**Implementing System for Employee Verification System using Blockchain**

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

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER ENGINEERING**

**SUBMITTED BY**

**SUCHIT GUPTA  
RUSHIKESH MOKASHI  
SRIHARI NAIR**

**Under the Guidance of**

**DR. HARSHALI PATIL**

**ASSISTANT PROFESSOR, M.E CO-ORDINATOR**

**TCET**

****

**UNIVERSITY OF MUMBAI**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**DEPARTMENT OF COMPUTER ENGINEERING (2017-18)**



**Project Report Approval for B.E**

This is to certify that the dissertation work entitled **“Implementing System for Employee Verification System using Blockchain”** for B.E. in Computer Engineeringsubmitted to University of Mumbai by **“Srihari Nair”, “Rushikesh Mokashi” and “Suchit Gupta”**, bonafide students of Thakur college of Engineering and Technology, Kandivali, Mumbai is approved for the award of **Bachelor of Engineering** Degree in **Computer Engineering.**

|  |  |
| --- | --- |
| Signature :-----------------------------  Name : Guide Name  Designation | Signature :------------------------------  Name : Dr. Sheetal Rathi  HOD-CMPN |
|  |  |

Signature: ---------------------------------------------

Name : Dr. B. K. Mishra

Principal,

Thakur College of Engineering and Technology.

|  |
| --- |
| **Examiners** |
| 1. Signature :--------------------------   Name : |
| 1. Signature :--------------------------     Name : |

Date:

Place:

**Declaration**

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

-----------------------------------------

(Signature)

Suchit Gupta-1 (Roll No. 33)

-----------------------------------------

(Signature)

Rushikesh Mokashi-2 (Roll No. 60)

-----------------------------------------

(Signature)

Srihari Nair-3 (Roll No. 62)

Date:

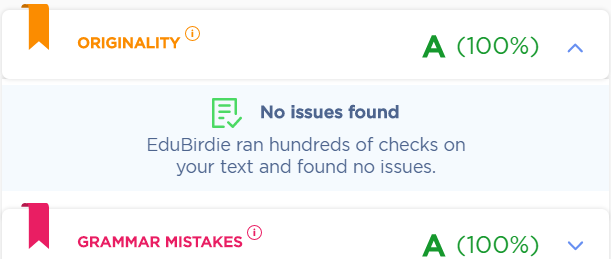
## Plagiarisms Report

## 

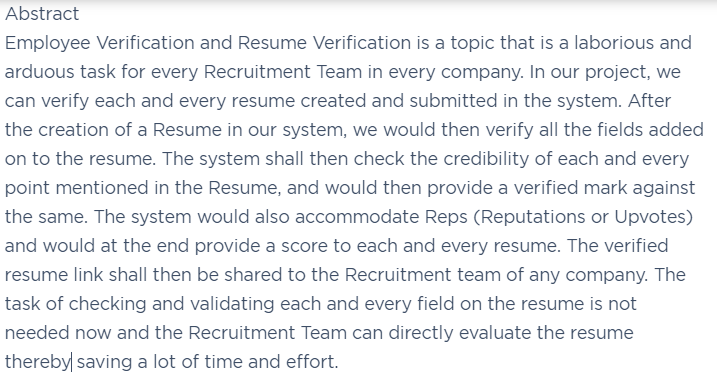
**Plagiarism Checker EduBirdie Originality Report**

**Similarity Found: 0%**

**Statistics:**



**Remarks: Low Plagiarism Detected - Your Document needs Optional Improvement.**



## Abstract

Employee Verification and Resume Verification is a topic that is a laborious and an arduous task for every Recruitment Team in every company. In our project, we can verify each and every resume created and submitted in the system. After creation of a Resume in our system we would then verify all the fields added on to the resume. The system shall then check the credibility of each and every point mentioned in the Resume, and would then provide a verified mark against the same. The system would also accommodate Reps (Reputations or Upvotes) and would at the end provide a score to each and every resume. The verified resume link shall then be shared to the Recruitment team of any company. The task of checking and validating each and every field on the resume is not needed now and the Recruitment Team can directly evaluate the resume there by saving a lot of time and effort.

Successful [resume verification](http://www.employeescreen.com/iqblog/resume-fraud/3-myths-about-resume-verification/) can identify fraudulent degrees, incorrect employment and

graduation dates, inflated salary histories and false job titles—issues that may help you determine a candidate is not right for you.

But what goes into a successful resume verification process? An important factor is the quality of the information provided by candidates in both the applicant release form and employment verification form (usually included on a job application or background screening questionnaire). If complete and correct, this information can be particularly helpful for completing background checks. However, if it’s incomplete or incorrect, it can hinder the verification process.

## List of Contents

[List of Figures](#_Toc354226634) iv

[List of Tables](#_Toc354226635) v

[Abbreviations and Symbols](#_Toc354226636) vi

[Chapter 1. Introduction](#_Toc354226637) ................... ...........Page no in decimal number.......1

[1.1 Problem Definition](#_Toc354226638) 1

[1.2 Aim](#_Toc354226639) 2

[1.3 Scope](#_Toc354226639) 2

[1.4 Motivation](#_Toc354226640) 3

[1.5 Organization of Dissertation Report](#_Toc354226640) 4

[Chapter 2. Review of Literature](#_Toc354226642) 5

[2.1 Literature Review](#_Toc354226643) 5

[2.2 Gap Identified](#_Toc354226644) 7

[2.3 Proposed Work](#_Toc354226644) 8

[Chapter 3. Related Theory](#_Toc354226648) 9

[3.1 SubTopic I](#_Toc354226649) 9

[3.2 Sub Topic II](#_Toc354226650) 10

[3.3 Sub Topic III](#_Toc354226650) 11

[**Chapter 4. Design Methodology**](#_Toc354226651) **6**

[4.1 Methodology for SubModule1](#_Toc354226652) 6

[4.2 Methodology for SubModule2](#_Toc354226653) 8

[4.3](#_Toc354226653) Methodology for Sub Module 3........................................................................9

[4.4 Technology & Configurations](#_Toc354226653) 10

[Chapter 5. Results and Discussions](#_Toc354226654) 24

[5.1 Implementation](#_Toc354226655) 24

[5.2 Results](#_Toc354226655) 32

[5.3 Testing](#_Toc354226655) 34

[5.4 Result Analysis](#_Toc354226655) 35

[Chapter 6. Conclusion & Future Scope](#_Toc354226656) 36

[Appendix](#_Toc354226660) 38

[References](#_Toc354226660) 39

[Publication](#_Toc354226660) 41

[Acknowledgement](#_Toc354226660) 42

List of Figures

Figure 3.5.2.1: Agile Development…………………………………………………Page No. 14

Figure 3.5.3.1: Gantt Chart (Part 1) ………………………………………………..Page No. 15

Figure 3.5.3.2: Gantt Chart (Part 1) ………………………………………………..Page No. 16

Figure 3.6.1.1: Data Flow Diagram – Level 0………………………………………Page No. 17

Figure 3.6.1.2: Data Flow Diagram – Level 1 ………………………………………Page No.17

Figure 3.6.2: Sequence Diagram……………………………………………………Page No. 18

Figure 3.6.3: Class Diagram ………………………………………………………..Page No.19

Figure 4.1.1: Architecture Diagram ………………………………………………..Page No.20

Figure 4.2.1: Flowchart Part 1………………………………………………………Page No. 21

Figure 4.2.2: Flowchart Part 2 ………………………………….…………………..Page No.22

Figure 5.1.1: Employee Table ……………………………………….……………..Page No.24

Figure 5.1.2: Ganache Event Log …………………………………………………..Page No.25

Figure 5.1.3: Ganache Event Log …………………………………………………..Page No.26

Figure 5.2.1: Employee Login ………………………………………………….…..Page No.27

Figure 5.2.2: Employee Resume Part 1 ……………………………………………..Page No.27

Figure 5.2.3: Employee Resume Part 2 ……………………………………………..Page No.28

Figure 5.2.4: Employee Resume Part 3 ……………………………………………..Page No.28

Figure 5.2.5: Employee Resume Part 4 ……………………………………………..Page No.29

Figure 5.2.6: Employer Login …………….………………………………………..Page No.29

Figure 5.2.7: Employer Dashboard ………..………………………………………..Page No.30

Figure 5.2.8: Register Page …………………….…………………………………..Page No.30

Figure 5.2.9: Ganache Accounts…………………….……………………………..Page No. 31

Figure 5.2.10: Ganache Event Log……..…………………………………………..Page No. 31

Figure 5.2.11: Ganache Transaction Log…………………………………………..Page No. 32

Figure 5.2.12: Meta Mask…………………………………………………………..Page No. 33

Figure 5.3.1: Login Failure………………………….……………………………..Page No. 35

Figure 5.3.2: Register Failure …………………………..…………………………..Page No.36

Figure 5.3.4: Successful Login ……………………………………………………..Page No.37

Figure 5.3.5: Password not accepted 1.……………………………………………..Page No. 38

Figure 5.3.6: Password not accepted 2……………………………………………..Page No.39

Figure 5.3.7: Login Failure……………………………….………………………..Page No. 40

Figure 5.3.8: Successful Login ……………………………………………………..Page No.41

List of Tables

[Table 2.1: Literature Survey](#_Toc354226615) Page no 5

[Table 3.5.3.1: Project Planning (Part 1)](#_Toc354226615) Page no 15

[Table 3.5.3.2: Project Planning (Part 2)](#_Toc354226615) Page no 16

Abbreviations and Symbols

1. UML: Unified Modelling Language

2. RAD: Requirement Analysis Document

3. FAD: Feasibility Analysis Document

4. SQL: Structured Query Language

5. XML: Extensible Mark-up Language

6. GUI: Graphical User Interface

**Chapter 1**

**Introduction**

* **Introduction**

Bad hires can cost billions of dollars in revenue for companies annually. The rates of theft, malpractice and other crimes can lead to a loss of productivity as well as the reputation of the organisation. In the U.S., losses pertaining to embezzlement at workplace leads to a loss of $4 billion annually. Close to 75% of workers stole repeatedly from the workplace according to the U.S. Department of Commerce figures. More than 30% of businesses, according to data, fail because of bad hires. Resume verification is the only way to avoid detrimental consequences of bad hires. With Resume verification, one can ensure that what is stated in their resume and what information produced is consistent and true. Exaggerated claims are identified and corrective actions can be ensured. As per Society for Human Resource Management, a bad hire translates to costs as much as five times the salary of the wrong candidate. Particularly when hiring key executives, great caution with resume verification is critical.

Successful resume-verification can identify fraudulent degrees, incorrect employment and graduation dates, inflated salary histories and false job titles—issues that may help you determine a candidate is not right for you.

But what goes into a successful resume verification process? An important factor is the quality of the information provided by candidates in both the applicant release form and employment verification form (usually included on a job application or background screening questionnaire). If complete and correct, this information can be particularly helpful for completing background checks. However, if it’s incomplete or incorrect, it can hinder the verification process.

* **Motivation**

The conventional method of Resume Verification by Human Resource and Recruitment teams is a laborious and an arduous task**.** It is a painstaking process and it requires going each and every component of the resume, then contacting the employer of the applicants

previous workplace and then verifying. This is a multi-staged process and often takes up

to weeks if not months. Through our project we would reduce this time by a significant

amount. This would benefit not only the recruitment teams of the company by reducing

their effort, but also help applicants find jobs quickly and easily.

* **Problem Definition**

The main problem definition is to about the painstaking task of evaluating and verifying each and every resume and all the contents of the same. Verifying the components of the resume can vary from place to place and from situation to situation. Often what is noticed is that a majority of people lie blatantly on their resumes and there is absolutely no way of verifying. Implementing System for Employee Verification System using Blockchain, which is our project would help ease this process by verifying each and every entry on any resume.

* **Objectives of Project**

The objectives of this project are:

* To verify resume for each and every user on the platform.
* To reduce the efforts of the Recruitment Team of verifying each and every resume.
* To increase the efficiency of the Recruitment Team and the process of recruitment.
* To provide a Reputation Score to each resume which could be used to evaluate an profile.
* Integration of Blockchain would help us maintain a sense of trust between the users and would substantially decrease the chance of any distrust on the resume.
* **Scope of the project**

The scope of this project is that it will be helpful to the Human Resource and Recruitment department of all companies, as well as to the people who want to apply for jobs and recruitments. It will be helpful to Human Resource departments as it shall reduce the time and efforts invested by actual humans in verifying each and every component of the resume. It shall benefit the applicants for various jobs as the verified resume will require lesser investigation thus saving them time and it would also help them connect with all companies on our Database. The scope of this project can also be to in future integrate with existing job portals like Naukri.com and even LinkedIn. It would help these service providers to properly verify each and every resume and also would help decrease the recruitment time and in turn help the applicants who shall be using this portal in order to apply for jobs and other posts.

* **Application of project**

The project has a lot of scope of becoming something revolutionary, purely because of the sheer application that it incorporates due to the constantly booming data sources. It is highly possible to expand the scope of the project to a wide array of applications. The project shall be useful in we can verify each and every resume created and submitted in the system. The system shall then check the credibility of each and every point mentioned in the Resume, and would then provide a verified mark against the same. The task of checking and validating each and every field on the resume is not needed now and the Recruitment Team can directly evaluate the resume there by saving a lot of time and effort.

The application can be extended to verify even the educational qualifications of every user by collaborating with various boards like ICSE, HSC, SSC etc.

Even institutes offering graduation and post-graduation degrees like Mumbai University and University of Pune to verify the same.

* **Projects Impact Analysis**

By implementing this project not only are we enabling companies to save precious time and money, but also reducing the laborious task of verifying each and every component of the resume which is otherwise done by the Human Resource and Recruitment Team. Also, since the project would consist a huge database of both Employers and Employees, hence this software can be used as a platform to interact with prospective employers. This would help applicants land jobs easily and help in reach out more and different companies. Our project would help the organizations in recruit new employees in a faster and a much efficient way by providing a Blockchain Verified Resume for each and every user on our Portal. In totality our project would help both employees and employers find and place new jobs.

* **Expected Outcome**

The conventional method of Resume Verification by Human Resource and Recruitment teams is a laborious and an arduous task**.** It is a painstaking process and it requires going each and every component of the resume, then contacting the employer of the applicants

previous workplace and then verifying. This is a multi-staged process and often takes up

two weeks if not months. Through our project we would reduce this time by a significant

amount. This would benefit not only the recruitment teams of the company by reducing

their effort, but also help applicants find jobs quickly and easily.

* **Organization of the Project report**

The organization of the project is as follows:

* Chapter 1: It consists of the introduction to the project. The problem definition of the project is explained. Also, the motivation for the project, scope, objectives, outcomes and impacts of the project have been discussed.
* Chapter 2: This chapter mainly deals focuses on the literature review and the proposed work of the project.
* Chapter 3: This chapter deals with the initial stages of the project such as requirement gathering, planning and analysis.
* Chapter 4: This chapter focuses on the design part of the project. It consists of the Data Flow Diagram (DFD), UML Diagrams such as Use Case Diagrams, Activity Diagram, Class Diagram, Flowchart, and System Architecture Diagram
* Chapter 5: This consists of the conclusion of the project.

**Chapter 2**

**Proposed Work and Literature Review**

* **2.1 Literature Survey**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Title | Author | Year | Analysis | Gap |
| Methods and systems for employment and education verification using blockchain | [Jaspreet Randhawa](https://patents.google.com/?inventor=Jaspreet+Randhawa) | Oct. 11, 2018 | How to use blockchain to deploy verification system | Lack of explanation about how technology stack may affect the effectiveness |
| Computerized method and system for creating a digital resume | [Daniel Jacobus deWaal](https://patents.google.com/?inventor=Daniel+Jacobus+deWaal) | May 8, 2014 | Advantages and steps of creating digital resume | Lack of research on security threats to digital resumes |
| A system and method for creating a database of independently validated educational and work experience profiles (curricula vitae) using blockchain smart contracts | Gary MckayAdi BEN-ARIBrian Mcnulty | 14 September 2017 | Detailed information about usage and advantage of blockchain from creation and verification of CVs | No sound method for verification of institution or individuals verifying the claims |
| FEEDBACK BASED REPUTATION ON TOP OF THE BITCOIN BLOCKCHAIN | Davide Carboni | *5 Feb 2015* | Points out advantages of reputation systems on a blockchain system | Fails to provide a wider spectrum of application of feedback-based reputation system |
| A Study on Privilege Elevation Attack Management for Smart Transaction Security on Blockchain Etherium Based System | Min, Youn-A | 2019.04.30 | Gives insights on prevention and protection systems in Etherium blockchain against various attacks | Fails to explain other Blockchain technologies and why they are not as effective. |
| Decentralizing Privacy: Using Blockchain to Protect Personal Data (2015) | Guy Zyskind; Oz Nathan; Alex 'Sandy' Pentland | 2015 | Data Ownership- The user cannot deny ownership to any verified data.  Data Transparency and Auditability – The any change made onto the system will be stored in a general ledger which can be viewed by anyone | Fails to completely validate data entered or existing for each user. |
| Etherium Smart Contract Development | Mayukh Mukhopadhyay | --- | Implementation of Smart Contracts for recording transactions Decentralizing Privacy: Using Blockchain to Protect Personal Data |  |
| Introducing Ethereum and Solidity | Chris Dannen | --- | Solidity to Implement Smart Contracts on Etherium | Using solidity to program smart contracts on the Etherium blockchain |

*Table 2.1: Literature Survey*

* **2.2 Proposed Work**

Users are requested to register on the portal and then login. Users would be then asked to fill out their Personal Information, followed by Educational Details, followed by Work Experiences and Extra Curriculars. All these fields are firstly stored in the databases. This stored data is then verified for each and every component of the resume.

This project would then generate a Verified resume, which would consist all the components verified by the user. Each resume would be sent across to with a score.

This score shall be calculated by the system based on Negations and Reputations.

The verified resume would help both parties in reducing their time and fastening this process.

**Chapter 3**

**Requirement Gathering, Analysis and Planning**

**3.1 Requirement Specification (RAD)**

**3.1.1. Functional Requirements**

1.) Interface Requirements:

Field 1: at the first level accepts problem images from user.

Field 2: display the recommendations suggested for the user.

2.) Business Requirements:

Template must be entered and mapped before a query can be processed.

All personnel authorised to access the database and system model will be trained accordingly.

3.) Regulatory/Compliance Requirements:

The database containing the data set mapping of problem-solutions will have a functional audit trail.

The system will limit access to authorized users.

4.) Security Requirements:

Customers may only send image and retrieve information.

Members of Data Group may only make changes according to new data taken, any previously mapped solution may not be edited if results provided are accurate. These members are not authorized for deletion of customer requests.

**3.1.2. Non-Functional requirements**​-

1.) Usability

Usability assessment of troubleshooting systems is a necessary and challenging problem. Machine learning techniques are effective tools for usability assessments.

NLP and Image processing can help higher precision in analysis of the issue and providing a recommendation.

2.) Reliability

Different templates of the same problem and image recognition helps narrow down to the problematic area and the solution mapping avoids any multiplicity resulting in user confusion. The extraction of features with image recognition and the solution data set, and the adaptiveness to new problem templates makes the system not only efficient but also accuracy incrementing and effective.

3.) Performance

The recommendation system allows users to upload image of issue terminal and the

System will provide the solution steps to the customers, more efficiently and effectively, without facing many hassles or disadvantages.

4.) Portability

The system helps the users to deal with their daily I.T. muddles with a click of a button on their device. The system can be accessed from any device with an active internet service. Thus, users can then be uninterrupted from their work for a long time, which is the case currently.

**3.2 Feasibility Study (FAD)**

**Technical Feasibility:**

Technology used in the front end: Express JS

Software Required: Php, MongoDB,

Editor: Sublime

**Economic Feasibility:**

A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful. Our aim is to achieve maximum accuracy from our recommendation system with the help of an adaptive environment. This will reduce the need of checking the problems manually, thus reducing efforts, time invested and ultimately cost of the whole process whilst increasing efficiency. Market Survey: The coming revolution is spearheaded by [image recognition and AI technologies](https://www.traxretail.com/technology/).  A key challenge for users of image recognition technology is finding the proper solution that will address their specific consumer goods applications. On an observation of the local requirements, recommendation systems is accessible, convenient and accurate addresses the daily technical issues hampering their routine work.

**Product attributes to look for:**

* **Reliability:** The system adapts according to previously provided results and expectations following a spiral methodology for requirement gathering, shortlisting and updating.
* **Positioning:** Problems met by non - technical people can be solved in a hassle-free manner without the users being trapped in the middle of multiple possibilities or communication barriers with the experts.

**3.3 Methodology**

There are primarily two approaches for achieving better troubleshooting results using the proposed technique:

• A. Resume Creation

To create a resume on our website, a user has to first create an account on our website and select his account type which will be an ‘employee’ or an ‘employer’. These will be the two roles on our website. Any claims made by the employee will be verified by the organization or his peers/supervisor. Upon logging in, the user has to complete a resume form, the data from which will be stored into the mongo DB database.

B. Database Storage

The mongo DB database will store login credentials, resume details, user details, score and multiple other data fields. The data from the mongo DB server will be inserted to and extracted from using REST API service. We are intended on using REST API as we are hopeful of further developing a mobile application and expand forward from our website. [3]

C. Server Scripting

We are using NodeJS and PHP for our server-side scripting. The major use of NodeJS is to fetch and post to our API’s from mongo DB. NodeJS will be mainly used for data transmission. [3] Also, PHP is being used as it works very well with WordPress, as WordPress itself is coded in PHP. Thus, PHP allows us to implement a variety of custom features on WordPress, as WordPress doesn’t match all our needs. WordPress has a variety of plugins we can use, but however a lot of additional programming is needed through PHP to make them work to our needs.

D. API Calls:

Every interaction with the database will happen through a REST API call. This is implemented keeping in mind future expansion of the website to other platforms and create a modular approach. Currently a fetch and a storage call will be made to the database. This might increase later as development continues.

E. Score System

Every profile will have a corresponding score associated with it. [1] This score can be changed via reps and negs. A rep is a positive increase in the users score, while a neg is a decrease in the score. We have reps and negs [2] tied to events in our website that shall trigger them. These have been discussed in depth further later.

F.Verification using Blockchain  
Every verified claim on the website and the particular change in score tied to it shall be updated on the blockchain. We will be using Ethereum as our blockchain and will be using Ethereum smart contracts to reflect the changes made to the rep score and verification status of a claim onto the Blockchain.

**3.4 Technology**

**3.4.1. Express JS**

Express.js, or simply Express, is a [web application framework](https://en.wikipedia.org/wiki/Web_application_framework) for [Node.js](https://en.wikipedia.org/wiki/Node.js), released as [free and open-source software](https://en.wikipedia.org/wiki/Free_and_open-source_software) under the [MIT License](https://en.wikipedia.org/wiki/MIT_License). It is designed for building [web applications](https://en.wikipedia.org/wiki/Web_application) and [APIs](https://en.wikipedia.org/wiki/API). It has been called the [de facto standard](https://en.wikipedia.org/wiki/De_facto_standard) server framework for [Node.js](https://en.wikipedia.org/wiki/Node.js).

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

With a myriad of HTTP utility methods and middleware at your disposal, creating a robust API is quick and easy.

Express provides a thin layer of fundamental web application features, without obscuring Node.js features that you know and love.

Apart from that Many [popular frameworks](https://expressjs.com/en/resources/frameworks.html) are based on Express.

For example, not only does Express JS power a huge number of business sites and blogs, it’s also a very [popular way to create an eCommerce store](https://kinsta.com/wordpress-market-share/) as well! With WordPress, you can create:

* Business websites
* eCommerce stores
* Blogs
* [Portfolios](https://kinsta.com/blog/portfolio-website/)

**3.4.2. MongoDB**

MongoDB is a document-oriented NoSQL database used for high volume data storage. MongoDB is a database which came into light around the mid-2000s. It falls under the category of a NoSQL database.

The features of MongoDB are-

* 1. Each database contains collections which in turn contains documents. Each document can be different with a varying number of fields. The size and content of each document can be different from each other.
  2. The document structure is more in line with how developers construct their classes and objects in their respective programming languages. Developers will often say that their classes are not rows and columns but have a clear structure with key-value pairs.
  3. As seen in the introduction with NoSQL databases, the rows (or documents as called in MongoDB) doesn't need to have a schema defined beforehand. Instead, the fields can be created on the fly.
  4. The data model available within MongoDB allows you to represent hierarchical relationships, to store arrays, and other more complex structures more easily.

MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time

The document model maps to the objects in your application code, making data easy to work with

Ad hoc queries, indexing, and real time aggregation provide powerful ways to access and analyse your data

MongoDB is a distributed database at its core, so high availability, horizontal scaling, and geographic distribution are built in and easy to use

**3.4.3. Amazon web service**

Amazon web service is a platform that offers flexible, reliable, scalable, easy-to-use and cost-effective cloud computing solutions.

AWS is a comprehensive, easy to use computing platform offered Amazon. The platform is developed with a combination of infrastructure as a service (IaaS), platform as a service (PaaS) and packaged software as a service (SaaS) offerings.

Amazon Web Services offers a wide range of different business purpose global cloud-based products. The products include storage, databases, analytics, networking, mobile, development tools, enterprise applications, with a pay-as-you-go pricing model.

**3.4.4. Ethereum**

Ethereum is a global, decentralized platform for money and new kinds of applications. On Ethereum, you can write code that controls money, and build applications accessible anywhere in the world.

 Ethereum is an open software platform based on blockchain technology that enables developers to build and deploy decentralized applications.

Ethereum is a distributed public blockchain network. Although there are some significant technical differences between the two, the most important distinction to note is that Bitcoin and Ethereum differ substantially in purpose and capability. Bitcoin offers one particular application of blockchain technology, a peer to peer electronic cash system that enables online Bitcoin payments. While the Bitcoin blockchain is used to track ownership of digital currency (bitcoins), the Ethereum blockchain focuses on running the programming code of any decentralized application.

In the Ethereum blockchain, instead of mining for bitcoin, miners work to earn Ether, a type of crypto token that fuels the network. Beyond a tradeable [cryptocurrency](http://blockgeeks.com/guides/what-is-cryptocurrency-everything-you-need-to-know-ultimate-guide/), Ether is also used by application developers to pay for transaction fees and services on the Ethereum network.

**3.4.5. NodeJS**

Node.js is an open-source server-side runtime environment built on Chrome's V8 JavaScript engine. It provides an event driven, non-blocking (asynchronous) I/O and cross-platform runtime environment for building highly scalable server-side application using JavaScript.

Node.js can be used to build different types of applications such as command line application, web application, real-time chat application, REST API server etc. However, it is mainly used to build network programs like web servers, similar to PHP, Java, or ASP.NET.

Features of NodeJS-

1. Node.js is an open-source framework under MIT license. (MIT license is a free software license originating at the Massachusetts Institute of Technology (MIT).)
2. Uses JavaScript to build entire server-side application.
3. Lightweight framework that includes bare minimum modules. Other modules can be included as per the need of an application.
4. Asynchronous by default. So, it performs faster than other frameworks.
5. Cross-platform framework that runs on Windows, MAC or Linux

**3.5 Gantt Chart and Process Model and Client Communication**

**3.5.1. Client Communication**

On communication with Audience:

1. Major problem faced with technology utilization?

Conclusion: Lack of understanding of actual problem and its cause.

2. How does this hamper you daily activities?

Conclusion: The customers have to wait for technical help which is not

instantaneous.

3. What about the existing sources of solutions?

Conclusion: Web search usually results in confusion due to the multiple solutions and causes that are available. The customer help services aren’t always helpful due to their unclear communication, and even when personal help is provided, it delays work to a great extent.

4. What in your view would help you overcome these problems?

Conclusion: A quick, easy, understandable and accessible service which would help us map down the exact problem and its solution would be of great help.

5. Your expectations from this type of system? Conclusion: Any system which would be able to troubleshoot without much description requirement and be able to give solution stepwise would usually solve the delay caused due to minor IT related problems.

**3.5.2. Process Model**

For this reason, the most suitable process model will be the Agile Method.

Agile Methodology unlike the waterfall model, is a practice that promotes continuous iteration.

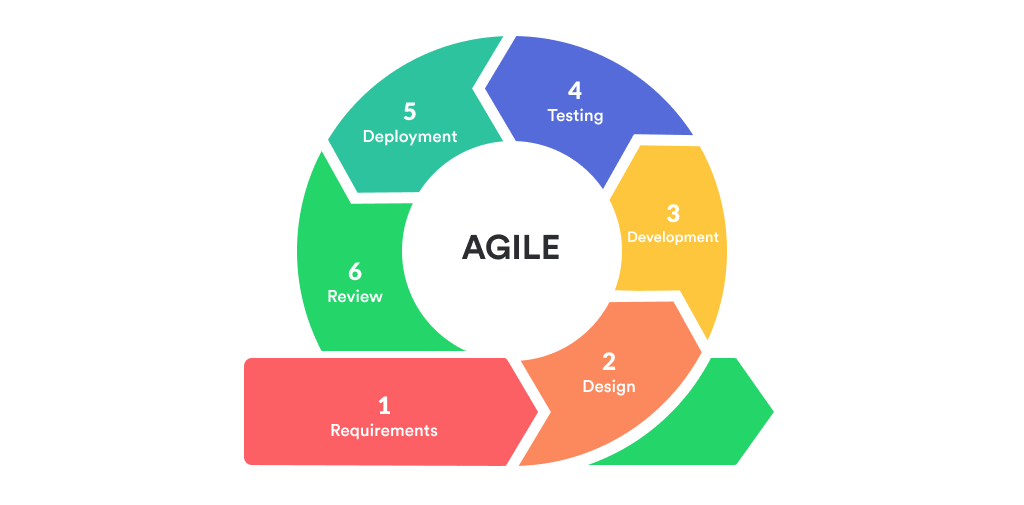
The agile software development emphasizes on four core values.

1. Individual and team interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

Agile typically refers to a professional tester who embraces changes, collaborates well with the business technical team and understands the concept of software document requirement and to derive them. They are willing to learn about the customer so that the tester can know what they can do according to the customer needs as per the software requirements.

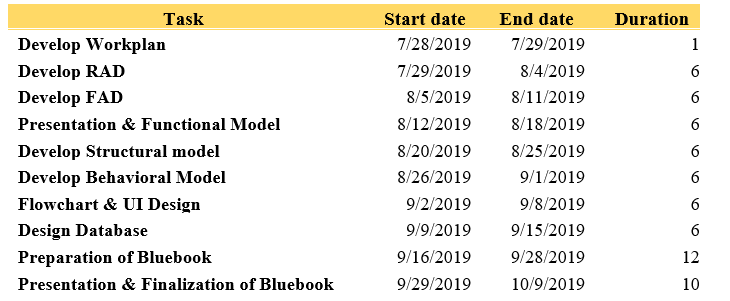
**3.5.2. Principles of Agile Testing**

* Provides continuous feedback. This is needed as he is the person leading the team and to improve their ability and to motivate them, feedback is necessary where they will know their feedback and can work on it.
* Deliver value to customer. Acceptance of tests helps all understanding of what it means on the realization of customer needs.
* Enable face to face communication. Here testers play a major role in bringing the customer and tester together on a face to face communication whereby then the problem can be solved regarding the bug and software’s. A tester can be the ones who physically brings these people together as well the ones who drives the common language between these parties.
* Self-organize. In a true agile team, everyone has the role to act as a tester. They should know how to shift focus from time to time when needed. They have to focus their attention towards verifying done attitude rather than done attitude.
* Focus on people. Testers are basically at the bottom line in a non-agile software team. To them the works are given, particular time to complete is given and the programmers looks as those lower to them.
* Keep it simple. Testers can help the customer to understand how to prominently deliver value. They have to provide a equal balance of iterative testing, just a right confidence to deliver the software.

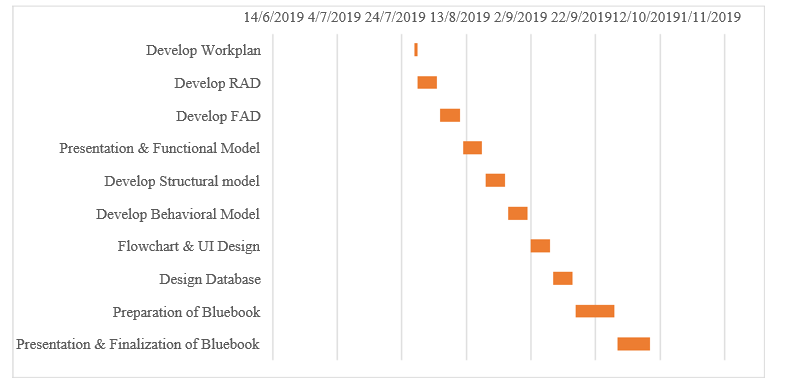


*Figure 3.5.2.1: Agile Development*

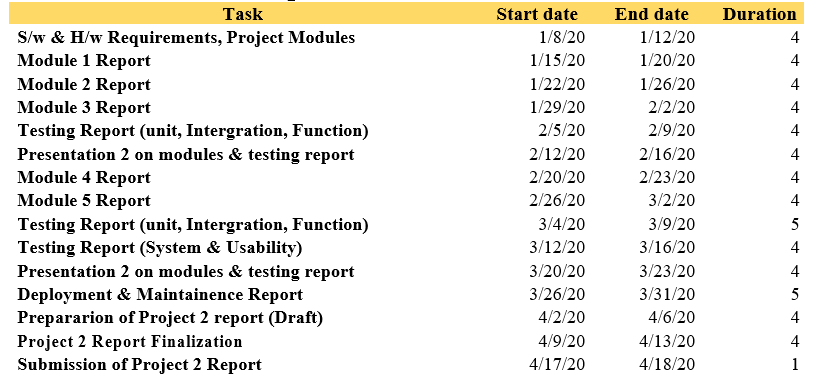
**3.5.3** **Gantt Chart and Process Model and Client Communication**



*Table 3.5.3.1: Project Planning (Part 1)*

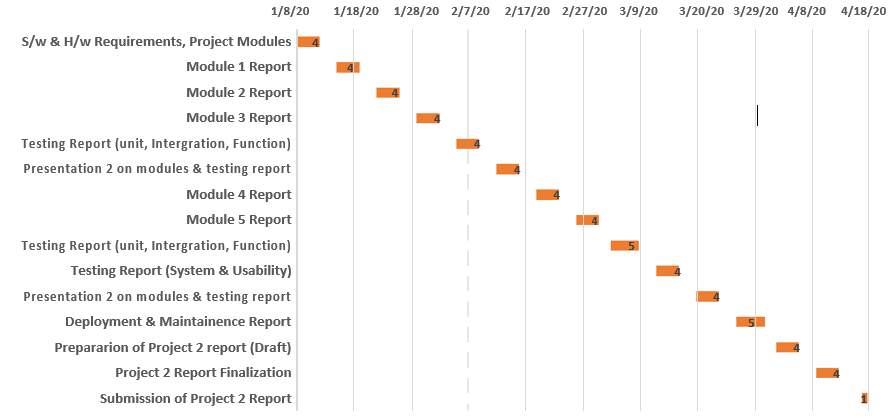


*Figure 3.5.3.1: Gantt Chart (Part 2)*



*Table 3.5.3.2: Project Planning (Part 2)*

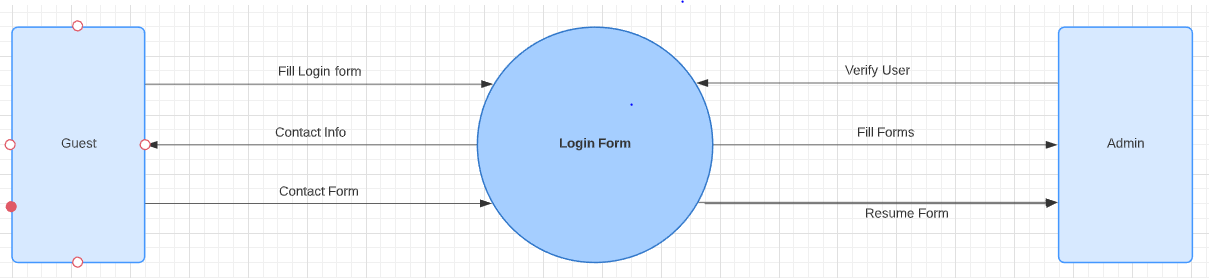
**3.5.3.2 Project Planning and Gantt Chart**



**3.6 System Analysis (functional model, structural model and behavioural model)**

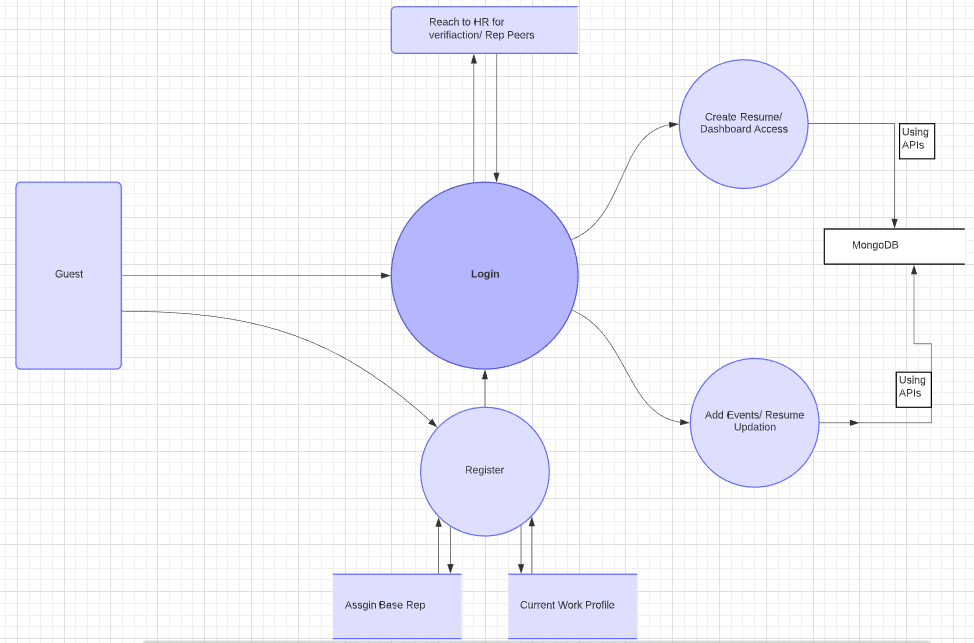
**3.6.1. Data Flow Diagram**

LEVEL 0



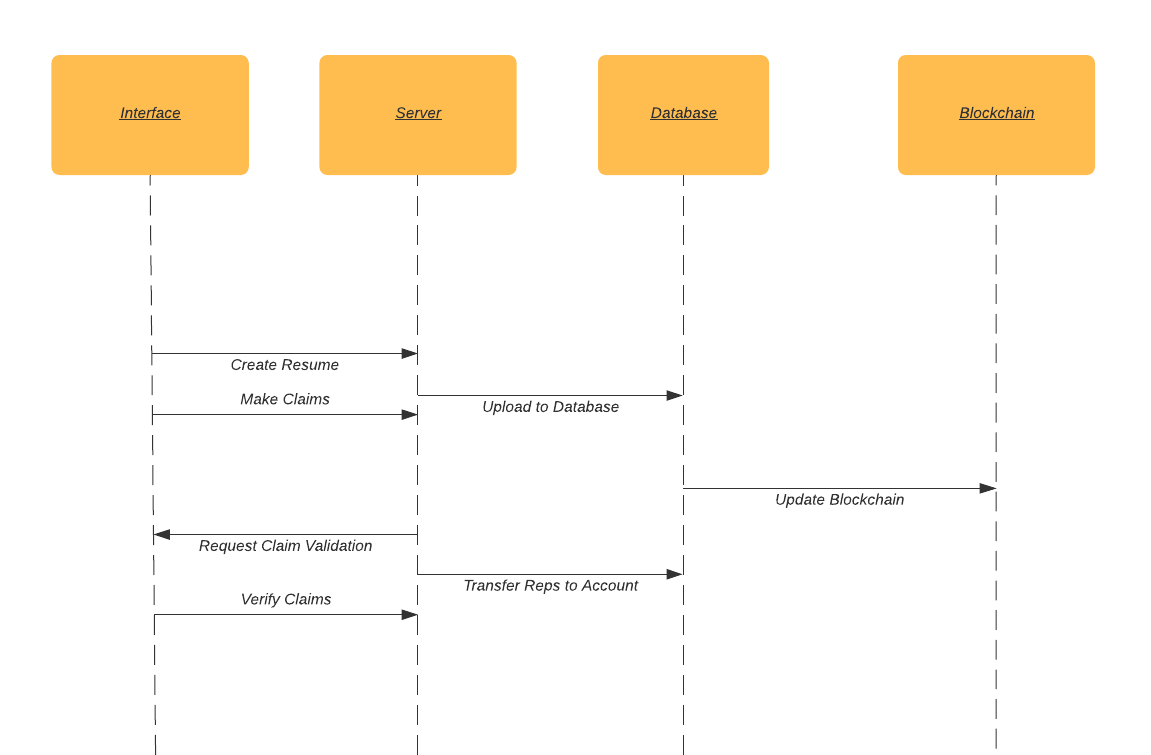
*Figure 3.6.1.1: Data Flow Diagram – Level 0*

LEVEL 1



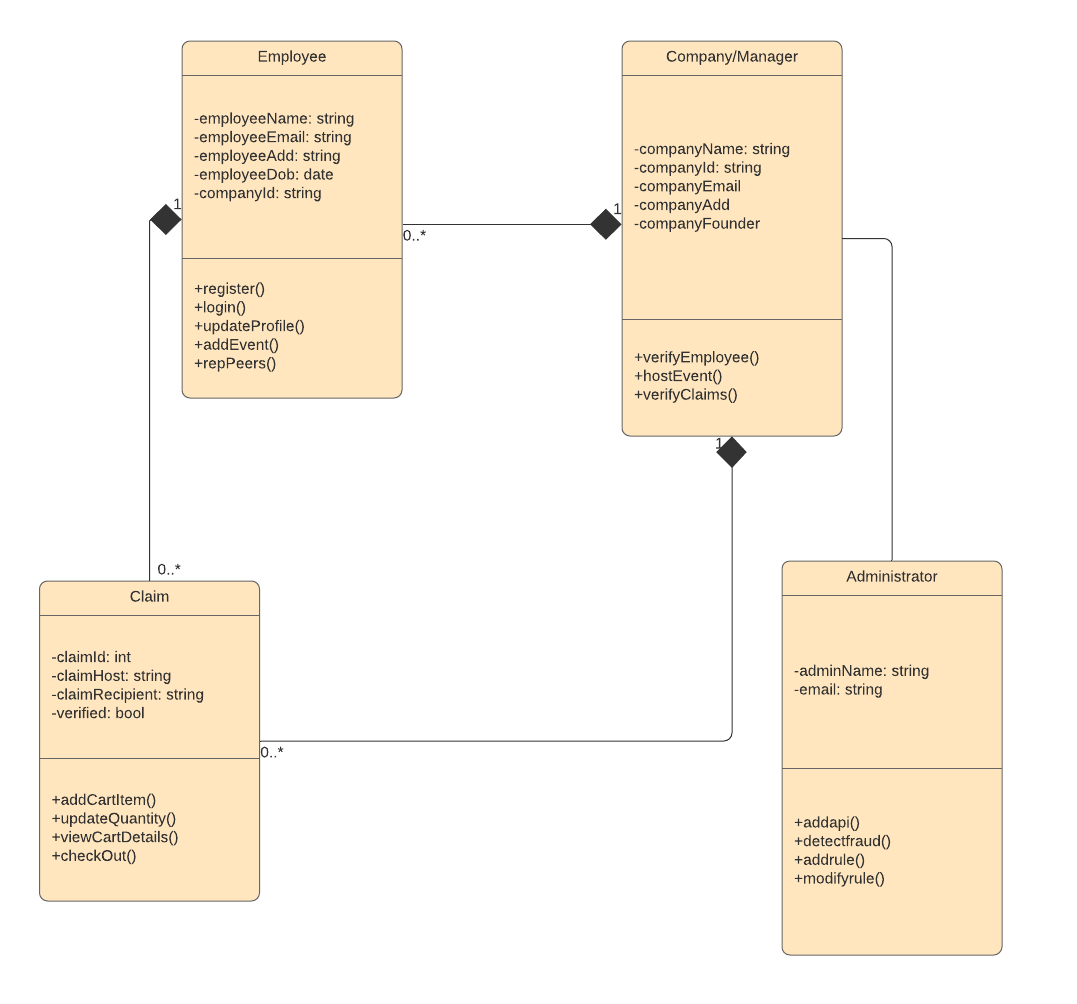
*Figure 3.6.1.2: Data Flow Diagram – Level 1*

**3.6.2. Sequence Diagram**



*Figure 3.6.2: Sequence Diagram*

**3.6.3. Class Diagram**

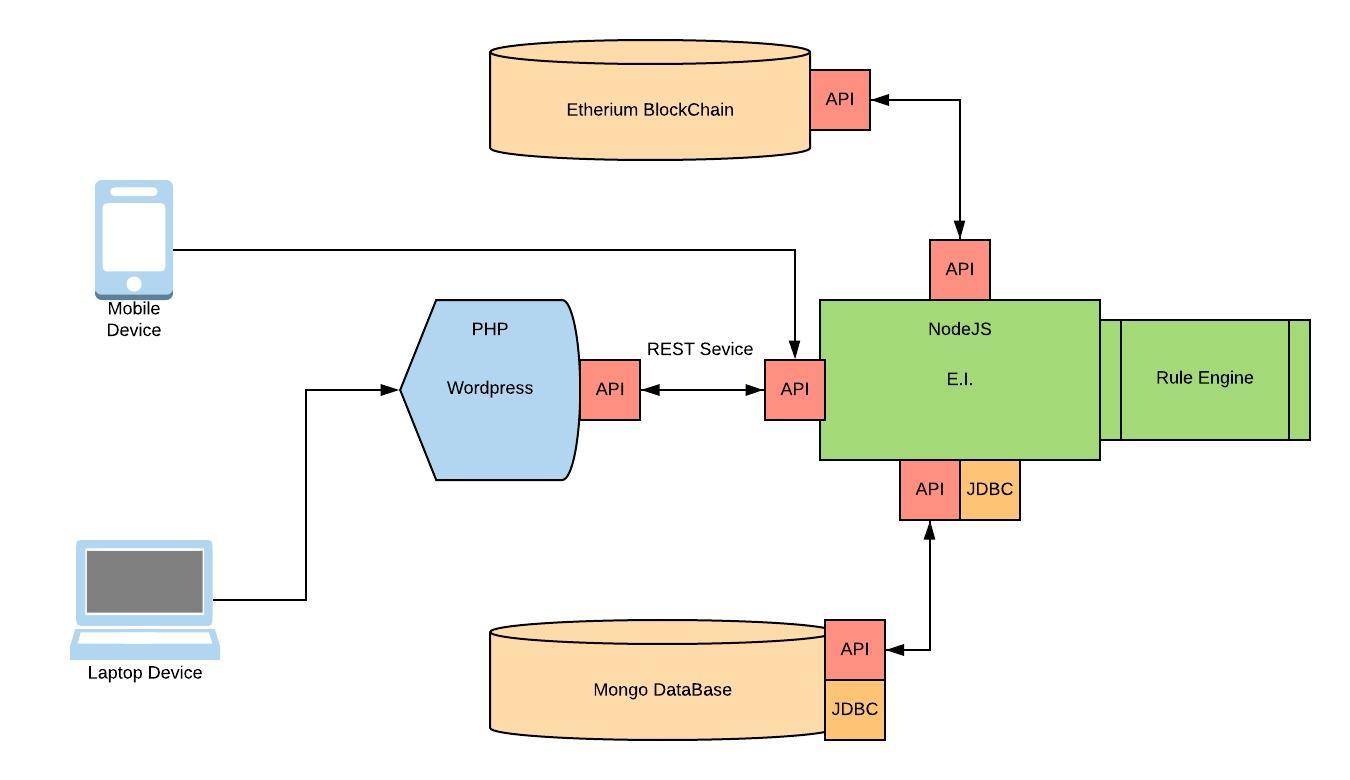


*Figure 3.6.3: Class Diagram*

**Chapter 4**

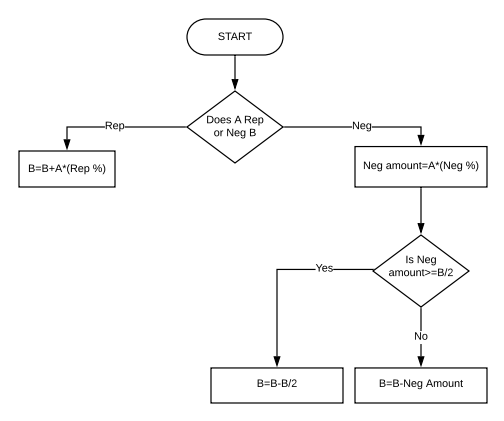
**Design phase**

**4.1 Block Diagram / Architecture**

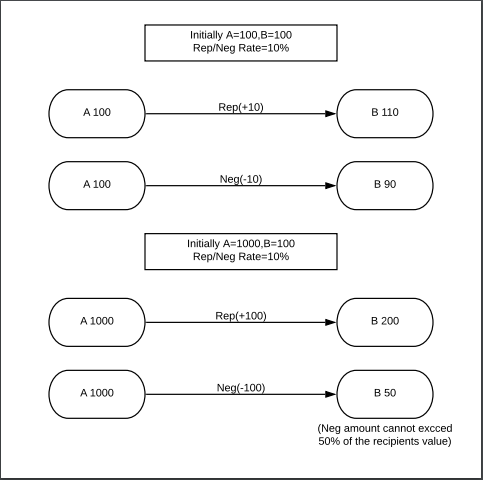


*Figure 4.1.1: Architecture Diagram*

**4.2 Flowchart**

****

*Figure 4.2.1: Flowchart Part 1*

****

*Figure 4.2.2: Flowchart Part 2*

**4.3** **SOFTWARE ARCHITECTURE DESIGN**

* Express JS
* Mongo DB
* Amazon Web Services
* Ethereum
* Node JS

**Chapter 5**

**Design Methodology**

**5.1** **Database Design**

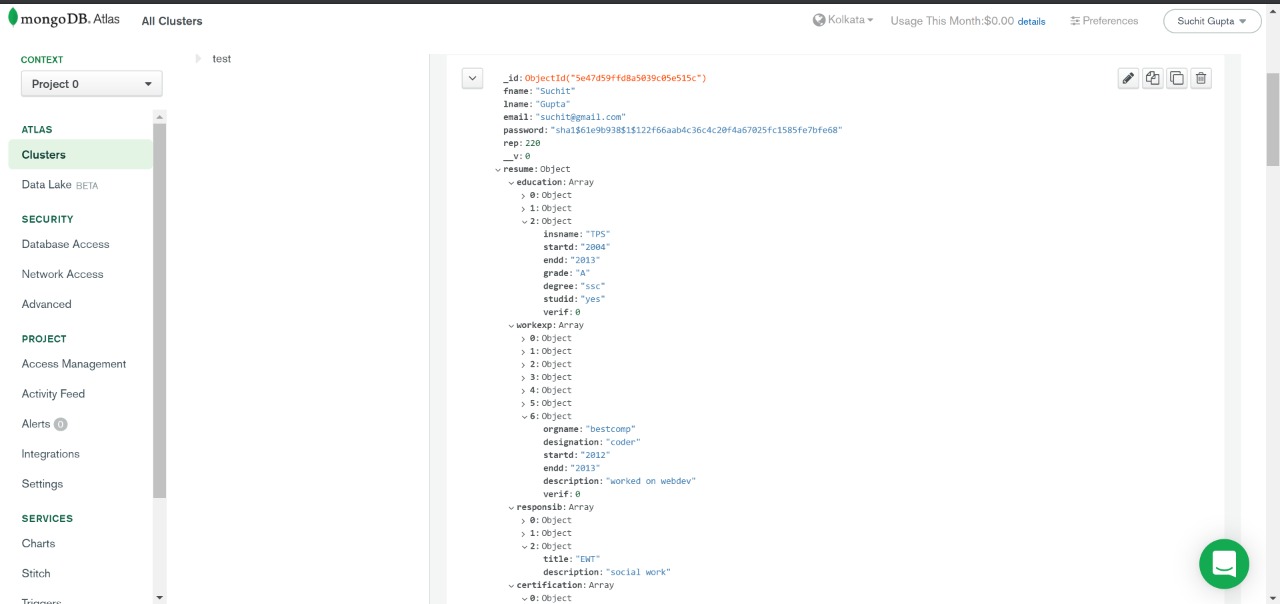
**Purpose**

The purpose of the database design is for producing a detailed data model of a database. This data model contains all the needed logical and physical **design** choices and physical storage parameters needed to generate a **design** in a data definition language, which can then be used to create a **database**.

**Description**

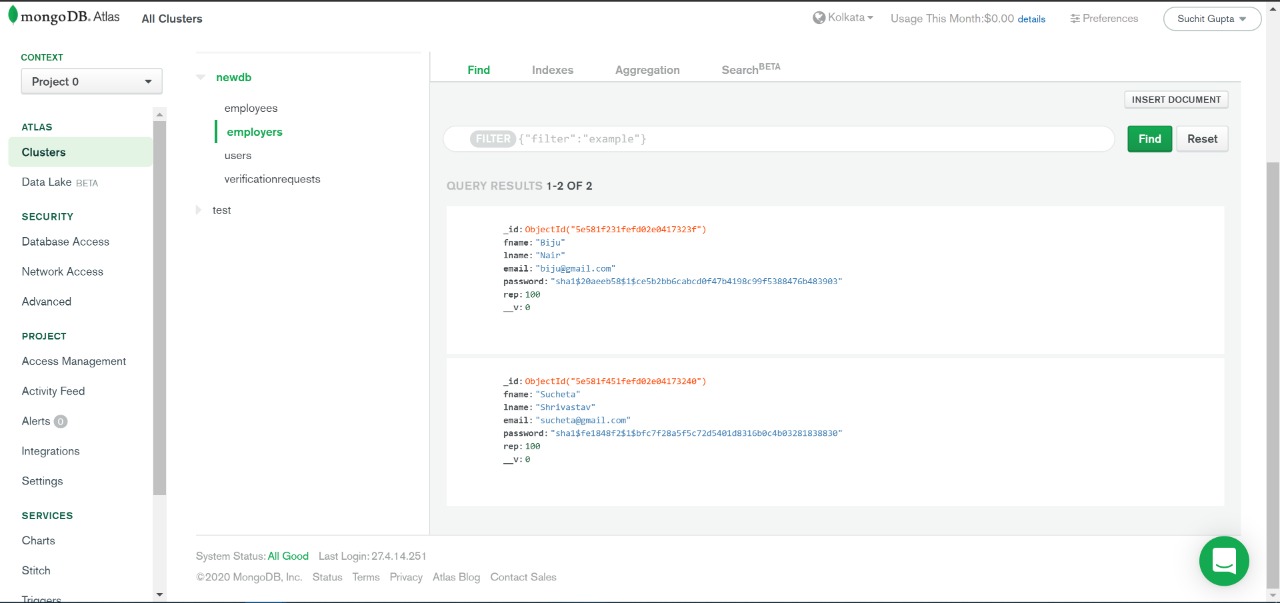
The databases are designed for 3 specific reasons:

* **Employee Table**



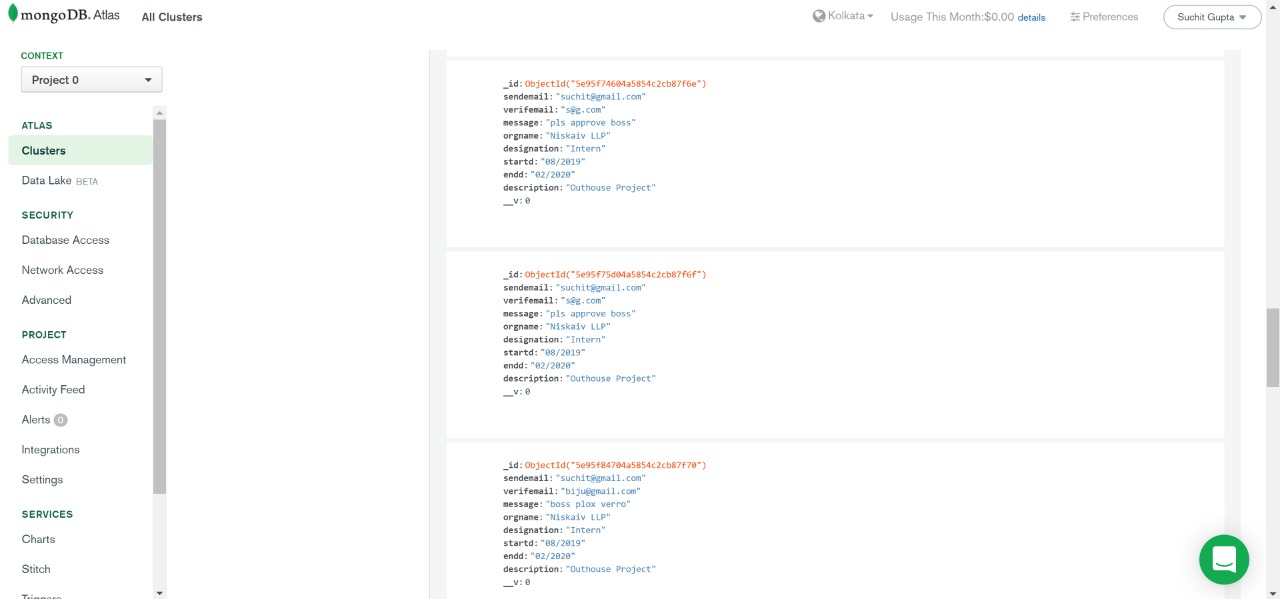
*Figure 5.1.1: Employee Table*

* **Employer Table**



*Figure 5.1.2: Ganache Event Log*

* **Verification Request Table**



*Figure 5.1.3: Ganache Event Log*

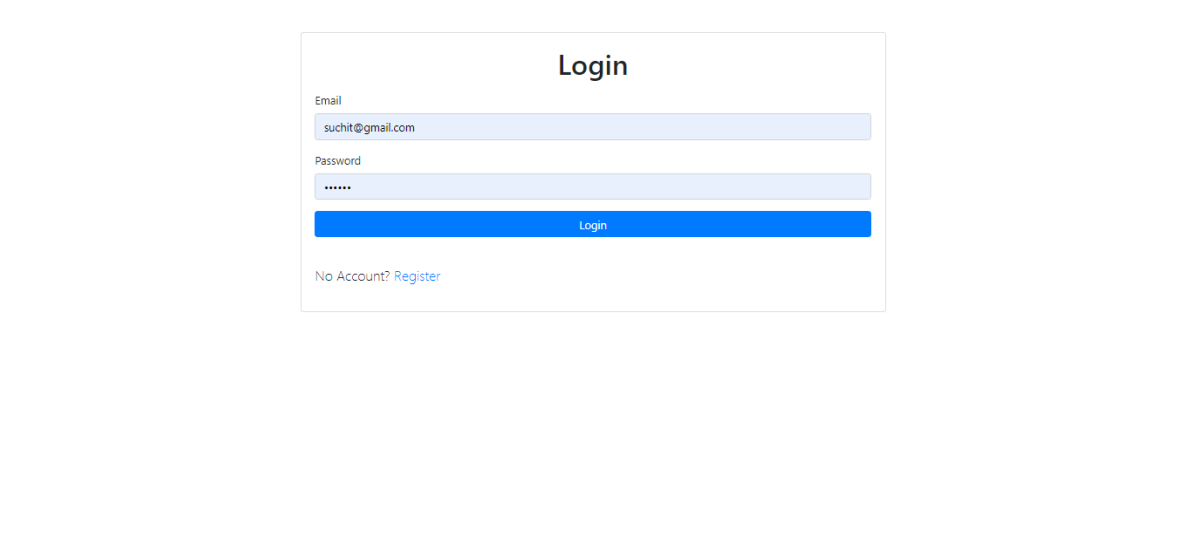
**5.2** **GUI Design**

**Purpose**

The graphical user interface (**GUI**) is a type of user interface that allows admin to interact with the system through graphical icons and visual indicators such as secondary notation, instead of text-based user interfaces, typed command labels or text navigation.

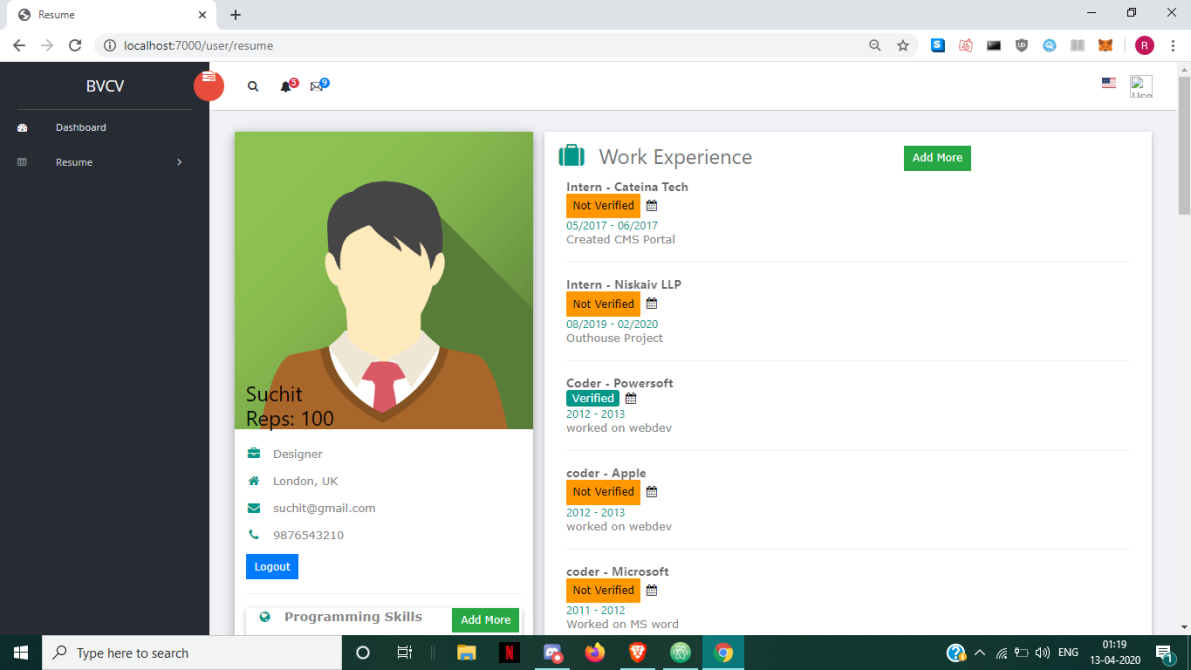
**Description**

**Employee Login:** It will show login page which after right credential forward to Student data.



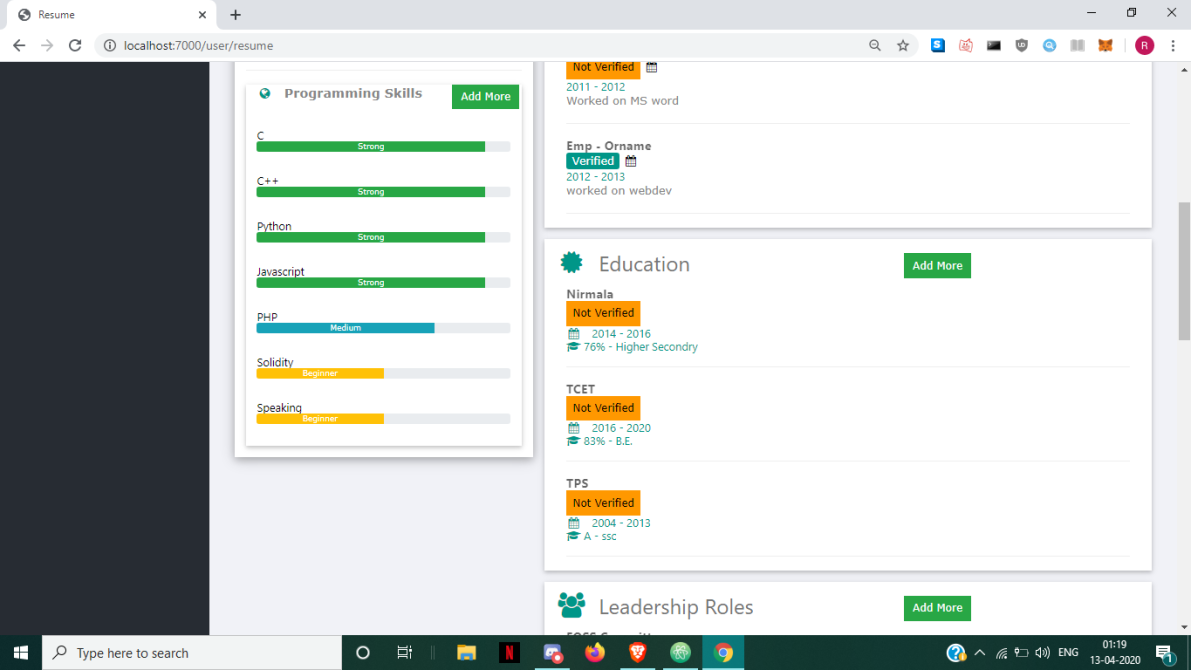
*Figure 5.2.1: Employee Login*

**Employee Resume Part 1:** It will show login page which after right credential forward to Student data.



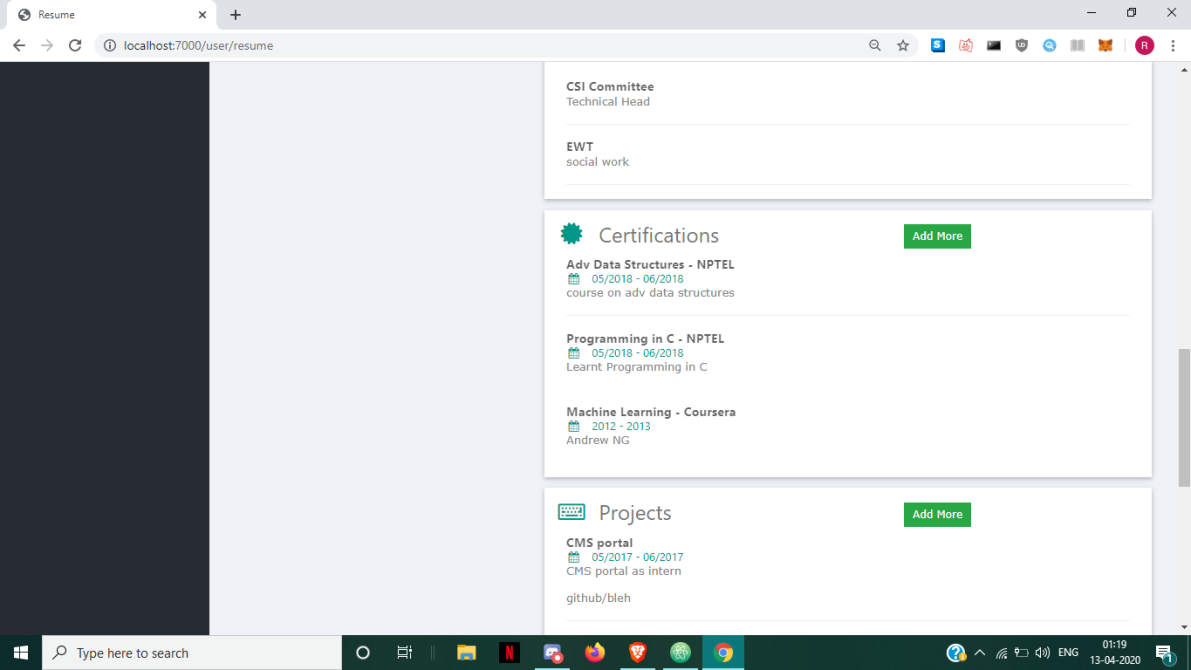
*Figure 5.2.2: Employee Resume Part 1*

**Employee Resume Part 2:** It will show login page which after right credential forward to Student data.



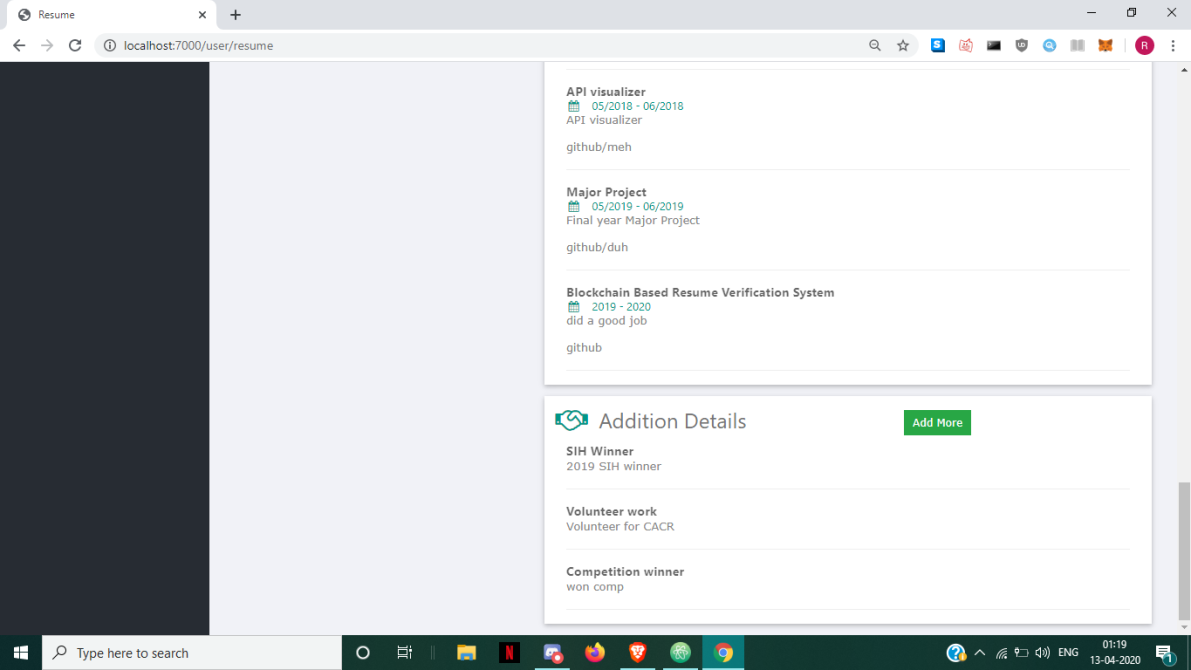
*Figure 5.2.3: Employee Resume Part 2*

**Employee Resume Part 3:** It will show login page which after right credential forward to Student data.



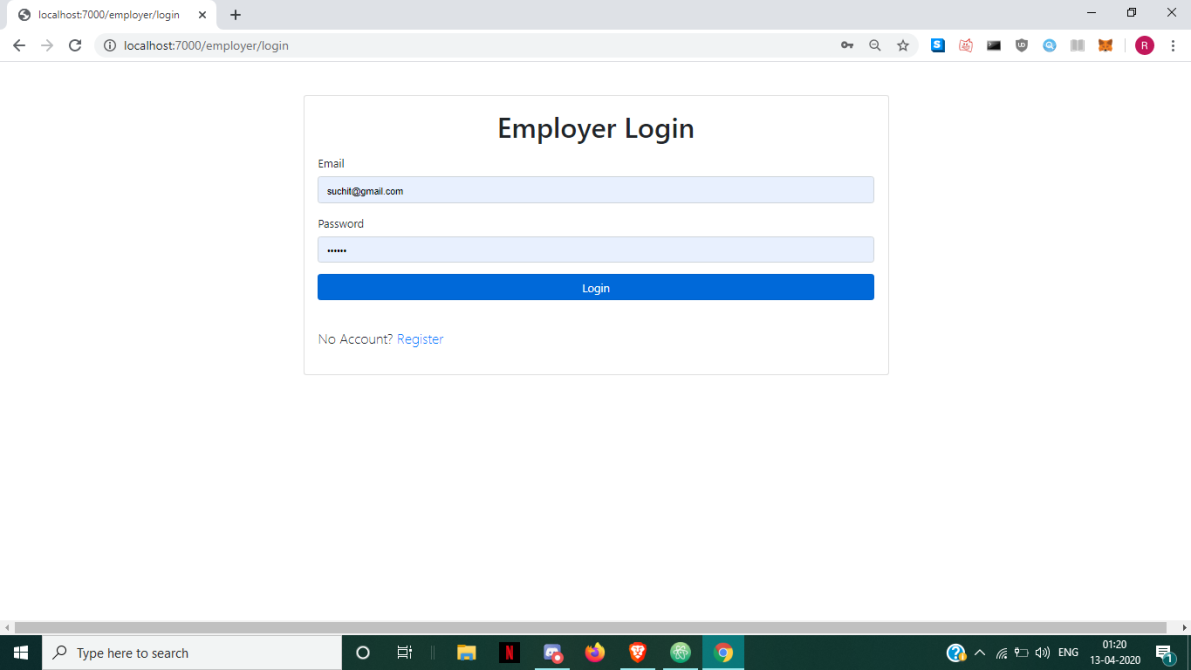
*Figure 5.2.4: Employee Resume Part 3*

**Employee Resume Part 4:** It will show login page which after right credential forward to Student data.



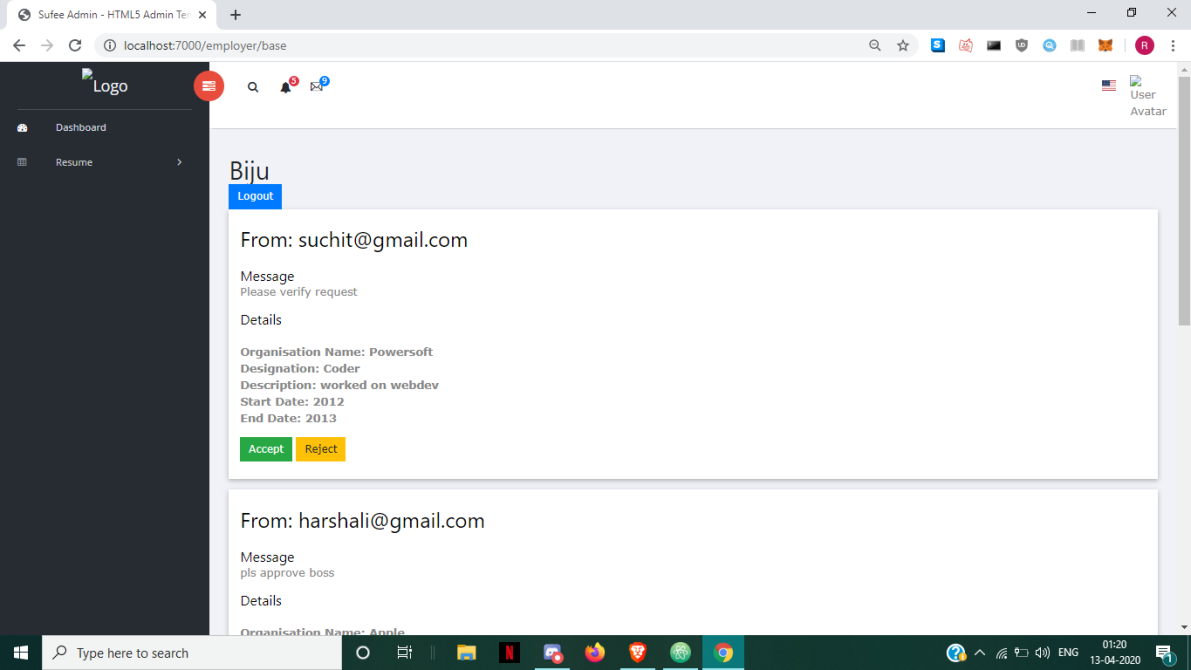
*Figure 5.2.5: Employee Resume Part 4*

**Employer Login:** It will show login page which after right credential forward to Student data.



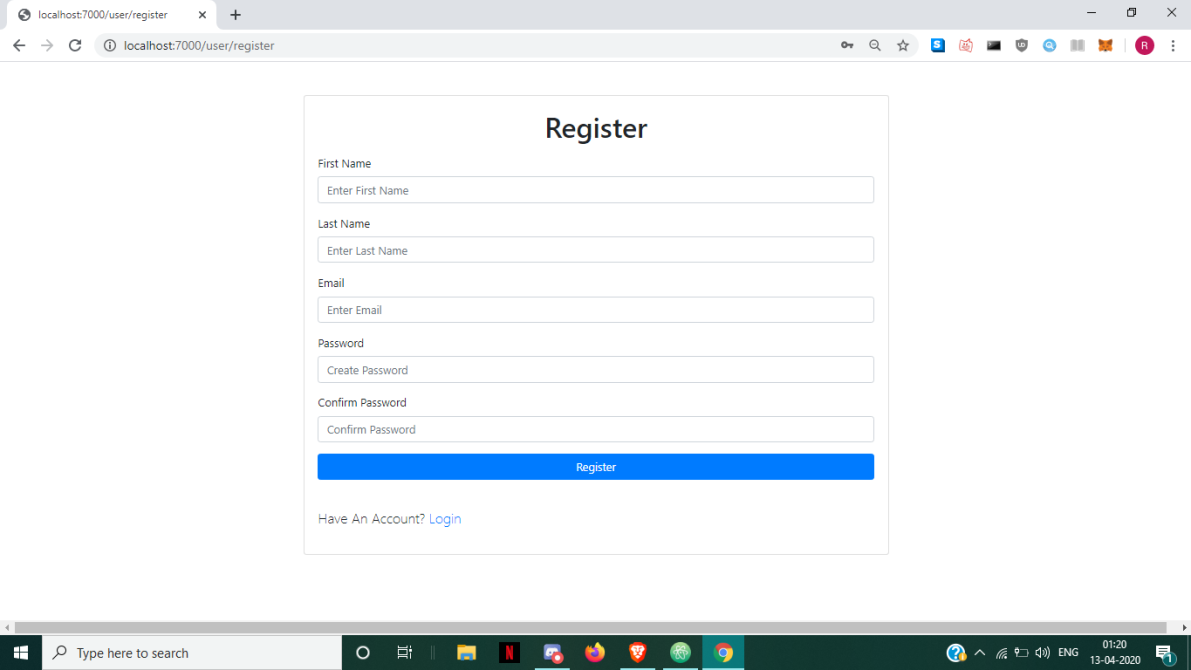
*Figure 5.2.6: Employer Login*

**Employer Dashboard:** It will show login page which after right credential forward to Student data.

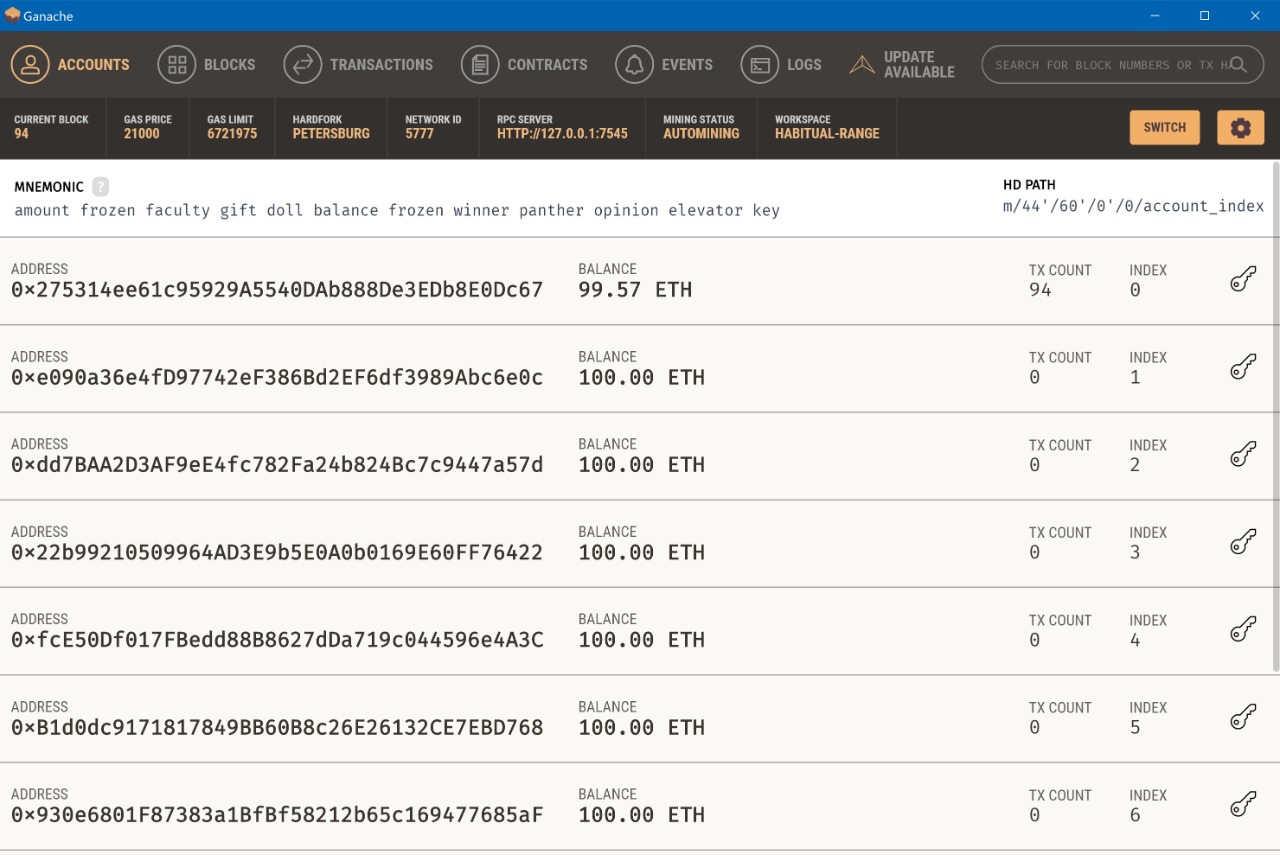


*Figure 5.2.7: Employer Dashboard*

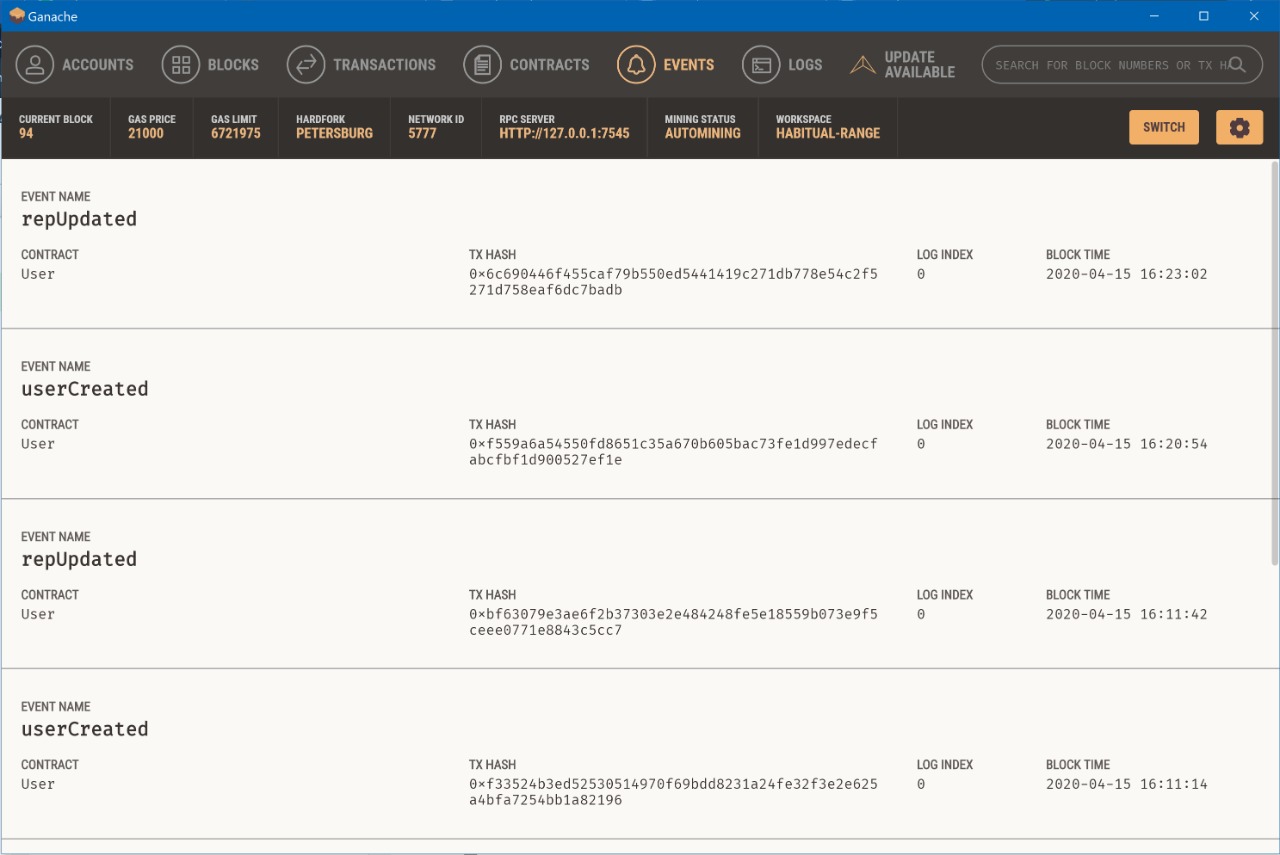
**Register Page:** It will show login page which after right credential forward to Student data.



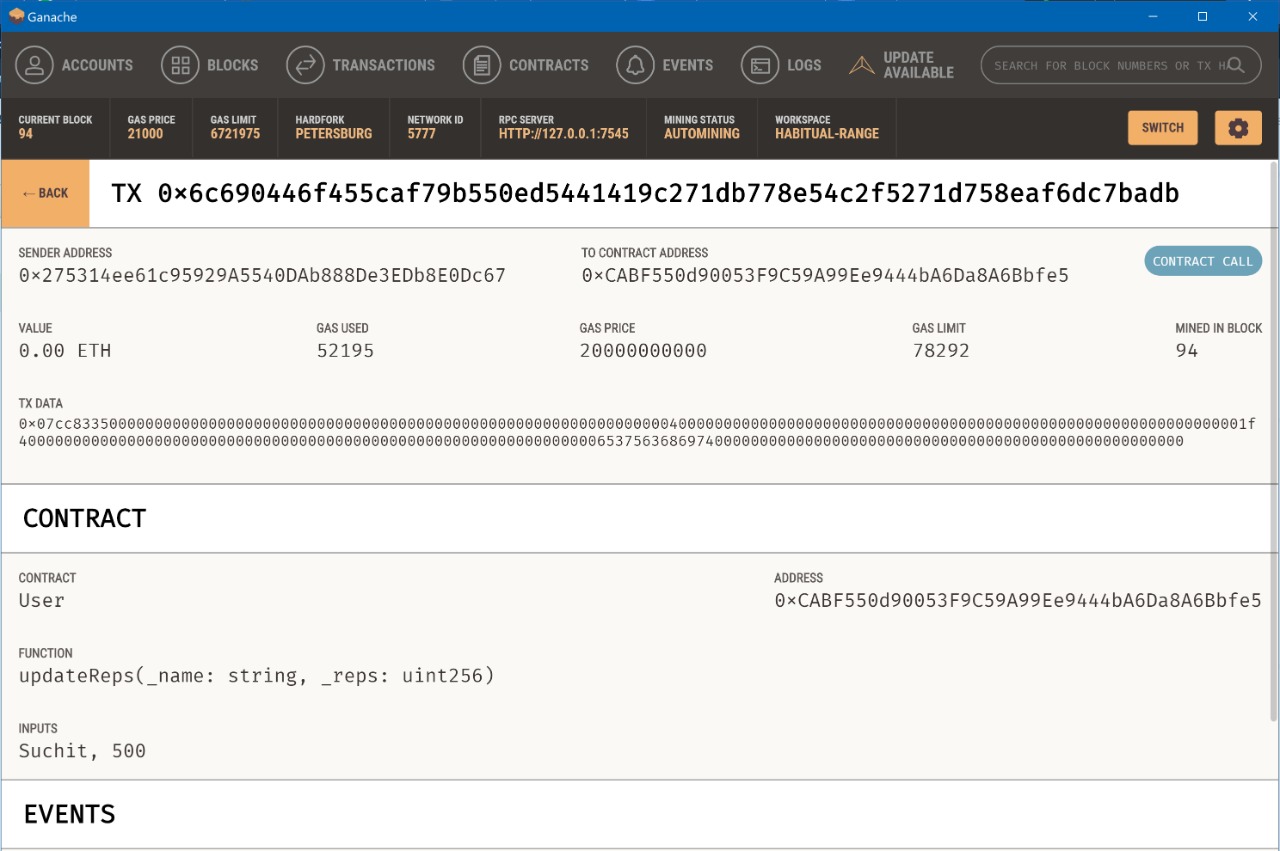
*Figure 5.2.8: Register Page*

**

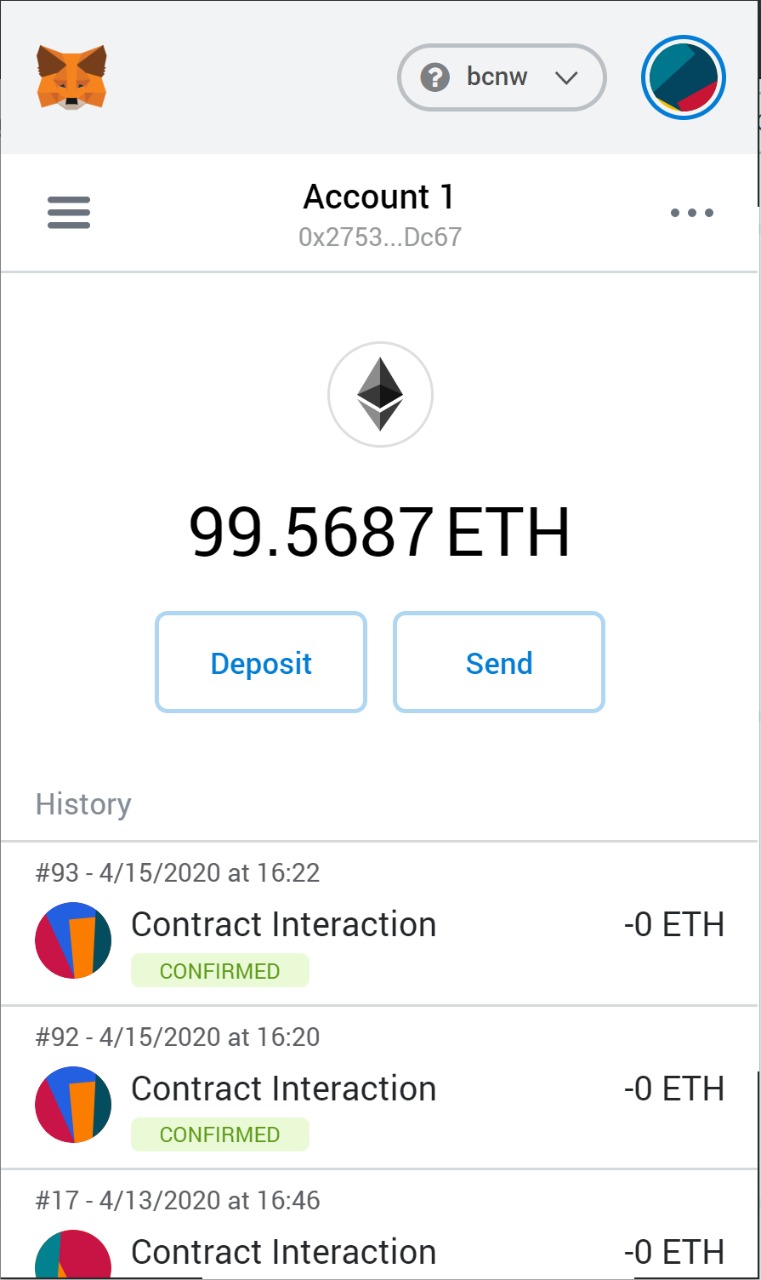
*Figure 5.2.9: Ganache Accounts*



*Figure 5.2.10: Ganache Event Log*



*Figure 5.2.11: Ganache Transaction Log*



*Figure 5.2.12: Meta Mask*

**5.3 Testing Report**

**Unit Testing:** In computer programming field, unit testing is a software testing method by which individual units/chunks of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. We have tested every unit of our system extensively one by one and our system units passed the test cases of ours very efficiently.

**Usability Testing:** Usability is the ease of use and Learnability of a human-made object. At Affordable Usability we simplify that down to "Usability means happy users." We base the usability of the websites that we design along at least five axis of usability goals: ▪ Efficiency ▪ Utility ▪ Ability to learn ▪ Ability to memorise Our system is very simple and it will only require an input to be given after an input is provided we only get a simple result. Hence, it passes our test of usability and simple to learn. There is no use to memorize the system.

**Integrating Testing:** Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. It is performed post unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

**System Testing System Testing:** is a level of software testing where a complete and integrated software is tested. System testing: The process of testing an integrated system to verify that it meets specified requirements. All the modules when integrated into a system functions according to our expectations and hence, our system passes system testing.

**Functional Testing Functional testing:** is a quality assurance (QA) process and a type of black-box testing that bases its test cases on the specifications of the software component under test. Each functionality is tested by feeding them as input and examining the output, and internal program structure is rarely considered (unlike white-box testing). Functional testing usually describes what the system is capable of doing. Functional testing does not imply that you are testing a function (method) of your module or class. Functional testing tests a slice of functionality of the whole system.

**Description:**

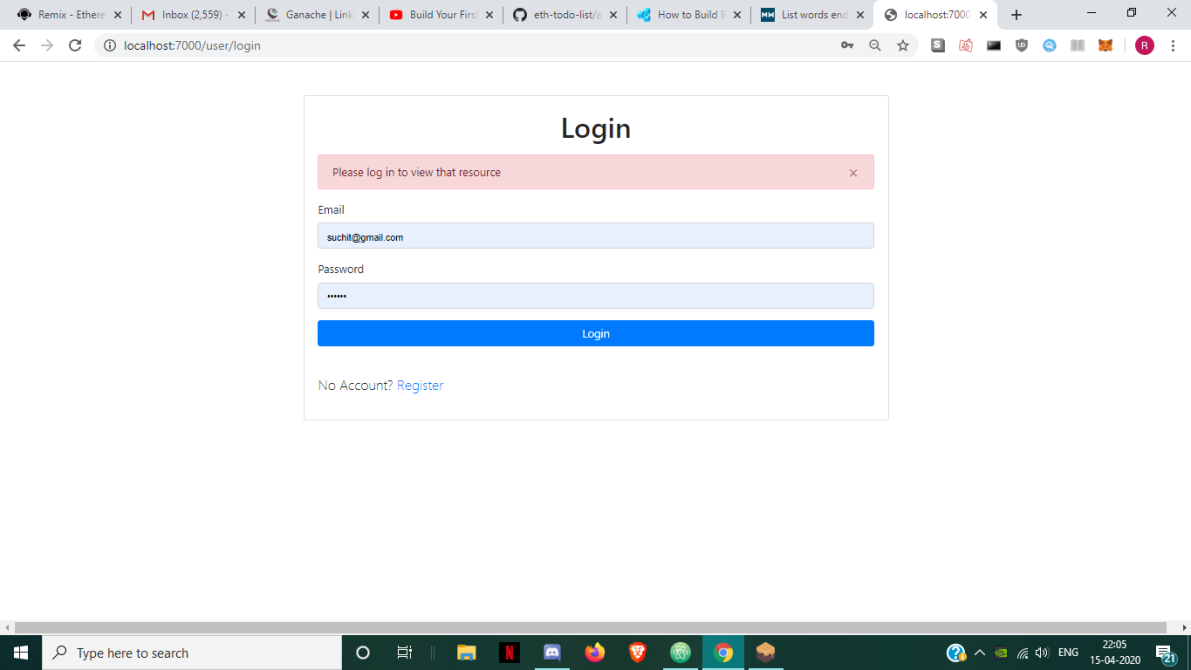
In this web application there are certain requirements which need to be dealt with. Requirements like its users/customers account, credentials etc. For that web application has to have a “Login” page that has these elements:

* Account/Username
* Password
* Login/Sign in Button

**Unit testing of above-mentioned case can be as follows:**

* Field length – username and password fields
  + In here the module is concerned with the field length of certain attributes like password which should be according to mentioned description (special characters).
* Input field
  + Value of input field should be valid like in case of user name should be appropriate.
* Login button will forward user to the home page only if the credentials match.

If not, then a pop up message of “invalid credentials” will appear.



*Figure 5.3.1: Login Failure*

* Register Failure
  + User needs to type in a email which has not been entered before

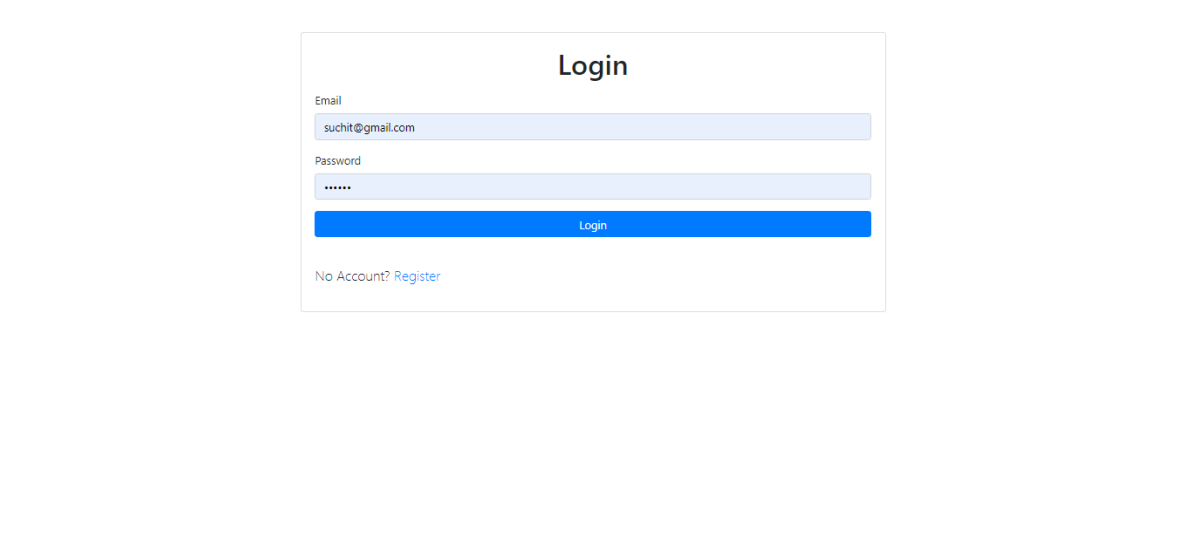
User should be navigated to the welcome page or home page after valid password is typed in.



*Figure 5.3.2: Register Failure*

**Integration testing, of particular case can be as follows:**

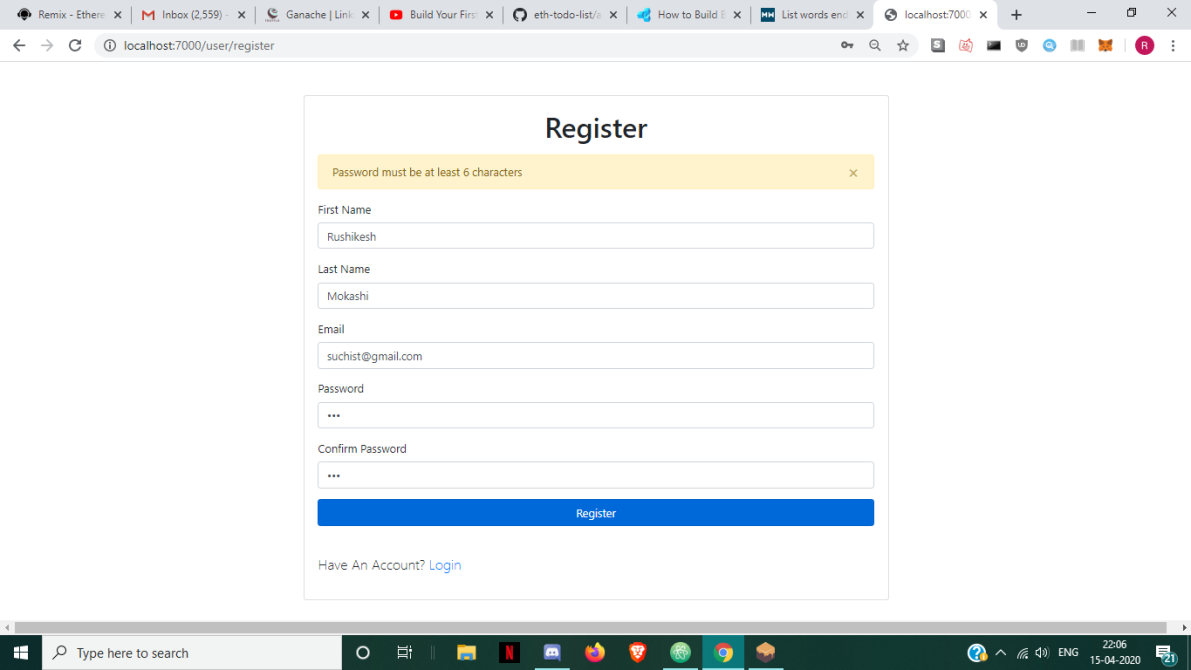
* Form Login
  + User sees the welcome message after entering valid values and pushing the login button.
  + User should be navigated to the welcome page or home page after valid entry and clicking Login button.



*Figure 5.3.4: Successful Login*

* Password not Accepted
  + User needs to type in a password of at least 6 characters to get authenticated

User should be navigated to the welcome page or home page after valid password is typed in.



*Figure 5.3.5: Password not accepted 1*

* Password not Accepted
  + User needs to type in a password of at least 6 characters to get authenticated

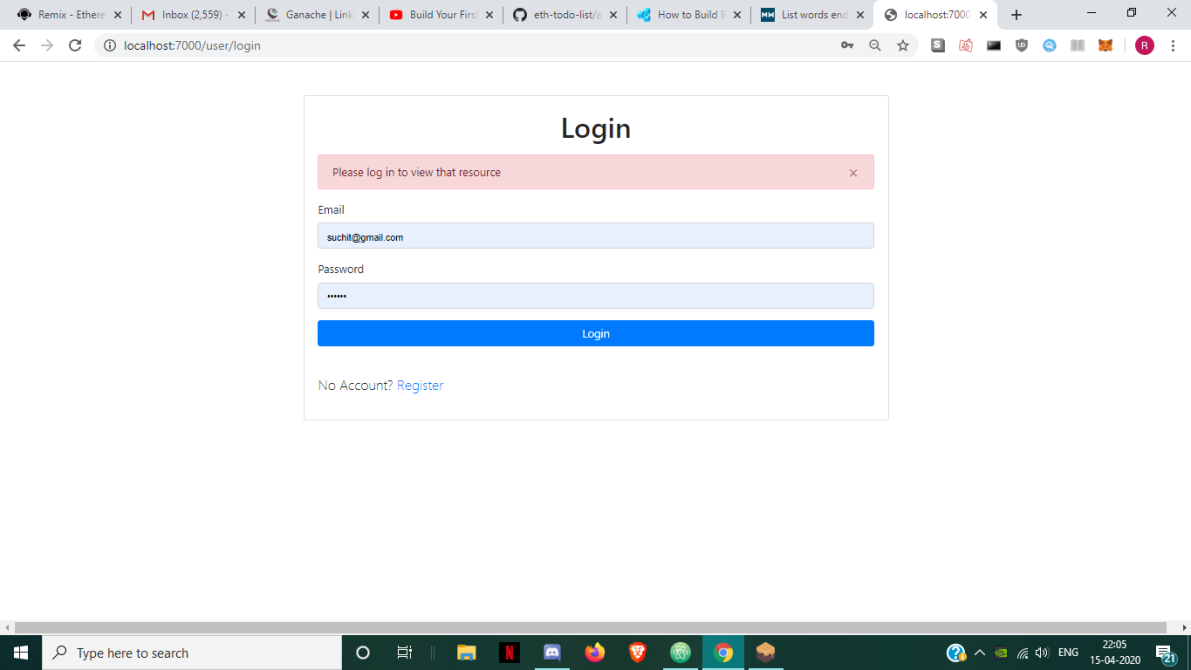
User should be navigated to the welcome page or home page after valid password is typed in.



*Figure 5.3.6: Password not accepted 2*

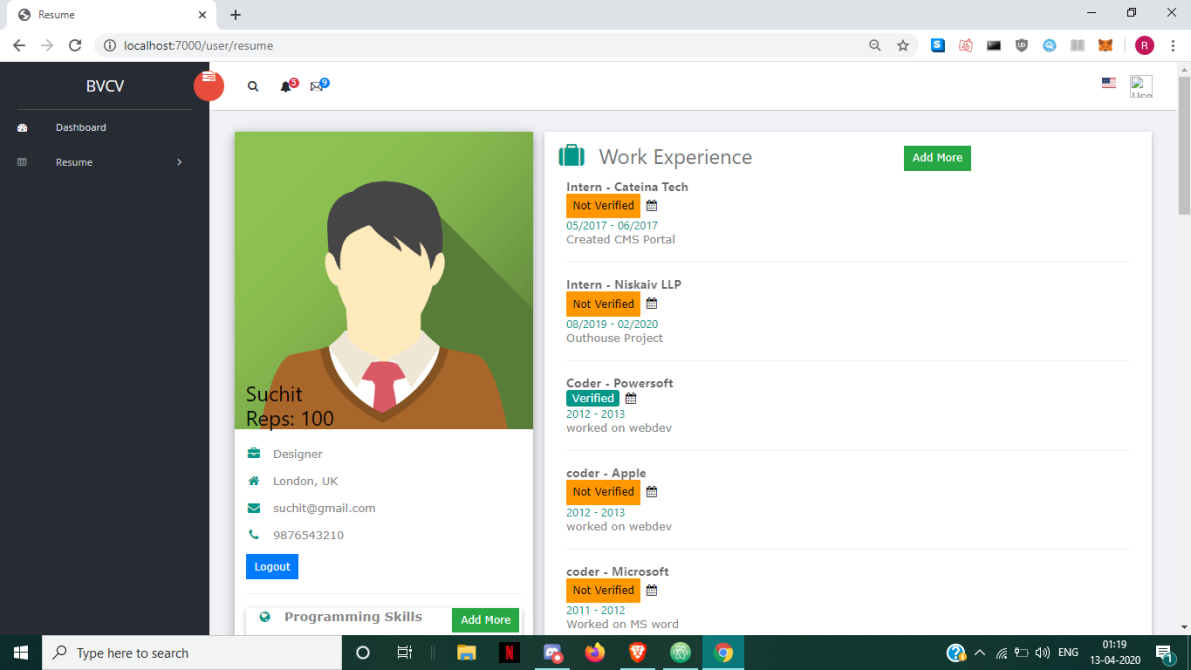
**In this particular case considered for functional testing, description can be as follows:**

1. The expected behaviour is checked, i.e. the user will be able to login on clicking the login button after entering valid username and password values
2. After successful login user is navigated to home page.
3. There should be an error message that should appear on invalid login.
4. To check whether there is any stored site cookies for login fields.
5. Confirmation of registration is send to respective student.



*Figure 5.3.7: LogIn Failure*

1. Successful Login occurs after the user enters everything properly and is properly authenticated.



*Figure 5.3.8: Successful LogIn*

**5.4 Sample Source Code**

* **HELP**

const express = require('express');

const expressLayouts = require('express-ejs-layouts');

const mongoose = require("mongoose");

const flash = require('connect-flash');

const session = require('express-session');

const passport = require('passport');

const app = express();

const db = require('./config/keys').MongoURI;

const ejsLint = require('ejs-lint');

app.use(express.static('public'));

// Passport Config

require('./config/passport')(passport);

//connect to mongodb

mongoose.connect(db, { useNewUrlParser: true,useUnifiedTopology: true})

.then(() => console.log("mdb connected"))

.catch(err => console.log(err));

//EJS

app.set('view engine', 'ejs');

//Bodyparser

app.use(express.urlencoded({extended: false}));

// Express session

app.use(

session({

secret: 'secret',

resave: true,

saveUninitialized: true

})

);

// Passport middleware

app.use(passport.initialize());

app.use(passport.session());

//Connect flash

app.use(flash());

// Global variables

app.use(function(req, res, next) {

res.locals.success\_msg = req.flash('success\_msg');

res.locals.error\_msg = req.flash('error\_msg');

res.locals.error = req.flash('error');

next();

});

app.use('/', require('./routes/index.js'));

app.use('/user', require('./routes/user.js'));

app.use('/forms', require('./routes/forms.js'));

app.use('/employer', require('./routes/employer.js'));

const PORT = process.env.PORT || 7000;

app.listen(PORT, console.log(Server started on port ${PORT}));

* **Users.js**

const express = require('express');

const session = require('express-session');

const router = express.Router();

const bcrypt = require('bcryptjs');

const passport = require('passport');

const User = require('../models/User');

const axios = require('axios');

const ejsLint = require('ejs-lint');

var ssn;

// Login Page

router.get('/login', (req, res)=>{

ssn = req.session;

if(ssn.user){

res.redirect('/user/resume');

} else {

res.render('login');

}

});

router.get('/resume', (req, res) => {

ssn = req.session;

if(ssn.user){

res.render('resumepage', {

user: ssn.user

});

} else {

req.flash('error\_msg', 'Please log in to view that resource');

res.redirect('/user/login');

}

});

// Register Page

router.get('/register', (req, res)=>{

ssn = req.session;

if(ssn.user){

res.redirect('/user/resume');

} else {

res.render('register');

}

});

router.post('/register', (req, resx) => {

const {

fname,

lname,

email,

password,

password2

} = req.body;

let errors = [];

//check req fields

if (!fname ||!lname || !email || !password || !password2) {

errors.push({

msg: 'Please fill in all the fields'

});

}

//Check passwords match

if (password !== password2) {

errors.push({

msg: 'Passwords dont match'

});

}

//Password length

if (password.length < 6) {

errors.push({

msg: 'Password must be at least 6 characters'

});

}

if (errors.length > 0) {

resx.render('register', {

errors,

fname,

lname,

email,

password,

password2

})

} else {

//Validation passed

axios.post('http://192.168.137.1:3000/employee/signup', {

"fname": fname,

"lname": lname,

"email": email,

"password": password

})

.then((res) => {

console.log(res.data);

if(res.data.message == 'you have successfully registered'){

req.flash('success\_msg', res.data.message);

resx.redirect('/user/login');

} else {

req.flash('error\_msg', res.data.message);

resx.redirect('/user/register');

}

})

.catch((error) => {

console.error(error)

})

}

});

router.post('/login', (req,resx)=>{

axios.post('http://192.168.137.1:3000/employee/login', {

"email": req.body.email,

"password": req.body.password

})

.then((res) => {

ssn = req.session;

ssn.user = res.data[0];

resx.redirect('/user/resume');

})

.catch((error) => {

console.error(error)

})

})

// Logout

router.get('/logout',function(req,res){

req.session.destroy(function(err) {

if(err) {

console.log(err);

} else {

res.redirect('/user/login');

}

});

});

module.exports = router;

* **Forms.js**

const express = require('express');

const router = express.Router();

const User = require('../models/User');

const axios = require('axios');

router.post('/verif', (req,res)=>{

axios.post('http://192.168.137.1:3000/employee/verifreq', req.body)

.then((res) => {

res.render('resumepage');

})

.catch((error) => {

console.error(error)

})

})

router.post('/changeverif', (req,resx)=>{

axios.post('http://192.168.137.1:3000/employee/changeverif', req.body)

.then((res) => {

axios.post('http://192.168.137.1:3000/employer/verifcomplete', req.body)

.then((res) => {

resx.redirect('/employer/base');

})

.catch((error) => {

console.error(error)

})

})

.catch((error) => {

console.error(error)

})

})

router.post('/submit', (req, resx) => {

var flag;

Object.keys(req.body).forEach(function(key) {

if(key == 'workexsubmit'){

req.body["verif"] = 0;

flag = "w";

}

if(key == 'edusubmit'){

req.body["verif"] = 0;

flag = "e";

}

if(key == 'leadrosubmit'){

flag = "l";

}

if(key == 'certifsubmit'){

flag = "c";

}

if(key == 'projsubmit'){

flag = "p";

}

if(key == 'adddetsubmit'){

flag = "a";

}

if(key == 'skillsubmit'){

flag = "s";

}

if (req.body[key] == "Submit")

delete req.body[key];

});

if(flag == "w"){

if(Array.isArray(req.session.user.resume.workexp)){

req.session.user.resume.workexp.push(JSON.parse(JSON.stringify(req.body)));

} else {

req.session.user.resume.workexp = [];

req.session.user.resume.workexp.push(JSON.parse(JSON.stringify(req.body)));

}

}

if(flag == "s"){

if(Array.isArray(req.session.user.resume.skill)){

req.session.user.resume.skill.push(JSON.parse(JSON.stringify(req.body)));

} else {

req.session.user.resume.skill = [];

req.session.user.resume.skill.push(JSON.parse(JSON.stringify(req.body)));

}

}

if(flag == "e"){

if(Array.isArray(req.session.user.resume.education)){

req.session.user.resume.education.push(JSON.parse(JSON.stringify(req.body)));

} else {

req.session.user.resume.education = [];

req.session.user.resume.education.push(JSON.parse(JSON.stringify(req.body)));

}

}

if(flag == "l"){

if(Array.isArray(req.session.user.resume.responsib)){

req.session.user.resume.responsib.push(JSON.parse(JSON.stringify(req.body)));

} else {

req.session.user.resume.responsib = [];

req.session.user.resume.responsib.push(JSON.parse(JSON.stringify(req.body)));

}

}

if(flag == "c"){

if(Array.isArray(req.session.user.resume.certification)){

req.session.user.resume.certification.push(JSON.parse(JSON.stringify(req.body)));

} else {

req.session.user.resume.certification = [];

req.session.user.resume.certification.push(JSON.parse(JSON.stringify(req.body)));

}

}

if(flag == "p"){

if(Array.isArray(req.session.user.resume.project)){

req.session.user.resume.project.push(JSON.parse(JSON.stringify(req.body)));

} else {

req.session.user.resume.project = [];

req.session.user.resume.project.push(JSON.parse(JSON.stringify(req.body)));

}

}

if(flag == "a"){

if(Array.isArray(req.session.user.resume.adddet)){

req.session.user.resume.adddet.push(JSON.parse(JSON.stringify(req.body)));

} else {

req.session.user.resume.adddet = [];

req.session.user.resume.adddet.push(JSON.parse(JSON.stringify(req.body)));

}

}

axios.post('http://192.168.137.1:3000/employee/updateres', req.session.user)

.then((res) => {

console.log(res.body);

})

.catch((error) => {

console.error(error)

})

resx.redirect('/user/resume');

})

module.exports = router;

* **Index.js**

const express = require('express');

const router = express.Router();

const User = require('../models/User');

const { ensureAuthenticated } = require('../config/auth');

router.get('/', (req,res) => res.render('index'));

//Dashboard

router.get('/dashboard', ensureAuthenticated, (req,res) =>

res.render('dashboard',{

user: req.user.name

}));

module.exports = router;

* **Employer.js**

const express = require('express');

const session = require('express-session');

const router = express.Router();

const bcrypt = require('bcryptjs');

const passport = require('passport');

const User = require('../models/User');

const axios = require('axios');

const ejsLint = require('ejs-lint');

var ssn;

router.get('/login', (req, res)=>{

ssn = req.session;

if(ssn.user){

res.redirect('/employer/base');

} else {

res.render('employerlogin');

}

});

router.get('/base', (req, res) => {

if(ssn.user){

console.log("1");

axios.post('http://192.168.137.1:3000/employer/verifreq', {

"email": ssn.user.email

})

.then((resy) => {

ssn.profile = resy.data;

console.log(ssn.profile);

console.log(ssn.user);

res.render('employerpage', {

user: ssn.user,

profile: ssn.profile

});

})

.catch((errorx) => {

console.error(errorx)

})

} else {

req.flash('error\_msg', 'Please log in to view that resource');

res.redirect('/employer/login');

}

});

router.post('/login', (req,resx)=>{

axios.post('http://192.168.137.1:3000/employer/login', {

"email": req.body.email,

"password": req.body.password

})

.then((res) => {

ssn = req.session;

ssn.user = res.data[0];

resx.redirect('/employer/base');

})

.catch((error) => {

console.error(error)

})

})

// Logout

router.get('/logout',function(req,res){

req.session.destroy(function(err) {

if(err) {

console.log(err);

} else {

res.redirect('/employer/login');

}

});

});

module.exports = router;

* **HELP**

const express = require('express');

const bodyParser = require('body-parser');

const mongoose = require('mongoose');

require('dotenv/config');

const cors = require('cors');

const app = express();

app.use(bodyParser.json());

app.use(cors());

const employeeRoute = require('./routes/employee.js');

const employerRoute = require('./routes/employer.js');

app.use('/employee',employeeRoute);

app.use('/employer',employerRoute);

app.get('/',(req,res)=>{

res.send('Welcome to homepage');

})

mongoose.connect(process.env.DB\_CONNECTION,

{ useNewUrlParser: true },

() => console.log('connected to db'));

app.listen(3000);

* **Employee.js**

const express = require('express');

const router = express.Router();

const Employee = require('../models/employee');

const passwordhash = require('password-hash');

const Resume = require('../models/testmodel');

const VerificationRequest = require('../models/verif');

router.post('/test', async (req, res) => {

try {

console.log(req.body.email);

var empname = req.body.email;

const post = await Employee.find({

email: empname

});

var wk = post[0]['resume']['workexp'];

console.log(wk.length);

res.json(post);

} catch (err) {

res.json({

message: err

});

}

})

router.post('/resume/', async (req, res) => {

try {

var empname = req.body.email;

var emp = await Employee.find({

email: empname

});

var reqs = await VerificationRequest.find({

verifemail: req.body.verifemail,

sendemail: req.body.email

});

emp = emp[0];

reqs = reqs[0];

res.json({

"resume": emp.resume,

"req": reqs

});

} catch (err) {

res.json({

message: err

});

}

})

router.post('/updateres', async (req, res) => {

try {

// const employee = await Employee.find({ email: req.body.email});

// console.log(post);

const employee = await Employee.updateOne({

email: req.body.email

}, {

$set: {

resume: req.body.resume

}

});

res.json(employee);

} catch (err) {

res.json({

message: err

});

}

})

router.post('/verifreq', async (req, res) => {

console.log(req.body.sendemail);

const verif = new VerificationRequest({

sendemail: req.body.sendemail,

verifemail: req.body.verifemail,

message: req.body.message,

orgname: req.body.orgname,

designation: req.body.designation,

startd: req.body.startd,

endd: req.body.endd,

description: req.body.description

});

verif.save()

.then(data => {

res.json(data);

})

.catch(err => {

res.json({

message: err

});

})

})

router.post('/eres', (req, res) => {

console.log(req);

const resume = new Resume({

username: req.body.username,

school: req.body.school,

jr\_college: req.body.jr\_college,

undergrad\_college: req.body.undergrad\_college,

job1: req.body.job1,

job2: req.body.job2,

job3: req.body.job3

});

console.log(resume);

resume.save()

.then(data => {

res.json(data);

})

.catch(err => {

res.json({

message: err

});

})

})

// router.get('/all', async (req,res)=>{

// try{

// const allposts = await Employee.find();

// res.json(allposts);

// }catch(err)

// {

// res.json({message:err});

// }

// })

// //NEW EMPLOYEE

// router.post('/signup',(req,res)=>{

// console.log(req.body);

// const user = new User({

// name: req.body.name,

// email: req.body.email

// });

// user.save()

// .then(data =>{

// res.json(data);

// })

// .catch(err=>{

// res.json({message:err});

// })

// })

//---------------------------------------------------------------------------------------------------------------

// //ALL EMPLOYEES

// router.get('/all', async (req,res)=>{

// try{

// const allposts = await User.find();

// res.json(allposts);

// }catch(err)

// {

// res.json({message:err});

// }

// })

router.get('/disp', async (req, res) => {

try {

const emp = await Employee.find();

console.log(emp);

res.json(emp);

} catch (err) {

res.json({

message: err

});

}

})

//NEW EMPLOYEE

router.post('/signup', (req, res) => {

console.log(req.body);

Employee.findOne({

email: req.body.email

})

.then(user => {

if (user) {

res.json({

message: "The email has already been registered"

})

} else {

const user = new Employee({

fname: req.body.fname,

lname: req.body.lname,

email: req.body.email,

password: passwordhash.generate(req.body.password),

rep: 100

});

user.save()

.then(data => {

res.json({

message: "you have successfully registered"

});

})

.catch(err => {

res.json({

message: err

});

})

}

})

})

//FIND EMPLOYEE

router.get('/find/:empName', async (req, res) => {

try {

const post = await Employee.findById(req.params.empName);

res.json(post);

} catch (err) {

res.json({

message: err

});

}

})

router.post('/findemp/', async (req, res) => {

try {

console.log(req.body.email);

var empname = req.body.email;

const post = await Employee.find({

email: empname

});

res.json(post);

} catch (err) {

res.json({

message: err

});

}

})

router.post('/login', async (req, res) => {

try {

console.log(req.body);

const user = await Employee.find({

email: req.body.email

});

if (!user) {

console.log("User not found");

res.json({

message: "User not found"

});

} else if (passwordhash.verify(req.body.password, user[0].password)) {

console.log(user);

res.json(user);

} else {

console.log("Password Incorrect");

res.json({

message: "Password Incorrect"

});

}

} catch (err) {

res.json({

message: err

});

}

})

//CONTACT FORM

router.post('/contact', async (req, res) => {

const contact = new Contact({

cname: req.body.cname,

cemail: req.body.cemail,

subject: req.body.subject,

message: req.body.message

})

contact.save()

.then(data => {

res.json(data);

})

.catch(err => {

res.json({

message: err

});

})

})

router.post('/update', async (req, res) => {

console.log(req.body.username);

const filter = {

username: req.body.username

};

const update = {

rep: 110

};

// const userdash = await User.find({username: req.body.username});

// console.log("userdash:"+userdash);

const user = await Employee.updateOne({

username: req.body.username

}, {

$set: {

rep: 1500

}

});

console.log(user);

// let doc = await User.findOneAndUpdate(filter, update);

// console.log(doc);

res.json(user);

})

router.post('/reps', async (req, res) => {

var emp1 = req.body.user1;

var emp2 = req.body.user2;

console.log(req.body);

console.log(req);

const temp = await Employee.find({

username: emp1

});

rep1 = temp[0].rep;

const temp2 = await Employee.find({

username: emp2

});

rep2 = temp2[0].rep;

const post = {

rep1: rep1,

rep2: rep2

};

console.log(post);

res.json(post);

})

router.post('/Repsdone', async (req, res) => {

const user1 = await Employee.updateOne({

username: req.body.user1

}, {

$set: {

rep: req.body.rep1

}

});

const user2 = await Employee.updateOne({

username: req.body.user2

}, {

$set: {

rep: req.body.rep2

}

});

res.json("Thanks for the rep!!");

})

// router.post('/verifreq',async(req,res)=>{

// const vreq = new VerificationRequest({

// username: req.body.username,

// verifier: req.body.verifier,

// })

// vreq.save()

// .then(data =>{

// res.json(data);

// })

// .catch(err=>{

// res.json({message:err});

// })

// })

router.post('/changeverif', (req, res) => {

Employee.findOne({

email: req.body.email

}, function(err, user) {

if(err){

console.log(err);

}else if (user) {

user = user.toJSON();

Object.keys(user.resume.workexp).forEach((item, i) => {

if (req.body.orgname == user.resume.workexp[item].orgname &&

req.body.designation == user.resume.workexp[item].designation &&

req.body.description == user.resume.workexp[item].description &&

req.body.startd == user.resume.workexp[item].startd &&

req.body.endd == user.resume.workexp[item].endd) {

user.resume.workexp[item].verif = 1;

}

});

console.log(user.resume.workexp);

var query = {email: req.body.email};

Employee.update(query, user)

.then(res.send("Doneerr"))

.catch(err => console.log(err))

}

})

});

module.exports = router;

**Employer.js**

const express = require('express');

const router = express.Router();

const Employer = require('../models/employer');

const Employee = require('../models/employee');

const passwordhash = require('password-hash');

const VerificationRequest = require('../models/verif');

router.get('/disp',async (req,res)=>{

try{

const emp = await Employer.find();

console.log(emp);

res.json(emp);

}catch(err)

{

res.json({message:err});

}

})

router.post('/login',async(req,res)=>{

try{

console.log(req.body);

const user = await Employer.find({email: req.body.email});

if(!user){

res.json({message:"User not found"});

}

else if(passwordhash.verify(req.body.password,user[0].password))

{

res.json(user);

}

else

{

res.json({message:"Password Incorrect"});

}

}catch(err){

res.json({message:err});

}

})

router.post('/signup',(req,res)=>{

console.log(req);

const user = new Employer({

fname:req.body.fname,

lname:req.body.lname,

email: req.body.email,

password: passwordhash.generate(req.body.password),

rep: 100

});

user.save()

.then(data =>{

res.json(data);

})

.catch(err=>{

res.json({message:err});

})

})

router.post('/verifreq',async (req,res)=>{

try{

const reqs = await VerificationRequest.find({verifemail:req.body.email});

console.log(reqs);

res.json(reqs);

}catch(err)

{

res.json({message:err});

}

})

router.post('/verifcomplete',async (req,res)=>{

try{

console.log(req.body);

var reqs = await VerificationRequest.deleteOne({verifemail:req.body.verifemail,sendemail:req.body.sendemail});

console.log(reqs);

res.json(reqs);

}catch(err)

{

res.json({message:err});

}

})

module.exports = router;

* **MODEL – employee**

const mongoose = require('mongoose');

const employeeSchema = mongoose.Schema({

fname:{

type: String,

},

lname:{

type: String,

},

email: {

type: String,

required: true

},

password: {

type: String,

required: true

},

rep: {

type: "number"

},

resume: {

phoneno: {

type: "number"

},

education: {

institution\_name: String,

startd: String,

endd: String,

grade: {

type: "number"

},

degree: String,

studid:{

type:"number"

},

verif:{

type:String,

default:null

}

},

workexp: {

orgname: String,

designation: String,

startd: String,

endd: String,

description: String,

verif:{

type:"number",

default:0

}

},

responsib: {

title: String,

description: String

},

certification: {

program: String,

orgname: String,

startd: String,

endd: String,

description: String

},

project: {

title: String,

startd: String,

endd: String,

description: String,

link: String

},

skill: {

skill: String,

rating: String,

},

adddet: {

title: String,

description: String

}

}

})

module.exports = mongoose.model('Employee',employeeSchema);

* **MODEL – employer**

const mongoose = require('mongoose');

const employerSchema = mongoose.Schema({

fname:{

type: String,

required:true

},

lname:{

type: String,

required: true

},

email: {

type: String,

required: true

},

password: {

type: String,

required: true

},

rep: {

type: "number"

},

phoneno:{

type: "number"

},

organization:{

type:String

}

})

module.exports = mongoose.model('Employer',employerSchema);

* **MODEL – verification**

const mongoose = require('mongoose');

const verifschema = mongoose.Schema({

sendemail: String,

verifemail: String,

message: String,

orgname: String,

designation: String,

startd: String,

endd: String,

description:String

})

module.exports = mongoose.model('VerificationRequest',verifschema);

* **Web 3 Code – This code converts the blockchain smart contracts into Java Scripts objects to be used in Front End**

App = {

loading: false,

contracts: {},

load: async () => {

await App.loadWeb3()

await App.loadAccount()

await App.loadContract()

await App.render()

},

// https://medium.com/metamask/https-medium-com-metamask-breaking-change-injecting-web3-7722797916a8

loadWeb3: async () => {

if (typeof web3 !== 'undefined') {

App.web3Provider = web3.currentProvider

web3 = new Web3(web3.currentProvider)

} else {

window.alert("Please connect to Metamask.")

}

// Modern dapp browsers...

if (window.ethereum) {

window.web3 = new Web3(ethereum)

try {

// Request account access if needed

await ethereum.enable()

// Acccounts now exposed

web3.eth.sendTransaction({/\* ... \*/})

} catch (error) {

// User denied account access...

}

}

// Legacy dapp browsers...

else if (window.web3) {

App.web3Provider = web3.currentProvider

window.web3 = new Web3(web3.currentProvider)

// Acccounts always exposed

web3.eth.sendTransaction({/\* ... \*/})

}

// Non-dapp browsers...

else {

console.log('Non-Ethereum browser detected. You should consider trying MetaMask!')

}

},

loadAccount: async () => {

// Set the current blockchain account

console.log(web3);

App.account = web3.eth.accounts[0]

},

loadContract: async () => {

// Create a JavaScript version of the smart contract

const userList = await $.getJSON('User.json')

App.contracts.User = TruffleContract(userList)

App.contracts.User.setProvider(App.web3Provider)

// Hydrate the smart contract with values from the blockchain

App.userList = await App.contracts.User.deployed()

console.log('contract loaded');

},

render: async () => {

// Prevent double render

if (App.loading) {

return

}

// Update app loading state

App.setLoading(true)

// Render Account

$('#account').html(App.account)

// Render Tasks

await App.renderTasks()

// Update loading state

App.setLoading(false)

},

renderTasks: async () => {

// Load the total task count from the blockchain

const userCount = await App.userList.userCount()

const $taskTemplate = $('.taskTemplate')

// var accs = web3.eth.getAccounts();

// console.log(accs);

// web3.eth.getAccounts((err,acc)=>{

// if(err){

// console.log('error.............',err);

// }

// console.log(acc);

// });

// Render out each task with a new task template

for (var i = 1; i <= userCount; i++) {

// Fetch the task data from the blockchain

const emp = await App.userList.users('Suchit')

const taskId = emp[0].toNumber()

const taskContent = emp[1]

const taskCompleted = emp[2].toNumber()

// Create the html for the taskgg

const $newTaskTemplate = $taskTemplate.clone()

$newTaskTemplate.find('.content').html(taskContent)

$newTaskTemplate.find('.reps').html(taskCompleted)

$newTaskTemplate.find('input')

.prop('name', taskId)

// Put the task in the correct list

$('#taskList').append($newTaskTemplate)

// Show the task

$newTaskTemplate.show()

}

},

createUser: async () => {

App.setLoading(true)

const name = $('#newemp').val()

const rep = $('#empreps').val()

await App.userList.createUser(name,rep)

window.location.reload()

},

updateReps: async () => {

App.setLoading(true)

const name = $('#empname').val()

const rep = $('#newreps').val()

await App.userList.updateReps(name,rep)

window.location.reload()

},

// toggleCompleted: async (e) => {

// App.setLoading(true)

// const taskId = e.target.name

// await App.todoList.toggleCompleted(taskId)

// window.location.reload()

// },

setLoading: (boolean) => {

App.loading = boolean

const loader = $('#loader')

const content = $('#content')

if (boolean) {

loader.show()

content.hide()

} else {

loader.hide()

content.show()

}

}

}

$(() => {

$(window).load(() => {

App.load()

})

})

* **BlockChain Smart Contracts**

pragma solidity ^0.5.0;

contract User{

uint public userCount = 0;

struct User{

uint id;

string name;

uint reps;

}

mapping(string => User) public users;

event userCreated(

uint id,

string name,

uint reps

);

event repUpdated(

string name,

uint reps

);

constructor() public {

createUser("Suchit",10000000);

}

function createUser(string memory \_name, uint \_reps) public {

userCount++;

users[\_name] = User(userCount, \_name, \_reps);

emit userCreated(userCount, \_name, \_reps);

}

function updateReps(string memory \_name, uint \_reps) public {

User storage \_emp = users[\_name];

\_emp.reps = \_reps;

users[\_name] = \_emp;

emit repUpdated(\_name, \_reps);

}

}

**Chapter 5**

**Conclusions**

Thus, this project will be implemented using Blockchain technology and the usage of smart contracts.

A system of reps and negs will be implemented as a system of points in the website to incentivize website activity and enable a profile scoring system.

Every transaction will cause the exchange of reps and negs to be updated in the persons resume

A verified resume will be created and reputation will be used to grade each resume.

This will surely help in reducing the time required for cross checking each and every component on the resume.

It would not only accelerate the verifying process but would also accelerate recruitment process and decrease the recruitment time.

The future scope with respect to this project is copious. A partnership to club this with pre existing Job Portals like Naukri and LinkedIn would be beneficial to both the parties. With their database of millions of employees and employers, it would be much efficient and necessary to implement such a system.

Partnering with all major universities and all certification providers is also a future scope. This would help the verification of not only Job Experiences but also Educational Qualifications.

**References**

The listing of references should be typed 2 spaces below the heading “REFERENCES” in order of its appearance in thesis in single spacing left – justified with Times New Roman font type. The name of the author/authors should be immediately followed by the year and other details. References should be cited in the text at appropriate places as [n] in serial order as they appear in the text. Every reference in the list must be cited at least once in the report.

A typical illustrative list given below relates to the citation example quoted above.

***For a paper in a Proceedings/Transactions/Journal***

1. “Methods and systems for employment and education verification using blockchain”- [Jaspreet Randhawa](https://patents.google.com/?inventor=Jaspreet+Randhawa)
2. “FEEDBACK BASED REPUTATION ON TOP OF THE BITCOIN BLOCKCHAIN”- Davide Carbon
3. Decentralizing Privacy: Using Blockchain to Protect Personal Data (2015) -Guy Zyskind ; Oz Nathan ; Alex 'Sandy' Pentland

***For a report in a conference record***

[1] “Computerized method and system for creating a digital resume”- [Daniel Jacobus deWaal](https://patents.google.com/?inventor=Daniel+Jacobus+deWaal).

[2] “A system and method for creating a database of independently validated educational and work experience profiles (curricula vitae) using blockchain smart contract”- Gary McKay Adi BEN-ARIBrian Mcnulty

***For an authored book***

[1] Etherium Smart Contract Development- Mayukh Mukhopadhyay

[2] Introducing Etherium and Solidity

***For a technical report***

[1] “A Study on Privilege Elevation Attack Management for Smart Transaction Security on Blockchain Etherium Based System”- Min, Youn-A

**Appendix A**

**Abbreviation and symbols**

1. UML: Unified Modelling Language

2. RAD: Requirement Analysis Document

3. FAD: Feasibility Analysis Document

4. SQL: Structured Query Language

5. XML: Extensible Mark-up Language

6. GUI: Graphical User Interface

**Appendix B**

**Definitions**

1. Classification: It is the process of identifying types/subproblems/classes of the problem, segregating the current instances into these types and deciding to which class a new observation belongs.
2. Blockchain Verification: In IT, blockchain verification is a branch of machine learning that emphasizes the recognition of data patterns or data regularities in a given scenario.
3. Repositories: In computer science, a central location in which data is stored and managed.
4. Parser: A parser is a compiler or interpreter component that breaks data into smaller elements for easy translation into another language.
5. Utilities: A utility or software utility is computer system software intended to analyse, configure, monitor, or help maintain a computer.

**Blockchain Based Resume Verification System**

Rushikesh Mokashi Suchit Gupta Srihari Nair

rushimok73@gmail.com suchitgoyal000@gmail.com sriharinair1998@gmail.com

Harshali Patil

harshali.patil@thakureducation.org

Department of Computer Engineering, Thakur College of Engineering and Technology

I. Abstract

Blockchain Based Resume Verification System is a system which will have the primary goal of verifying and confirming the educational and professional qualifications of every individual. This system will be implemented on Ethereum as our blockchain, WordPress for our CMS, mongo DB as our database and NodeJS, php for server scripting. Also, a scoring system will be added on our website, based on a system of reps and negs discussed later. [2]

Previous applications which have implemented something similar to this software is LinkedIn. But an area which it lacked in was the verification of the qualifications. [1] We believe that with our system we could overcome this hurdle and fulfil our main objective. [1]

Our main objective being - to combat people lying on their resume about their qualifications. [1] We aim to create a trustworthy system and thus effectively end this problem.

**Keywords: Blockchain, Ethereum, Resume, Verification, WordPress, NodeJS, Reps, Negs, Score, mongo DB**

II. Introduction

For a long time, people have been lying on their resumes to get better opportunities and they have largely been able to get away with it because currently, there is no way to verify the claims made on the resume. This is a big problem for corporations and educational institutes for hiring’s and admissions, and we aim to solve this problem. Also, there presently exists no resume scoring system, as it is hard to put a value on a largely subjective approach to analysing resumes. The system of reps and negs [2] we aim to implement will solve this problem, and also make the task of Hiring departments in many organizations much easier.

A. Blockchain

Blockchain is a database that is distributed and maintains records of a continuously-growing list secured from tampering and revision. Each block is connected to the previous block via a link which has a timestamp attached on it. [5] The main technical innovation of bitcoin - which was conceived in 2008 and first implemented in 2009 - serves as the public ledger for bitcoin transactions. New blocks can be created by every user, they are also allowed to send new transactions and attempt to verify those transactions, and create new blocks. The inspiration for other applications has been the bitcoin blockchain.

B. Ethereum

Ethereum is used for decentralized applications or dApps and is basically blockchain-based platform.

An Ethereum Virtual Machine or EVM is a decentralized virtual machine provided by Ethereum, which can execute scripts using an international network of public nodes. [6] In contrast to the Bitcoin Script, the virtual machine's instruction set, is thought to be Turing-complete. An internal transaction pricing mechanism, known as “Gas”, is used to mitigate spam and allocate resources on the network.

C. WordPress

WordPress based on PHP and MySQL is a content management system that is mostly used with the SQLite database engine but can also be used by MySQL or MariaDB database servers. It has a template system and a plugin architecture, known as themes inside WordPress. Blogging is something which is most associated with WordPress but is now often used to support other types of web content as well including more traditional mailing lists and forums & online stores.

D. Mongo DB

Mongo DB is a database program which provides cross-platform document-oriented database, it uses JSON-like documents with schema classified as a NoSQL database program.

E. Smart Contracts

A smart contract is such that it is used to digitally verify and protect and enforce the terms under which the contract had been initially written. These allow the performance of credible transactions without third parties or any external participation.[6] These transactions are trackable and irreversible. Proponents of smart contracts may be made partially or fully self-executing, self-enforcing, or both, a claim that many kinds of contractual clauses make. The main purpose of a smart contracts is to provide a guarantee and protection, that is superior to traditional contract law and to reduce other transaction costs associated with contracting. Most of these smart contracts are used in implementation of various types of smart contracts.

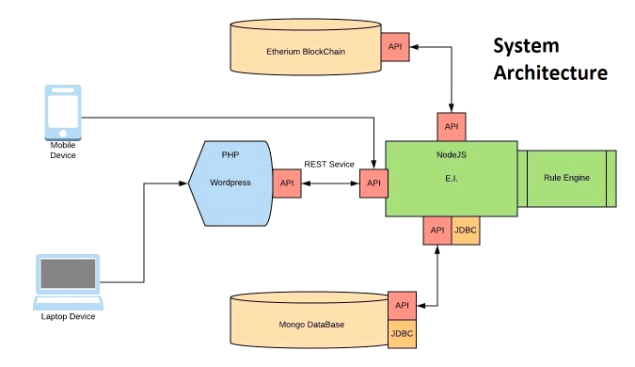
III. Literature Survey

We have surveyed several resources and papers. From these papers we understood several principles and did a gap analysis. We first looked at Methods and systems for employment and education verification using blockchain we understood how to use blockchain to deploy verification system but we felt that it had a lack of explanation about how technology stack may affect the effectiveness. Further, we took a look at multiple papers regarding Computerized method and system for creating a digital resume, [4] A system and method for creating a database of independently validated educational and work experience profiles (curricula vitae) using blockchain smart contracts, feedback based reputation on top of the bitcoin blockchain, A Study on Privilege Elevation Attack Management for Smart Transaction Security on Blockchain Ethereum Based System. A gap analysis was conducted on all of these and taken into consideration for our paper. [5]

IV. Methodology

A. Resume Creation

To create a resume on our website, a user has to first create an account on our website and select his account type which will be an ‘employee’ or an ‘employer’. These will be the two roles on our website. Any claims made by the employee will be verified by the organization or his peers/supervisor. Upon logging in, the user has to complete a resume form, the data from which will be stored into the mongo DB database.



B. Database Storage

The mongo DB database will store login credentials, resume details, user details, score and multiple other data fields. The data from the mongo DB server will be inserted to and extracted from using REST API service. We are intended on using REST API as we are hopeful of further developing a mobile application and expand forward from our website. [3]

C. Server Scripting

We are using NodeJS and PHP for our server-side scripting. The major use of NodeJS is to fetch and post to our API’s from mongo DB. NodeJS will be mainly used for data transmission. [3] Also, PHP is being used as it works very well with WordPress, as WordPress itself is coded in PHP. Thus, PHP allows us to implement a variety of custom features on WordPress, as WordPress doesn’t match all our needs. WordPress has a variety of plugins we can use, but however a lot of additional programming is needed through PHP to make them work to our needs.

D. API Calls:

Every interaction with the database will happen through a REST API call. This is implemented keeping in mind future expansion of the website to other platforms and create a modular approach. Currently a fetch and a storage call will be made to the database. This might increase later as development continues.

E. Score System

Every profile will have a corresponding score associated with it. [1] This score can be changed via reps and negs. A rep is a positive increase in the users score, while a neg is a decrease in the score. We have reps and negs [2] tied to events in our website that shall trigger them. These have been discussed in depth further later.

F. Verification using Blockchain  
Every verified claim on the website and the particular change in score tied to it shall be updated on the blockchain.[7] We will be using Ethereum as our blockchain and will be using Ethereum smart contracts to reflect the changes made to the rep score and verification status of a claim onto the Blockchain. [1]

V. Conclusion

Thus, this project will be implemented using Blockchain technology and the usage of smart contracts. A system of reps and negs [2] will be implemented as a system of points in the website to incentivize website activity and enable a profile scoring system. Every transaction will cause the exchange of reps and negs [2] to be updated in the persons resume. A verified resume will be created and reputation will be used to grade each resume.

VI. References

1. [Jaspreet Randhawa](https://patents.google.com/?inventor=Jaspreet+Randhawa), “*Methods and systems for employment and education verification using blockchain*”, lens.org, Oct 2018
2. Davide Carbon, “*Feedback Based Reputation on Top of the Bitcoin Blockchain*”, arxiv.org, Feb 2015
3. Guy Zyskind, Oz Nathan, Alex 'Sandy' Pentland “*Decentralizing Privacy: Using Blockchain to Protect Personal Data*”, IEEE, May 2015
4. [Daniel Jacobus deWaal](https://patents.google.com/?inventor=Daniel+Jacobus+deWaal), “*Computerized method and system for creating a digital resume*”, patents.google.com, Nov 2012
5. Gary McKay Adi BEN-ARIBrian Mcnulty, “*A system and method for creating a database of independently validated educational and work experience profiles (curricula vitae) using blockchain smart contract*”, patents.google.com, March 2016
6. Mayukh Mukhopadhyay, “*Ethereum Smart Contract Development*”, Feb 2018
7. Chris Dannen, “*Introducing Ethereum and Solidity*”, March 2017
8. Min, Youn ,“*A Study on Privilege Elevation Attack Management for Smart Transaction Security on Blockchain Etherium Based System*”, koreascience.kr, Feb 2020

**ACKNOWLEDGEMENT**

With deep sense of gratitude we would like to thank all the people who have lit our path with their kind guidance. We are very grateful to these intellectuals who did their best to help during our project.

It is our proud privilege to express deep sense of gratitude to our Project Guide Dr. Harshali Patil for her continuous guidance and support throughout our project. It would never have been possible for us to complete this project successfully without his guidance and support.

We remain indebted to our H.O.D Dr. Sheetal Rathi CMPN Department, Dean Academic Dr. R. R. Sedamkar and Principal Dr. B.K. Mishra for their comments and kind permission to complete this project. We would also like to thank them for their timely suggestions and valuable guidance.

And lastly we would like to thank our friends and the people who are directly or indirectly related to our project.