RV Educational Institutions ®
RV College of Engineering ®

Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi

Approved by AICTE, New Delhi

Go, change the world

Major Project
18MCA61

on

"Media Management for Product 360"

Submitted by

Bhavika Vasandani

USN: 1RZ18MCA05

Under the Guidance

of

Dr. Andhe Dharani
Professor and Director
Department of MCA
RV College of Engineering®
Bengaluru – 560059

Rahul Kulkarni
Manager II, IPS
Informatica Business Solutions Pvt. Ltd.
Bengaluru – 560093

Submitted in partial fulfillment of the requirements for the award of degree of

MASTER OF COMPUTER APPLICATIONS

2020-2021

RV COLLEGE OF ENGINEERING®

(Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi)

DEPARTMENT OF

MASTER OF COMPUTER APPLICATIONS

Bengaluru– 560059

CERTIFICATE

Certified that the project work titled **"Media Management for Product 360"** carried out by **Bhavika Vasandani, USN: 1RZ18MCA05**, a bonafide student of **RV College of Engineering®, Bengaluru** submitted in partial fulfilment for the award of **Master of Computer Applications** of **RV College of Engineering®, Bengaluru affiliated to Visvesvaraya Technological University, Belagavi** during the year **2020-21**. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirement in respect of project work prescribed for the said degree.

Dr. Andhe Dharani
Professor and Director
Department of MCA
RVCE, Bengaluru –59

Dr. K. N. Subramanya
Principal
RVCE, Bengaluru–59

RV COLLEGE OF ENGINEERING®

(Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi)

DEPARTMENT OF

MASTER OF COMPUTER APPLICATIONS

Bengaluru– 560059

DECLARATION

I, **Bhavika Vasandani,** student of sixth semester MCA in **Department of Master of Computer Applications,** RV College of Engineering®, Bengaluru declare that the project titled **"Media Management for Product 360"** has been carried out by me. It has been submitted in partial fulfilment of the course requirements for the award of degree in **Master of Computer Applications** of RV College of Engineering®, Bengaluru affiliated to Visvesvaraya Technological University, Belagavi during the academic year **2020-21**. The matter embodied in this report has not been submitted to any other university or institution for the award of any other degree or diploma.

Date of Submission: 15th June 2021 **Signature of the Student**

Student Name: Bhavika Vasandani
USN: 1RZ18MCA05
Department of Master of Computer Applications
RV College of Engineering®
Bengaluru-560059

COMPANY LETTERHEAD

Informatica

Informatica Business Solutions Pvt. Ltd.
No. 66/1, Bagmane Commerz 02
Bagmane Tech Park
C V Raman Nagar
Bangalore, Karnataka
India - 560 093

May 25, 2021

To Whom It May Concern:

This Letter confirms that Bhavika Vasandani (Registration Number: 1RZ18MCA05, College Name: R.V. College of Engineering) is currently pursuing an internship with Informatica Business Solutions Pvt. Ltd from 01/03/2021 to 04/07/2021.

His/Her internship project title is Media Management for P360 is being carried out under the guidance of Rahul Kulkarni, Manager II-IPS.

This provisional letter is provided at the intern's request.

For Informatica Business Solutions Pvt. Ltd

Krishna Narasimhan
Managing Director – Informatica India

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement served a beacon light and served out effort with success.

I would like to profoundly thank management of RV College of Engineering® for providing such a healthy environment for the successful completion of the project. My first sincere appreciation and gratitude to our College Principal Dr. K N Subramanya, Principal, RV College of Engineering®, Bengaluru, for his encouragement that motivated for the success of the project.

I express my sincere thanks and credit to my internal guide Dr. Andhe Dharani, Professor & Director of the Department of MCA, RV College of Engineering® for her constant encouragement, support and guidance during the report work.

I would like to express my sincere gratitude to my external guide Rahul Kulkarni for his insightful comments and suggestions. Without his tremendous understanding and encouragement it would be impossible for me to complete my project successfully.

Finally, my deepest appreciation and gratitude to my beloved parents and friends who have been a fountain of inspiration and have provided unrelenting encouragement and support. And I also thank all staff and friends of the RV College of Engineering® for their kind corporation

Bhavika Vasandani

Department of MCA

RV College of Engineering®

Bengaluru-59

ABSTRACT

The need for businesses to improve the consistency and quality of their essential data assets, such as product data, asset data, customer data, and location data, prompted the development of master data management (MDM). Product 360 offers a mature platform that can handle large amounts of data and complex scenarios with billions of attributes. It can handle and manage millions of hierarchies, classes, goods, objects, and variants without sacrificing performance. Media Management is one of the solution offered by Product 360

The current Media Asset Manager (MAM) available for use with Informatica P360 has not been upgraded well and thus lack to satisfy various customer requirements leading to fair amount of frustration among them. The Media manager solution is aimed to address several limitations of Informatica Media Manager (IMM). To provide scalability the applications are written in terms of set microservices where each microservice provides a different service.

The object-oriented methodology is used to build this application as the complexities of functionalities are high and the application is dynamic. The project is developed using an incremental agile development approach, the programming approach is a bottom-up approach. The domain of the project is Master Data Management. Tools used are VS code, Eclipse IDE, AWS, MongoDB Compass. Technologies used are Java, React JS. AWS S3 for storing the media files and MongoDB for storing the metadata. GIT has distributed version control system which is used to deploy the code online.

The outcome of the application is to provide a Web UI which will perform all the media management tasks. The proposed system is designed to be more user friendly and also aimed to improve upon the previous technology stack that was used. The key challenges faced while implementing MAM solution using Informatica Media Manager are Complicated UI- difficult to navigate and Multi module complex application which are trying to overcome using our present solution

List of Publications

Publication Type	Publication Details
Journal	" Migrating from Monolithic to Microservice Architecture using Open Source tools " , PENSEE Journal, Volume 51, Issue 06, June-2021. Manuscript ID: PNS-0621-114

Table of Contents

CONTENTS	PAGE NO.
College Certificate	i
Company Certificate	ii
Declaration by student	iii
Acknowledgement	iv
Abstract	v
List of Publications	vi
Table of Contents	vii
List of Tables	viii
List of Figures	ix
Chapter 1: Introduction	
1.1 Project Description	1
1.2 Company Profile	2
1.3 Dissertation organisation	4
Chapter 2: Literature Review	
2.1 Literature Survey	5
2.2 Existing and Proposed System	7
2.3 Tools and Technologies used	9
2.4 Hardware and Software Requirements	11
Chapter 3: Software Requirement Specifications	
3.1 Introduction	24
3.2 General Description	25
3.3 Functional Requirement	27
3.5 Non-Functional Requirements	29
Chapter 4: System Design	
4.1 System Perspective /Architectural Design	30
4.2 Context Diagram	33

Chapter 5: Detailed Design

5.1 System Design 34

Chapter 6: Implementation

6.1 Code Snippets / PDL 40
6.2 Implementation 47

Chapter 7: Software Testing

7.1 Test cases 52
7.2 Testing and Validations 54

Chapter 8: Conclusion 55

Chapter 9: Future Enhancements 56

Bibliography 57

Plagiarism Report 62

Certification of Publication 63

Technical Paper 64

List of Tables

Table No.	Table Description	Page No.
2.1	Minimum Hardware Requirement	11
2.2	Recommended Hardware Requirement	11
2.3	Software Requirement	12
3.1	Acronyms and Abbreviations	24
4.1	Data dictionary of Media	31
4.2	Data dictionary of media Hierarchy	31
7.1	Test Cases for Home Page	49
7.2	Test Cases for Media Metadata Page	56

List of Figures

Figure No.	Figure Description	Page No.
4.1	Block Diagram	30
4.2	Context Diagram	33
5.1	Architectural Diagram	34
5.2	Activity Diagram	35
5.3	Sequence Diagram	36
5.4	Data Flow Diagram Level 0	37
6.1	Home Page	51
6.2	Media Page	52
6.3	Media Management Page	53
6.4	Menu containing hierarchy	54
6.5	Search Page	55
7.1	File upload successful	58
7.2	File upload failed	58

Chapter 1: Introduction

This chapter gives an insight about basic introduction of the project along with theory and concept relevant to the project.

1.1 Project Description

Product 360 is one of the solution offered by Informatica which is responsible for PIM solution powered by master data. It is designed to handle commercial use cases while also providing sophisticated product content management offering reliable, rich, and relevant product data across all channels. Product 360 provides a unique omnichannel product experience and client engagement. Product 360 is in charge of master data management, which is meant to function in conjunction with other data management software.

PIM is the process of organizing and managing the data needed to market and sell items through various distribution channels. Websites, print catalogues, ERP systems, and electronic data flows to trading partners can all benefit from a centralized set of product data.

Media management is one of Product 360's services, and it relates to the online management of multimedia assets. The user can do a search for certain media files, upload or download them, and allocate them to specific products or commodities. Additional material, such as photographs or text documents, can be found in catalogue items in addition to raw data. MIME is an Internet standard that stores this information. "MIME data," "MIME information," and "multimedia attachments" are frequently interchanged.

Media management is one of the services that Product 360 offers which refers to management of multimedia assets through the web. The Web UI makes it simple to manage all types of multimedia materials. The user can do a search for certain media files, upload or download them, and allocate them to specific products or commodities. The usage list provides a good understanding of all of the media files' dependencies.

1.2 Company Profile

Informatica Corporation is the leading independent data integration software supplier in the world. Organizations all around the world gain a competitive advantage in today's global information economy by using timely, relevant, and trustworthy data for their essential business imperatives.

Organizations rely on Informatica to help them streamline their supply chain, leverage powerful analytics, and speed the modernization of data warehouses and apps. We are trusted by 85 of the Fortune 100. We combine their data so they can view the current clearly and make informed judgments. We may use Informatica to turn our data into a dynamic asset for analysis, modernization, contextualization, monetization, and even harmonization. Informatica helps companies bring their data to life, whether it's on-premises, in multiple clouds, or anywhere else.

Intelligent Data Management Cloud spans all data services and is built on an API-driven, serverless platform that is powered by cutting-edge AI to scale to large enterprises across all industries. Intelligent Data Management Cloud transforms it from a static asset to a dynamic asset that propels our business forward.

Whether data is multi-cloud or on-premises, our data integration technologies connect all of our data and apps together in batch or real time. We can combine data and applications in minutes and handle new and complicated integration patterns with ease. Get great performance, dependability, and near-universal connection for mission-critical business processes.

Cloud Data Integration- Using any of the top cloud platforms, create a cloud data warehouse or keep one on-premises. To get up and running quickly and easily integrate huge volumes of data, connect to on-premises data sources and cloud apps**.**

Cloud Application Integration-Integrate applications from Salesforce, Microsoft Azure, and Amazon Web Services, as well as on-premises apps, services, and communications platforms. Automate processes and speed up transactions using the Informatica Cloud Application.

Powercenter- Informatica PowerCenter helps us complete on-premises data integration and data warehouse projects faster. From launching first project to directing

mission-critical business installations, quickly support the whole data integration lifecycle.

The glue that holds our systems and data together is a master data management solution. It's our data-driven digital transformation's one source of truth, offering reliable, accurate, and full data for our customer experience programme, marketing and sales operations, omnichannel commerce, supply chain optimization, governance activities, compliance initiatives, and more.

Informatica provides a modular, all-in-one MDM solution that is built for flexibility: Begin by addressing our most important data issues and business requirements, then expand the system as our requirements increase. As we get more clarity with our data, we will be able to extract more business value from it.

- A unified view of the information: From fragmented, duplicate, and contradictory data sources to develop a reliable perspective of our mission-critical data
- A complete picture of the relationships: Locate links between consumers, products, suppliers, and more by identifying relationship insights inside our data.
- A full picture of all interactions: To get a complete picture of a customer's activity, connect transactions and interactions.

Informatica MDM makes use of artificial intelligence and machine learning to make sure we can find, access, and use reliable data when and when we need it As a true end-to-end solution, it provides data quality, data integration, business process management, and data security, allowing us to:

- Obtain data rapidly, regardless of the source (on-premises, in the cloud, or from third parties)
- Get a comprehensive understanding of our data, relationship patterns, and variations—and make any modifications that are necessary.
- Easily add data from other sources to master data records.
- Provide a trustworthy view and securely communicate data.

1.3 Dissertation Organization

The following is how the rest of the dissertation is structured: In Chapter 2 background material in the form of a brief historical perspective of last few years of related software engineering research paper are considered and studied. Specifically, Chapter Two covers understanding about MDM tools. It also contains details on the tools and technologies used and required for developing and implementing this system.

Chapter Three discusses software requirements, in terms of functional and non-functional requirements. In Chapter Four presents the system architecture through block diagram, architectural diagram, module specification & context diagram. Chapter Five describes the detailed design of the system through activity diagram, sequence diagram.

In Chapter Six the PDL of every module along with screenshots of the system with a short description. Chapter Seven reveals the testing's that was performed.. The screenshots of validations done are also present in this chapter. Finally, Chapter Eight presents conclusions drawn and Chapter Nine future work.

This chapter introduced the project as well as the theory and concept relevant to the project and the company profile is well understood in detail.

Chapter 2: Literature Review

This chapter gives an insight about the description of the survey of literature, existing and proposed system, Hardware and Software Requirements along with Tools and Technologies used in implementing the project.

2.1 Literature Survey

In paper[24] author Kokemüller, Jochen & Anette, Weisbecker has described Master Data Management. The discipline of master data management is the process of developing and maintaining high-value, high-quality data. We define this data category in this contribution, emphasising its importance in maintaining a high degree of cooperative data quality. A discussion of the current state of commercial master data management solutions is given. It is based on the findings of six systems of two surveys that we have conducted.

The article[1] discusses MDM's global forecast. During the projected period, the global master data management (MDM) market is estimated to increase from USD 11.3 billion in 2020 to USD 27.9 billion in 2025, with a Compound Annual Growth Rate (CAGR) of 19.8%. The master data management market is predicted to rise due to an increase in the usage of data quality tools for data management, as well as a growing requirement for compliance that accurately offers projections for speedier decision making. However, one of the problems impeding the industry's expansion is data security worries.

The paper[20] deals with composition of data, processes, and information systems is one master data management framework. As a result, the main data difficulties are that master data definitions are ambiguous, and overall data quality is low. Inadequately defined data ownership, inconsistent data management methods, and a lack of continual data quality practices are all challenges in master data management operations. When developing a comprehensive one master data, integrations between apps are a major issue to overcome.

This paper[45] intends to provide a single master data specification for cross-application consistency. The notions of master data management have been considered in a larger sense. The current issues that businesses face while deploying MDM systems are discussed. We used a case study to demonstrate why Master Data Management is

critical for businesses to optimise their operations. We've also identified some of the long-term benefits of MDM implementation for businesses.

The article[2] deals with React. React is the best method to use JavaScript to create large, fast Web apps. It helps scale quite effectively. React's ability to make us think about apps as we build them is one of its many strengths. It talks through the process of using React to create a searchable product data table.

AWS S3 is a highly trustworthy online service that allows developers to securely store and retrieve object data in the Amazon Web Services cloud, according to the paper [3]. After Amazon EC2, Amazon S3 is one of the most popular services. Within a region, data on Amazon S3 is spread automatically over several devices and availability zones. Amazon's S3 service is an object-based storage service. It's fantastic for storing files, but it can't install an operating system, so it's not suitable for EC2 storage. In Amazon S3, data is saved using a keyvalue system with globally unique keys. Some of the most essential topics that developers will come across when dealing with Amazon S3 are as follows: a bucket, a key for an object, a value for an object, a version ID, a storage type, and a subrand object metadata.

The proposed system will be developed using an headless architecture and the technologies and tools used to develop the web application will be React JS, a light weight tool with large community and structure based on components makes it very easy to use and render faster. Secondly, MongoDB a Document based NoSQL which stands powerful way to store and retrieve the data with the support for huge volume of data and traffic. AWS S3 to store files and retrieve any amount of data at any time, from anywhere on the web. P360 to Preview template / Customized media views & perspectives Integration to be performed using REST APIs to coordinate server activities.

2.1.1 Literature Survey inference

Ref no. 24 intends to provide a single master data specification for cross-application consistency. The notions of master data management have been considered in a larger sense. The current issues that businesses face while deploying MDM systems are discussed. Considering case study to demonstrate why Master Data Management is

critical for businesses to optimise their operations. We've also identified some of the long-term benefits of MDM implementation for businesses.

Ref no 2 deals with React implementation to create a searchable product data table.

Ref no 3 explains how to create Amazon S3 buckets and manage the content in them using the AWS management console.

2.2 Existing and Proposed System

2.2.1 Problem statement and Scope of the project

i. Problem statement

The key challenges faced while implementing MAM solution using Informatica Media Manager are:

- Complicated UI- difficult to navigate
- Multi module complex application

ii. Scope of the project

The media manager solution is aimed to address several limitations of Informatica Media Manager[IMM]. The solution will compromise of the following features:

- Ability to store and maintain media files using a bucket service such as S3
- Ability to store and maintain media metadata in a way that it is easy to extend
- Integration with Product 360
- Ability to configure image derivative generation
- Security on assets based on the item and hierarchy
- Ability to display images under an item
- Ability to extract a specific frame from a video or thumbnail
- Automatically crop pictures
- Display image hierarchy
- Standard checks using DQ and AVOS

- Ability to scale for handling media requests, media uploads/downloads and derivative generation
- Containarize MAM application

However the full-fledged media manager solution shall be developed in various phases.

2.2.2 Methodology adopted in the proposed system

The proposed system will be developed using an headless architecture and the technologies and tools used to develop the web application will be React JS, a light weight tool with large community and structure based on components makes it very easy to use and render faster. Secondly, MongoDB a Document based NoSQL which stands powerful way to store and retrieve the data with the support for huge volume of data and traffic. AWS S3 to store files and retrieve any amount of data at any time, from anywhere on the web. P360 to Preview template / Customized media views & perspectives Integration to be performed using REST APIs to coordinate server activities.

2.2.3 Technical Features of the proposed system

- The existing media manager module of Product 360 is not very user-friendly thus the aim is to improve upon the existing UI.
- Ability to store and maintain media files using a bucket service such as S3.
- Ability to store and maintain media metadata in a way that is easy to extend.
- Integration with Product 360. - Ability to configure image derivative generation.
- Ability to scale for handling media requests, media uploads/downloads and derivative generation.

2.3 Tools and Technologies used

Visual Studio Code

Microsoft's Visual Studio Code is a freeware source-code editor. Debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git are among the features. Users can customise the theme, keyboard shortcuts, and preferences, as well as instal extensions that offer additional functionality.

Eclipse IDE

Our Java Integrated Development Environment (IDE) is well-known, but we also have a variety of other useful IDEs, such as our C/C++ IDE, JavaScript/TypeScript IDE, PHP IDE, and others. Any of our basic packages may simply be customised and extended to support many languages and other features, and the Eclipse Marketplace allows for nearly endless modification and extension.

Spring Tools 4

Spring Tools 4 is the next generation of Spring tooling for our development environment of choice. It offers world-class support for developing Spring-based enterprise apps, regardless of whether we choose Eclipse, Visual Studio Code, or Theia IDE.

Java

Java is a high-abstraction object-oriented programming language with as few implementation dependencies as feasible. It's an overall programming language that lets developers write once and run anywhere (WORA), which implies that compiled Java code may run on any platform that supports Java with no need to recompile. Java programmes are often compiled to bytecode, which may run on any Java virtual machine (JVM) independent of computer architecture. Java has a comparable syntax to C and C++, although it has less low-level functionality

Amazon Web Services (AWS)

Amazon Web Services (AWS) is an Amazon subsidiary that provides individuals, corporations, and governments with billed pay-as-you-go cloud services and APIs. A range of basic abstract technological infrastructure and cloud control

building blocks and utilities are included in these web services for cloud computing. Amazon Elastic Compute Cloud (EC2) is another one of these services that enables clients to easily access a virtual machine cluster across the Internet. On AWS, virtual machines feature hardware and graphics processing units (GPUs), and the local / RAM storage capacity, hard disc / SSD systems, networking and pre-loaded programmes such as web servers, databases, and the customer relation management (CRM).

Amazon S3

Amazon S3 is an object storage service with industry-leading scalability, data availability, security, and performance. Clients of different sizes and areas can utilize it to store and defend any amount of information for an assortment of utilization cases, including data lakes, websites, mobile apps, backup and restore, archive, business applications, IoT device and big data analytics. We can coordinate our information and build up finely-tuned admittance controls to coordinate with our particular business, authoritative, and consistence prerequisites utilizing Amazon S3's not difficult to-utilize organization capacities Amazon S3 is worked to last 99.999999999 percent of the time (11 9's) and saves information for a large number of applications for organizations everywhere on the world.

React JS

React makes making intuitive UI exceptionally straightforward.Create basic views for each state of our project, and React will update and render the appropriate components as our data changes. Declarative views improve the predictability and debugability of our code. Compose encapsulated components that handle their own state to create complicated user interfaces. We can simply transmit rich data through our app and keep state off of the DOM because component functionality is written in JavaScript rather than templates. We can use React to create new features without having to rewrite old code. React can likewise utilize Node to deliver on the worker and React Native to control mobile applications.

2.4 Hardware and Software Requirements

Hardware and software requirements show what will be the minimum and recommended requirement of a system if it wants to deploy this application, operating systems are not an issue since the application will be launched in a Docker container.

2.4.1 Hardware Requirements

Table 2.1 Minimum Hardware Requirement

Processor	Intel Pentium 1.8 GHZ
RAM	1 GB
Disk Space	512 MB

Table 2.1 represents the minimum requirement to deploy the application, but this is the bare minimum and the system will struggle with the performance and would not perform up to standard.

Table 2.2 Recommended Hardware Requirement

Processor	Core i5, 2.5 GHZ
RAM	4 GB
Disk Space	20 GB

Table 2.2 shows the recommended requirement to deploy the application, and with this configuration, the application will also perform well and up to the required performance.

2.4.2 Software Requirements

Table 2.3 Software Requirement

Operating System	Windows 7/8/10, Mac OS 10.8.5, Linux
Languages	Java and React

Tools used	Visual Studio Code & Eclipse IDE
Browser	Firefox, chrome, safari

In table 2.3 represent the software required for the application. The application is written in java for the backend and React for the frontend since it is component-based and has a virtual DOM. VS Code is required since the application is a web application and the editor are made for and compatible for web development languages.

Installation and Usage

Application can be installed by downloading docker containers. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

Pre-requisite: Docker Installed

BACKEND CONTAINER

To containerize spring boot server application a docker file would be needed which consists of commands to install the dependencies and copy the code to docker container.

```
FROM openjdk:17-jdk-alpine
ENV AWS_SECRET_ACCESS_KEY=<provide the key>
ENV AWS_ACCESS_KEY_ID=<provide the key>
ARG JAR_FILE=target/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

Figure 2.1: Sample Docker File for spring boot application containerization.

Create docker image by running below command from the directory where docker file is present.

- docker build -t springio/mam-api .

Once the docker image is created any user can create containers from the image and start using the application in their local machine by running below command.

- docker run -dp <docker-port>:<server-port> springio/mam-api

FRONTEND CONTAINER

To containerize reactjs application a docker file would be needed which consists of commands to install the dependencies and copy the code to docker container.

```
# pull official base image
FROM node:latest



RUN mkdir /app
# set working directory
WORKDIR /app

# add `/app/node_modules/.bin` to $PATH
ENV PATH /app/node_modules/.bin:$PATH

# install app dependencies
COPY package.json /app
COPY package-lock.json /app
RUN npm install


# add app
COPY . /app

# start app
CMD ["npm", "start"]
```

Figure 2.2: Sample Docker File for ReactJs application.

Create docker image by running below command from the directory where docker file is present.

- docker build -t mamwebapp/react-app .

Once the docker image is created any user can create containers from the image and start using the application in their local machine by running below command.

- docker run -d -it -p <docker-port>:<react-port> /tcp --name react-app mamwebapp/react-app

API endpoints

1. GET

Below are different GET APIs to fetch the information about media, media hierarchies and derivative configuration.

- Media Metadata: This is used to fetch media metadata like File name, ID, extension, size, etc

URL Pattern	/media
Method	GET
Return type	application/json
Result	Returns metadata for all the medias.

Example	"_embedded": {
	"medias": [
	{
	"id": "60826c2ba5cc4d12dbec3998",
	"fileName": "PokemonJigglypuff.png",
	"fileExtension": "png",
	"mimeType": "image/png",
	"fileSize": 176804,
	"fileEncoding": "UTF8",
	"url": "https://jbowring-mam-hackathon.s3.ap-southeast-2.amazonaws.com/60826c2ba5cc4d12dbec3998/PokemonJigglypuff.png",
	"width": 0,
	"height": 0,
	"hierarchyCode": "1234-",
	"derivatives": {},
	"_links": {
	"self": {
	"href": "http://localhost:8080/media/60826c2ba5cc4d12dbec3998"
	},
	"medias": {
	"href": "http://localhost:8080/media"
	}
	}
	}

	] }

- Media Metadata by Id: This API is used to fetch metadata for specific media by specifying the id in the get request

URL Pattern	/media/{id}
Method	GET
Return type	application/json
Result	Returns metadata for specific media whose id is part of the get request

Example	{
	"id": "60826c2ba5cc4d12dbec3998",
	"fileName": "PokemonJigglypuff.png",
	"fileExtension": "png",
	"mimeType": "image/png",
	"fileSize": 176804,
	"fileEncoding": "UTF8",
	"url": "https://jbowring-mam-hackathon.s3.ap-southeast-2.amazonaws.com/60826c2ba5cc4d12dbec3998/PokemonJigglypuff.png",
	"width": 0,
	"height": 0,
	"hierarchyCode": "1234-",
	"derivatives": {},
	"_links": {
	"self": {
	"href": "http://localhost:8080/media/60826c2ba5cc4d12dbec3998"
	},
	"medias": {
	"href": "http://localhost:8080/media"
	}
	}
	}

- File by Id: This API downloads media file from Amazon S3.

URL Pattern	/media/{id}/file
Method	GET
Return type	application/json
Result	Downloads specific file from Amazon S3 to local system by specifying id as part of get request

- Zip file to download medias: This API downloads one or more files as zip file from Amazon S3.

URL Pattern	/media/file
Method	GET
Return type	application/json
Result	Downloads one or more files from Amazon S3 to local system as zip file.

- Media Hierarchy: This API returns all the media hierarchies.

URL Pattern	/mediaHierarchy
Method	GET
Return type	application/json
Result	Returns media hierarchies stored in Mongo DB

Example	"_embedded": {
	"mediaHierarchies": [
	{
	"id": "602e495ad114ca7fd3fa99f2",
	"hierarchyCode": "1234",
	"hierarchyName": "Images",
	"children": [
	{
	"id": null,
	"hierarchyCode": "1234-1",
	"hierarchyName": "Product Images",
	"children": []
	},
	{
	"id": null,
	"hierarchyCode": "1234-2",
	"hierarchyName": "Item Images",
	"children": []
	}
	]
	},
	{
	"id": "602e899019e8c76caecfe664",
	"hierarchyCode": "12345",
	"hierarchyName": "Documents",
	"children": [
	{
	"id": null,

```
                "hierarchyCode": "1235-1",
                "hierarchyName": "SDS",
                "children": []
            },
            {
                "id": null,
                "hierarchyCode": "1235-2",
                "hierarchyName": "Reports",
                "children": []
            }
        ]
    }
]
}
```


- Media Hierarchy by Id: This API returns specific media hierarchy.

URL Pattern		/mediaHierarchy/{id}
Method		GET
Return type		application/json
Result		Returns specific media hierarchy stored in Mongo DB which is specified by id as part of get request

Example		{
		"id": "602e495ad114ca7fd3fa99f2",
		"hierarchyCode": "1234",
		"hierarchyName": "Images",
		"children": [
		{
		"id": null,
		"hierarchyCode": "1234-1",
		"hierarchyName": "Product Images",
		"children": []
		},
		{
		"id": null,
		"hierarchyCode": "1234-2",
		"hierarchyName": "Item Images",
		"children": []
		}
		]
		}

- Media derivative config: This API returns derivative configs for medias.

URL Pattern	/config
Method	GET
Return type	application/json
Result	Returns media derivatives configuration stored in MongoDB

Example	{
	"_embedded": {
	"configs": [
	{
	"id": "derivatives",
	"properties": {
	"thumbnail": "th:50:50",
	"large": "lrg:2000:-1",
	"web": "web:300:300"
	}
	}
	]
	},
	"_links": {
	"self": {
	"href": "http://localhost:8080/config"
	}
	}
	}

- Media derivative config by Id: This API returns specific derivative configs for medias.

URL Pattern	/config/{id}
Method	GET
Return type	application/json
Result	Returns media derivatives configuration stored in MongoDB

Example	{
	"id": "derivatives",
	"properties": {
	"thumbnail": "th:50:50",
	"large": "lrg:2000:-1",
	"web": "web:300:300"
	}
	}

2. PUT

The put APIs are used to update media, media hierarchies and derivatives. Below are the put APIs available:

- Update Media Metadata doc for an id : This API updates media metadata document for media specified by media id

URL Pattern	/media/{id}
Method	PUT

Request Body	{
	
	"hierarchyCode": 123456,
	"fileName": "test.jpg"
	
	}

Result	Update's media metadata information stored in MongoDB for media specified by media id as part of the URL

Response	{
	"id": "60826c29a5cc4d12dbec3995",
	"fileName": "test.jpg",
	"fileExtension": null,
	"mimeType": null,
	"fileSize": null,
	"fileEncoding": null,
	"url": null,
	"width": 0,
	"height": 0,
	"hierarchyCode": "123456",
	"derivatives": {},
	"_links": {
	"self": {
	"href":
	"http://localhost:8080/media/60826c29a5cc4d12dbec3995"
	},
	"medias": {
	"href": "http://localhost:8080/media"
	}
	}
	}

- Update Media Hierarchy: This API updates media hierarchy information for hierarchy specified by id

URL Pattern	/mediaHierarchy/{id}
Method	PUT

Request Body	<pre>{
	"hierarchyCode": "12345",
	"hierarchyName": "Documents",
	"children": [
	{
	"hierarchyCode": "1235-1",
	"hierarchyName": "SDS",
	"children": []
	},
	{
	"hierarchyCode": "1235-2",
	"hierarchyName": "Reports",
	"children": []
	}
	]
	}</pre>

Result	Update's media hierarchy information stored in MongoDB for hieracrhy specified by id as part of the URL

Response	<pre>{
	"id": "602e899019e8c76caecfe664",
	"hierarchyCode": "12345",
	"hierarchyName": "Documents",
	"children": [
	{
	"id": null,
	"hierarchyCode": "1235-1",
	"hierarchyName": "SDS",
	"children": []
	},
	{
	"id": null,
	"hierarchyCode": "1235-2",
	"hierarchyName": "Reports",
	"children": []
	}
	]
	}</pre>

- Update Derivative Configuration: This API updates media derivative configuration for configuration specified by id

URL Pattern	/config/{id}
Method	PUT

Request Body	{
	"id": "derivatives",
	"properties": {
	"thumbnail": "th:50:50",
	"large": "lrg:2000:-1",
	"web": "web:300:300"
	}
	}

Result	Update's media derivative configuration stored in MongoDB for configuration specified by id as part of the URL

Respose	{
	"id": "derivatives",
	"properties": {
	"thumbnail": "th:50:50",
	"large": "lrg:2000:-1",
	"web": "web:300:300"
	}
	}

3. POST

The POST APIs are used to create medias, media hierarchies and derivative configuration, Below are the POST APIS currently available:

- Create (Upload) Media:

URL Pattern	/media
Method	POST
Encoding Type	"multipart/form-data"
Result	Upload's media files to amazon s3 and creates metadata doc in mongodb

- Create Media Hierarchy

URL Pattern	/mediaHierarchy/create
Method	POST
Request Body	{

	"hierarchyCode": "1234",
	"hierarchyName": "Images",
	"children": [
	{
	"hierarchyCode": "12341",
	"hierarchyName": "Product Images",
	"children": []
	},
	{
	"hierarchyCode": "12342",
	"hierarchyName": "Item Images",
	"children": []
	}
	]
	}

Result	Creates new media hierarchy.

- Create Derivative Configuration

URL Pattern	/config
Method	POST

Request Body	"properties": {
	"thumbnail": "th:50:50",
	"large": "lrg:2000:-1",
	"web": "web:300:300"
	}
	}

Result	Creates new derivative configuration.

4. **DELETE**

The Delete APIs delete medias and its metadata information, media hierarchies and derivative configurations:

- Delete Media:

URL Pattern	/media/{id}
Method	DELETE
Result	Deletes media and its metadata information specified by id.

- Delete Media Hierarchy:

URL Pattern	/mediaHierarchy/{id}
Method	DELETE
Result	Deletes media hierarchy specified by id.

- Delete Derivative Configuration:

URL Pattern	/config/{id}
Method	DELETE
Result	Deletes derivative configuration specified by id.

Chapter 3: Software Requirements Specification

This chapter gives a brief description about functional and non-functional requirements.

3.1 Introduction

The Software Requirements Specification lays the foundation for the systematic approach towards the development of this project. It provides a general overview of the system. It helps to translate the ideas in the minds of the clients. It includes understanding of the problem domain, identifying all external entities that will interact with the system, the users and defining their functions, the constraints that would exist and the overall functioning of the system.

The glue that holds our systems and data together is a master data management solution. It's our data-driven digital transformation's one source of truth, offering reliable, accurate, and full data for our customer experience, marketing and sales operations, omnichannel commerce, supply chain optimization, governance activities, compliance initiatives, and more.

Product 360 is a robust PIM solution that enables business users to gather, author, and publish product information more efficiently, from any location. Digital asset management combines embedded digital asset management with fully automated data processing capabilities to centrally manage complicated product data and vast collections of media assets in any format.

3.1.1 Definitions, Acronyms and Abbreviations

Table 3.1 Acronyms and Abbreviations

MDM	Master Data Management
RAM	Random Access Memory
PIM	Product Information Management
SRS	Software Requirements Specification

3.2 General Description

This section will give an overview of the service. The service will be explained in its context to show how the service interacts with other products and introduce the basic functionality of it. At last, the constraints and assumptions for the service will be presented.

3.2.1 Product Perspective

The Web UI promotes a collaborative approach as part of company-wide involvement in the product information management process because of the huge number of potential users, which includes product managers, sales and marketing employees, text creators and translators, and visual designers. The emphasis is on ease of use and straightforward operation.

- **User Dashboards**- provide a visual depiction of an aggregated view of data quality, process, and task information, as well as KPI information. This helps with a variety of tasks in the product mastering process (e.g. data stewards, campaign managers, management, buyers) Individual dashboards can be created.Digital asset management.
- **Media Management**- The Web UI makes it simple to manage all types of multimedia materials. The user can look for certain media files, upload or download them, and allocate them to various product categories. The usage list provides a good understanding of all of the media files' dependencies.
- **Data Upload-** The import of this data begins when we submit a CSV, Excel, or XML file. In a single step, media asset files can be uploaded and assigned to products and items.
- **Immediate Export-** Immediate export allows us to export single, many, or chosen data sets in a variety of file formats (e.g. CSV, XML, etc.). If the user prepared numerous files using the immediate export, he or she can download them all at once.
- **Data Maintenance-** Tables and forms are used to edit item and product data, and a rich text editor is used to enter text formatting.The maintenance status indicates whether or not the items are complete. Text mark-ups can also be used in structure group descriptions on the Server.

3.2.2 Product Functions

The goal is to create a data media management service that combines embedded digital asset management with fully automated data processing capabilities to handle and centrally manage complicated product data and massive collections of media assets in any format.

- Ability to store and maintain media files using a bucket service such as S3
- Ability to store and maintain media metadata in a way that it is easy to extend
- Integration with Product 360
- Ability to configure image derivative generation
- Security on assets based on the item and hierarchy
- Ability to display images under an item
- Ability to extract a specific frame from a video or thumbnail
- Automatically crop pictures
- Display image hierarchy
- Standard checks using DQ and AVOS
- Ability to scale for handling media requests, media uploads/downloads and derivative generation
- Containarize MAM application

3.2.3 User Characteristics

The user will be able to carry out the following functions:

- Manage media upload/download/deletion
- Manage media attribute information
- Search functionality- to search the media assets
- Derivative generation
- Media assignment to items
- Media classification- Hierarchy
- Integration with P360- using rest APIs and Preview Templates
- Containerized application

3.2.4 General Constraints

Docker must be Installed

3.2.5 Assumptions and Dependencies

The user should make sure they have a stable internet connection. Also, the browser must be at its most recent update.

3.3 Functional Requirements

The goal is to create a data media management service that combines embedded digital asset management with fully automated data processing capabilities to handle and centrally manage complicated product data and massive collections of media assets in any format.This project is built in five modules. The description of the modules is given below:

Module Name : Home page Module

Purpose : Responsible for displaying all the media along with all the features

Input : Loading the url – a https url should be loaded which will be bind with a domain server.

Function:

- Display all the media
- Display all other navigation menu

Output : All the media added by the admin are displayed on page fetched from the database

Module Name : Media Module

Purpose : To display all the details and features of the products to the user

Input : Click on any media in home page

Function:

- Details of selected product is displayed on page
- download, delete and update options are provide

Output : Details of selected product is displayed on page with download, delete and update option details will be loaded from the database

Module Name : Media Management Module

Purpose : Responsible for deleting or download a media file

Input : Click on delete or download button

Function:

- Manage the media
- Perform various operations like update ,delete, download

Output : The media file is deleted, downloaded or updated.

Module Name : Hierarchy Module

Purpose : To display all media related to that particular hierarchy

Input : Click on hierarchy name in the library menu

Function:

- Manage the hierarchy and subhierarchy
- Media under hierarchy and subhierarchy can be displayed

Output : Display filtered media based on hierarchy code

Module Name : Media Search Module

Purpose : Responsible to display all media having similar key or filename

Input : Key or File name for filter

Function:

- Display all media having similar key or filename based on pattern matching

Output : Display filtered media based on key or filename

3.4 External Interfaces Requirements

3.4.1 User Interface

The user interface for the software shall only be compatible to modern browsers.

3.4.2 Hardware Interface

- INTERNET: A stable online connection
- MONITOR: Any LCD, LED, TFT displays can be used The above hardware requirements are mandatory without which the web application won't be able to display.

3.4.3 Software Interface

System browser, APIs are used as well as different technologies and libraries have been made use of.

3.4 Non-Functional Requirements

- Reliability: The application should ensure that it will not fail even under extreme situations. The Media management application should not break when dealing with huge media files.
- Availability: The services of the applications must be available to all users at all times; if servers are being maintained, it should be ensured that the maintenance will not take long and that it will be done when the number of users using the application is low.
- Security: All the services of the application are be secured and can only be accessed when we login to the system. Security would be ensured by performing authorization and authentication through MAM specific user management (non-SSO/LDAP)
- Scalability: The application must be able to handle a large number of requests at the same time and provide high scalability.

This chapter showed brief description about project and also functional and non-functional requirements.

Chapter 4: System Design

4.1 System Perspective/Architectural Design

4.1.1 Problem Specification

The problem statement involves

- Update the MongoDB Media model to store derivative information
- Create / Update MediaAssetFile object in PIM
- Delete MediaAssetFile Object in PIM
- Web UI for MAM functionality
- Asset Create, Update, Delete and Validate Using AVOS
- Integration of PIM with MAM
- Download a selected image

4.1.2 Block Diagram

The block diagram is a diagram of a system in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks.

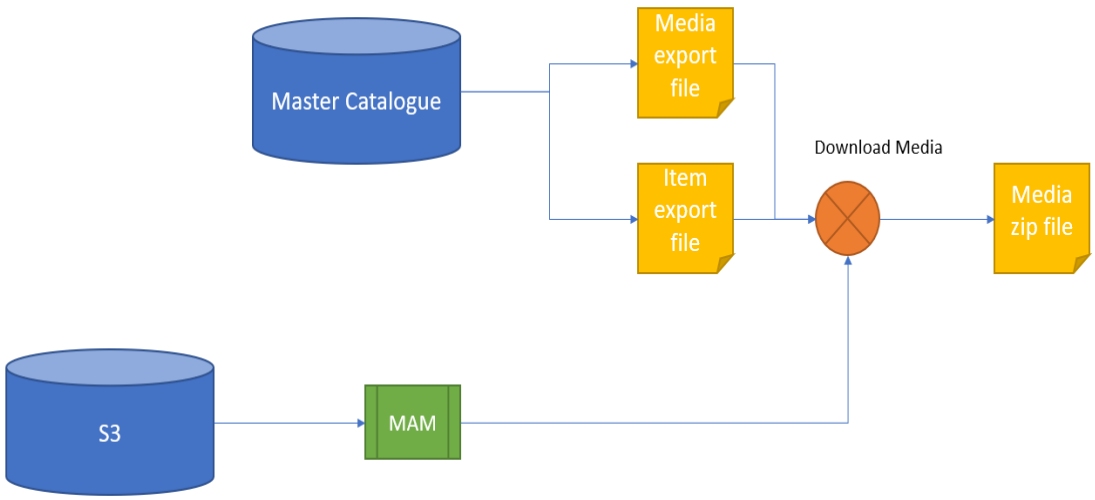


Figure 4.1: Block diagram

4.1.3 Data Definition

Media

Table 4.1: Data dictionary of media

Column name	Constraint	Data Types
Id	Primary Key	objectid
Class	Not null	String
fileEncoding	Not null	String
fileExtension	Not null	String
fileSize	Not null	Long
Height	Not null	int32
hierarchyCode	Not null	String
mimeType	Not null	String
url	Not null	String
Width	Not null	String

mediaHierarchy

Table 4.1: Data dictionary of mediaHierarchy

Column name	Constraint	Data Types
Id	Primary Key	objectId
Class	Not null	String
Children	Not null	varchar
hierarchyCode	Not null	String
hierarchyName	Not null	String

4.1.4 Module Specification

Module 1: Home Module

Home Module is responsible to display all the media along with all the links for navigation when loading the url – a https url should be loaded which will be bind with a domain server.

Module 2: Media Module

Media Module is responsible to display the details of selected product is displayed on page with download, delete and update option details will be loaded from the database.

Module 3: Media Management Module

Media Management service is responsible for managing the media either audio or video. Manage the media by perform various operations like update or delete

Module 4: Hierarchy Module

Hierarchy Module is responsible to display all media related to that particular hierarchy. Also manage various hierarchies and sub-hierarchies.

Module 5: Media Search Module

Media Search Module is responsible to display filtered media based on key or filename based on the typed text pattern.

4.2 Context Diagram

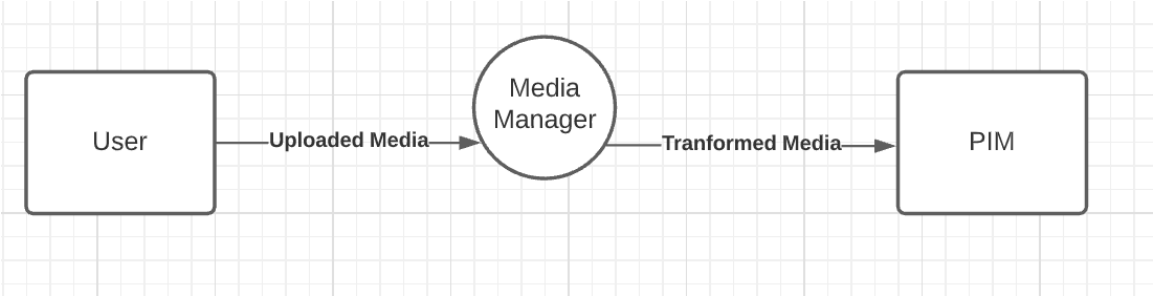


Figure 4.2: Context diagram

Context diagram also called a level 0 data-flow diagram, is drawn in order to define and clarify the boundaries of the software system. It identifies the flows of information between the system and external entities. As shown above, the user data is processed based on the requirements and stored in the PIM(Product Information Management).

Chapter 5: Detailed Design

This chapter discusses the system designs of the project.

5.1 System design

One of the most important steps in software development is system design. It depicts all of the system's requirements in a graphical format. It serves as a supporting document for database designers and developers.

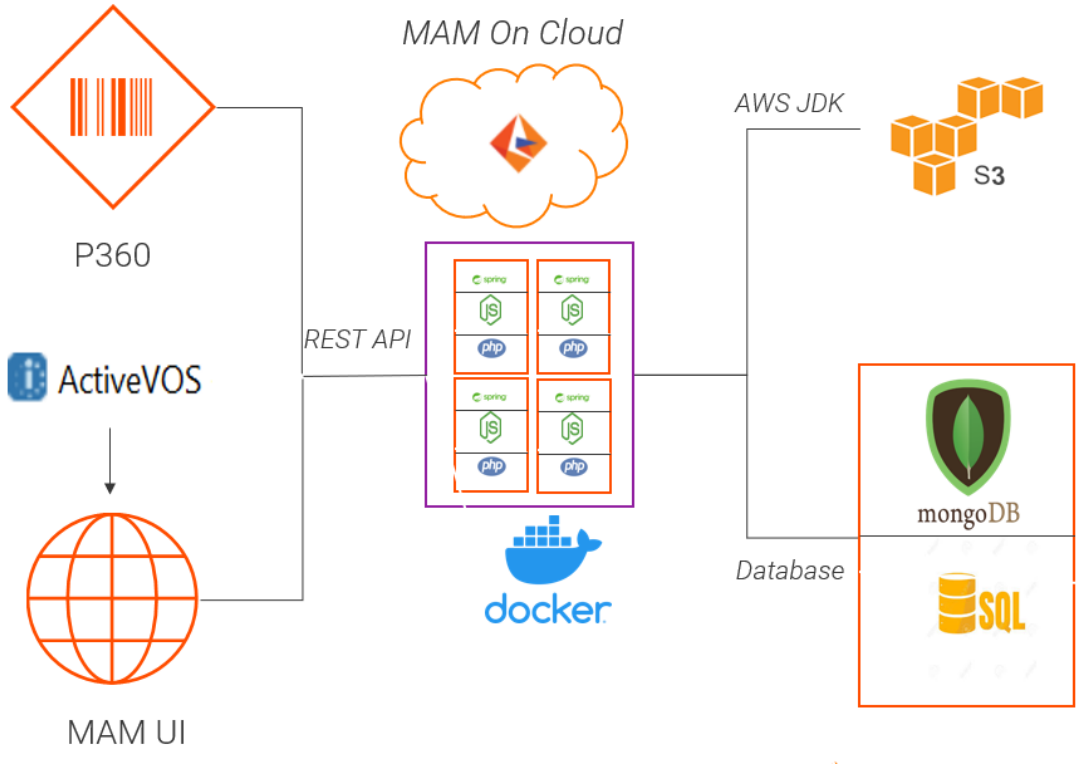


Figure 5.1: Architecture diagram

5.1.1 Dynamic modeling

A dynamic model represents the behavior of an object over time. It is used where the object's behavior is best described as a set of states that occur in a defined sequence.

- **Activity Diagram**

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another.

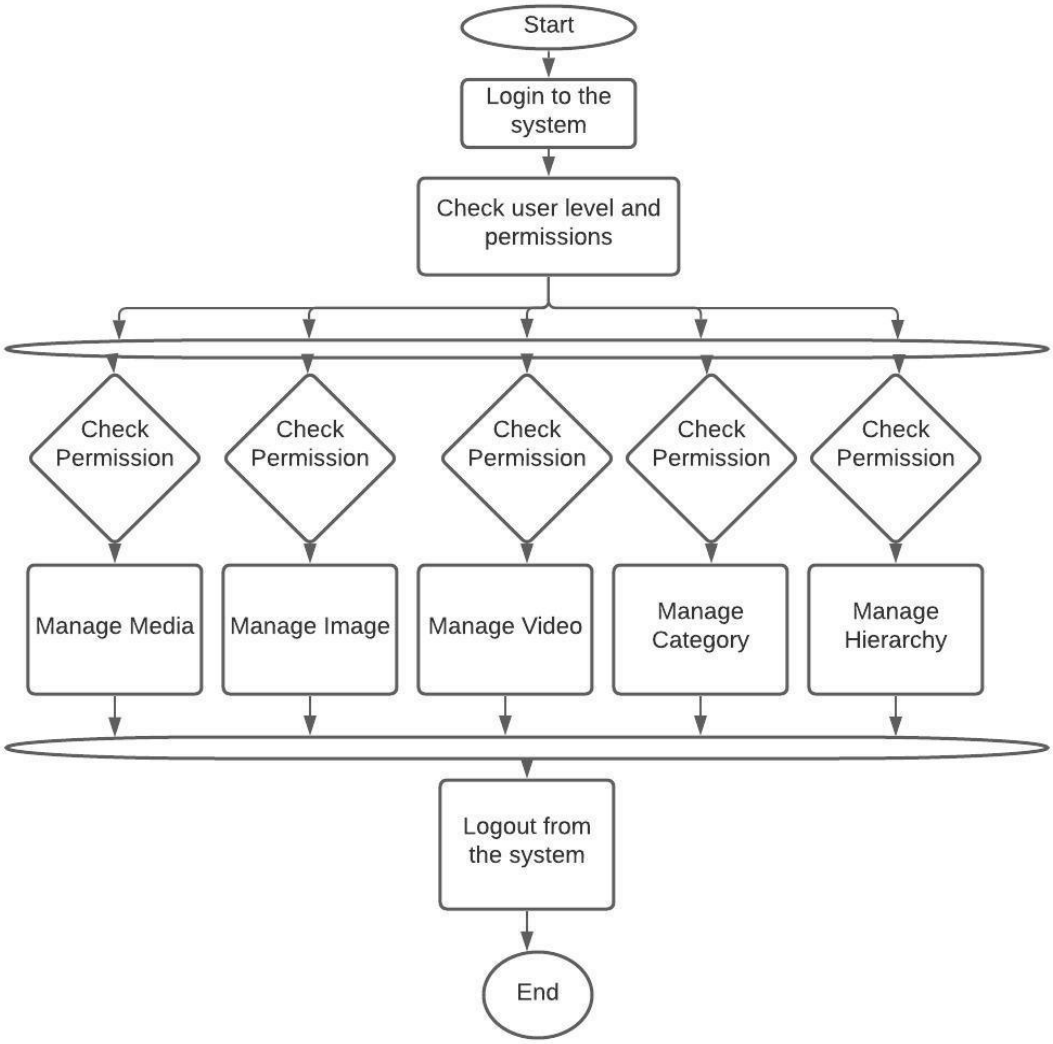


Figure 5.2: Activity diagram

The media management activity starts by logging in to the system and then based user level and permission perform various activities and consequently log out of the system.

• Sequence Diagram

A sequence diagram depicts item interactions in chronological order. It illustrates the scenario's objects and classes, as well as the sequence of messages sent between them in order to carry out the scenario's functionality.

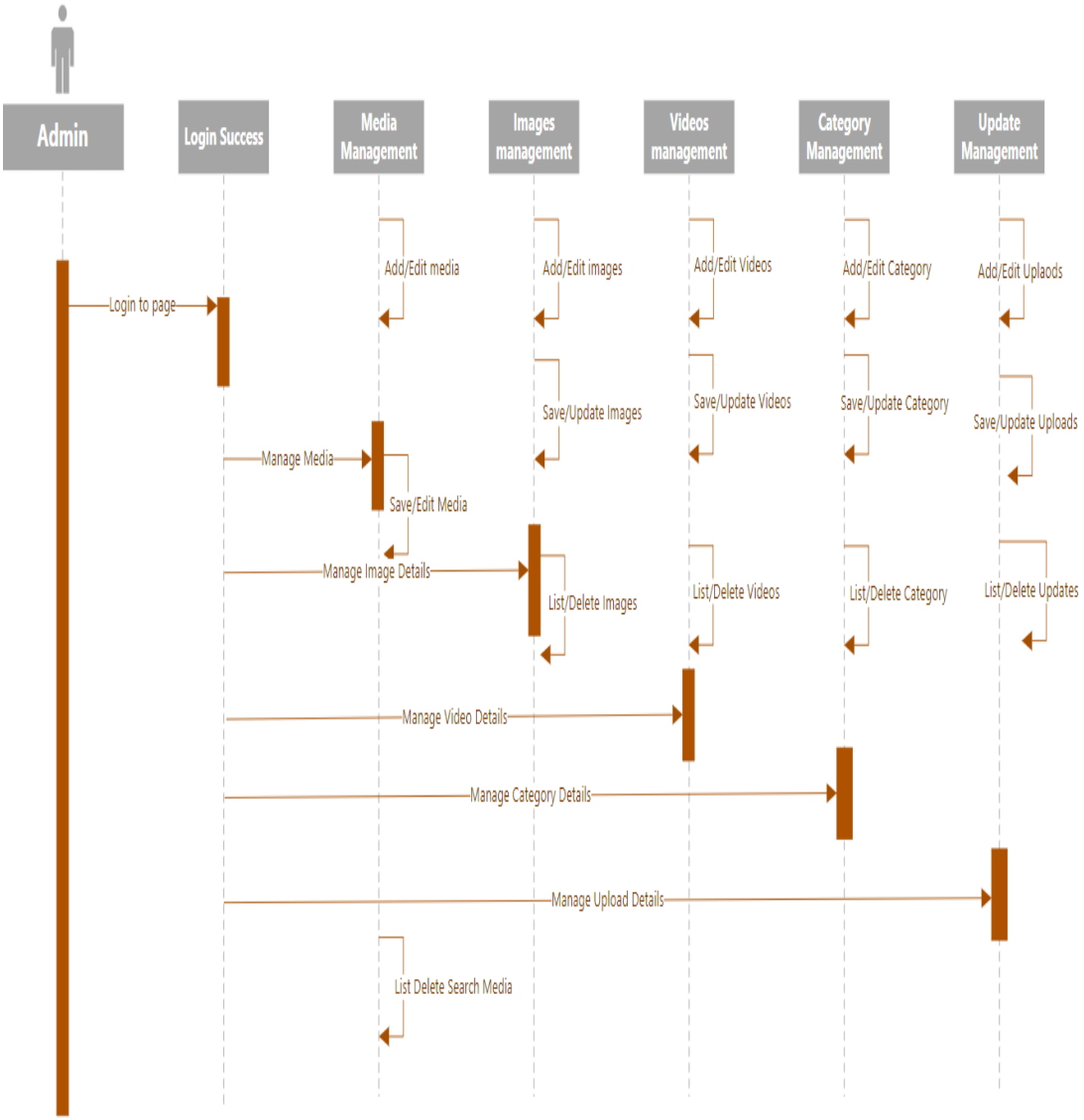


Figure 5.3: Sequence diagram

Shows the sequence diagram where first the admin login to the system and can perform media management wherein he can perform image or video management. The user can also view the hierarchy of images alongwith performing various updation operation.

5.1.2 Functional modeling

The functional model handles the process perspective of the model, corresponding roughly to data flow diagrams. Main concepts are process, data store, data flow, and actors

- Data Flow Diagram

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination

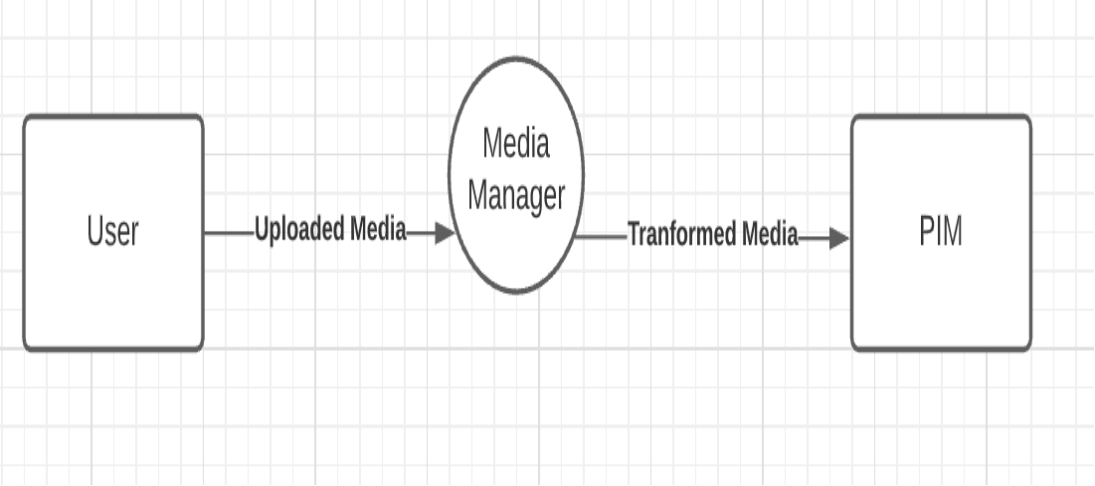


Figure 5.4: Data Flow Diagram Level 0

Context diagram also called a level 0 data-flow diagram, is drawn in order to define and clarify the boundaries of the software system. It identifies the flows of information between the system and external entities. As shown above, the user uploads media which is given as input to the Media Manager and the output is stored in the Product Information Management

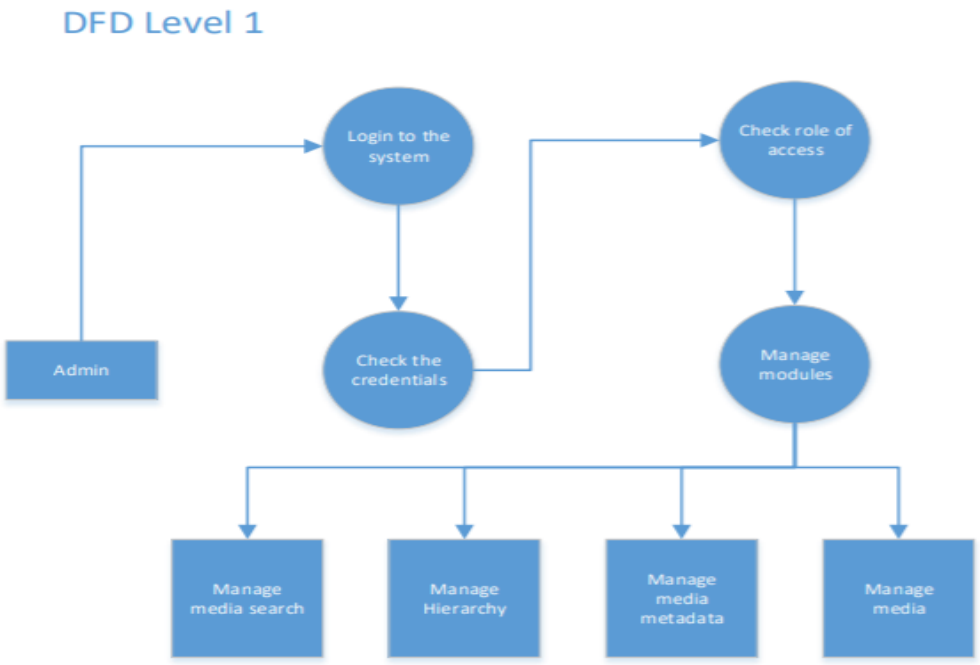


Figure 5.4: Data Flow Diagram Level 1

A level 1 data flow diagram (DFD) is more comprehensive than a level 0 DFD, but less so than a level 2 DFD. It deconstructs the primary processes into subprocesses, which may then be examined and improved on a more detailed level. Shows the data flow from admin to various services like manage media, media metadata, media hierarchy, media search etc.

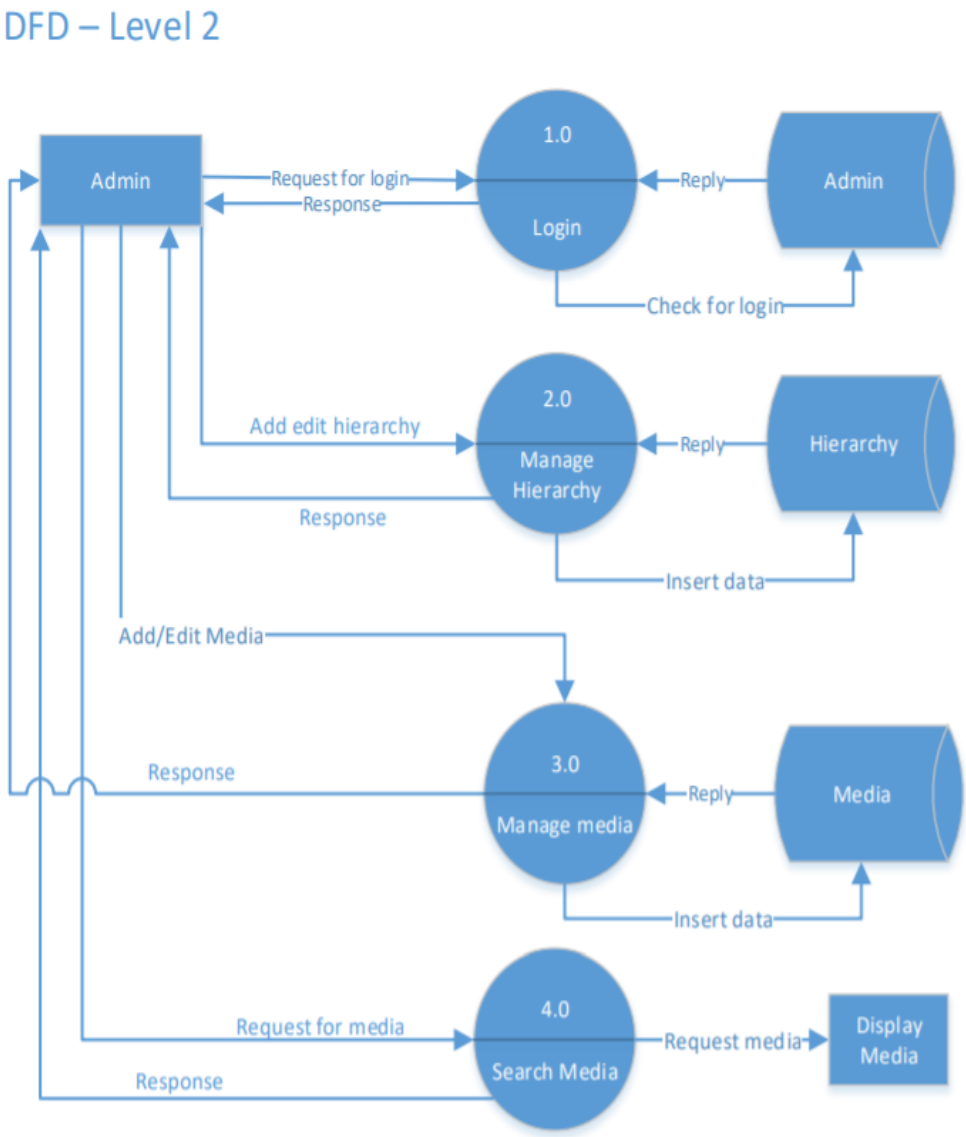


Figure 5.4: Data Flow Diagram Level 2

A level 2 data flow diagram (DFD) provides a more comprehensive view of the components that make up an application software than a level 1 DFD. The admin can login to the system if he has been registered and the required data is stored in the admin database thereby he can perform insert or update to hierarchy database. He can also display the required media based on the search pattern.

In this chapter, different representations of the architectural and system designs of the project were observed

Chapter 6: Implementation

6.1 Code Snippets

```
import React from 'react';
import axios from 'axios';
import './App.css';
import InfaLogo from './infa-logo.svg';
import { MediaModel } from './MediaModel';
import 'react-notifications/lib/notifications.css';
import { NotificationContainer } from 'react-
notifications';
import { NotificationManager } from 'react-
notifications';
import Header  from './Header';
import Sidebar from './Sidebar';
import Button from '@material-ui/core/Button';
import PhotoLibraryIcon from '@material-
ui/icons/PhotoLibrary';
class App extends React.Component {
  constructor(props) {
    super(props);
    this.handleSubmit = this.handleSubmit.bind(this);
    this.fileInput = React.createRef();
    this.handleMedia = this.handleMedia.bind(this);
    this.fetchMedia = this.fetchMedia.bind(this);
    this.state = {
      medias: [],
      selectedMedia: null,
      popoutVisible: false,
      popoutClass: 'app-aside-hidden',
      showVisible:true,
      showClass:'show',
      setVisible: false,
      setClass: 'hide',
      MediaClass:'app-content-hide',
      MediaVisible:false,
      hierarchyCode:'1234-1',
     }
  }
  componentDidMount() {
    axios({
      method: 'get',
      url: 'http://localhost:8080/media'
    })
    .then((response) => {
      let newMedias = this.state.medias;
      response.data._embedded.medias.map((data, i) => {
        return newMedias.push(new MediaModel(data));
      });
```

```
      this.setState( {medias: newMedias });
    });
  }
  closePopout() {
    this.setState({
      'popoutClass': 'app-aside-hidden',
      'popoutVisible': false,
      'showClass':'show',
      'showVisible':true,
    });
  }

  selectMedia(index, event) {

    let media = this.state.medias[index];
    let popoutClass = '';
    let popoutVisible = false;
    let sideClass = '';
    let sideVisible = false;
    let showClass = '';
    let showVisible = true;
    if(!this.state.popoutVisible) {

      popoutClass = 'app-aside-visible';
      popoutVisible = true;
      sideClass = 'side-visible';
      sideVisible = true;
      showClass = 'hide';
      showVisible = false;
      this.setState({
        'popoutClass': popoutClass,
        'popoutVisible': popoutVisible,
        'selectedMedia': media ,
        'showClass':showClass,
        'showVisible':showVisible,
        'sideClass':sideClass,
        'sideVisible':sideVisible,
      });
    }
    else {

      this.setState({
        'selectedMedia': media
      });
    }
  }
  deleteRow(id, e){
    axios.delete(`http://localhost:8080/media/${id}`)
      .then(res => {
        console.log(res);
        console.log(res.data);
```

```
        NotificationManager.success('Media Deleted!','',5
00);
         const medias = this.state.medias.filter(item =>
item.id !== id);
          this.setState({ medias });
      })

  }
  mediaLibrary(e){
    axios({
      method: 'get',
      url: `http://localhost:8080/mediaHierarchy`
      })
      .then((response) => {
      let newHierarchy = this.state.hierarchies;
        response.data._embedded.mediaHierarchies.map((dat
a, i) => {
          console.log('hierarchy name -
'+data.hierarchyName);
          this.setState({
          hierarchyName:this.state.hierarchyName.concat(d
ata.hierarchyName)
        })
          this.setState({
          data:response.data._embedded.mediaHierarchies
        })
           let childs=data.children.map((child)=>{
            console.log('childre name '+child.hierarchyNa
me)
          });
        //  return newHierarchy.push(new HierarchyModel(d
ata));
        });
        this.setState( {hierarchies: newHierarchy });

      });
  }
  handleAttrChange(id, event) {
    axios({
      method: 'put',
      url: `http://localhost:8080/media/${id}`,
      data: {
        [event.target.id]: event.target.value
      }
    })
    .then((response) => {
      console.log(response)
    });
  }
handleSubmit(event) {
     // Prevent the default submit
```

```
    event.preventDefault();
   // Define API constants
    const url = 'http://localhost:8080/media';
    const formData = new FormData();
    const config = {
        headers: {
          'content-type': 'multipart/form-data'
        }
      }
       // Loop through each file and append them to the f
orm
    Object.keys(this.fileInput.current.files).forEach(key
 => {
      formData.append("files", this.fileInput.current.fil
es[key]);
    });
    // Send the files to the backend
    axios.post(url, formData,config).then(response=>{cons
ole.log(response.data);
      window.location.reload();})
    .catch(function (error) {
      console.log(error);
      NotificationManager.success( 'File Upload Failed!')
;
    });
    NotificationManager.success( 'File Upload was Success
ful!');
  }
  handleMedia(e){
    if(this.state.MediaVisible)
    {
    this.setState(state=>({
        MediaVisible:false,
        MediaClass:'app-content-hide'
    }
    ));
      }else if(!this.state.MediaVisible)
        {
            this.setState(state=>({
            MediaVisible:true,
             MediaClass:'app-content-menu'
        }));
        }
  }
  fetchMedia(hierarchyCode,e) {
    axios({
      method: 'get',
      url: 'http://localhost:8080/media'
    })
    .then((response) => {
      this.setState(state=>({
```

```
        medias:[]}));
      let newMedias = this.state.medias;
      response.data._embedded.medias.map((data, i) => {
        if(data.hierarchyCode===hierarchyCode)
        return newMedias.push(new MediaModel(data));
      });
      this.setState( {medias: newMedias });
    });
  }
  render() {
    const hierarchyCode=this.state.hierarchyCode;
    return (
     <div className="App">
        <Header/>
        <div className='app__page'>
        <Sidebar className="this.state.side-
class" action={ this.handleMedia }  mediaAction={ this.fe
tchMedia }  />
        <div className={ this.state.showClass }>
        <section className="app-content">
          <div className={ this.state.MediaClass }>
            <center>
            <h1>Media</h1>
            <br/><br/>
            <form onSubmit={this.handleSubmit}>
            <Button onClick={() => this.fileInput.click()
}>
            <PhotoLibraryIcon/>
            </Button>
            <input type="file"ref={(fileInput) => {
                    this.fileInput = fileInput;
                  }} multiple style={{ display: 'none'}}
 />

            <Button variant="contained" type="submit">Sub
mit</Button>
            </form>
            </center>
          </div>
          <div className="app-content-outer">
            <div className="app-content-inner">
              {
                  this.state.medias.map((media, index) =>
 {
                    return <img key={ media.id } src={ me
dia.url} alt={ media.fileName } className="grid-
image" onClick={ (event) => this.selectMedia(index, event
) } />
                  })
              }
            </div>
```

```
          </div>
          </section>
        </div>
        </div>
          <aside className={ this.state.popoutClass }>
          <span className="closeButton" onClick={ () => t
his.closePopout() }>&times;</span>
          <form className="popForm">
          <div>

              <img className="popImage"  src={ this.state
.selectedMedia != null ? this.state.selectedMedia.url : '
' }/>
              <div id='div1'>
            <Button className="deleteMedia" variant="cont
ained"  onClick={(e) => this.deleteRow(this.state.selecte
dMedia.id, e)}>Delete</Button>
 </div>
          <div id='div2'>
          <Button className="downloadMedia" variant="cont
ained"   href={ this.state.selectedMedia != null ? this.s
tate.selectedMedia.url : '' } download>Download</Button>
          </div>
            </div>
            <div>
              <label className="popLabel">ID:</label>
              <label>{ this.state.selectedMedia != null ?
 this.state.selectedMedia.id : '' }</label>
            </div>
            <div>
              <label className="popLabel">File Name:</lab
el>
              <label>{ this.state.selectedMedia != null ?
 this.state.selectedMedia.fileName : '' }</label>
            </div>
            <div>
              <label className="popLabel">File Extension:
</label>
              <label>{ this.state.selectedMedia != null ?
 this.state.selectedMedia.fileExtension : '' }</label>
            </div>
            <div>
              <label className="popLabel">File Encoding:<
/label>
              <input className="popInput" type="text" id=
"fileEncoding" name="popFileEncoding" defaultValue={ this
.state.selectedMedia != null ? this.state.selectedMedia.f
ileEncoding : '' } onChange={ (event) => this.handleAttrC
hange(this.state.selectedMedia.id, event) }/>
            </div>
            <div>
```

```
              <label className="popLabel">File Size:</lab
el>
              <input className="popInput" type="text" id=
"fileSize" name="popFileSize" defaultValue={ this.state.s
electedMedia != null ? this.state.selectedMedia.fileSize
: '' } onChange={ (event) => this.handleAttrChange(this.s
tate.selectedMedia.id, event) }/>
            </div>
            <div>
              <label className="popLabel">Mime Type:</lab
el>
              <input className="popInput" type="text" id=
"mimeType" name="popMimeType" defaultValue={ this.state.s
electedMedia != null ? this.state.selectedMedia.mimeType
: '' } onChange={ (event) => this.handleAttrChange(this.s
tate.selectedMedia.id, event) }/>
            </div>
            <div>
              <label className="popLabel">URL:</label>
              <label>{ this.state.selectedMedia != null ?
 this.state.selectedMedia.url : '' }</label>
            </div>
            <div>
              <label for="hierarchy" className="popLabel"
>Hierarchy Code:</label>
              <input className="popInput" type="text" id=
"hierarchyCode" name="popHierarchyCode" list="hierarchyLi
st" defaultValue={ this.state.selectedMedia != null ? thi
s.state.selectedMedia.hierarchyCode : '' } onChange={ (ev
ent) => this.handleAttrChange(this.state.selectedMedia.id
, event) }/>
                          <datalist id="hierarchyList">
                        <option value="1234-
1">Product Images</option>
                        <option value="1234-
2">Item Images</option>
                        <option value="1235-
1">SDS</option>
                        <option value="1235-
2">Reports</option>
              </datalist>
              </div>
          </form>
        </aside>
        <NotificationContainer />
      </div>
    ); // End return
  } // End render()

} // End class App
export default App;
```

6.2 Implementation

Home Module

Input : Loading the url – a https url should be loaded which will be bind with a domain server.

Process : To display all the available media to the user

Output : All the media added by the admin are displayed on page fetched from the database

PDL of Home Module

PROCEDURE HOME

Display all the media from the media table with view metadata feature.

DO FOR each product

print (media list with view metadata feature)

ENDDO

Figure 6.1: Home Page

Media Module

Input : Click on any media in home page

Process : To display all the details and features of the products to the user

Output : Details of selected product is displayed on page with download, delete and update option – details will be loaded from the database

PDL of Media Module:

PROCEDURE MEDIA_DETAILS

Display the metadata of the selected media with download, delete and update feature.

IF #mediaId is valid THEN

print (metadata details along with download, delete and update)

ELSE print (error message)

ENDIF

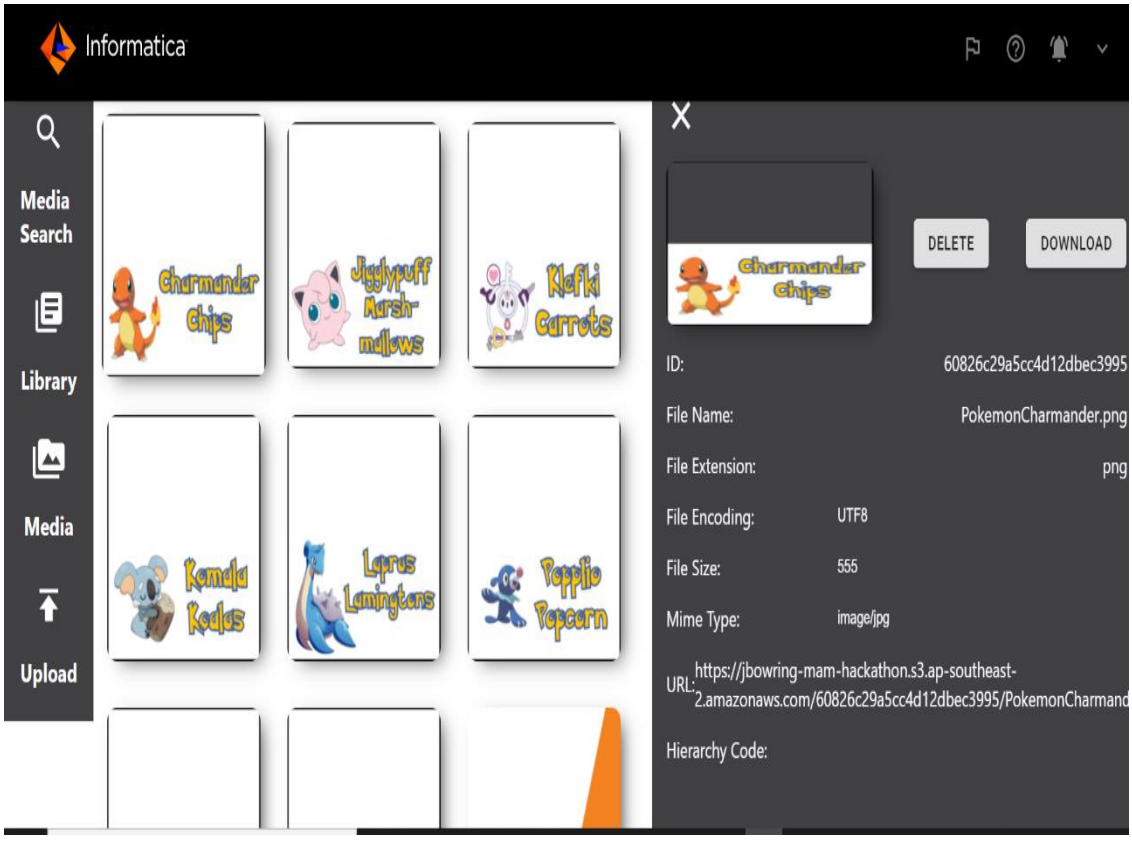


Figure 6.2: Media Page

Media Management Module

Input : Click on delete or download button

Process : To delete or download a media file

Output : The media file is deleted or downloaded

PDL of Media Management Module:

PROCEDURE Media_Management

Download or delete particular media

IF #mediaId is valid THEN

print (download or detail that mediaId)

ELSE print (error message)

ENDIF

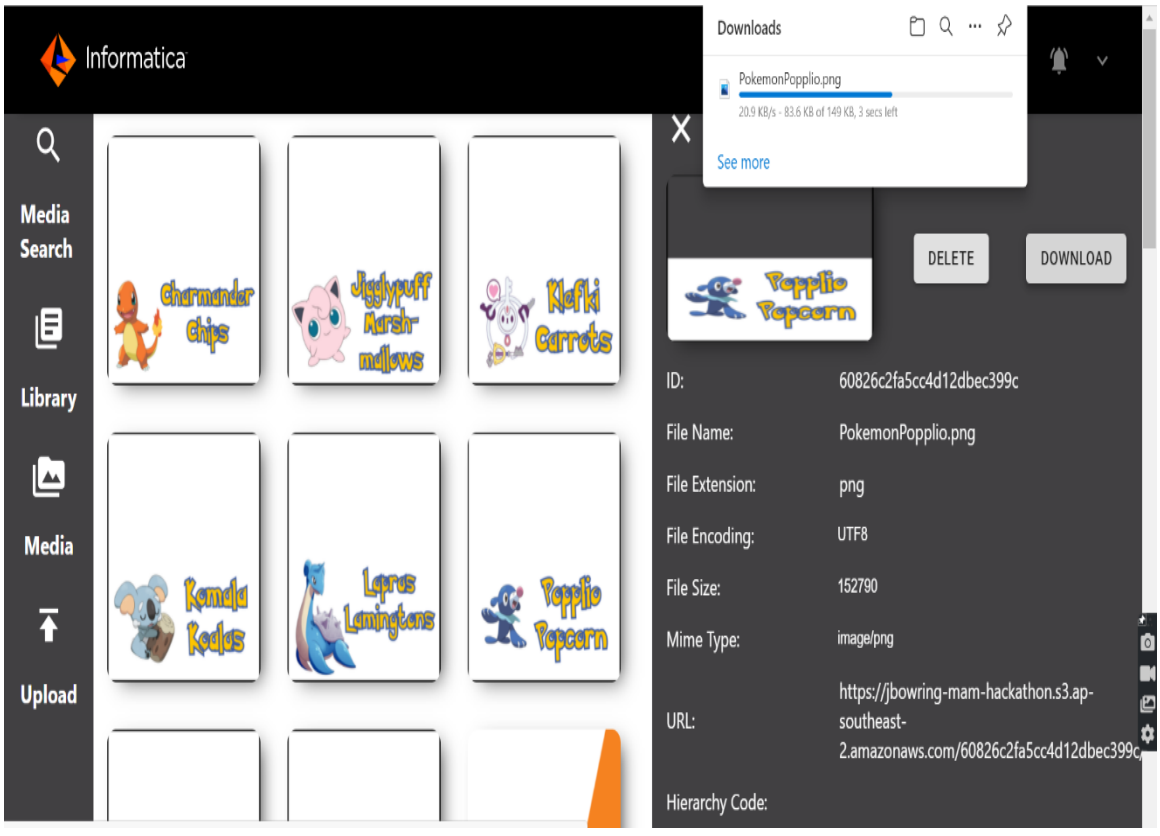


Figure 6.3: Media Management Page

Hierarchy Module

Input : Click on hierarchy name in the library menu

Process: To display all media related to that particular hierarchy

Output: Display filtered media based on hierarchy code

PDL of Hierarchy Module:

PROCEDURE HIERARCHY

Display all the media with particular hierarchy code.

DO FOR each product

IF #mediaId is valid And #hierarchycode=hierarchycode required THEN

print (media list)

ELSE print (error message)

ENDDO

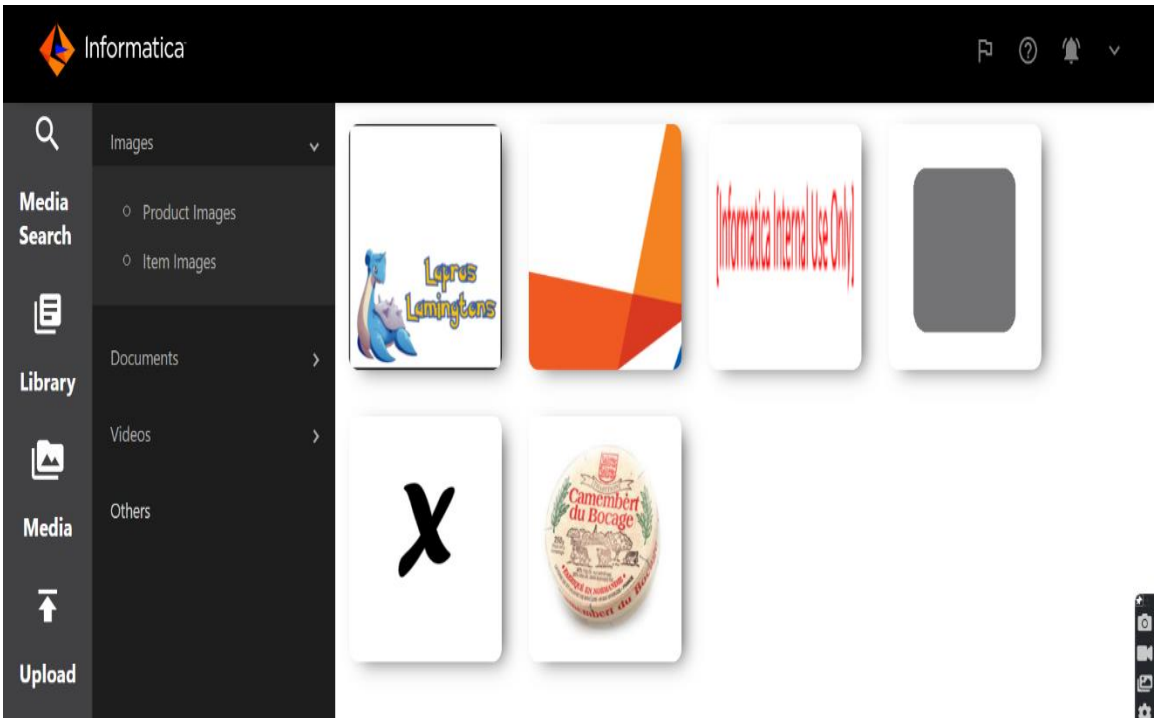


Figure 6.4: Menu containing hierarchy

Media Search Module

Input: Key or File name for filter

Process: To display all media having similar key or filename

Output: Display filtered media based on key or filename

PDL of Media Search Module:

PROCEDURE HIERARCHY

Display all the media with similar key or filename.

DO FOR each product

IF #mediaId is valid And #key or #filename is like filter pattern required THEN

print (media list)

ELSE print (error message)

ENDDO

Figure 6.5: Search Page

Chapter 7: Software Testing

7.1 Test cases

Table 7.1: Test Cases for Home Screen

TC_ID	Description	Sample Input	Expected and Actual Output	Test Result	Remarks
TC_01	Verify whether homescreen is loaded when url is entered	https url	Homescreen should be loaded properly	PASS	Test case passed successfully
TC_02	Verify whether all the navigations present in homescreen are working	Click on any link	Once any link is clicked it should be redirected to that page	PASS	Test case passed successfully
TC_03	Verify whether all the media are displayed successfully	Load the https url	All media must be displayed in the homescreen	PASS	Test case passed successfully
TC_04	Verify whether each media detail is displayed successfully	Load the https url	Each media detail should be properly displayed	PASS	Test case passed successfully
TC_05	Verify whether media metadata is displayed on clicking the media	Click on the media	Media metadata page should open show individual media details	PASS	Test case passed successfully
TC_05	Verify whether media is uploaded	Click on media upload	Media upload Failed	FAIL	Test case failed

Table 7.2: Test Cases for Media Metadata Page

TC_ID	Description	Sample Input	Expected and Actual Output	Test Result	Remarks
TC_01	Verify whether selected media metadata detail is displayed properly	Click on any media in homepage	Alll the details of selected media should be displayed properly	PASS	Test case passed successfully
TC_02	Verify whether on click of delete button the item is deleted	Click on delete button	The media should be deleted when delete button is clicked	PASS	Test case passed successfully
TC_03	Verify whether on click of download button the item is downloaded	Click on download button	The media should be downloaded when download button is clicked	PASS	Test case passed successfully
TC_04	Verify whether filtered media is displayed when we click on particular hierarchy name	Click on hierarchy name	All the media related to that particular hierarchy name is displayed	PASS	Test case passed successfully
TC_05	Verify whether filtered media search based on key or filename is displayed	Type filter text in searchbar	All the media with similar key or filename should be displayed	PASS	Test case passed successfully
TC_06	Verify whether can accept special character	Type filter text in search bar	Does not display required media	FAIL	Test case failed

7.2 Testing and Validations

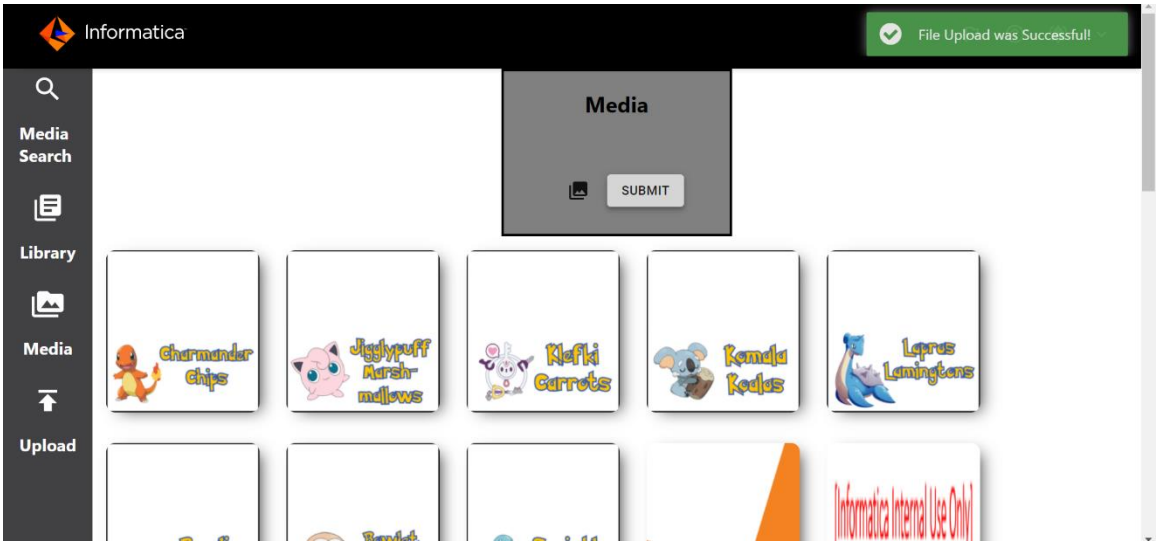


Figure 7.1: File upload successful when proper internet connectivity is available

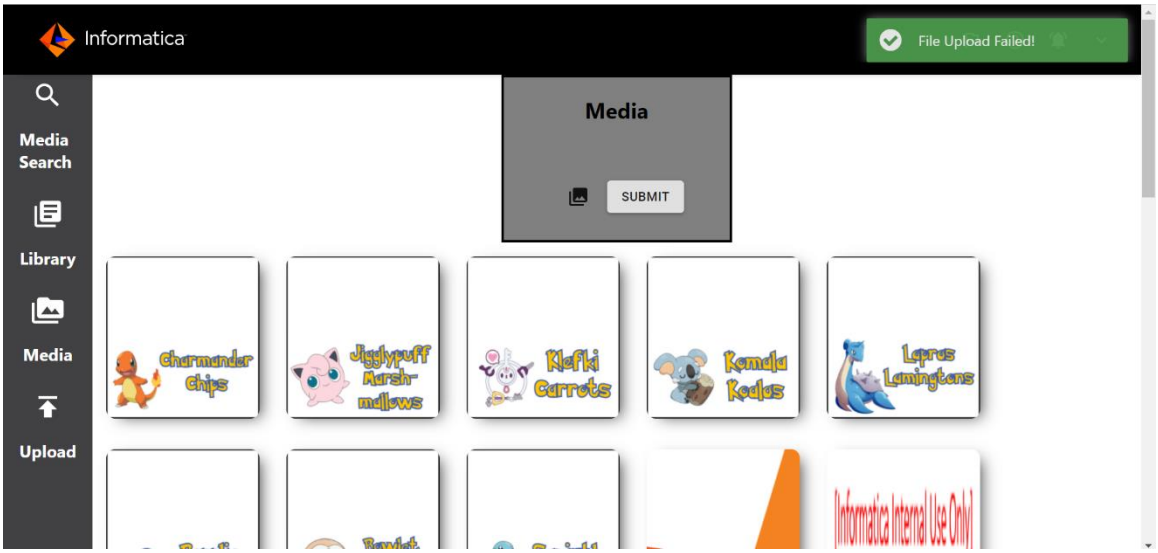


Figure 7.2: File upload failed when there is no proper internet connectivity

Chapter 8: Conclusion

Product 360 offers a mature platform that can handle large amounts of data and complex scenarios with billions of attributes. It can handle and manage millions of hierarchies, classes, goods, objects, and variants without sacrificing performance. Improve the efficiency of operations and workflows. Improve the efficiency of operations and workflows. Product 360 enables us to handle complicated product data in the formats offered by suppliers, such as photos, drawings, documents, audio files, and videos, with ease and efficiency. A fully integrated digital asset management (DAM) system can handle large collections of media assets from a central location. Embedded digital asset management with fully automated data processing capabilities to centrally manage complicated product data and massive collections of media assets in any format.

Media management is one of Product 360's services, and it relates to the online management of multimedia assets. The user can do a search for certain media files, upload or download them, and allocate them to specific products or commodities. Additional material, such as photographs or text documents, can be found in catalogue items in addition to raw data. The key challenges faced while implementing MAM solution using Informatica Media Manager are Complicated UI- difficult to navigate and Multi module complex application which are trying to overcome using our present solution

Chapter 9: Future Enhancement

The MAM is a complex application and the development would be carried out in various phases.The existing service is developed using standalone apps like Springboot, PHP or Javascript. Subsequently we can use docker swarm in order to improve the performance through scaling. Performing various Media Quality Checks is also a very integral part and would be carried out in the next phases. Application authorization and authentication would be carried out by various non-SSO/LDAP mechanism. Performing media check assignment to item or product before deletion. Change the look and feel of the UI by adopting a Crystal clear theme for MAM WEB UI. Hotfolder upload for Medias to carry out mass upload are some the few changes would be done in the future releases.

Bibliography

[1] Master Data Management Market by Component (Solutions & Services), Data Type, Deployment Type (Cloud & On-Premises), Organization Size (SMEs & Large Enterprises), Vertical (BFSI, Retail, Manufacturing, and Healthcare), and Region - Global Forecast to 2025, June 2020

[2] ReactJS, "React Guide" [Online], Available: https://facebook.github.io/react/docs/thinking-in-react.html

[3] Mishra, Abhishek. (2019). Amazon S3. 10.1002/9781119556749.ch9.

[4] Madhuri A. Jadhav et al, "Single Page Application using AngularJS", International Journal of Computer Science and Information Technologies, Vol. 6 (3) , 2015, 2876-2879

[5] AngularJS, "Angular Guide" [Online], Available: https://docs.angularjs.org/guide

[6] Github, "virtual DOM" [Online], Available: https://github.com/Matt-Esch/virtual-dom

[7] "The Future of Sharing on Facebook, Twitter and Google Plus," Mashable, Feb. 15, 2012, http://mashable.com/2012/02/15/future-of-sharing-facebook-twitter-google/.

[8] Shea Bennett, "How Social is B2B?" AllTwitter, March 28, 2012, http://www.mediabistro.com/alltwitter/b2b-social-marketing_b20019.

[9] Emil Protalinski, "Facebook Has Over 901 Million Users, Over 488 Million Mobile Users," ZDNet, April 23, 2012, http://www.zdnet.com/blog/facebook/facebook-has-over-901-million-users-over-488-million-mobile-users/12105.

[10] "Gartner Highlights Three Trends That Will Shape the Master Data Management Market," Gartner, Inc., May 4, 2011, http://www.gartner.com/it/page.jsp?id=1666414.

[11] Das, Tapan & Mishra, Manas. (2011). A Study on Challenges and Opportunities in Master Data Management. International Journal of Database Management Systems. 3. 10.5121/ijdms.2011.3209.

[12] Beth Schultz, "Social Media Data and Your Single Source of Truth," All Analytics, August 2, 2011, http://www.allanalytics.com/author.asp?section_id=1411&doc_id=231958.

[13] Butler,David., Stackowiak,Bob., "Master Data Management", Oracle Corporation.available at www.oracle.com.

[14] Gartner MDM Summit,(2011),UK International Journal of Database Management Systems (IJDMS), Vol.3, No.2, May 2011 139

[15] Malcolm, Chisholm(Dec 2010),"The Governance Challenge for Master Data Management", Data Governance Conference, Orlando, Florida

[16] "MDM Fundamentals and Best practices",www.elearningcurve.com

[17] Rittman,Mark."Introduction to Master Data Management". www.rittmanmead.com

[18] Schumachar, Scott (Oct 2010),"MDM: Realizing the same benefits through different implementations", www.initiatesystems.com

[19] Schönemann, N., Fischbach, K. Schoder, D. "P2P Architecture for Ubiquitous Supply Chain Systems", 17th European Conference on Information Systems (ECIS), Verona, Italy, 2009

[20] Breuer, T. (2009), "Data quality is everyone's business – designing quality into your datawarehouse – part 1", Journal of Direct, Data and Digital Marketing Practice, Vol. 11,pp. 20-9.

[21] Cappiello, C. Comuzzi, M. "A utility-based model to define the optimal data quality level in IT service offerings", 17th european conference on Information Systems (ECIS), 2009

[22] Loshin, D. (2009), "Master Data Management", Morgan Kaufmann, Burlington,MA.

[23] Kokemüller, J., Kett, H., Höß, O. Weisbecker, A. "An Architecture for Peer-to-Peer Integration of Interorganizational Information Systems", 15th Americas Conference on Information Systems (AMCIS), San Francisco, USA, August 6 - 9, 2009

[24] Kokemüller, Jochen & Anette, Weisbecker. (2009).” Master Data Management: Products and Research.” 8-18.

[25] Toronto MDM Summit.(2008),”MDM Challenges and solutions from the real world” available at www.adastracorp.com

[26] Hose, K., Roth, A., Zeitz, A., Sattler, K. Naumann, F. "A research agenda for query processing in large-scale peer data management systems", Information Systems, 2008

[27] Kokemüller, J., Kett, H., Höß, O. Weisbecker, A. "A Mobile Support System for Collaborative Multi-Vendor Sales Processes", Proceedings of the Fourteenth Americas Conference on Information Systems, Toronto, ON, Canada, August 14th-17th, 2008

[28] Schemm, J. W., Legner, C. Österle, H. , Global Data Synchronization — Lösungsansatz für das überbetriebliche Produktstammdatenmanagement zwischen Konsumgüterindustrie und Handel, PhysicaVerlag HD, 2008

[29] Eschinger, C. , Report Highlight for Market Trends: Master Data Management Growing Despite Worldwide Economic Gloom, 2007-2012, Gartner, 2008

[30] Rachuri, S., Subrahmanian, E., Bouras, A., Fenves, S., Foufou, S. and Sriram, R. (2008),“Information sharing and exchange in the context of product lifecycle management: role ofstandards”, Computer-Aided Design, Vol. 40 No. 7, pp. 789-800.

[31] Berson, A. and Dubov, L. (2007), Master Data Management and Customer Data Integration for aGlobal Enterprise, McGraw-Hill, New York, NY.

[32] Leser, U. Naumann, F. , Informationsintegration, dpunkt.verlag, Heidelberg, 2007

[33] Moss, L. (2007), “Critical success factors for master data management”, Cutter IT Journal, Vol. 20No. 9, pp. 7-12.

[34] Yang, X., Moore, P.R., Wong, C.-B., Pu, J.-S. and Chong, S.K. (2007), "Product lifecycle information acquisition and management for consumer products", Industrial Management & Data Systems, Vol. 107 No. 7, pp. 936-56.

[35] Boyd, M. (2006), "Product information management – forcing the second wave of data quality",available at: ww.thecopywritingpro.com/pages/samples_assets/2nd-waveDQ.pdf(accessed 27 April 2010).

[36] Wolter,Roger.,Haselden,Kirk.(2006),A White paper on MDM, Microsoft Corporation.

[37] Haas, L. "Beauty and the Beast: The Theory and Practice of Information Integration", Database Theory – ICDT 2007, 2006, pp. 28-43

[38] White, A., Newman, D., Logan, D. and Radcliffe, J. (2006), "Mastering master datamanagement", available at: http://kona.kontera.com/IMAGE_DIR/pdf/MDM_gar060125_MasteringMDMB.pdf (accessed 12 April 2010).

[39] Dreibelbis,Allen , Hechler,Eberhard, Milman ,Ivan(2009), "Enterprise Master Data Management",Pearson Education.

[40] Dayton, M. (2007), "Strategic MDM: the foundation of enterprise performance management",Cutter IT Journal, Vol. 20 No. 9, pp. 13-17.

[41] Elmagarmid, A. K., Ipeirotis, P. G. Verykios, V. S. "Duplicate Record Detection: A Survey", IEEE Transactions on Knowledge and Data Engineering, 19, 1, Los Alamitos, CA, USA, 2007, pp. 1-16

[42] Russon,Philip. (2006), Master Data Management – TDWI Best practice Report a vailable at www.tdwi.org.

[43] Knolmayer, G. and Ro¨thlin, M. (2006), "Quality of material master data and its effect on theusefulness of distributed ERP systems", Lecture Notes in Computer Science, Vol. 4231,pp. 362-71.

[44] Lee, Y.W., Pipino, L.L., Funk, J.D. and Wang, R.Y. (2006)," Journey to Data Quality", MIT Press, Cambridge, MA.

[45] Walter, P., Werth, D. Loos, P. "Peer-to-Peer-Based Model-Management for Cross-Organizational Business Processes", Los Alamitos, CA, USA, June, 2006, pp. 255-260

[46] Dumas, M., Aalst, W. and Ter Hofstede, A. (2005), Process-aware Information Systems: BridgingPeople and Software Through Process Technology, Wiley, Hoboken, NJ.Managing onemaster data161

[47] Chen, Z., Kalashnikov, D. V. Mehrotra, S. "Exploiting relationships for object consolidation", IQIS '05: Proceedings of the 2nd international workshop on Information quality in information systems, New York, NY, USA, 2005, pp. 47-58

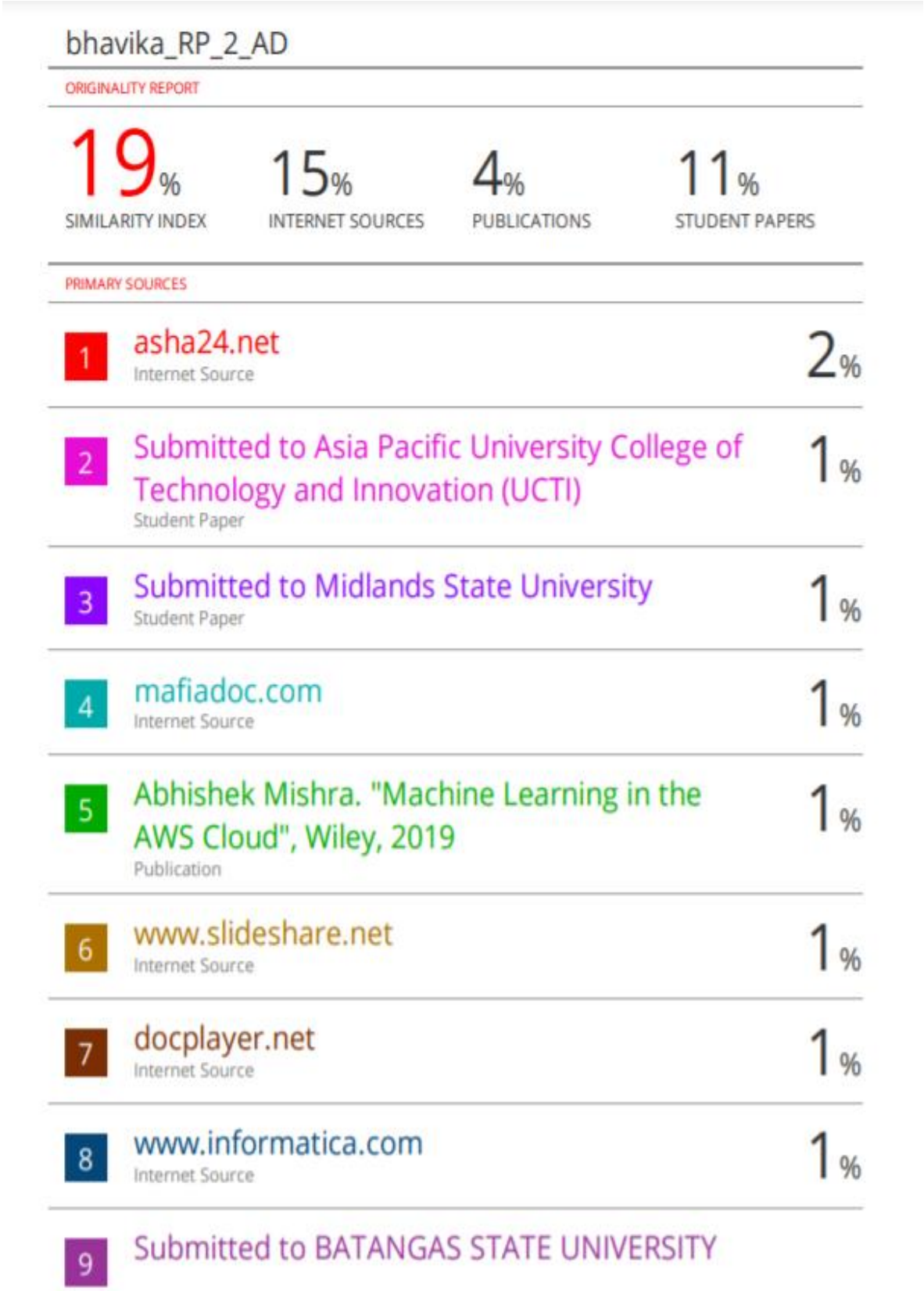
[48] Androutsellis-Theotokis, S. Spinellis, D. , “A Survey of Peer-to-Peer Content Distribution Technologies”, 2004

[49] Erasala, N., Yen, D. C. Rajkumar, T. M. "Enterprise Application Integration in the electronic commerce world", Computer Standards und Interfaces, 25, 2, 2003, pp. 69 – 82

[50] Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P. Fienberg, S. "Adaptive name matching in information integration", Intelligent Systems, IEEE, 18, 5, Sep/Oct, 2003, pp. 16-23

[51] Rahm, E. Do, H. H. "Data Cleaning: Problems and Current Approaches", IEEE Data Engineering Bulletin, 23, 4, 2000, pp. 3-13

Plagiarism Report



bhavika_RP_2_AD

ORIGINALITY REPORT

19%	15%	4%	11%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	asha24.net Internet Source	2%
2	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	1%
3	Submitted to Midlands State University Student Paper	1%
4	mafiadoc.com Internet Source	1%
5	Abhishek Mishra. "Machine Learning in the AWS Cloud", Wiley, 2019 Publication	1%
6	www.slideshare.net Internet Source	1%
7	docplayer.net Internet Source	1%
8	www.informatica.com Internet Source	1%
9	Submitted to BATANGAS STATE UNIVERSITY	

Certification of Publication

Manuscript ID: PNS-0621-114

Title of Article: Migration from Monolithic to Microservice architecture using Open Source tools

Publication: Pensee International Journal

Source type: Journal

Scopus coverage years:from 2006 to Present

Publishing Volume/Issue: Volume 51 Issue 6-2021

Status: Published

Technical Paper

PONSEE ISSN 2321-477

Migrating from Monolithic to Microservice Architecture using Open Source tools

[1]Bhavika Vasandani, [2]Dr. Andhe Dharani

Department of Master of Computer Applications
RV College of Engineering, Bengaluru, India

Abstract: *Organizations typically would design a monolithic architecture wherein we create an application that has a single code base with many modules. Modules can be divided based on various features like business or technical. Monolithic application can be built as a single and inseparable unit. With the various technological advances, enterprises have understood the necessity to modernize the applications by migrating from a monolith architecture to a microservice architecture so as to modernize a system. These changes have seemed to become very popular within the recent years. While a monolithic application act as a single unified unit, the microservices design separates it down into a group of smaller independent units where every unit carry out separate service performing discrete tasks. Each module can also use a different technology stack. Thus the services have their own function and the database as well as each service carry out the specific functions. This paper presents methodologies to transform the present monolithic architecture to microservice architecture using various microservice tools like Kubernetes and Docker swarm and also comparing these tools.*

Keywords: Monolithic, Microservice, Kubernetes, Docker Swarm

1. Introduction

The microservice architectural style is gaining a lot of traction in the business because of technology advancements. The microservices architectural style, in contrast to the monolithic architectural style, is based on the service-oriented architectural style for designing software made up of small services that can be delivered and scaled independently by fully automated deployment machinery which contains minimum centralized management. Microservices are services that are defined by distinct business functions. Every microservice is in charge of its own operation and communicates with other microservices via lightweight means, such as APIs. Microservices address the problems that monolithic applications was not able to handle. Because they are tiny, they can restart at a faster rate when updates or failure recovery are required. Microservices consists of loosely coupled system so that the failure of one microservice will not affect with the working of other microservices of the system. Thus making the application more scalable, flexible and more efficient since each microservice is able to work at its own pace. Virtualization at the operating system level is aided by containerization. Containers are typically light weight and portable. We may implement containerization using a variety of open source tools. Kubernetes and Docker Swarm are two open source solutions that aid in the containerization process.

2. Literature Work

Traditionally software development and deployment was done using a centralized monolithic architecture where the modules were tightly coupled. Modules can't be extended separately, and they can't use multiple technology stacks. To solve this, we must move away from monolithic architecture and toward microservice architecture. Microservice architecture is the most basic type of service-oriented architecture. The empirical analysis of migration to microservices provides advise on how to