CONTENT MANAGEMENT SYSTEM WITH ADOBE EXPERIENCE MANAGER

Industrial Project Report submitted in fulfillment of requirement of the Degree

BACHELOR OF TECHNOLOGY

By

ARPIT RATHI

(133202)

Department of Computer Science and Engineering (CSE)

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

WAKNAGHAT, SOLAN (H.P)

DECLARATION BY THE SCHOLAR

I hereby declare that this project report titled '**Content Management system with Adobe Experience Manager**' submitted at **Jaypee University of Information Technology, Waknaghat, Solan** is an authentic record of my work carried out under supervision of **Ms. Ruhi Mahajan.** I have not submitted this work elsewhere for any other degree or diploma. I am fully responsible for the contents of my B.Tech. Industrial Project.

(Student signature)

Arpit Rathi, 133202

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)
Ms. Ruhi Mahajan
Department of computer science and engineering
Dated:

ACKNOWLEDGEMENT

This internship opportunity at Accunity Software is a great chance for learning and professional development. I would like to express my deepest gratitude and special thanks to the founder and CEO of the company Mr. Ankur Mittal who in spite of being extraordinarily busy with his duties, took time out to hear, guide and keep me on the correct path of learning and developing.

I express my deepest thanks to Mr. Pankaj Bansal, Managing Partner and Chief Technical Officer for taking part in useful decision & giving necessary advices and guidance and arranged all facilities in the office.

It is my radiant sentiment to place on record my best regards, deepest sense of gratitude to my mentors, Mr. Nishant Gupta, Technical Delivery Manager, Mr. Saurabh Gupta, Software Developer, Ms. Shivani Garg, Software Developer for their careful and precious guidance which were extremely valuable for my study both theoretically and practically.

I would like to acknowledge guidance of my institute mentor, Ms. Ruhi Mahajan who constantly guide me during my internship and suggest me to improve on every aspect.

TABLE OF CONTENT

ABOUT THE COMPANY (ACCUNITY SOFTWARE)

Accunity Software is a new age digital services startup which focus to uniquely combine client's idea and requirement with power of Technology, Analytics, Marketing, & Content for digital transformation. Being a reliable web design agency, Accunity transforms your vision into brand building promotional entity. The company stroke down the best ideas or start from a blank page to engineer an artifact enabling the client to entice the intended audience. The web application developers fabricate technically complex yet professional and innovative web solutions and services that infuse utmost prospective of the latest web technologies to generate an interactive and appealing plea for your patrons. We are most mindful of your business objectives, goals and expectations during development. Accunity's support team is always ready to handle issues, and keep our customers up and running 24×7 and meet all kind of challenges. Our meaning of support is to go hand in hand with our customers and help them gain maximum value from our services.

ABSTRACT

During my internship period at Accunity Software I got a training of one month on various tools and technologies that are used in the project handling content management system for the client. After the successful training of one month, I was put to a live project where I was given real time problems to work upon. I was assigned various tasks under the project where I refined my technical skills of Java coding, HTML and CSS basic introduction, learning of GIT and Maven, JavaScript and other such technologies. Since, the live project follows Agile Scrum methodology, the entire project is divided into different sprints and my task was to complete some of the stories in each sprint. Meanwhile, I was learning these technologies to be applied in the real world scenario. I learnt making of components that render on the web pages and present the final website. I learnt to use front end logic with the back end code and integrate them together to display the final output to the users.

LIST OF ACRONYMS / ABBREVIATIONS

AEM	Adobe Experience Manager
API	Application Program Interface
CMS	Content Management System
CSS	Cascading Style Sheet
CTA	Call To Action
DAM	Digital Asset Management
DOM	Document Object Model
ECM	Enterprise Content Management
ECMA	European Computer Manufacturers Association
HTL	HTML Templating Language
HTTP	HyperText Transfer Protocol
JCR	Java Content Repository
JSON	JavaScript Object Notation
MIME	Multi-purpose Internet Mail extension
MVCC	Multi Version Concurrency Control
OOB	Out Of the Box
OSGI	Open Services Gateway Initiative
POC	Proof Of Concept
POJO	Plain Old Java Object
POM	Project Object Model
REST	Representational State Transfer

SEO	Search Engine Optimization
UI	User Interface
URL	Uniform resource Locator
VCS	Version Control System
WCM	Web Content Management
XSS	Cross Site Scripting

LIST OF FIGURES

CHAPTER 1
INTRODUCTION AND SCOPE

1.1. INTRODUCTION

A content management system (CMS) is defined to be a software application or a suite or a set of related logics and programs that are used for the creation and management of digital content. The CMSes are used for mainly two categories namely enterprise content management (ECM) and web content management (WCM) [18]. The features of content management system includes providing of a simple, transparent, reliable and accessible website interface for the end users that can be used for the final addition of content to a page to be published finally but in a highly-structured manner with wise use of components available. Hence, the overall crux of a CMS is to allow standard content to be published in an efficient manner.

AEM, Adobe Experience Manager is a type of CMS that is mainly a web-based client-server architectural system made and managed for building, creating, storing, managing and deploying different commercial websites. It is a bundle and an integrated package of several infrastructure and application level functionalities and features.

There are a few building blocks that comprise the overall structure of AEM namely:

- Web Application Server: According to this AEM can either be deployed as a standalone mode (using an integrated Jetty web server) or in the form of a web application which resides within a third-party application server (WebSphere, WebLogic, etc).
- Web Application Framework: AEM resides upon the Sling Web Application Framework playing as a role of communication framework that refines the writing of RESTful. Hence, the communication of AEM is dependent upon the Sling architecture.

- Content Repository: Java Content Repository (JCR) is used in AEM which is a type of hierarchical database especially designed for handling both unstructured and semi-structured data. JCR repository stores the user-facing content that is final for the website, and all code, pages, templates, attachment data and internal data used by the application and the client.

AEM also offers many application and web level features for the management of variety of interfaces used in present times such as facility of AEM Websites, AEM Mobile Applications, AEM Communities, Digital Publications, AEM Forms, Digital Asset Management, Online eCommerce [11]. The server of AEM is Java-based and hence it can run on most of the available operating systems that support that platform.

Websites that are developed using AEM as a Content Management System have a highly modular, bundled, structured and decoupled (loosely coupled) architecture where every module coded and designed is an OSGI bundle (OSGI bundle is a JAR file with metadata information along with it)- whether it is by default provided AEM modules or custom made modules for tailored functionalities. In this way, the entire structure of the website can be developed and designed using available and existing components or custom made components. AEM gives a very nice authoring as well as publishing experience to the clients which ensures their satisfaction and adds to the business value of the project. AEM installations on any machine usually involve installing of at least two instances with different port numbers, typically running on same or separate machines for different environment testing and experience. Author Run Mode: An AEM instance which is used to create, manage, upload and edit the final content of the website and which is used to to administer the website. Once content is finalized, approved and ready to go live, it is finally replicated (with due permissions) to the other publish instance. Publish Run Mode: An AEM instance that is responsible for serving the published and finalized content to the public or the end users of the website. It also works in the form of modular OSGI bundles where any changes made to any of the module are reflected “on-the-fly” [2] rather than creating even a minor delay in the deployment of the webpages and therefore, all of these functionalities clubbed and integrated together give AEM an excellent edge over other Content Management System packages available in markets like Drupal, Joomla, WordPress etc.

1.2. PROBLEM STATEMENT

The requirement of the client is to make an efficient, reliable and dynamic website using Adobe Experience Manager that will give the rights to the authors to create the final content and preview them before publishing it to the production server. The task of the development team is to make reusable and generic components that are well tested before presenting it to the clients. The clients hence can independently use these components, author them with the content and finally publish their website as per their requirements. The development of these components will include knowledge of Java, HTML, CSS, Sightly, JavaScript etc.

1.3. EXISTING SYSTEMS AND IT'S LIMITATIONS

The websites can be developed using plethora of technologies that provide multitude of features for website development. But, there can be many drawbacks that might be faced using these technologies. Though AEM is quite expensive tool but the final quality of the website rendered in excellent in incomparable. Users can see the design of the website and it is can be smoothly integrated with many of other Adobe tools for extension of the website.

Some of the limitations of the present systems are:

- Static content

Usually websites that are hard coded using simple HTML, CSS or JS displays the static content which is authored one time. No data can be processed or displayed in real time and therefore, this is one of the major limitation that can be overcome by back end integration of logic in AEM using models, sling models and servlets.

- Efficiency

Many of the other content management systems like Drupal fail to provide efficient and scalable solutions for the heavy and big websites as the scripts that are used take a lot of time of load hampering the real performance of the website. This is because of wide range of features that are provided by Drupal that may compromise the speed and the efficiency. Server load may be decreased by using caching but then the problem of real time updates get affected.

- Flooding customization options

Many of the CMS provide a way too many options to the users for use that they are not able to fully exploit the features of the tool ending up in learning fail to customize.

- High Learning Curve

Many of the available CMS have a very high learning curve that make the process of development quite slow for the beginners. Since, AEM is based on HTML and Java platform, it is quite easy to grasp the essentials and start building the components and products right away.

- Need of plugins

In the existing systems, there is a requirement of installation of many plugins to make a task work efficiently. These plugins are not provided as in build features and hence integration and management becomes cumbersome at sometimes.

- Insufficient documentation

Many of CMS doesn't have proper documentation that can help a beginner to start and learn. there is extra time spent to examine the codes and understand the functionalities. Also, there is no technical support in case there is any tool specific problem or bug.

- Hacking Risks

Since, many simple and cheap CMS are widely used, hackers target them to get into the system many a times. eg. Wordpress uses MySQL as a backend database, so hackers can easily get into the site and play with the content. Hence, security is quite vulnerable in many CMS suites.

1.4. SCOPE OF THE PROJECT

The ongoing project aims at development of a robust, efficient, scalable and effective solution for the present client using Adobe Experience Manager by complying with their entire requirements and developing the website in the form of independent modules which are reusable and highly useful in generic terms. Adobe Experience Manager provides varied features which will help in making the website look good as per the designs and function great as per the features. The project has delivered many features in the previous releases and will keep on adding many more until the further releases.

CHAPTER 2

TOOLS AND TECHNOLOGIES

2.1. GIT

Git is a distributed form of a version control system (VCS) which is used for the purpose of tracking any changes such as creation, modification, deletion [16] etc. in computer files pertaining to a project linked with git repository and coordinating all the work done on those files which can be distributed among multiple people in different geographical locations. It is mainly used for software development and tracking, but it can be used to keep track of changes in any files that needs to be tracked. Also, used as a distributed revision control system, git aims at speed, efficiency, data integrity, and support for all the distributed, modular, structural, non-linear workflows. Every Git directory that is initiated on every computer is in the form of a full-fledged repository (storehouse) with a complete log and history and full version tracking capabilities which will be independent of any network access or connection with a central server.

Git defines main states for the files: committed (ready to be sent), modified (untracked changes), and staged (tracked changes) [9]. Committed means that the data is final and is safely stored in your local database and is ready to be sent to the final repository. Modified means that user has made some changes in the file but have not made them tracked or committed it to local database yet. Staged means that the user has marked or flagged a modified file (created, modified or deleted file) in its current version in local repository to go into the next commit snapshot to the central or remote repository.

There are variety of git commands used for working on large projects that use git and managing them well across the project dimensions.

- git-init - creates an empty Git repository on the local machine or reinitialize an existing one. We can see a hidden .git folder made with this command in the present directory.
- git add - adds the modified files to the staging area or the queue to be committed later. Files are not committed yet or finalized but just added in the staging area to be tracked by the git system.
- git commit - commits the files that have been added previously and creates a new revision which is accompanied with a log. Commit command commits the current changes on the local repository and keeps them ready for the remote repository.
- git checkout – to switch or move between the working branches present in git repository.
- git checkout -b branch_name – to create a new branch with name branch_name on the local repository.
- git status – to get the current status of the branch in terms of files that are tracked and changed or modified files that are still left untracked.
- git log – gives a detailed log of all the commits made previously with commit id, name, time, commit message and other details for a full history record.
- git remote add origin – to set the path for github or bitbucket repository in which the codes needs to be committed. origin acts as a variable holding the value of the url of the remote git repository on which the code needs to be sent and merged into the project.
- git pull origin branch_name – to pull the latest code from the branch specified with branch_name. It fetches the latest merged codes from the remote repository.
- git push origin branch_name – to push the committed changes in the branch on the remote repository. If the remote repository doesn't contain this branch then one is created with this commit otherwise the same branch gets a new commit.
- git stash – to save the work done in the current branch before switching the branch or taking a pull from the remote repository.

```
sukri@LAPTOP-HDSK5GUP MINGW64 /d/aem jar/Bhf-AEM/bhf-aem (develop)
$ git checkout -b my_branch
Switched to a new branch 'my_branch'

sukri@LAPTOP-HDSK5GUP MINGW64 /d/aem jar/Bhf-AEM/bhf-aem (my_branch)
$ git status
On branch my_branch
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        core/src/main/java/com/bhf/aem/workflow/process/MyWorkflow.java

nothing added to commit but untracked files present (use "git add" to track)

sukri@LAPTOP-HDSK5GUP MINGW64 /d/aem jar/Bhf-AEM/bhf-aem (my_branch)
$ git add core/src/main/java/com/bhf/aem/workflow/process/MyWorkflow.java

sukri@LAPTOP-HDSK5GUP MINGW64 /d/aem jar/Bhf-AEM/bhf-aem (my_branch)
$ git commit -m "Workflow class added"
[my_branch 6877bd3] Workflow class added
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 core/src/main/java/com/bhf/aem/workflow/process/MyWorkflow.java

sukri@LAPTOP-HDSK5GUP MINGW64 /d/aem jar/Bhf-AEM/bhf-aem (my_branch)
$ git status
On branch my_branch
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        deleted:    core/src/main/java/com/bhf/aem/workflow/process/ImageRenditionProcess.java

no changes added to commit (use "git add" and/or "git commit -a")

sukri@LAPTOP-HDSK5GUP MINGW64 /d/aem jar/Bhf-AEM/bhf-aem (my_branch)
$ git add .

sukri@LAPTOP-HDSK5GUP MINGW64 /d/aem jar/Bhf-AEM/bhf-aem (my_branch)
$ git commit -m "rendition workflow deleted"
[my_branch 6072d60] rendition workflow deleted
 1 file changed, 180 deletions(-)
 delete mode 100644 core/src/main/java/com/bhf/aem/workflow/process/ImageRenditionProcess.java

sukri@LAPTOP-HDSK5GUP MINGW64 /d/aem jar/Bhf-AEM/bhf-aem (my_branch)
$ git log
commit 6072d60550e2f297e7649735ba015ce6377aea00
Author: citizen-sukriti <sukriti@accunitysoft.com>
Date:   Sat May 13 17:20:49 2017 +0530

    rendition workflow deleted

commit 6877bd346da9cd01962398161fc9eada693729ba
Author: citizen-sukriti <sukriti@accunitysoft.com>
Date:   Sat May 13 17:20:00 2017 +0530

    Workflow class added
```

Fig. 2.1 Snapshot of Git commands

There are many other useful commands in git which can be used as per the use case such as rebase, reflog, reset etc.

2.2. MAVEN

Maven is a project management tool that is based on POM (project object model) file for it's entire execution. It is used for the tasks of project's build, dependency resolutions and documentation [14]. A build tool takes care of everything that is required for building a process and executing the code. It generates the source code in case some automatic feature is required, compiles source code, builds the source code, packages the compiled code into JAR of ZIP file for the process of installation and finally installs the packaged code in the specified local repository. Hence, maven is a comprehensive tool providing ultimate features that narrow the tasks of developers into running of just one command that can handle all the build tasks together.

Build lifecycle is termed as a list of phases and included goals that orders the system for execution of the defined tasks. Maven lifecycle includes three main phases: Clean (cleans up all the artifacts or content that is created by any prior builds occurred in the same project), default (it comprises of many goals in a line such as validate-validation of code, compile-compilation of source code, test-testing using suites, package-packaging into JAR, verify-verify testing, install-install on the repository, deploy-on the final server) [15] and site (for site documentation). Maven archetypes are the basic building blocks which are blueprints of the projects that defines the skeleton and sets up the structure of the project. it creates the required modules of the project as required by the type of the project using the archetype.

A central feature present in Maven is of dependency management. it includes various dependencies that are resolved according to the scope provided. Maven's dependency-handling process is centred around a very coordinated system that identifies the individual artifacts or modules and resolves any dependencies they have on any other package. POM has some major and important configurations that are widely used for the management of the project such as project dependencies, repositories, plugins, goals, build profiles, developers, project version, id, groups etc. This is a sample code snippet of a POM.xml file.

```
<groupId>com.internship</groupId>
<artifactId>internship-reactor</artifactId>
<version>1.1.0-SNAPSHOT</version>
<packaging>pom</packaging>
<name>2017 Internship AEM - Reactor Project</name>
<description>Maven Multimodule project for Internship AEM.</description>
<dependencyManagement>
  <dependencies>
      <dependency>
          <groupId>org.osgi</groupId>
          <artifactId>org.osgi.core</artifactId>
          <version>4.2.0</version>
          <scope>provided</scope>
      </dependency>
<dependency>
  <groupId>org.apache.felix</groupId>
  <artifactId>org.apache.felix.scr.annotations</artifactId>
  <version>1.9.8</version>
  <scope>provided</scope>
</dependency>
<dependency>
```

```
  <groupId>javax.jcr</groupId>
  <artifactId>jcr</artifactId>
  <version>2.0</version>
  <scope>provided</scope>
</dependency>
</dependencies>

<repositories>
  <repository>
      <id>adobe</id>
      <name>Adobe Public Repository</name>
      <url>https://repo.adobe.com/nexus/content/groups/public/</url>
      <layout>default</layout>
  </repository>
</repositories>
<pluginManagement>
  <plugins>
      <plugin>
          <groupId>org.apache.felix</groupId>
          <artifactId>maven-scr-plugin</artifactId>
          <version>1.20.0</version>
      </plugin>
</plugins>
</pluginManagement>
<profiles>
  <profile>
      <id>autoInstallBundle</id>
      <build>
          <plugins>
```

```
                <plugin>
                    <groupId>org.apache.sling</groupId>
                    <artifactId>maven-sling-plugin</artifactId>
                    <executions>
                        <execution>
                            <id>install-bundle</id>
                            <goals>
                                <goal>install</goal>
                            </goals>
                        </execution>
                    </executions>
                </plugin>
            </plugins>
        </build>
    </profile>
</profiles>
<modules>
    <module>core</module>
    <module>ui.apps</module>
</modules>
```

The above code snippet is an example of parent POM.xml having major properties such as artifact id, group id, version, dependencies, plugins, repositories, modules etc. This is the main file that is required for a Maven project to run and fulfill the build functionalities required. The child POM of core and apps have a similar structure but they do not specify the versions or the scope.

All of these properties of Maven POM.xml helps in building and deploying of the source code and fulfills the task of this project management tool.

2.3. AEM (ADOBE EXPERIENCE MANAGER)

AEM is a Web Content Management System which provides real time experience management for the websites that are developed using this tool [1]. AEM Stack consists of Apache Sling, JCR, OSGI as the main ingredients which forms a part of this entire structure however, they act as independent project entities that can be used as stand-alone to build applications.

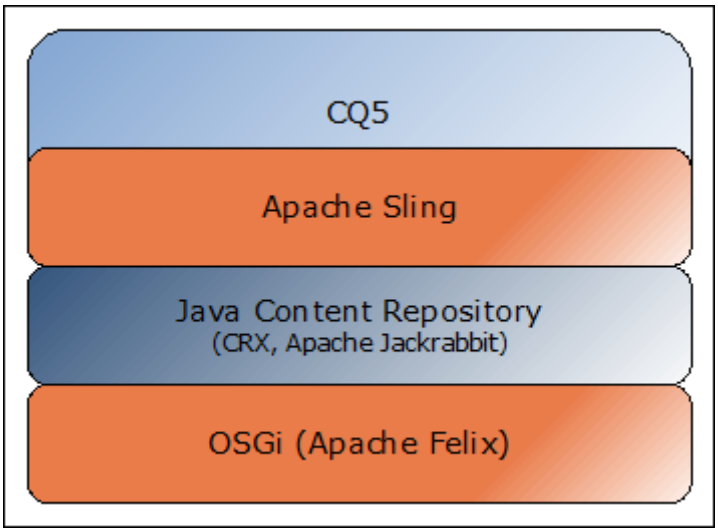


Fig 2.2: AEM Stack

CQ5 is the older version of AEM 6.2 (6.3 is in beta version presently) and is the top layer of user interface. AEM includes an OSGi framework which is implemented based on Apache Felix (Apache Felix Console is provided to look into bundles and their states), which implements OSGi Service Platform used for AEM. OSGi is termed as a dynamic software component system which is used for Java. In an OSGi-based system like AEM Stack, an application is build up of an assemblage and smooth integration of components, called bundles in OSGi, which are loosely coupled and can be dynamically installed, started paused, stopped, resumed and uninstalled at runtime [8], without shutting down and restarting the entire application because of it's feature list. A bundle is a jar file holding Java classes and a special metadata file (in the jar file's META-INF) [7]. All the elements of AEM are written on top of the AEM platform backed by OSGi and are implemented as OSGi bundles which can be seen on the felix console. In a running AEM instance at specified port number, bundle management can be seen at http://<host>:<port>/system/console/bundles.

All data within AEM (user created data, website data, custom data) is stored in the CRX content repository**,** which is Java Content Repository Specification (JCR) implemented in the form of Oak. JCR is a repository implemented by Jackrabbit. JCR is unstructured but hierarchical data and hence is arranged in the form of directory like structure which is not like other traditional database structures. Oak is implementation of a scalable and hierarchical content used for websites. Oak has the concept of MVCC (multi version concurrency control) [10] in which many versions of a single file can be concurrently used, modified, worked upon and added in the project.

AEM is built using Sling architecture which is a Web application framework having it's roots in RESTful principles [5]. Sling uses a JCR repository provided by AEM, such as Apache Jackrabbit as its data store. The main consideration in using Sling is to worry about whether the URL resolves to a stored content object or not where a script can be found for rendering. This hit of sling request is of prime importance for the correct content to be picked up and rendered on screen. Sling is content-centric framework that means that everything is taken as a part of content stored in the repository. This also means that processing task has it's main focus on the content based on REST as each (HTTP) request is mapped to content in the form of a JCR resource (because everything in JCR is considered as a resource) [19]. Sling script resolution is one of the main concepts on which the entire website rendering process is based upon. It resolves which resource to call and which script to choose from that resource when a request is sent. It has selectors, extensions, suffix into it which have their own priorities while execution. There is a priority or a sequence in which the resources are accessed according to the script. Sling Script Resolution: It includes sending of a http request with a method, path, selector and suffix [4]. Then, URL resolves to the content object stored at the resource hit for which script is taken up for rendering. Then the resource type is extracted from the resource properties and looked up in the apps folder. Script is located in either apps folder or libs folder. Script names are resolved best suited or closely matched.

The dispatcher in AEM serves the role of caching and load balancing. The dispatcher caches the pages and send them to the user as and when required. It may also cause trouble sometimes when cache flush of the dispatcher is not automated and it is required to test some new functionality in the project.

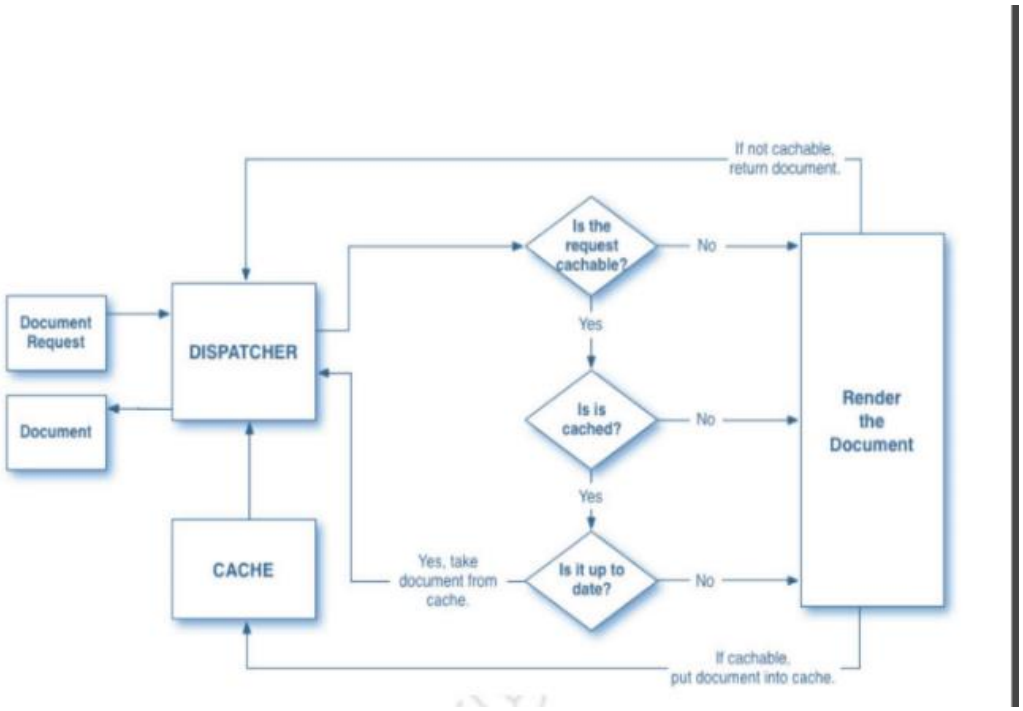


Fig 2.3 Dispatcher in AEM

2.4. HTL (HTML TEMPLATING LANGUAGE)- SIGHTLY

The purpose of HTML Template Language (HTL), which is supported by Adobe Experience Manager (AEM) as a templating language, is to offer a productive enterprise-level framework [6] for the web that increases security as compared to the use of jsp files earlier, and allows HTML developers to participate in AEM projects and integrate with Java without having much knowledge of the language.

The HTML Template Language was introduced with AEM 6.0 version only, and takes the place of JSP (JavaServer Pages) as being the recommended server-side language templating system for HTML and front end rendering. For web developers, to build robust enterprise or corporate level websites, this Sightly i.e. HTML Template Language helps to achieve security and efficiency. Advantages of slightly include being lightweight – having no dependencies, fast to render, fast learning curve and lean management; Secure – Automatic XSS protection as compared to previous server side languages used and URL externalization which enhances security of the website [6]; code-less feature – This enforces separation of

logic (handled by back end sling models) and markup (handled by HTML and CSS); powerful – Straight-forward APIs used for building logic; Intuitive – Clear, lightweight, simple & small feature set.

HTL consists of various elements that should be known to the developers for integrating front end with the back end such as block statements (use, test, resource, include) and expression language (conditional expressions, conditional operators etc.), global objects (properties, pageProperties, currentPage, currentStyle, currentSession, request, resource, sling, resourcePage, wcmmode etc.)

The following code snippet uses global objects given by Sightly to access some basic properties.

```
<data-sly-include="/libs/foundation/global.jsp">
    <h2> Current Page </h2>
      Title: ${currentPage.title} <br/>
      Name: ${currentPage.name} <br/>
      Path : ${currentPage.path} <br/>
      Depth :${currentPage.depth} <br/>
    <h2> Current Node </h2>
      Name: ${currentNode.name} <br/>
      Path : ${currentNode.path} <br/>
      Depth :${currentNode.depth} <br/>
```

Fig. 2.4 Sightly Global Objects

2.5. JAVASCRIPT

JavaScript is a lightweight, interpreted programming language that is used in AEM for handling client-side scripting. JavaScript is very easy to implement because it is integrated with HTML and can be used for plethora of functions. It is open and used as cross-platform. JavaScript can be implemented using JavaScript statements that are placed within the <script>... </script> HTML tags in a web page using HTML language.

In AEM, all of the js code is placed in clientlibs. Client Library (or ClientLib) functionality manages all the JavaScript and CSS resources in the application and the project. It takes care of dependency management, minifying content (removing unnecessary white spaces) before sending it to the network to reduce the load. We have clientlibs folder in etc or apps where every js file is placed and used. Therein, we have a folder for all the js files and all the css files required. Also, there will be js.txt and css.txt file which will have names of all the files to use. The clientlib folder has a property called categories which uniquely identifies the clientlib i.e. it can be a unique identifier for the folder. The clientlib is used in the component with use statement and call statement to call css or js or all files of that particular clientlib. Clientlibs also have embed and dependencies properties which will have list of all the dependencies of that clientlib and all the other clientlibs that needs to be embedded before it's use. Dependencies are transitive but embeds are non transitive in the case of clientlibs i.e. a hierarchy of dependencies will be loaded but not in the case if the property says embed. A client-side library folder is a repository node of type cq:ClientLibraryFolder, placed anywhere within the /apps, /libs and /etc (can be configured in the configuration manager of

AEM); The categories property is a multi-valued property that allows a client library folder to be part of more than one categories at the same time for logical interpretation; dependencies: This is a list of other client library categories on which this library folder depends. For example, given two cq:ClientLibraryFolder nodes A and B, if a file of js in A requires another file in clientlib B in order to function properly or load it's content, then at least one of the categories of B clientlib should be included among the dependencies of A; embed: Used to embed code from other client libraries. If node A embeds nodes B and C, the resulting HTML will be a concentration of content from nodes B and C.

2.6. JAVA (SLING MODELS)

They are Java classes like POJO simple to implement and connected to Sightly which contain multiple implicit objects and features to perform a certain task on the components rendered using HTML and acting as a business layer in the code logic.

There are various annotations provided by the sling model such as inject, sling object, aem object etc. that are used to have an access to objects presented from APIs. These sling models are called in the slightly code by specific use of commands. Some of the APIs used are Page, Resource, PageManager, Node, Navigation etc. We can manipulate the properties of the components using sling models as the entire logic lies in here. We can use the functions of the APIs to fit in the requirement so the logic.

@Model

declares a model class or interface. it is necessary to append this above the class name of the sling model to be used with Sightly.

@Inject

marks a field or method as injectable. This means that we need to get this property from the JCR and inject them into the sling model before the methods are called.

@PostConstruct

This calls the init method which is the start of the sling model to call upon model option creation.

@Named

declare a name for the injection property (This is used when the name of the property might differ otherwise, default values based on field or method name are taken).

@Optional

marks a field or method injection as optional. The sling model can proceed with the function even if this property is not found or injected in the code.

@Filter

an OSGi service filter. it is used to declare a Java class as one

@Default

set default values for a field or method

Java backend logic can also be used in other forms that Sling models. They can be in the form of normal POJO classes (models), OSGi services, servlets, Sling Services, Workflow classes, constant maintaining classes, APIs, Utility classes etc.

CHAPTER 3

UML DIAGRAMS FOR PROJECT

3.1. FLOW DIAGRAM

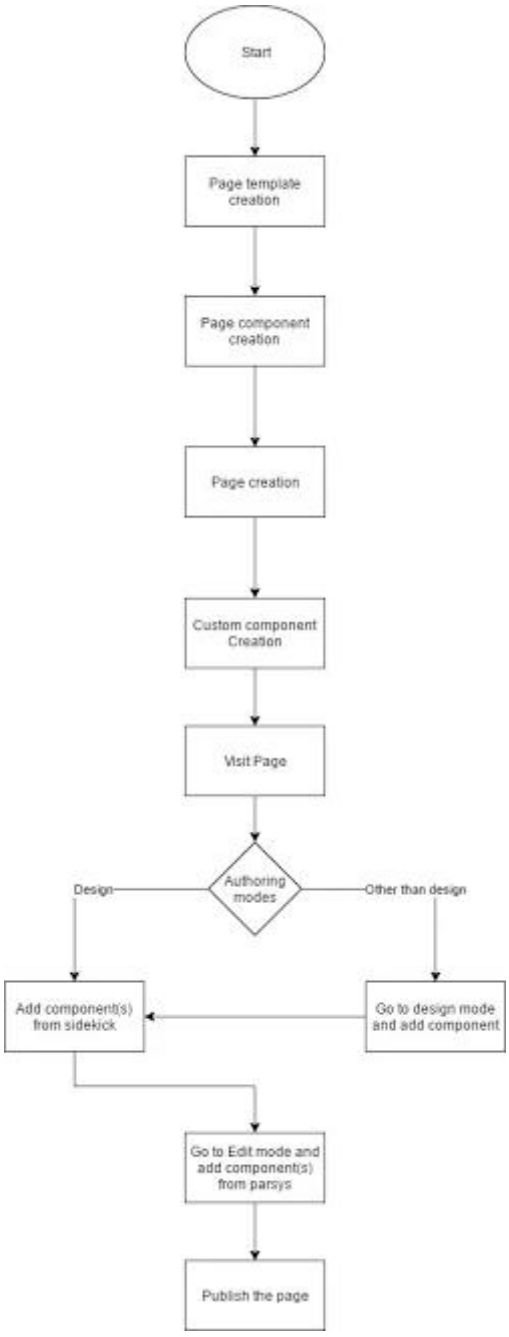


Fig. 3.1. Flow Diagram of project

3.2. DATA FLOW DIAGRAM

3.2.1 DFD 0 LEVEL

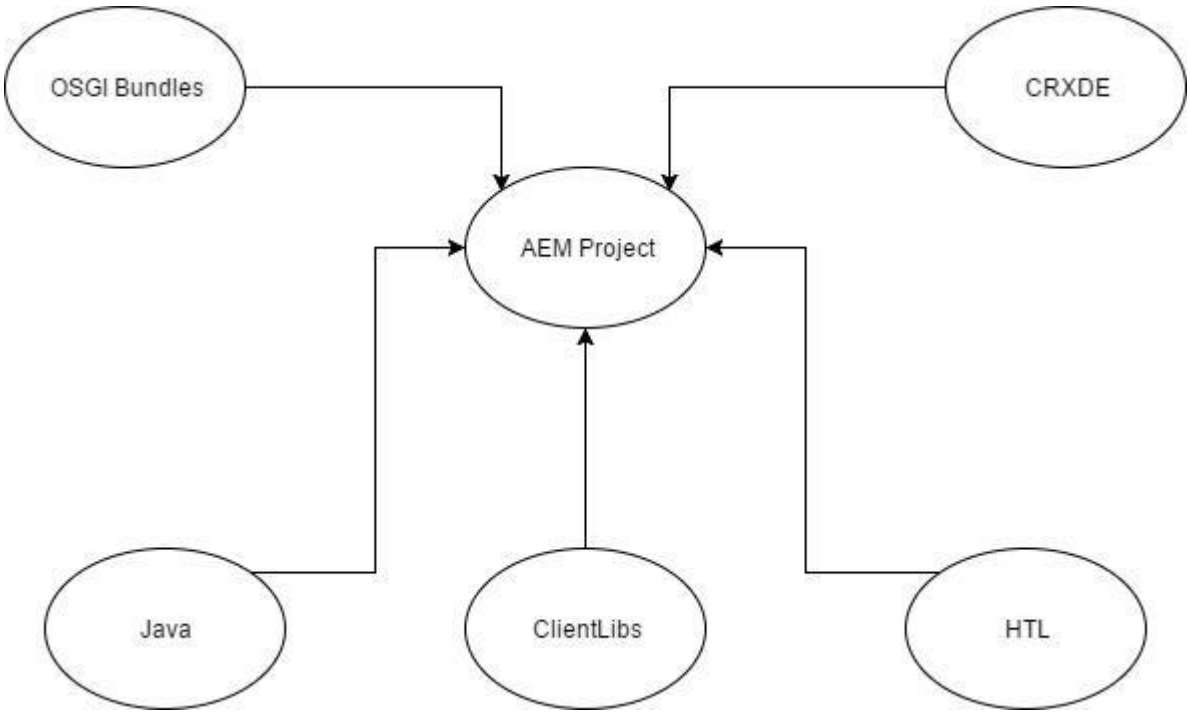


Fig. 3.2 0 LEVEL DFD

3.2.2 DFD LEVEL 1 (a)

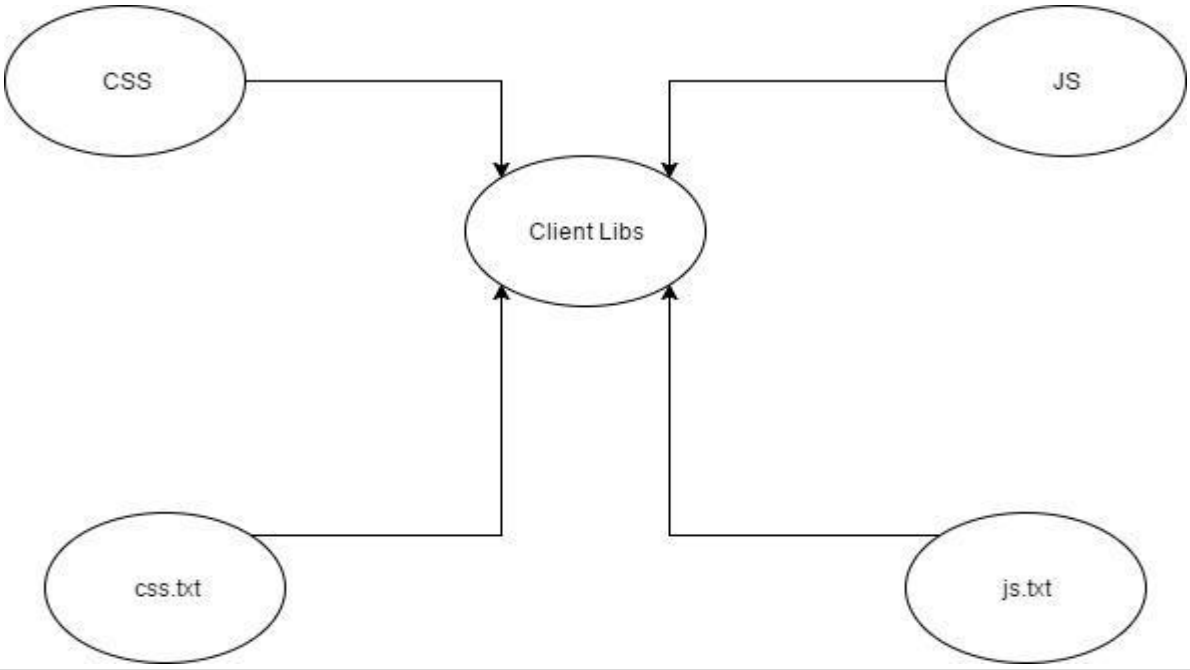


Fig. 3.3 LEVEL 1 DFD

3.2.3 DFD LEVEL 1 (b)

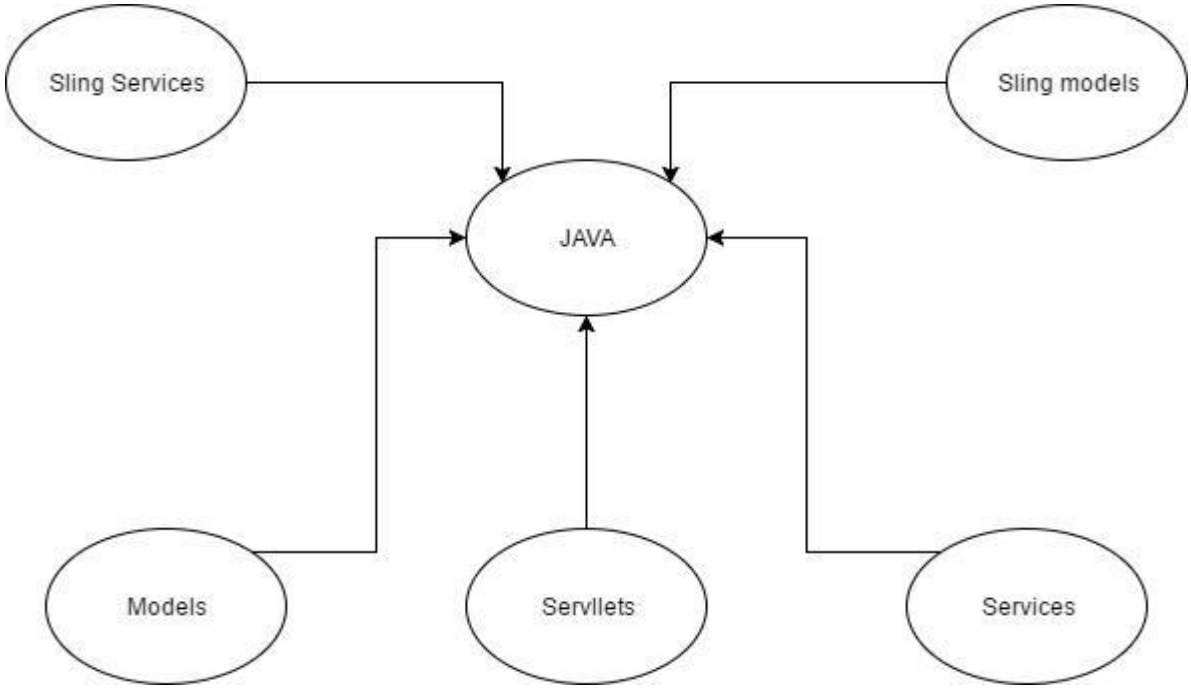


Fig. 3.4 LEVEL 1 DFD

3.2.4 DFD LEVEL 1(c)

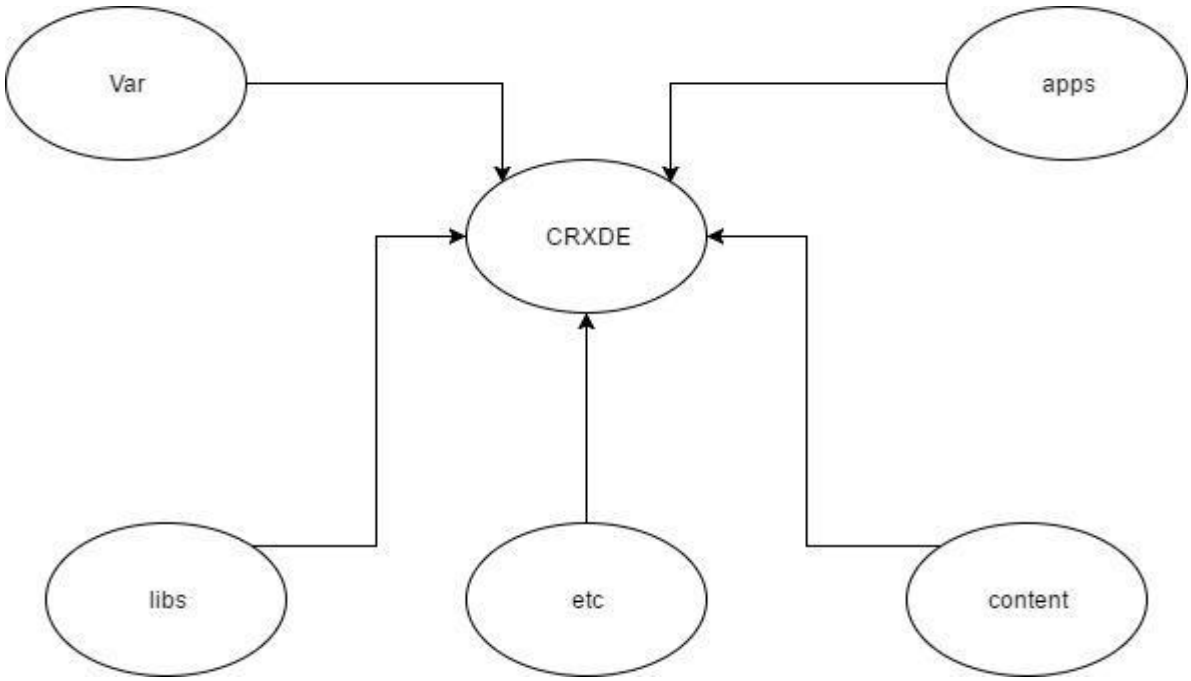


Fig. 3.5 LEVEL 1 DFD

3.2.5 DFD LEVEL 2(a)

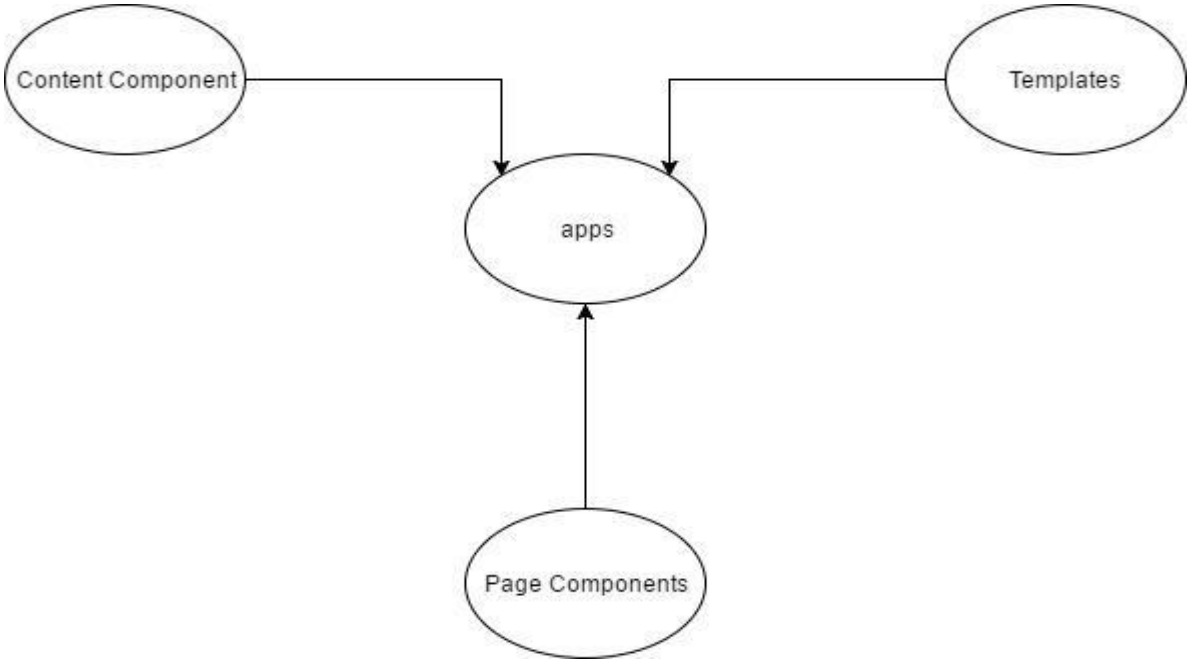


Fig 3.6 LEVEL 2 DFD

3.2.6 DFD LEVEL 2(b)

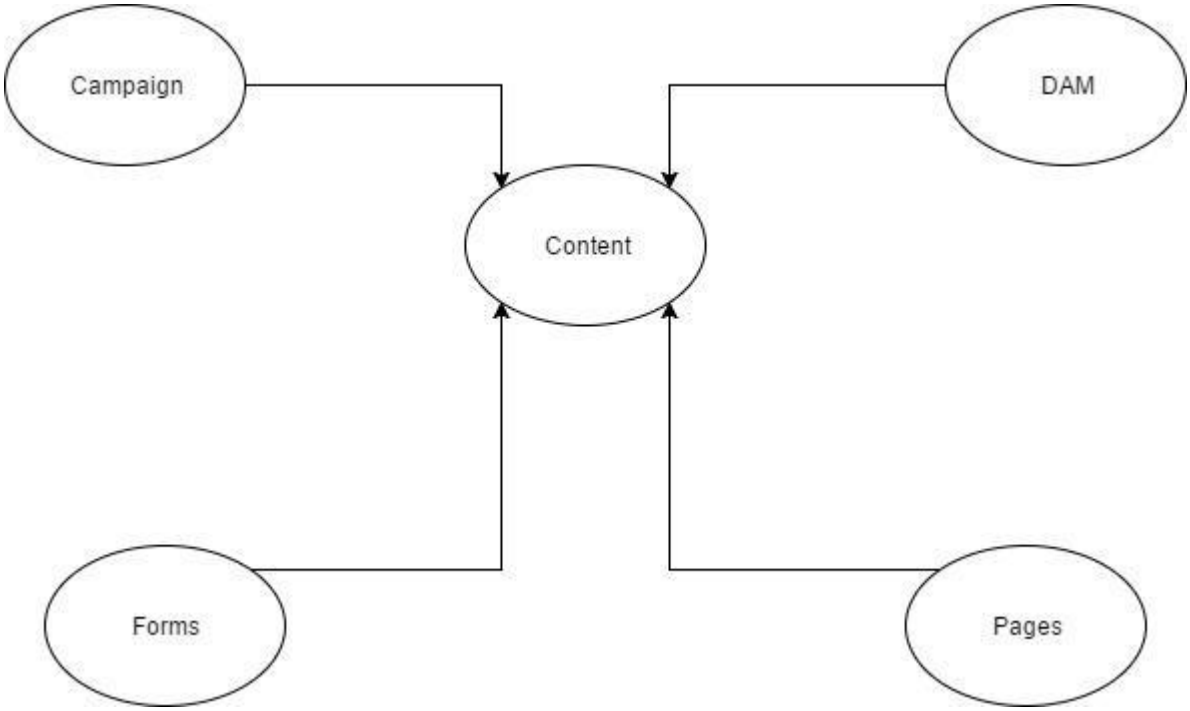


Fig. 3.7 LEVEL 2 DFD

3.2.7 DFD LEVEL 2(c)

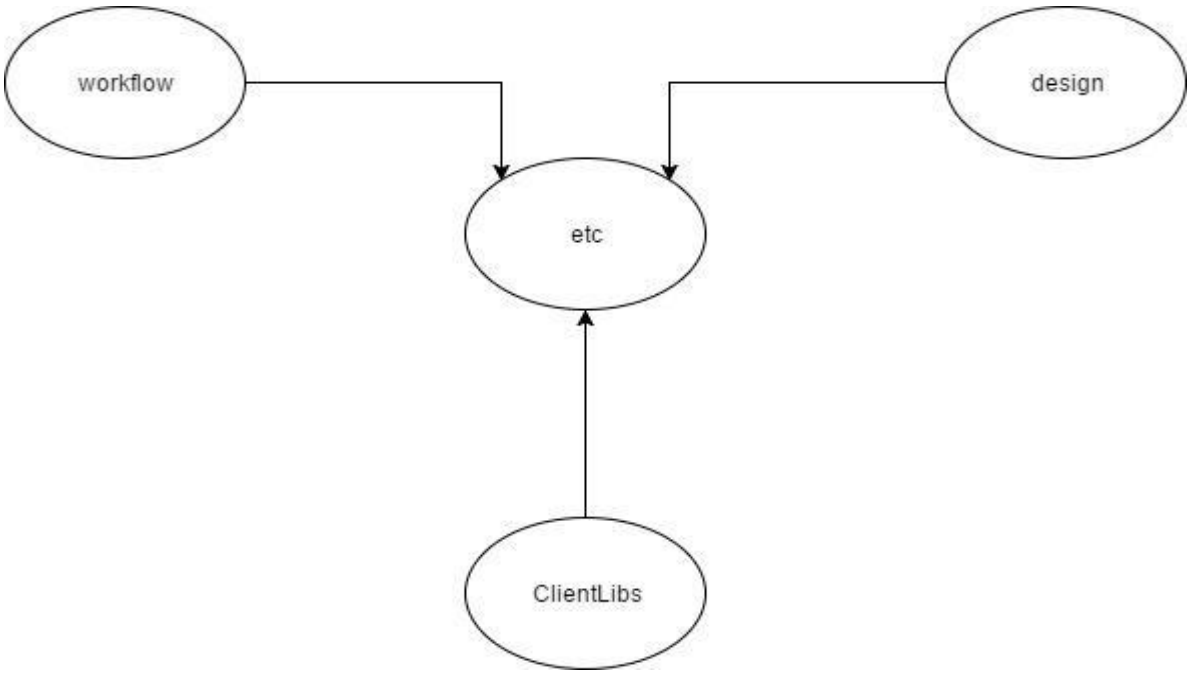


Fig. 3.8 LEVEL 2 DFD

3.3. USE CASE DIAGRAM

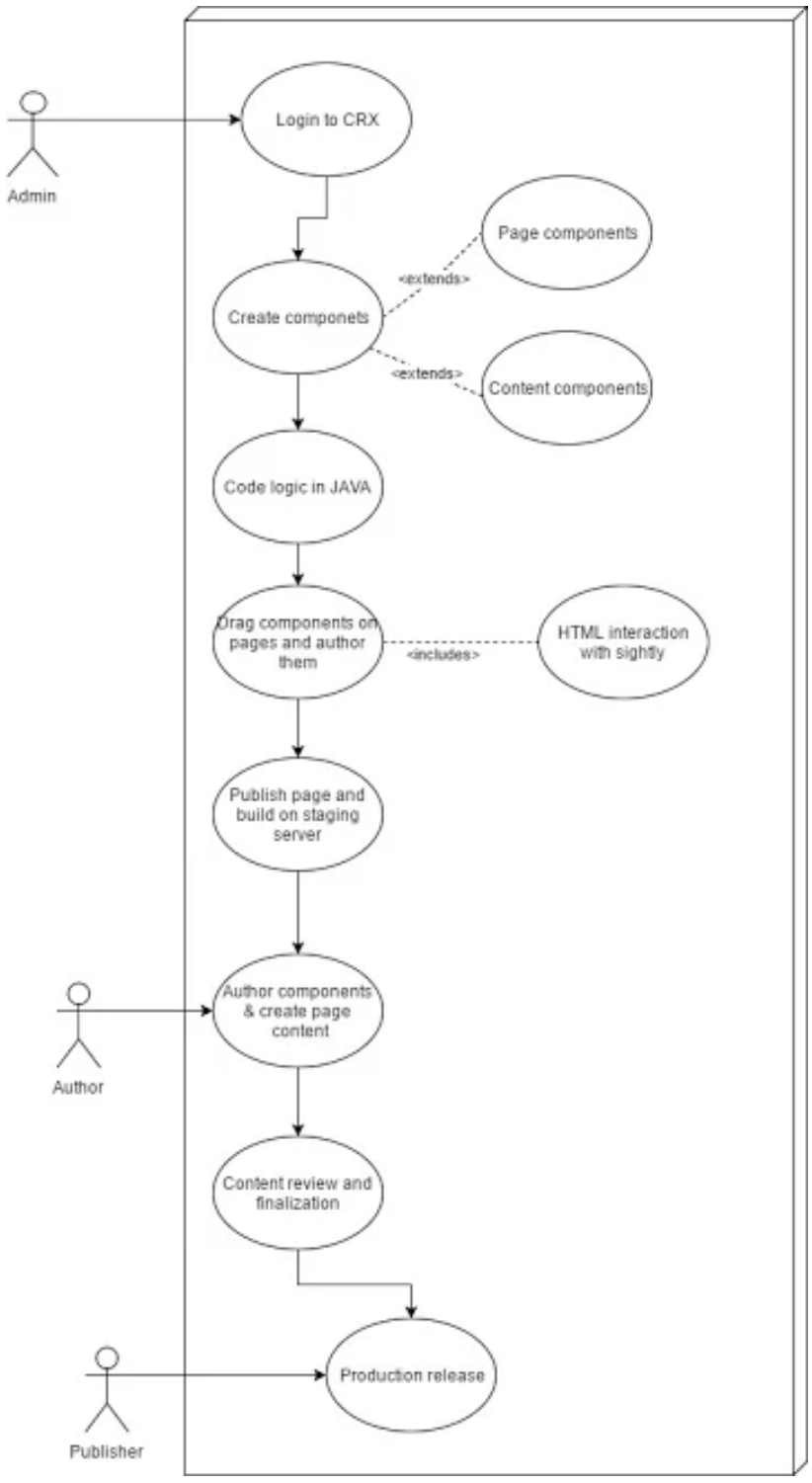


Fig 3.9 Use Case Diagram

3.1. SYSTEM ARCHITECTURE

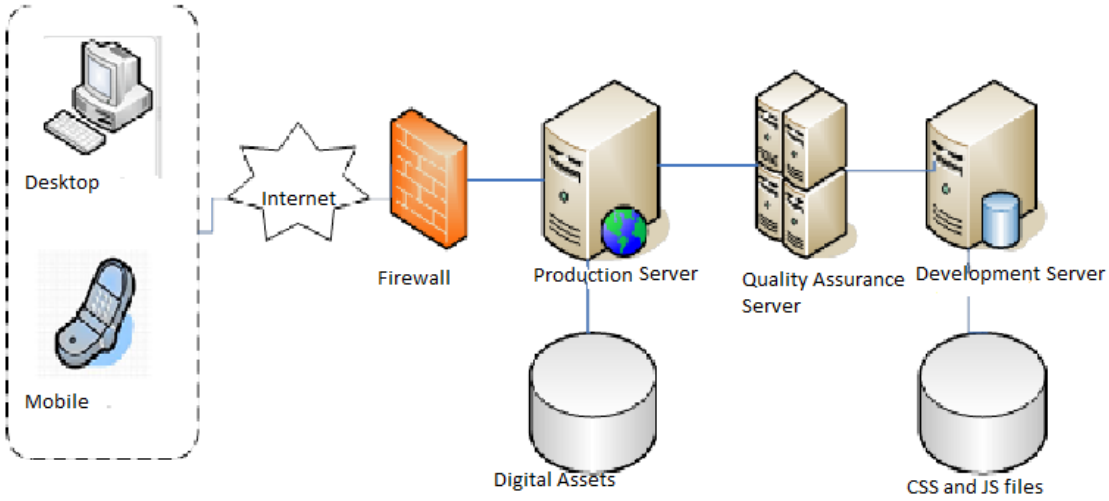


Fig. 3.10 System Architecture

CHAPTER 4

THEORETICAL LEARNINGS AND RESEARCH

4.1. CLIENTLIBS IN AEM

Many modern websites rely on client-side processing handled by JavaScript code. To deal with this issue, AEM provides Client-side Library Folders in order to put js and css code at one place to make it manageable and efficient to load.

Clientlibs or Client libraries in AEM provided by Adobe will manage all JavaScript and CSS resources in the application. We can edit the clientlibs folder in /apps, /libs, /etc but it is recommended to edit only in etc folder. The process to make a clientlib is

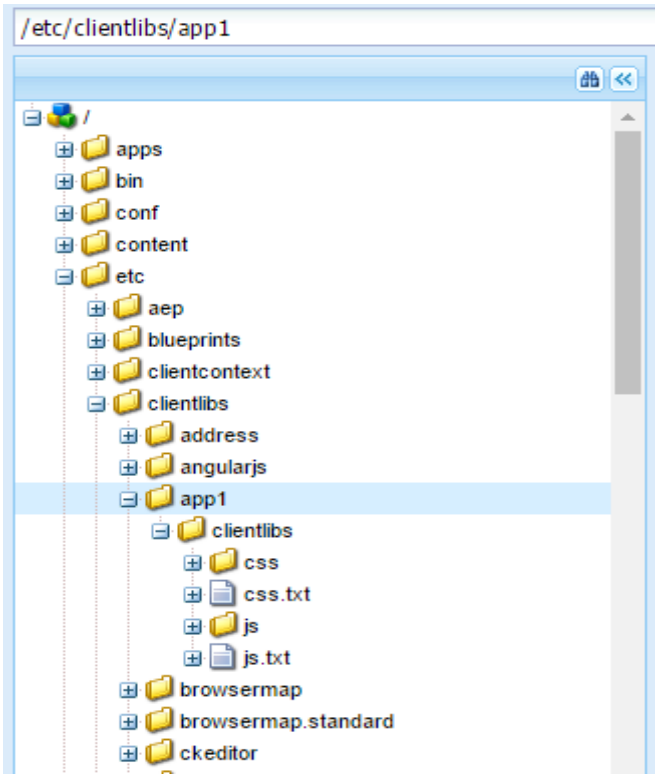


Fig. 4.1 ClientLib File Structure

- Go to Crxde
- Make a new folder in etc/clientLibs folder
- Create new node in that new folder.

- Name: clientlib (Any name)
- Type: cq:ClientLibraryFolder

- Add categories property (required) to clientlib node
  - Name: categories
  - Type: String
  - Value: apps.aem.training (This name will be used to identify respective clientlibs folder)
- Create a css and js folder parallel to clientlib folder, where our css and js files will be placed.
- Create a js.txt and css.txt file parallel to cilentlibs folder.

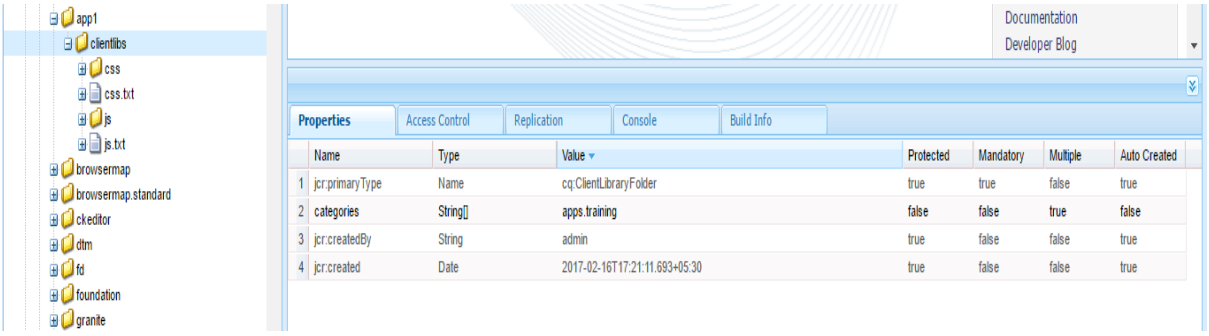


Fig. 4.2 Properties of ClientLibs

Adding JavaScript and CSS resources

Add all the css and js files in css and js folder respectively.

For example

header.css

```
.color-header{
   background-color: blue;
}
```

headerfont.css

```
.header,.p {
   font-weight: bold;
 font-family: verdana;
}
```

example.js

```
alert("a.js ");
```

test.js

```
alert("b.js");
```

In the css.txt file write
#base=css

[Names of css files in css folder (one file name per line)]

Example:

#base=css

header.css

headerfont.css

In the js.txt file write

#base=js

[Names of js files in js folder (one file name per line)]

Example:

#base=js

example.js

test.js

The dependencies must be another cq:ClientLibraryFolder. To identify dependencies, add a property to your cq:ClientLibraryFolder node with the following attributes:

- Name: dependencies
- Type: String[]
- Values: The value of the categories property of the cq:ClientLibraryFolder node that the current library folder depends on.

To embed the library, add a property to the embedding cq:ClientLibraryFolder node, using the following property attributes:

- Name: embed
- Type: String[]
- Value: The value of the categories property of the cq:ClientLibraryFolder node to embed.

<sly data-sly-use.clientlib="/libs/granite/sightly/templates/clientlib.html"

data-sly-call="${clientlib.all @ categories='apps.training'}"/>

This code snippet allows addition of clientlib to the file in Sightly.

4.2. AEM FORMS

Adobe has developed AEM Forms tool that makes the development of forms easy with efficient management. This tools enables creation of multi-step interactive and adaptive forms. The authoring of these forms in similar to authoring done in AEM for rendering the content using components. Loading of AEM forms is faster because of use of "Lazy loading" technique. When files (images, ppt, word etc.) are uploaded in form they are automatically converted into PDF format to retain the formatting and make it compatible for all the browsers and environments.

AEM forms provides a lot of out of the box components like

- E-signature
- File attachment feature
- Form fragment (a generic panel created once and used for multiple forms)

It also provides many inbuilt functionalities and features like

- Pre filling of forms using default values
- Theme development console to give look and feel to the forms
- Rule editor
- Inbuilt submit actions like PDF generation
- Emailing the form data to the initiator on successful submission of the content of the form.

The basic form will look like the following figure providing some fields by default which are configurable,

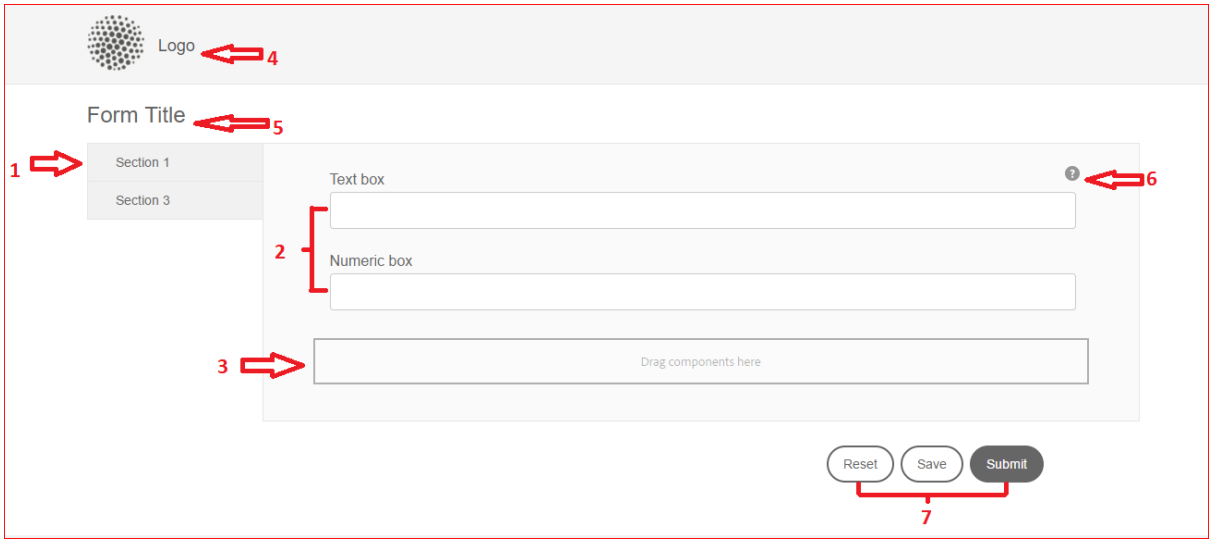


Fig. 4.3 Form Console

1. We can divide the form in sections/tabs to integrate related fields. We can add, remove, rename these sections.
2. These are the default fields that we can change its field label , input type, placeholder text, Regex applied in the property field in the sidekick. They are highly flexible to use.
3. We can add more fields and widgets in the form by dragging those from parsys similar to the components in AEM.
4. We can configure logo of the form by simply clicking on the logo icon and edit it . The image can be uploaded from DAM or file system.
5. Form title can also be configured.
6. This is help content, which we can provide with each field as a description.
7. These are the basic by default buttons which we can modify on both frontend and backend logic.

There are some of the widgets that are provided by default for the form.

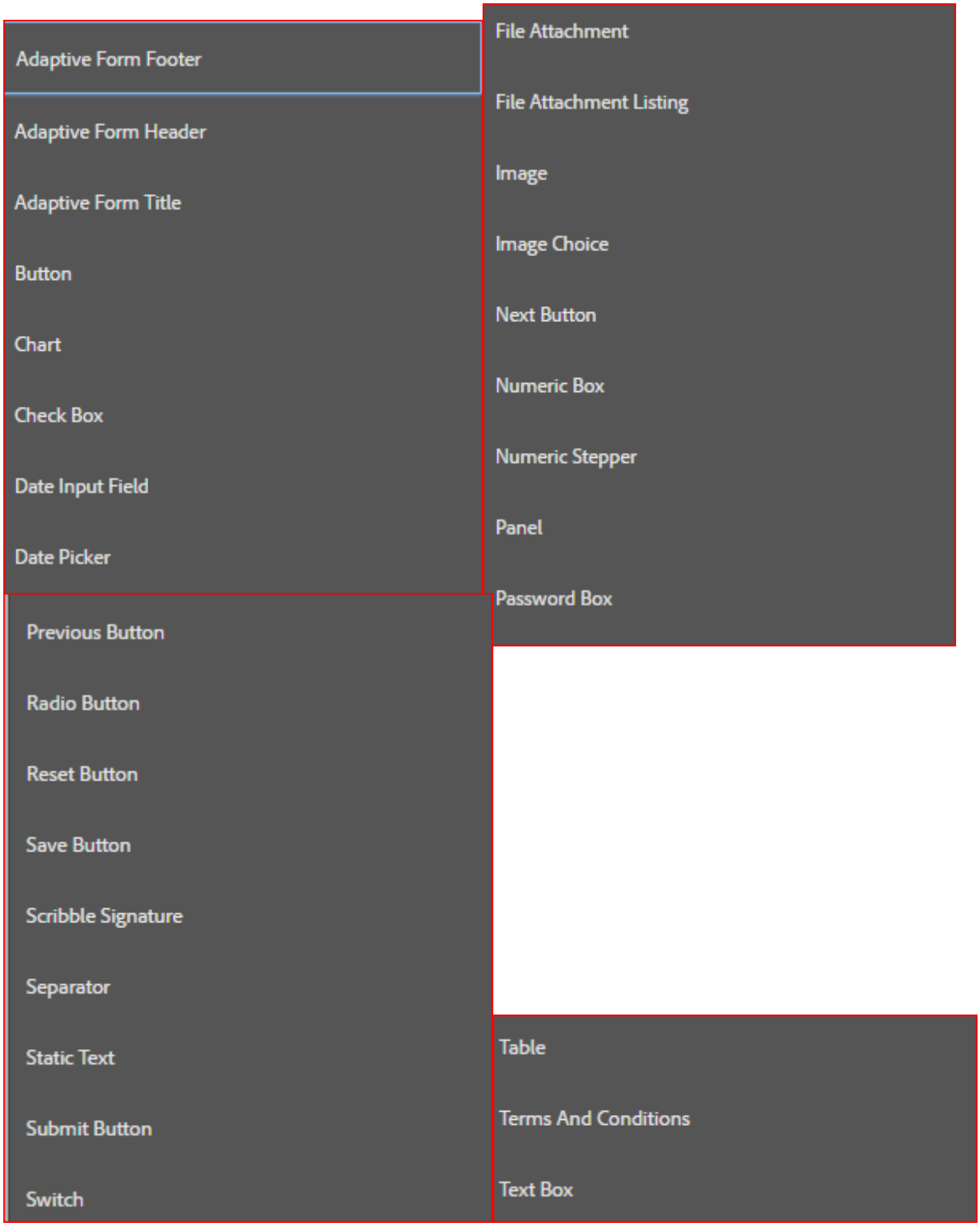


Fig. 4.4 Form Default Component List

Rule Editor for fields: There are many situations when some special rules are applied to forms in the dropdown fields like being applied differently for different countries/states/locations etc. Example, in the case of PIN code there can be different hence requiring different rules to be applied on the field. There is no need to write code for it rather it is a functionality provided by AEM Forms.

- To make a rule select the section or field on which you want to apply the rule and select the rule editor symbol(hammer and plate).

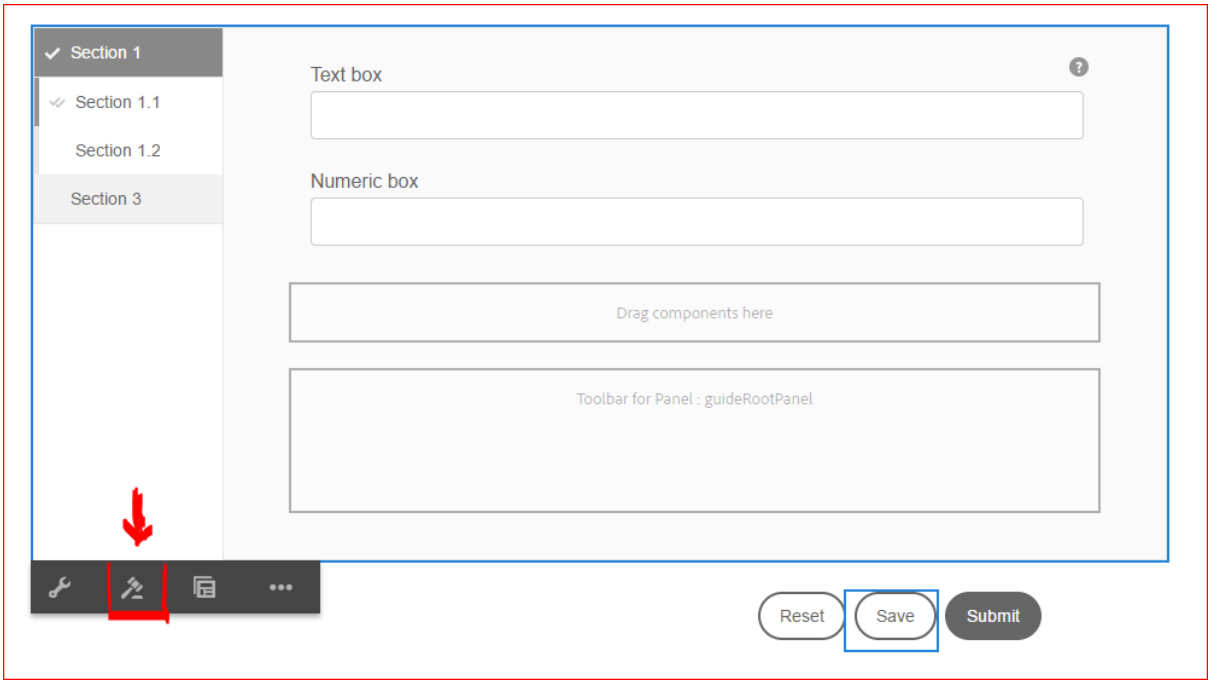


Fig. 4.5 Rule Editor for Form

- Select the section from sidekick in the left

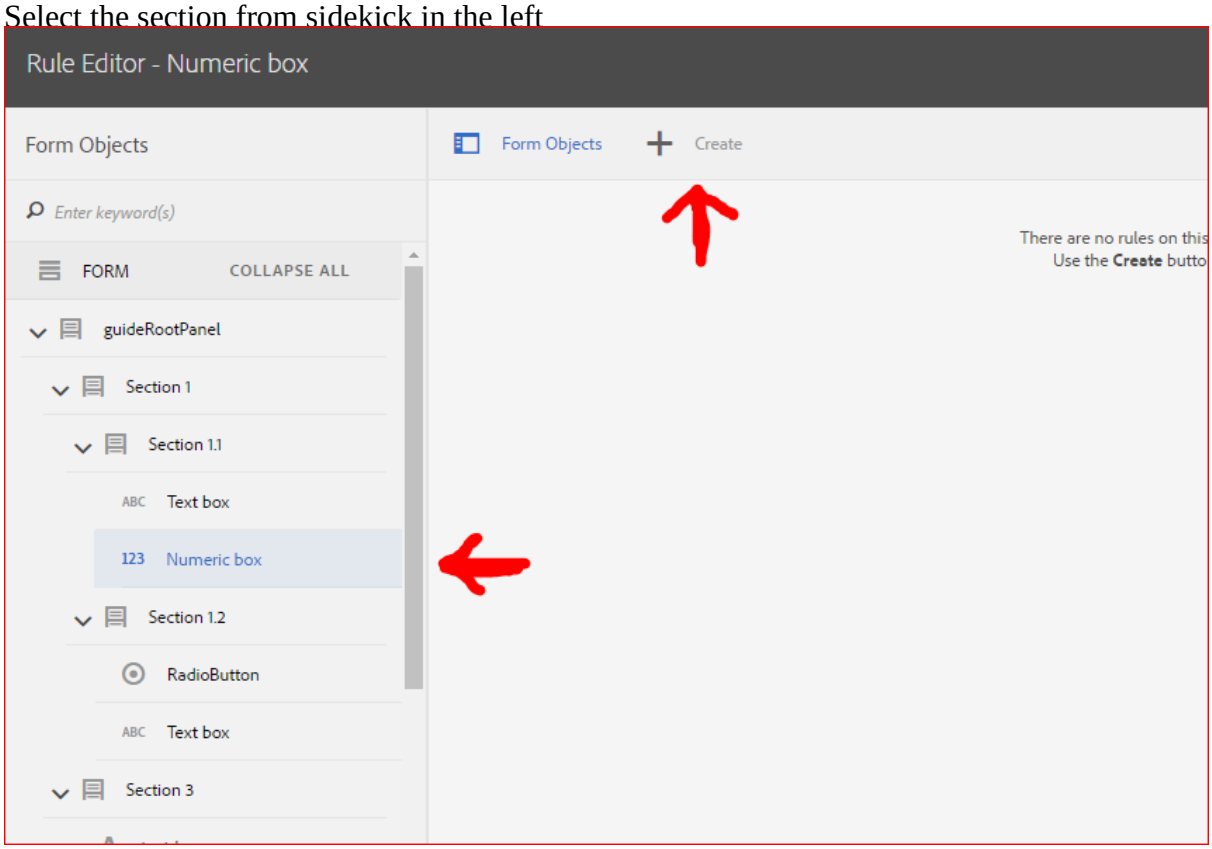


Fig 4.6 Rule Editor for Form (b)

- Select create (plus + icon on top)
- Select the type of operation when is it to be performed.

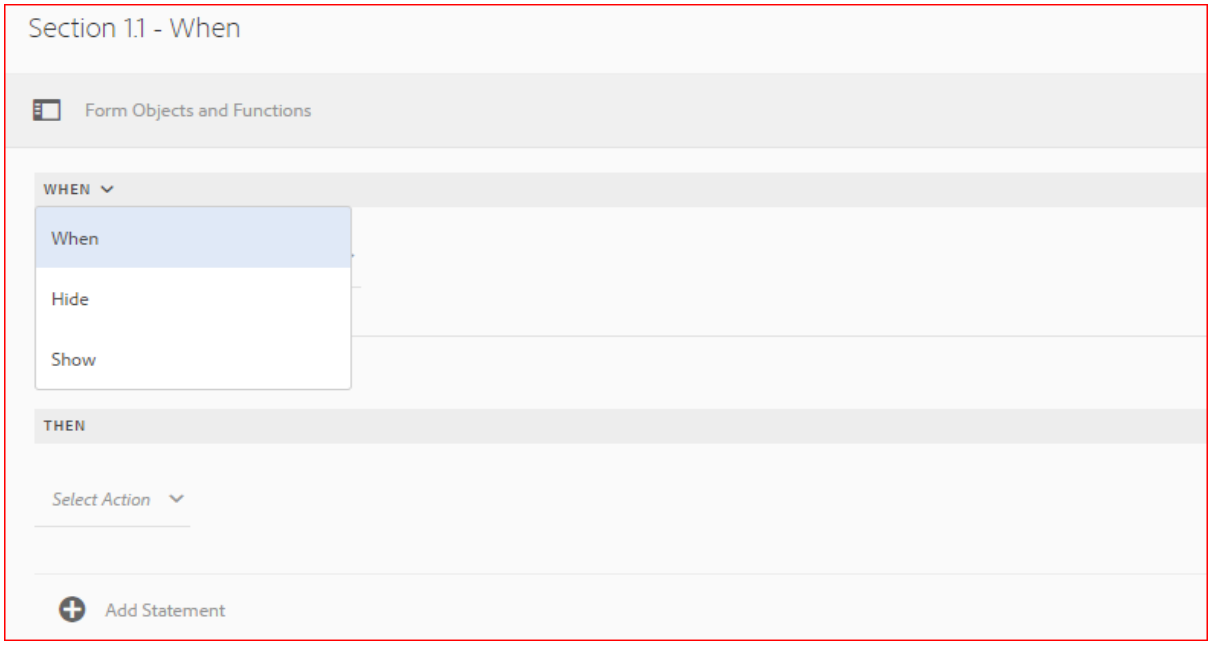


Fig 4.7 Rules for Forms

- Set the desired operation eg.

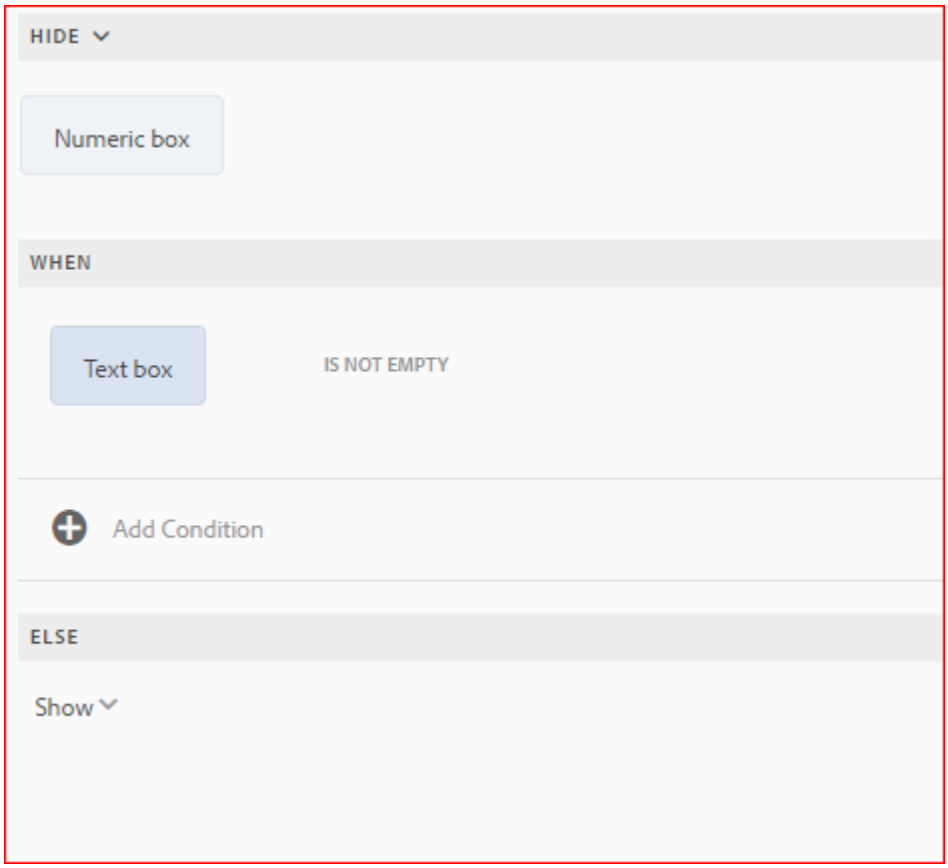


Fig. 4.8 Rule for Forms

A form designer is used to create common segments which will be used generic for all the forms like name and address, family details, income details, and so on. These reusable, generic and standalone segments are called adaptive form fragments and are widely used to design forms in AEM.

- Select create adaptive form fragment from form console

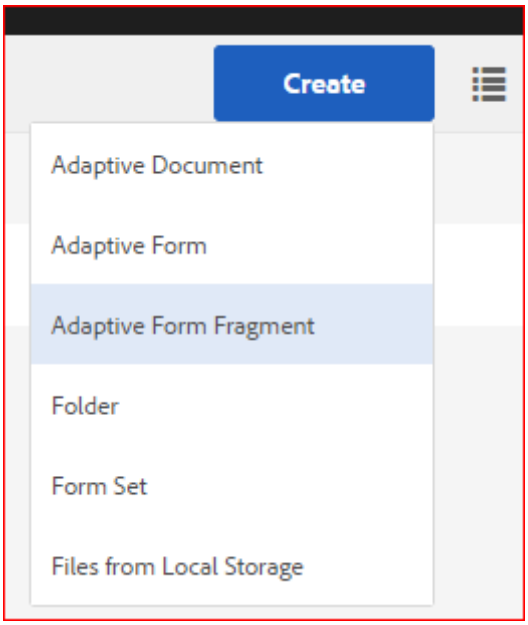


Fig. 4.9 Adaptive Form

- Fill the basic details like name and description.
- A blank page with a parsys will appear where we can drop.

While making the forms we can see how it will look in different breakpoints (i.e. Screen sizes). This feature is available at top of the form which can help in emulation.

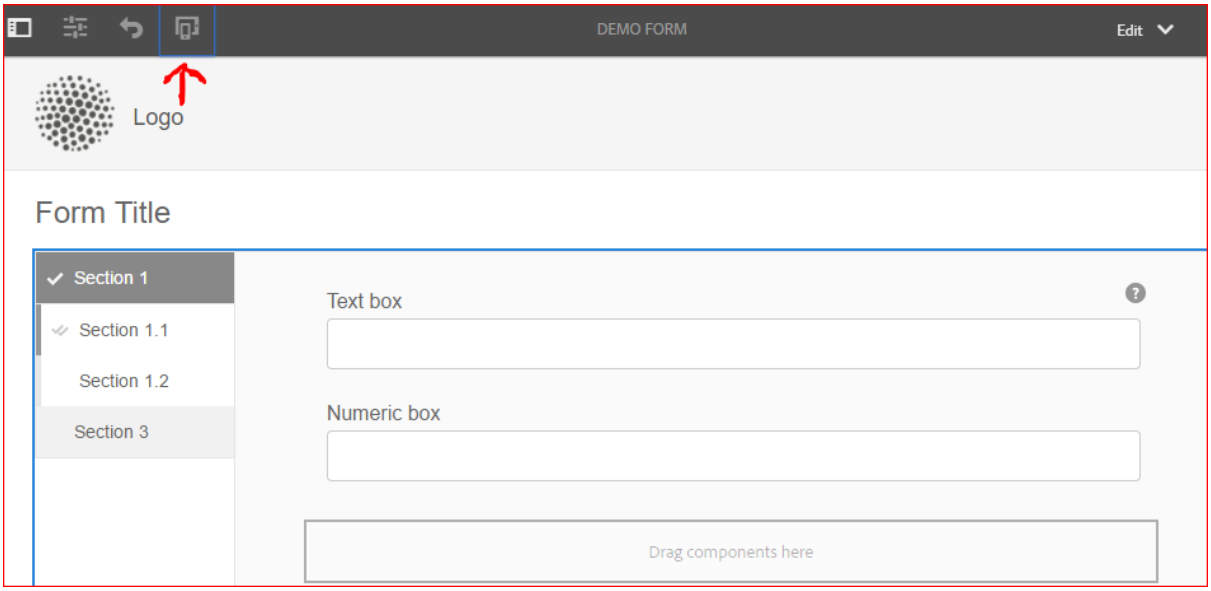


Fig. 4.10 Breakpoint Renditions

- We can select device and change its orientation very easily with few mouse click

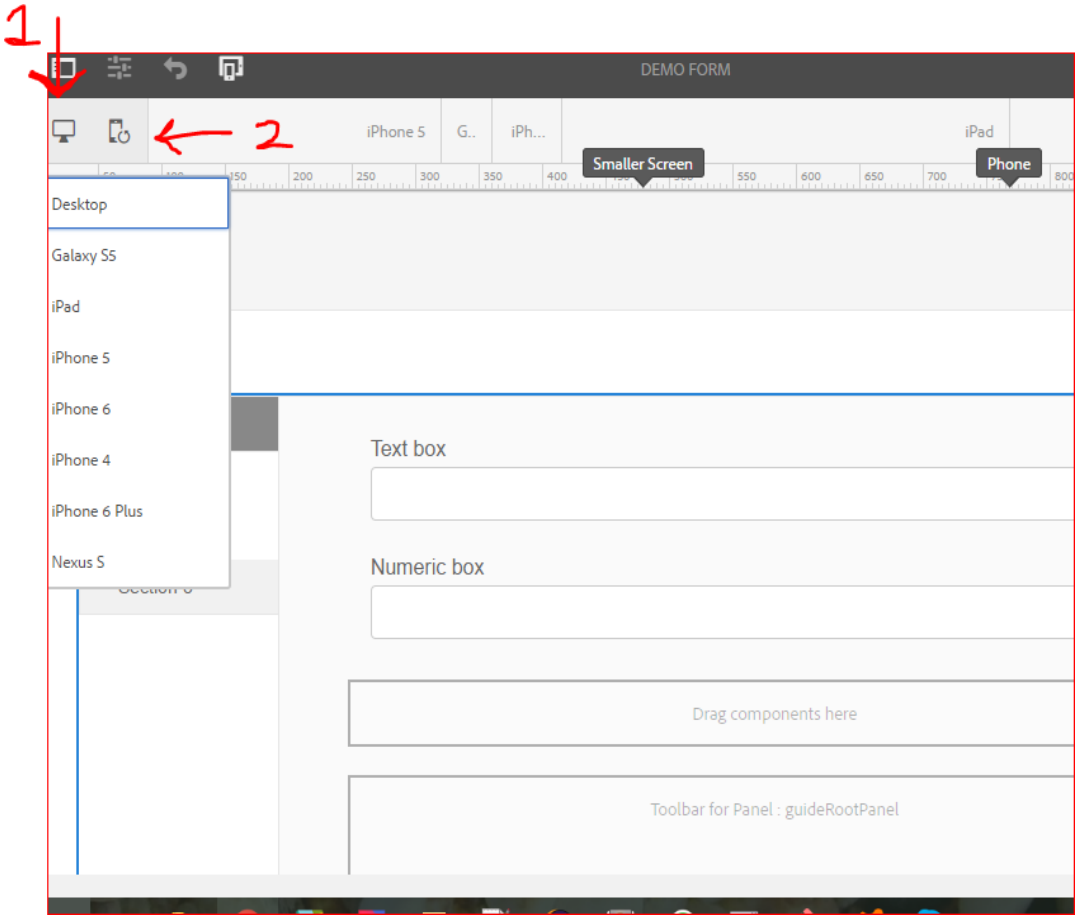


Fig 4.11 Different media screens

4.3. WORKFLOWS IN AEM

There are plethora of scenarios in a project that occur continuously and hence the need is to automate these processes so that they can work without explicitly calling the processes. Such use cases for automation of some steps or activities in a project can be achieved with the help of workflows in AEM. Workflows typically define a sequence of steps to be followed after some action has been performed.

Workflow is a well defined series of steps that allows to automate activities of Experience Manager. It is divided into various steps including people (participants), procedures (process) or some tools. The flow of workflow can be either sequential or parallel when multiple procedures can run parallely to each other. In short, a workflow is similar to a flow-chart which describes the series of procedures to be executed having multiple participants to achieve a desired outcome.

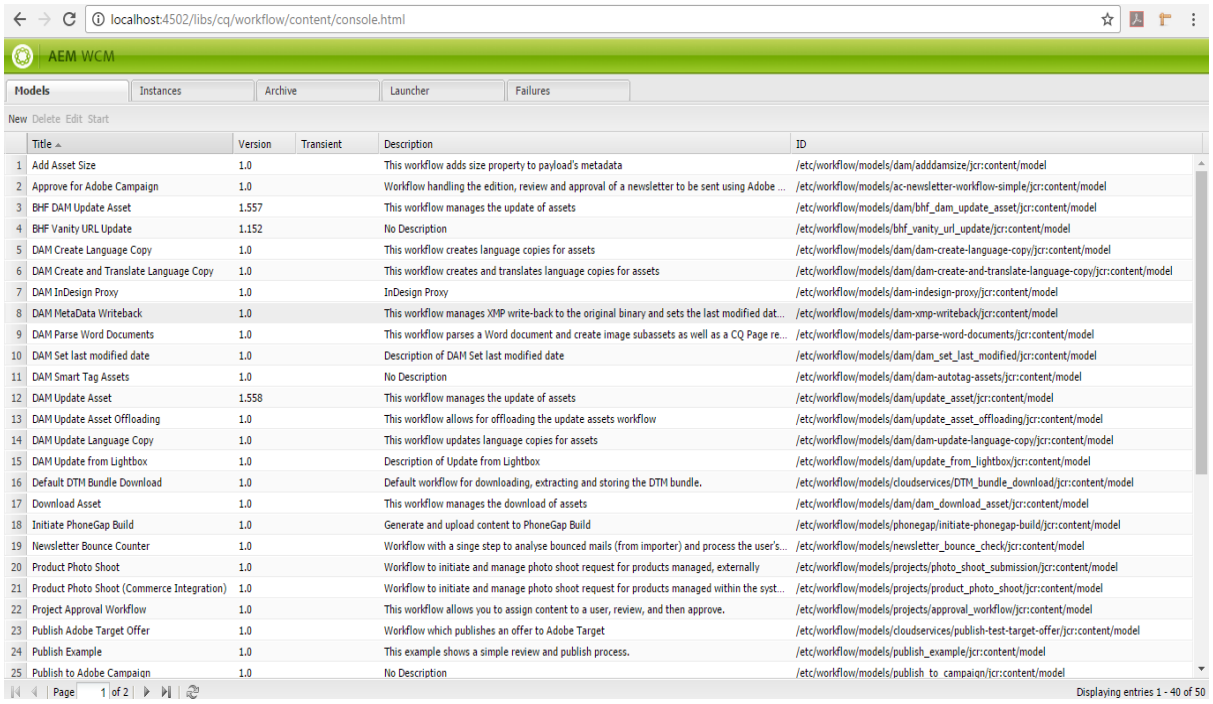
There are many out of the box workflow models available for use and we can write our custom workflows which can be tailored as per the business needs of the company.

There can be plethora of use cases for writing workflows that may depend on the business needs. For example, the process of publishing a page can be customized and tailored for inclusion of some special participants that need to approve the request for activation or send a mail after approval or rejection.

Workflow Model is a container or blueprint of the workflow that combines the workflow steps and link them logically. It has a start and an end node (plus an optional participant step) by default. It can contain different nodes and the transitions that link them. The workflow nodes can get any input from author or can do processing via Java code or ECMA scripts. Workflow payload is an important term which is considered as a resource on which workflow will perform it's functions.

List of workflows can be seen on http://localhost:4502/libs/cq/workflow/content/console.html

.



	Title	Version	Transient	Description	ID
1	Add Asset Size	1.0		This workflow adds size property to payload's metadata	/etc/workflow/models/dam/adddamsize/jcr:content/model
2	Approve for Adobe Campaign	1.0		Workflow handling the edition, review and approval of a newsletter to be sent using Adobe ...	/etc/workflow/models/ac-newsletter-workflow-simple/jcr:content/model
3	BHF DAM Update Asset	1.557		This workflow manages the update of assets	/etc/workflow/models/dam/bhf_dam_update_asset/jcr:content/model
4	BHF Vanity URL Update	1.152		No Description	/etc/workflow/models/bhf_vanity_url_update/jcr:content/model
5	DAM Create Language Copy	1.0		This workflow creates language copies for assets	/etc/workflow/models/dam/dam-create-language-copy/jcr:content/model
6	DAM Create and Translate Language Copy	1.0		This workflow creates and translates language copies for assets	/etc/workflow/models/dam/dam-create-and-translate-language-copy/jcr:content/model
7	DAM InDesign Proxy	1.0		InDesign Proxy	/etc/workflow/models/dam-indesign-proxy/jcr:content/model
8	DAM MetaData Writeback	1.0		This workflow manages XMP write-back to the original binary and sets the last modified dat...	/etc/workflow/models/dam-xmp-writeback/jcr:content/model
9	DAM Parse Word Documents	1.0		This workflow parses a Word document and create image subassets as well as a CQ Page re...	/etc/workflow/models/dam-parse-word-documents/jcr:content/model
10	DAM Set last modified date	1.0		Description of DAM Set last modified date	/etc/workflow/models/dam/dam_set_last_modified/jcr:content/model
11	DAM Smart Tag Assets	1.0		No Description	/etc/workflow/models/dam/dam-autotag-assets/jcr:content/model
12	DAM Update Asset	1.558		This workflow manages the update of assets	/etc/workflow/models/dam/update_asset/jcr:content/model
13	DAM Update Asset Offloading	1.0		This workflow allows for offloading the update assets workflow	/etc/workflow/models/dam/update_asset_offloading/jcr:content/model
14	DAM Update Language Copy	1.0		This workflow updates language copies for assets	/etc/workflow/models/dam/dam-update-language-copy/jcr:content/model
15	DAM Update from Lightbox	1.0		Description of Update from Lightbox	/etc/workflow/models/dam/update_from_lightbox/jcr:content/model
16	Default DTM Bundle Download	1.0		Default workflow for downloading, extracting and storing the DTM bundle.	/etc/workflow/models/cloudservices/DTM_bundle_download/jcr:content/model
17	Download Asset	1.0		This workflow manages the download of assets	/etc/workflow/models/dam/dam_download_asset/jcr:content/model
18	Initiate PhoneGap Build	1.0		Generate and upload content to PhoneGap Build	/etc/workflow/models/phonegap/initiate-phonegap-build/jcr:content/model
19	Newsletter Bounce Counter	1.0		Workflow with a singe step to analyse bounced mails (from importer) and process the user's...	/etc/workflow/models/newsletter_bounce_check/jcr:content/model
20	Product Photo Shoot	1.0		Workflow to initiate and manage photo shoot request for products managed, externally	/etc/workflow/models/projects/photo_shoot_submission/jcr:content/model
21	Product Photo Shoot (Commerce Integration)	1.0		Workflow to initiate and manage photo shoot request for products managed within the syst...	/etc/workflow/models/projects/product_photo_shoot/jcr:content/model
22	Project Approval Workflow	1.0		This workflow allows you to assign content to a user, review, and then approve.	/etc/workflow/models/projects/approval_workflow/jcr:content/model
23	Publish Adobe Target Offer	1.0		Workflow which publishes an offer to Adobe Target	/etc/workflow/models/cloudservices/publish-test-target-offer/jcr:content/model
24	Publish Example	1.0		This example shows a simple review and publish process.	/etc/workflow/models/publish_example/jcr:content/model
25	Publish to Adobe Campaign	1.0		No Description	/etc/workflow/models/publish_to_campaign/jcr:content/model

Fig 4.12: AEM Workflow Console

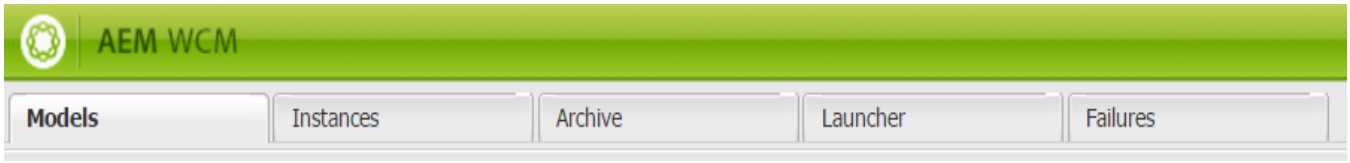


Fig. 4.13: Tabs in Workflow Console

These are the tabs that contain the information about the workflows.

- Models tab has workflow models having title of the workflow, version, description, and model id.
- Instances tab lists the active instances of workflows having status (running, suspended etc.), initiator, start time, workflow model, payload associated with the workflow.
- Archive tab contains the history of all the workflow instances. It contains the status (completed, aborted etc.), initiator, start and end time, workflow model, payload, workflow title etc.
- Launcher tab contains the launchers of workflow models that trigger initiation of workflows automatically on some action. It contains fields such as event type (created, deleted, modified), nodetype, globbing or path specified for action of node, condition, workflow model that needs to be triggered, description, value of enabled, exclusion list, run modes.

- Failure tab shows failed instances of workflow on which some error occurred during execution. The issues can be fixed and workflow instances can be resumed.

To create a new workflow we can add the title and the name (cannot contain spaces) of the workflow by clicking on New tab.

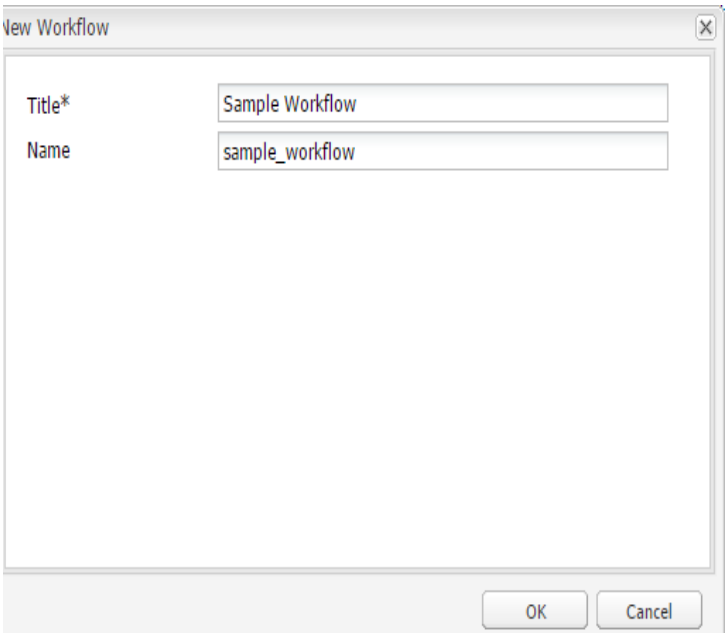


Fig 4.14: Create a new Workflow Model

Thereafter, we can edit the workflow created to customize it. This is the by default view of the workflow wherein there is a start and an end node with a default participant step.

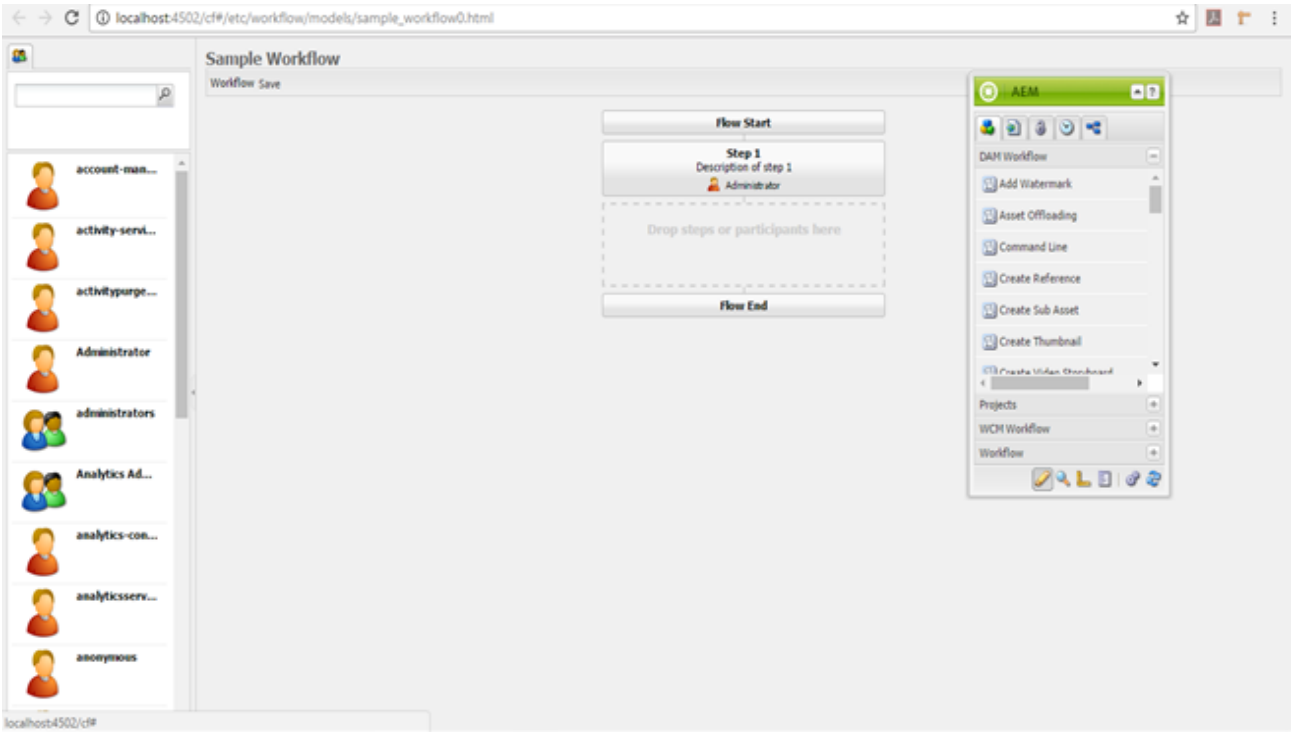


Fig. 4.15: By default view of workflow

We can view the logical layout of the workflow in the console whereas to see the node structure stored for the workflow we can see /etc/workflow/models/sample_workflow.

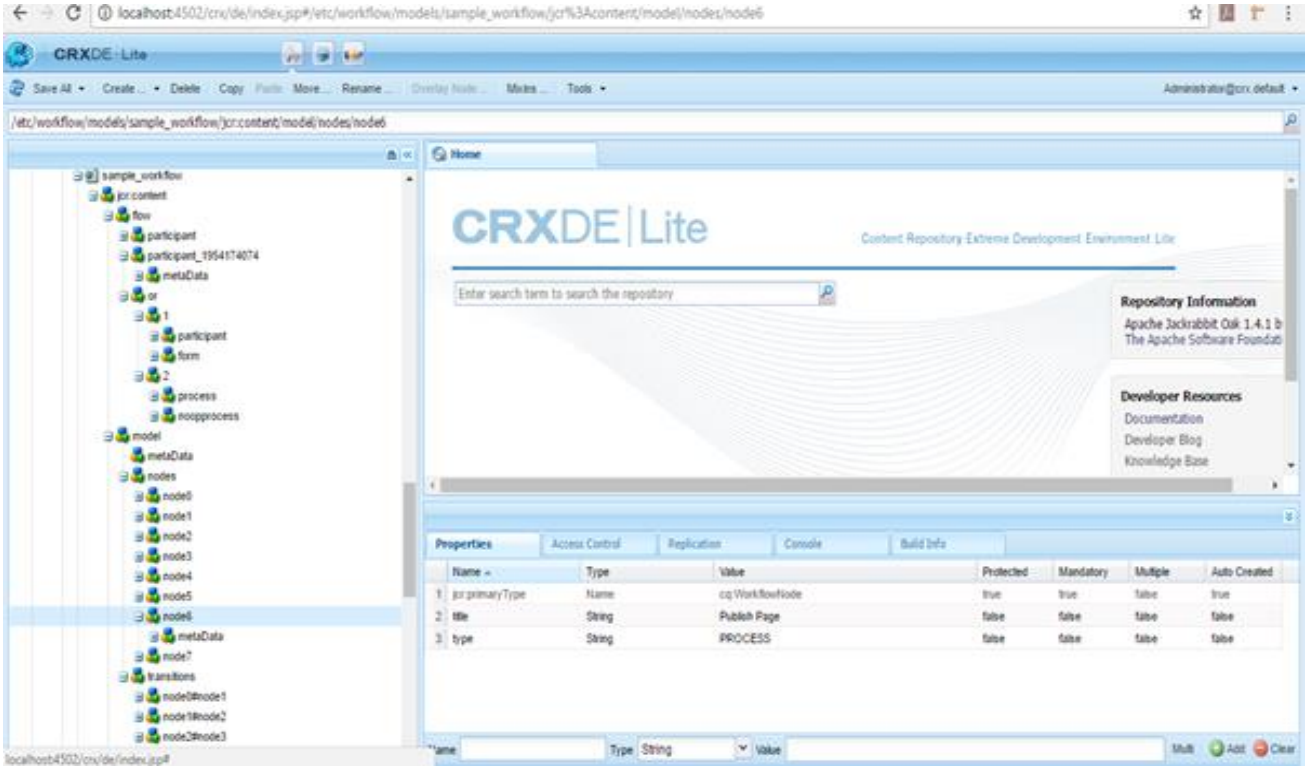


Fig. 4.16: Node structure of workflow in CRX

It contains metadata about the workflow stored in JCR. We can notice flow and model nodes. It has the nodes between start and end node present in the workflow. The model node has all the steps added in the workflow marked as nodes alongwith the transitions they have between them. We can see the metadata of node 6 in the snapshot since it is a process step having title as Publish Page.

We can see the running instance of the workflow under /etc/workflow/instance path is crx.

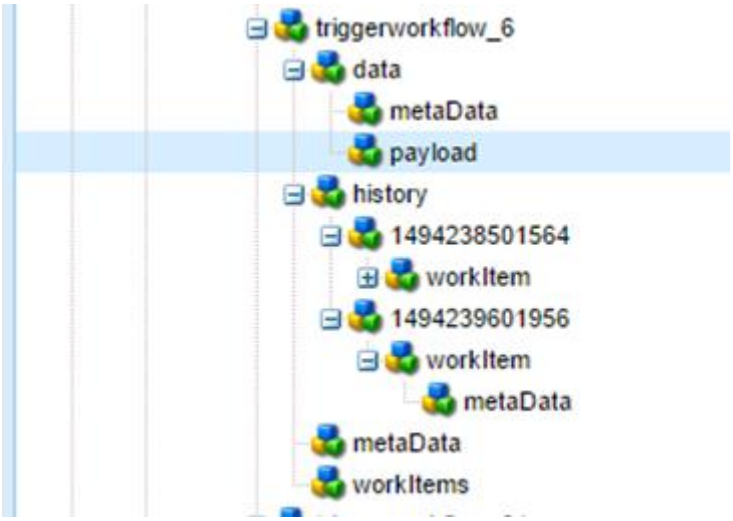


Fig 4.17: Nodes in triggerworkflow instance

This is an example of triggerworkflow having data (metadata and payload of which workflow instance is working upon), history, metadata of a particular step and workItems associated with the workflow.

There are many workflow nodes that can be dragged and dropped such as Participant step, process step, and split, or split, dialog participant step, form participant step, container step, goto step etc.

This is a participant step that can be dragged and dropped from the sidekick from workflow tab.

In this step the user can give a title and description and select a user or a group without whose permission/ approval the workflow cannot proceed to the next step. The email option will send an email in the inbox of the user/group.

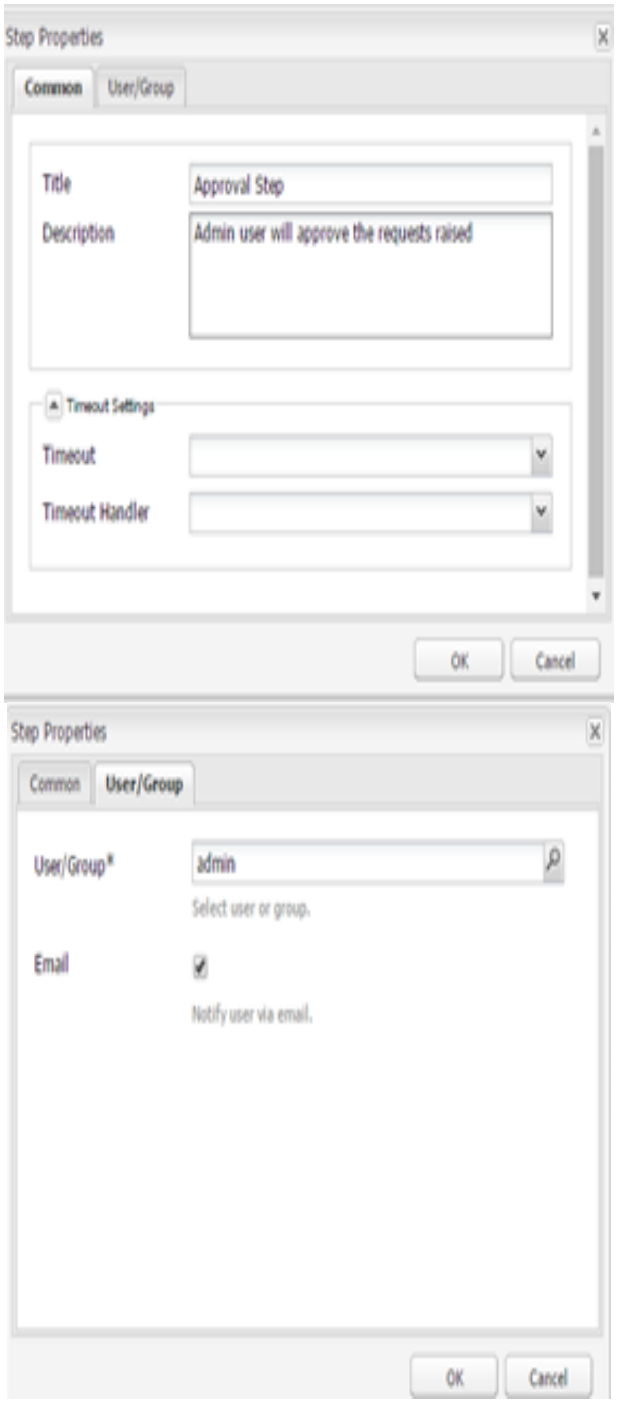


Fig 4.18: Participant Step

This is a process step which can be similarly dragged and dropped form the sidekick in the workflow. In this step the common tab can contain title and description of the process step. There are timeout settings too in which we can set the timeout limit of the workflow after which the timeout handler will start it's execution. In the process tab we select the process from the dropdown (We can add our own process in the list using java). We have to check the

option of Handler Advance so as to allow the workflow process step to proceed to the next step after performing it's task. There is another field named as Arguments in which we can pass the arguments to the process step for function and task.

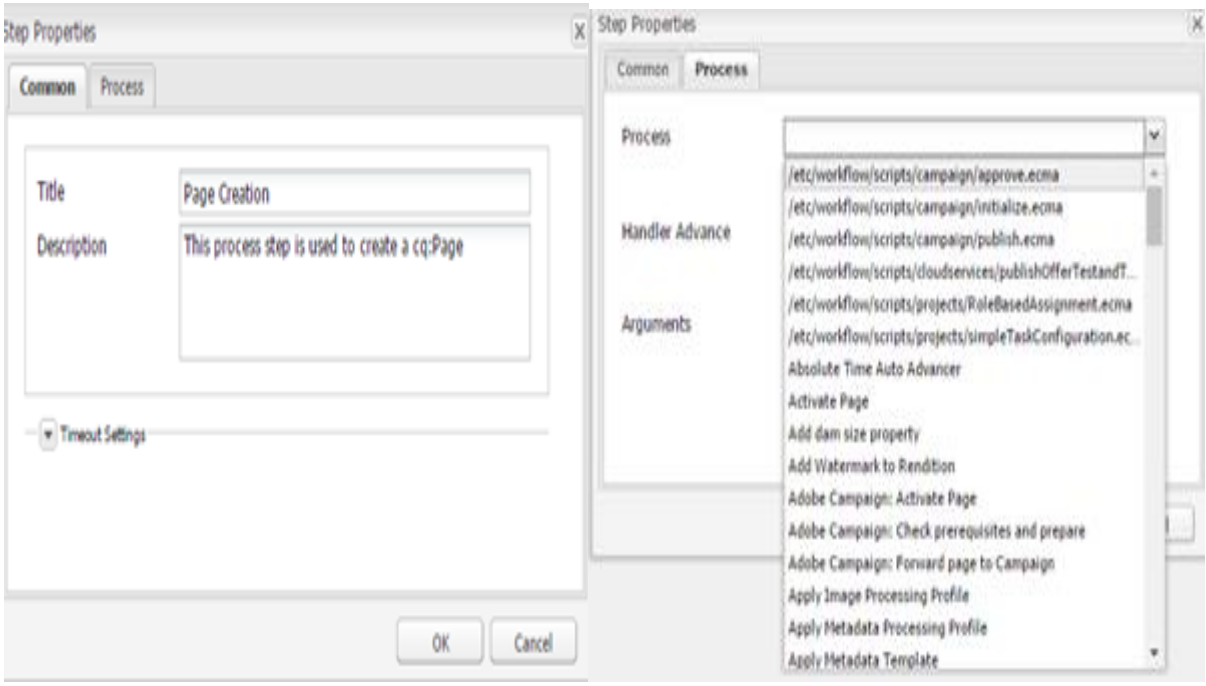


Fig. 4.19: Process Step

Now, to add our own process in the list we have to make a workflow process and register it as a service in java and can add the logical code that will execute on this process step.

```
@Component(immediate = true, enabled= true, metatype = true)
@Service(WorkflowProcess.class)
@Property(name = "process.label", value = "Create Page Service", propertyPrivate =
true)
public class TriggerWorkflow implements WorkflowProcess {

@Override
public void execute(WorkItem workItem, WorkflowSession workflowSession,
MetaDataMap  args)
    throws WorkflowException {
```

This code snippet shows the basic skeleton of a workflow process that it required for process step.

@Component will register our Java class as a component with the properties that enables it immediately and helps it get the metadata provided by the execute method of WorkflowProcess class.
@Service will register our Java class as a service to be used in the Workflow process.
@Properties will add certain properties such as process.label which will show our class in the process list of the process step. The value of this property will be shown in the drop down list.

The Java class should implement WorkflowProcess interface and give definition to execute method. This method provides three arguments namely WorkItem, WorkflowSession, MetaDataMap.

- WorkItem defines the currently started java process. It contains the data of the workflow and can be fetched using this argument such as Id, workflow, workflow data etc. We can access the workflow object using this. We can get payload of the data using a method of workItem.

```
String payload = workItem.getWorkflowData().getPayload().toString();
```

- WorkflowSession controls the workflow by providing methods that can deploy, complete, create, start, stop, resume, suspend, retrieve, complete workflow models, instances or user/groups. We can start another workflow using workflow session of current workflow.
- MetadataMap is a Map<String, Object> of properties that are entered in the dialog of the process step. We can fetch the list of the arguments from metadata map.

There can be many other types of steps that can be configured within the workflow and can be tailored using java code, scripts, parameters etc. to provide the desired functionality.
An OR split can be used to introduce branching in the workflow with which only one branch of the workflow can be activated (conditional) based on the script which is mentioned in the dialog. The user can give either the path of the script or can write ECMA script in the given tab. The Common tab has an option to select the number of branches required in the OR Split.

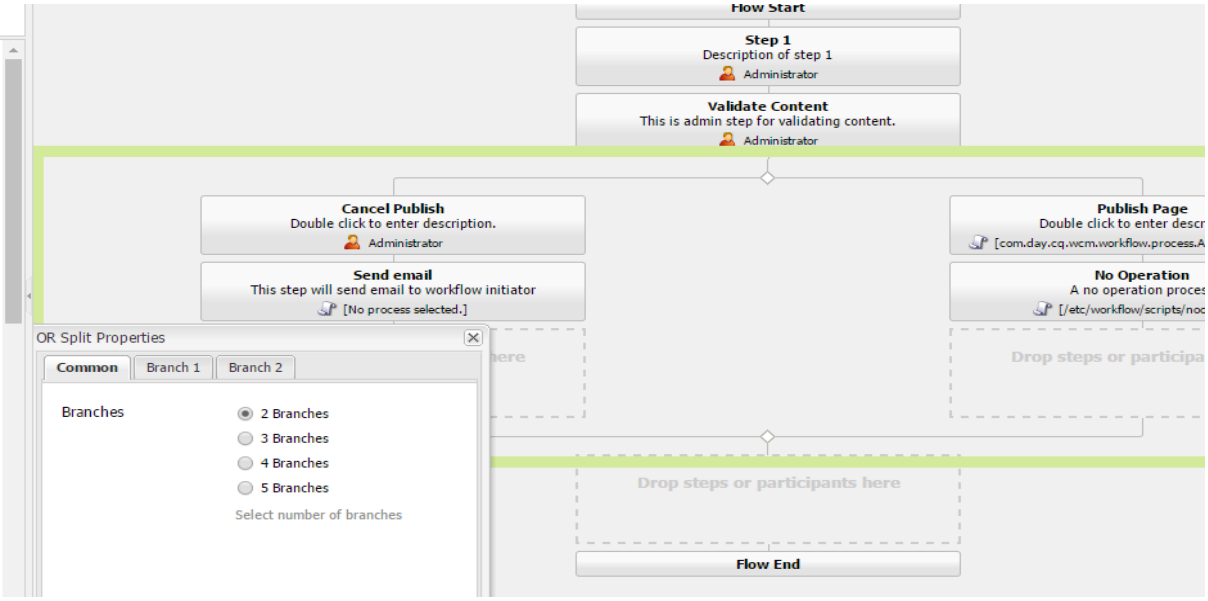
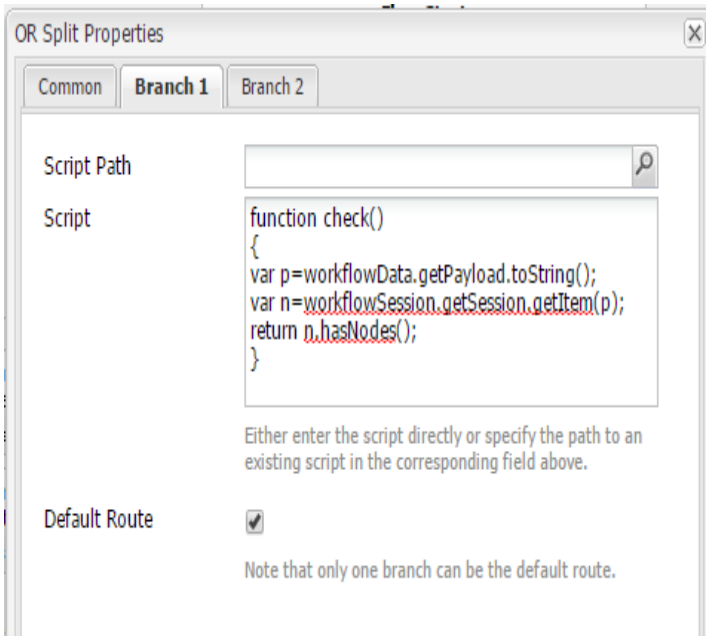


Fig 4.20: OR Split (marked with green)



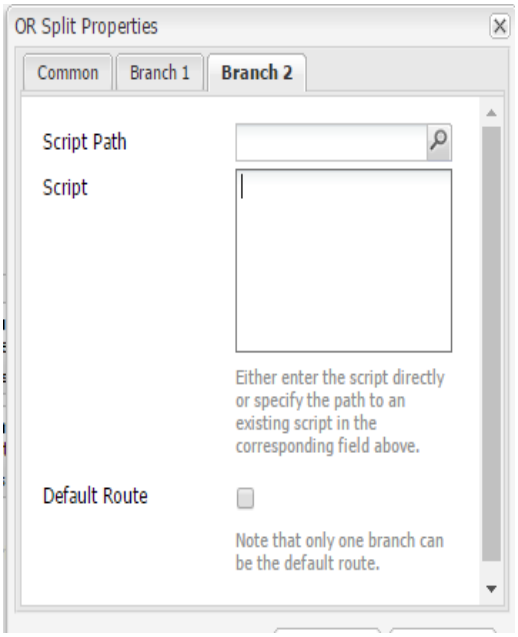


Fig 4.21: ECMA script written for branch 1 (set as default route).

We can write different condition for branch 2 or can upload a script.

Similarly there is an AND split wherein all the branches go parallely and we can execute tasks for multiple processing.

A Container step can allow another workflow to be executed as a child of the current workflow. This workflow is a sub workflow that can be instantiated. Another way of performing the same action is via Java code where we can start a workflow using workflowSession.

```
WorkflowModel workflowModel =
workflowSession.getModel("/etc/workflow/models/childworkflow/jcr:content/model");
WorkflowData workflowData =
workflowSession.newWorkflowData("JCR_PATH",
"/content/application/poc/sampleWorkflowPage");


workflowSession.startWorkflow(workflowModel, workflowData);
```

This code snippet gets the model of the child workflow that needs to be invoked. Thereafter, the data of the workflow is created which is used to start the workflow in the current session.

Goto step traverses between the steps in the workflow based on the result of the ECMA script. It can be used to implement the use case of looping in the workflow process.

Dialog participant step is a special scenario of participant step where the user/group needs to

give some information that is to be used in the workflow. The additional step in this step is to provide a path of the dialog. We have to make that dialog in CRX (/etc/workflows/dialog) and provide the path of that dialog to receive the input from the selected user/group. The structure of the dialog should be cq:dialog type. This data can be stored in the form of payload (when we set the value of the property as ./jcr:content/nodename) which can be overwritten in subsequent uses of this dialog. The data can also be stored with the work item (when the value is nodename) metadata and is persisted with subsequent use of dialog.

Workflow launchers can be used to trigger the workflow based on some event that can be specified in the launcher. We can add a launcher under the Launcher tab.

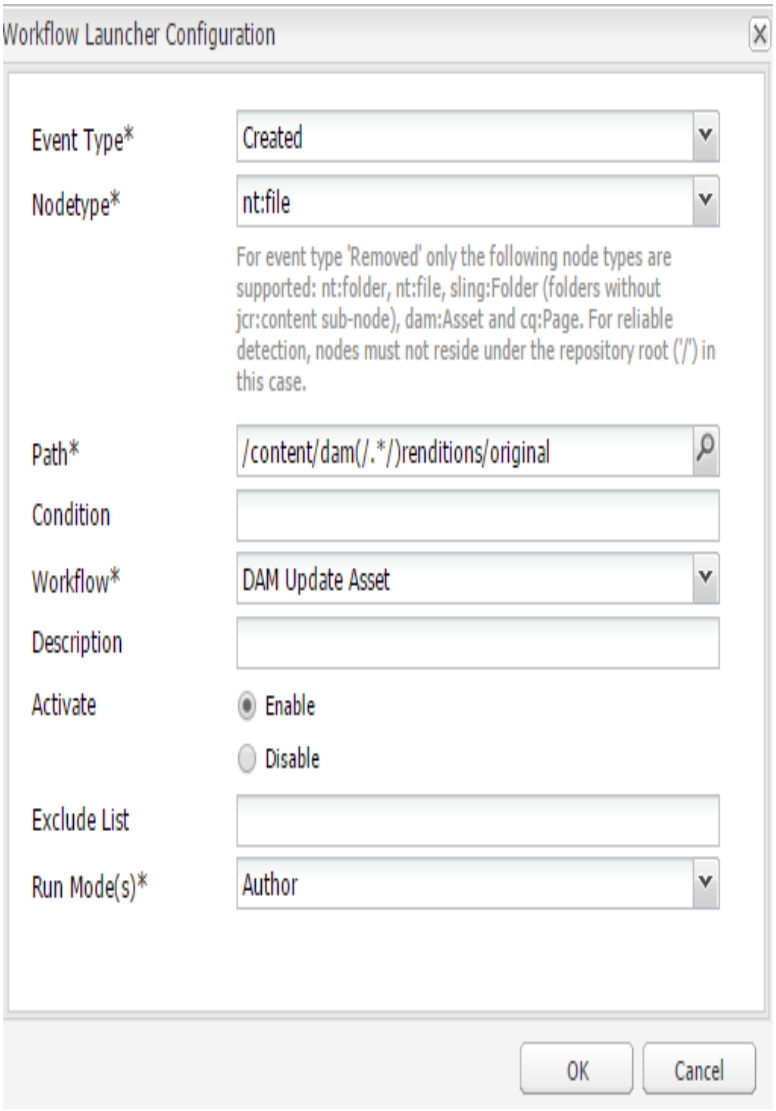


Figure 4.22: Creation of Launcher for Workflow

The event type can be either created, modified or deleted i.e. the action on which the launcher will trigger the workflow mentioned. The nodetype is for the type of the node on which the launcher will apply to start the workflow. The path specifies the path on which this action will be applied to during the workflow initiation i.e. on the which the launcher of the workflow will be applied to. The condition tab determines the condition to be applied when the workflow is launched. This is optional in case a user wants to start the launcher based on a specific condition only. The launcher needs to be associated with a workflow model so that it can be determined that what will be triggered with the help of this launcher. The description can be set and activate should be enabled to enable the launch of the workflow launcher. The exclude list is a comma separated list of JCR events or items to be ignored via workflow triggering. The run modes in this dialog is to specify the server on which this will be applied to (author, publish, author & publish).

Workflows automate some processes and are composed of participants, processes and events. They can either be invoked by some other workflow or triggered by a launcher or triggered on some action. We can use either out of the box workflows, processes or can create the custom workflows or our own processes to fulfill different use cases.

4.4. LOGS IN AEM

Troubleshooting is one of the most soughted feature in AEM and logs help in troubleshooting by providing details about the functioning of AEM. Logs are present in crx-quickstart>logs path in the file system.

- Request.log: It contains a set of requests and responses for AEM instance. The performance of that particular instance can be monitored by looking into the output of this log file. Requests may consist of HEAD, GET, POST, PUT as methods, protocol being HTTP or HTTPS. The response with have details of status code, MIME Type, and the response time taken by the header.
- Access.log: It gives information about who is accessing which resource and at what time.
- Stdout.log: These contain the messages that are written in the Java files for the logging. All the INFO messages are stored in this log.

stdout.log - Notepad

File Edit Format View Help

```
Loading quickstart properties: default
Loading quickstart properties: instance
Quickstart startup at Fri May 05 20:43:05 IST 2017
UpgradeUtil.handleInstallAndUpgrade has mode RESTART
05.05.2017 20:43:06.228 *INFO * [main] Setting sling.home=D:\aem jar\crx-quickstart (command line)
05.05.2017 20:43:06.228 *INFO * [main] Starting Apache Sling in D:\aem jar\crx-quickstart
05.05.2017 20:43:06.268 *INFO * [main] Sling  Extension Lib Home : D:\aem jar\crx-quickstart\launchpad\ext
05.05.2017 20:43:06.300 *INFO * [main] Checking launcher JAR in folder D:\aem jar\crx-quickstart\launchpad
05.05.2017 20:43:06.324 *INFO * [main] Existing launcher is up to date, using it: 5.4.0.2_6_10-B004 (org.apache.sling.launchpad.base.jar)
05.05.2017 20:43:06.324 *INFO * [main] Loading launcher class org.apache.sling.launchpad.base.app.MainDelegate from org.apache.sling.launchpad.base.jar
05.05.2017 20:43:06.328 *INFO * [main] External Libs Home (ext) is null or does not exists.
05.05.2017 20:43:06.384 *INFO * [main] Setting sling.properties=conf/sling.properties
05.05.2017 20:43:06.384 *INFO * [main] Setting sling.home=D:\aem jar\crx-quickstart
05.05.2017 20:43:06.384 *INFO * [main] Setting sling.launchpad=D:\aem jar\crx-quickstart\launchpad
05.05.2017 20:43:06.384 *INFO * [main] Setting org.osgi.service.http.port=4502
05.05.2017 20:43:06.384 *INFO * [main] Starting launcher ...
05.05.2017 20:43:06.396 *INFO * [main] HTTP server port: 4502
05.05.2017 20:43:59.275 *INFO* [FelixStartLevel] org.apache.sling.commons.logservice Service [org.apache.sling.commons.logservice.internal.LogServiceFactory,
05.05.2017 20:43:59.283 *INFO* [FelixStartLevel] org.apache.sling.commons.logservice Service [org.apache.sling.commons.logservice.internal.LogReaderServiceFa
05.05.2017 20:43:59.283 *INFO* [FelixStartLevel] org.apache.sling.commons.logservice BundleEvent STARTED
05.05.2017 20:43:59.303 *INFO* [FelixStartLevel] org.apache.sling.installer.core BundleEvent RESOLVED
05.05.2017 20:43:59.303 *INFO* [FelixStartLevel] org.apache.sling.installer.core BundleEvent STARTING
05.05.2017 20:43:59.659 *INFO* [FelixStartLevel] org.apache.sling.installer.core Service [Apache Sling Bundle Install Task Factory,20, [org.apache.sling.inst
05.05.2017 20:43:59.735 *INFO* [FelixStartLevel] org.apache.sling.installer.core Service [Apache Sling Installer Controller Service,22, [org.apache.sling.ins
05.05.2017 20:43:59.735 *INFO* [FelixStartLevel] org.apache.sling.installer.core Service [Apache Sling Installer Controller Service,23, [org.apache.sling.ins
05.05.2017 20:43:59.739 *INFO* [FelixStartLevel] org.apache.sling.installer.core BundleEvent STARTED
05.05.2017 20:43:59.759 *INFO* [OsgiInstallerImpl] org.apache.sling.installer.core.impl.OsgiInstallerImpl Apache Sling OSGi Installer Service started.
05.05.2017 20:43:59.771 *INFO* [FelixStartLevel] org.apache.sling.installer.factory.configuration BundleEvent RESOLVED
05.05.2017 20:43:59.771 *INFO* [FelixStartLevel] org.apache.sling.installer.factory.configuration BundleEvent STARTING
Attempting to load ESAPI.properties via file I/O.
Attempting to load ESAPI.properties as resource file via file I/O.
Not found in 'org.owasp.esapi.resources' directory or file not readable: D:\aem jar\ESAPI.properties
Not found in SystemResource Directory/resourceDirectory: .esapi\ESAPI.properties
Not found in 'user.home' (C:\Users\sukri) directory: C:\Users\sukri\esapi\ESAPI.properties
```

Fig. 4.23 stdout.log

- Stderr.log: This contains the error messages with complete details (if any).

stderr.log - Notepad

File Edit Format View Help

```
        at java.lang.Thread.run(Thread.java:745)
Opening browser using cmd=rundll32 url.dll,FileProtocolHandler "http://localhost:4502/"
Startup time:217 seconds
http://localhost:4502/
Quickstart started
Exception in thread "CQWorkflowStatisticsService-Processor" java.lang.NullPointerException
        at com.day.cq.workflow.impl.ServiceLoginUtil.createWorkflowServiceSession(ServiceLoginUtil.java:52)
        at com.day.cq.workflow.impl.statistics.CQWorkflowStatisticsService.getWorkflowSession(CQWorkflowStatisticsService.java:283)
        at com.day.cq.workflow.impl.statistics.CQWorkflowStatisticsService.processEvent(CQWorkflowStatisticsService.java:221)
        at com.day.cq.workflow.impl.statistics.CQWorkflowStatisticsService$EventProcessor.run(CQWorkflowStatisticsService.java:206)
        at java.lang.Thread.run(Thread.java:745)
Exception in thread "CQWorkflowStatisticsService-Processor" java.lang.NullPointerException
        at com.day.cq.workflow.impl.ServiceLoginUtil.createWorkflowServiceSession(ServiceLoginUtil.java:52)
        at com.day.cq.workflow.impl.statistics.CQWorkflowStatisticsService.getWorkflowSession(CQWorkflowStatisticsService.java:283)
        at com.day.cq.workflow.impl.statistics.CQWorkflowStatisticsService.processEvent(CQWorkflowStatisticsService.java:221)
        at com.day.cq.workflow.impl.statistics.CQWorkflowStatisticsService$EventProcessor.run(CQWorkflowStatisticsService.java:206)
        at java.lang.Thread.run(Thread.java:745)
Exception in thread "CQWorkflowStatisticsService-Processor" java.lang.NullPointerException
        at com.day.cq.workflow.impl.ServiceLoginUtil.createWorkflowServiceSession(ServiceLoginUtil.java:52)
        at com.day.cq.workflow.impl.statistics.CQWorkflowStatisticsService.getWorkflowSession(CQWorkflowStatisticsService.java:283)
        at com.day.cq.workflow.impl.statistics.CQWorkflowStatisticsService.processEvent(CQWorkflowStatisticsService.java:221)
        at com.day.cq.workflow.impl.statistics.CQWorkflowStatisticsService$EventProcessor.run(CQWorkflowStatisticsService.java:206)
        at java.lang.Thread.run(Thread.java:745)
Exception in thread "CQWorkflowStatisticsService-Processor" java.lang.NullPointerException
        at com.day.cq.workflow.impl.ServiceLoginUtil.createWorkflowServiceSession(ServiceLoginUtil.java:52)
        at com.day.cq.workflow.impl.statistics.CQWorkflowStatisticsService.getWorkflowSession(CQWorkflowStatisticsService.java:283)
        at com.day.cq.workflow.impl.statistics.CQWorkflowStatisticsService.processEvent(CQWorkflowStatisticsService.java:221)
        at com.day.cq.workflow.impl.statistics.CQWorkflowStatisticsService$EventProcessor.run(CQWorkflowStatisticsService.java:206)
        at java.lang.Thread.run(Thread.java:745)
Exception in thread "EMailNotificationService-Processor" java.lang.NullPointerException
        at com.day.cq.workflow.impl.email.NotificationHelper.getEmailAddress(NotificationHelper.java:150)
        at com.day.cq.workflow.impl.email.NotificationHelper.getParticipants(NotificationHelper.java:110)
        at com.day.cq.workflow.impl.email.NotificationHelper.getParticipants(NotificationHelper.java:91)
        at com.day.cq.workflow.impl.email.EMailNotificationService.sendNotification(EMailNotificationService.java:241)
```

Fig. 4.24 Stderr.log

- Upgrade.log: This log contains all the upgrade operations that are being performed. The level of this log is set to Info by default.
- Error.log: This log shows various messages of INFO, DEBUG, ERROR etc.

4.5. REPORTS IN AEM

AEM provides a functionality to have an access of default reports which can be custom configured by the users.

- Page Activity Report
- Workflow report
- Workflow Instance Report
- User Report
- Disk Usage
- Health Check
- Component Report

The reports can be seen on the path: **localhost:4502/miscadmin#/etc/reports**

We can create our custom reports too by choosing a template and naming it as per our needs. We can add fields to the report from the sidekick. We can click the column title and see the list of operations provided in the drop down. The columns can be arranged in ascending or descending order. Graphical representation of groups can be done with group by command. Filters can be applied too.

- Page Activity Report

This is the report of the pages and the actions that occur on the page. This report will monitor pages that are recently or least used, active/ inactive users, latest modifications made to the pages. The information in this report is populated from the audit log. The information of this report includes page, time, type, and the user.

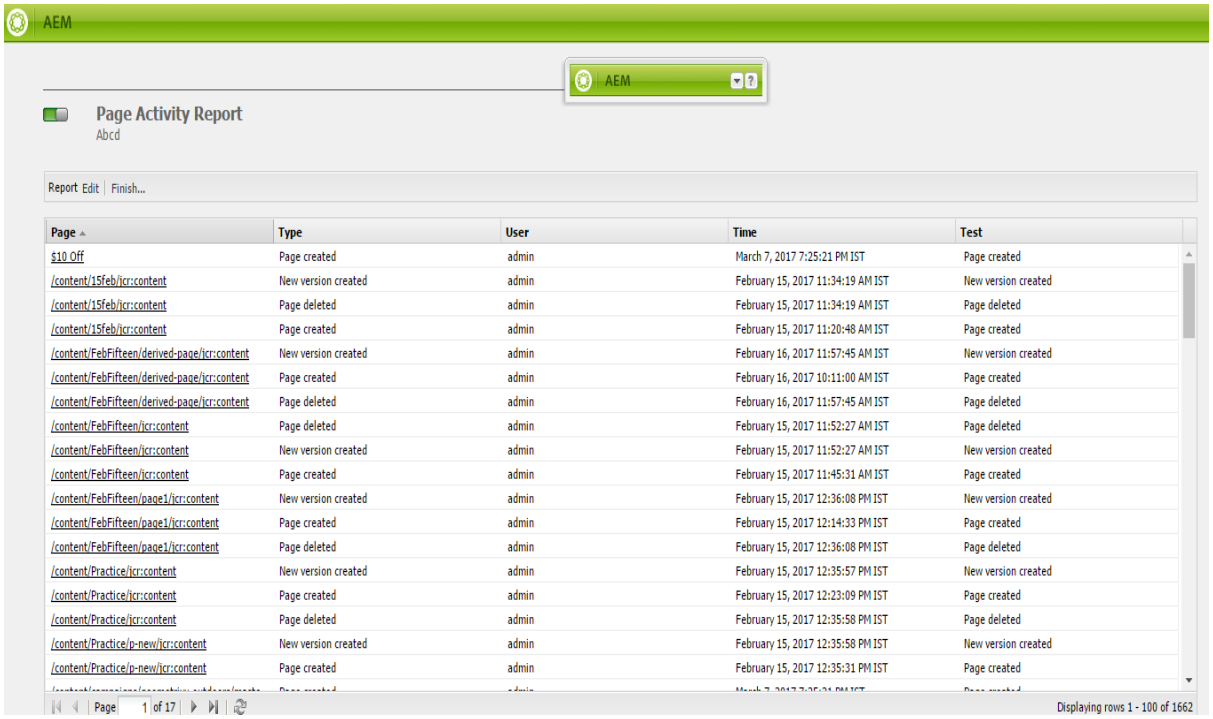


Fig. 4.25 Page Activity Report

- Workflow Report

This report is the statistics of the running workflows on the present instance.

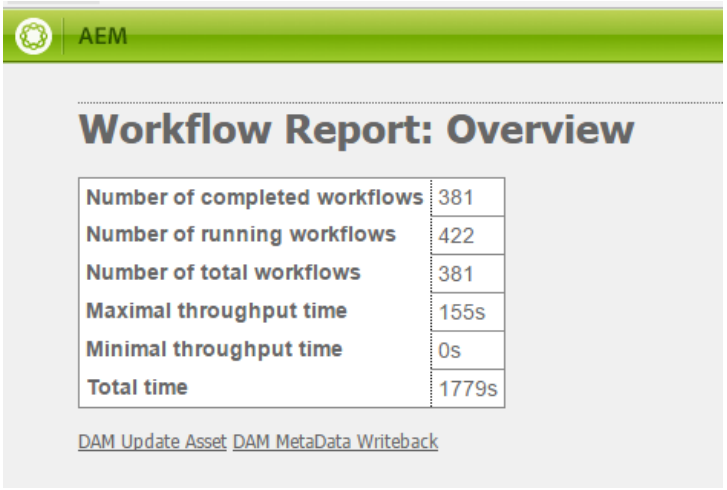


Fig. 4.26 Workflow Report

- Workflow Instance Report

This report gives inherent details about each of the workflow instances running or are completed. The columns of this report depicts duration, initiator, workflow model, payload used, status etc. This report is helpful for knowing the issues that might come up with any instance of the running workflow.

- User Report

This is the report to monitor the users that have registered. They can be authors or the external visitors. We can rightly check the demographic spread and the profiles. The columns of this report are user id, name, age, country, email id, interests, domain, language etc.

- Disk Usage Report

This is a good stored of the information about the data stored in the repository. Starting with the root we can dive into further branch.

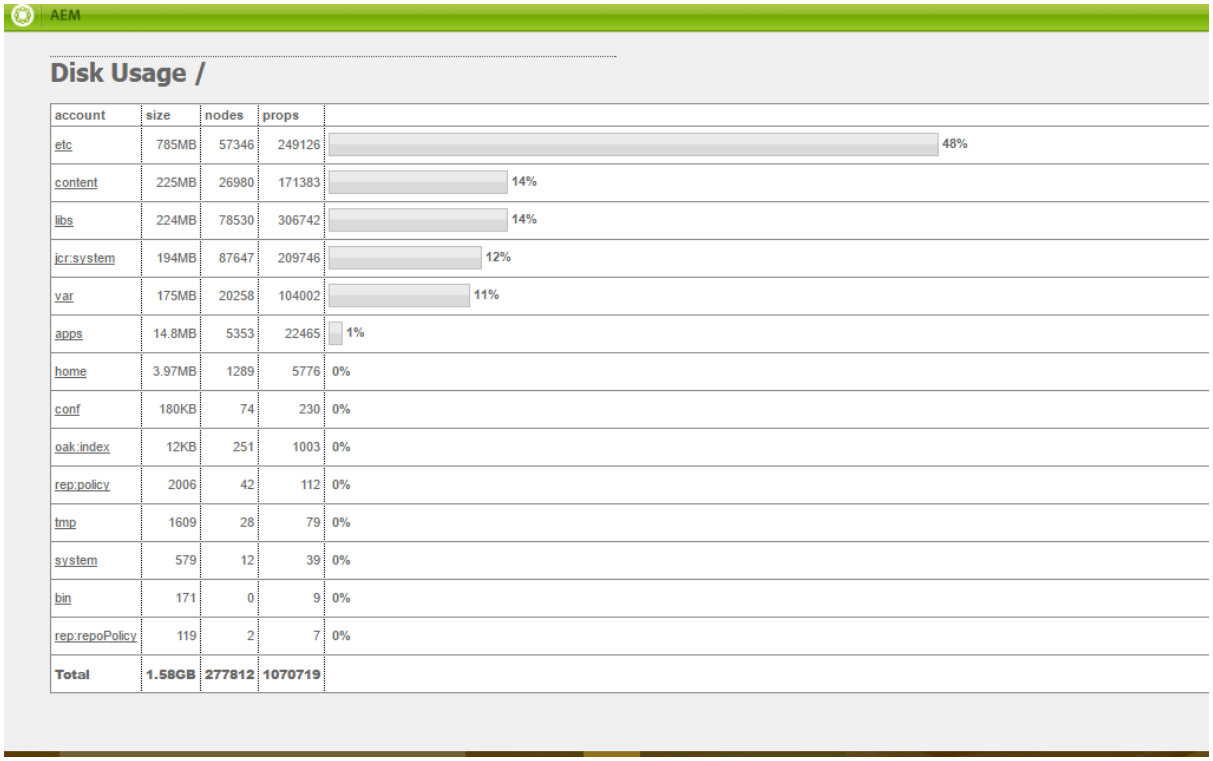


Fig. 4.27 Disk Check Report

- Health Check Report

This report fetches the data from the request.log and identifies the requests that are considered as most expensive.

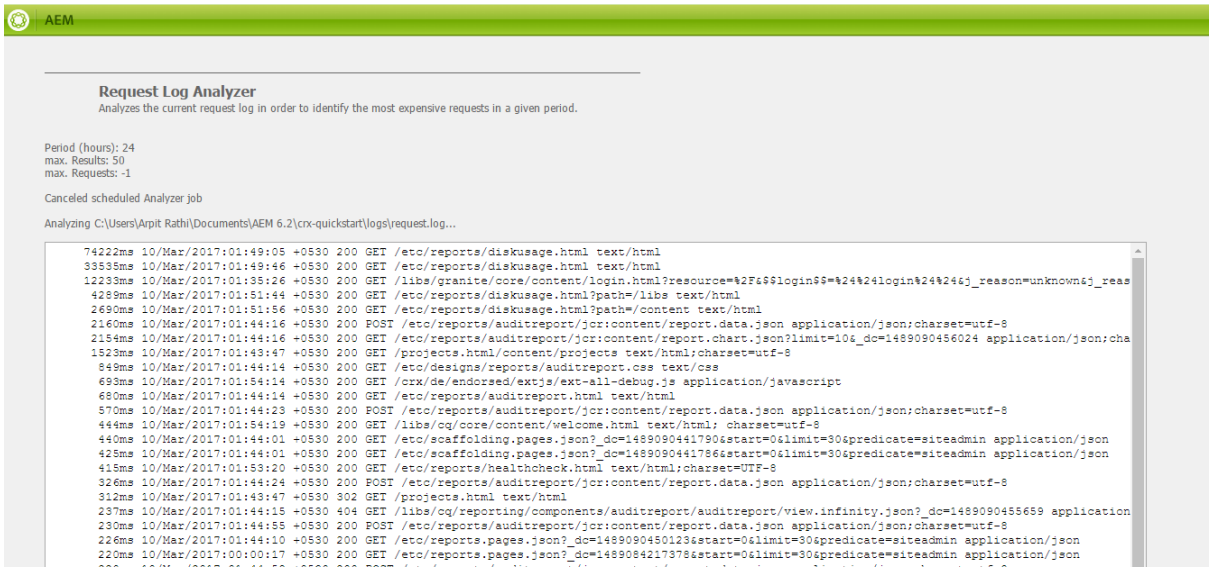


Fig. 4.28 Health Check Report

- Component Report

This is a report of how the components are used on the website. The information of author, component type, component path, page, modified by etc. are provided. The advantage of this report are to know where the components are used and how instances are spread. It can give a view of the page content development over a period of time. We can set the root path from where all the components should be shown in the report.

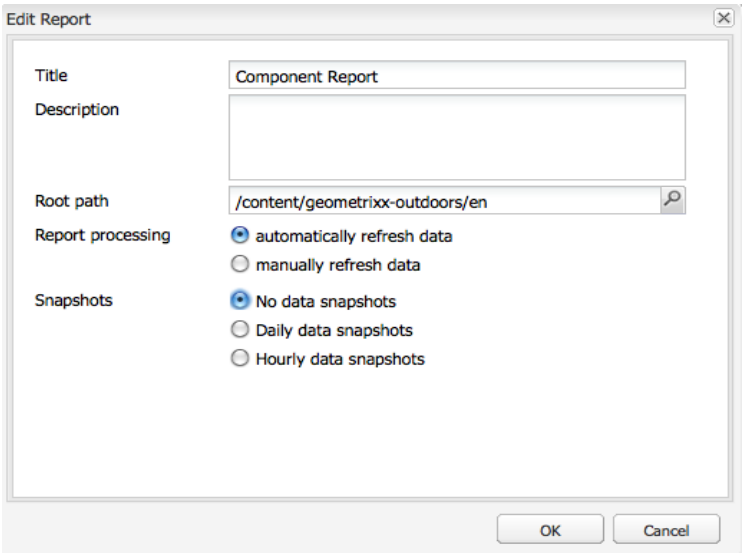


Fig. 4.29 Component report

4.6. AGILE METHODOLOGY (SCRUM)

Agile development combines various methodologies that include incremental and iterative approaches for software development. the agile methodology that is followed in our office is Scrum. each and every techniques of agile methodology incorporates the feature of iteration and a continuous response, interaction and feedback to refine the software in successive terms and deliver a comprehensive system in different build cycles. There is a continuous evolution of the software project and the entire system which includes parallel tasks of development, planning, testing, integration and other tasks. all the steps which are continuously a part of the cycle are adaptable to the changing needs or platforms and hence are quite lightweight to handle [13]. there is an empowerment of the entire team to utilize all the resources efficiently and making them coherently together in the decision making of the firm to deliver the software. this in turns make the productivity enhanced exponentially by motivating the entire team to work adaptively.

Advantages of Agile Methodology:

it is a lightweight framework which makes the entire team functional and hence focusing on the rapid development of the business deliverables. With the feature of being adaptive and working in small iterations, the overall risk that might be associated with the entire project or software release minimizes to the bare minimum and any changes or drawbacks can be incorporated in the future iterations or deliveries of the project. there is an acceleration in the delivery model which enhances the business value which is a result of iterative planning and continuous feedback. There is an added benefit of doing regression testing whenever there is a release of new functionalities or logics. User acceptance is an important feature in this methodology and hence interaction with users make sure that they get the product they aspired for. The benefit achieved is the alignment of the client's needs with the deliverables by easily adapting with the requirements. An increase in visibility, decrease in risk, parallel implementation of work and testing and a trait of adaptability, agile development process guarantees business value.

Scrum Methodology

Scrum is an umbrella term that wraps up variety of agile engineering practices and yet is quite lightweight and simple to implement providing guaranteed productivity and results in the business scenario. the methodology includes the concept of backlogging some features, bugs, or requirements that non-functional [17] as of now and then prioritize them accordingly into different sprints that are stretched across the working lifecycle of the project. The team meetings are tailored around prioritizing the work and diversifying the task across cross-functional teams. the final output of the meetings is the work divided into sprints to be able to deliver potentially acceptable increments of the project in the successive sprint duration. There is reprioritization of work after every sprint to decide the next set of deliverables. The four main pillars of Scrum methodology are:

- Division of project tasks into sprints
- Sprint planning and prioritization
- Daily Stand Up calls to discuss the status of the team's tasks.
- Retrospection on the deliverables as per the sprint goals.

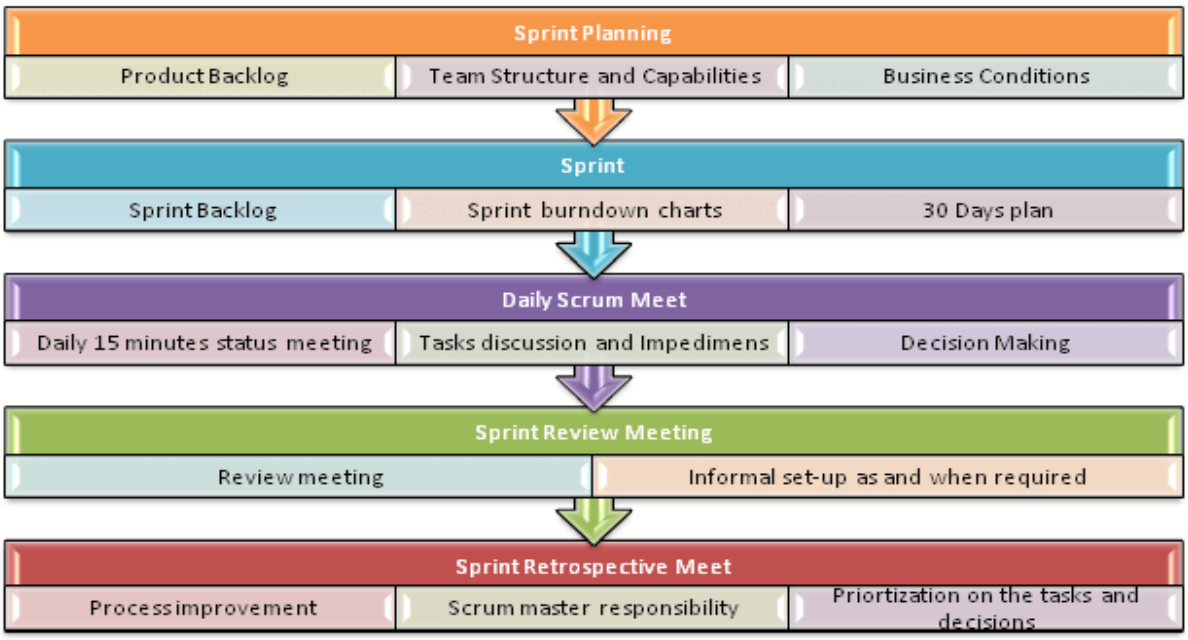


Fig. 4.30 Scrum Agile

Sprint planning is the first pillar of scrum methodology wherein prioritization of tasks take place to estimate them, assign them to team, adjust the scope of the present sprint with the help of version management, stroy points, backlog grooming in real time.
Sprint includes scoping of tasks to focus on the end product. There are workflows followed, release hub functionality, bug scrub etc.
Daily scrums are short meetings to check the work done, see the future plan of the sprint and manage the daily tasks in the working hours.
Scrum retrospection is the final reviewing of the sprint before the release including the important decisions to be taken such as release plans. There can be several reports and charts to see the sprint before closure.

JIRA is a commercial tool that is mainly used for project management, bug tracking, and issue tracking the the agile development of the project. It allows to create different user stories, distribute tasks to the team, plan sprints and prioritize backlog providing high level of visibility. it provides Scrum Board in order to do sprint planning and prioritization of tasks. it is composed of swimlanes to demarcate different activities going on during the sprint and plan accordingly. JIRA Dashboards provide an efficient way to capture the daily standup calls and provide a summary of the entire sprint work.

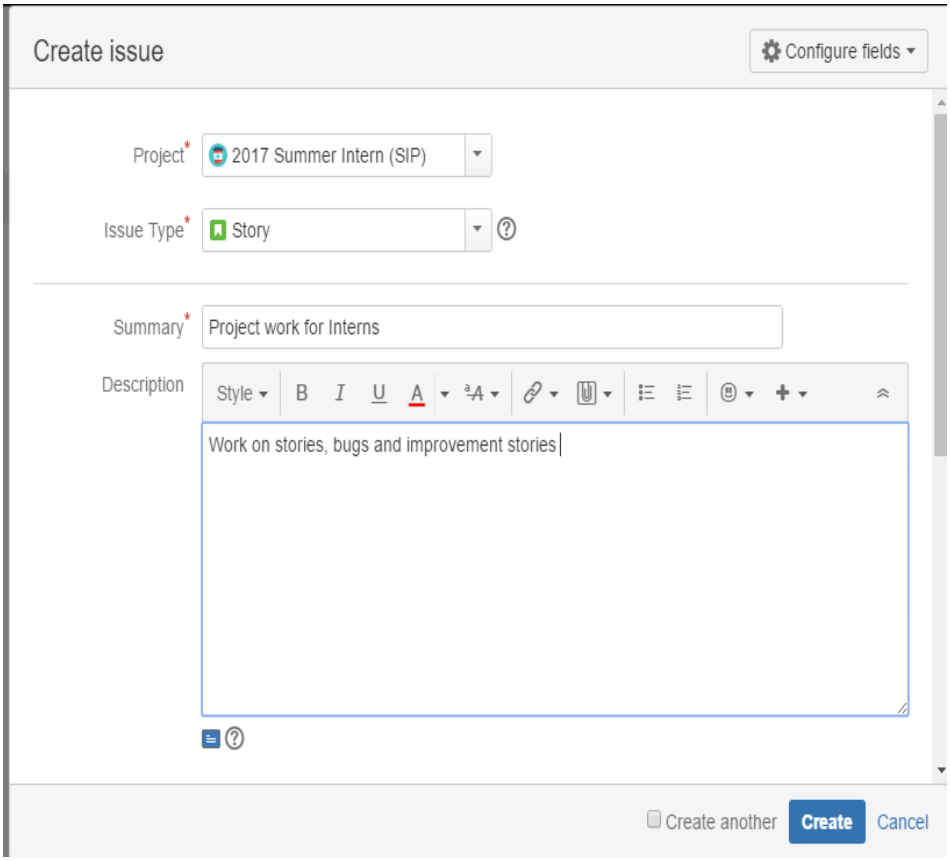


Fig. 4.31 Creating an issue in JIRA

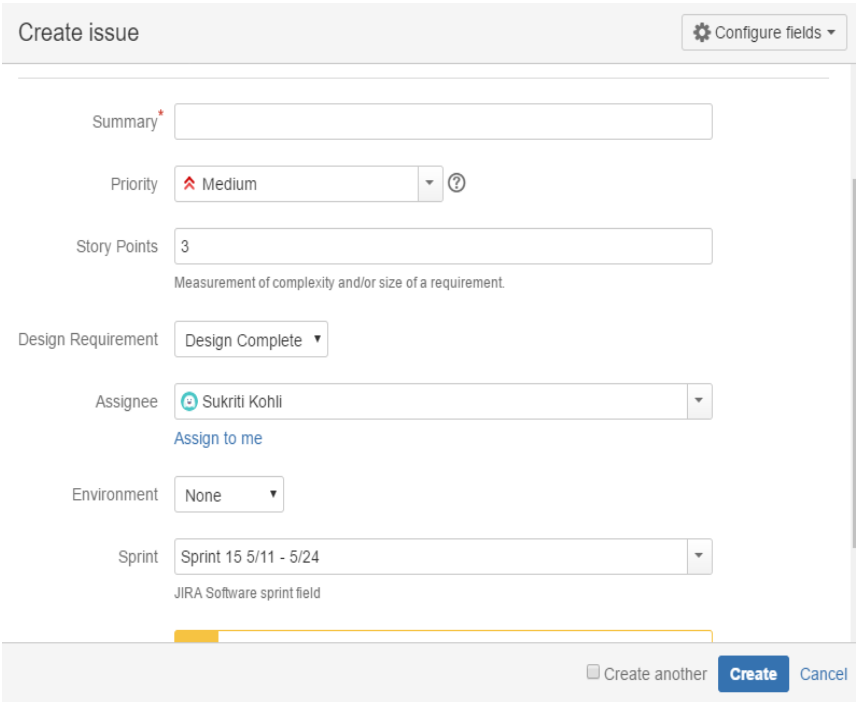


Fig. 4.32 Details of story in JIRA

4.7. RESEARCH PERSPECTIVE

The industrial training inherently included some research perspective where we were able to explore various dimensions and new features that required to be updated in the present system. Code was written in as a part of the research in order to rectify the present problem situation. The research was spread across the training phase where we researched on various aspects of the tools and had hands-on for the practical knowledge.

- Approach for pagination and load more

The earlier approach decided for implementation of lazy loading was formation of a blank page where the data is fetched from the java code and is feeded to the front end via this blank page. Ajax call was used to hit this blank page to get the results to the front end in Sightly. Research was done for a better approach to remove this blank page and modularize the results that were better for pagination approach. The POC resulted in finalization of the approach of making the ajax call hit a servlet which will run the query builder in the back end logic and bring the results required to the servlet. This servlet will return the data in JSON format which will be parsed in the front end and displayed in HTML format using mustache template. This approach did not include the concept of a blank page and servlet approach was quite clean in managing the results.

- DAM Update Workflow

The present functionality of the DAM Update workflow notified the changes made in the DAM assets either created, modified, deleted. According to this, whenever an asset was uploaded, some default renditions of the images were made for different breakpoints (screens). This technique of creation of the image renditions involved squeezing the entire image uniformly. This lead to loss of the quality of the image as squeezed image got blurred sometimes. Research was done extensively to find an alternate for generation of image rendition. The present requirement of the system was to have renditions for different breakpoints suiting up the portrait and landscape requirements of the devices too. Also, the image required to be cropped uniformly from all the sides rather than being squeezed or resized.

Research pointed towards updating the present DAM Update workflow to our customized workflow where we can give specified size of the rendition as well as give an algorithm that would help in cropping of the image uniformly from all the sides.

To rectify this situation, a step was added in the present workflow and cropping algorithm was applied on the original image to save them into different required sizes and generate the respective renditions for the same. The algorithm took the size of the image as the inputs and calculated the required height and the width of the image. Then dividing this required height and width by two we first cropped the image from left and top using Image class in Java, rotated the image by 180 degrees, applied the same operation and re rotated it to get the original image. These renditions are then stored in the specified directory.

The entire algorithm was a part of POC and it's implementation required various trials in the methodology.

- Multifield Touch UI functionalities

Multifield component in Touch UI does not support the complete list of functionalities that might be required by the developers due to bugs in the Coral approach. Multifield component did not support checkboxes and drop down features. The values were not retained and were not accessible using the properties global object. The approach which was adopted for rectification of this problem required research in various libraries and APIs. It required research in different packages and codes of the public projects. Custom js code was written for rectifying the issue. In the js code, classes were added to the multifield component and js was written which enabled storing of values and accessing of these values from JCR.

- Coral UI bug scrub

Coral UI is used to give look and feel to the Touch UI available in AEM. There were some issues with the dialogs and their css present in the Coral UI library. Research in this domain included finding the new features in the future version of AEM (earlier in beta phase) and comparing the code for noting the differences and features required for the present instance. Hence, bug scrubbing was done for Coral UI 2 and issues were rectified by researching in Coral UI 3.

CHAPTER 5

PRACTICAL IMPLEMENTATION

5.1. AEM BASIC IMPLEMENTATION

The following figure shows the start of the jar file of AEM

Fig. 5.1 AEM JAR startup

The following figure shows the Touch UI screen for AEM where we can easily navigate among different features provided by AEM for integration.

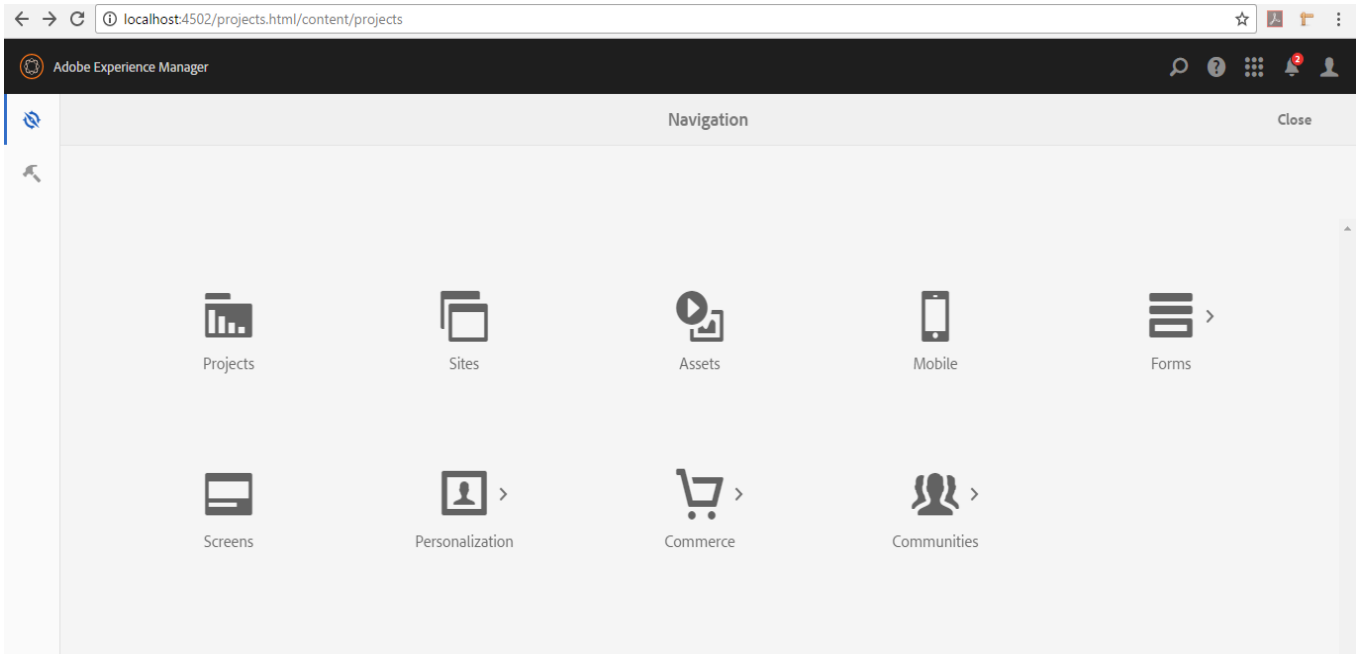


Fig. 5.2 Touch UI console for AEM

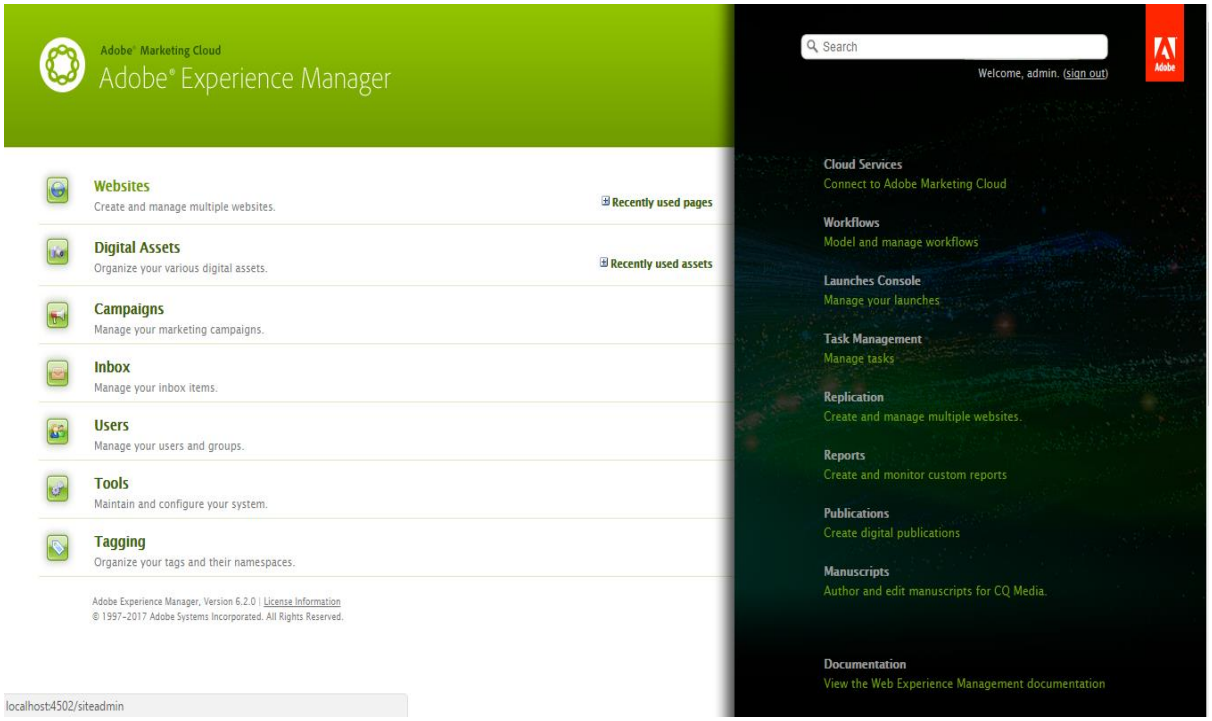


Fig. 5.3 AEM Welcome Screen

This is the welcome screen of AEM which provides an easy navigation to various functionalities that can be used. It can be used to access the siteadmin (the website and the webpages of the project), damadmin (digital asset storing repository where images, files, videos, files etc. are stored for the use of the project), campaigns (integration feature with Adobe Campaign), Inbox (the mail inbox of the logged in user to get requests), useradmin (user profiles and their access rights to the project), tools and tagging features. The right hand side console has various unique and diversified features including workflows, replication, publications etc. Replication is the process of activating the pages from author to publish environment and check them in real time. The agenda of replication tab is to check whether the replication agents are active or not.

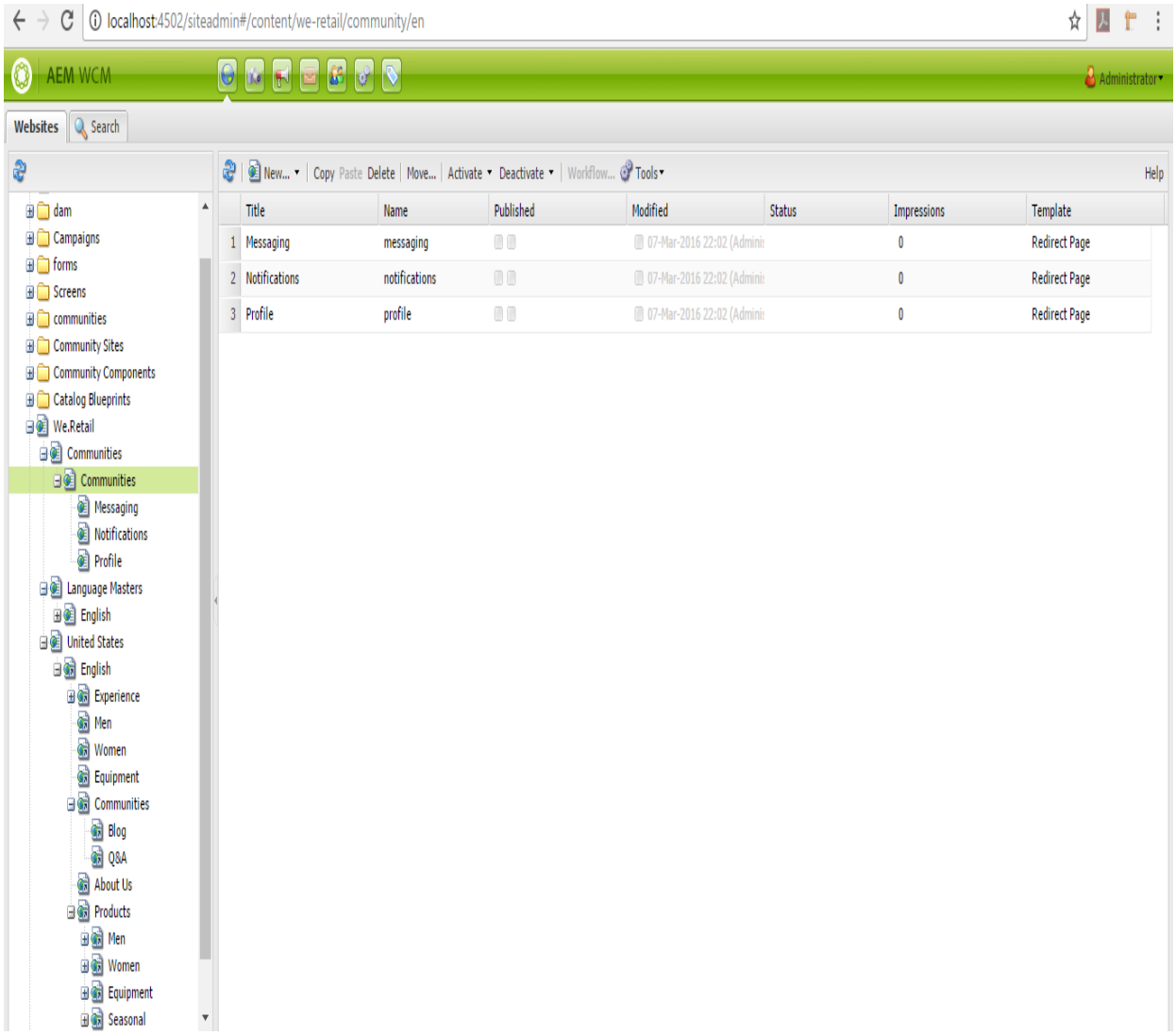


Fig. 5.4 Siteadmin for different web pages

This is a generic view of the siteadmin where we can see the hierarchy of the pages that are made in the project. This is a sample project provided.

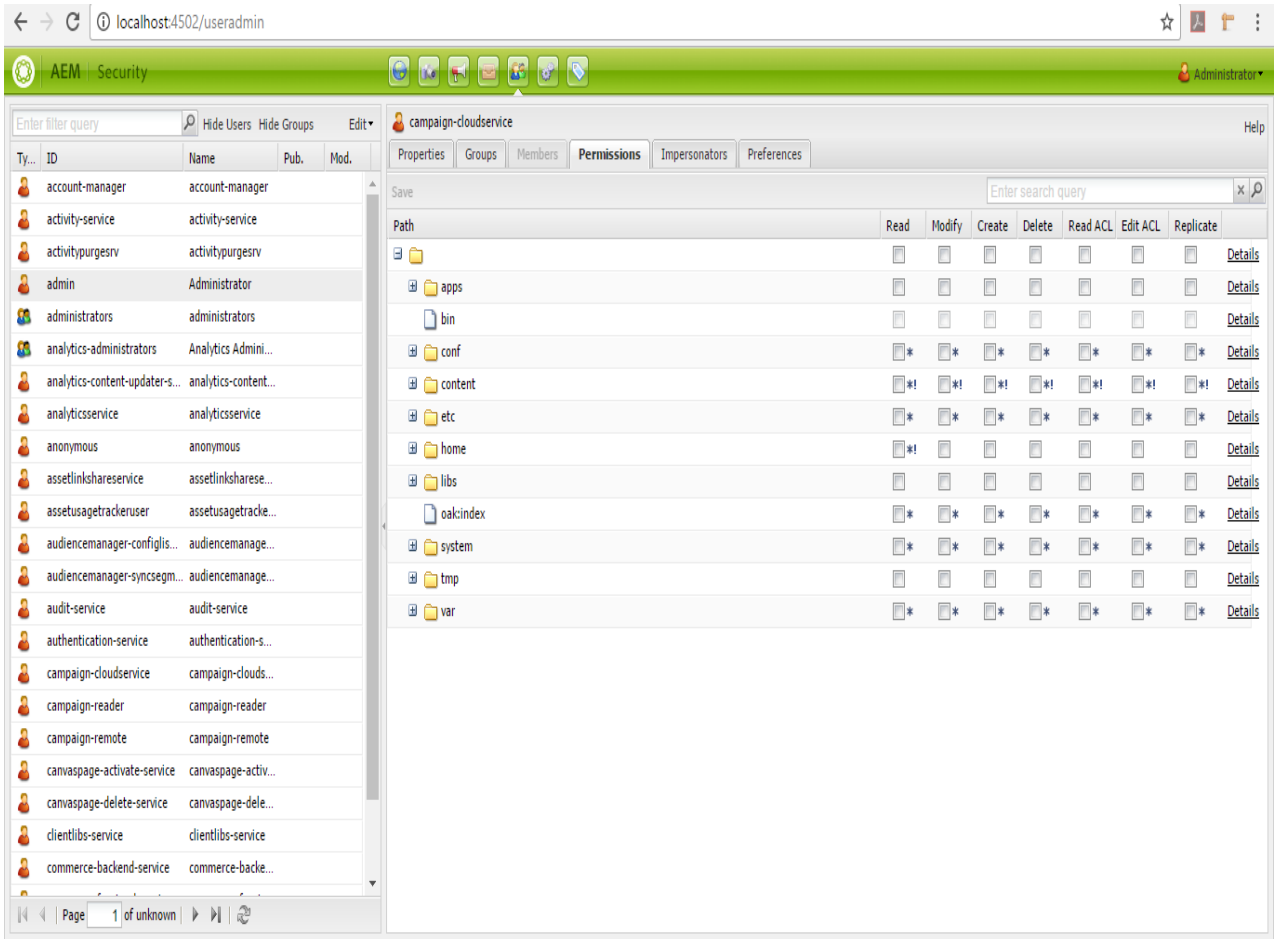


Fig 5.5 User admin of AEM

This is the useradmin console where the users and the groups can be defined.

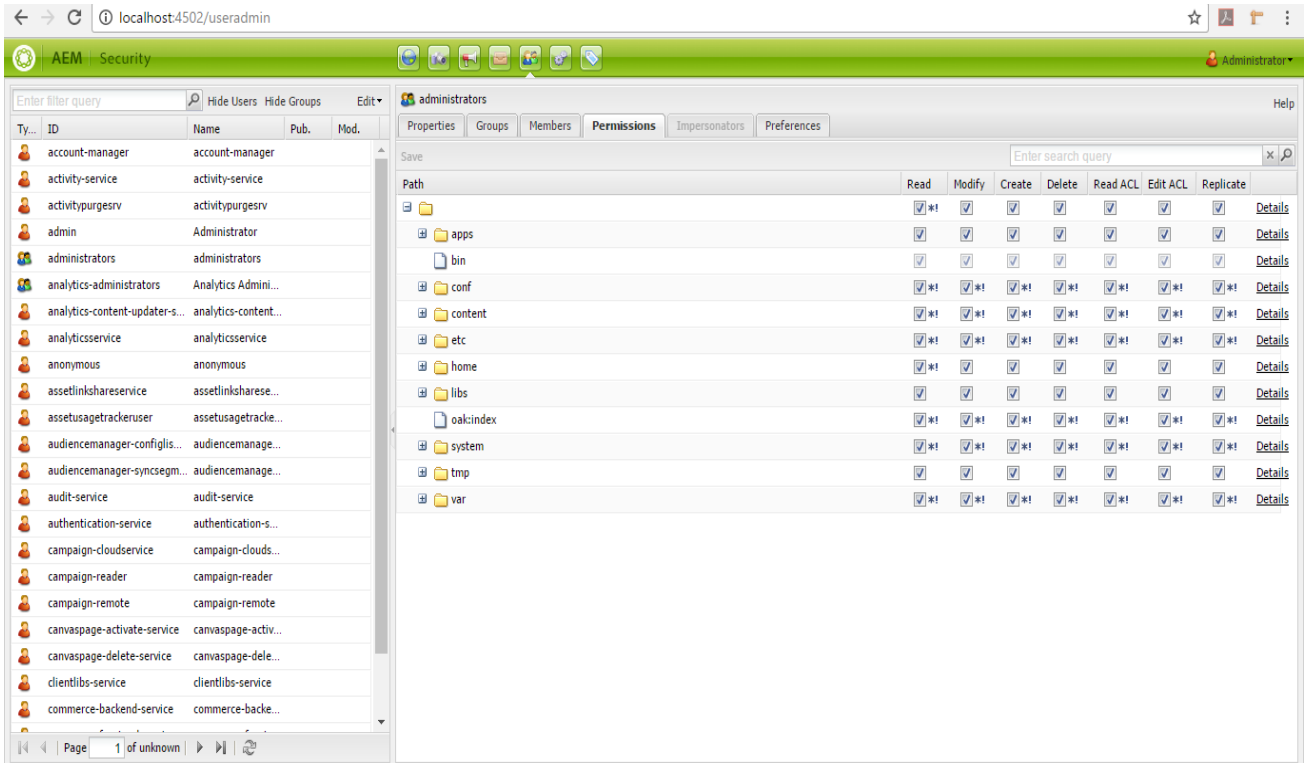


Fig 5.6 Access Rights for Users

This figure depicts the permissions given to the users for all the directories and all the types of permissions.

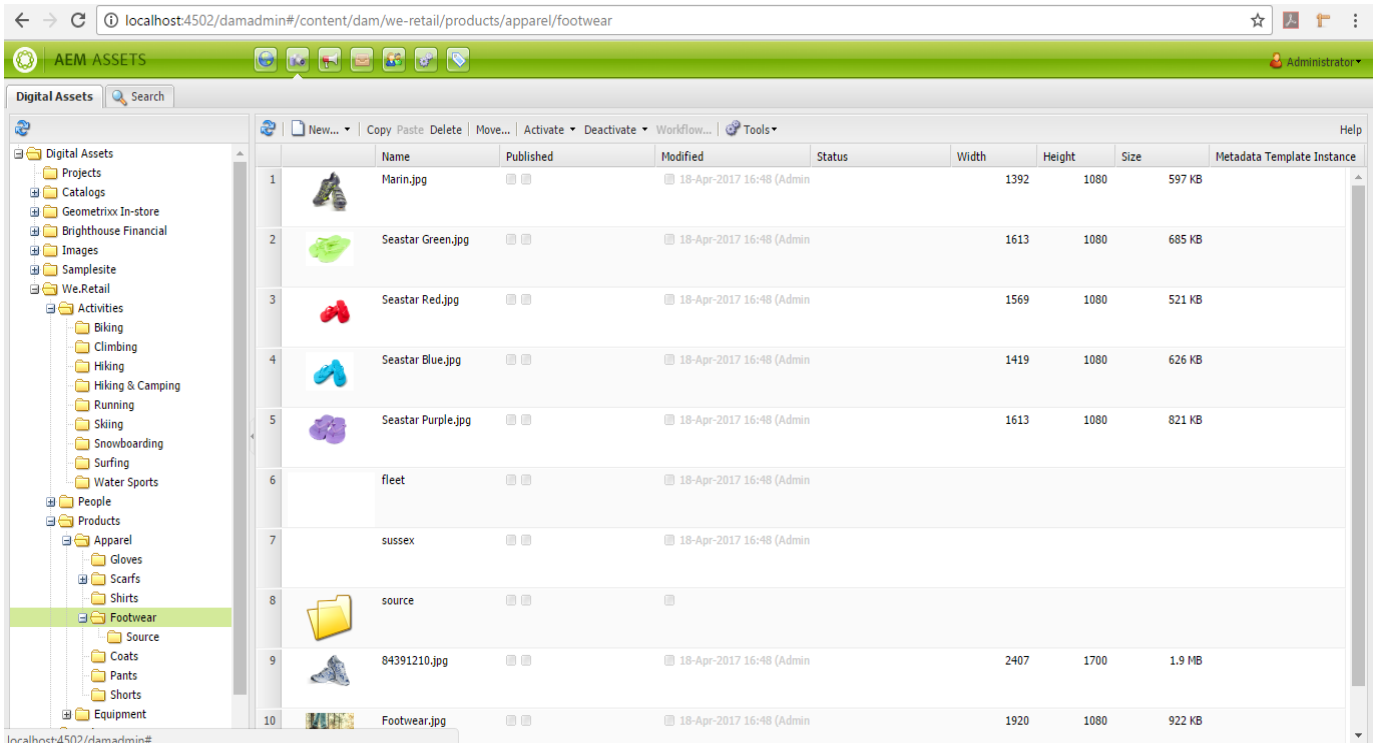


Fig. 5.7 DAM admin for AEM

This is snapshot for DAM admin where all the assets of the project are stored may it be images, videos, files, folders etc.

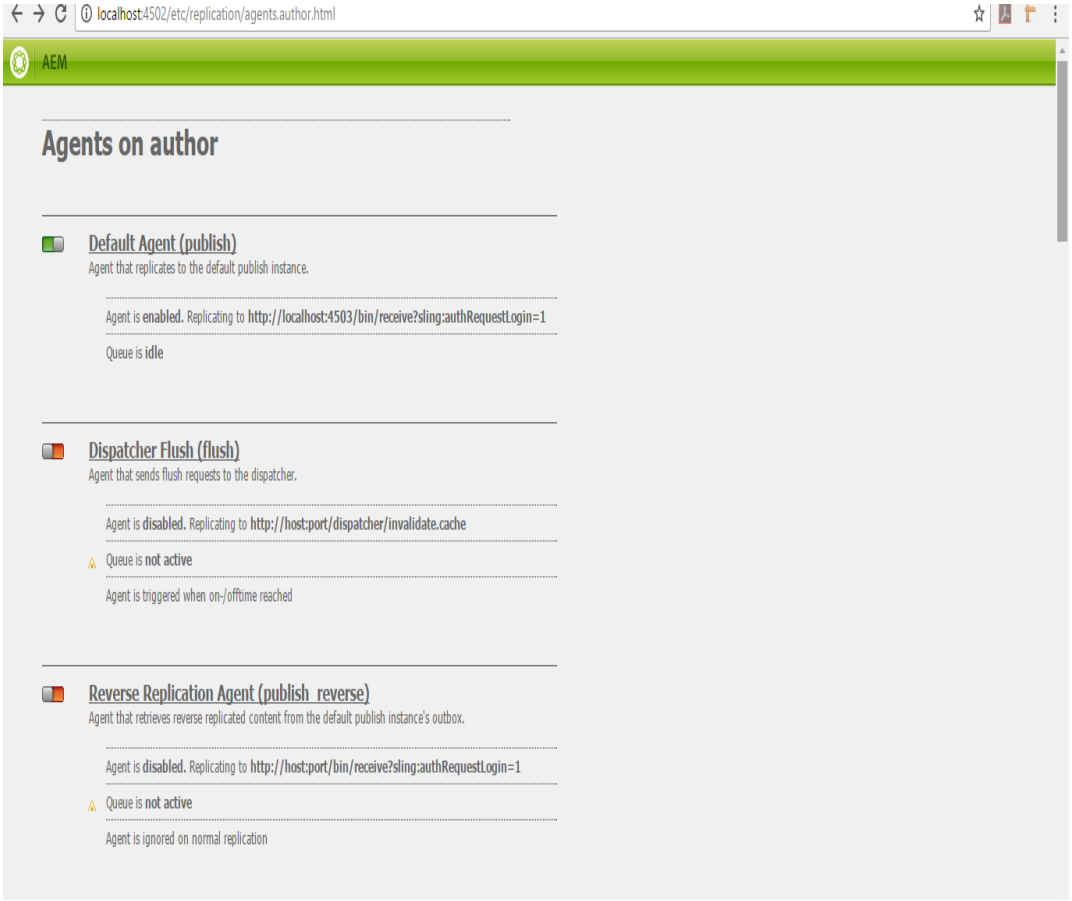


Fig 5.8 Replication Agents for AEM

This is the console for the task of replication. This screen depicts the agents configured on author such as replication agent and reverse replication agent. Each of the agent can be edited as per the project requirements. There are separate queues for the agents that pile up the tasks to be completed.

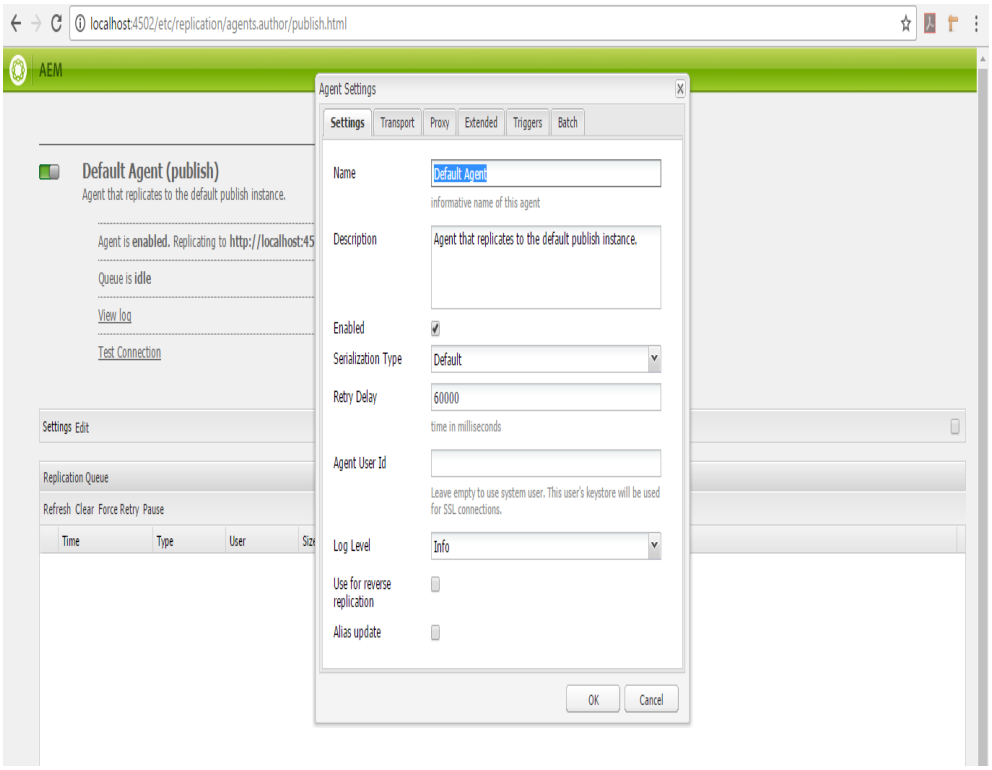


Fig 5.9 Configuration for Replication Agent

Templates in AEM are created with these properties having label and title and linked to a page component by resourceType property that will help in rendering of the pages that will be created with these templates when chosen. The rank of the template decides the place where it will be seen in the options during creation of the webpages in the project. Allowed Paths property enables a template to be available to create the pages from the specified path.

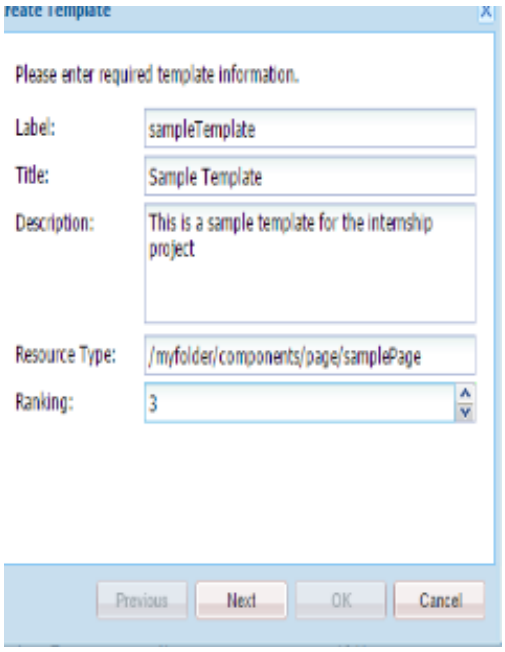


Fig 5.10 Template Creation in AEM

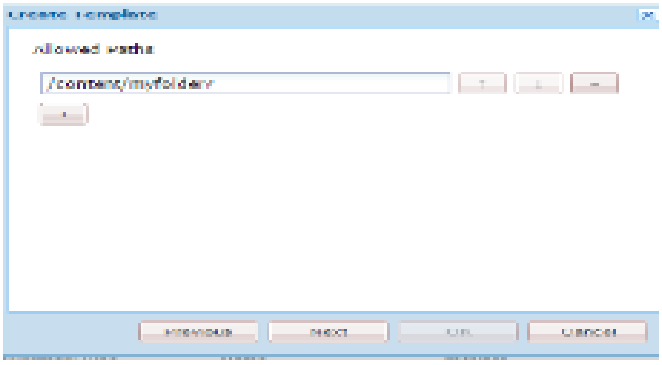


Fig 5.11 Allowed Paths for Template

5.2. PROJECT WORK

The ongoing project serves one of the biggest clients in United States of America for development of their website which is based over AEM. The first release of the project was done on 6th March 2017 followed by the second release on 5th May 2017. The third release of the project is planned to be on 30th May 2017. The entire website is composed of different reusable components made by the team via AEM for handling the content management system of the client. The components can be dragged and dropped by the author of the website which makes the website ready to be published. There are basically two domains or environments basically: Author does the authoring of the website which is the final content that needs to be published. This result goes to first publisher who verifies the details and rectifies the correct use of the components and the content entered. It then goes to the second publisher for the final review. Finally, the build is given on the production server which makes the website to go live. Hence, the hierarchy of the servers is local, Develop, Internal Quality Assistance, Staging or External Authoring, UAT or User Acceptance Team and finally Production. Components in AEM are supported by HTML, CSS (for the look and feel of the website as decided by the designers hired by the clients and as given to our team in the form of style guides and zeplin designs), Javascript (To handle the client side functionality of some of the features of the website), Sightly (HTML Templating Language to connect the backend Java logic from the front end HTML designed for the components), Java (Sling Models, OSGI Configurations, OSGI services, Servlets, Models, Workflows for handling the back end logic that runs behind the components. It is especially in the case when something is required by Logic and should be automated than authored.). Components are used to provide an interface to authors in the form of dialogs. These dialogs are designed in CRX in the form of nodes with a specified structure that needs to be followed to give appearance to the dialogs. The content is filled in by the authors that is published on the final website. CSS provides the look and feel to the web pages as approved by the clients. Each of the CSS specifications are provided by the client itself. Javascript is used to handle the events that need to be captured on the webpages for the required actions to be performed. Sightly is used to handle HTML DOM for the webpages in which the code for the websites are embedded. Sling models handle the java code that executes behind the web pages such as services, osgi services, sling models, utilities, workflows etc.

IntelliJ Idea is used for Java coding for the website. CRXDE Lite is the implementation of the AEM components. However, Eclipse can also be used for the backend coding logic.

To start with the project, the primary task is to create some templates, pages and basic structure for the website. For creating a page, it is required to create a template and a page component. A template is the base and the skeleton for a specific type of page. When creating a page in the Websites tab the user has to select a template which is either provided by default or created by the developers either as nodes or using Template Editor functionality provided by AEM.. The new page is then created by copying this template and contains the elements that are defined in the template. A template is a hierarchy of nodes formed that has the same structure as decided and as the page to be created, but without actual content. It defines the bases of the page component used to render the page on the website and the default content (top-level content). The content defines the way it is rendered as AEM is considered to be content-centric. It has cq:template as primaryType which needs to be specified when creating the template and allowed paths include path of content folder where the entire content of the website is stored in JCR. It has sling:resourceType as the name of the page component using it. These are the bare minimum properties that are required for a template to be created. A page is considered to be an 'instance' of a template. It has resourceType as granite/ui/foundation/components/parsys and resourceSuperType as foundation/components/page. This is required to give the basic look of a page.

The paragraph system is a key part of a website and is frequently used for page authoring as it manages a list of paragraphs provided to the authors to use components. It is used to hold, position and structure the individual developed components that can hold the actual content authored. Within the AEM a component is used to render the actual content of a resource which is to be displayed on the pages. The definition of a component is basically: the code used to render the final content. a developed dialog for the user input (author input) and the configurations stored of the resulting content in JCR. Hierarchy of a developed dialog of a component goes as: Dialog: (content) container: (layout) tabs: (items) section: (layout) fixedcolumns: (column) container. this hierarchy needs to be followed minimum so as to give the structure to the dialog. There are many widgets (used to make dialog fields) that are provided by form (API) which can be used in the component where some of the basic being textfield, radiobutton, checkbox, pathfield, fileUpload, numberselector, textarea etc. Each of

these widgets have properties by the medium of which they can be manipulated for use such as mandatory, default value. cq:editConfig (cq:EditConfig) - This node defines the edit properties given to the component. It enables the component to behave in the manner defined by the Edit Config Listeners and make the components able to appear in the Components browser or Sidekick.

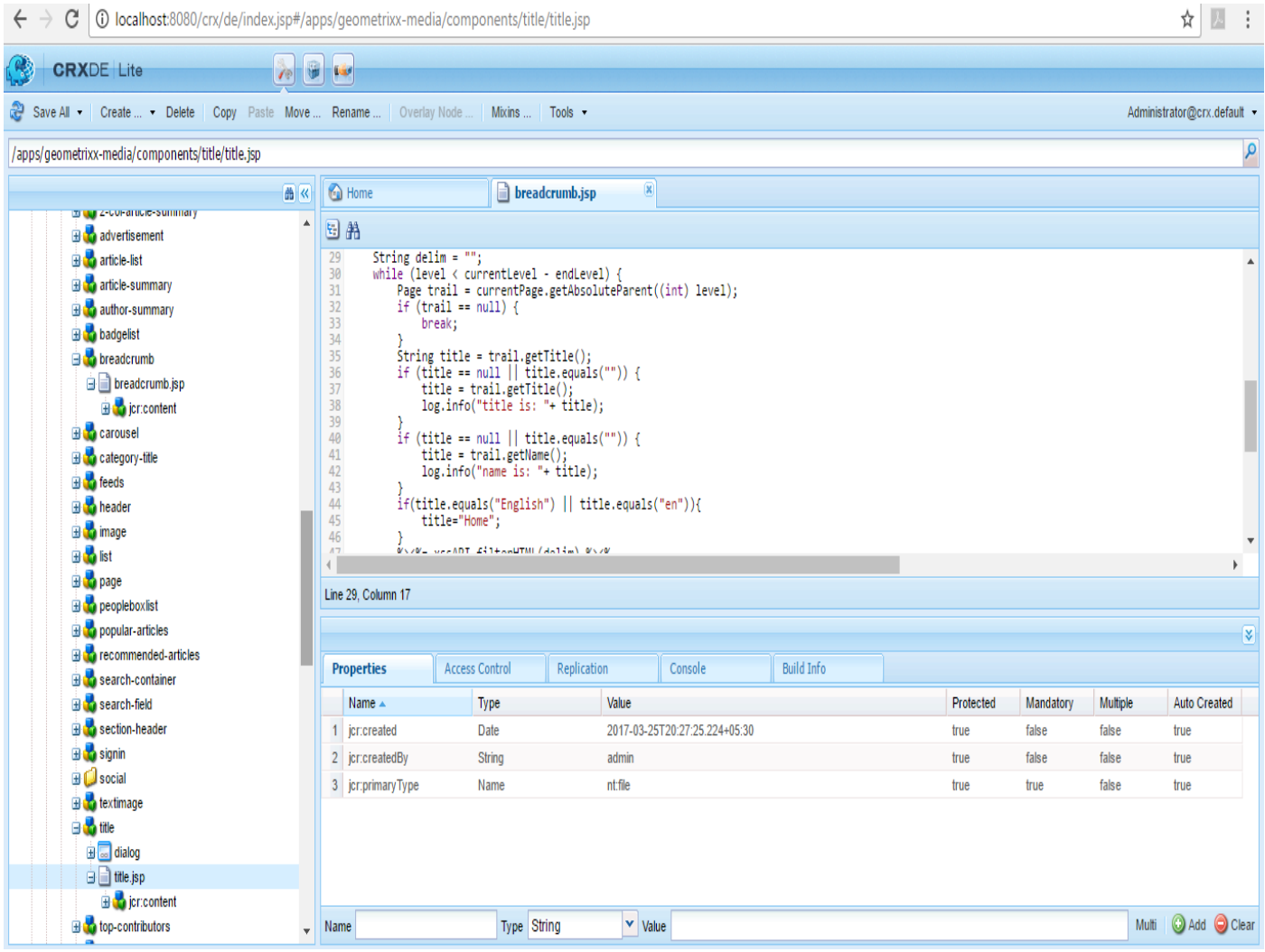


Fig 5.12: CRX DE Lite snapshot

The above snapshot shows components on the left panel, the html code for a particular component in the workspace and properties of the selected node.

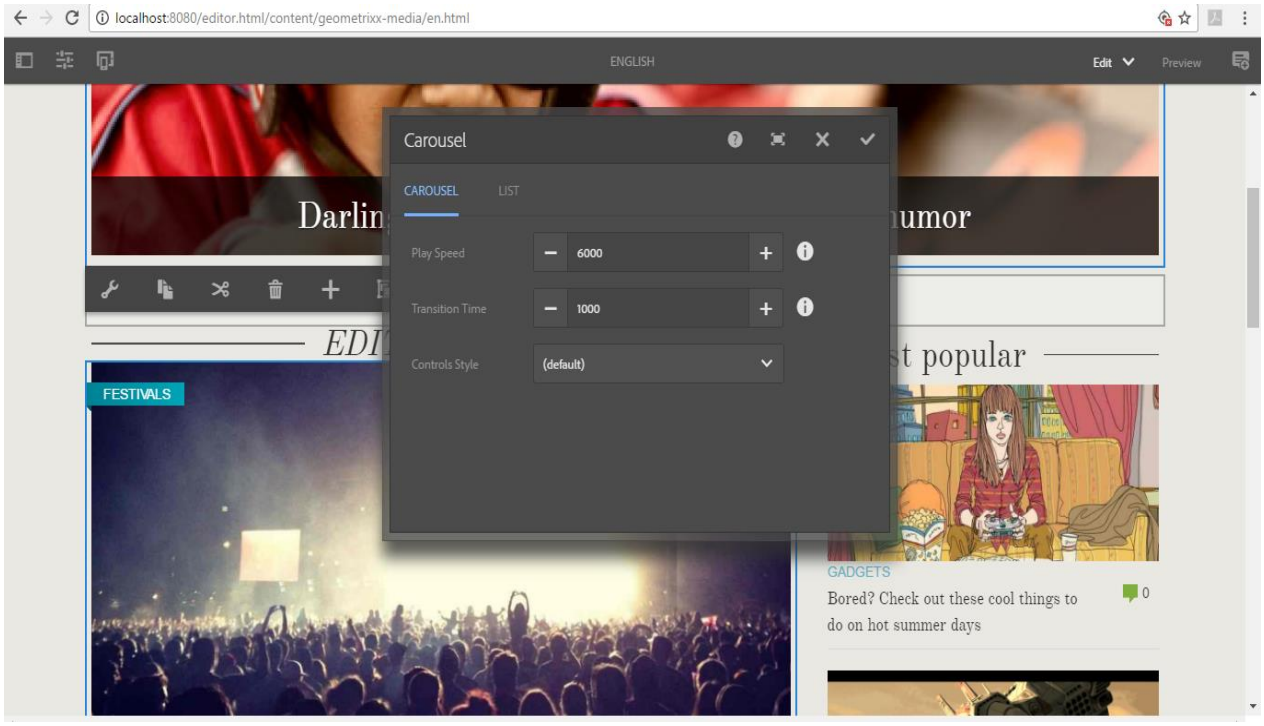


Fig 5.13 Dialog snapshot for Carousel

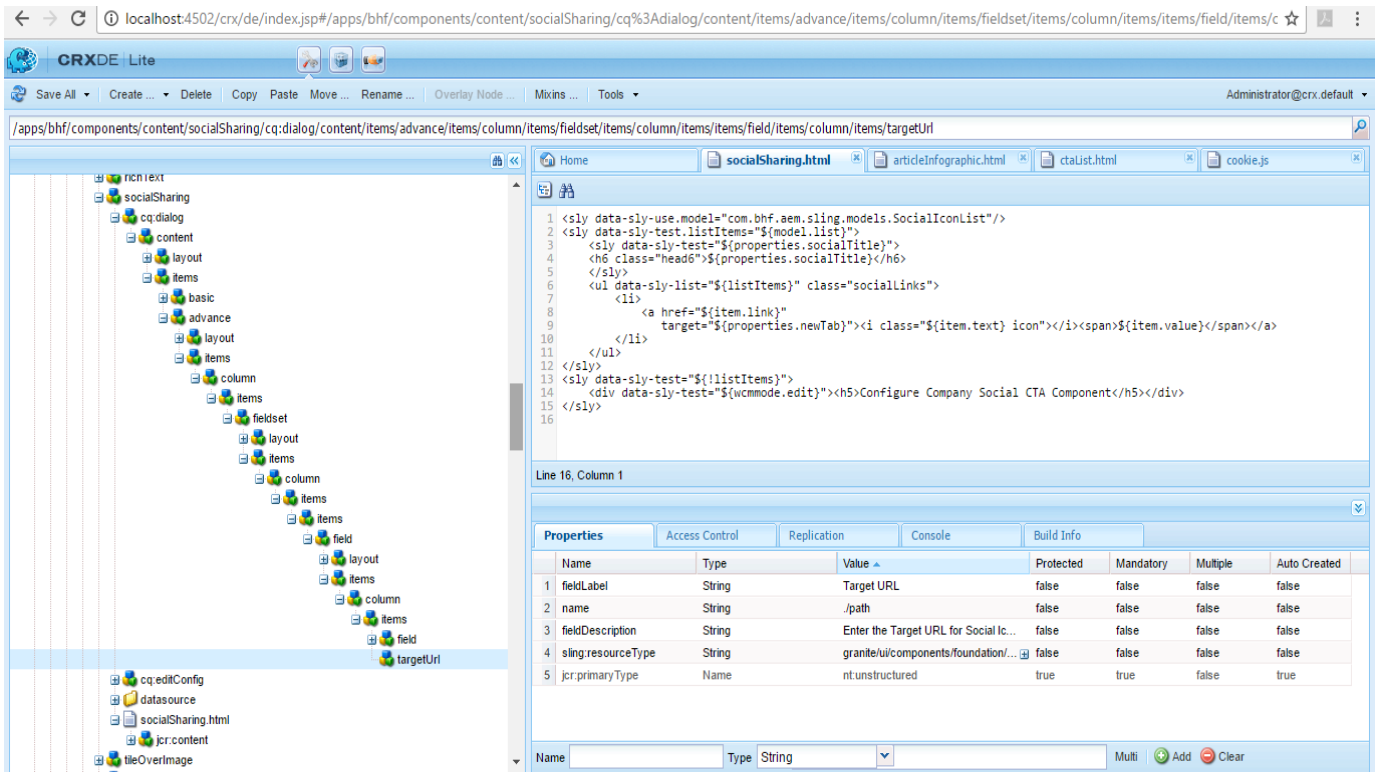


Fig 5.14 Snapshot of the CTA List component (socialSharing)

This is an example of one of the component in the project which is used to make the component where there are links provided to go on other websites. It consists of a container having a dialog which has different widgets dropped on it. These widgets are provided by Touch UI granite API to give functionalities to the users. There is a textfield and patthbrowser are used in this component in order to provide links to external sites. The dialogs can be made by using variety of these widgets to store use information to be rendered on the page. The style of the dialog depends upon the requirement of the clients which is defined in the stories created by them.

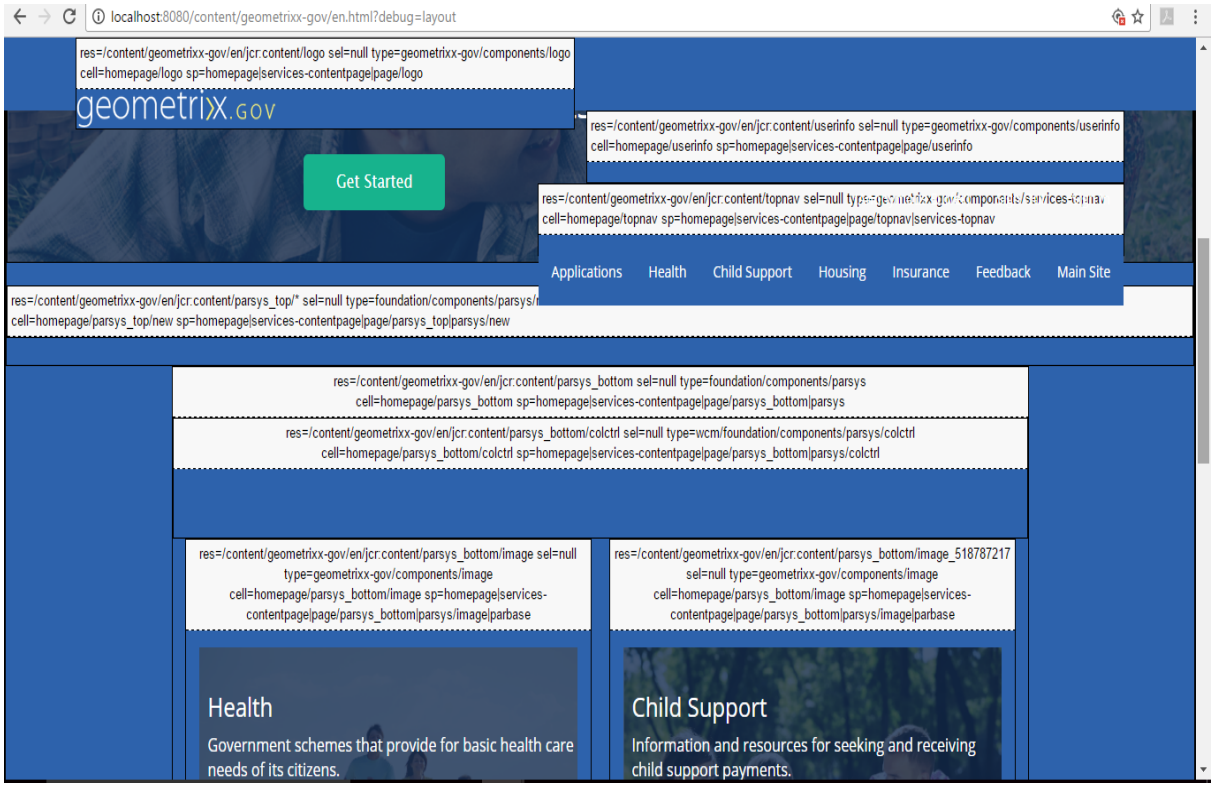


Fig 5.15 Snapshot of the final website showing different components used in the page

The following is a sample of code integrating logic stored in JCR with HTML front end.

```
<sly data-sly-test.fileReference="${properties.fileReference}"/>
<sly data-sly-test.captions="${properties.captions}"/>
<sly data-sly-test.attributions="${properties.attributions}"/>
<sly data-sly-test.condition="${fileReference || captions || attributions}">
  <div class="row">
    <div class="small-12 medium-12 large-12 columns fullwidthImageMobile ">
      <img src="${fileReference @context= 'html'}"  alt="${properties.altText}"/>
      <div class="body-caption-sm paddingTop5">${properties.captions}<br>
        <span>${properties.attributions}</span>
      </div>
    </div>
  </div>
</sly>
<sly data-sly-test="${wcmmode.edit && !fileReference && !captions && !attributions}">
  <div class="cq-placeholder cq-marker-start" data-emptytext="Image With Caption"></div>
</sly>
```

This code snippet is a very simple Sightly example where a component of Image with Caption is rendered with the help of HTML. Sightly language is used to test the existence of the properties in the component and is used to fetch them to populate in the HTML DOM structure. fileReference property stores the location of the image from the DAM. Other properties are saved with their names and will be fetched in Sightly using properties object.

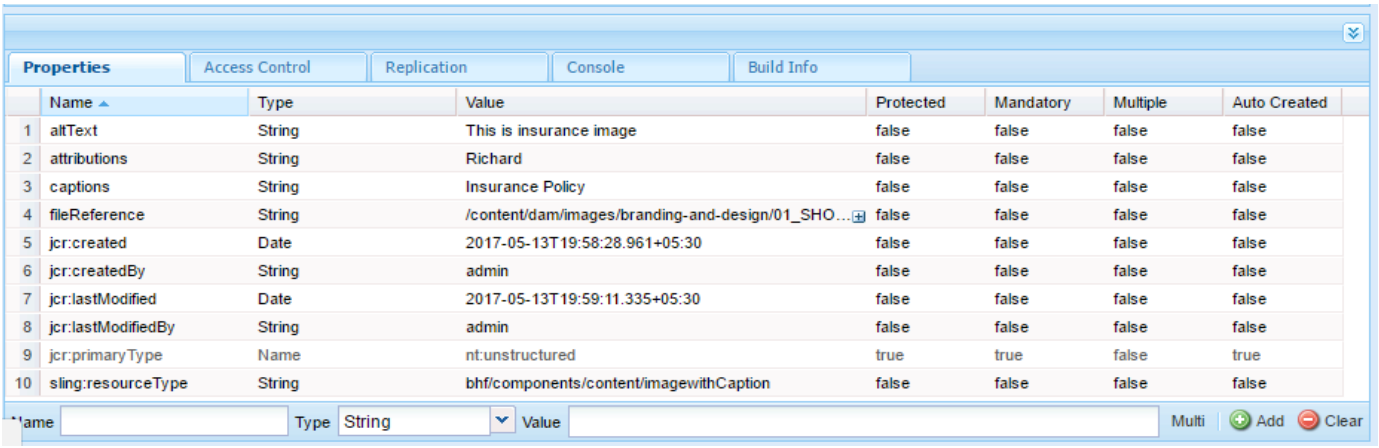


Fig 5.16 Properties stored in JCR with names and values

Some of the components developed during the training are, top navigation, custom multifield modifications, login component, logo component, advisor interstitial, generic interstitial, generic hero component, authoring improvements, latest feed and load more functionality etc,

Top navigation is developed with the help of Navigation API using Page API, PageFilter API etc. Various Java APIs help a developer to use the functions and complete the desired activity by modulating the functions accordingly. Top navigation logic uses these APIs and is then integrated with the front end design of the top navigation. Advisor interstitial uses client side cookies written in Javascript. This functionality is to notify the users when they are trying to leave the present site and redirecting to a section of website which is for some special users only. Client side logic is handled with the JavaScript logic. During the shadow project I integrated the HTML DOM with the logo component Sightly code to implement its functionality. In this task, the role was to use the Logo component and front end HTML design of the component and integrate them with the logics of Sightly providing different conditions and scenarios. There was development of some generic components that will make the author use them at variety of places with different look and feel as per the requirements. The task also included using of HTML classes and modifying the CSS accordingly to be able to get an idea of front end development too. Latest feed is a component to fetch the articles and posts according to their published date so my task was to introduce pagination concept in the existing component which was done using, Java logic, JavaScript code and Front End templating.

```
$.ajax({
  url: "/latestfeed/latestFeedServlet.createLatestFeed/",
  type:"get",
  data:{"path":path},
  success: function(data){
  resultSetFeed = data;
  var parsed_response = JSON.parse(resultSetFeed);
});
```

This code snippet is a generic Ajax call that is used to call a servlet which will do some required tasks. The role of ajax call is to send an asynchronous request to the servlet without reloading the page. Hence, the pagination task required the use of ajax call in js code to call the servlet that would run the backend logic and provide the result in JSON format. This JSON is then parsed to get the results in the front end which is fed to mustache template provided by HTML to display the desired results.

There are many other authoring improvement tasks that are taken up in the current sprint of the project and will be continued till the next releases.

Sometimes, according to AEM 6.2 versions, the widgets provided do not have complete list of functionalities that might be required by the developers to fulfill the tasks. To overcome this issue, we need to write our custom JavaScript classes that will be specific to authoring of the dialogs so that we can get the authors the flexibility, speciality and details required. For example, custom multifield component that can be used in providing features of adding different countries having different states does not limit the number of fields that can be added. A particular use case required the maximum limit ot be set and hence a custom JS was written for the same.

```
$(document).on("dialog-ready", function () {
$(".js-coral-Multifield-add").click(function() {
  var field = $(this).parent();
  var size = field.attr("data-maxlinksallowed");
  if (size) {
      var ui = $(window).adaptTo("foundation-ui");
      var totalLinkCount = $(this).prev('ol').children('li').length;
      if (totalLinkCount >= size) {
          ui.alert("Warning", "Maximum " + size + " posts are supported!",
"notice");
          return false;
      }
  }
});
});
```

This code snippet is used to make maxlinksallowed property for custom multifield to be used as per the use case.

Similarly, checkboxes does not work and persist values in case of custom multifield but as per a use case of giving a checkbox option to the author to open the link in new tab or not, a custom js was appended to be able to execute this functionality.

```
function isCheckbox($field){
  return !_.isEmpty($field) && ($field.prop("type") === "checkbox");
}

function setCheckBox($field, value){
  $field.prop( "checked", $field.attr("value") == value);
}

function setWidgetValue($field, value){
```

```
  if(_.isEmpty($field)){
      return;
  }

  if(isSelectOne($field)){
      setSelectOne($field, value);
  }else if(isCheckbox($field)){
      setCheckBox($field, value);
  }else{
      $field.val(value);
  }
}
```

This code snippet was added to the by default given implementation of the multifield component.

Other tasks included research on the topics given in Adobe Immerse which was an educational initiative to share knowledge regarding AEM 6.2 and Touch UI. So, AEM forms, ClientLibs, Workflows, AEM Communities, Sightly etc. were learned as a part of it.

CHAPTER 6

PROJECT LEGACY

6.1. RISK ANALYSIS

There are many things that needs to be kept in mind in order to code in AEM as it requires special expertise in many fields. There are many risks associated with the implementation.

For AEM itself, in the implemented environment there are various OOB (Out Of the Box) components available for the use of the authors that are yet a little buggy (including minor defects and not covering entire use cases) and not fully ready to go even without some minor tweaks for the real project use. Since, this software (earlier made and acquired by Day CQ) was acquired not very late by Adobe, therefore a lot of Adobe integrated features and solutions are not fully ready-to-use or baked yet. This makes the integrations cumbersome and introduce various defects that are difficult to catch and fix. This AEM tool is not much suitable for small business or start-ups making their websites due to the cost incurred in purchase of the software and high level of developer expertise. Debugger in AEM does not function every time as required because of some untracked errors. Also, the IntelliVault tool which is used to pull or push the code from or to CRXDLite to IntelliJ/ Eclipse does not work every time as per the desired and the expected behaviour due to unknown reasons.

There are certain functionalities provided by Touch UI which are not fully functional and Javascript needs to be written to make them compatible with the custom made components. Every new version of AEM covers the small bugs that are incurred in the present version supporting Touch UI. Also, special form widgets used do not extend every property that might be required during the time of development hence, extra effort needs to be put in making the code generic and compatible for all the components developed in the future.

While working with ClientLibs in AEM (for executing the JavaScript code) there might be cases when JavaScript code breaks on the console due to some properties which might not be supported on the browsers. There might be scenarios where cross-site scripting may creep in and break the code. So, these risks are associated with the JavaScript implementation which needs to be taken care of. The issues can be browser specific, run modes specific, environment specific, server specific etc.

There are many risks associated while working on git as many conflicts arise while taking pull before committing a code. There conflicts might sometime lead to loss of code on a machine when stashing is not done for the current working branch. Hence, working with git can introduce risks on the work that we are trying to send to the remote repository from our local repository.

6.2. FUTURE SCOPE

The present project has been released and the present work is going around modifying the functionalities to make them more generic and flexible. These will be further releases of the project which will be logically divided into sprints and will modify the existing components. The project now will include working on improvement stories to modify the website and to make the components more generic and authorable in order to give more freedom to the authors to display relevant content on the website. For implementing this, we need to modify the dialog structure or the existing components or the look and feel of the components as per the updated designs provided by the client. This may also include changing the desired behaviour of the existing components by doing some modifications in the backend logic implemented in the specific components. Also, many new pages shall be authored requiring some special components that are template-specific to extend the pages and content of the website.

The work of the developers will include making some new components to fulfil the demands of the client. Also, the work will focus around modifying the existing components and including enhanced functionalities via Javascript coding. There will be use of servlets and ajax calls for the functionality that is specific to some requirement or is triggered based on some conditions. The existing components will be modified as a part of improvements in

order to make them flexible to be dragged and dropped anywhere on the page. Many features will be automated as part of the plan and some dynamic features will also be included for the future. There will be integration of Adobe Analytics with the existing framework and automation of task by making different custom workflows that will reduce the coding intervention.

This project shall be extended further in order to work on the login functionality of the present website. It will include customization of the entire login process that will be confidential and handled with security layers. The future of this project shall also include starting of email functionality for the stakeholders dependent on the platform of Office 365. For implementing this functionality we need to write some custom OSGI configurations which shall be implemented in this email workflow that will be send triggered on some specific activities.

The project can include use of AEM Forms to convert the present form functionality developed with the help of HTML and jQuery and JS validations into AEM Forms. This will make the entire website based on Adobe products (Forms, Analytics etc.) and integrations will be highly efficient and refined.

There are two more projects lined up for this year from different clients which will be based on AEM development and might include dynamic content inclusion which will be handled by the back end logic applied to the custom components.

The scope of learning will extend to learning of various features specific to AEM:

- Dispatcher configurations
- Advanced workflow implementations
- Links and Custom AEM Reports
- Different APIs used in backend logic
- SEO functionalities
- Sonar checking and integration
- Advanced UI and frontend technologies

6.3. CONCLUSION

The internship duration was enriched with learning experience, where a lot of new technologies were learnt theoretically and implemented in the live project. The internship duration was divided into training phase where we learnt the concepts and got the opportunity to research on some domains. The second part of the phase included opportunity to work on a live project to get practical industrial knowledge and experience. The tasks were reviewed and finally sent for the final project merge. This training was fruitful in both industrial and research perspective as there was findings and hands-on on technologies that are new to the industry and have a very wide scope in the near future. Hence, the training was an amalgamation of learning, experience, research and practical knowledge.

REFERENCES

1. Closser, S. (2013). Adobe Experience Manager Quick-Reference Guide: Web Content Management [formerly CQ]. Adobe Press.
2. Hall, R., Pauls, K., McCulloch, S., & Savage, D. (2011). OSGi in action: Creating modular applications in Java. Manning Publications Co.
3. http://www.accunitysoft.com/
4. https://docs.adobe.com/content/docs/en/aem/6-2.html
5. https://docs.adobe.com/docs/en/aem/6-2/develop/ref/granite-ui/api/jcr_root/libs/granite/ui/index.html
6. https://docs.adobe.com/docs/en/htl/overview.html
7. https://sling.apache.org/documentation/bundles/models.html
8. https://docs.adobe.com/docs/en/cq/5-6-1/exploring/architecture-overview.html
9. Loeliger, J., & McCullough, M. (2012). Version Control with Git: Powerful tools and techniques for collaborative software development. " O'Reilly Media, Inc.".
10. Patil, S. (2006). What is Java content repository. O'Reilly Onjava. com.
11. Phil, I. C. E. (2013). Advancing Platform Technologies in Online Learning. In Conference proceedings of» eLearning and Software for Education (eLSE) (No. 01, pp. 616-621). Universitatea Nationala de Aparare Carol I.
12. Rellermeyer, J. S., Alonso, G., & Roscoe, T. (2007, November). R-OSGi: distributed applications through software modularization. In Proceedings of the ACM/IFIP/USENIX 2007 International Conference on Middleware (pp. 1-20). Springer-Verlag New York, Inc..
13. Schwaber, K., & Beedle, M. (2002). Agile software development with Scrum (Vol. 1). Upper Saddle River: Prentice Hall.
14. Siriwardena, P. (2014). Mastering Apache Maven 3. Packt Publishing Ltd.
15. Siriwardena, P. (2015). Maven Essentials. Packt Publishing Ltd.
16. Somasundaram, R. (2013). Git: Version control for everyone. Packt Publishing Ltd.
17. Sutherland, J., Schwaber, K., Scrum, C. C. O., & Sutherl, C. J. (2007). The scrum papers: Nuts, bolts, and origins of an agile process.
18. Vishnu, K. (2017). Web Content Management Infrastructure Migration. International Journal of Research in Computer Engineering & Electronics, 6(2).
19. XUE, S. J., & CHENG, M. (2009). Research on Java Content Repository and

Application in CMS [J]. Computer Technology and Development, 1, 067.