and may also encompass model–view–presenter and model–view–adapter.)

5) ***Database-Postgres***

PostgreSQL (also referred to as Postgres) is an open-source relational database management system (RDBMS) emphasizing extensibility and standards compliance. It can handle workloads ranging from single-machine applications to Web services or data warehousing with many concurrent users. It is the default database for macOS Server, and is also available for Linux, FreeBSD, OpenBSD, and Windows. PostgreSQL is ACID-compliant and transactional.

6) ***Amazon Web Services***

Amazon Web Services (AWS) is a secure cloud services platform, offering compute power, database storage, content delivery and other functionality to help businesses scale and grow. Explore how millions of customers are currently leveraging AWS cloud products and solutions to build sophisticated applications with increased flexibility, scalability and reliability.

CHAPTER 3. SYSTEM ARCHITECTURE AND COMPONENTS

3.1 JARGONS

- ***Instance:*** An instance is an isolated workspace for a specific client where users can build and execute flows. An instance can be referred as a collection of different projects. So, different clients can build and execute flows on their respective instances

- ***Environment:*** It is an environment for developing and executing flows in a project. When an instance is created in DAG Designer, a Development environment gets created automatically.

  Development Environment: This is where the flows will be developed and edited. Flows can be saved, committed and executed for testing purpose in this Development Environment.

  Execution Environment: Once a project has been tested and released in the Development Consumer App, the respective release will be available in the Execution Consumer App. Execution environment should be used only for running the final flows and editing of flows is not supported here.

- ***Project***: Various projects can be created within an instance. It can be considered analogous to a folder in the conventional file system wherein various mappings (sub-folders/files) are present which contain the actual flows.

- ***Mapping:*** Inside each project, various mappings can be created in which flows will be designed. Every mapping can have child mappings as well and proper hierarchy can be designed.

- ***Flows:*** A logical combination of nodes to perform any desired ETL operation.
- ***Job:*** It is a mechanism of creating and registering a flow category node for execution.

- ***Job Definition*****:** It is a way of registering the job with Flow nodes (nodes that trigger an action) in a mapping. Job Definition creation is a one-time process. The user can then execute this definition numerous times.

- ***Strategy:*** SWLH strategy can be dubbed as a configuration that describes a way in which the user wants to configure the agent (i.e. the executor) on which those jobs will be executed. Flow Builder allows users to configure different instances with different strategies based on the need of the client.

  DAG Designer currently supports two SWLH Strategies -

  - ***Global Strategy:*** A master strategy config that will be applicable across all the instances and consumer apps present in the application that have subscribed to Global Saas
  - ***Per Instance:*** This strategy config allows for more user control and the ability to tweak the settings as and when needed. The strategy will be applicable across all the consumer apps present in the instance that have subscribed to Per Instance strategy.

3.2 ARCHITECTURE

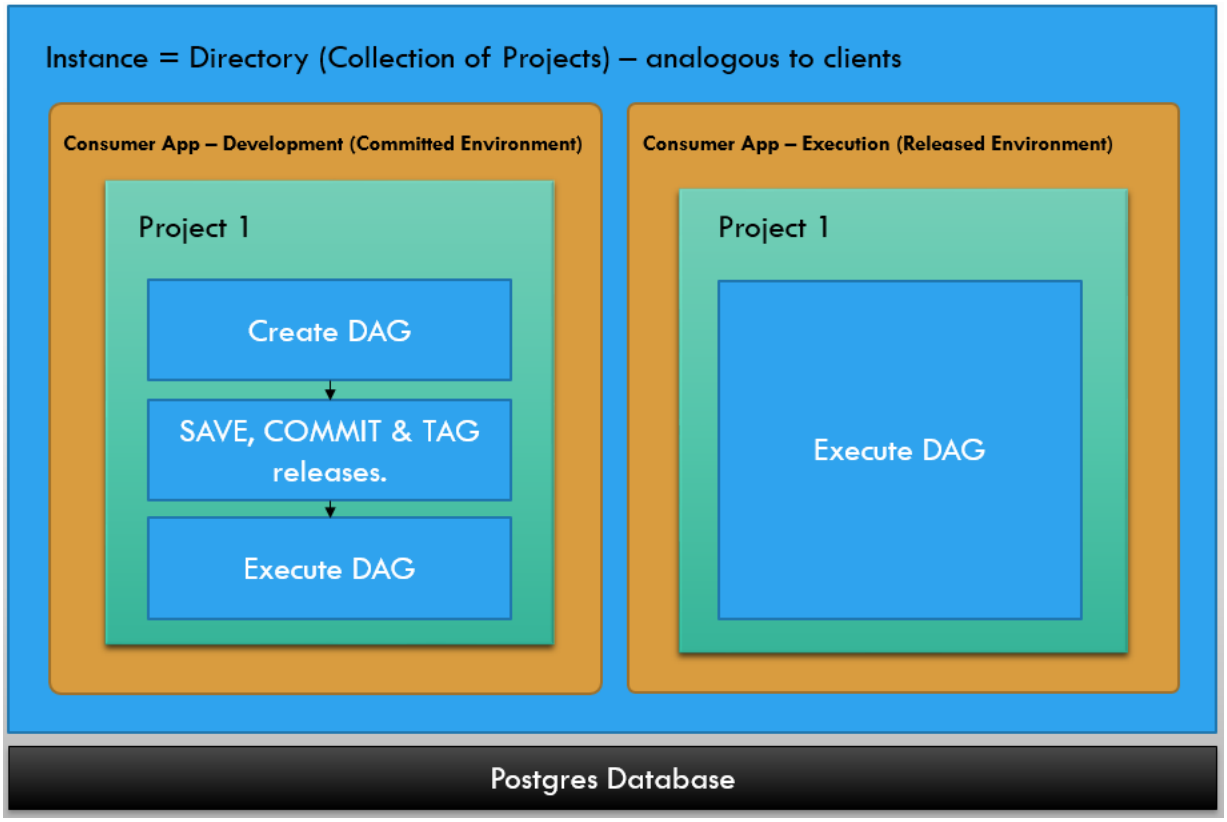


Figure 1: Architecture at an abstract level

3.3 MODULES

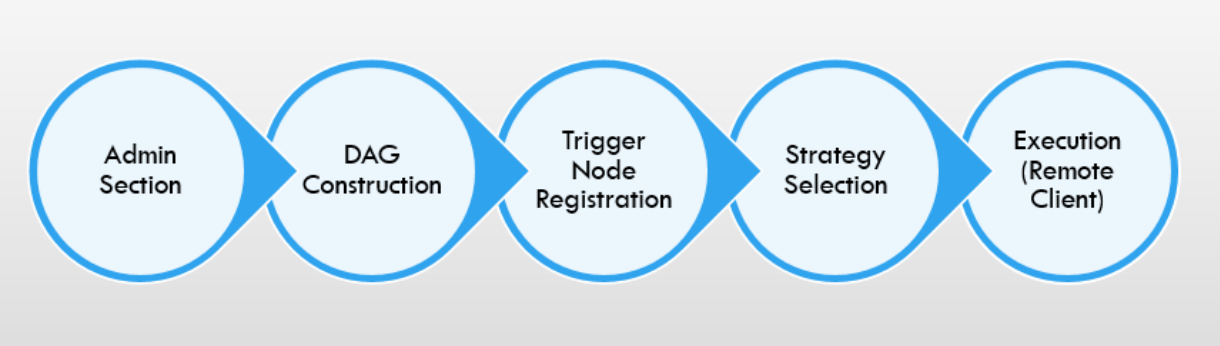


Figure 2: Modules of DAG Designer (End-To-End)

1. **Admin Section**

   In this section, the user, based upon his/her user roles, creates an instance or a consumer app. In other words, it creates a directory and inside that directory, the user can create various environments for developing flows or explicitly for executing flows.

2. **DAG Construction**

   After the creation of the environments, if a development environment is created, then the user creates various projects in which s/he will finally create flows for ETL operations. This phase is known as the DAG Construction.

3. **Trigger Node Registration**

   Once the DAG has been created, saved and committed, the user then must register a node which will serve as the trigger node. In other words, once the flow is executed, this trigger node will be the one that the application will search for, to start execution.

4. **Strategy Selection**

   Based upon the strategies explained in the previous section, the user then selects one of the strategies best suited for his/her needs and saves the changes.

5. **Remote Client Execution**

   This is the most crucial step with the help of which the saved flows are then executed remotely. The application uses AWS EMR clusters for computation purposes and thus achieves remote execution.

CHAPTER 4. SYSTEM DESCRIPTION

4.1 WORKING OF THE PROJECT

Consider the following excel -

Node	Parent	Operation	Contained_Action
Source			
SQL	Source	select * from tablename	
Write	SQL		
Trigger			Write

Figure 3: A typical excel

When a trigger node has been registered using a job, in this case, the node named 'TRIGGER', the application searches for the corresponding Contained_Action attribute. It finds that a node named 'WRITE' is in the contained action attribute so it finds the node with the given name. Now, from this point forth, the application searches for the corresponding 'PARENT' attribute. So, for the node named 'WRITE', its corresponding parent attribute is 'SQL'. The application then searches for the parent of node named 'SQL' and finds that a node named 'SOURCE' is the parent of the given node. It searches for the aforementioned node and finds that the parent of this node is NULL. Hence, the application assumes this node named 'SOURCE' to be the starting point of the flow and hereafter, utilizes the top-down approach of flow execution.

Thus, in reality, the application searches for a node which has no parent i.e. parent equal to NULL and once such a node is found, assumes it to be the starting node of the flow and starts the flow with this node. It then performs the given ETL operations unless the complete execution is successful.

4.2 PROJECT FLOW

In this section, an end to end run of the complete application is explained.

1. At the offset, a user, based upon his/her user roles, creates either an instance or an environment. Once done with this, next, the user creates a project and within the created project, the user creates a mapping. Once the mapping has been successfully created, the user can now create flows for performing basic operations.
2. Inside the newly created mapping, the user is provided a canvas wherein the user can utilize the pre-existing nodes to perform certain operations and to create a logical flow using these nodes. Some of the categories of nodes are 'Source', 'SQL', 'Action' etc.

3. Once a logical flow has been created, the user then saves and commits the flow so that it is then visible to all the other users of the same instance or consumer app.
4. Once the flow is successfully saved and committed, the user then registers one or many trigger nodes during the job definition creation. These nodes will serve as the starting points when a flow is executed. Simultaneously, the user also selects a strategy best suited for his/her requirements. The strategy can be either ‘Global SaaS’ or ‘Per Instance’.
5. Now, an agent is activated. The agent uses AWS EMR clusters for leveraging its computational facilities. Once the agent is up and running, the user then clicks on the ‘RUN’ button to execute a flow. The application and the agent (backend) club together to execute the flow successfully.
6. These steps complete one end-to-end cycle of execution of flows.

4.3 TOKENIZATION

Once the agent is initialized, how does the information from the DAG Builder application gets conveyed to the agent? The answer lies in tokenization. Refer the diagram below –

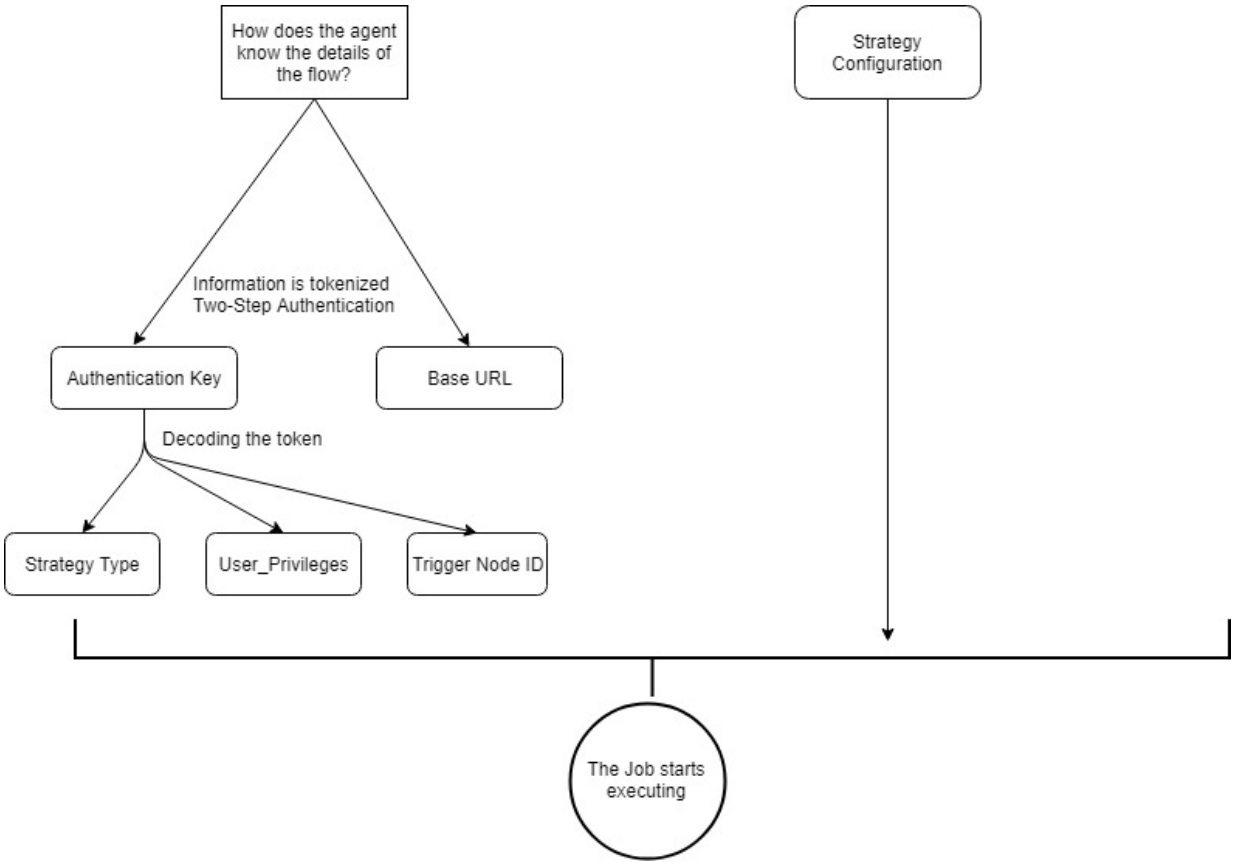


Figure 4: Flow Diagram for Job Execution

The information that has to be passed from the application to the agent is tokenized. Once decoded, we find that it comprises of the authentication key and the base URL. When the authentication key is again decoded, we find that is comprises of several details such as the user role of the user that is currently logged into the application, the type of strategy that the user has selected and based of it, what are the dependencies that the agent has to resolve and also, what is the ID of the trigger node of the flow from which flow execution has to start. This is only a handful of detail that can be decoded from the token. This complete information packet along with the strategy configuration helps the agent to execute the jobs successfully.

4.4 APPLICATIONS

On the offset and at its core, DAG Designer is an ETL Tool which will be further used for suggestion generation for reps and in various other projects within our organization. Due to its ease of operation and visual representation, it is being accepted as a staple tool for ETL purposes where one can also perform complex computations. The ease of communicating with databases like Oracle, Postgres and applications like Salesforce adds on the usefulness and serves as a multi-purpose tool which can be used by other applications.

CHAPTER 5. RESULTS

SCREENSHOTS OF THE PROJECT

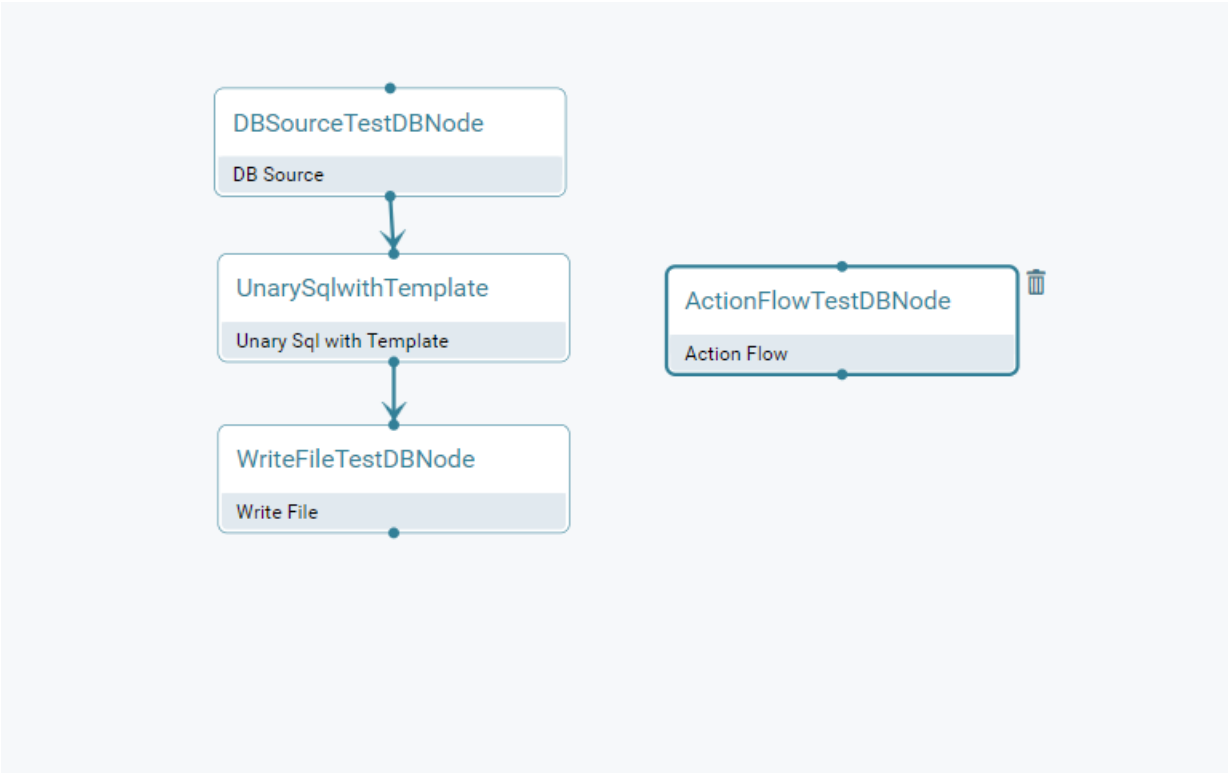


Figure 5: A sample logical flow

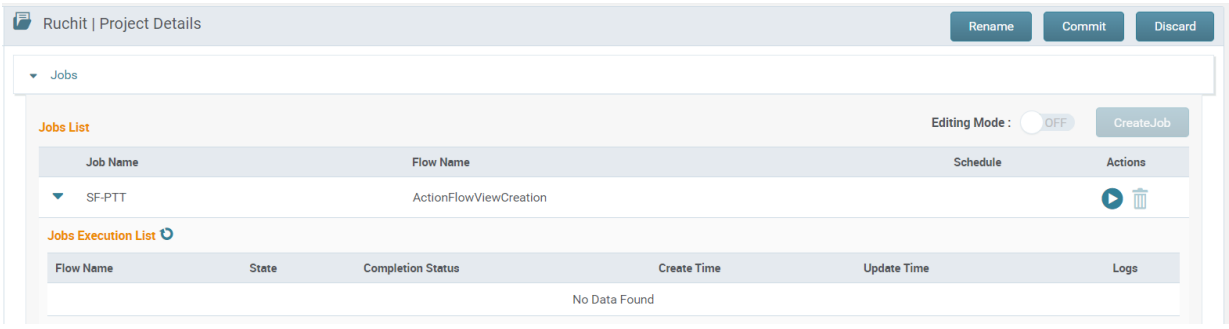


Figure 6: Job Definition creation and trigger node registration

CHAPTER 6. INTERNSHIP WORK

6.1 PROJECT INTERN ROLE

During the first month of my internship, a large amount of time was spent in getting training about the way in which ZS worked. Also, we were familiarized with the tools used for testing purposes during this month. Once we had been assigned projects, the main aim was to get a detailed understanding of the project and increase the product knowledge so that I can contribute more to the project and be of assistance to the team.

The second month of internship primarily focussed on automation testing. Automation was achieved for some modules of the project. Also, around this time, I had to create master flows that consisted of all the nodes present in the application to check whether each node worked properly as expected and did not misbehave.

In the third month, the feature of remote client execution was developed and now my focus was on the strategy implementation and remote client execution. In this month, I also started studying about BDD Cucumber framework and will be continuing with this after the internship as well

Test case writing in Qtest was a continuous task which was later used during regression purposes. During the last few weeks of internship, I was also involved in the regression of the project, thus understanding the full cycle of product development.

Thus, this summarises my internship role during the three months of internship.

6.2 QTEST

Qtest is a test case management tool that allows Quality Assurance Engineers to document test cases within test scenarios. This began with each sprint cycle whenever a new feature was brainstormed. Once the requirements were specified by the user, it was my job to document the detailed description of how the application should behave given such circumstances and how will the application be tested.

6.3 QUALITIA – AUTOMATION

Qualitia is a wrapper over Selenium Web Driver which is used for automation purposes. It was my job to automate certain modules of the project and it was done with the help of Qualitia. Consider the following diagram –

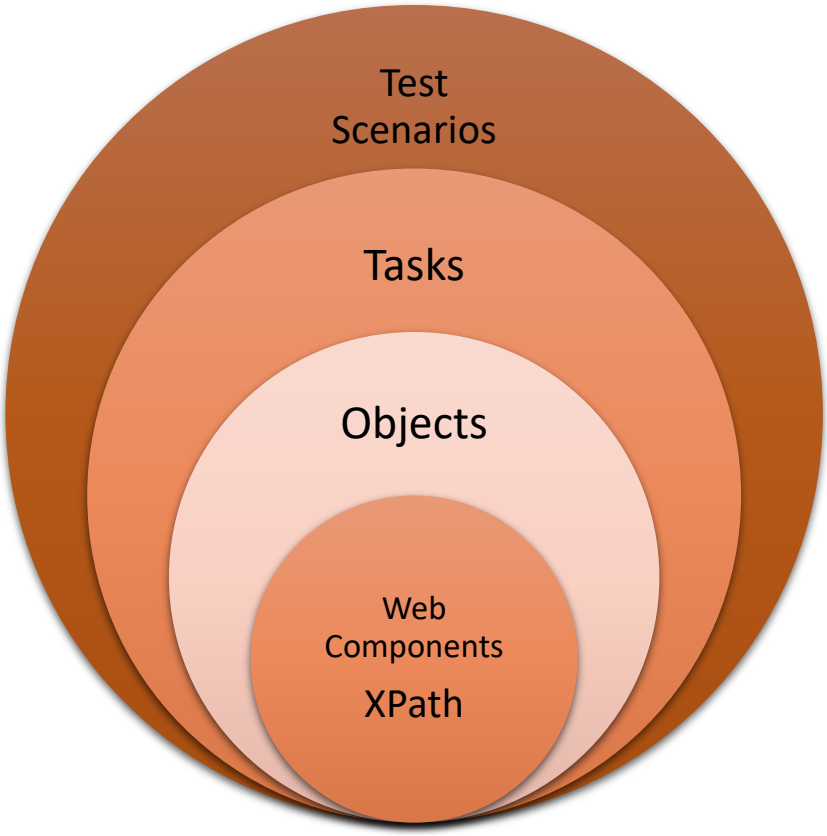


Figure 7: Qualitia Overview

Initially, web components are considered, the XPATHs of which are used. The web components group together to create Objects. The group of objects then create tasks which are in turn used to create test scenarios. It is these test scenarios which are eventually run in order to run the automated test case module.

6.4 SPRINT CYCLE

Consider the following pictorial representation –

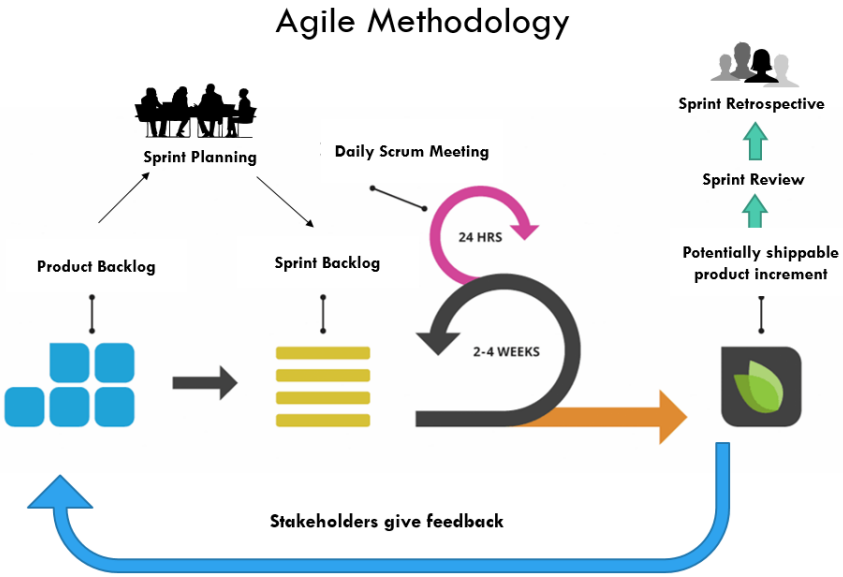


Figure 8: Agile Methodology (A Sprint Cycle)

REFERENCES

Books

- Software Testing Techniques (2nd Edition, Boris Beizner)
- Testing Object-oriented Systems: Models, Patterns, and Tools (Robert V.Binder)
- Lessons Learned in Software Testing: A Context-Driven Approach (Cem Kaner)
- How to Break Software: A Practical Guide to Testing (James Whittaker)

Research Papers

- Nicey Paul, Robin Tommy, "An Approach of Automated Testing on Web Based Platform Using Machine Learning and Selenium", *Inventive Research in Computing Applications (ICIRCA) 2018 International Conference on*, pp. 851-856, 2018

Hyperlinks

- Information about various technologies

  https://www.wikipedia.com

- Tutorial files related to Qualitia and BDD Cucumber

  https://www.toolsqa.com/cucumber-tutorial/

  https://www.qualitiasoft.com/

- For application testing tools and error solving purposes
  - www.w3schools.com
  - stackoverflow.com
  - www.wikipedia.com
  - www.cocodataset.com