

International Institute of Information Technology, Bangalore

Software Production Engineering Mini Project

Custom-eBooks

Vaibhav Aggarwal



Nomula Rohith Yogi
IMT2016072

Dachepalli Soumith Kumar
IMT2016110

Bukka Nikhil Sai
IMT2016091

Arunaav Reddy
IMT2016108

Table of Contents

Table of Contents	1
Abstract	3
Introduction	3
Importance of the problem	3
Work plan	3
Why DevOps?	4
System Configuration	4
Software Development Life Cycle (SDLC)	5
Source Code Management (SCM) :	5
Build :	6
Maven :	7
Test :	8
Junit	8
Selenium	8
Test Example	9
Archive :	9
Deployment :	10
Continuous Integration - jenkins.	11
Installation Procedure	12
SCM - git.	12
Build - maven.	12
Test -selenium/JUnit.	13
Artifact - Docker.	14
Prerequisites:	14
Deploy - rundeck.	16
Monitor - ELK.	17
Installing Elasticsearch	18
elasticsearch running in localhost:9200	18
Installing Filebeat	18
Installing Logstash	20
Installing Kibana	21
kibana running in localhost:5601	22
Finally in Kibana: Management > Index Pattern	22
Click on Next Step > logstash-* :	23
Go to Discover > logstash-*:	23

Kibana Dashboard :	24
Continuous Integration - jenkins.	25
Experimental Setup	26
Functional Requirements	27
Non - Functional Requirements	27
Architecture and Design	28
Code Walkthrough	28
Servlets	28
Add Book Servlet:	28
Upload servlet :	28
Download Servlet :	29
Get Book Servlet :	29
Search Results Servlet :	30
Remove chapter :	31
Upload ChapterServlet :	31
User Login Servlet :	31
Front end :	32
Results	32
Database - eXist-DB	32
eXist-db running on localhost:8081 :	33
Website - Custom-eBooks	33
Generic page :	33
Publisher page :	34
Book info page :	34
Upload Chapter :	35
Chapter Uploaded successfully :	35
Customer page :	36
Chapter Added :	37
Book Generated :	37
Additional functionalities :	38
Our Team :	41
GitHub :	41
Docker Hub :	41
Future Work and Conclusion	41
References	42

Abstract

Custom Ebooks is a project aimed to provide an application that enables users to compose ebooks as per their preferences. Ebooks give us a lot of flexibility in terms of what their contents can be unlike paperbacks where the content is fixed and we aim to make use of this flexibility and provide users with tools to take advantage of this. We use DevOps tools and approaches to develop the said application and we have summarised the tools we have used, problems we faced and the development process in this report along with details regarding the design and features of our application.

Introduction

Importance of the problem

This project sets out to solve the problem of generating e-books based on preferences associated with the user. More and more people are preferring ebooks to a hard copy. But the trend has been towards treating them like normal books in terms of their contents (fixed content). The idea is that unlike with hard copies, we can have a lot of flexibility with ebooks. One such ability is to customize them the way we want. This is similar to taking selected pages from different hardcover books and binding them together. To be more precise, we can have the ability to create ebooks that are composable with smaller components, and the ability to select and assemble parts of many books to create them. The aim of our project is to create an application to make this possible.

Work plan

The plan is to create a simple application that is easy to use where a user can upload pdf (we refer to them as chapters i.e. each unit is a chapter) and combine them as required to create custom books. The content management framework which handles storing the ebooks and enables search through stored books is available so we plan to build on it to create a frontend for users to utilize these services. The proposed solution includes two interfaces for publishers and customers. The publisher interface will allow

them to upload books. The customer view allows the customer to define a structure to the book, search for various topics and create a custom book. The books will be stored in a native XML database. Representing data using XML conforms to the way books are structured using hierarchies as chapters and sections. Hence, using XML as a primary storage mechanism makes it easier to work with and to avoid mapping. At the time of checkout, we assemble all the different topics that are selected and arrange them in a user-defined order and let the customer define the structure.

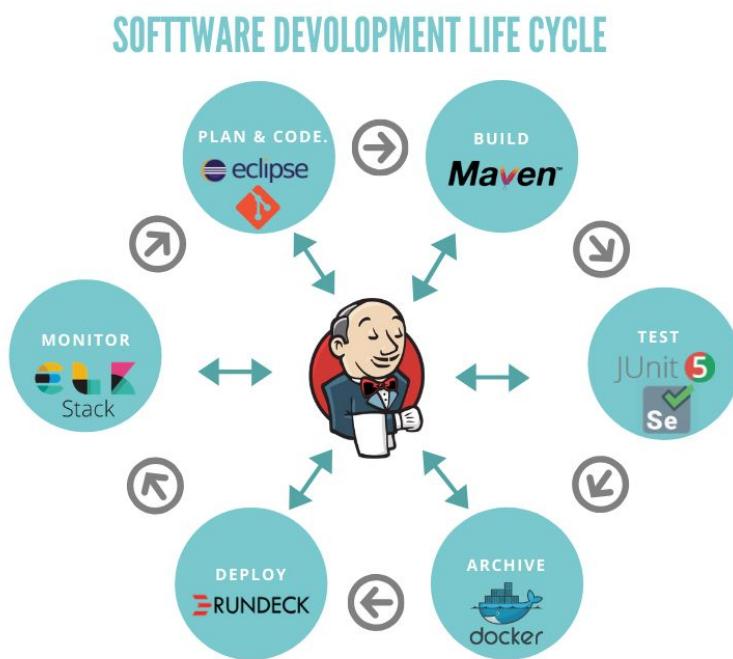
Why DevOps?

We plan to build this project in an incremental manner. The design consists of views and services which function fairly independently. Given the complexity of the project, it's not feasible for one of us to build and test all the code manually every time we make a small change. Also since there are four of us working from different locations, an automated pipeline will not only make our work easier, it also makes it more efficient. The amount of communication that has to happen between us will reduce. DevOps enables us to focus on important aspects of the project, improves efficiency, stability and security. There's less scope for manual errors. Also since we plan to ship this product at some point, continuous delivery makes it easy. Also, monitoring allows us to understand the usage better and helps us improve the application.

System Configuration

NAME	SPECS
OS - Ubuntu	16.04 LTS
CPU	Intel Core I5 (8 cores)
RAM	8GB
Kernel version	Linux 4.15.0-101 generic
Languages used	Java, JSP, HTML, CSS, javascript and XML
Others	eXist-DB and tomcat server

Software Development Life Cycle (SDLC)



Source Code Management (SCM) :

A source code manager is a software tool used by teams of programmers to manage source code. SCMs are used to track revisions in software. Each revision is given a timestamp and includes the name of the person who is responsible for the change. Various revisions may be compared, stored, and merged with other revisions.

We used Git as SCM for this project.

Why Git?

- A distributed version control system.
- Easy to use and merge.
- Easy to track changes in any file of the project.
- It allows us to revert the code files back to their previous state.

The list of commands used for code management are:

```
$ git init  
$ git add -all  
$ git commit -m "commit message"  
$ git push
```

Build :

We used Maven is a build automation tool used primarily for Java projects. It is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting, and documentation from a central piece of information.

Maven uses Convention over Configuration, which means developers are not required to create the build process themselves. Maven project structure and contents are declared in an XML file, pom.xml, referred to as Project Object Model (POM), which is the fundamental unit of the entire Maven system.

We used the following command to build the maven project by skipping the test cases:

```
$ mvn install -DskipTests
```

Build Result:

```
[INFO] [INFO] --- maven-install-plugin:2.5.2:install (default-install) @ CustomEbooks ---  
[INFO] Installing /home/arunav/Desktop/8th-semester/SPE/project/CustomEbook/target/CustomEbooks.war to /home/arunav/.m2/repository/com/Custom-E-Books/web/CustomEbooks/0.0.1-SNAPSHOT/CustomEbooks-0.0.1-SNAPSHOT.war  
[INFO] Installing /home/arunav/Desktop/8th-semester/SPE/project/CustomEbook/pom.xml to /home/arunav/.m2/repository/com/Custom-E-Books/web/CustonEbooks/0.0.1-SNAPSHOT/CustomEbooks-0.0.1-SNAPSHOT.pom  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 02:23 min  
[INFO] Finished at: 2020-05-23T14:42:23+05:30  
[INFO] -----  
arunav@rundeck:~/Desktop/8th-semester/SPE/project/CustomEbook$
```

A war file is created upon the successful build. Now to run the test cases, move the created war file to web apps folder in tomcat then start the tomcat server and run the following command:

```
$ mvn test
```

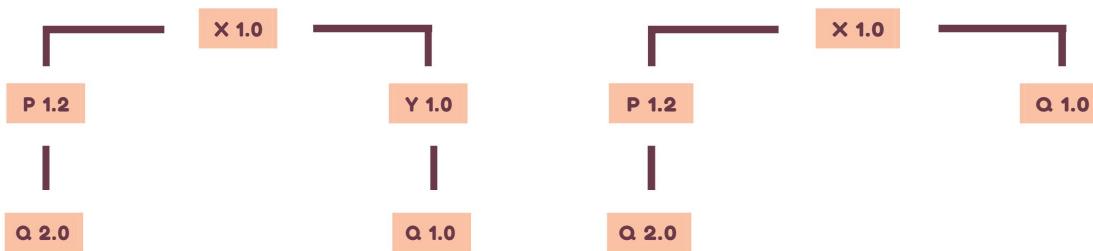
Test Result :

```
[INFO] -----  
[INFO] T E S T S  
[INFO]  
[INFO] Running Tests  
ERROR StatusLogger No Log4j 2 configuration file found. Using default configuration (logging only errors to the console), or user programmatically provided configurations. Set system property 'log4j2.debug' to show Log4j 2 internal initialization logging. See https://logging.apache.org/log4j/2.x/manual/configuration.html for instructions on how to configure Log4j 2  
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 3.994 s - in Tests  
[INFO]  
[INFO] Results:  
[INFO]  
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0  
[INFO]  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO]  
[INFO] Total time: 12.121 s  
[INFO] Finished at: 2020-05-23T15:09:42+05:30  
[INFO]  
[INFO] -----  
orunav@rundeck:~/Desktop/8th-semester/SPE/project/CustomEbook$ []
```

Issues faced while building our project:

Dependency conflict :

Dependencies can be linked to other dependencies with different versions that can produce conflicts. Suppose the dependency tree scheme for our project X looks something like this:



From the tree scheme, our project X will use all the libraries (Y, P, and Q) even if we didn't specify them explicitly in the POM. In this case, maven dependency mechanism imports the library's version whose node is nearest to the root (project X) in the dependency tree. However, if there are several versions of the same library whose nodes are on the same level in the tree, the first library version found is used.

To tackle this conflict, we can either add the latest version of the dependency directly to the POM file or we can exclude the specific dependency using the following command:

```
<!-- https://mvnrepository.com/artifact/xalan/serializer -->
<dependency>
    <groupId>xalan</groupId>
    <artifactId>serializer</artifactId>
    <version>2.7.2</version>

    <exclusions>
        <exclusion>
            <groupId>xerces</groupId>
            <artifactId>xercesImpl</artifactId>
        </exclusion>
        <exclusion>
            <groupId>xml-apis</groupId>
            <artifactId>xml-apis</artifactId>
        </exclusion>
    </exclusions>

</dependency>
```

Test :

Testing is done using JUnit and selenium.

Junit

It is an open-source framework, which is used for writing and running tests. Provides annotations to identify test methods, assertions for testing expected results, etc.

Selenium

It is an open-source tool that is used for automating the tests carried out on web browsers (Web applications are tested using any web browser).

Test Example

```
@Before
public void setUp() {
    // Create a new instance of the html unit driver
    driver = new HtmlUnitDriver();

    //Navigate to desired web page
    driver.get("http://localhost:8080/Custom-eBooks");
}

@Before
public void cleanUp() {
    driver.close();
}

@Test
public void WebsiteUpTest() {

    String expectedTitle = "Custom-eBooks";

    String actualTitle = driver.getTitle();
    Assert.assertEquals(expectedTitle, actualTitle);

}

@Test
public void ConsumerTest() {
    driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
    String expectedTitle = "Consumer Page";

    WebElement element = driver.findElement(By.id("consumer"));
    element.click();
}
```

Archive :

Archiving is the process by which inactive information, in any format, is securely stored for long periods of time. A war file is generated after the project code successfully builds and passes all the test cases.

Then a docker image is built which contains a tomcat server and the war file generated from building the project will be copied to the same image. A dockerfile is written to perform this task, It contains the following commands.

```
FROM tomcat:8.5.55-jdk8-openjdk
COPY ./target/CustomEbooks.war /usr/local/tomcat/webapps/
```

We create a docker image and push this image to the docker hub repository. The image contains the latest version of the source code which can be built successfully to see all the features that have been implemented so far.

Deployment :

We have used Rundeck for the deployment of the application. It is an open-source software operation management platform.

Rundeck provides a number of features that will alleviate time-consuming grunt work and make it easy for you to scale up your automation efforts and create self-service for others.

Rundeck allows us to run tasks on any number of nodes from a web-based interface.

The screenshot shows the Rundeck web interface with a completed job titled "Customebooks Web App and Database Update". The job status is "Succeeded" at 0.03.31 at 8:59 pm. The job ID is #111. The job details table shows 100% completion with 1/1 steps completed. The steps listed are:

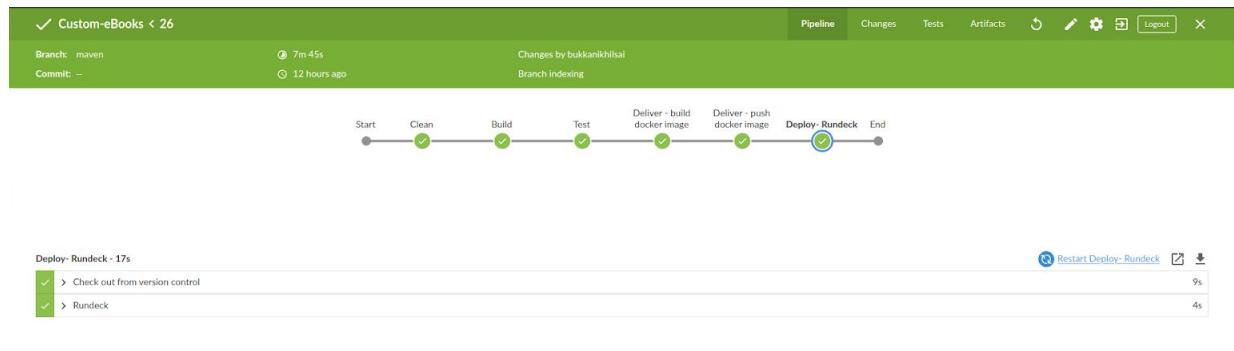
Step	Status	Start Time	Duration
All Steps OK	OK	8:56:25 pm	0.00:43
Stop existing CustomEbooks App Container	OK	8:56:32 pm	0.00:06
Stop existing ExistDB Database Container	OK	8:56:40 pm	0.00:07
Removing existing CustomEbooks App Container	OK	8:56:43 pm	0.00:01
Removing existing Existdb Database Container	OK	8:56:44 pm	0.00:05
Pulling Latest CustomEbooks WebApp docker image	OK	8:56:50 pm	0.00:03
Pulling Latest ExistDB Database docker image	OK	8:56:53 pm	0.00:08
Starting ExistDB Database Container	OK	8:57:01 pm	0.00:07
Starting CustomEbooks WebApp Container	OK	8:57:08 pm	0.00:21
Checking CustomEbooks WebApp is Up	OK	8:57:30 pm	0.02:24
Setting the initial state of ExistDB Database	OK	8:59:54 pm	0.00:01
Storing the logs on local host	OK		

At the bottom, there are statistics: 29 EXECUTIONS, 17% SUCCESS RATE, and 3m14s AVG DURATION. The footer includes copyright information and links to Community News, Project Settings, and Help.

Continuous Integration - Jenkins.

We created a pipeline using Jenkins. Each job in the pipeline initiates the downstream job on successful execution except for the last job which has no downstream job (Build pipeline plugin used).

With the Pipeline plugin, we can implement a project's entire pipeline in a Jenkinsfile and store that alongside their code, treating their pipeline as another piece of code checked into source control. The pipeline was created with the GUI based Jenkins plugin Blue Ocean. It asks us to connect our github repository which should contain the Jenkinsfile file in the root folder.



Builds - Jenkins.

STATUS	RUN	COMMIT	BRANCH	MESSAGE	DURATION	COMPLETED
✓	26	e53df24	maven	updated Jenkins File	7m 45s	12 hours ago
✓	25	6f2661e	maven	Started by user Nikhil Sai Bukka	7m 19s	a day ago
✓	22	6f2661e	maven	changed jenkins file	18m 19s	a day ago
✗	21	70821d7	maven	Restarted from build #19, stage Testing	1m 30s	a day ago
-	20	70821d7	maven	Restarted from build #19, stage Testing	24s	a day ago
✗	19	70821d7	maven	Started by user Nikhil Sai Bukka	6m 35s	a day ago
✓	18	70821d7	maven	removed password for database	4m 36s	2 days ago
✓	17	e25b3cd	maven	Started by user Nikhil Sai Bukka	6m 37s	2 days ago

Installation Procedure

SCM - git.

We should start out by running general OS and package updates. On Ubuntu we'll do this by running:

```
$ sudo apt-get update
```

After you have run the general updates on the server you can get started with installing Git.

1. Install Git using the following command :

```
$ sudo apt-get install git-core
```

2. You may be asked to confirm the download and installation; simply enter **y** to confirm. It's that simple, Git should be installed and ready to use!
3. Confirm Git the installation

With the main installation done, first check to ensure the executable file is set up and accessible. The best way to do this is simply to run Git with the version command.

```
$ git --version
```

Build - maven.

Prerequisites:

Maven 3.3+ requires JDK 1.7 or above to be installed. We'll install OpenJDK, which is the default Java development and runtime in Ubuntu.

Start by updating the package index and install the OpenJDK package by using :

```
$ sudo apt update  
$ sudo apt install default-jdk
```

Verify the installation by running the following command:

```
$ java -version
```

Install Maven by typing the following commands:

```
$ sudo apt install maven
```

Verify the installation by running the maven version command:

```
$ mvn -version
```

The output should look something like this:

```
arunav@rundeck:~$ mvn -version  
Apache Maven 3.6.0  
Maven home: /usr/share/maven  
Java version: 11.0.7, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd64  
Default locale: en_IN, platform encoding: UTF-8  
OS name: "linux", version: "5.3.0-51-generic", arch: "amd64", family: "unix"  
arunav@rundeck:~$
```

Test -selenium/JUnit.

No separate installation needed. Here are the maven dependencies:

```
<dependencies>  
    <dependency>  
        <groupId>junit</groupId>  
        <artifactId>junit</artifactId>  
        <version>4.8.1</version>  
        <scope>test</scope>  
    </dependency>  
  
    <dependency>  
        <groupId>org.seleniumhq.selenium</groupId>  
        <artifactId>selenium-java</artifactId>  
        <version>3.4.0</version>  
    </dependency>  
</dependencies>
```

Artifact - Docker.

Docker is an application that makes packaging and running an application process easy. It provides OS-level virtualization to deliver the software in a package called the container.

There are two methods to install Docker.

1. Installing it on an existing operating system (Ubuntu 16.04).
2. Using a tool called Docker Machine.

We are going to follow the first method. You can find the second method [here](#).

Prerequisites:

- Laptop with Ubuntu 16.04 installed in it
- [Docker Hub account](#) to create and push your own images to the docker hub.

Installing Docker

The docker installation package available on Ubuntu 16.04 repo may not be the latest version. We need to download and install the latest version from the official docker repository.

First, in order to ensure the downloads are valid, add the GPG key for the official Docker repository to your system

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Add the Docker repository to APT sources to your system

```
$ sudo add-apt-repository "deb [arch=amd64]  
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

Update the Ubuntu package Database

```
$ sudo apt-get update
```

To make sure we are installing from official docker repository run the following command :

```
$ apt-cache policy docker-ce
```

The output would look like this :

```
Installed: (none)

Candidate: 5:19.03.9-3-0-ubuntu-xenial

Version table:

 5:19.03.9-3-0-ubuntu-xenial 500
 500 https://download.docker.com/linux/ubuntu xenial/stable amd64 Packages
```

Finally Install docker

```
$ sudo apt-get install -y docker-ce
```

Docker should be installed now. Checking that it's running

```
$ sudo systemctl status docker
```

The output should look like :

```
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2020-05-25 20:36:12 IST; 1h 42min ago
       Docs: https://docs.docker.com
    Main PID: 2435 (dockerd)
      Tasks: 30
     Memory: 40.7M
        CPU: 11.075s
      CGroup: /system.slice/docker.service
              └─ 2435 /usr/bin/dockerd -H fd://
                --containerd=/run/containerd/containerd.sock
                  └─ 15162 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0
                    -host-port 8081 -container-ip 172.17.0.2 -container-port 8080
```

If the status of the docker is off. You can start the docker using the command

```
$ sudo systemctl start docker
```

Executing the docker without sudo command

By default running the docker commands required root privileges (prefix **sudo** to commands). A **docker group** is created by default. All the users in the docker group can run the docker commands without root privileges. If the user is not added to the docker group and tries to run the docker commands the output would look like this

```
Output
docker: Cannot connect to the Docker daemon. Is the docker daemon running
on this host?.
See 'docker run --help'.
```

To avoid typing sudo, add your username to the docker group

```
$ sudo usermod -aG docker ${USER}
```

For the new group membership to apply you need to logout and login or restart.

If you want to add another user who is logged in you can use the following command.

```
$ sudo usermod -aG docker username
```

You can get the documentation on all the docker commands [here](#). or run **docker** in command prompt. Tutorials on how to work with docker images and containers are available [here](#).

To Deploy our Web application we are using the Tomcat web server and ExistDB as the database.

Deploy - rundeck.

Rundeck is used to automate the ad-hoc and routine procedures in the data center or cloud environments.

Installation procedure for rundeck

Minimum Requirements

- Java version 1.8

Installing Rundeck

To install the Rundeck run the following commands.

```
$ echo "deb https://rundeck.bintray.com/rundeck-deb /" | sudo tee -a  
/etc/apt/sources.list.d/rundeck.list  
  
$ curl 'https://bintray.com/user/downloadSubjectPublicKey?username=bintray' | sudo apt-key  
add -  
  
$ sudo apt-get update  
  
$ sudo apt-get install rundeck
```

Rundeck service can be started using the command

```
$ sudo service rundeckd start
```

And you can check the status of Rundeck using the command

```
$ sudo service rundeckd status
```

Logging in for the first time

1. Navigate to <http://localhost:4440/> in a browser.
1. Log in with the username **admin** and password **admin**

Rundeck is now up and running!

Tutorials on how to use rundeck can be found [here](#).

Monitor - ELK.

ELK stands for three tools namely elastic search, logstash, and kibana. We have installed these three tools in Ubuntu 16.04 using exe available in elastic.co website.

<https://www.elastic.co/> . First update your system using: *sudo apt-get update*

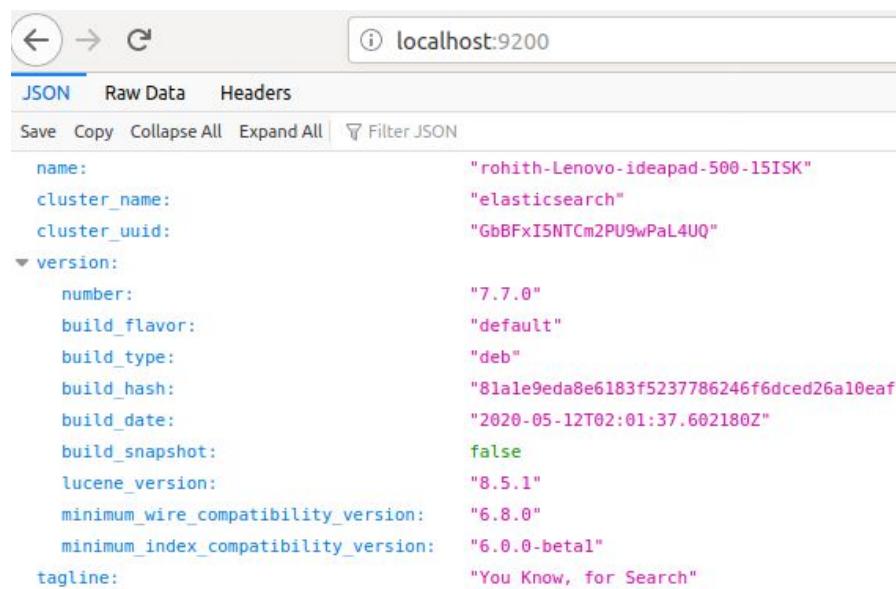
Installing Elasticsearch

Elasticsearch is a text search and analytics engine. It provides the ability to store, search and analyze huge amounts of data in real-time. And the best part of these tools is they are all open-source software.

For Ubuntu 16.04 bit install using :

```
$ echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" |  
sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list  
$ sudo apt-get install elasticsearch  
$ sudo systemctl start elasticsearch  
$ sudo systemctl enable elasticsearch
```

elasticsearch running in localhost:9200



```
localhost:9200  
JSON Raw Data Headers  
Save Copy Collapse All Expand All Filter JSON  
name: "rohith-Lenovo-ideapad-500-15ISK"  
cluster_name: "elasticsearch"  
cluster_uuid: "GbBFxI5NTCm2PU9wPaL4UQ"  
version:  
  number: "7.7.0"  
  build_flavor: "default"  
  build_type: "deb"  
  build_hash: "81ale9eda8e6183f5237786246f6dcfd26a10eaf"  
  build_date: "2020-05-12T02:01:37.602180Z"  
  build_snapshot: false  
  lucene_version: "8.5.1"  
  minimum_wire_compatibility_version: "6.8.0"  
  minimum_index_compatibility_version: "6.0.0-beta1"  
tagline: "You Know, for Search"
```

Installing Filebeat

File Beats are lightweight agents that help us to collect different types of logs so that we can forward into the logstash or elasticsearch.

For Installing File beat we run

```
$ sudo apt-get install filebeat  
$ sudo systemctl start filebeat  
$ sudo systemctl enable filebeat
```

```
root@rohith-Lenovo-ideapad-500-15ISK:~# sudo apt-get install filebeat  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following NEW packages will be installed:  
  filebeat  
0 upgraded, 1 newly installed, 0 to remove and 20 not upgraded.  
Need to get 28.1 MB of archives.  
After this operation, 91.2 MB of additional disk space will be used.  
Get:1 https://artifacts.elastic.co/packages/7.x/apt stable/main amd64 filebeat amd64 7.7.0 [28.1 MB]  
Fetched 28.1 MB in 29s (966 kB/s)  
Selecting previously unselected package filebeat.  
(Reading database ... 284054 files and directories currently installed.)  
Preparing to unpack .../filebeat_7.7.0_amd64.deb ...  
Unpacking filebeat (7.7.0) ...  
Processing triggers for systemd (229-4ubuntu21.28) ...  
Processing triggers for ureadahead (0.100.0-19.1) ...  
ureadahead will be reprofiled on next reboot  
Setting up filebeat (7.7.0) ...  
Processing triggers for systemd (229-4ubuntu21.28) ...  
Processing triggers for ureadahead (0.100.0-19.1) ...  
root@rohith-Lenovo-ideapad-500-15ISK:~# subl /etc/filebeat/filebeat.yml
```

Now the next step is to configure Filebeat to collect the Tomcat access log file and forward it to Logstash for processing.

```
filebeat.inputs:  
  
# Each - is an input. Most options can be set at the input level, so  
# you can use different inputs for various configurations.  
# Below are the input specific configurations.  
  
- type: log  
  
  # Change to true to enable this input configuration.  
  enabled: true  
  
  # Paths that should be crawled and fetched. Glob based paths.  
  paths:  
    - /opt/tomcat/logs/localhost_access_log.*.txt  
    #- c:\programdata\elasticsearch\logs\*  
  
#----- Logstash output -----  
output.logstash:  
  # The Logstash hosts  
  hosts: ["localhost:5044"]
```

Enable all the filebeat modules by using *sudo filebeat modules enable system*

Now all the tomcat paths are correct and the outputs are also redirected to the correct path.

Installing Logstash

Logstash is essentially a log aggregator that collects data from various input sources, like servers, applications, docker containers etc and performs various transformations and pushes it into elasticsearch.

Again for Ubuntu 16.04 bit install using :

```
$ sudo apt-get install kibana  
$ sudo systemctl start kibana
```

After that run these commands :

```
root@rohit-Lenovo-ideapad-500-15ISK:~# subl /etc/logstash/conf.d/tomcat.conf  
root@rohit-Lenovo-ideapad-500-15ISK:~# sudo service logstash start  
root@rohit-Lenovo-ideapad-500-15ISK:~# sudo service filebeat start
```

We can write the tomcat config file to send the output to elasticsearch as below. In this file tomcat.conf, we used the beats input plugin, a number of filter plugins these are optional, and the Elasticsearch output plugin. All the output is redirected to elasticsearch.

```
1 |input {  
2 |  beats {  
3 |    port => 5044  
4 |  }  
5 |}  
6 |  
7 |filter {  
8 |  grok {  
9 |    match => { "message" => "%{COMBINEDAPACHELOG}" }  
10 |  }  
11 |  date {  
12 |    match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]  
13 |  }  
14 |  geoip {  
15 |    source => "clientip"  
16 |  }  
17 |}  
18 |}  
19 |  
20 |output {  
21 |  elasticsearch {  
22 |    hosts => ["localhost:9200"]  
23 |  }  
24 |}  
25 |
```

Here we can check if the output is redirected correctly to localhost:9200 - elasticsearch.

```
root@rohit-Lenovo-ideapad-500-15ISK:~# sudo service filebeat start
root@rohit-Lenovo-ideapad-500-15ISK:~# sudo service logstash start
root@rohit-Lenovo-ideapad-500-15ISK:~# curl -X GET "localhost:9200/_cat/indices?v"
health status index          uuid                               pri rep docs.count docs.deleted store.size pri.store.size
yellow open  logstash-2020.05.23-000001 Tjl93pMQSB-oSQbwVTAhog 1   1    20242      0      5.5mb      5.5mb
green  open  .apm-custom-link  6CmJSG9QY61xx-U01Ln_g 1   0      0       0      208b      208b
green  open  .kibana_task_manager_1 Wj_mg4IDS3K_QcLX4-JD_g 1   0      5       0      31.9kb     31.9kb
green  open  .apm-agent-configuration PE_fle3aqSuqqT781xiAwW 1   0      0       0      208b      208b
green  open  .kibana_1        WrIrgxemSbiOniFgyCHC8Q 1   0    1438      50    752.4kb    752.4kb
yellow open  filebeat-7.7.0-2020.05.23-000001 zIuQ92nNT920RsZ3JTp37g 1   1      0       0      208b      208b
root@rohit-Lenovo-ideapad-500-15ISK:~#
```

Installing Kibana

Kibana is used for visualization and it works on top of Elasticsearch. It also provides us with the ability to analyze or query the data. These visualizations could be in the form of tables, plots, charts etc.

Again for Ubuntu 16.04 bit install using :

- *sudo apt-get install kibana*

I have faced this error while running this command

```
rohit@rohit-Lenovo-ideapad-500-15ISK:~$ sudo apt-get install kibana
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  kibana
0 upgraded, 1 newly installed, 0 to remove and 20 not upgraded.
Need to get 289 MB of archives.
After this operation, 918 MB of additional disk space will be used.
Get: https://artifacts.elastic.co/packages/7.x/apt stable/main amd64 kibana amd64 7.7.0 [289 MB]
Err:1 https://artifacts.elastic.co/packages/7.x/apt stable/main amd64 kibana amd64 7.7.0
  GnuTLS recv error (-54): Error in the pull function.
E: Failed to fetch https://artifacts.elastic.co/packages/7.x/apt/pool/main/k/kibana/kibana-7.7.0-amd64.deb  GnuTLS recv error (-54): Error
in the pull function.

E: Unable to fetch some archives, maybe run apt-get update or try with --fix-missing?
rohit@rohit-Lenovo-ideapad-500-15ISK:~$ sudo apt-get update
```

Which was solved by running *sudo apt-get update* before installing it again.

kibana running in localhost:5601

The screenshot shows the Kibana home page at localhost:5601/app/kibana#/home. The interface is divided into several sections:

- Observability** section:
 - APM**: APM automatically collects in-depth performance metrics and errors from inside your applications. Buttons: [Add APM](#), [Add log data](#), [Add metric data](#).
 - Logs**: Ingest logs from popular data sources and easily visualize in preconfigured dashboards.
 - Metrics**: Collect metrics from the operating system and services running on your servers.
 - SIEM**: Centralize security events for interactive investigation in ready-to-go visualizations. Button: [Add events](#).
- Add sample data**: Load a data set and a Kibana dashboard.
- Upload data from log file**: Import a CSV, NDJSON, or log file.
- Use Elasticsearch data**: Connect to your Elasticsearch index.
- Visualize and Explore Data** section:
 - APM**: Automatically collect in-depth performance metrics and errors from inside your applications.
 - Canvas**: Showcase your data in a pixel-perfect way.
 - Dashboard**: Display and share a collection of visualizations and saved searches.
 - Discover**: Interactively explore your data by querying and filtering raw documents.
- Manage and Administer the Elastic Stack** section:
 - Console**: Skip cURL and use this JSON interface to work with your data directly.
 - Index Patterns**: Manage the index patterns that help retrieve your data from Elasticsearch.
 - Monitoring**: Track the real-time health and performance of your Elastic Stack.
 - Rollups**: Summarize and store historical data in a smaller index for future analysis.

Finally in Kibana: Management > Index Pattern (Create Index Pattern) - logstash-*

The screenshot shows the Kibana Management > Index Patterns > Create index pattern screen at localhost:5601/app/kibana#/management/index_patterns/create_index_pattern. The interface is a wizard:

- Step 1 of 2: Define index pattern**
 - index pattern**: `logstash*`
 - Help text: You can use a * as a wildcard in your index pattern.
You can't use spaces or the characters \, /, ?, <, >, |.
 - Success message: `✓ Success! Your index pattern matches 1 index.`
 - Result: `logstash-2020.05.23-000001`
 - Rows per page: 10

We can find the indices that are loaded in elasticsearch in kibana index patterns and load them for visualizing.

Click on Next Step > logstash-* :

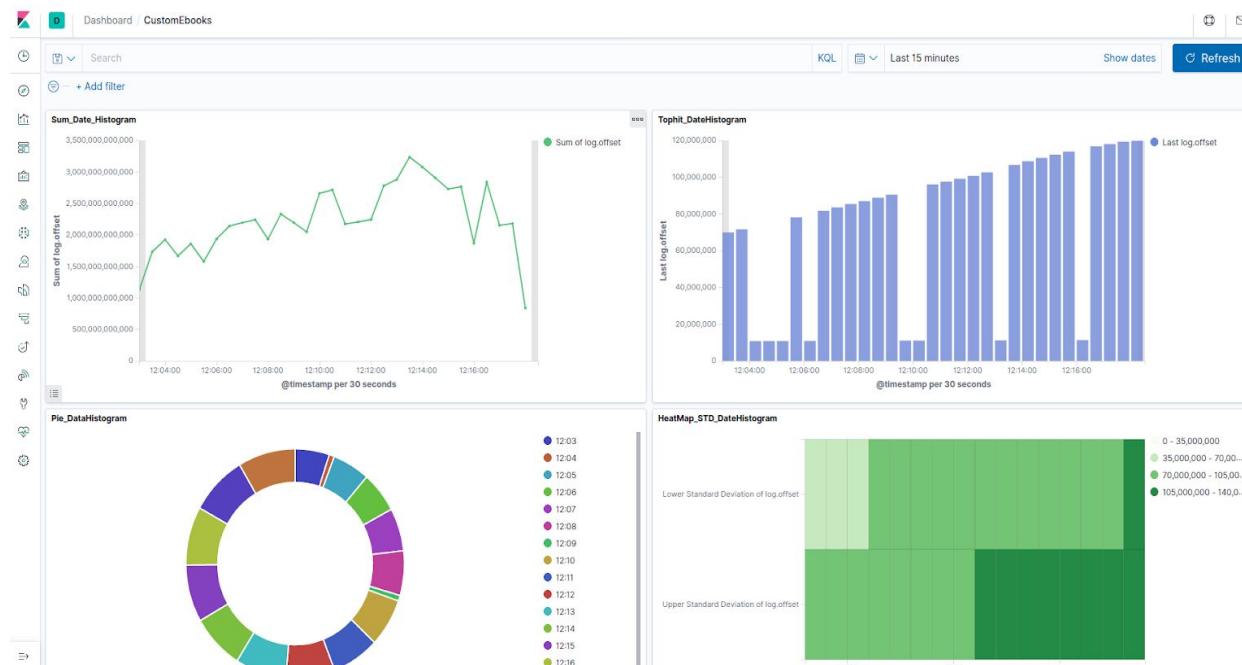
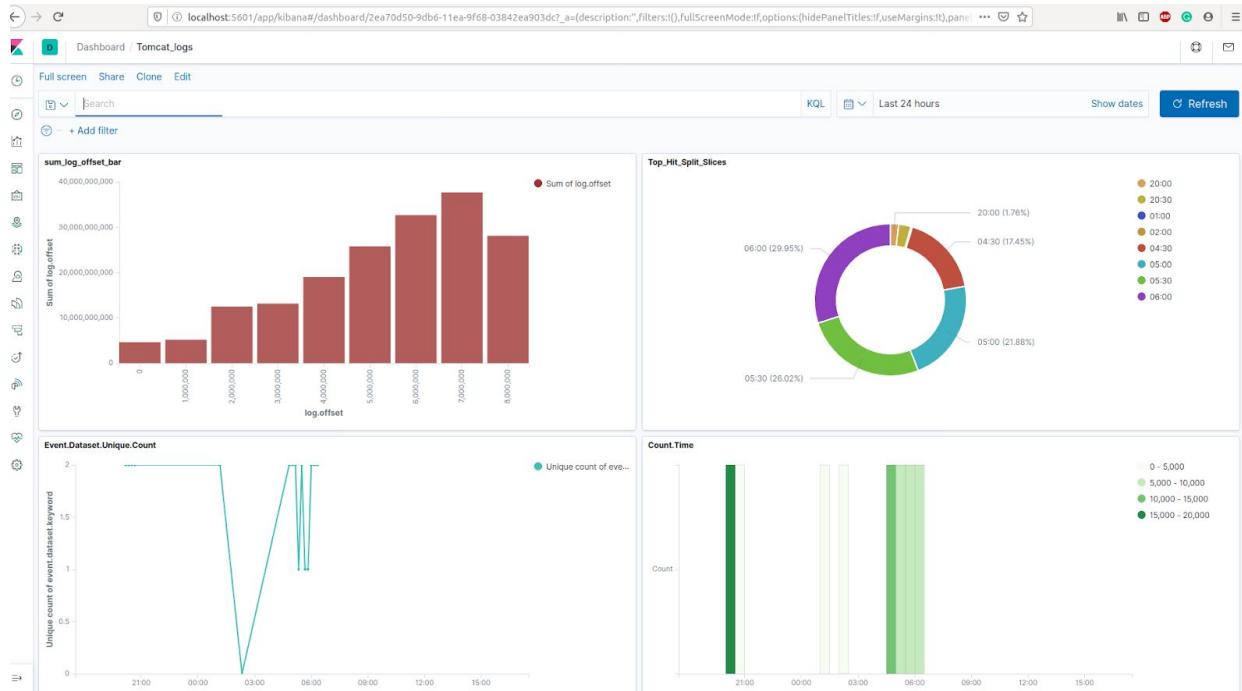
The screenshot shows the Elasticsearch Management interface under the 'Index patterns' section for the 'logstash-*' index. On the left, there's a sidebar with links for Elasticsearch (Index Management, Index Lifecycle Policies, Transform, Rollup Jobs, Snapshot and Restore, License Management, Remote Clusters, 8.0 Upgrade Assistant) and Kibana (Index Patterns, Alerts and Actions, Spaces, Saved Objects, Reporting, Advanced Settings). The main area has a title '★ logstash-*' with a star icon. Below it is a sub-header: 'Time Filter field name: @timestamp [Default]'. A note says: 'This page lists every field in the logstash-* index and the field's associated core type as recorded by Elasticsearch. To change a field type, use the Elasticsearch Mapping API %'. There are three tabs at the top: 'Fields (66)', 'Scripted fields (0)', and 'Source filters (0)'. A search bar labeled 'Filter' and a dropdown 'All field types' are present. A table lists 66 fields with columns for Name, Type, Format, Searchable, Aggregatable, and Excluded. Fields include @timestamp, @version, _id, _index, _score, _source, _type, agent.ephemeral_id, agent.ephemeral_id.keyword, and agent.hostname. At the bottom, there's a 'Rows per page: 10' dropdown and a navigation bar with pages 1 through 7.

Go to Discover > logstash-*:

The screenshot shows the Elasticsearch Discover interface for the 'logstash-*' index. The top navigation bar includes 'New', 'Save', 'Open', 'Share', 'Inspect', 'Search', '+ Add filter', 'Last 30 minutes', 'Show dates', and 'Refresh'. The left sidebar has sections for 'Selected fields' (containing @source) and 'Available fields' (listing @timestamp, @version, _id, _index, _score, _type, agent.ephemeral_id, agent.hostname, agent.id, agent.type, agent.version, ecs.version, event.dataset, event.module, event.timezone, fileset.name). The main area displays a histogram titled '406 hits' showing the count of events over time from May 23, 2020, at 20:30:00 to 20:55:00. Below the histogram, several log entries are shown in a table, each with a timestamp and a detailed log message. The log messages are identical, showing system logs related to a user session being opened for root user by (uid=0).

Time	_source
May 23, 2020 @ 20:40:43.1	input.type: log event.dataset: system.auth.event.timezone: +05:30 event.module: system.log.offset: 757,804 log.file.path: /var/log/syslog host.mac: 1c:39:47:25:32:b2, e0:94:67:40:ad:cb, 02:42:08:e9:9b:57, 02:42:9c:7b:25:11 host.os.family: debian host.os.codename: xenial host.os.version: 16.04.6 LTS (Xenial Xerus) host.os.kernel: 4.15.0-96-generic host.os.platform: ubuntu host.os.name: Ubuntu host.hostname: rohit-Lenovo-ideapad-500-15ISK host.ip: 192.168.0.119, fe80::413:fec0:cfc5:db0f, 172.17.0.1, 172.18.0.1 host.containedIn: false host.id: b47d8e69318f48e394198fb7709e53c1 host.name: rohit-Lenovo-ideapad-500-15ISK host.architecture: x86_64 ecs.version: 1.5.0 message: May 23 20:40:53 rohit-Lenovo-ideapad-500-15ISK filebeat[22353]:
May 23, 2020 @ 20:40:43.510	input.type: log event.dataset: system.auth.event.timezone: +05:30 event.module: system.log.offset: 74,236 host.mac: 1c:39:47:25:32:b2, e0:94:67:40:ad:cb, 02:42:08:e9:9b:57, 02:42:9c:7b:25:11 host.os.family: debian host.os.codename: xenial host.os.version: 16.04.6 LTS (Xenial Xerus) host.os.kernel: 4.15.0-96-generic host.os.platform: ubuntu host.os.name: Ubuntu host.hostname: rohit-Lenovo-ideapad-500-15ISK host.ip: 192.168.0.119, fe80::413:fec0:cfc5:db0f, 172.17.0.1, 172.18.0.1 host.containedIn: false host.id: b47d8e69318f48e394198fb7709e53c1 host.name: rohit-Lenovo-ideapad-500-15ISK host.architecture: x86_64 ecs.version: 1.5.0 message: May 23 20:40:53 rohit-Lenovo-ideapad-500-15ISK sudo: root : TTY pts/2 ; PWD=/home/rohit ; USER=root ;
May 23, 2020 @ 20:40:43.518	input.type: log event.dataset: system.auth.event.timezone: +05:30 event.module: system.log.offset: 74,488 host.mac: 1c:39:47:25:32:b2, e0:94:67:40:ad:cb, 02:42:08:e9:9b:57, 02:42:9c:7b:25:11 host.os.family: debian host.os.codename: xenial host.os.version: 16.04.6 LTS (Xenial Xerus) host.os.kernel: 4.15.0-96-generic host.os.platform: ubuntu host.os.name: Ubuntu host.hostname: rohit-Lenovo-ideapad-500-15ISK host.ip: 192.168.0.119, fe80::413:fec0:cfc5:db0f, 172.17.0.1, 172.18.0.1 host.containedIn: false host.id: b47d8e69318f48e394198fb7709e53c1 host.name: rohit-Lenovo-ideapad-500-15ISK host.architecture: x86_64 ecs.version: 1.5.0 message: May 23 20:40:53 rohit-Lenovo-ideapad-500-15ISK sudo: root : TTY pts/2 ; PWD=/home/rohit ; USER=root ;
May 23, 2020 @ 20:40:43.518	input.type: log event.dataset: system.auth.event.timezone: +05:30 event.module: system.log.offset: 74,518 host.mac: 1c:39:47:25:32:b2, e0:94:67:40:ad:cb, 02:42:08:e9:9b:57, 02:42:9c:7b:25:11 host.os.family: debian host.os.codename: xenial host.os.version: 16.04.6 LTS (Xenial Xerus) host.os.kernel: 4.15.0-96-generic host.os.platform: ubuntu host.os.name: Ubuntu host.hostname: rohit-Lenovo-ideapad-500-15ISK host.ip: 192.168.0.119, fe80::413:fec0:cfc5:db0f, 172.17.0.1, 172.18.0.1 host.containedIn: false host.id: b47d8e69318f48e394198fb7709e53c1 host.name: rohit-Lenovo-ideapad-500-15ISK log.offset: 74,518 log.file.path: /var/log/auth.log

Kibana Dashboard :



Continuous Integration - jenkins.

Prerequisites :

Jenkins is a Java application, the first step is to install Java. Update the package index and install the Java 8 OpenJDK package with the following commands :

```
$ sudo apt update  
$ sudo apt install openjdk-8-jdk
```

To add the Jenkins Debian repository. Import the GPG keys of the Jenkins repository using the following wget command :

```
$ wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
```

The command above should output OK which means that the key has been successfully imported and packages from this repository will be considered trusted.

Next, add the Jenkins repository to the system with :

```
$ sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
```

Once the jenkins repository is enabled , update the apt package list and install jenkins using following commands :

```
$ sudo apt update  
$ sudo apt install jenkins
```

Jenkins service will automatically start after the installation process is complete. You can verify it by printing the service status :

```
$ systemctl status jenkins
```

Output should look similar to this :

```
arunav@rundeck:~$ sudo systemctl status jenkins  
[sudo] password for arunav:  
● jenkins.service - LSB: Start Jenkins at boot time  
  Loaded: loaded (/etc/init.d/jenkins; generated)  
  Active: active (exited) since Tue 2020-05-26 09:17:21 IST; 2h 48min ago  
    Docs: man:systemd-sysv-generator(8)  
   Process: 2148 ExecStart=/etc/init.d/jenkins start (code=exited, status=0/SUCCE
```

Git in jenkins :

Click on the Manage Jenkins button on your Jenkins dashboard and open manage plugins, search for GIT Plugin and choose install without restart . Installation might take some time to download all the dependencies,restart the jenkins after installation successfully finished.



Maven in Jenkins :

Maven Integration plugin helps us to build projects that use Apache Maven in Jenkins.Go to Manage Jenkins, select Manage Plugins, select the Available tab, then find the Maven Integration plugin,install it and restart the jenkins.

Docker in Jenkins :

To add docker to the jenkins pipeline, first we have to install Docker plugin in jenkins. Add jenkins in the docker group to execute docker commands from jenkins.

```
$ sudo usermod -aG docker jenkins
```

Then add docker hub credentials in the credentials section in jenkins. Restart Jenkins after doing this. Jenkinsfile can now be edited to add the docker build and push stages.

Rundeck in jenkins :

Install Rundeck plugin in Jenkins. In Manage Jenkins > Configure System > Rundeck , fill the required details and do Test Connection . If it is working, it shows a successful message. If not, execute command

```
$ service rundeckd restart
```

then open <http://localhost:4440/> to verify if Rundeck is working.

Experimental Setup

Functional Requirements

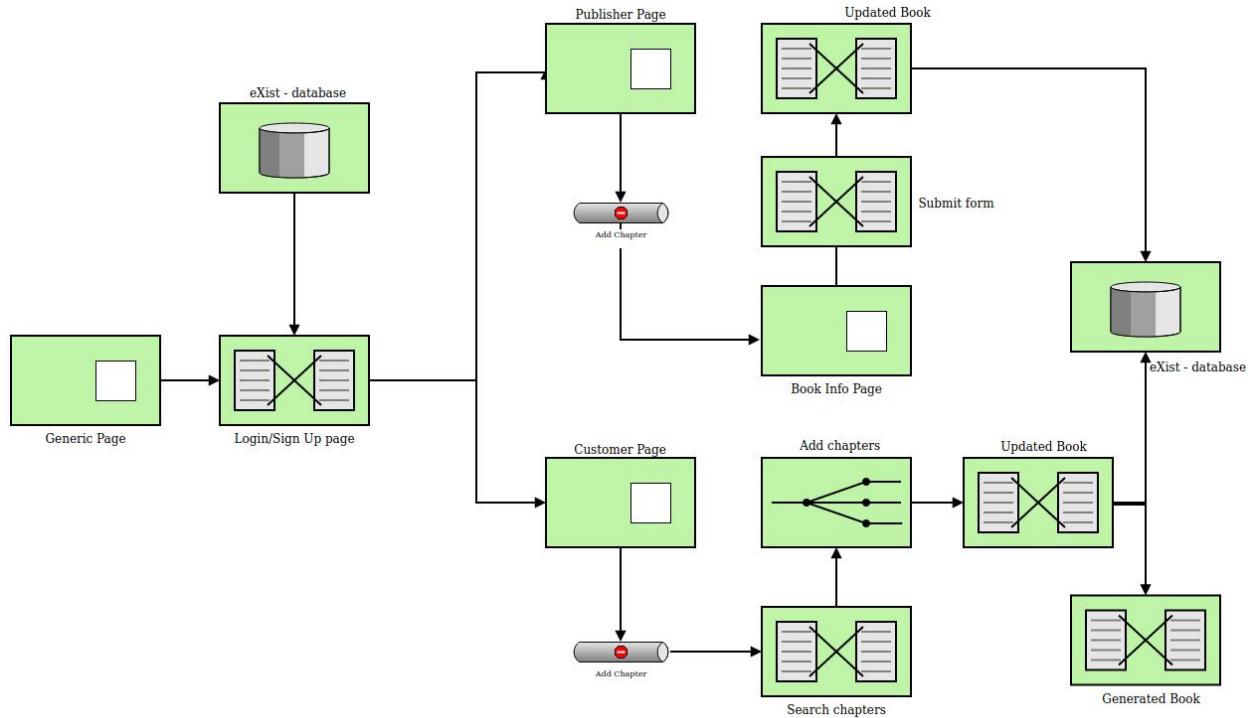
Two kinds of users : publishers and consumer

- Ability for publishers to post books (in pieces) and for customers to accumulate these pieces to create a custom book.
- Publishers should be able to upload chapters (in the form of pdf). Publishers will also provide the name and keywords associated with the chapter.
- The publisher should be able to review and delete items before publishing.
- The consumer should be able to build a book from scratch combining the chapters uploaded by different publishers.
- Consumers should be able to search for chapters based on keywords and chapter names.
- Consumers should be able to preview the chapter before adding it to the book.
- Consumers should be able to review the book structure and delete things before generating the book.

Non - Functional Requirements

- Given the model of having books made up of smaller units each with their own metadata, storing and retrieving might cause bottlenecks.
- The data storing and retrieval should not cause bottlenecks so requires an efficient content management framework.
- The search should be fast. This means we should be able to parse through the whole database in very less time.
- The user will be able to preview the documents. This should be efficient.

Architecture and Design



Code Walkthrough

Servlets

Add Book Servlet:

Adds a book to the exist database.

```
public class AddBookServlet extends HttpServlet {

    public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
        String id = request.getParameter("name");
        PrintWriter out = response.getWriter();
        HttpSession session = request.getSession();
        ExistSearchUtil ex = (ExistSearchUtil)session.getAttribute("ex");
        CustomEbookBuilder cb = (CustomEbookBuilder)session.getAttribute("cb");

        Element chapter = ex.ChapterAtIndex(Integer.parseInt(id));
        cb.addChapter(chapter);

        session.setAttribute("cb", cb);
        session.setAttribute("ex", ex);
    }
}
```

Upload servlet :

Gets the uploaded PDF and stores it.

```
int book_id = ex.noOfBooks();

cb.setBookId(String.valueOf(book_id));

String appPath = request.getServletContext().getRealPath("");
String savePath = appPath + File.separator + "uploads";

// creates the save directory if it does not exists
File fileSaveDir = new File(savePath);
if (!fileSaveDir.exists()) {
    fileSaveDir.mkdir();
}

File file = new File(savePath+File.separator+"example.xml");
if(file.delete()){
    System.out.println("File deleted");
} else System.out.println("File doesn't exist");

cb.saveAsXML(savePath+File.separator+"example.xml");
ex.StoreIntoCollection(savePath+File.separator+"example.xml",null);

cb = new CustomEbookBuilder("Database Systems");
session.setAttribute("cb",cb);
session.setAttribute("ex",ex);
```

Download Servlet :

Downloads the user defined book in the form of a PDF.

```

ServletContext context = getServletContext();

// gets MIME type of the file
String mimeType = context.getMimeType(filePath);
if (mimeType == null) {
    // set to binary type if MIME mapping not found
    mimeType = "application/octet-stream";
}
System.out.println("MIME type: " + mimeType);

// modifies response
response.setContentType(mimeType);
response.setContentLength((int) downloadFile.length());

// forces download
String headerKey = "Content-Disposition";
String headerValue = String.format("attachment; filename=\"%s\"", downloadFile.getName());
response.setHeader(headerKey, headerValue);

// obtains response's output stream
OutputStream outStream = response.getOutputStream();

byte[] buffer = new byte[4096];
int bytesRead = -1;

while ((bytesRead = inStream.read(buffer)) != -1) {
    outStream.write(buffer, 0, bytesRead);
}

inStream.close();
outStream.close();

```

Get Book Servlet :

Data api call to get the details of a book. Used in results display and book creation.

```

public class GetBookSrcServlet extends HttpServlet {

    public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
        String id = request.getParameter("name");
        PrintWriter out = response.getWriter();
        HttpSession session = request.getSession();
        ExistSearchUtil ex = (ExistSearchUtil)session.getAttribute("ex");
        CustomEbookBuilder cb = (CustomEbookBuilder)session.getAttribute("cb");
        String ret = ex.PdfLocationAtIndex(Integer.parseInt(id));
        String absPath = "../../uploads/";
        Path path = Paths.get(ret);
        ret = path.getFileName().toString();
        |
        // get source of pdf into ret
        out.println(absPath + ret);
        session.setAttribute("ex",ex);
    }
}

```

Search Results Servlet :

Data api call for results of a search keyword.

```
public class GetSearchResults extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
        doPost(request, response);
}

public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
    String keywords = request.getParameter("search_text");
    PrintWriter out = response.getWriter();
    HttpSession session = request.getSession();
    session.setAttribute("chapter_search_keywords", keywords);
}
}
```

Remove chapter :

Removes a chapter from the book that the customer is putting together. (Note: not for deleting a book in the database, just provides the customer with a functionality of deleting certain added parts of the book while putting things together.

```
public class RemoveChapterServlet extends HttpServlet {

    public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
        String id = request.getParameter("name");
        PrintWriter out = response.getWriter();
        HttpSession session = request.getSession();
        CustomEbookBuilder cb = (CustomEbookBuilder) session.getAttribute("cb");

        cb.removeChapter(Integer.parseInt(id));
        session.setAttribute("cb", cb);
    }
}
```

Upload ChapterServlet :

Stores a chapter into the database.

```

String chapter_path = savePath + File.separator + randomString() + "_" + fileName ;
filePart.write(chapter_path);

try{

    HttpSession session = request.getSession();
    int id;
    if(session.getAttribute("id")!=null)
        id = (int) session.getAttribute("id")+1;
    else
        id = 1;
    String chapter_id = String.valueOf(id);

    CustomEbookBuilder cb = (CustomEbookBuilder)session.getAttribute("cb");
    Element chap = cb.createChapter(chapter_name,chapter_id,chapter_tags,chapter_path);

    if(!chapter_name.trim().isEmpty() && !chapter_tags.trim().isEmpty() && !fileName.trim().isEmpty()){
        cb.addChapter(chap);
        session.setAttribute("id",id);
        session.setAttribute("cb",cb);
    }

    response.sendRedirect("../index.jsp");

} catch (Exception e) {
    System.out.println("DEBUG : Unable to create builder instance");
    e.printStackTrace();
}

```

User Login Servlet :

```

public class UserLoginServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
        doPost(request, response);
}

public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
    String id = request.getParameter("name");
    PrintWriter out = response.getWriter();
    HttpSession session = request.getSession();
    session.setAttribute("type",id);
}
}

```

Front end :

We won't include the html, JS and Jsp code here for brevity.

Results

Database - eXist-DB

eXist-db running on localhost:8081 :

The screenshot shows two windows. On the left is the eXist-db dashboard, which includes links for eXide (XQuery IDE), eXist-db Documentation, Java Admin Client, Markdown Parser in XQuery, Package Manager, Shutdown, Usermanager, and XQuery Function Documentation. On the right is the Collection Browser window, showing a list of XML files in the /db/CustomEbooks directory with columns for Name, Permissions, Owner, Group, and Last-modified.

Name	Permissions	Owner	Group	Last-modified
CRWXR-XR-X	SYSTEM	dba		May 01 2020 15:46:19
10574b26.xml	-rw-r--r--	admin	dba	May 10 2020 10:50:37
23e4388.xml	-rw-r--r--	admin	dba	May 11 2020 10:52:44
3:c853f1.xml	-rw-r--r--	admin	dba	May 11 2020 10:05:47
4180c8a1.xml	-rw-r--r--	admin	dba	May 08 2020 12:35:51
83e967ba.xml	-rw-r--r--	admin	dba	May 16 2020 12:09:42
8ae880d.xml	-rw-r--r--	admin	dba	May 11 2020 10:10:30
b87a1e3a.xml	-rw-r--r--	admin	dba	May 11 2020 11:09:49
c21aae32.xml	-rw-r--r--	admin	dba	May 11 2020 10:13:12
custom_ebooks_schema.xsd	-rw-r--r--	admin	dba	May 08 2020 12:21:29
d7ff80a8.xml	-rw-r--r--	admin	dba	May 08 2020 12:41:00

Website - Custom-eBooks

Generic page :

The screenshot shows a Mozilla Firefox browser window displaying the Custom-eBooks website at localhost:8090/Custom-eBooks/jsp/generic.jsp. The page features a large blue header with a stack of books icon and the text "CUSTOM-EBOOKS". Below the header, it says "This application lets you create an ebook having chapters and sections with your say in it". There are two red buttons: "I'M A CUSTOMER" and "I'M A PUBLISHER". The browser status bar shows the URL and the time (9:52 PM).



Retina Ready

All the ebooks are available for reading in form of pdfs so that the user will be able to select from the lot for their own ebook.



Fully Responsive

User friendly interface with fully responsive add and remove buttons for the users to define the structure for their ebook.

Publisher page :

A screenshot of a Mozilla Firefox browser window titled "Books - Mozilla Firefox (Private Browsing)". The address bar shows "localhost:8090/Custom-eBooks/index.jsp". The main content area has a blue header with the text "CUSTOM-EBOOKS.". Below the header is a search bar with the placeholder "Enter book name" and a red "SEARCH" button. To the right of the search bar is a navigation menu with links: Home, About, eBooks, How?, Contact, and a red "Login" button. Below the header is a large graphic featuring a blue and white abstract design with the text "COVER DESIGN" and "SUBHEADING HERE". To the right of the graphic is a white box with a blue header containing the word "Chapters" and a red "Add" button. Below this is a red "Upload PDF" button.

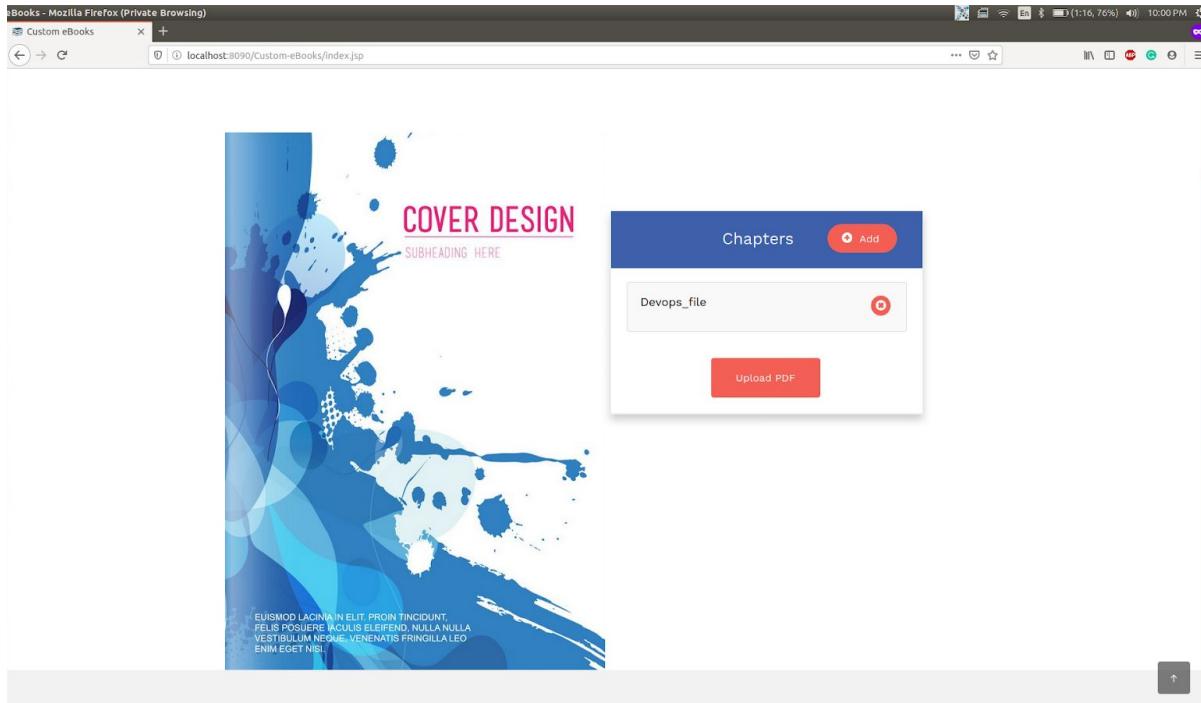
Book info page :

A screenshot of a Mozilla Firefox browser window titled "Books - Mozilla Firefox (Private Browsing)". The address bar shows "localhost:8090/Custom-eBooks/html/info.html". The main content area has a blue header with the text "CUSTOM-EBOOKS.". Below the header is a large white box with the text "BOOK INFORMATION" in bold capital letters. At the top of this box is a sub-header "Chapters Info". Below it are two input fields: "Chapter Name" with a placeholder "Enter chapter name" and "Chapter Tags" with a placeholder "Enter keywords or tags". At the bottom of the box is a large dashed rectangular area with a central icon of an upward arrow and the text "CLICK TO BROWSE OR DROP FILE HERE".

Upload Chapter :

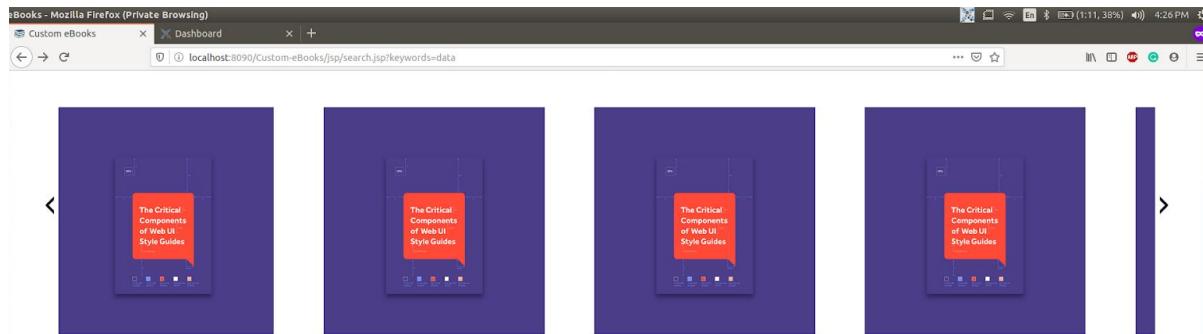
The screenshot shows a Mozilla Firefox browser window with the title bar "Books - Mozilla Firefox (Private Browsing)" and the address bar "localhost:8090/Custom-eBooks/html/info.html". The main content area is titled "Chapters Info". It contains fields for "Chapter Name" (with "Devops_file" entered) and "Chapter Tags" (with "devops spe software engineering"). Below these is a file upload section with a dashed border and a central box containing an upward arrow icon and the text "CLICK TO BROWSE OR DROP FILE HERE". A file path "Rundeck_IEEE_CONECCT2019.pdf" is displayed below the box. A blue "submit" button is located at the bottom right of the form.

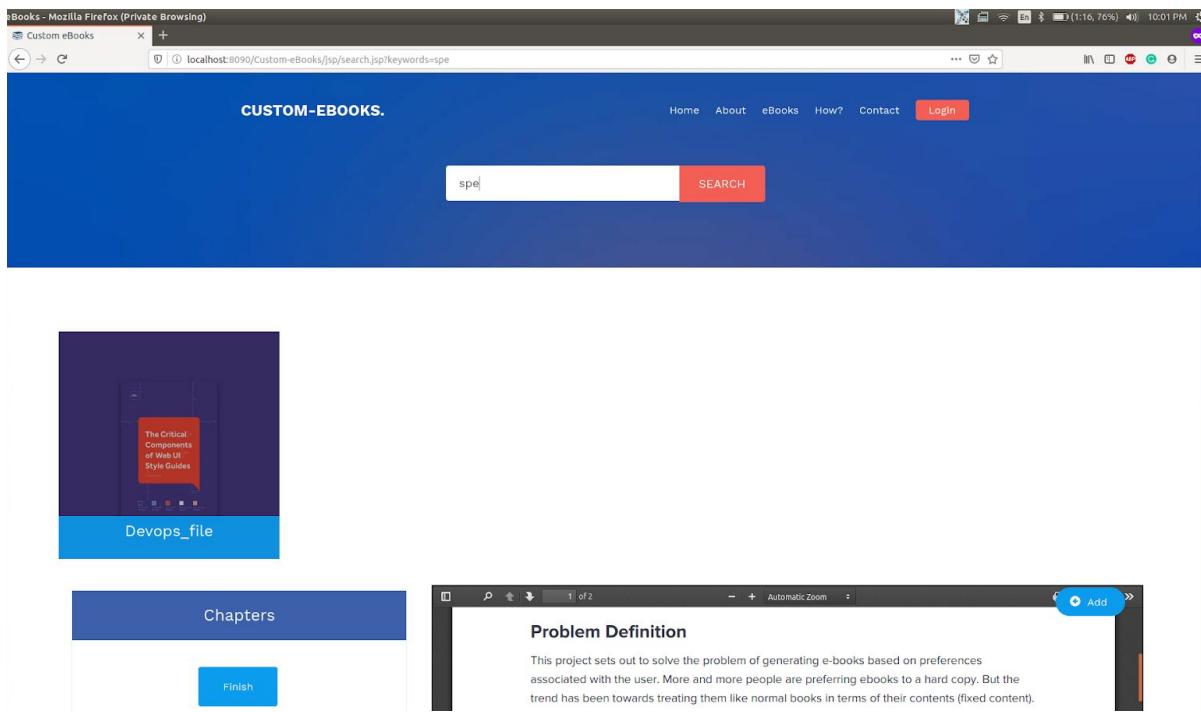
Chapter Uploaded successfully :



Customer page :

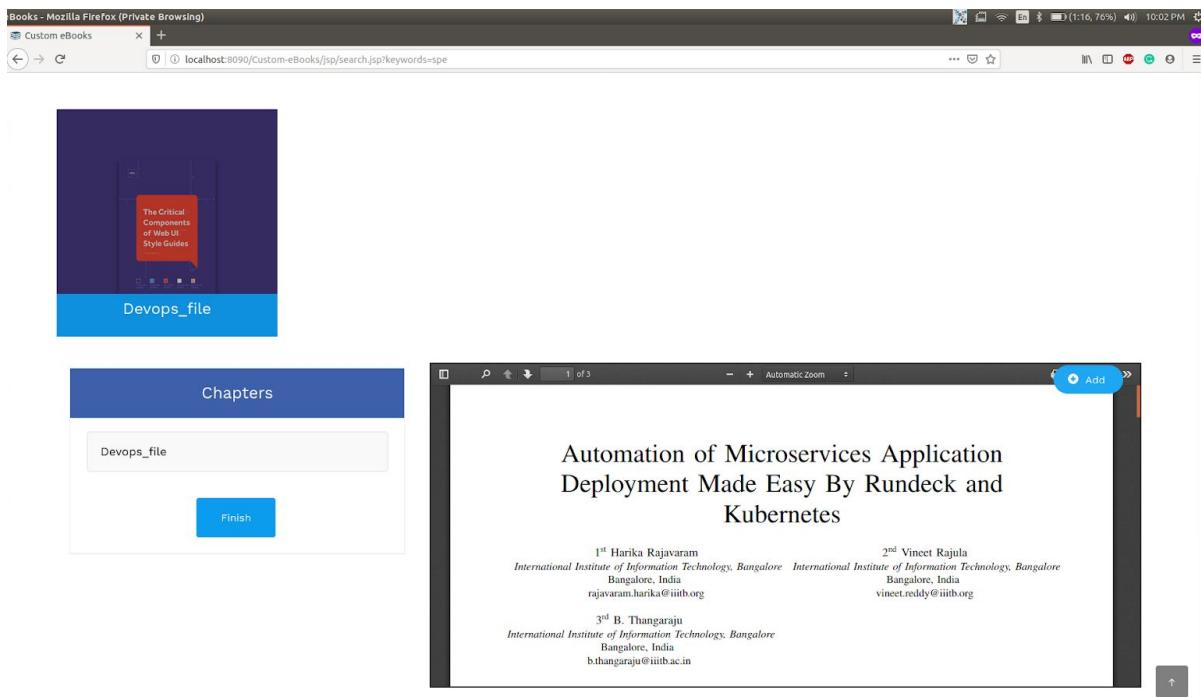
Click on any chapter to view



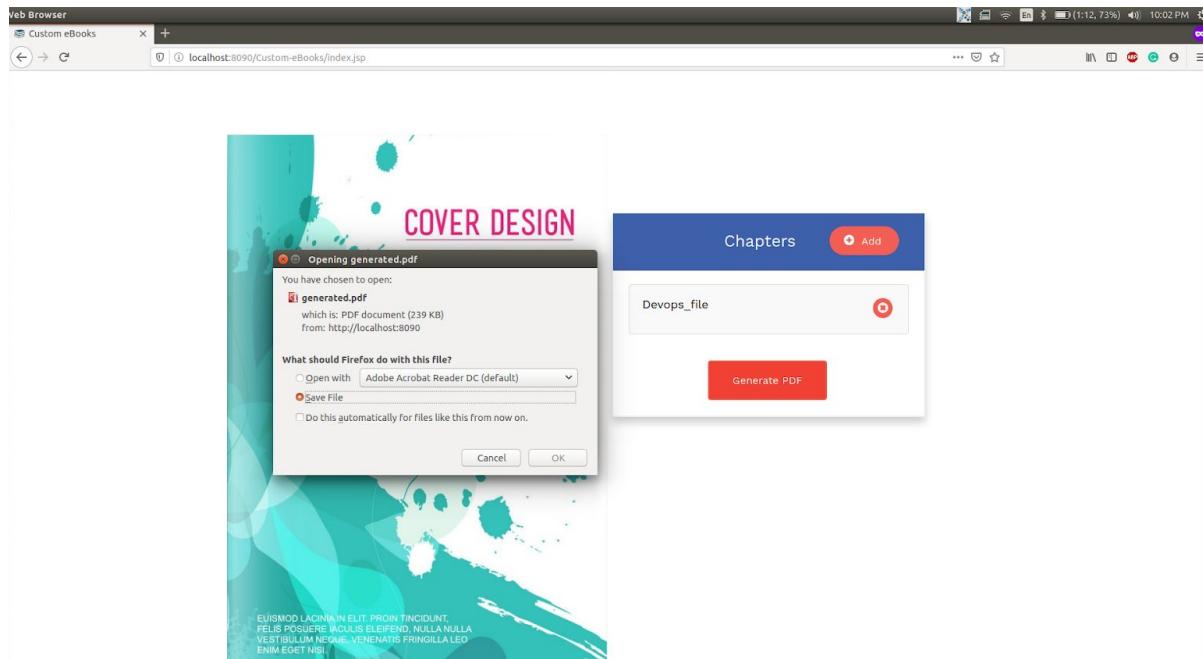
Chapter Added :

Click on add button to add the chapter to the book.



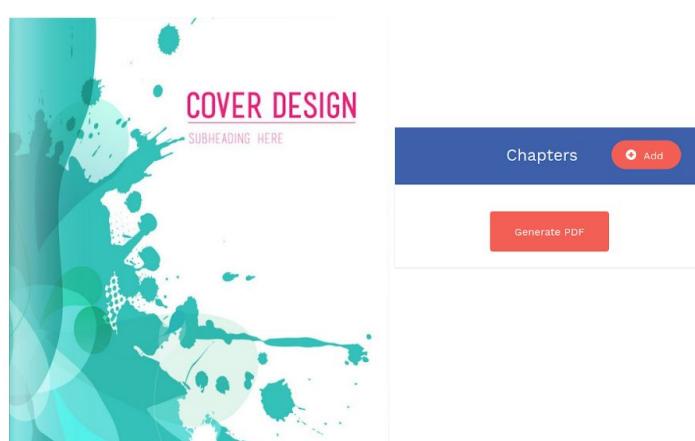
Book Generated :

Your custom-eBook is generated

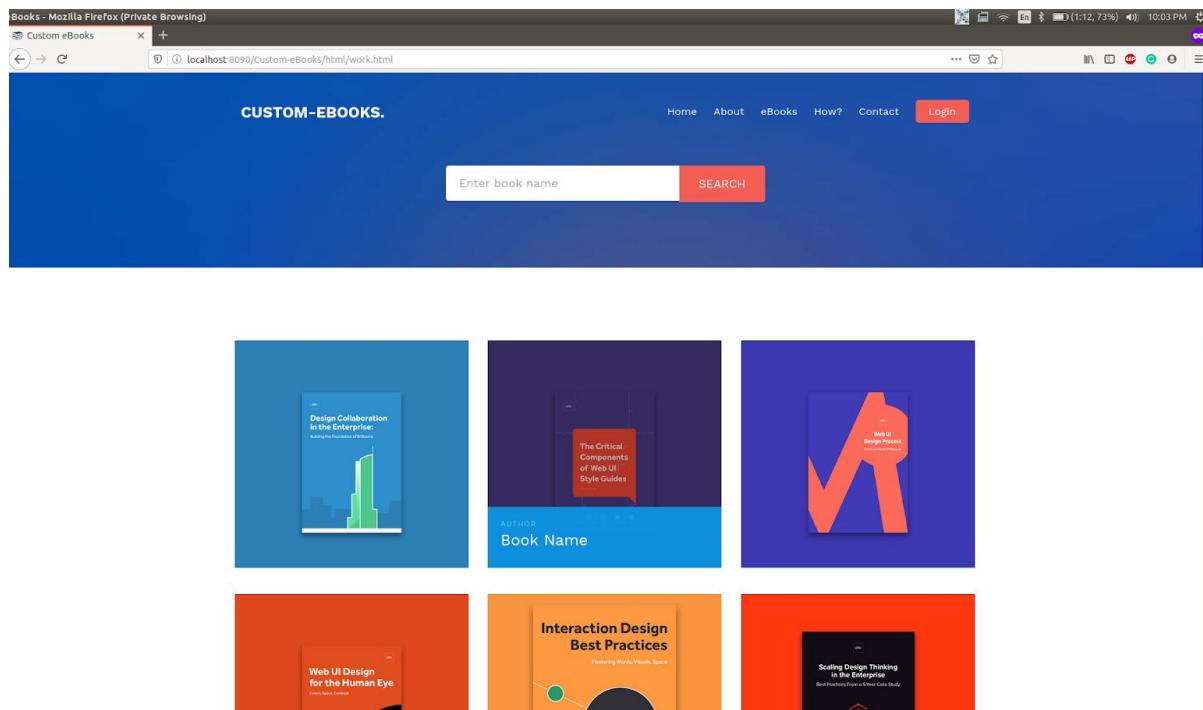


Additional functionalities :

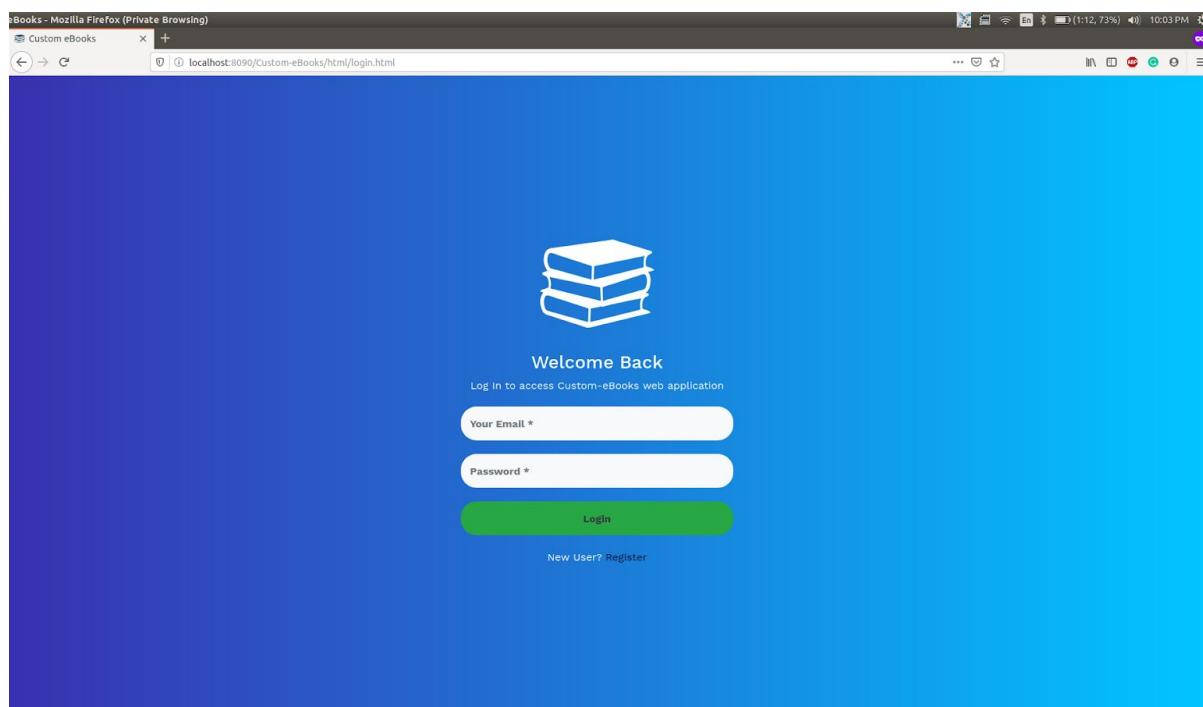
Customer can remove a added Chapter



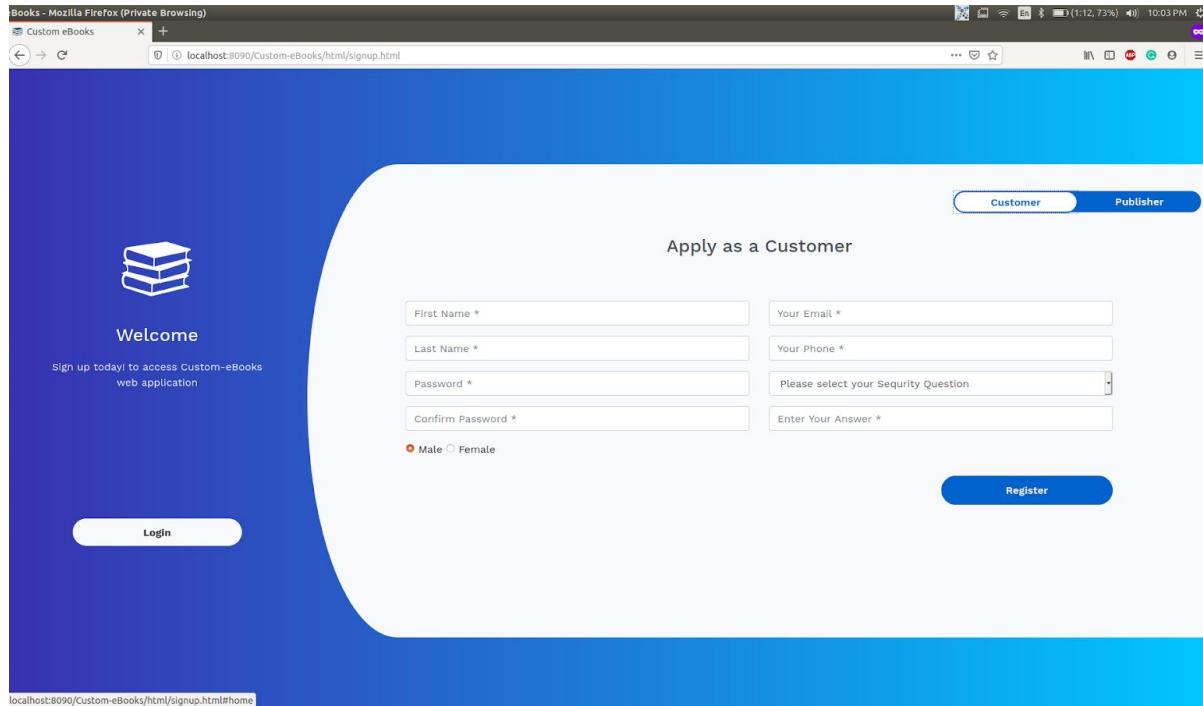
Generated Books are stored in eBooks page



Login Page :



Sign Up Page :



About Page :

The screenshot shows a Mozilla Firefox browser window with a blue header bar. The main content area has a white background with a blue gradient footer. On the left, there's a large 'ABOUT US' title in white. Below it are two rectangular placeholder boxes, one gray and one blue. To the right, there are two sections: 'History' and 'Mission & Vission'. The 'History' section contains text about the project's purpose and team. The 'Mission & Vission' section contains text about the project's goals and how it differs from traditional hard copies. The URL in the address bar is 'localhost:8090/Custom-eBooks/html/about.html'.

History

This web application is done as part of our Software Production Engineering course under the guidance of Prof.B Thangaraju and our mentor Valbhav Aggarwal. We are Group : 11

Sticking to a three level hierarchy of book, chapter and section will let us provide a uniform interface since most books will have at least this level of structure. This will allow publishers to easily add more content. Allowing the customer to define the structure of the custom book will make it easier for users to search for different components to put the custom book together.

Mission & Vission

This project sets out to solve the problem of generating e-books based on preferences associated with the user. More and more people are preferring ebooks to a hard copy. But the trend has been towards treating them like normal books in terms of their contents (fixed content).

The idea is that unlike with hard copies, we can have a lot of flexibility with ebooks. We can have the ability to create ebooks which are composable with smaller components, and the ability to select and assemble parts of many books to create them.

Our Team :

PRODUCTIVE CLASSMATES

Meet Us

We are Integrated M.Tech 2016 batch students from International Institute of Information Technology Bangalore (IIITB).



Soumith Kumar
Web Designer

Summer Analyst at Morgan Stanley. Former Intern at Siemens Technology India. Interested in Computer Vision, Data Analytics and Machine Learning.

[Facebook](#) [Twitter](#) [GitHub](#) [LinkedIn](#)



Rohith Yogi
Front-end Developer

EDG Summer Intern at Mathworks. Former Research Associate Intern at Accenture. Interested in Big Data, Software Development and Machine Learning.

[Facebook](#) [Twitter](#) [GitHub](#) [LinkedIn](#)



Nikhil Sai
Back-end Developer

Summer Intern at Qualcomm. Teaching Assistant at IIITB. Interested in Natural Language processing, Networking and Machine Learning.

[Facebook](#) [Twitter](#) [GitHub](#) [LinkedIn](#)



Arunaav Reddy
DevOps Engineer

Summer Intern at Bel Labs, Bangalore. Teaching Assistant at IIITB. Interested in Signal processing, Networking and Machine Learning.

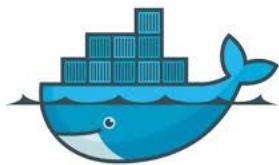
[Facebook](#) [Twitter](#) [GitHub](#) [LinkedIn](#)

GitHub :



<https://github.com/RohithYogi/Custom-eBooks/>

Docker Hub :



<https://hub.docker.com/repository/docker/bukkanikhilsai/tomex/general>

Future Work and Conclusion

Taking this application to production will include hosting this on the cloud and putting some security protocols in place. In terms of features, this can be treated as a complete product and there's scope for improvement and addition in many areas. Some of them include personal user space where a user can view the books he/she have uploaded and put together. Also there can be more lower levels of hierarchy (in our case it's a chapter) making things more flexible. The idea of custom assembled books changes the notion of how we look at books. This gives users a lot of flexibility in terms of being able to choose what should be in the custom book they are creating. This will enable us to make the most of flexibility ebooks provide in terms of not having a fixed content.

References

- <https://www.elastic.co/>
- <https://github.com/Alakazam03/ELK-Tutorial>
- <https://www.digitalocean.com/community/tutorials/how-to-install-elasticsearch-logstash-and-kibana-elastic-stack-on-ubuntu-16-04>
- <https://www.jenkins.io/doc/tutorials/build-a-java-app-with-maven/>
- <https://logz.io/blog/apache-tomcat-monitoring/>
- <https://logz.io/learn/complete-guide-elk-stack/>
- <https://www.codeproject.com/articles/143430/test-your-web-application-s-ui-with-junit-and-sele>
- <https://www.selenium.dev/maven/>
- <https://www.selenium.dev/selenium/docs/api/java/overview-summary.html>
- <https://junit.org/junit5/docs/current/user-guide/>
- <https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-16-04>

