

# **E-BOOK CREATE & STORE**

**A Project Report**

*Submitted by:*

*Minal Aggarwal(161218)*

**Under the guidance of: Dr. Amit Kumar**  
Assistant Professor (Senior Grade)



**June 2020**

*Submitted in partial fulfilment for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

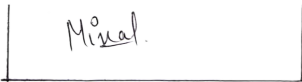
**Department of Computer Science & Engineering**

**JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,  
WAKNAGHAT, SOLAN-173234, HIMACHAL PRADESH**

## DECLARATION

I hereby declare that the project entitled “**E-BOOK CREATE & STORE**” submitted for the B. Tech. (CSE) degree is my original work and the project has not formed the basis for the award of any other degree, diploma, fellowship or any other similar titles.

Minal Aggarwal(161218)



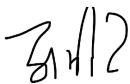
**Signature of the Student**

**Place:** Jaypee University of Information Technology, Solan(H.P.)

**Date:** 24 - 08 - 2020

## CERTIFICATE

This is to certify that the work titled “**E-BOOK CREATE & STORE**” submitted by “**Minal Aggarwal(161218)**” in partial fulfilment for the award of degree of **B-Tech** of Jaypee University of Information Technology, Solan has been carried out under my supervision. As per best of my knowledge and belief there is no infringement of intellectual property right and copyright. Also, this work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma. In case of any violation concern student will solely be responsible.



**Signature of Supervisor**

(Dr. Amit Kumar)

**Designation:** Assistant Professor (Senior Grade)

**Date:**

## **ABSTRACT**

In today's digital world where there is an endless variety of content to be consumed like books, videos, articles, movies etc. and finding the content of one's liking has become a difficult task.

On the other hand digital content providers want to engage as many users on their service as possible for the maximum time. This is where e-book system comes into picture where the content providers recommend users the content according to the users like. In this report we have proposed a E-book system and this report provides a detailed summary of the project. Our objective is to provide best e-books to users for reading or creating their own e-books. The report includes a description of the topic, system architecture, and provides various approaches and tools used so far.

## **ACKNOWLEDGEMENT**

When undergoing through the making of a certain project, there exists a lot of contribution and guidance of other people who support us in achieving what lies in front. Not only do they guide our path, but also play a vital role in the establishment and understanding of what we really want to achieve.

Thus, this is to show our gratitude towards our Mentor, Dr. Amit Kumar for guiding us through our entire Report. He supported us immensely in the better understanding of the project, guiding us towards the better knowledge of the project we were working on. The weekly interaction with him brought to us more clarification on the topic and thus simplifying our task.

## LIST OF FIGURES

<b>S.NO.</b>	<b>CAPTION</b>	<b>PAGE NO.</b>
Figure 1.1	MEAN Stack	3
Figure 1.2	MongoDB data scheme	5
Figure 4.1	Workflow Diagram	39
Figure 4.2	Dataflow Diagram Level-0	40
Figure 4.3	Dataflow Diagram Level-1	41

# CONTENTS

CAPTION	PAGE NO.
Title Page	i
Declaration of the Student	ii
Certificate of the Guide	iii
Abstract	iv
Acknowledgement	v
List of Figures	vi
<b>1. INTRODUCTION</b>	<b>1-2</b>
1.1 Overview	1
<b>2. BACKGROUND MATERIAL</b>	<b>3-9</b>
2.1 MEAN Stack	3
2.2 Introduction to NodeJS	3
2.3 Introduction to Express	4
2.4 Introduction to Angular	5
2.5 Introduction to MongoDB	5
2.6 Introduction to GITHUB and GIT	6
2.7 Introduction to Docker	6
2.8 Introduction to Modern Web Applications	8
<b>3. CONTRIBUTIONAL WORK</b>	<b>10-37</b>
<b>4. REQUIREMENT ANALYSIS&amp; DESIGN</b>	<b>38-41</b>
4.1 Requirement Specification	38
4.1.1 Soft Skills	38
4.1.2 Tools Used	38
4.2 Workflow Diagram	39
4.3 Dataflow Diagram	40
4.3.1 Level-0	40

4.3.2 Level-1	41
<b>5. CONCLUSION</b>	<b>42</b>
<b>6. REFERENCES</b>	<b>43</b>



# CHAPTER 1 – INTRODUCTION

Websites are one of the most effective ways to present and propagate information to people around the world. The web browser is a software application for retrieving, presenting and traversing information resources on the World Wide Web. More than ten years ago, a website was constructed with raw HTML document and served by simple HTTP request. The birth of CSS and JavaScript made web application become more and more convenient in displaying information. In this period, developing server side of a web application required developer know one of many programming languages, which are PHP, Java, C#. Otherwise, JavaScript is the primary scripting language to make reactive website. The development process is not an easy task for junior developers because they have to learn many programming languages along with client-side mark-up language.

## 1.1 PROJECT OVERVIEW

E-book create and store is a web-based application based on NodeJS, Angular & MongoDB. The main aim of our project to provide a platform for the user to read books that are available by using the reader in our web application, they can upload the book & they can write their book using the editor that we are providing on our web application.

The web application contains several features like login, signup, dashboard, reader, editor and uploader page. The web application has two main parts first one is the reader & another one is the editor. Readers of the book can rate and read the book. The web application contains one of the important feature is to track the progress percentage to user dashboard the aim of this feature is to notify the user how much content of the book he or she had read.

Our Editor is a word processor which means it is to designed for text-based documents. These can be a business report, student papers, informal notes for a call, lecture notes, letter to a friend, relative or it can be an official letter. Editor aimed to target that user who is either poet or author of the book, students, professors, teachers, office staff, etc. so that they can write their book and publish their e-book on the platform and other uses too. The editor can be used to enter text, format text, save the document and maintain compatibility.

The Dashboard is an information management page. It plays a crucial role in our web-application that can track the reading progress, rating of the books or several PDF available on the platform and it also

shows all the available books to the user on the application. It is user-friendly, easily understandable dashboard, real-time updates, save time & resources and decision making of users.

Our web application is hosted on the AWS Cloud platform by using its service known as EC2 stands for Elastic Compute Cloud that provides scalable computing capacity in the amazon web services cloud. The domain name of our web-based application is <http://ebookcs.bewithrits.tech>.

Travis CI is a hosted, distributed continuous integration service used to build and test projects hosted at GitHub. Travis CI automatically detects when a commit has been made and pushed to a GitHub repository that is using Travis CI, and each time this happens, it will try to build the project and run tests.

TravisCI is used to build docker images for the frontend and backend services, which are then pushed to Docker Hub Container Registry for storage.

Our AWS Server can then be manually updated to use the latest images.

## CHAPTER 2 - BACKGROUND MATERIAL

E-book create and store is a web-based application based on NodeJS, Angular & MongoDB which is MEAN stack based project, it also includes some cloud features, continuous integration and continuous deployment for hosting the application the application on AWS.

### 2.1. MEAN STACK

In a full stack web application, JavaScript is the consistent language and syntax. The MEAN stack is comprised of four main JavaScript oriented technologies. MongoDB is the database, Express is back-end web framework, AngularJS is front-end web framework and Node.js is the back-end runtime environment. The reason why the stack is selected because developers will have improved view of greater picture by understanding the different areas and how they fit together. Frameworks and libraries are no longer limited in supporting user interfaces but also can be combined with other layers of applications. However, wrong architecture, low-level design and unbalance foundation in initial construction can dramatically affect the development process (Linnovate, 2014).



Fig 1.1: MEAN Stack

### 2.2. INTRODUCTION TO NodeJs

Node.js is a software platform allowing developers to create web servers and build web applications on it. Meanwhile PHP needs to run separate web server program such as Apache or Internet Information Services, Node.js server can be configured when building an application. However, it is not limited in web projects. More phone apps and desktop applications are built on top of Node.js. (Node.js Foundation, 2012).

Node.js provides many utilities' packages in order to create a powerful web server that runs JavaScript on the server side, for example, asynchronous I/O, single-threaded or multi-threaded, sockets, and HTTP connections. When building web applications using other server-side programming languages, such as Java, PHP, and .NET, the server creates a new thread for each new connection. Assumed that there is a popular website receiving more than two million requests daily, developers probably need more servers or even to invest in more hardware. This approach works well as long as the server has enough hardware infrastructure to provide services to the customers. When the number of clients exceed the server's ability, it starts to slow down and the customers have to wait (Holmes, 2015).

The single threaded, non-blocking I/O changes how a request is handled by the server. Instead of creating a new thread for each connection, the web server receives client requests and places them in a queue, which is known as Event Queue. The Event Loop picks up one client request from the queue and starts process it. If the request does not include any blocking I/O operation, the server process everything, creates a response and sends it back to the client. In the case that the request can block the I/O operation, there is a different approach. An available thread from the thread pool will be picked up and assigned to the client request. That thread has then processed the blocking I/O, processed the request and created a response to send back to the client (Jörg, 2016). Node servers can support tens of thousands of simultaneous connections. It does not allocate one thread per connection model, but uses a model process per connection, creating only the memory that is required for each connection (Monteiro 2014, 32). Grap 3 explains single threaded application.

## **2.3 INTRODUCTION TO EXPRESS**

Even though Node.js is a great choice to make a web server, it is just a run time environment with many built-in modules. The default Node API is not able to handle complex applications as route management. A web framework is needed to do the heavy work. Express is a mature and flexible web framework to build web applications on top of the Node eco system. By default, the Express framework uses the Pug engine for supporting templates (Node.js Foundation, 2010).

Furthermore, Express reduces difficulties in setting up a webserver to handle incoming requests and return relevant response. That is the reason why it is the best choice to render HTML on the server side for large-scale application. It still provides an elegant set of features to deal with including SPA, the RESTful API, and the MEAN stack. Express follows good development practices, an oriented RESTful

architecture that uses the main methods of the HTTP protocol (such as GET, POST, PUT, and DELETE), and very difficult-to-build complex web applications (Monteiro, 2014, 39).

## 2.4. INTRODUCTION TO ANGULAR

At the moment, there is a enormous amount of front end libraries and frameworks supporting front end web development. AngularJS is one of them, which is designed to work with data directly in the front end. It allows developers to use HTML as template language and extend HTML's syntax. There are two core features: Data binding and Dependency injection. AngularJS reduces most of codes, which are needed to write. Developers can use Node.js, Express, MongoDB to build a data-drive web application and pass data via HTTP protocols. Angular works with data directly in the frontend and puts the HTML templates together based on the provided data. Developers need less resources with this approach (Google, 2010).

## 2.5. INTRODUCTION TO MONGO DB

MongoDB is one of the most popular open source NoSQL databases written by C++. In 02/2015, MongoDB ranked at 4<sup>th</sup> in list of common databases. MongoDB is document-oriented, and it stores data in the key-value format, using binary JSON (BSON). It is agile, scalable and high performance. Node.js. and MongoDB unleashed all the power for high-performance web applications with Express (MongoDB Inc, 2008)

```
{  
  name: "sue",  
  age: 26,  
  status: "A",  
  groups: [ "news", "sports" ]  
}
```



The diagram shows a JSON document with four fields: 'name', 'age', 'status', and 'groups'. To the right of each field name, there is a blue arrow pointing left towards the field name, and next to it is the text 'field: value' in blue. This illustrates that each field in the document is treated as a key-value pair.

Fig 1.2: MongoDB data scheme (MongoDB Inc, 2008)

Document in MongoDB has the same structure as JSON, which means developers can map data as a key-value. We can understand the document as a record in MYSQL. Because MongoDB is written in C++, it is able to make a high-speed calculation, unlike other database management systems. Instead of writing a massive amount of SQL commands, MongoDB supports fairly complex and powerful queries in most use cases (MongoDB Inc, 2008).

## 2.6 INTRODUCTION TO GITHUB AND GIT

GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere. A repository is usually used to organize a single project. Repositories can contain folders and files, images, videos, spreadsheets, and data sets – anything your project needs. We recommend including a README, or a file with information about your project. GitHub makes it easy to add one at the same time you create your new repository.

Branching is the way to work on different versions of a repository at one time.

By default, our repository has one branch named master which is considered to be the definitive branch. We use other branches to experiment and make edits before committing them to master.

A version Control system is a system which maintains different versions of your project when we work in a team or as an individual. (system managing changes to files) As the project progresses, new features get added to it.

Distributed Version control system means every collaborator (any developer working on a team project) has a local repository of the project in his/her local machine unlike central where team members should have an internet connection to every time update their work to main central repository.

Some commands we mostly had used :

```
git add .
```

```
git commit -m "message"
```

```
git push origin master
```

## 2.7. INTRODUCTION TO DOCKER

Docker, represented by a logo with a friendly looking whale, is an open source project that facilitates deployment of applications inside of software containers. Its basic functionality is enabled by resource isolation features of the Linux kernel, but it provides a user-friendly API on top of it. The first version was released in 2013, and it has since become extremely popular and is being widely used by many big players such as eBay, Spotify, Baidu, and more. In the last funding round, Docker has landed a huge \$95 million, and it's on track to become a staple of DevOps services.

At a quick glance, virtual machines and Docker containers may seem alike. Applications running in virtual machines, apart from the hypervisor, require a full instance of the operating system and any supporting libraries. Containers, on the other hand, share the operating system with the host. Hypervisor is comparable to the container engine (represented as Docker on the image) in a sense that it manages

the lifecycle of the containers. The important difference is that the processes running inside the containers are just like the native processes on the host, and do not introduce any overheads associated with hypervisor execution. Additionally, applications can reuse the libraries and share the data between containers.

As both technologies have different strengths, it is common to find systems combining virtual machines and containers.

By default, the main registry is the Docker Hub which hosts public and official images. Organizations can also host their private registries if they desire. On the right-hand side we have images and containers. Images can be downloaded from registries explicitly ( `docker pull imageName` ) or implicitly when starting a container. Once the image is downloaded it is cached locally.

Containers are the instances of images - they are the living thing. There could be multiple containers running based on the same image.

There is the Docker daemon responsible for creating, running, and monitoring containers. It also takes care of building and storing images. Finally, on the left-hand side there is a Docker client. It talks to the daemon via HTTP. Unix sockets are used when on the same machine, but remote management is possible via HTTP based API.

Before jumping in and building your own Docker image, it's a good practice to first check if there is an existing one in the Docker Hub or any private registries you have access to. For example, instead of installing Java ourselves, we will use an official image: `node.9`.

To build an image, first we need to decide on a base image we are going to use. It is denoted by *FROM* instruction. Here, it is an official image for Java 8 from the Docker Hub. We are going to copy it into our Java file by issuing a *COPY* instruction. Next, we are going to compile it with *RUN*. *EXPOSE* instruction denotes that the image will be providing a service on a particular port. *ENTRYPOINT* is an instruction that we want to execute when a container based on this image is started and *CMD* indicates the default parameters we are going to pass to it.

```
## Specifies the base image we're extending
FROM node:9

## Create base directory
RUN mkdir /src

## Specify the "working directory" for the rest of the Dockerfile
WORKDIR /src

## Install packages using NPM 5 (bundled with the node:9 image)
COPY ./package.json /src/package.json
COPY ./package-lock.json /src/package-lock.json
RUN npm install --silent

## Add application code
COPY ./app /src/app
COPY ./bin /src/bin
COPY ./public /src/public

## Add the nodemon configuration file
COPY ./nodemon.json /src/nodemon.json

## Set environment to "development" by default

ENV NODE_ENV development

## Allows port 3000 to be publicly available
EXPOSE 3000

## The command uses nodemon to run the application
CMD ["node", "node_modules/.bin/nodemon", "-L", "bin/www"]
```

## **2.8 INTRODUCTION MODERN WEB APPLICATIONS**

The strategy in web development has changed dramatically in the recent years, thanks to the improvement of information technology. The browser is no longer used to serve static information. JavaScript is used heavily to serve dynamic elements in the browser. The browser is becoming a kind of mini-operating system, which avails itself in the net of various data sources- the services of developer's servers.

The modern web application, which is called web app, connects to the server to retrieve data dynamically. It only exists in the browser. When users approach the app for the first time, the app is rendered by the server and supports its services such as database access or transaction. Deriving from



modern web application, E-book create & store project is designed to build application service provider called an APIs (Application Programming Interfaces) and use JSON (JavaScript Object Notation) as formatting language.

Atwood (2017), as a collar to Tim Barnes-Lee's Rule of Least Power, proposed Atwood's Law which states that any application that can be written in JavaScript, will eventually be written in JavaScript. Ryan Dahl (2009), wrote Node.js, an open-source, cross-platform JavaScript runtime environment. After several years of enhancement, it has combined with Google's V8 JavaScript engine, an event loop and low-level I/O API, and has supported executing JavaScript code in the server side.

## CHAPTER 3 - CONTRIBUTIONAL WORK

Our project E-book create and store is divided into Front-end, Back-end, deployment and hosting the project on the internet. We had create our project repository on github.

### 3.1.DOCKER FILE

```
File - \Dockerfile

FROM node:8.16.0 as backendBuilder

WORKDIR /apps/ah

COPY package.json ./

RUN npm install

COPY . .

RUN npm run sdk

CMD ["npm", "start"]

FROM node:10.12.0-alpine as clientBuilder

WORKDIR /apps/ah

COPY ./client/package.json ./

RUN npm install

COPY ./client ./

COPY --from=backendBuilder/apps/ah/client/src/app/shared/lb-sdk ./src/
app/shared/lb-sdk

RUN npm run build

FROM nginx:stable as rpBuilder

WORKDIR /usr/share/nginx/html

COPY --from=clientBuilder/apps/ah/dist/usr/share/nginx/html

COPY ./reverse-proxy/nginx.prod.conf/etc/nginx/conf.d/default.conf

File - \.travis.yml
```

### 3.2.TRAVIS YML

```
language: ruby

services:
```

```

- docker

matrix:

fast_finish: true

env:

global:

- COMMIT=${TRAVIS_COMMIT::8}

script:

- echo "Ritik@123" | docker login -u "161b178" --password-stdin

- docker build --target backendBuilder -t 161b178/ebook-backend:${COMMIT} -t 161b178/ebook-backend:latest .

- docker build --target rpBuilder -t 161b178/ebook-proxy:${COMMIT} -t 161b178/ebook-proxy:latest .

- docker push 161b178/ebook-backend:${COMMIT}

- docker push 161b178/ebook-backend:latest

- docker push 161b178/ebook-proxy:${COMMIT}

- docker push 161b178/ebook-proxy:latest

```

### 3.3. PACKAGE.JSON FILE

File - \package.json

```

{
  "name": "ebook-create-store",
  "version": "1.0.0",
  "main": "server/server.js",
  "engines": {
    "node": ">=6"
  },
  "scripts": {
    "lint": "eslint .",
    "start": "node .",
    "install:server": "npm install",
    "install:client": "cd client && npm install",

```

```

"dev": "./node_modules/nodemon/bin/nodemon.js --ignore 'docker/' --
ignore 'uploads/' --ignore 'client/' server/server.js",
"sdk": "DEBUG=boot:,common:models:,server:* SKIP_SEEDING=true ./
node_modules/.bin/lb-sdk server/server.js client/src/app/shared/lb-sdk -i
enabled -w enabled",
"setup": "run-s install:serverinstall:clientsdk"
},
"dependencies": {
"@mean-expert/loopback-sdk-builder": "^2.3.1",
"compression": "^1.0.3",
"cors": "^2.5.2",
"font-awesome": "^4.7.0",
"helmet": "^3.10.0",
"html-pdf": "^2.2.0",
"loopback": "^3.22.0",
"loopback-boot": "^2.6.5",
"loopback-component-explorer": "^6.2.0",
"loopback-connector-mongodb": "^5.2.0",
"loopback-ds-timestamp-mixin": "^3.4.1",
"multer": "^1.4.2",
"serve-favicon": "^2.0.1",
"strong-error-handler": "^3.0.0"
},
"devDependencies": {
"eslint": "^3.17.1",
"eslint-config-loopback": "^8.0.0",
"nodemon": "^2.0.1",
"npm-run-all": "^4.1.5"
},
"repository": {
"type": "",

```

```
"url": ""  
  
},  
  
"license": "UNLICENSED",  
  
"description": "ebook-create-store"  
  
}
```

File - \docker-compose.yml

### 3.4 DOCKER COMPOSE FILE

```
version: "3.2"  
  
services:  
  
  client:  
  
    image: node:10.12.0-alpine  
  
    working_dir: /apps/ah  
  
    command: npm start -- --host 0.0.0.0 --disable-host-check  
  
    ports:  
  
    - "4200:4200"  
  
    volumes:  
  
    - type: bind  
  
      source: ./client  
  
      target: /apps/ah  
  
    reverse-proxy:  
  
      image: nginx:stable  
  
      ports:  
  
      - "80:80"  
  
      volumes:  
  
      - type: bind  
  
        source: ./reverse-proxy/nginx.ah.conf  
  
        target: /etc/nginx/conf.d/default.conf  
  
      - type: bind  
  
        source: ./wait-for  
  
        target: /wait/wait-for
```

```
depends_on:

- web

- client

web:

image: node:8.16.0

working_dir: /apps/ah

ports:

- "3000:3000"

environment:

NODE_ENV: local

env_file: .env

command: npm run dev

volumes:

- type: bind

source: .

target: /apps/ah

depends_on:

- db

db:

image: mongo:4.1

ports:

- "27017:27017"

volumes:

- type: bind

source: ./docker/volumes/mongo

target: /data/db

File - \client\src\index.html
```

### **3.5.ANGULAR – COMPONENTS FILE**

```
<!doctype html>

<html lang="en">
```

```
<head>

<meta charset="utf-8">

<title>EBook - Create And Store</title>

<base href="/">

<meta content="width=device-width, initial-scale=1" name="viewport">

<link href="favicon.ico" rel="icon" type="image/x-icon">

</head>

<body>

<app-root></app-root>

</body>

</html>
```

File - \client\src\styles.scss

```
$enable-flex: true;

$enable-grid-classes: true;

@import '~bootstrap/scss/bootstrap-grid';

@import '~ngx-bar-rating/themes/br-default-theme';

//@import "~bulma/css/bulma.min.css";

.ngx-editor-textarea {

min-height: 300px !important;

background-color: white !important;

}

.router-link-active{

color: blueviolet;

}
```

File - \client\src\app\app.module.ts

```
import {BrowserModule} from '@angular/platform-browser';

import {NgModule} from '@angular/core';

import {NgxEditorModule} from 'ngx-editor';

import {PdfViewerModule} from 'ng2-pdf-viewer';
```

```

import {AppRoutingModule} from './app-routing.module';

import {AppComponent} from './app.component';

import {SharedModule} from './shared/shared.module';

import {ReaderComponent} from './reader/reader.component';

import {DashboardComponent} from './dashboard/dashboard.component';

import {BookComponent} from './dashboard/book/book.component';

import {EditorComponent} from './editor/editor.component';

import {BarRatingModule} from "ngx-bar-rating";

import {FormsModule} from '@angular/forms';

import {TabModule} from "angular-tabs-component";

@NgModule({

declarations: [

AppComponent,

ReaderComponent,

DashboardComponent,

BookComponent,

EditorComponent

],

imports: [

BrowserModule,

AppRoutingModule,

SharedModule,

PdfViewerModule,

NgxEditorModule,

BarRatingModule,

FormsModule,

TabModule

],

providers: [],

exports: [],

bootstrap: [AppComponent]

```



```

    })

    export class AppModule {

    }

    File - \client\src\app\auth.guard.ts

    import {Injectable} from '@angular/core';

    import {ActivatedRouteSnapshot, CanActivate, Router, RouterStateSnapshot,
    UrlTree} from '@angular/router';

    import {Observable} from 'rxjs';

    import {LoopBackAuth} from './shared/lb-sdk/services/core';

    @Injectable({
    providedIn: 'root'
    })

    export class AuthGuardimplements CanActivate {

    constructor(private authService: LoopBackAuth,
    private router: Router,
    ) {

    }

    canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean> | Promise<
    boolean> | boolean {

    return this.isAuthenticated(state.url, next);

    }

    isAuthenticated(url, route: ActivatedRouteSnapshot): boolean {

    constuserId = this.authService.getCurrentUserId();

    if (!userId) {

    this.router.navigate(['auth', 'login'], {queryParams: {
    redirectUrl: url}});

    } else {

    if (route.data&&route.data.roles) {

    constrole = this.authService.getCurrentUserData().roles[

```

```

0].name;

const isAuthorized = route.data.roles.includes(role);

// if (!isAuthorized)

// this.router.navigate(['/']);

return isAuthorized;

}

return true;

}

}

}

```

File - \client\src\app\app.component.ts

```

import {Component, OnInit} from '@angular/core';
import {LoopBackAuth} from '../shared/lb-sdk/services/core';
import {UserModelApi} from '../shared/lb-sdk/services/custom';
import {UserModel} from '../shared/lb-sdk/models';
import {NavigationEnd, Router} from '@angular/router';
import { LoopBackConfig } from '../shared/lb-sdk';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
})
export class AppComponent implements OnInit {
  isAuthenticated: boolean = false;
  user: UserModel;
  navExpanded = true;
  constructor(
    private authService: LoopBackAuth,
    private userApi: UserModelApi,
    private router: Router
  ) {

```

```

}

ngOnInit(): void {
  LoopBackConfig.setBaseUrl(location.origin);

  this.isAuthenticated = !!this.authService.getCurrentUserId();
  this.user = <UserModel>this.authService.getCurrentUserData();

  this.router.events.subscribe((val) => {
    if (val instanceof NavigationEnd) {
      this.isAuthenticated = !!this.authService.
        getCurrentUserId();
      this.user = <UserModel>this.authService.
        getCurrentUserData();
    }
  });
}

logout() {
  this.authService.clear();
  this.router.navigate(['auth', 'login'])
}
}

File - \client\src\app\auth\signup\signup.component.

<div *ngIf="error" class="notification is-danger">
  {{error}}
</div>

<progress *ngIf="!!loading" class="progress is-small is-primary" max="100
">15%
</progress>

<div class="box">
  <h3 class="subtitle is-3">
    Sign Up
  </h3>
  <div class="container">

```

```

<label for="uname" class="subtitle is-6"><b>First Name</b></label>
<input [(ngModel)]="firstName" class="input is-info" type="text"
placeholder="First Name" name="uname" required>
<br>
<label for="uname" class="subtitle is-6"><b>Last Name</b></label>
<input [(ngModel)]="lastName" class="input is-info" type="text"
placeholder="Last Name" name="lname" required>
<br>
<label for="email" class="subtitle is-6"><b>Email</b></label>
<input [(ngModel)]="email" class="input is-info" type="text"
placeholder="Email" name="email" required>
<br>
<!--<p *ngIf="/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/\.test (
email) == false">
enter correct email
</p> -->
<label for="psw" class="subtitle is-6"><b>Password</b></label>
<input [(ngModel)]="password" class="input is-info" type="password"
placeholder="Password" name="psw" required>
<br>
<label for="cpsw" class="subtitle is-6"><b>Confirm Password</b></
label>
<input [(ngModel)]="confirmPass" class="input is-info" type="password
" placeholder="Confirm Password" name="cpsw" required>
<p *ngIf="password!=confirmPass">
password doesn't match
</p>
<p *ngIf="password.length< 8">
password length must be greater than or equal to 8.
</p>
<br>

```

```

<button *ngIf="password == confirmPass&&password.length>= 8" class
="button is-success is-light" type="submit" (click)="registerUser()">Sign
Up</button>

<br>

</div>

</div>

```

### 3.6.BACKEND

```

const multer = require('multer');
const fs = require('fs');
const path = require('path');
var pdf = require('html-pdf');
var options = {format: 'Letter'};
module.exports = function(Books) {
var uploadedFileName = '';
var storage = multer.diskStorage({
destination: function(req, file, cb) {
// checking and creating uploads folder where files will be
uploaded
var dirPath = 'uploads/';
if (!fs.existsSync(dirPath)) {
var dir = fs.mkdirSync(dirPath);
}
cb(null, dirPath + '/');
},
filename: function(req, file, cb) {
// file will be accessible in `file` variable
var ext = file.originalname.substring(file.originalname.lastIndexOf
('.'));
var fileName = Date.now() + ext;
uploadedFileName = 'uploads/' + fileName;

```

```

cb(null, fileName);

},

});

Books.remoteMethod('addByPDF', {
  accepts: [
    {arg: 'title', type: 'string', required: true},
    {arg: 'req', type: 'object', 'http': {source: 'req'}},
    {arg: 'res', type: 'object', 'http': {source: 'res'}},
  ],
  returns: {root: true, type: 'object'},
  http: {path: '/addByPDF', verb: 'post'},
});

Books.addByPDF = function addByPDF(title, req, res, cb) {
  var upload = multer({
    storage: storage,
    onError: function(err, next) {
      console.log('error', err);
      next(err);
    },
  }).single('file');
  upload(req, res, function(err) {
    if (err) {
      // An error occurred when uploading
      console.log(err);
      return cb(err);
    }
    return Books.create({
      title: title,
      url: '/api/' + uploadedFileName,
      rating: 0,
    })
  })
}

```

```

.then(book => {
console.log(book);
cb(null, book);
})
.catch(err => {
File - \common\models\books.js
Page 2 of 2
console.log(err);
cb(err);
});
});
};
Books.remoteMethod('addByContent', {
accepts: [
{arg: 'title', type: 'string', required: true, http: {source: 'form
'}},
{arg: 'content', type: 'string', required: true, http: {source: '
form'}}},
],
returns: {root: true, type: 'object'},
http: {path: '/addByContent', verb: 'post'},
});
Books.addByContent = function addByContent(title, content, cb) {
constbookName = Date.now() + '.pdf';
pdf.create(content, options).toFile(path.join(__dirname, '..', '..',
'uploads', bookName), function(err, res) {
if (err) {
return cb(err);
}
console.log(res); // { filename: '/app/businesscard.pdf' }
Books.create({

```

```

title: title,
url: '/api/uploads/' + bookName,
rating: 0,
})

.then(book => {
console.log(book);
cb(null, book);
})

.catch(err => {
console.log(err);
cb(err);
});
});
};
};

```

File - \common\models\books.json

```

{
"name": "Books",
"base": "PersistedModel",
"idInjection": true,
"options": {
"validateUpsert": true
},
"properties": {
"title": {
"type": "string",
"required": true
},
"content": {
"type": "string"

```



```
    },
    "url": {
      "type": "string",
      "required": true
    },
    "rating": {
      "type": "number"
    }
  },
  "validations": [],
  "relations": {
    "userModel": {
      "type": "belongsTo",
      "model": "UserModel",
      "foreignKey": ""
    },
    "userBookProgress": {
      "type": "hasMany",
      "model": "UserBookProgress",
      "foreignKey": "booksId"
    }
  },
  "mixins": {
    "TimeStamp": true
  },
  "acls": [],
  "methods": {}
}
File - \common\models\user-model.js

'use strict';
```

```

module.exports = function(UserModel) {
  UserModel.remoteMethod('markBookRead', {
    accepts: [
      {arg: 'id', type: 'string', required: true}, // get the id of the
      frog to save image to
      {arg: 'bookId', type: 'string', required: true}, // pass the
      request object to remote method
    ],
    returns: {root: true, type: 'object'},
    http: {path: '/:id/markBookRead', verb: 'post'},
  });
  UserModel.markBookRead = async function upload(id, bookId) {
    const userBook = await UserModel.app.models.UserBookProgress.findOne(
      {
        where: {
          booksId: bookId,
          userModelId: id,
        },
      });
    console.log(userBook);
    if (!userBook) {
      return UserModel.app.models.UserBookProgress.create({
        userModelId: id,
        booksId: bookId,
      });
    } else {
      return Promise.resolve(userBook);
    }
  };
};
};

File - \common\models\user-model.json

```

```
{
  "name": "UserModel",
  "base": "User",
  "idInjection": true,
  "options": {
    "validateUpsert": true
  },
  "properties": {
    "firstName": {
      "type": "string",
      "required": true,
      "default": ""
    },
    "lastName": {
      "type": "string",
      "required": true,
      "default": ""
    }
  },
  "validations": [],
  "relations": {
    "userBookProgress": {
      "type": "hasMany",
      "model": "UserBookProgress",
      "foreignKey": "userModelId"
    },
    "books": {
      "type": "hasMany",
      "model": "Books",
      "foreignKey": "userModelId"
    }
  }
}
```

```

}

},

"acls": [

{

"principalType": "ROLE",

"principalId": "$owner",

"permission": "ALLOW",

"property": "*"

},

{

"principalType": "ROLE",

"principalId": "$everyone",

"permission": "ALLOW",

"property": "create"

},

{

"principalType": "ROLE",

"principalId": "$everyone",

"permission": "ALLOW",

"property": "login"

},

{

"principalType": "ROLE",

"principalId": "$owner",

"permission": "ALLOW",

"property": "__get__userBookProgress"

}

],

"methods": {}

}

```

File - \common\models\user-book-progress.js

```

'use strict';

module.exports = function(UserBookProgress) {

UserBookProgress.observe('after save', async function(ctx, next) {

let id;

if (ctx.instance) {

id = ctx.instance.id;

} else {

id = ctx.where.id;

}

console.log({id});

if (id) {

constprogress = await UserBookProgress.findOne({

where: {

id,

},

});

console.log({progress});

constratings = await UserBookProgress.find({

where: {

booksId: progress.booksId,

},

fields: 'rating',

});

constrating = ratings.reduce((total, num) => {

return {rating: total.rating + num.rating}; // returns object

with property x

}).rating / ratings.length;

console.log({rating});

await UserBookProgress.app.models.Books.update({

id: progress.booksId,

```

```
}, {  
  rating: rating ? rating : 0,  
});  
return Promise.resolve();  
}  
});  
};
```

File - \common\models\user-book-progress.json

```
{  
  "name": "UserBookProgress",  
  "base": "PersistedModel",  
  "idInjection": true,  
  "options": {  
    "validateUpsert": true  
  },  
  "properties": {  
    "progress": {  
      "type": "number",  
      "default": 0  
    },  
    "rating": {  
      "type": "number"  
    }  
  },  
  "validations": [],  
  "relations": {  
    "userModel": {  
      "type": "belongsTo",  
      "model": "UserModel",  
      "foreignKey": ""  
    }  
  }  
}
```

```

},
"books": {
  "type": "belongsTo",
  "model": "Books",
  "foreignKey": ""
}
},
"acls": [],
"methods": {}
}
File - \server\server.js

```

```

// Copyright IBM Corp. 2016. All Rights Reserved.
// Node module: loopback-workspace
// This file is licensed under the MIT License.
// License text available at https://opensource.org/licenses/MIT
'use strict';
var loopback = require('loopback');
var boot = require('loopback-boot');
var app = module.exports = loopback();
app.start = function() {
  // start the web server
  return app.listen(function() {
    app.emit('started');
    var baseUrl = app.get('url').replace(/\/$/, '');
    console.log('Web server listening at: %s', baseUrl);
    if (app.get('loopback-component-explorer')) {
      var explorerPath = app.get('loopback-component-explorer').mountPath;
      ;
      console.log('Browse your REST API at %s%s', baseUrl, explorerPath);
    }
  });
}

```

```
});  
  
};  
  
// Bootstrap the application, configure models, datasources and  
middleware.  
  
// Sub-apps like REST API are mounted via boot scripts.  
boot(app, __dirname, function(err) {  
  if (err) throw err;  
  
  // start the server if ` $ node server.js`  
  if (require.main === module)  
    app.start();  
  
});  
File - \server\config.json
```

```
{  
  "restApiRoot": "/api",  
  "host": "0.0.0.0",  
  "port": 3000,  
  "remoting": {  
    "context": false,  
    "rest": {  
      "handleErrors": false,  
      "normalizeHttpPath": false,  
      "xml": false  
    },  
    "json": {  
      "strict": false,  
      "limit": "100mb"  
    },  
    "urlencoded": {  
      "extended": true,  
      "limit": "100mb"
```



```
},  
"cors": false  
}  
}
```

File - \server\middleware.json

```
{  
  "initial:before": {  
    "loopback#favicon": {}  
  },  
  "initial": {  
    "compression": {},  
    "cors": {  
      "params": {  
        "origin": "*",  
        "credentials": true,  
        "maxAge": 86400  
      }  
    },  
    "helmet#xssFilter": {},  
    "helmet#frameguard": {  
      "params": {  
        "action": "deny"  
      }  
    },  
    "helmet#hsts": {  
      "params": {  
        "maxAge": 0,  
        "includeSubdomains": true  
      }  
    }  
  },  
}
```

```
"helmet#hidePoweredBy": {},
"helmet#ieNoOpen": {},
"helmet#noSniff": {},
"helmet#noCache": {
  "enabled": false
}
},
"session": {},
"auth": {},
"parse": {},
"routes": {
  "loopback#rest": {
    "paths": [
      "${restApiRoot}"
    ]
  }
},
"files": {},
"final": {
  "loopback#urlNotFound": {}
},
"final:after": {
  "strong-error-handler": {}
}
}
File - \server\datasources.json

{
  "db": {
    "host": "db",
    "port": 27017,
```

```

"database": "ebook",

"name": "db",

"connector": "loopback-connector-mongodb"

}

}

```

File - \server\boot\root.js

```

'use strict';

module.exports = function(server) {

// Install a '/' route that returns server status

var router = server.loopback.Router();

router.get('/', server.loopback.status());

server.use(router);

};

```

File - \server\boot\03\_rest-api.js

```

const path = require('path');

const fileName = path.basename(__filename, '.js'); // gives the filename
without the .js extension

module.exports = function mountRestApi(server) {

var restApiRoot = server.get('restApiRoot');

const dir = path.join(__dirname, '..', '..', 'uploads');

server.use('/api/uploads', server.loopback.static(dir));

server.use(restApiRoot, server.loopback.rest());

};

```

File - \server\boot\01\_authentication.js

```

'use strict';

module.exports = function enableAuthentication(server) {

// enable authentication

server.enableAuth();

};

```

### 3.8. REVERSE PROXY – NGINX FILE

File - \reverse-proxy\nginx.ah.conf

```
upstream webbackend {

server web:3000;

}

upstream clientdev {

server client:4200;

}

server {

listen 80 default;

server_name _;

client_max_body_size 200M;

location /explorer/ {

proxy_pass http://webbackend/explorer/;

proxy_http_version 1.1;

proxy_set_header Upgrade $http_upgrade;

proxy_set_header Connection 'upgrade';

proxy_set_header Host $host;

proxy_set_header X-Real-IP $remote_addr;

proxy_set_header X-Forwarded-For

$proxy_add_x_forwarded_for;

proxy_set_header X-Forwarded-Proto $scheme;

proxy_cache_bypass $http_upgrade;

}

location /api/ {

proxy_pass http://webbackend;

proxy_http_version 1.1;

#This deals with the Aggregating issue

chunked_transfer_encoding off;

proxy_buffering off;

proxy_cache off;
```

```
#This deals with the connection closing issue

proxy_read_timeout 13000;

proxy_set_header Upgrade $http_upgrade;

proxy_set_header Connection 'upgrade';

proxy_set_header Host $host;

proxy_set_header X-Real-IP $remote_addr;

proxy_set_header X-Forwarded-For

$proxy_add_x_forwarded_for;

proxy_set_header X-Forwarded-Proto $scheme;

proxy_cache_bypass $http_upgrade;

}

location / {

proxy_pass http://clientdev/;

proxy_http_version 1.1;

proxy_set_header Upgrade $http_upgrade;

proxy_set_header Connection 'upgrade';

proxy_set_header Host $host;

proxy_set_header X-Real-IP $remote_addr;

proxy_set_header X-Forwarded-For

$proxy_add_x_forwarded_for;

proxy_set_header X-Forwarded-Proto $scheme;

proxy_cache_bypass $http_upgrade;

}

}
```

# CHAPTER 4-REQUIREMENT ANALYSIS AND DESIGN

System analysis is a process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components. System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem-solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose. Analysis specifies what the system should do.

System design is a process of planning a new business system or replacing an existing system by defining its components or modules to satisfy the specific requirements. Before planning, you need to understand the old system thoroughly and determine how computers can best be used in order to operate efficiently. System Design focuses on how to accomplish the objective of the system.

## 4.1 REQUIREMENT SPECIFICATION

### 4.1.1 Soft Skills:

1. Web development
2. Angular
3. NodeJs

### 4.1.2 Tools used:

1. Docker
2. NGINX
3. AWS

## 4.2 WORK FLOW DIAGRAM

Workflow diagram depicts a series of actions that define a job or how work should be done. A workflow diagram visualizes how tasks will flow between resources, whether they're machines or people and what conditions allow the sequence to move forward. This workflow can be illustrated or described with a flowchart using abstract boxes and diamonds or it can be created with depictions of real-life objects using graphics and pictures that represent customers, forms, finance, products, shipping, payments, and more. For software development, a workflow diagram defines a series of steps a process must execute consistently.

A workflow chart is commonly used for documentation and implementation purposes since it provides a general overview of a business process. It's often the foundation for other documentation including flowcharts, data flow diagrams, projects, and more.

Each step in the workflow is represented with a pictorial symbol or an abstract shape like a box. The steps are connected with arrows that indicate the flow from beginning to end.

Work flow diagram for user is shown in figure4.1

- New user: The user is logged in/ registered. If he/she is a registered user then they'll be taken to E-BOOK CREATE & STORE Dashboard else login page will appear.
- E-BOOK CREATE & STORE Dashboard: This will list out all the books available that can be read or edited.

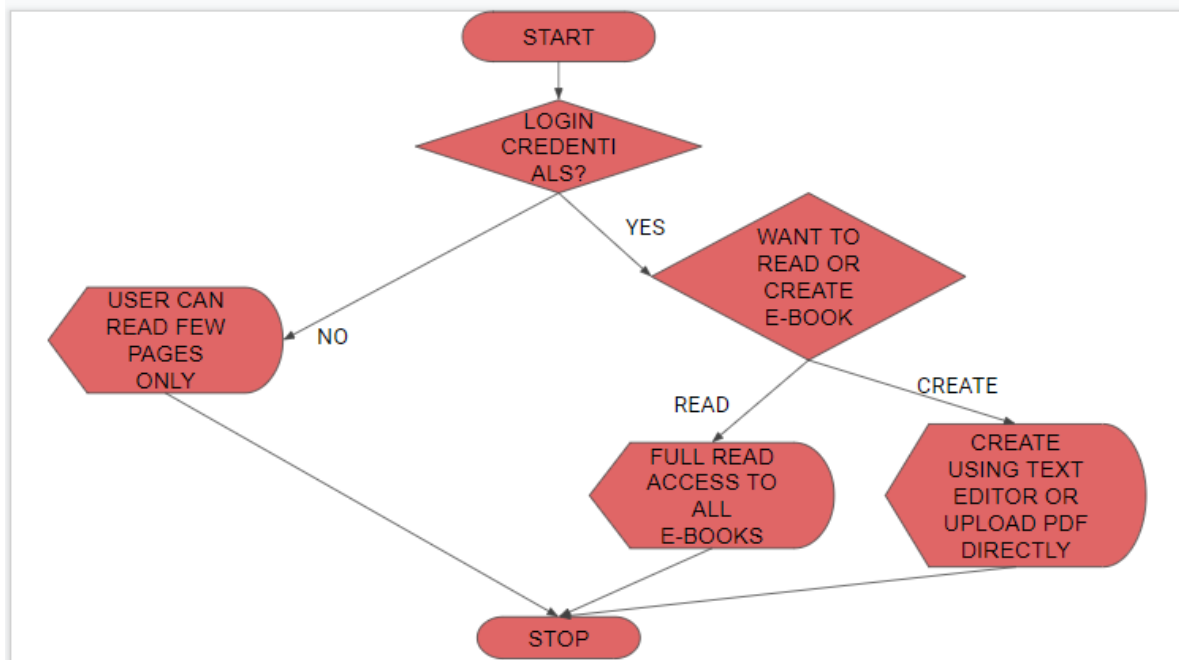


FIGURE 4.1 WORK FLOW DIAGRAM FOR USER

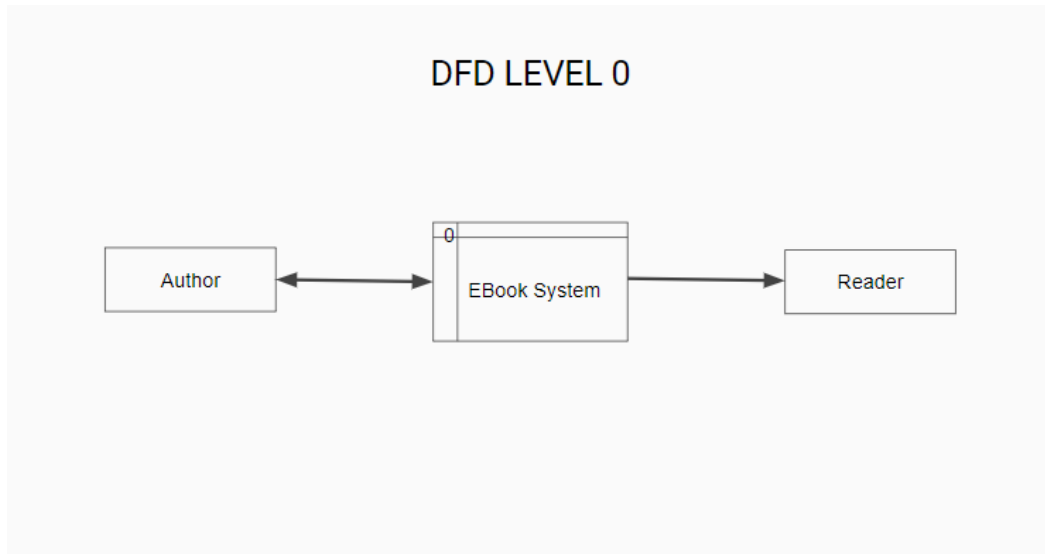
### 4.3 DATA FLOW DIAGRAM

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. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That’s why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

**4.3.1 LEVEL 0:** DFD Level 0 is also called a Context Diagram. It’s a basic overview of the whole system or process being analysed or modelled. It’s designed to be an at-a-glance view, showing the system as a

single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.

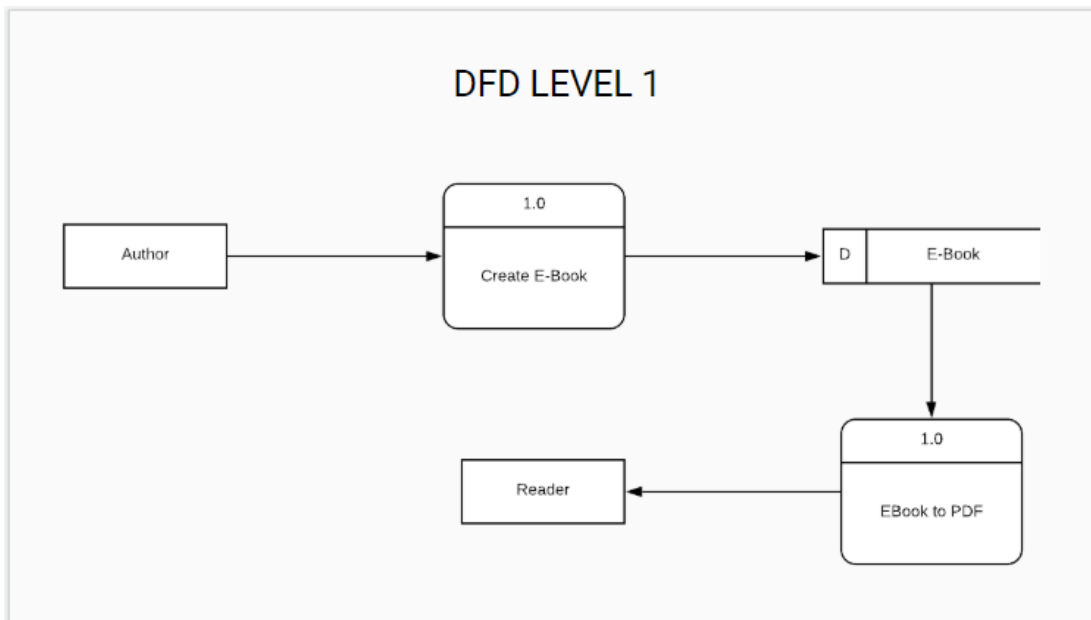
Data flow diagram- level 0 is shown in figure 4.4



**FIGURE 4.2 DATA FLOW DIAGRAM- LEVEL 0**

**4.4.2 LEVEL 1:** DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram. You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its sub processes.

Data flow diagram- level 1 is shown in figure 4.5



**FIGURE 4.3 DATA FLOW DIAGRAM- LEVEL 1**



## **CHAPTER 5 - CONCLUSIONS**

The aim of the thesis was to express the idea how to make a JavaScript-oriented application. It was a challenge, but contained a massive number of tasks needs to be finished. The system was built from scratch as it was the best opportunity to learn MEAN stack. The key features when building a Web based application were user-friendly interface and easy to use API.

The application took two months to finish before deployment. The application successfully covered three main sections of the company's website. The primary features of Simple application are user authentication, fast and responsive web interface, editable content, and it is easy to use the REST API. This project will be useful for both non-technical users and developers.

## CHAPTER 6 - REFERENCES

- [1] Movie Lens Data Sets .LocationGroup Lens Research Project at theUniversity of Minnesota.  
URL: <https://grouplens.org/datasets/>
- [2] Costin-Gabriel Chiru, Vladimir-Nicolae Dinu ,CtlinaPreda, MateiMacri ; “Movie Recommender System Using the User's Psychological Profile” in IEEE International Conference on ICCP, 2015.
- [3] By IJIRAE - International Journal of Innovative Research in Advanced Engineering
- [4] Brent Smith(Amazon.com) ,Greg Linden(Microsoft) "Two Decades of Recommender Systems at Amazon.com" Published by the IEEE Computer Society 2017 IEEE INTERNET COMPUTING.
- [5] Tejal Arekar, Mrs. R.S. Sonar, Dr. N. J. Uke, IT Dept./Pune University “ A Survey on Recommendation System ” International Journal of Innovative Research in Advanced Engineering (IJIRAE) ISSN: 2349-2163Volume 2 Issue 1 (January 2015)

# E-BOOK CREATE & STORE

---

## ORIGINALITY REPORT

---

6%

SIMILARITY INDEX

6%

INTERNET SOURCES

0%

PUBLICATIONS

5%

STUDENT PAPERS

---

## PRIMARY SOURCES

---

1

[www.dietms.org](http://www.dietms.org)

Internet Source

3%

2

Submitted to University of Greenwich

Student Paper

2%

3

Submitted to University of Bahrain

Student Paper

2%

---

Exclude quotes  On

Exclude bibliography  On

Exclude matches  < 10 words

**JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT**  
**PLAGIARISM VERIFICATION REPORT**

Date: 04-08-2020

Type of Document (Tick):  **PhD Thesis**  **M.Tech Dissertation/ Report**  **B.Tech Project Report**  **Paper**

Name: MINAL AGGARWAL Department: CSE Enrolment No 161218

Contact No. 7853900001 E-mail. minalaggawal96@gmail.com

Name of the Supervisor: Dr. AMIT KUMAR

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): E-BOOK CREATE AND STORE

**UNDERTAKING**

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/ revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

**Complete Thesis/Report Pages Detail:**

- Total No. of Pages = 52
- Total No. of Preliminary pages = 8
- Total No. of pages accommodate bibliography/references = 1

Minal  
(Signature of Student)

**FOR DEPARTMENT USE**

We have checked the thesis/report as per norms and found **Similarity Index** at 06.....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)

Signature of HOD

**FOR LRC USE**

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	<ul style="list-style-type: none"> <li>• All Preliminary Pages</li> <li>• Bibliography/Images/Quotes</li> <li>• 14 Words String</li> </ul>		Word Counts	
<b>Report Generated on</b>			Character Counts	
		<b>Submission ID</b>	Total Pages Scanned	
			File Size	

Checked by  
Name & Signature

Librarian

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at [plagcheck.juit@gmail.com](mailto:plagcheck.juit@gmail.com)