

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgaum-590018



A PROJECT REPORT (15CSP85) ON

“DIGITAL RIGHTS MANAGEMENT”

Submitted in Partial fulfillment of the Requirements for the Degree of
Bachelor of Engineering in Computer Science & Engineering

By

AKASH A S (1CR16CS001)

ABHILASH S (1CR16CS004)

HARI HARAN S (1CR16CS054)

K GOWTHAM (1CR16CS061)

Under the Guidance of,

Dr. Sreekanth Malladi

Professor, Dept. of CSE



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI, BANGALORE-560037

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI, BANGALORE-560037

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

Certified that the project work entitled “**Digital Rights Management**” carried out by **Mr Akash A S, USN 1CR16CS001, Mr Abhilash S, USN 1CR16CS004, Mr Hari Haran S, USN 1CR16CS054, Mr K Gowtham, USN 1CR16CS061**, bonafide students of CMR Institute of Technology, in partial fulfillment for the award of **Bachelor of Engineering** in Computer Science and Engineering of the Visveswaraiiah Technological University, Belgaum during the year 2019-2020. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library.

The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Dr. Sreekanth Malladi
Professor
Dept. of CSE, CMRIT

Dr. Prem Kumar Ramesh
Professor & Head
Dept. of CSE, CMRIT

Dr. Sanjay Jain
Principal
CMRIT

External Viva

Name of the examiners

- 1.
- 2.

Signature with date

DECLARATION

We, the students of Computer Science and Engineering, CMR Institute of Technology, Bangalore declare that the work entitled "**Digital Rights Management**" has been successfully completed under the guidance of Prof. Dr. Sreekanth Malladi, Computer Science and Engineering Department, CMR Institute of technology, Bangalore. This dissertation work is submitted in partial fulfillment of the requirements for the award of Degree of Bachelor of Engineering in Computer Science and Engineering during the academic year 2019 - 2020. Further the matter embodied in the project report has not been submitted previously by anybody for the award of any degree or diploma to any university.

Place: CMRIT

Date: 05/06/2020

Team members:

AKASH A S (1CR16CS001)

ABHILASH S (1CR16CS004)

HARI HARAN S (1CR16CS054)

K GOWTHAM (1CR16CS061)

ABSTRACT

Digital rights management (DRM) is a systematic approach to copyright protection for digital media. The purpose of DRM is to prevent unauthorized redistribution of digital media and restrict the ways consumers can copy content they've purchased. DRM is implemented by embedding code that prevents copying, specifies a time period in which the content can be accessed.

The goal of the project is to come up with a software solution that would prevent the piracy of digital media.

Most of the existing system are hardware based which makes them expensive and difficult to maintain. The solution proposed is software based and tries to overcome these drawbacks

The proposed solution combines traditional license key mechanisms and offline document viewing to produce a solution that prevents documents from illegally being distributed without authors consent. It must be noted that this is a software based DRM solution and as such offers good flexibility and adoptability, however this DRM solution is not fool proof and the main purpose of this DRM is to make it extremely inconvenient to pirate documents.

We make use of standard encryption protocols as well as an offline viewing mode of documents that also has a timing mechanism to ensure that documents are not used beyond their validity period. Our solution in its current form works on Windows operating system, however making it portable to other OS as well as devices should be simple as we have coded the front end in Electron.JS.

ACKNOWLEDGEMENT

We take this opportunity to express my sincere gratitude and respect to **CMR Institute of Technology, Bengaluru** for providing me a platform to pursue my studies and carry out my final year project

We have a great pleasure in expressing my deep sense of gratitude to **Dr. Sanjay Jain**, Principal, CMRIT, Bangalore, for his constant encouragement.

We would like to thank **Dr. Prem Kumar Ramesh**, Professor and Head, Department of Computer Science and Engineering, CMRIT, Bangalore, who has been a constant support and encouragement throughout the course of this project.

We consider it a privilege and honor to express my sincere gratitude to my guide **Dr. Sreekanth Malladi, Professor**, Department of Computer Science and Engineering, for the valuable guidance throughout the tenure of this review.

We also extend my thanks to all the faculty of Computer Science and Engineering who directly or indirectly encouraged us.

Finally, We would like to thank our parents and friends for all their moral support they have given me during the completion of this work.

TABLE OF CONTENTS

	Page No.
Certificate	ii
Declaration	iii
Abstract	iv
Acknowledgement	v
Table of contents	vi
List of Figures	viii
List of Tables	ix
1 INTRODUCTION	1
1.1 Relevance of the Project	1
1.2 Problem Statement	2
1.3 Objective	3
1.4 Scope of the Project	3
1.5 Methodology	3
1.6 Chapter Summary	4
2 LITERATURE SURVEY	5
2.1 Digital Rights Management: Model, technology and applications	5
2.2 Digital Rights Management for Content Distribution	6
2.3 Comparisons	6
3 SYSTEM REQUIREMENTS SPECIFICATION	7
3.1 Functional Requirements	7
3.2 Non-Functional Requirements	7
3.3 Software Requirements	8
3.4 Hardware Requirements	8
4 SYSTEM ANALYSIS AND DESIGN	9
4.1 System Architecture	9
4.2 Flowchart: Upload content and generate access keys	10
4.3 Data Flow of Sequence: Redeem the content	11

5	IMPLEMENTATION	13
	5.1 Algorithm for Uploading and redeeming documents	13
	5.2 Algorithm for Offline viewing of Documents	14
	5.3 Back End implementation	16
	5.4 Front-End implementation	19
6	RESULTS AND DISCUSSION	24
7	TESTING	27
8	CONCLUSION AND FUTURE SCOPE	30
	8.1 Conclusion	30
	8.2 Future Scope	30
	REFERENCES	31

LIST OF FIGURES

	Page No.
Fig 4.1 System architecture	9
Fig 4.2 Upload content and generate access keys	10
Fig 4.3 Redeem the content	11
Fig 5.1 Process of uploading and redeeming documents	13
Fig 5.2 Process of Offline Viewing of documents	14
Fig 5.3 Code Snippet of Authentication	17
Fig 5.4 Code Snippet of Authorization	17
Fig 5.5 Code snippet of Redeeming License Key	18
Fig 5.6 Code Snippet of Stored Documents	19
Fig 5.7 Code Snippet of Register UI	20
Fig 5.8: Code Snippet of Login UI	21
Fig 5.9: Code Snippet of Navigation bar UI	22
Fig 5.10: Code Snippet of Rendering Window	23
Fig 6.1 Transmittance of Auth Token	24
Fig 6.2 Front End Login Page	25
Fig 6.3 Upload Document and generate keys	25
Fig 6.4 Viewing of the File	26
Fig 6.5 Library	26
Fig 7.1 Testing process of Login through Postman	28
Fig 7.2 Testing of Generating License Keys	28
Fig 7.3 Testing of Getting Owned Documents.	29

LIST OF TABLES

	Page No.
Table 1.1 Summary of the Different Approaches	2
Table 1.2 Summary of the Methodology	3
Table 5.1 Tabular Description of Private Modules	16
Table 5.2 Tabular Descriptions of Public modules	16

Chapter 1

INTRODUCTION

The rise of digital media and analog-to-digital conversion technologies has vastly increased the concerns of copyright-owning individuals and organizations, particularly within the music and movie industries.

Also the rise of personal computers as household appliances has made it convenient for consumers to copy media (which may or may not be copyrighted). This, combined with the internet and popular file-sharing tools, has made unauthorized distribution of copies of copyrighted digital media (also called as digital piracy) much easier.

To eliminate this problem, we develop a software solution that can be installed on the clients' machine which will prevent such unauthorized copying of content.

1.1 Relevance of the Project

In an online learning and information environment, DRM can facilitate trade and exchange of learning objects between institutions such as universities and colleges on a free or fee basis. Universities and other tertiary institutions have held hundreds of thousands of learning objects, many of which are useful to other teaching institutions as well. Promoting the exchange and re-use of quality learning objects, while respecting and rewarding the intellectual property of the various contributors, are the two key issues which have to be solved before online learning can become cost effective. A flexible and effective DRM solution can manage the creation, retrieval, trading and distribution of online learning objects and support collaborative development.

1.1.1 Approaches Used

The below table lists some of the common approaches in literature used for implementing DRM Systems.

Table 1.1 Summary of the Different Approaches

Name of the approach	Research Paper	Author(s)
UPX	General Public License	1996-2018 Markus Oberhumer, Laszlo Molnar & John Reiser
Key Generation using FBPS	A Secure License Key Generation using FBPS	Fatangare Sonal Taksal Ashwini Todmal Satish.R.
MAC Address	License Generator Using MAC Address for Industrial Application	Prof. Ruchi Rautela, Mr. Rahul Parandwal, Mr. Pankaj Upadhyay

1.2 Problem Statement

The goal of the project is to come up with a software solution that would prevent the piracy of digital media.

Most of the existing system are hardware based which makes the expensive and difficult to maintain. The solution proposed is software based and tries to overcome these drawbacks.

The objective is to build a software solution that provides the user of a software with a DRM based anti-tampering protection to the content of the application, and also to generate license keys using the software.

Given a software Suite, we have to provide mechanism to generate MAC based license keys with validity period. We also have to provide an inline encryption

technology to encrypt server side data based on the client side key(MAC/Hardware). We are required to provide executable encryption.

1.3 Objective

The main objectives of the proposed system are:

To prevent users from copying digital media such as movies, songs, documents and rebrand it as their own.

DMR technologies enables content publishers to enforce their own access policies on content, such as restrictions on copying or viewing.

The objective is to build a software solution that provides the user of a software with a DRM based anti-tampering protection to the content of the application, and also to generate license keys using the software.

1.4 Scope of the Project

- Development of local strategies and action plan to further improve DRM.
- Limiting the time period for accessing the content, after which content disappears.
- Limiting the number of devices on which the content may be installed.

1.5 Methodology

Table 1.2 Summary of the Methodology

Story ID	Requirement description	User stories/Task
1	To provide an interface for the user to navigate through the application	Setting up the basic UI for the application
2	Provide a mechanism to integrate the software to other	Provide a mechanism to host the softwares

	applications	
3	Implementing a DRM model, which will be integrated to applications to protect its content	Implement a drm model
4	Integrate the unit components and apply it to the process of real world testing	Process of real world testing

1.6 Chapter Summary

The main purpose of this chapter was to introduce the concept of DRM systems, the major algorithms used in this field and the scope of research done. We believe that building a robust DRM systems involves having a good understanding of the current systems in the field and comes with its fair share of challenges to maximise user satisfaction.

Chapter 2

LITERATURE SURVEY

In this chapter we discuss various research papers and their findings that paves way to demonstrate our idea and helps to set goals to implement them.

2.1 IEEE paper on Digital Rights Management: Model, technology and applications

This paper on Digital Rights Management on model, technology and applications states that with rapid achievement of current information technology and computing ability and applications, much more digital content such as films, cartoons, design drawings, office documents and software source codes are produced in daily work, however to protect the content being copying, shared or deliberately stolen by inside or outside, digital rights management (DRM) became more and more important for digital content protection. In this paper, we studied various DRM model, technology and application, and first proposed DRM Security Infrastructure (DSI), in which the authors defined encryption, hash, signature algorithm, watermarking algorithms, authentication, usage control, trusted counter, conditional trace, secure payment, and based on the DSI, the authors have then proposed a whole classification approach and architecture of all kinds of DRMs, in which they proposed 6 typical classes of copyrights and content protection DRMs architecture: Software-oriented DRM, eBook- oriented DRM, Video-oriented DRM, Image-Oriented DRM, Unstructured data oriented DRM, Text-oriented DRM. Based on the above DSI, the authors then proposed a dynamic DRM model selection method for various DRM application, which can be adapted dynamically for different technology of different applications, which can provide a whole solution for variant DRM development in a rapid and customized mode.

2.2 Digital Rights Management for Content Distribution

This paper on Digital Rights Management for content distribution sheds light on transferring the traditional business model for selling digital goods linked to physical media to the online world leads to the need for a system to protect digital intellectual property. Digital Rights Management. (DRM) is a system to protect high- value digital assets and control the distribution and usage of those digital assets. The authors present a review of the current state of DRM, focusing on security technologies, underlying legal implications and main obstacles to DRM deployment with the aim of providing a better understanding of what is currently happening to content management on a legal and technological basis and well prepared for grasping future prospects. The proposed DRM method, technology and application in this paper provided a common, flexible and extendable solution for variant DRM scenes, and can support rapid and customized development.

2.3 Comparisons

Upon studying the aforementioned Research studies, we can compare the compare and differentiate the contents of paper of same domain.

The first paper provided us with various DRM model, technology and application, and first proposed DRM Security Infrastructure (DSI), in which we defined encryption, hash, signature algorithm, watermarking algorithms, authentication, usage control, trusted counter, conditional trace, secure payment, and based on the DSI we then proposed a whole classification approach and architecture of all kinds of DRMs. Moreover, we proposed an opinion that the future life will enter into a new era that the content usage and consumption will not again adopt DRM technology rather than with law, liberty and morality.

The second paper provided us with a review of the current state of DRM, focusing on security technologies, underlying legal implications and main obstacles to DRM deployment with the aim of providing a better understanding of what is currently happening to content management on a legal and technological basis and well prepared for grasping future prospects

Chapter 3

SYSTEM REQUIREMENTS SPECIFICATION

3.1 Functional Requirements

- The software automatically generates license key for customers who wants to purchase on the platform.
- The software must provide APIs for software organization to communicate with the platform server.
- The software must be able to deliver dynamic interface to users based on the software organization requirement.
- Only Software organization should have access to their own data The platform system should be integrated with banking API.
- The client platform software must be able to handle the non-trivial decryption required to execute the software.

3.2 Non Functional Requirements

- Performance – Response Time must be within a tolerable limit and decryption must happen such that it doesnt cause too much strain on the CPU.
- Scalability - The server must be able to handle multi-user traffic and the software must be developed in such a manner that it must be scalable for the future.
- Capacity - The server side hardware must have large enough storage to handle storing of software on the platform.
- Availability - The server must use mechanism to ensure ROP of a few minutes
- Recoverability - There must be some redundancy mechanism to ensure lost/corrupted data can be recovered.
- Maintainability - The server side logic must be functional and modular to ensure maintainability.
- Security - Sufficient encryption technology at backend must be applied to prevent unauthorized access to software stored on the platform.

- Data Integrity - A checksum based verification at both servers/client must be done to ensure that data sent/received is correct.
- Usability - Both the software platform as well as API must be well documented to ensure quality of service.

3.3 Software Requirements

OS: Microsoft Windows 7/8/10 (32-bit or 64-bit)

.NET FRAMEWORK

3.4 Hardware Requirements

- Processor: 1 gigahertz (GHz) or faster processor or SoC.
- RAM: 1 gigabyte (GB) for 32-bit or 2 GB for 64-bit.
- Hard disk space: 400 MB for 32-bit OS and 64-bit OS.
- Graphics card: DirectX 9 or later with WDDM 1.0 driver. Display: 1920x1080

Chapter 4

SYSTEM ANALYSIS AND DESIGN

4.1 System Architecture

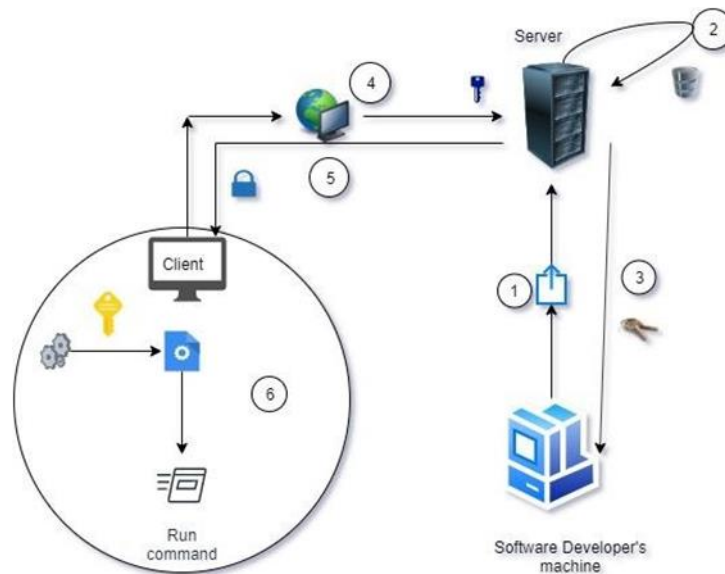


Fig 4.1 System architecture

Workstation connects to the server requesting for the software to be uploaded. It also requests for keys to be generated for the product. Server receives the software from the workstation and stores it in the machine.

Server also generates license keys with validity and expiry dates. The server sends the keys back to the workstation where the owner of the software can distribute keys to the clients. Client connects to the server and requests for a software along with a key, Server validates the key license.

After validation, it encrypts the software using executable compression where in the key is unique to the client. A process at the client can then decrypt and execute the software.

4.2 Flowchart: Upload content and generate access keys

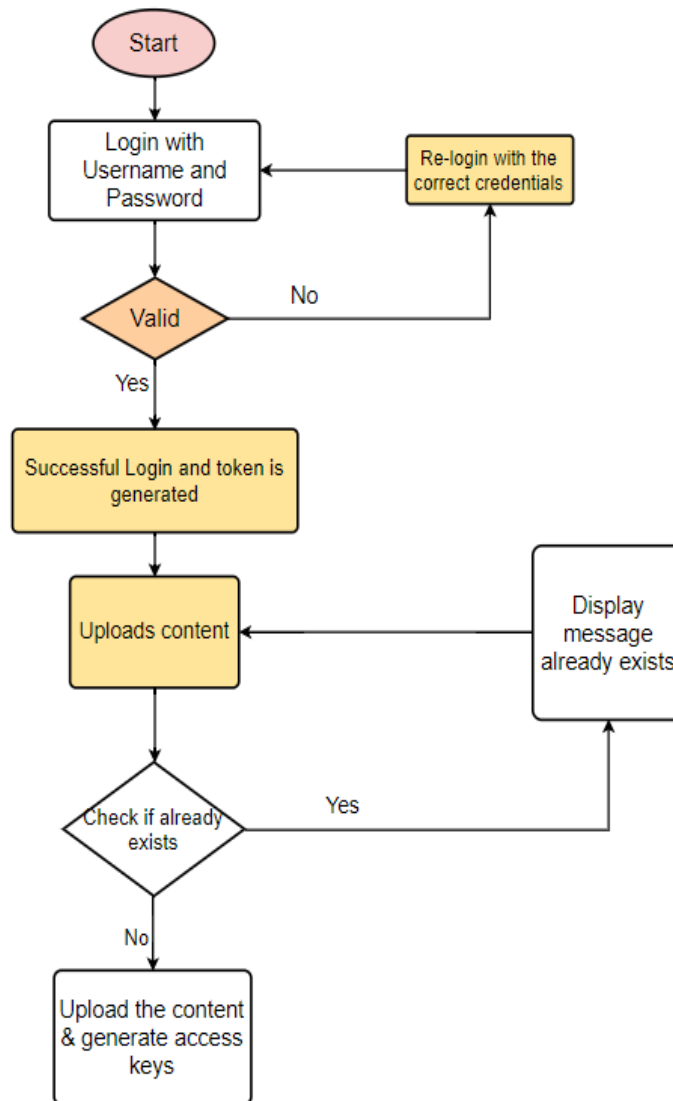


Fig 4.2: Upload content and generate access keys

Step 1: The content publisher will be able to login through the application with the provided credentials, and if the provided credentials are verified as valid, He/she will subsequently be able to access the application from their perspective.

Step 2: The content publisher will be provided with the privilege of publishing a content in the application.

Step 3: The content publisher will then publish an authentic digital content, which will then be compared across already existing content to ensure the eradication of data redundancy.

Step 4: Once the content is uploaded on to the application, the publisher will be able to generate various number of keys, which can be distributed amongst different users to redeem the content

4.3 Data Flow of Sequence: Redeem the content

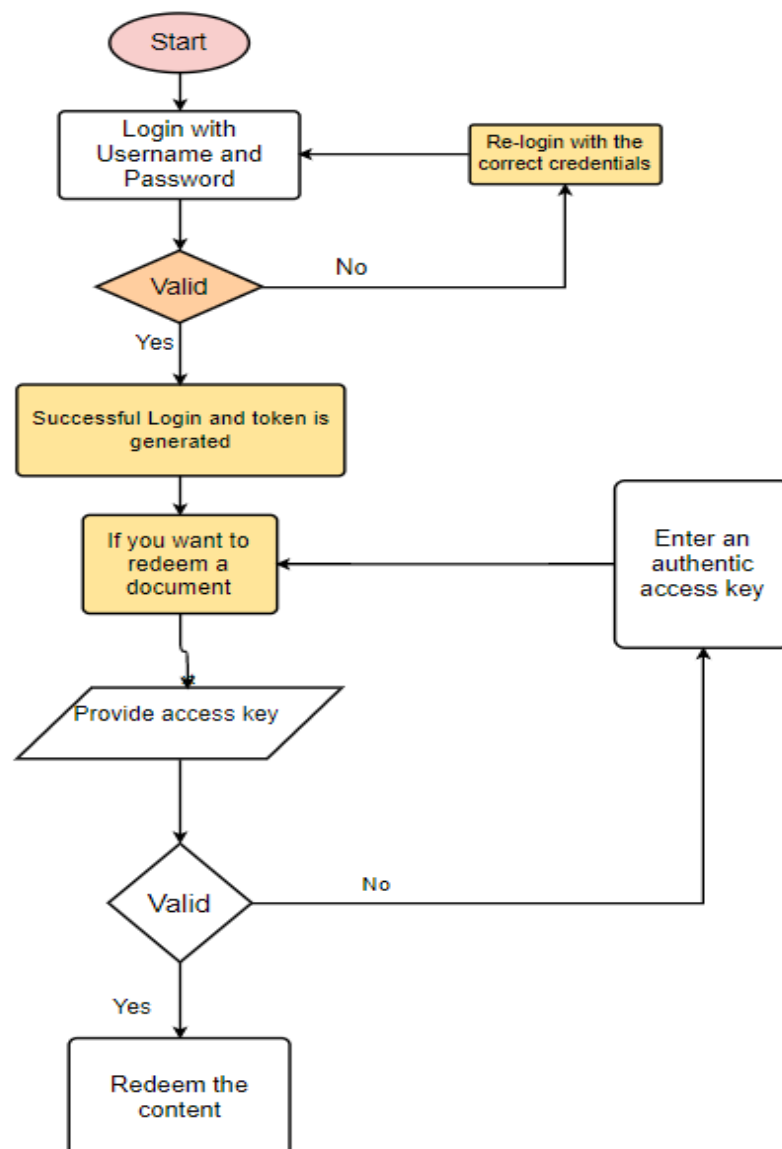


Fig 4.3: Redeem the content

Digital Rights Management

Step 1: The user will be able to login through the application with the provided credentials, and if the provided credentials are verified as valid, He/she will subsequently be able to access the application from their perspective.

Step 2: The user will have a restricted privilege, which enables He/she to redeem a particular content.

Step 3: The user will provide an access key in order to redeem the document.

Step 4: The user will be able to access the document and view it, provided, the access key is verified as a valid key.

Chapter 5

IMPLEMENTATION

5.1 Algorithm for Uploading and redeeming documents

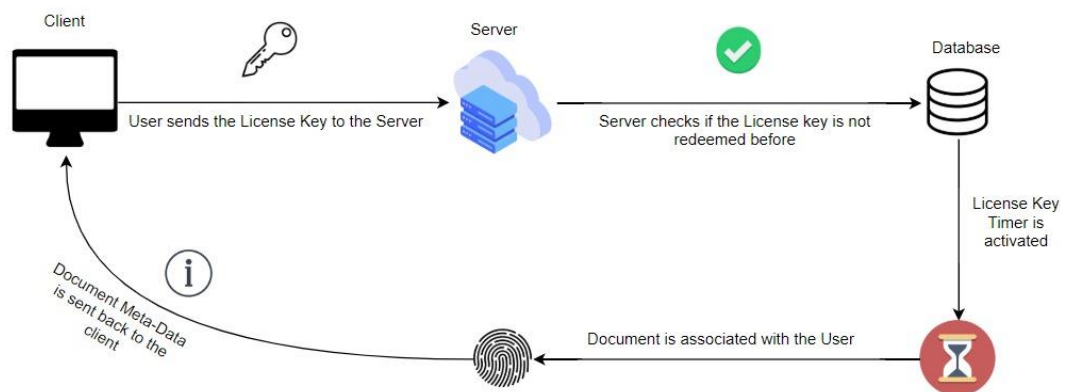


Fig 5.1: Process of uploading and redeeming documents

When a user uploads a document, the user has the choice of setting the validity period as well as the number of keys that should be generated for the document. When a user redeems a document, the user must provide a valid license key.

The above is described in the below steps

1. A user provides a valid license key (The user may receive it from any source)
2. The front end application sends this as a JSON request to the Node.JS server.
3. The server then activates a Timer (if there is a validity associated with the document). This will cause the license key to be deleted once the Timer expires.
4. The server sends the document meta data which includes expiry, document author etc. but most importantly, it also sends the url to where the document is stored. The license key is now linked to the particular user who redeemed it.
- 5) The client application will now prepare the document to be viewed natively. It fetches the document from the server based on the url the server sends. The server

validates if the user has access to the document and then sends the document to the client.

6) The control of the document is now handed over to the client side application which must ensure that the document can be viewed offline.

5.2 Algorithm for Offline viewing of Documents

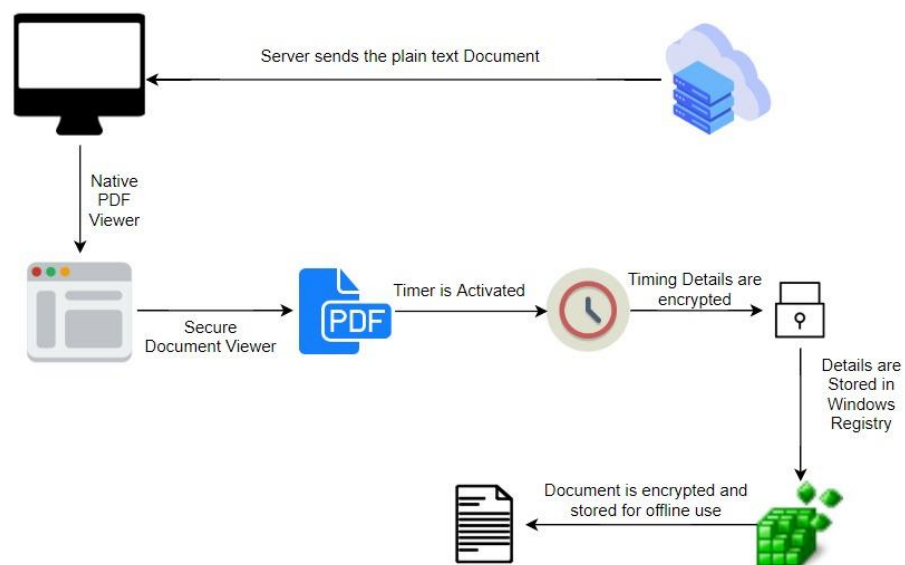


Fig 5.2: Process of Offline Viewing of documents

1. Begin
2. At a particular point in time when the document is being viewed offline by the user, the document will be retrieved by the application from the server.
3. For every minute the document is being viewed in the user's application, the seconds will be updated.
4. With the usage of validation token key, the value of seconds will be encrypted.
5. The encrypted value of seconds will be transmitted securely to the registry.
6. Once the encrypted value of seconds is received at the registry, the value will be decrypted.

7. This decrypted value of seconds will be compared with the allocated time period during the document redeeming process.
8. If the value of seconds is greater than or equal to the allocated time period of the redeemed document, then the document will be no longer made available to the user's application.
9. If the value of seconds is lesser than the allocated time period of the redeemed document, then the document will be made available to the user until the viewing period of the document expires for that particular user.
10. The complete process takes place when the user is offline and the document can be viewed for only the allocated duration even when the user's application is not connected to the internet.
11. End

5.3 Back End Implementation

Various methods such as /download, /newDocument, /generateLicenseKeys, /redeemLicenseKeys, /getAllDocuments, /getOwnedDocuments, /getAllKeys were implemented.

Table 5.1: Tabular Description of Private Modules

ROUTE	HTTP METHOD	DESCRIPTION
/download	GET	In this route, access to the document is made available
/downloadImage	GET	The cover image associated with the respective document is made available
/newDocument	POST	In this route, a new document can be uploaded as well as license keys for the document can be generated
/generateLicenseKeys	GET	Here license key can be generated for already uploaded documents
/redeemLicenseKey	GET	Users can redeem license keys from this route
/getAllDocuments	GET	All documents owned by a user can be viewed here
/getOwnedDocuments	GET	Here all documents owned (The uploader) can be viewed
/getAllKeys	GET	All keys that are generated for a particular owner is shown here

Table 5.2: Tabular Descriptions of Public modules

ROUTE	HTTP METHOD	DESCRIPTION
*/	ALL	Here the server checks if the application it is communicating to is authentic or not
/	POST	This route is just used to check if the server is alive or not
/register	POST	This route is used to register a new user
/login	POST	This route is used to login to an already existing account

5.3.1 Authentication

```
1  const jwt = require('jsonwebtoken')
2  const app_config = require('../model/app_config')
3  module.exports = (req,res,next)=>{
4
5      const token = req.header('auth-token')
6      if(!token)
7          return res.status(401).send({'message':'error','details':'Access Denied'})
8  try{
9      const verify = jwt.verify(token,app_config.TOKEN_SECRET)
10     req.user = verify._id
11     next()
12 }catch(err){
13     return res.status(400).send({'message':'error','details':'Invalid Token'})
14 }
15 }
```

Fig 5.3: Code Snippet of Authentication

- In the authentication module, we want to allow entry to only authenticated users. To do this we issue a token to every user who logs in successfully.
- For further access to our backend server, the client must provide this token. In this module we verify if the token is valid and decrypt it.
- Once we verify this, the backend server knows which user is trying to communicate with it.

5.3.2 Authorization

```
1  const mongoose = require('mongoose')
2
3  module.exports = (req,res,next)=>{
4
5      mongoose.connection.db.collection('authorization', function (err, collection) {
6          collection.find({role:'normal'}).toArray((err,docs)=>{
7              if(docs[0].url.indexOf(req.url.split("?")[0])=-1)
8                  res.status(400).send({'message':'error','details':'Unauthorized Access'})
9              else
10                 next()
11          })})
12 }
13 }
```

Fig 5.4: Code Snippet of Authorization

- In the authorization module, our backend server has two types users.
- A user who uploads documents and another user who redeems and views documents.
- In order to make this work we have specified the routes that each user can take (this info is stored in mongo DB).

5.3.3 Redeeming the License Key

```

1  const license = require('../model/License')
2  const moment = require('moment')
3  module.exports = async (req,res,next)=>{
4    const key = req.body.key
5    const lick = await license.findOne({'key':key})
6    if(lick && lick.redeemer_id==undefined){
7      //ensure if license is not redeemed by checking if redeemer id is not empty
8      //check for validity before doing the below line
9      const expiry = lick.validity!=undefined?moment().add(lick.validity,'seconds').toDate():undefined
10     await license.findOneAndUpdate({'key':key},{'expireAt':expiry,'redeemer_id':req.user})
11     return res.send({'message':'success','details':lick.document_id})
12   }
13   else{
14     return res.send({'message':'error','details':'Invalid License Key or License key might have already been redeemed'})
15   }
16 }

```

Fig 5.5: Code snippet of Redeeming License Key

- In this module, the users send their access key to the server and we check if this key exists in Mongo DB. During this we check for two things,
- If the key exists in the database and if the key has already been redeemed or not. If any of the two if false we send an error message to user.
- After the checking is done the validity period for the key is calculated and updated in seconds in the database. And the document ID is sent to the user for download.

5.3.4 Stored Documents

```
1  const documents = require('../model/Document')
2  const license = require('../model/License')
3  module.exports = async(req,res,next)=>{
4      console.log(req.user)
5      const redeemer_id = await license.find({'redeemer_id':req.user}) //we dont need to verify
6      if(!redeemer_id)
7          return res.send({'message':'success','details':'No documents available'}) //no result
8      var list_of_documents = []
9      for(let items in redeemer_id){
10         const result = await documents.findOne({'_id':redeemer_id[items].document_id})
11         list_of_documents.push({
12             'author' : result.author,
13             'company': result.company,
14             'file_name' : result.file_name,
15             'cover_url' : result.cover_url,
16             'document_id': result._id
17         })
18     }
19
20     return res.send({'message':'success','details':list_of_documents})
```

Fig 5.6: Code Snippet of Stored Documents

- In this module, the application helps the user to view the documents that He/she has access to view.
- Here, we will be checking the license documents, to verify if the redeemer ID is matching with a particular user. If there isn't any match, we print out the message saying that there are no documents available, else, the user will have documents, so we create an empty list called list_of_documents.
- Now, we iterate through the redeemer ID, which will contain all the license keys that the user has redeemed, and push the details of a particular user into the empty list.
- Once we finish collecting it, we send the result.

5.4 Front End Implementations

Various Front End modules such as Login page, Register page, Upload document and generate license keys page, Navigation bar, Rendering Window, Library etc.

5.4.1 Register

```
<Container>
  <Row>
    <Card
      className="shadow p-3 mb-5 bg-white rounded"
      style={{ width: "26rem", height: "26rem" }}
    >
      <Card.Header>Register</Card.Header>
      <Card.Body>
        <Form>
          <Form.Group controlId="formBasicName">
            <Form.Label>Name</Form.Label>
            <Form.Control
              type="text"
              required="true"
              placeholder="Name"
            />
          </Form.Group>
          <Form.Group controlId="formBasicEmail">
            <Form.Label>Email address</Form.Label>
            <Form.Control
              type="email"
              required="true"
              placeholder="Enter email"
            />
            <Form.Text className="text-muted">
              We'll never share your email with anyone else.
            </Form.Text>
          </Form.Group>
          <Form.Group controlId="formBasicPassword">
            <Form.Label>Password</Form.Label>
            <Form.Control
              type="password"
              required="true"
              placeholder="Password"
            />
          </Form.Group>
          <Button variant="outline-dark" type="submit">
            Submit
          </Button>
        </Form>
      </Card.Body>
    </Card>
  </Row>
</Container>
```

Fig 5.7: Code Snippet of Register UI

- In this register module, sections of code are composed to store the information such as Name, Email address and password.
- The attribute required is assigned to true, to ensure that all the fields are filled mandatorily.
- The submit button ensures the data of the user is stored to the back end server and can be accessed by the user whenever necessary.

5.4.2 Login

```
<Container>
  <Row>
    <Card
      className="shadow p-3 mb-5 bg-white rounded"
      style={{ width: "26rem", height: "26rem" }}
    >
      <Card.Header>Login</Card.Header>
      <Card.Body>
        <Form>
          <Form.Group controlId="formBasicEmail">
            <Form.Label>Email address</Form.Label>
            <Form.Control
              type="email"
              required="true"
              placeholder="Enter email"
            />
            <Form.Text className="text-muted">
              We'll never share your email with anyone else.
            </Form.Text>
          </Form.Group>
          <Form.Group controlId="formBasicPassword">
            <Form.Label>Password</Form.Label>
            <Form.Control
              type="password"
              required="true"
              placeholder="Password"
            />
          </Form.Group>
          <br />
          <br />
          <br />
          <Button variant="outline-dark" type="submit">
            Submit
          </Button>
        </Form>
      </Card.Body>
    </Card>
  </Row>
</Container>
```

Fig 5.8: Code Snippet of Login UI

- In the Login module, sections of code are composed to ensure that the authenticated users are able to access the application.
- To access the application, the user must fill in the sections of Email address and password that was provided by the user during register process.
- The submit button will verify the credentials and will grant access to the application, provided, the credentials are valid

5.4.3 Navigation Bar

```
7   const Styles = styled.div`
8     .navbar {
9       background-color: #222;
10    }
11    .navbar-brand,
12    .navbar-nav,
13    .nav-link {
14      color: #bbb;
15      &:hover {
16        color: white;
17      }
18    }
19  `;
20
21  const NavBar = () => (
22    <Styles>
23      <Navbar fixed="top" bg="dark" variant="dark" expand="lg">
24        <Navbar.Brand href="#home">DRM</Navbar.Brand>
25        <Navbar.Toggle aria-controls="basic-navbar-nav" />
26        <Navbar.Collapse id="basic-navbar-nav">
27          <Nav className="mr-auto">
28            <Nav.Link href="/Home">Home</Nav.Link>
29            <Nav.Link href="/Login">Login</Nav.Link>
30            <Nav.Link href="/Register">Register</Nav.Link>
31          </Nav>
32        </Navbar.Collapse>
33      </Navbar>
34    </Styles>
35  );
36
37  export default NavBar;
```

Fig 5.9: Code Snippet of Navigation bar UI

- In the Navigation bar module, sections of code are written to direct the user to certain aspects of application such as the home page, Library page etc.
- The attribute, fixed enables the navigation bar to be visible at all times in the application page irrespective of the scrolling.

5.4.4 Rendering Window

```
drm > public > JS electron.js > ...
1  const electron = require("electron");
2  const app = electron.app;
3  const BrowserWindow = electron.BrowserWindow;
4
5  const path = require("path");
6  const url = require("url");
7  const isDev = require("electron-is-dev");
8
9  let mainWindow;
10
11 function createWindow() {
12   mainWindow = new BrowserWindow({ width: 900, height: 680 });
13   mainWindow.loadURL(
14     isDev
15     ? "http://localhost:3000"
16     : `file://${path.join(__dirname, "../build/index.html")}`
17   );
18   mainWindow.on("closed", () => (mainWindow = null));
19 }
20
21 app.on("ready", createWindow);
22
23 app.on("window-all-closed", () => {
24   if (process.platform !== "darwin") {
25     app.quit();
26   }
27 });
28
29 app.on("activate", () => {
30   if (mainWindow === null) {
31     createWindow();
32   }
33 });
34
```

Fig 5.10: Code Snippet of Rendering Window

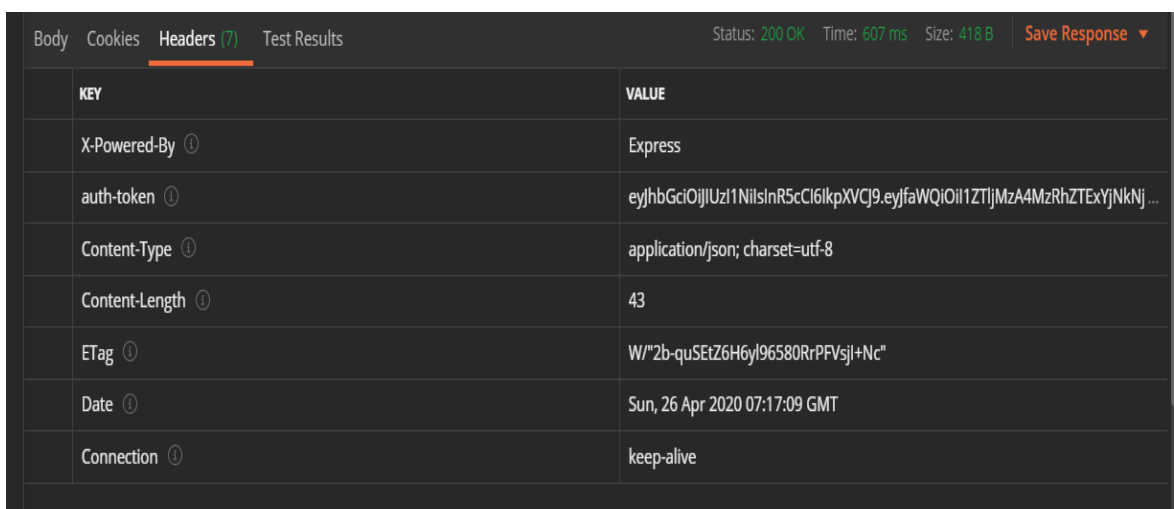
- The front end application built with react can be packaged into a desktop application through the Electron.
- In this module, we're creating a rendering window once the app is ready and rendering the content through it

Chapter 6

RESULTS AND DISCUSSION

In this chapter, we present the results of the executed methodology for the problem statement

6.1 Auth-Token

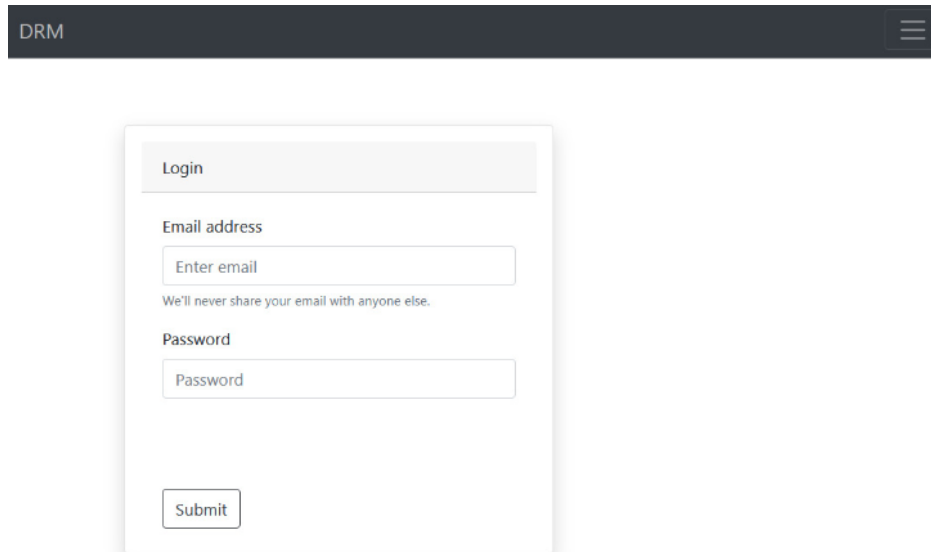


KEY	VALUE
X-Powered-By	Express
auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI1ZTljMzA4MzRhZTEyYjNkNj...
Content-Type	application/json; charset=utf-8
Content-Length	43
ETag	W/"2b-quSEtZ6H6yl96580RrPFVsjl+Nc"
Date	Sun, 26 Apr 2020 07:17:09 GMT
Connection	keep-alive

Fig 6.1: Transmittance of Auth Token

The picture above signifies the importance of auth-token which is used for the recognition of the user in the application.

6.2 Front End Login Page



The screenshot shows a dark grey header with the text 'DRM' on the left and a hamburger menu icon on the right. Below the header is a white login form with a light grey border. The form has a title 'Login' at the top. It contains two input fields: 'Email address' with a placeholder 'Enter email' and a note 'We'll never share your email with anyone else.' below it, and 'Password' with a placeholder 'Password'. At the bottom of the form is a 'Submit' button.

Fig 6.2: Front End Login Page

The picture above displays the login page in the front end of the application.

6.3 Upload Document and generate keys



The screenshot shows a blue header with navigation links 'Home', 'Login', and 'About', and a user profile icon. Below the header, the text 'Hello Author!' is displayed. There are two blue buttons: 'LIBRARY' and 'FILE'. Below the 'FILE' button is a text input field for 'Upload files'. Further down is an 'Expiry Date' input field. At the bottom is a 'Number of Keys' input field with a dropdown arrow. A green button labeled 'GENERATE KEYS' with a right-pointing arrow is at the bottom.

Fig 6.3: Upload Document and generate keys

The picture above displays the procedure in which the user will be able to upload the document through the front end of the application and generate multiple license keys for the same.

6.4 Viewing of the File

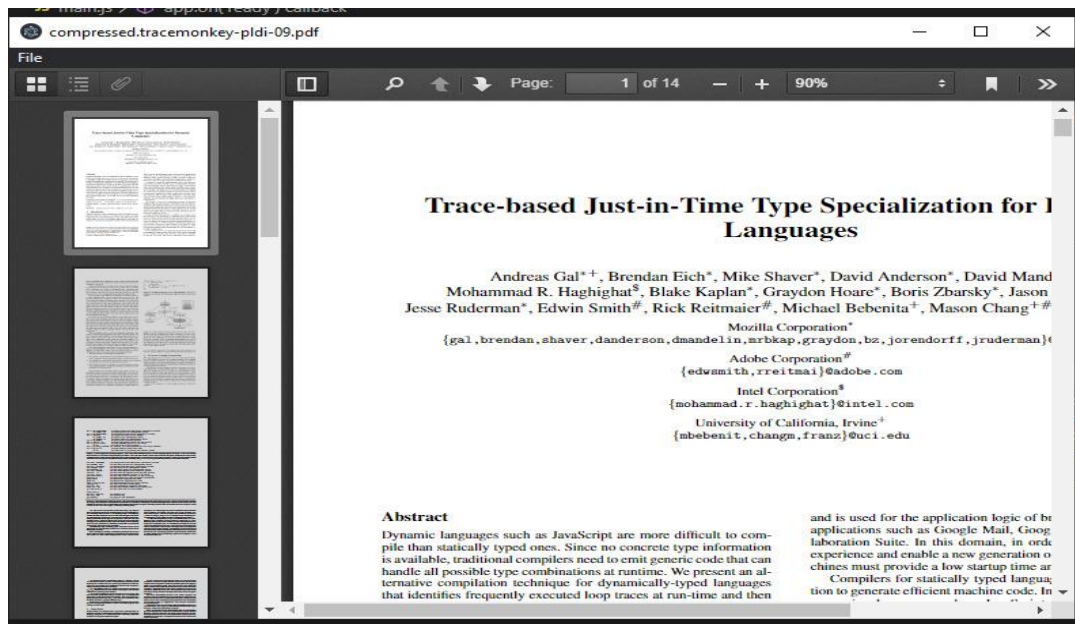


Fig 6.4: Viewing of the File

The picture above displays the viewing of a file which exists in the user’s library of redeemed documents.

6.5 Library

DRM Signed in as: Mark Otto

#	Name	Document ID	License Key
1	Mark	b73bf7d3ba1a517644661bc4bcd85f9a	b73bf7d3ba1a517644661bc4bcd85abc
2	Jacob	a33bf7d3ba1a517644661bc4bcd85f9a	a73bf7d3ba1a51754361bc4bcd85abc
3	Mark	b73bf7d3ba1a517644661bc4bcd85f9a	b73bf7d3ba1a517644661bc4bcd85abc
4	Jacob	a33bf7d3ba1a517644661bc4bcd85f9a	a73bf7d3ba1a51754361bc4bcd85abc
5	Mark	b73bf7d3ba1a517644661bc4bcd85f9a	b73bf7d3ba1a517644661bc4bcd85abc
6	Jacob	a33bf7d3ba1a517644661bc4bcd85f9a	a73bf7d3ba1a51754361bc4bcd85abc
7	Mark	b73bf7d3ba1a517644661bc4bcd85f9a	b73bf7d3ba1a517644661bc4bcd85abc
8	Richard	a33bf7d3ba1a517644661bc4bcd85f9a	a73bf7d3ba1a51754361bc4bcd85abc
9	Suez	b73bf7d3ba1a517644661bc4bcd85f9a	b73bf7d3ba1a517644661bc4bcd85abc
10	Jacob	a33bf7d3ba1a517644661bc4bcd85f9a	a73bf7d3ba1a51752341bc4bcd85abc
11	Mark	b73bf7d3ba17517644661bc4bcd85f9a	b73bf7d3xysa517644661bc4bcd85abc
12	Christine	a33bf7d3ba1a517644661bc4bcd85f9a	a73bf7d3ba1a51754361bc4bcd85abc

Transferring data from maxcdn.bootstrapcdn.com...

Fig 6.5: Library

The Library consists of the documents that the user has access to view

Chapter 7

TESTING

The modules developed in the Back End server was tested with the help of Postman API.

Postman is a collaboration platform for API development. Postman's features simplify each step of building an API and streamline collaboration so you can create better APIs—faster. Tests are automated by creating test suites that can run again and again. Postman can be used to automate many types of tests including unit tests, functional tests, integration tests, end-to-end tests, regression tests, mock tests, etc. Automated testing prevents human error and streamlines testing.

Postman is a simple yet effective tool when trying to dissect RESTful APIs made by others or test ones that we have made ourselves. In our project we have extensively used POSTMAN to test the backend server. Our server is a RESTful API that handles request and sends out response in the form of JSON. JSON is a format that is useful in transmission of large data over the internet. It is easy to transmit this as it is easy to convert it to binary format necessary for transmission. Postman offers a wide variety of tools that makes it helpful to test our backend server. It allows sending HTTP POST, GET, PUT, DELETE etc. and also allows us to modify the body of the request, such as JSON, RAW XML, FORM-DATA etc. POSTMAN also allows us to add custom header to the HTTP request (we have used this feature quite extensively). POSTMAN also allows for self-signing of certificate. We have tested all routes of our server with POSTMAN (both public and private) and found the server to be running as per requirement.

7.1 Testing of Login

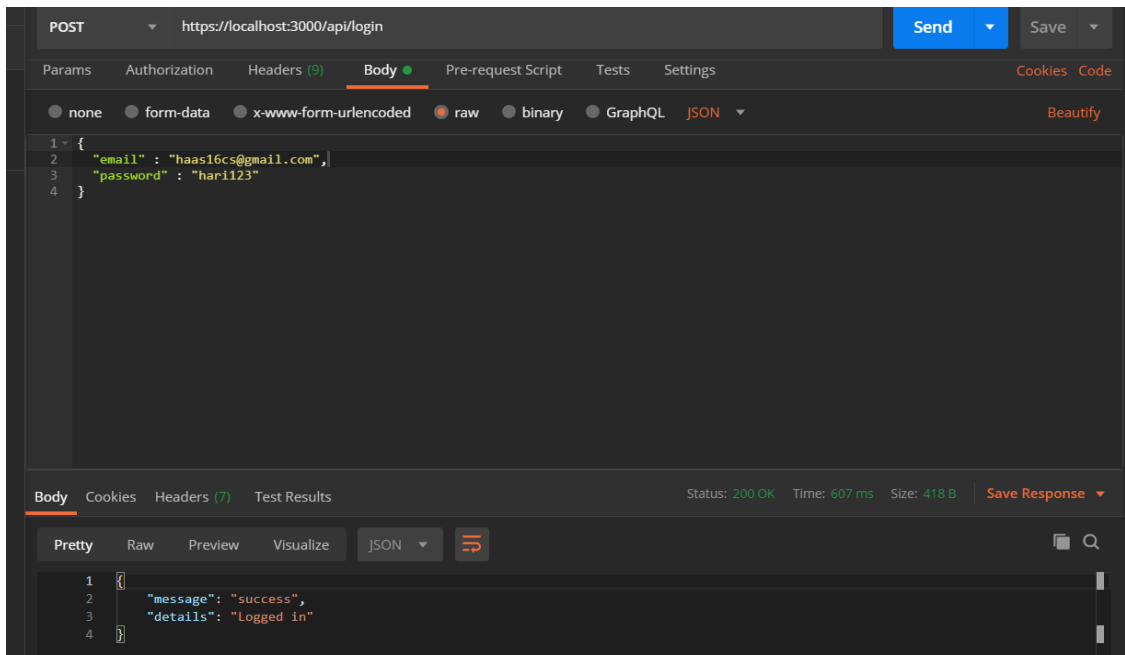


Fig 7.1: Testing process of Login through Postman

The picture above displays the successful testing of logging in to the application from the Back End Server through Postman API

7.2 Testing of Generating License Keys

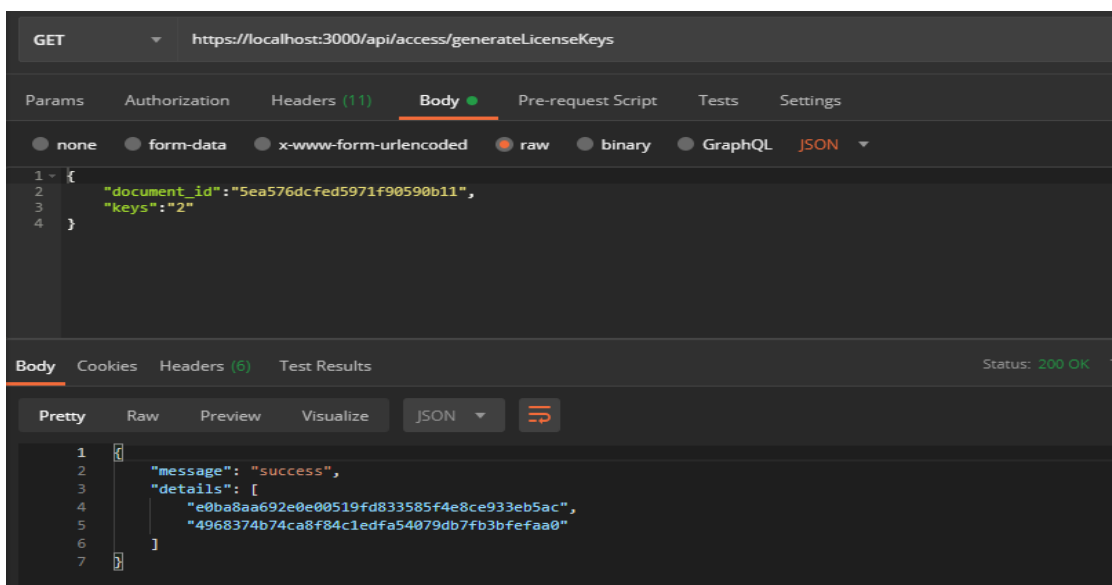


Fig 7.2: Testing of Generating License Keys

The above picture displays the successful testing of the process of generating license Keys from the Back End Server through Postman API.

7.3 Testing of Getting Owned Documents

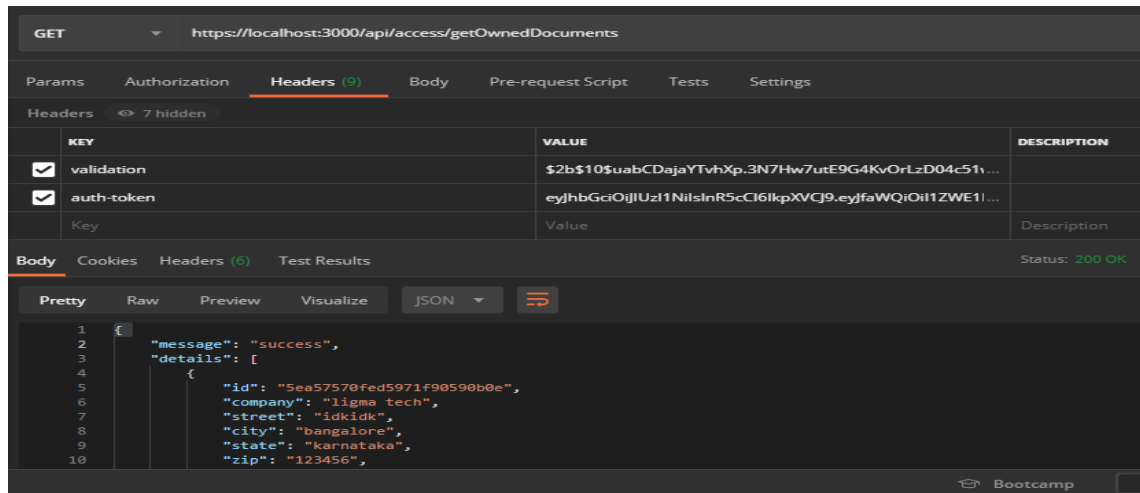


Fig 7.3: Testing of Getting Owned Documents.

The above picture displays the successful testing of the process of getting owned documents from the Back End Server through Postman API.

Chapter 8

CONCLUSION AND FUTURE SCOPE

8.1 Conclusions

The DRM based application that we developed acts a secure content provider as well as protector of these contents in the user's applications. However, to deploy a successful digital online service, it is not sufficient to apply cryptographic primitives and security measures to encrypt and distribute digital content online. Security technologies can certainly help raise the bar against circumvention of the system and make it more difficult to defeat the protection scheme

To summarize, we have implemented various modules required to achieve the provided problem statement such as /download, /newDocument, /generateLicenseKeys, /redeemLicenseKeys, /getAllDocuments, /getOwnedDocuments, /getAllKeys.

These implemented modules were tested separately (unit testing) using the Postman API, which yielded successful executions of the modules.

The front end of the application was asynchronously connected with the back end server that consisted of these various modules and the application was integrated

8.2 Future Scope

- In the upcoming versions, we plan to integrate DRM facilities to audio and video files such as movies, video clips, songs etc.
- We are also planning on adding a streaming feature.
- Cross-Platform Implementations.

REFERENCES

[1] IEEE paper on Digital Rights Management system design and implementation issues published on 27th Dec 2007

Authors: Hassan Elkamchouchi, Faculty of Engineering, Alexandria University, Egypt Yasmine Abouelseoud Faculty of Engineering, Alexandria University, Egypt

Link: <https://ieeexplore.ieee.org/document/444703>

[2] IEEE paper on Digital Rights Management: Model, technology and application published on 28th June 2017

Authors: Zhaofeng Ma School of Cyberspace Security, Beijing University of Posts and Telecommunications

Link: <https://ieeexplore.ieee.org/document/7961371>

[3] Google Scholar was used to look up Journals on Digital Rights Management namely, a journal on “Digital Rights Management for Content Distribution”.

Authors: Qiong Liu, Reihaneh Safavi-Naini and Nicholas Paul Sheppard.

Link:

https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=Digital+Rights%20+Management+for+Content+Distribution&btnG=