**CHAPTER 1:**

**INTRODUCTION**

1. **INTRODUCTION**

 A freelancer is regarded as a person who is self-employed and is not necessarily committed to an employer on a long-term basis. This type of employment is made possible especially using ICT. The Information Age has affected the workforce in several ways. Workers are made to compete in a global job market. Technology has increased job opportunities and has promoted the globalization of the workforce. Often, workers in low income have a competitive advantage which translates into increased opportunities and higher wages. The work from home concept is fast picking up the trend of the current job market. In Nigeria, most people have turned to freelance in order to support or balance the rising cost of living. With the aid of the internet, these freelancing jobs are increasing by the day and creating a new way to earn a living.

According to [1], the history of freelance dates back to the 1970’s where a freelancer was initially referred to as “Boundaryless Worker” at the Massachusetts Institute of Technology. This name was given because freelancers have no boundaries to get work. Oxford Advanced Learners Dictionary defines freelance as earning money by selling your work to several different organizations rather than being employed by a particular organization. It also defines a freelancer as one who does freelance.

The introduction of the system would make finding freelancers a whole lot easier for clients as many employers look for freelancers to handle their short-term projects. The system is designed in such a way that clients can find a freelancer based on the skill he or she is looking for. These skills may include web design, web development, content writing and so on.

**1.1** **PROBLEM** **STATEMENT**

The existing platforms for freelancers have a lot of shortcomings in satisfying user experiences to a certain level. These are some problems associated with the existing freelancing systems.

1. Difficulty to work as a newly accepted freelancer
2. Low response rate
3. Inability to limit the amount of jobs a freelancer could work on at once.

**1.2 AIM OF THE PROJECT**

The main aim of this project is to develop a web-based freelance system that enables freelancers and clients work together thereby allowing the freelancer earn for performing a work for the client.

**1.3 SPECIFIC** **OBJECTIVES**

 The specific objectives are:

1. to verify freelancers;
2. enable companies to find qualified and verified freelancers;
3. provide youth with alternative employment; and
4. provide a payment platform for freelancers.

**1.4 SIGNIFICANCE** **OF** **THE** **RESEARCH**

The development of this project is of significance to employers, freelancers and the government.

This system is of benefit to employers because it allows the employers to find a verified freelancer to get some tasks done. Employers stand to gain from this system in a lot of ways. By hiring a freelancer, an employer has automatically saved a lot of money that would have been spent by employing another worker. Freelancers are experts in a certain area, so by employing a freelancer, an Employer is assured a perfect and expected result to be delivered.

Freelancers are one of the major beneficiaries of this system and this is because this system allows the freelancer to earn by performing jobs related to his/her expertise.

The Government is also a major beneficiary of this system. By providing a means whereby people can earn, working as freelancers, through this system, increases the economy of the country. It enables people to earn by working as freelancers which in turn reduces the rate of unemployment in the country and also reduces the rate of crime in the country.

**1.5 SCOPE OF THE PROJECT AND STUDY**

This project is limited to freelancers in the areas of web design, web development, android development, graphic design and content writing. Due to the use of apache server and lack of funds, the limitation of this project is also on the payment method of the system which works when the proposed system is hosted on a live server.

**CHAPTER 2**

**2.0 INTRODUCTION**

In the modern world, especially in Africa, the issues of unemployment of graduates of higher educational institution are prevalent. Young people began to think about the place of work long before their graduation or the end of the university. In connection with the changes that have taken place in the country, young people in the twenty-first century have been known to have more opportunities and different ways to integrate into the society, but the same situation has led to the rise of barriers relating to employment. According to [2], among all unemployed, every one third is a youth or a young person with a higher or secondary school education. The expectations of graduates about the labor market are not always justified and this makes them think of other possible means of earning like freelancing. They would sell their work to organizations without necessarily being employed by a particular company. This tends to give freelancers flexible work hours.

Freelance professionals are not permanently employed by any company and therefore are free to take on work on a project-by-project basis or a client-by-client basis. There are some common myths about freelancing which makes most people prefer being freelancers to being permanently employed by a company and these myths include the freedom to work from anywhere, being your own boss, controlling

how they earn, and being able to work on things that interest them. For a long time, freelancing was associated with hopeless unemployed. However, over the past decades, this trend has reached a new level. For most modern large companies, the withdrawal of certain functions for outsourcing helps to reduce production cost.

The internet revolutionized the way job search and business is done. Job search has gone from newspaper advertisement to a more advanced way all with the help of the internet. Today, freelancing has become one of the major terms in the labor market. Anyone with a pc and an internet connection can become a freelancer.

**2.1 THEORETICAL BACKGROUND**

With rise in the use of internet as a medium to helping us in the way we do business, it has also revolutionized the way job search is done. As trends in the digital nomad lifestyle grow in popularity, the number of freelance resources out there has increased as well and so has the technologies. The following tools and technologies are used in this project: HTML, CSS, JavaScript, jQuery, Bootstrap, PHP and MySQL for the database.

[3,4,5,6,7] are one of the several research works related to the proposed system. However, they differ in some way or the other from the proposed system. The differences range from procedures and functionalities. The major difference is the aspect whereby there is no limit as to how many jobs a freelancer can acquire. These research works would be further discussed in the review of related literature.

**2.2 REVIEW OF RELATED LITERATURE**

S. Wangmo et al, in G-task: Proposal Development of a Freelancing Android Based Mobile Application in Bhutan [3] proposed an android application that allows freelancers to apply and get jobs thereby reducing unemployment in Bhutan. The proposed system, G-task freelancing android application, made use of the descriptive approach as its methodology. The technologies used include; Laravel, PHP, XAMPP, and it made use of android studio as its development environment. Similar to the current proposed system, the G-task freelance application provides an opportunity to freelancers to apply and get work posted by clients. In as much as there are similarities, there are also some differences between the current system and the G-task application. Unlike the current proposed system, the G-task freelancing android application is an android application whereby the current proposed system is a web-based system, G-task freelancing android application allows the admin to post jobs, allows users to give feedback on the system, has a news page where news are maintained and makes use of descriptive approach as its methodology.

A. Mwathi and B.S. Kasamani, in Design and Implementation of a collaborative Freelancers’ Sourcing Platform [4] proposed a freelancing system that allows freelancers to apply for jobs posted by the companies and thereby allowing freelancers earn while working form anywhere at all. The proposed system made use of software development lifecycle as its methodology and the technologies used include; Html, CSS, Php, Laravel and MySQL. Similar to the current project research, Clients are allowed to post jobs, freelancers can access and apply for work posted by a client and are later paid after completing a job; and unlike the current project research, freelancers are not allowed to post jobs which they can offer. Freelancers are also allowed to delete his/her account and also allows companies(clients) to set their generate hiring rate. The proposed system in this research work also using PayPal as a mean of payment.

The authors, C. Dissanayake et al in Freelancing System [7], proposed a system that enables the citizens of Sri Lanka access and apply for freelancing work using EER as the methodology and Html, CSS, Angular CLI, PHP, MySQL and node.js as the technologies. Similar to the current project research, the proposed system allows freelancers to access and apply for work posted by client but unlike the current project research, the proposed system in this research work focuses only on Sri Lankans whereby only Sri Lankans can register as both freelancer and client. Citizens of other countries can only register as clients. The interesting functionality of this system is that it allows users to register and get verified with the use of their IP address in order to know who is Sri Lankan and who is not Sri Lankan.

Tushar et al [6], in the research work, ROZGAAR: Android App for Freelancing, proposed a system which was developed using the technologies react native, node.js and mongoDB, and object oriented as a modelling method. The application allows a freelancer get work which is mostly household chores or any other work thereby allowing them earn a monthly income. This system is similar to the current project research in terms of helping clients find verified freelancers, providing a payment method for freelancers to get paid for their services and allowing freelancers to apply for work posted by client. In as much as these similarities exist, there are also differences between them. In terms of verification and identification, the authors of the proposed system made use of bank details. The freelancing application employs escrow as a method of payment. Each work done is onsite and allows the client to notify when the freelancer has reached the location and a freelancer can choose to be part-time. It allows both freelancers and clients to get rated on the platform and also allows them to change the ratings for a nominal charge but the change in the ratings come with a price which is that all the previous rating would be cleared and the user would have a clear chart.

P. Deshmukh et al in the research work [7], Decentralized Freelancing using Ethereum Blockchain, proposed a blockchain-based decentralized freelancing system built on Ethereum blockchain. By this, it means a system that does not need a centralized form of control in the interactions between the freelancer and the client and it also includes making payments in for of cryptocurrency. The focus is placed on a system which allows users to connect and work together through the platform. The tools and technologies used the platform include; node.js, web3.js, metaMask, truffle and webd3.js. This platform makes use of smart contract to build the business logics system. Similarities between the proposed system and the current project research is that it allows freelancers apply for work or place bids on different works posted by various clients and it also ensures that freelancers are paid through a payment platform. Although these research works share some similarities, there are also some differences between them which include the technologies and methodology, implemented smart contracts are built are deployed and tested on the local Ethereum blockchain called Ganache. The proposed system also uses the IPFS which is a distributed storage to store clients and freelancer’s information. Both the client and the freelancer must have Ethereum accounts that are eth-address which would be used to uniquely identify them. Freelancers are only allowed to bid on jobs posted by a client with a bidding price after which payment is done in Ethereum blockchain. The research does not specify a limit to the number of jobs which a freelancer can acquire.

However, the major and unique feature of this system is its ability to limit the amount of work that can be done by a freelancer to 3 jobs at a time thereby increasing work speed, response rate or patronage and also giving newly verified freelancers the opportunity to access a freelance work.

**CHAPTER 3:**

**SYSTEM ANALYSIS AND DESIGN**

**INTRODUCTION**

System analysis and design is a term which describes methodologies for developing high quality information system which combines Information Technology, people and data to support business requirements [8]. It deals with planning the development of information systems through understanding and giving specific detail of what a system should do and how the components of the system should be implemented and work together. Analyzing and designing a system requires the use of various software design methodologies and the one used for this project is Object-Oriented Analysis and Design Methodology (OOADM).

Object-oriented analysis and design methodology (OOADM) is a technical approach used in the analysis and design of an application or system through the application of the object-oriented paradigm and concepts including visual modeling. This is applied throughout the development life cycle of the application or system, fostering better product quality and even encouraging participation and communication. Object-oriented analysis and design methodology (OOADM) use a set of diagrams or models called Unified Modelling Language (UML) diagrams to represent various views and functionalities of the system.

**3.0.1 REASONS FOR CHOOSING OOADM**

After a brief study of various methodologies such as Function Oriented Methodology, SSADM etc. It was discovered that they only take into consideration just a specific part of a system. For instance, the SSADM focuses on the system and can be divided into modules that perform a specific task. It does not show how the modules of the system relate with each other. For this reason, OOADM was chosen for this project because of the way it organizes and structure real life systems into objects by considering both the data and the functions of the system. It also shows how these functions relate with one another.

Some of the benefits of using OOADM for this project includes;

1. Modularity for easier troubleshooting;
2. reuse of code through inheritance;
3. flexibility through polymorphism; and
4. Effective problem solving.

**3.1 DESCRIPTION OF THE EXSISTING SYSTEM**

Online freelancing platforms in existence allows users register as either freelancer or client and post job listings thereby connecting freelancers to potential clients. These existing platforms work as a third party in helping individuals who work as freelancers earn online from any in the world. After extensive study of the existing system, some shortcomings were discovered. They include:

1. Difficulty for a newly registered freelancer to get a job,
2. low rate of patronage due to high competition, and
3. Inability to limit the number of jobs a freelancer can acquire thereby making it almost impossible for most people to get work.

**3.2 ANALSIS OF THE PROPOSED SYSTEM**

The analysis of the proposed system would be done with the use of the Unified Modelling Language (UML) tools which include use case diagram, class diagram and activity diagram. This proposed system has been designed to overcome the shortcomings of the existing system.

1. The ability to get work easily for newly registered freelancer;
2. High rate of patronage;
3. Ability to limit the number of jobs that a freelancer can acquire at a time.

**3.2.1 REQUIREMENT SPECIFICATIONS**

The analysis of the proposed system would be done using the following tools which include; Use case diagram, Class diagram and Activity diagram.

**3.2.1.1 USE CASE DIAGRAM**

A use case diagram is a graphical depiction of a user’s interaction with a system. It is used to show the details of system’s users and their interactions with the system.

The figure 3.0 shows the use-case diagram for the system.

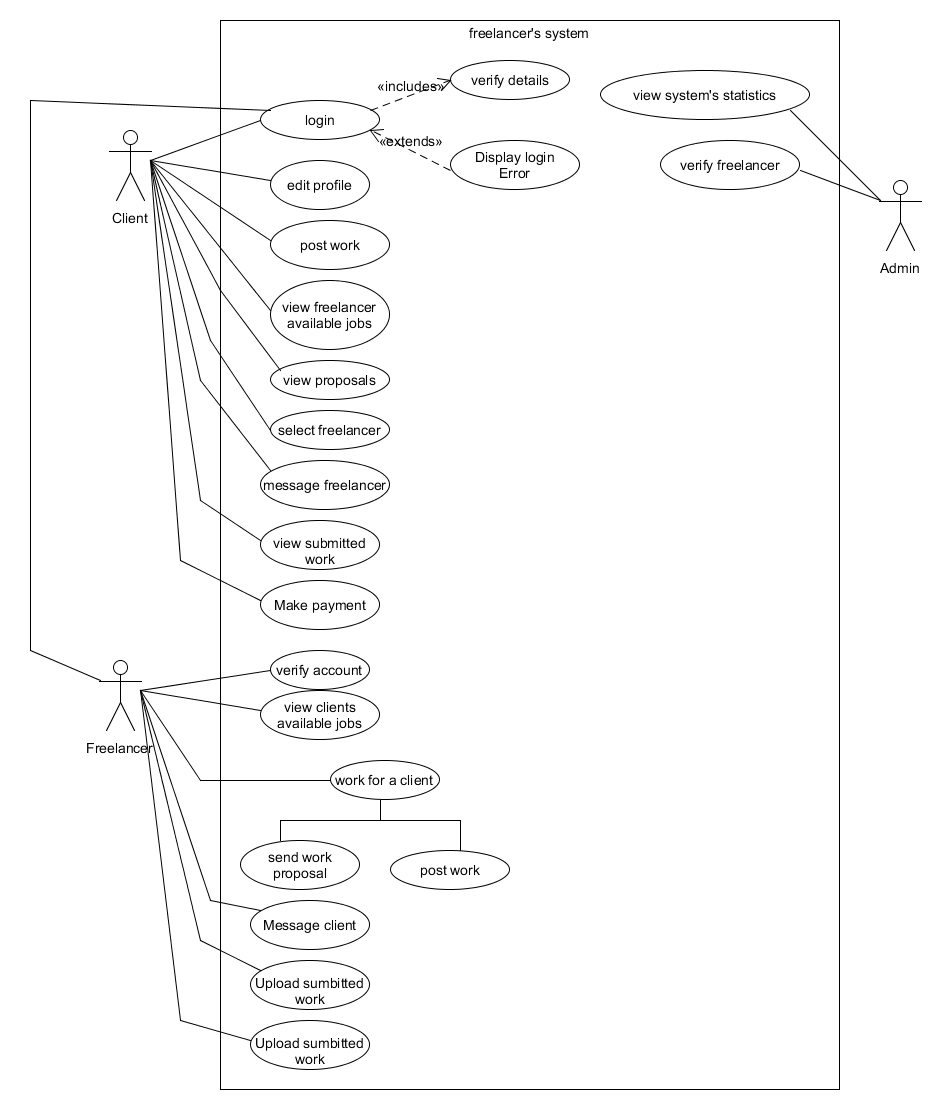


Figure 3.0: Diagram showing the use case diagram of the proposed system.

In figure 3.0 above, the diagram shows the various actors in the proposed system and how they interact with the system. The actors include; the client, the freelancer and the admin. The diagram indicates the functions these various actors can perform on the system.

**3.2.1.2 CLASS DIAGRAM**

Class diagram is a tool that is used to show or describe the attributes and operations of a class and also the constraints imposed on a system. It shows the system’s classes, their attributes, operations(methods), and the relationships among objects. The class diagram is shown in figure 3.1.

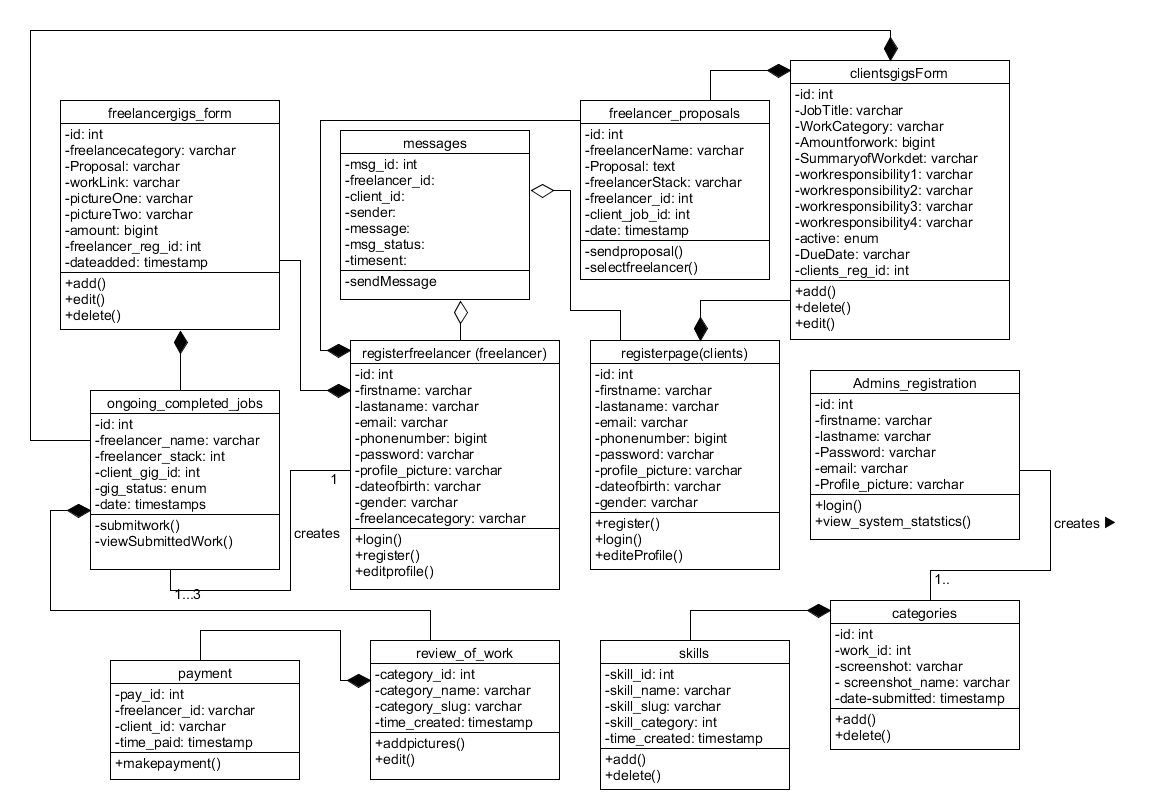


Figure 3.1: Class diagram of the proposed system

The figure, figure 3.1, above shows the class diagram of the proposed system. It shows in detail how each table or category are linked together. It also shows the functions of each category, the data type needed for each category.

**3.2.1.3 Activity Diagram**

Activity diagram is defined as a graphical representation of workflow of stepwise activities and actions with support for choice, iteration and concurrency. The activity diagram for the proposed system is shown in figure 3.2.

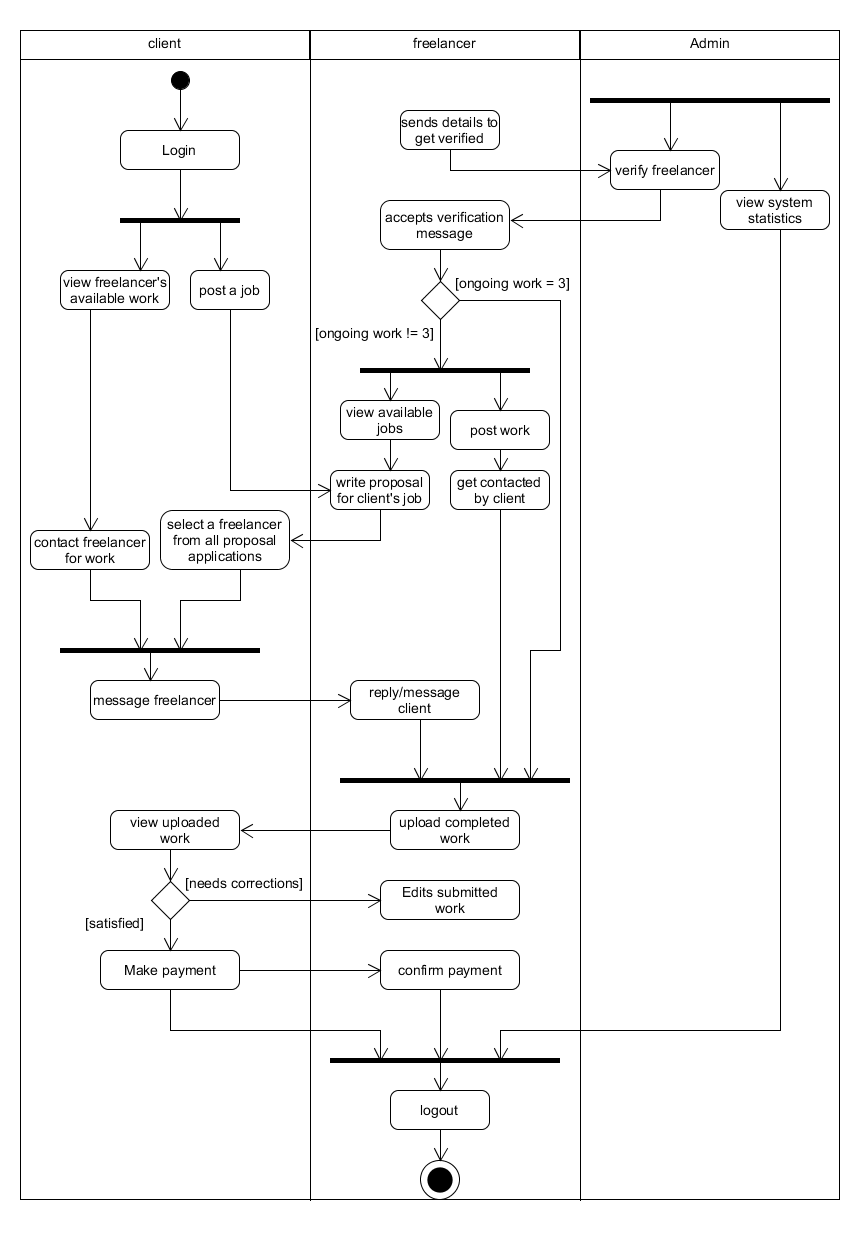


Figure 3.2: Activity diagram of the proposed system

The figure, figure 3.2, above shows the various activities that can be performed on the proposed system. It goes ahead to show in detail how the system operates and the various activities that can be performed by each user.

**3.3** **DESIGN OF THE PROPOSED SYSTEM**

System design is defined as the process of defining, developing and designing systems which satisfies the specific needs and requirements of a project. The design of the proposed system will be done using;

**3.3.0 Database Design**

MySQL database serves as the database used for this project. It is an open-source relational database management system based on the structured query language. The following tables show the tables that were used in this project.

Table 3.1 shows the register freelancer table. This is the table used to store the details of all registered freelancers.

**Table 3.1 Registerfreelancer**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | Data type | Size | Null | Description | Action | Extra |
| id | int | 11 | No | user id | Primary key | Auto increment |
| firstname | text | 255 | No | Freelancer’s name |  |  |
| lastname | text | 255 | No | Freelancer’s lastname |  |  |
| email | varchar | 255 | No | Freelancer’s email |  |  |
| pass | varchar | 255 | No | password |  |  |
| Profile\_picture | varchar | 255 | No | Profile picture |  |  |
| dateofbirth | text | 255 | No | Date of birth |  |  |
| gender | text | 255 | No | user’s gender |  |  |
| freelancercategory | varchar | 40 | No | Work category |  |  |
| user\_status | enum | 2 | No | User status |  |  |
| test status | int | 11 | No | Test status |  |  |
| test\_result\_status | int | 11 | No | User test result |  |  |
| dateregistered | timestamp | 6 | No | Date registered |  |  |

Table 3.2 shows the register client table. This is the table used to store the details of all registered clients.

**Table 3.2 registerclient**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | Data type | Size | Null | Description | Action | Extra |
| id | int | 11 | No | user id | Primary key | Auto increment |
| firstname | varchar | 255 | No | client’s name |  |  |
| lastname | varchar | 255 | No | client’s last name |  |  |
| email | varchar | 255 | No | client’s email |  |  |
| pass | varchar | 255 | No | password |  |  |
| Profile\_picture | varchar | 255 | No | Profile picture |  |  |
| dateofbirth | text | 255 | No | Date of birth |  |  |
| gender | Text | 255 | No | user’s gender |  |  |
| dateregistered | timestamp | 6 | No | Date registered |  |  |

Table 3.3 shows the message table. This is the table used to store all messages between clients and freelancers.

**Table 3.3 message**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | Data type | Size | Null | Description | Action | Extra |
| msg\_id | int | 11 | No | Unique id | Primary key | Auto increment |
| fl\_id | int | 11 | No | Freelancer id | Foreign key |  |
| cl\_id | int | 11 | No | client id | Foreign key |  |
| sender | enum | 2 | No | Sender identifier |  |  |
| message | text | 255 | No | Message sent |  |  |
| message\_status | enum | 2 | No | Seen status |  |  |
| time\_sent | timestamp | 6 | No | Time sent |  |  |

Table 3.4 shows the Admin table. This is the table used to store the details of the admin.

**Table 3.4 Admin**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | Data type | Size | Null | Description | Action | Extra |
| id | int | 11 | No | Admin id | Primary key | Auto increment |
| firstname | varchar | 255 | No | Admin firstname |  |  |
| lastname | varchar | 255 | No | Admin lastname |  |  |
| password | varchar | 255 | No | Admin’s password |  |  |
| email | varchar | 255 | No | Admin’s email |  |  |
| Profile\_picture | varchar | 255 | No | Admin’s profile picture |  |  |

Table 3.5 shows the freelancers’ gigs form table. This is the table used to store the details of all the works posted by every freelancer.

**Table 3.5 FreelancersGigsForm**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | Data type | Size | Null | Description | Action | Extra |
| id | int | 11 | No | Unique id | Primary key | Auto increment |
| freelancecategory | varchar | 255 | No | Freelancer category |  |  |
| proposal | varchar | 255 | No | Work proposal |  |  |
| workLink | varchar | 255 | No | Work link |  |  |
| pictureOne | varchar | 255 | No | Picture of work |  |  |
| pictureTwo | varchar | 255 | No | Picture of work |  |  |
| amount | bigint | 255 | No | Amount charged |  |  |
| Freelancer\_reg\_id | int | 11 | No | Freelancer registration id | Foreign key |  |
| dateadded | timestamp | 6 | No | date created |  |  |

Table 3.6 shows the clients jobs table. This is the table used to store the details of all the works posted by every client.

**Table 3.6 ClientsJobs**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | Data type | Size | Null | Description | Action | Extra |
| id | int | 11 | No | Unique id | Primary key | Auto increment |
| client\_name | varchar | 255 | No | Client’s name |  |  |
| jobTitle | varchar | 100 | No | Job title |  |  |
| WorkCategory | varchar | 255 | No | work category |  |  |
| AmountforWork | bigint | 20 | No | Price of work |  |  |
| SummaryofWorkdet | varchar | 255 | No | Work details |  |  |
| Workresponsibility1 | varchar | 255 | No | Work responsibilities 1 |  |  |
| Workresponsibility2 | varchar | 255 | No | Work responsibilities 2 |  |  |
| Workresponsibility3 | varchar | 255 | No | Work responsibilities 3 |  |  |
| Workresponsibility4 | varchar | 255 | No | Work responsibilities 4 |  |  |
| active | enum | 2 | No | Work visibility |  |  |
| DueDate | varchar | 50 | No | deadline |  |  |
| Clients\_reg\_id | int | 255 | No | Related client’s id | Foreign key |  |

Table 3.7 shows the categories table. This is the table used to store the details of all the available categories on the system under which a freelancer can register.

**Table 3.7 Categories**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | Data type | Size | Null | Description | Action | Extra |
| ctgy\_id | int | 11 | No | Categories id | Primary Key | Auto increment |
| ctygy\_name | varchar | 255 | No | Category name | Foreign key |  |
| ctgy\_slug | varchar | 255 | No | Category type |  |  |
| time\_creeated | timestamp | 6 | No | Time category was created |  |  |

Table 3.8 shows the skills table. This is the table used to store the details of all the skills under a particular category.

**Table 3.8 Skills**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | Data type | Size | Null | Description | Action | Extra |
| skill\_id | int | 11 | No | Unique id | Primary key | Auto increment |
| skill\_name | varchar | 211 | No | Skill name |  |  |
| skill\_slug | varchar | 255 | No | Skill type |  |  |
| skill\_category | int | 11 | No | Category of skill | Foreign key |  |
| time\_created | timestamp |  | No | time skills were created |  |  |

Table 3.9 shows the freelancers proposal table. This is the table used to store the details of all the proposals sent by freelancers to a particular job.

**Table 3.9 freelancers\_proposals**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | Data type | Size | Null | Description | Action | Extra |
| id | int | 11 | No | Unique id | Primary key | Auto increment |
| freelancerName | varchar | 211 | No | Freelancer’s name |  |  |
| Proposal | text | 255 | No | Proposal details |  |  |
| freelancerStack | varchar | 255 | No | Skill set |  |  |
| Freelancer\_id | int | 11 | No | Freelancer’s id | Foreign key |  |
| Client\_job\_id | int | 11 | No | client’s work id |  |  |
| date | timestamp |  | No | Date proposal was sent |  |  |

Table 3.10 shows ongoing and completed work table. This is the table used to store the details of all the works that have been requested for, that are still ongoing and works that have been completed.

**Table 3.10 Ongoing\_and\_completed\_work**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | Data type | Size | Null | Description | Action | Extra |
| id | int | 11 | No | Unique id | Primary key | Auto increment |
| Freelancer\_name | varchar | 211 | No | Freelancer’s name |  |  |
| Freelancer\_stack | varchar | 255 | No | Skill set |  |  |
| Freelancer\_id | int | 11 | No | Freelancer id | Foreign key |  |
| Client\_gig\_id | int | 11 | No | Client’s work id |  |  |
| gig\_status | enum |  | No | Status of work |  |  |
| date | timestamp |  | No | time skills were created |  |  |

Table 3.11 shows the review of work table. This is the table used to store the details of all the screenshots of work sent to a particular client for review.

**Table 3.11 Review\_of\_work**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | Data type | Size | Null | Description | Action | Extra |
| id | int | 11 | No | Unique id | Primary key | Auto increment |
| work\_id | varchar | 211 | No | id of work | Foreign key |  |
| screenshot | varchar | 255 | No | Screenshot of work done |  |  |
| Screenshot\_name | varchar | 11 | No | Name of screenshot |  |  |
| date\_submitted | timestamp |  | No | time skills were created |  |  |

**3.3.1 System Architecture**

In the development of this project, a 3-tier architecture was used in the design of the system. A three-tier architecture is a client-server architecture in which the functional process logic, data access, computer data storage and user interface are developed and maintained as independent modules on separate platforms.

The tiers are the presentation tier, the middle tier and the data tier. The presentation tier which is the user interface was implemented using HTML, CSS, JavaScript, Bootstrap and jQuery. The middle tier was designed using PHP and the data tier which stores the data was implemented using MySQL. The system architecture is shown in figure 3.3.

Server

HTML

CSS

JavaScript

PHP

MySQL

Database

Middle Tier

Data Tier

Figure 3.3: Diagram showing the system architecture.

The figure above shows the architecture of the system. It indicates the three different tiers of the system and also indicates the technologies used in the presentation tier, the middle tier and the data tier.

**CHAPTER 4**

**SYSTEM IMPLEMENTATION**

**INTRODUCTION**

System implementation can be defined as a process whereby a newly developed system is made available to a prepared set of users. It is a process of ensuring that a system is operational. System implementation uses the system design structured which was created and also uses the result of the system analysis in order to construct system elements that meet the project requirements.

This chapter discusses the necessary implementation and development requirements for achieving the proposed system. It describes briefly the choice of development environment which, for this project, is sublime text. It also goes ahead to describe the implementation architecture and the various software tests that were carried out on the system during the development phase. This chapter discusses how the system was developed, ensuring that it is operational and provides all the necessary requirements.

**4.1 CHOICE OF DEVELOPMENT ENVIRONMENT**

A development IDE (Integrated Development Environment) is a development environment is a software application that provides comprehensive facilities to computer programmers for software development. IDEs are designed to maximize programmer productivity by providing tight-knit components with similar user interfaces.

Sublime text chosen as the integrated development environment for the development of the freelancers’ system because of its autocomplete ability, simultaneous editing and its mini-map feature which makes it easier and faster to get from one part of the code file to another. The aforementioned features enabled faster debugging the system and they helped reduce total development time of the system.

The system was developed using some web technologies which include HTML, CSS, Bootstrap, JavaScript, jQuery which are used for the design of the system, and PHP and MySQL were used to configure the database of the system.

**4.2 IMPEMENTATION ARCHITECTURE**

This shows how the system works, the technologies used and the various ways these components are linked together.

DATEBASE

(MySQL)

Apache Web Server

APPLICATION TECHNOLOGIES

(html, css, bootstrap, javascript, jquery and php)

USERS

(admin, clients and freelancers)

Figure 4.0: Diagram showing the implementation architecture of this system.

In figure 4.0, the implementation architecture is shown. It indicates how the system functions. The application technologies get data from the user and also displays information to the user. The data gotten from the user is saved in the database through the apache server.

**4.3 SOFTWARE TESTING**

Software testing can be defined as a method of checking whether the actual software product matches expected requirements and to ensure that software product is free of any defects or faults. It involves execution of software system components using manual or automated tools to evaluate one or kore properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.

For this project, this system was tested at every stage of its development in order to detect errors and remove them immediately. The software testing done on the system was in two phases:

1. Testing done during the development phase; and
2. Testing done by running the website on a web browser.

Figure 4.1 requests a user to enter valid details to register

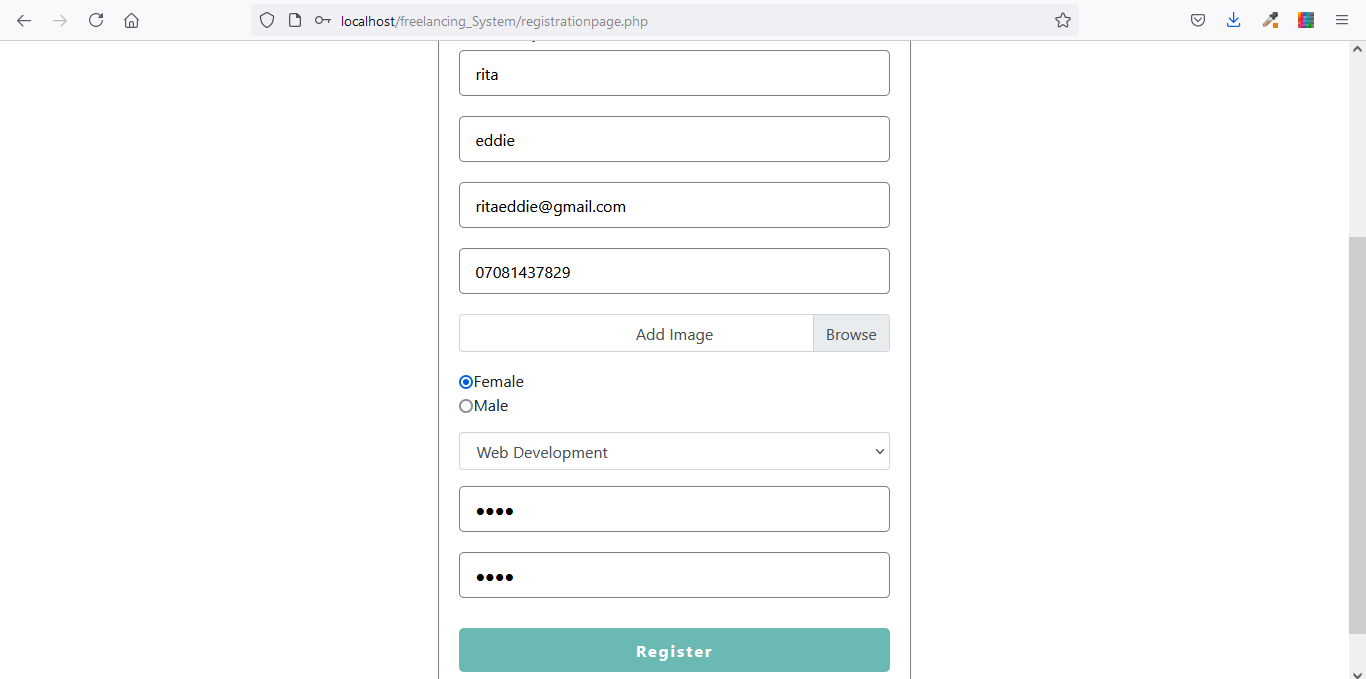


Figure 4.1: A request made to the user for registration

The figure, figure 4.1, above shows the registration page. This form is used to get registration details from the user before the user would be able to make use of the system. Without registration, the user cannot make use of the system and therefore would not be recognized.

Figure 4.2 Shows successful registration and a request made to the user to login

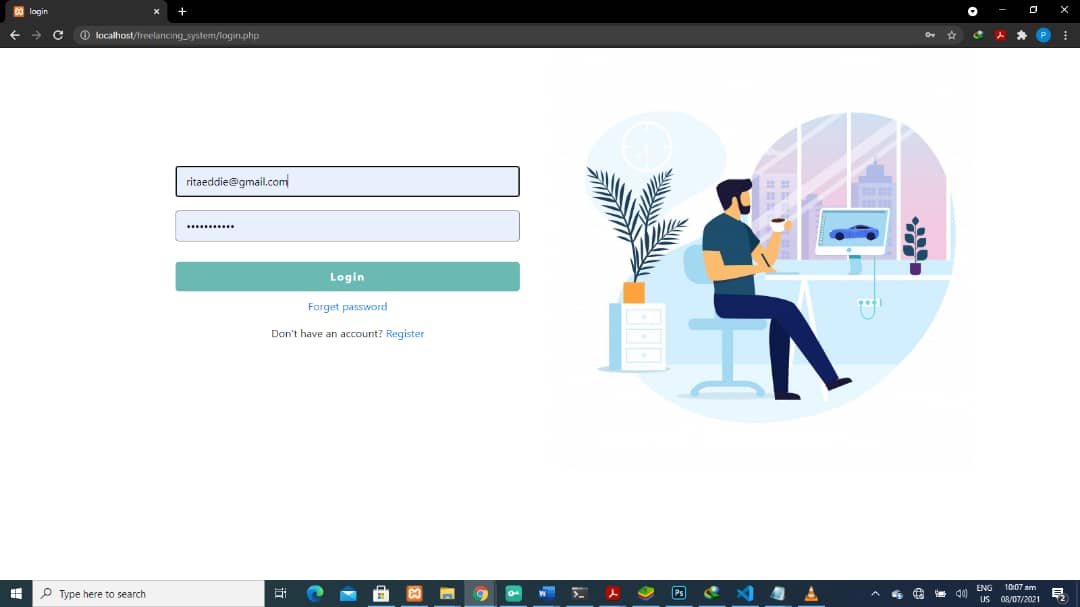


figure 4.2: A login request made to the user.

Figure 4.2 shows the login page. The user is sent to this page after successful registration. The user is then prompted to enter his email and password which the user registered with.

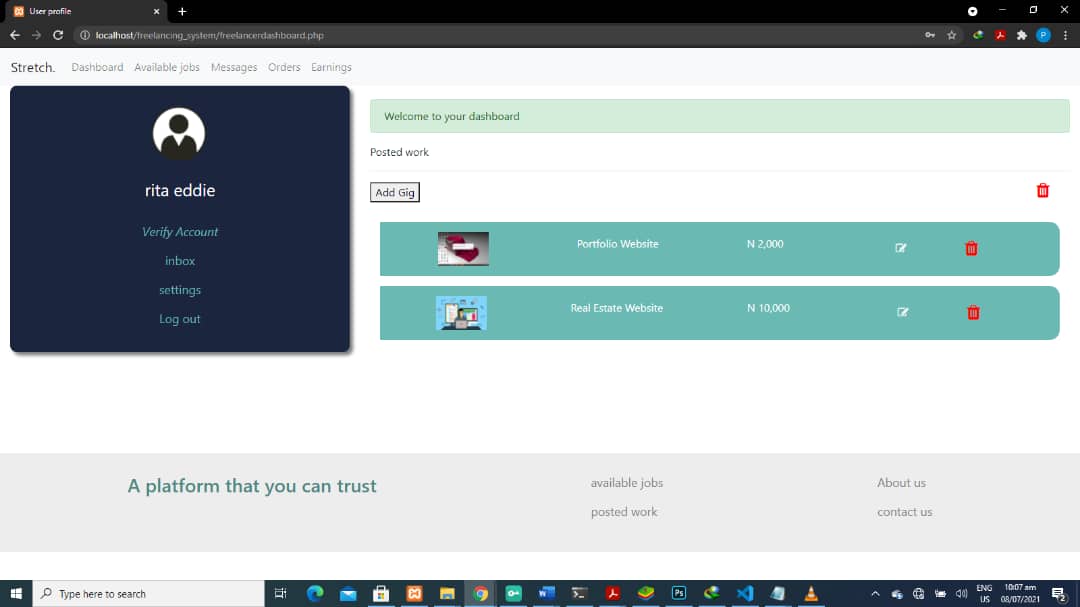


Figure 4.3: Shows successful login of the user into his/her account.

Figure 4.3 above shows that the user is successfully logged into the system. The page shows the users dashboard through which the user can perform other activities.

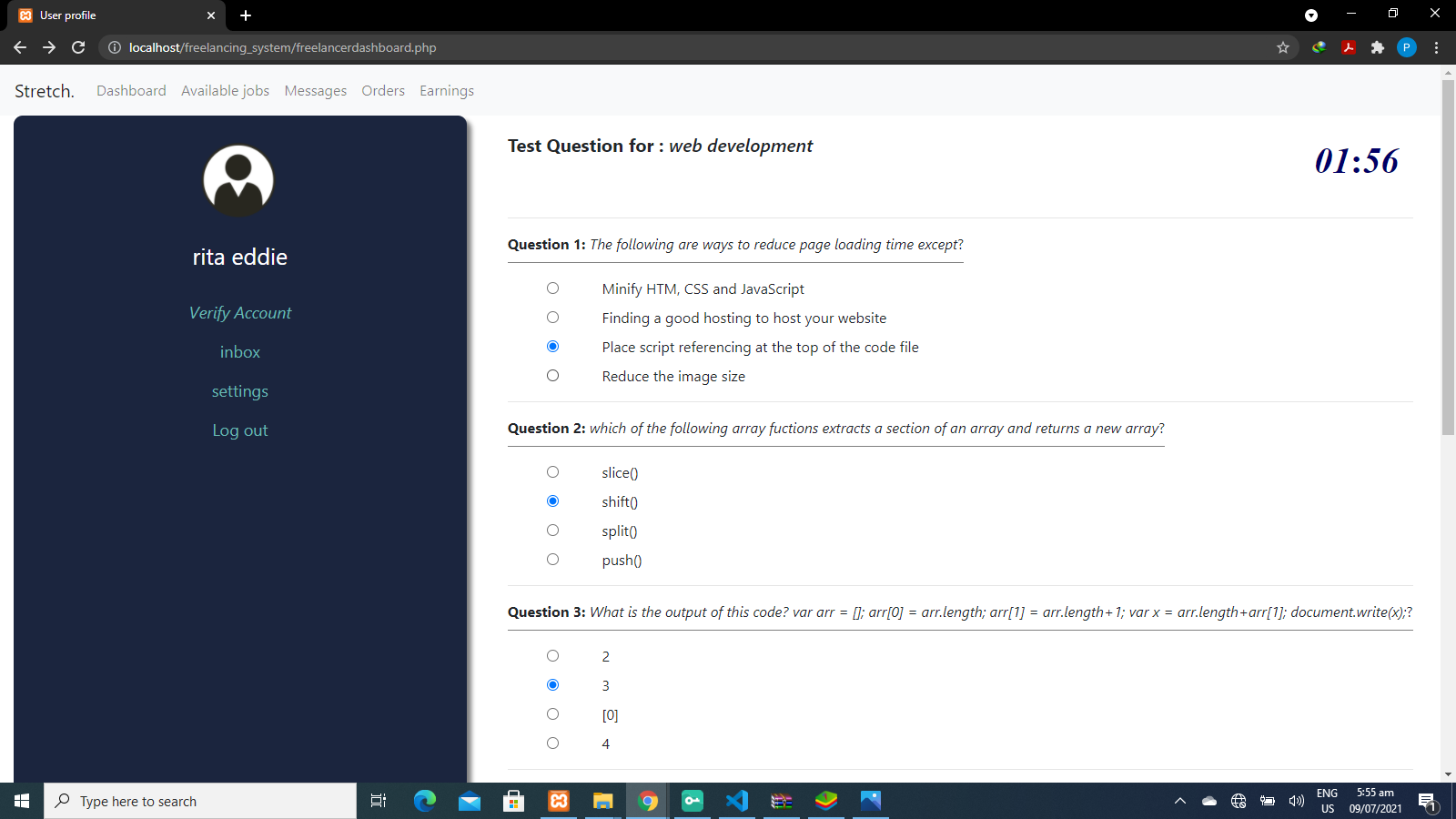


Figure 4.4: Diagram showing the verification page.

In the figure 4.4, The user goes through the verification process before he or she can apply for work. The feature is only made available to the freelancers. When the user submits the form, he is graded and verified if the user’s grade is up to the pass mark set for the verification questions.

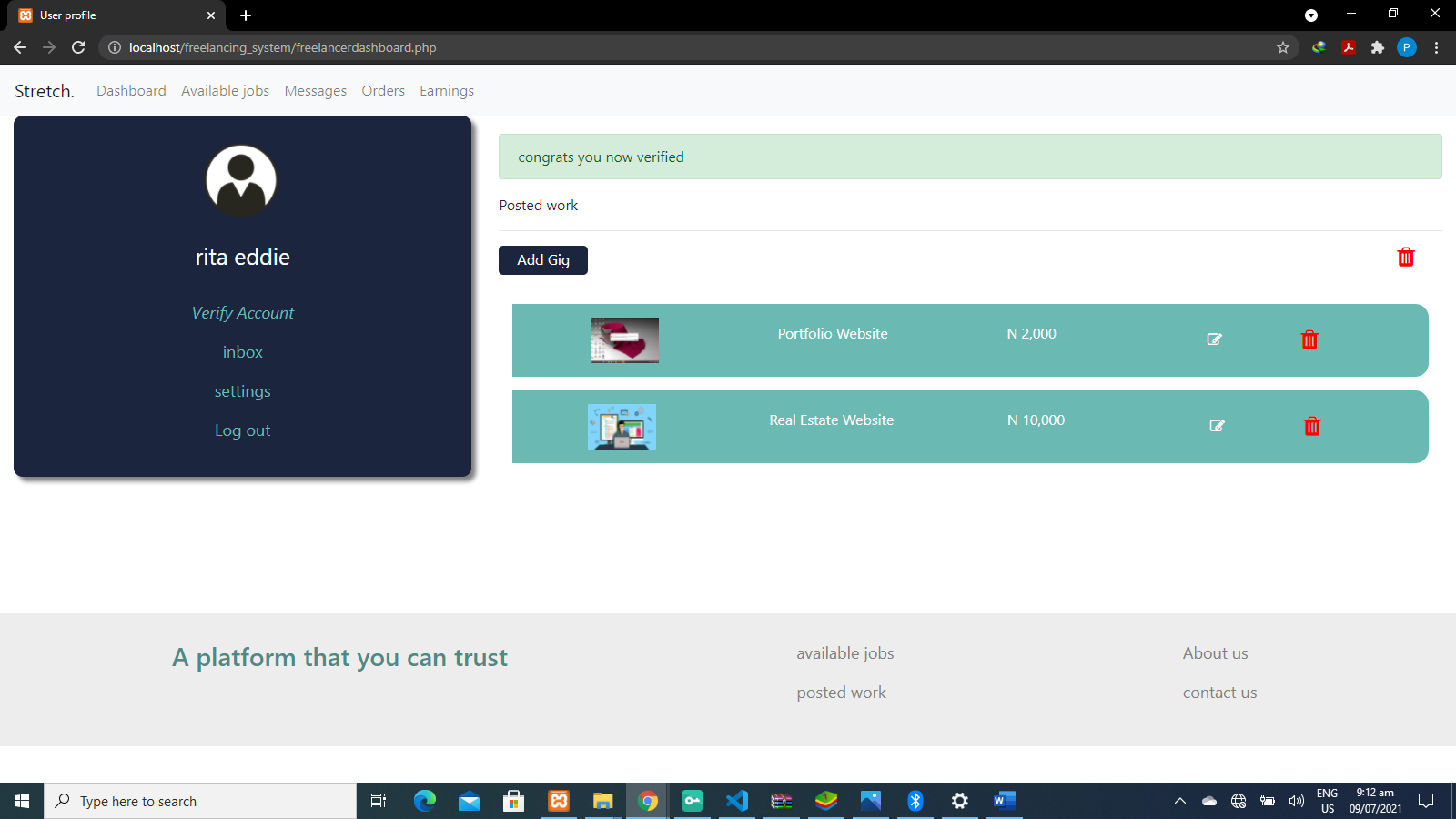


Figure 4.5: Showing that the verification test was successful

In figure 4.5, the user has completed the verification process successfully. The user would now be able to apply for work posted by the client. The freelancer can also post various works which he or she can do.

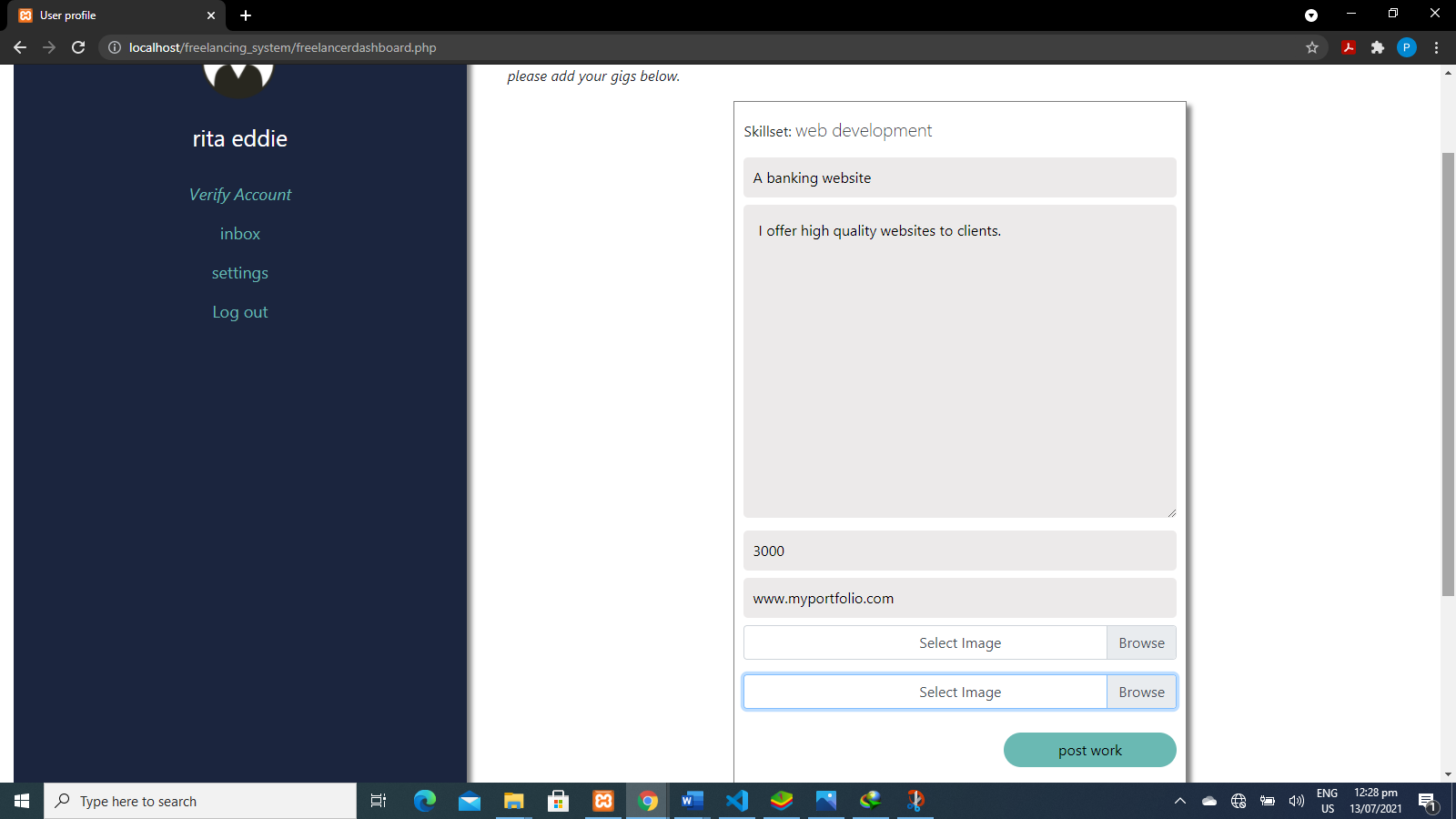


Figure 4.6: Showing the work page.

In the figure 4.6 above, the user is requested to enter the necessary details to post the work which he or she can perform. This page is used to add work the freelancer can do and it is then made visible to clients on the system.

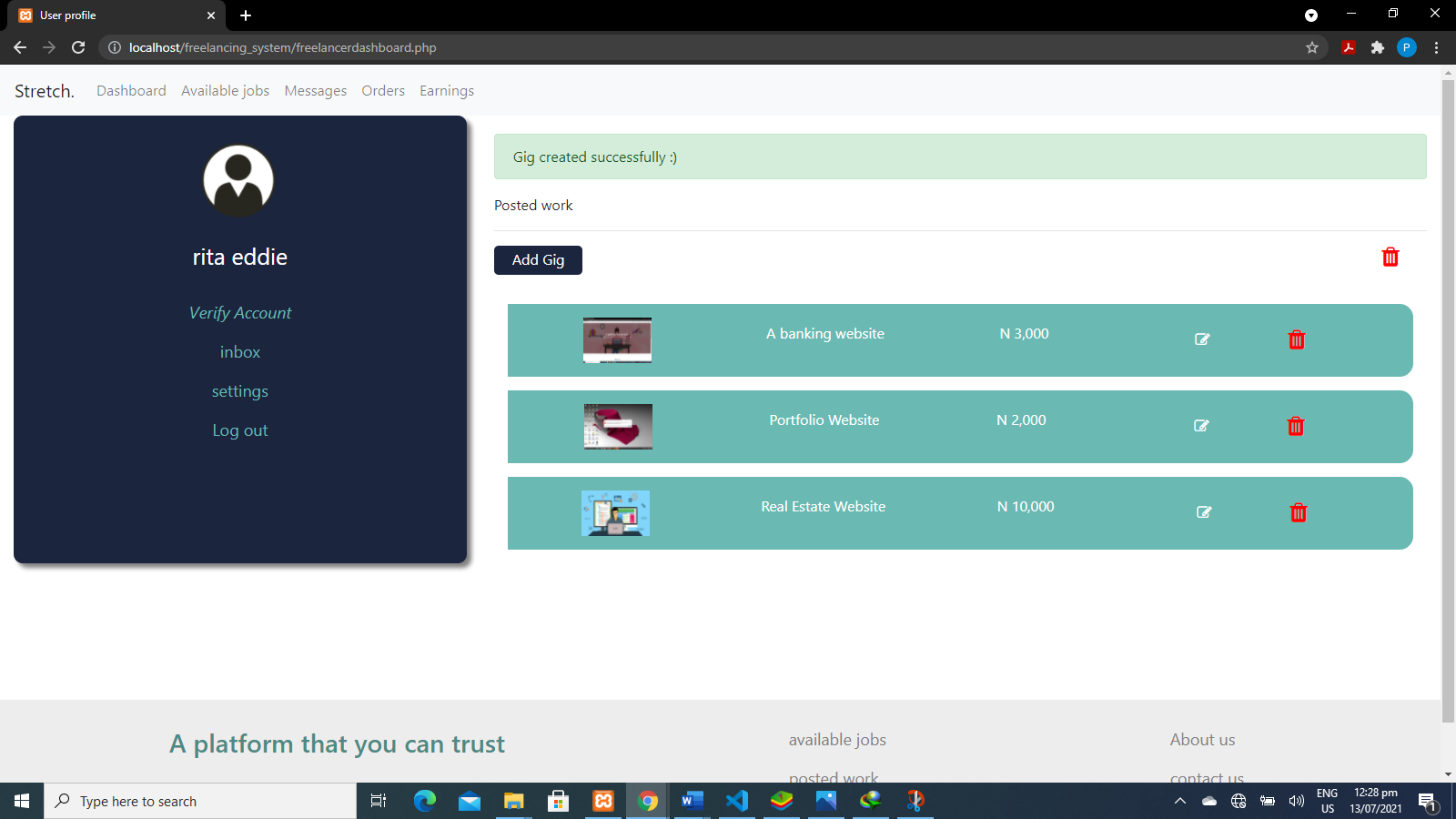


Figure 4.7: Showing successful upload of available work.

Figure 4.7 above shows that the form has been submitted successfully and it also indicates that the details in the form has been saved successfully in the database and also made visible to clients.

**4.4 DOCUMENTATION**

Documentation of software is defined as any written text that describes a software or program to its users. It can also be defined as all written documents and materials dealing with software product development. The term is also sometimes used to mean the source of information about the product contained in design documents, detailed code comments and white papers. Documentation consists of internal and external documentation. For this system, Internal documentation was done with the use of comments. Comments are defined as a readable explanation written in the source code of a program which is not executable during compilation process of that program. It is categorized into single line comments and multi-line comments. Single line comments are used to give a text description for one line of code while multi-line comments are used when the text description for a code is too big to fit into one line and both categories of comments make a program easy to understand and easily maintainable for future use.

**4.4.1 USER MANUAL**

A user manual is a text description that gives assistance to people using a particular system. It contains step by step procedures of a system access and use. For this project, the user manual is the external documentation. A brief explanation on how the user can use the system is given below.

1. The user first sees the home page where he or she sees the categories that are available in the system.
2. The user can either login or register if he or she is a new user.
3. The user manual also shows the user how he or she can register as a freelancer or a client.

**4.4.2 SOURCE CODE LISTING**

The source code listing can be found in the appendix of this research work.

**CHAPTER 5:**

**SUMMARY AND CONCLUSION**

**5.0 SUMMARY**

This system is designed in such a way that individuals can register as a freelancer or client. It gives freelancers easy access to job opportunities and also gives potential employers access to verified freelancers. Freelancers are allowed to register according to a particular skill set or work category whereby he or she can only apply to jobs in that particular category. After registration the freelancer has to be verified by answering some test questions related to his/her category or skill set in order to be able to apply for jobs that have been posted by an employer. The freelancer applies to client’s job offer by sending proposals to the client.

Clients are allowed to view the jobs which the freelancer can offer and is also allowed to contact the freelancer for the work he/she posted. Freelancers are payed after completing the client’s work to the client’s satisfaction through flutterwave rave payment API but due to limitations of this research work, this functionality only executes when the system is hosted on a live server. This system helps reduce the difficulty for new freelancers to get job and the low response rate from clients by limiting the maximum amount of active jobs that can be acquired by a freelancer to 3 jobs. This improves productivity and fast delivery of a client’s work and, most importantly, allows freelancers earn after successfully completing a job for a client.

**5.1 CONLUSION**

Most freelancing websites that exist lack the ability to aid new freelancers get a job without any kind of payment for promotion. Although this method helps to give a freelancer the needed exposure on the platform but it would not be suitable for a freelancer who is just starting out in the freelance field. The platform being developed in this project gives new freelancers the opportunity to apply for work by limiting the number of jobs that each freelancer can handle at a time.

With the project, individuals can have a way to earn a living from the comfort of their homes or anywhere at all. This enables these individuals to meet up with the cost of living in the country thereby reducing the rate of unemployment and crime.

**5.3 RECOMMENDATIONS**

Throughout the course of this project, I have been able to solve a little part of the problems which freelancers face on existing freelance platforms. Further research should focus on adding more functionality such as e-wallet whereby the freelancer can withdraw at any time he or she pleases and also ID card verification to ensure that each verified user is a real person thereby reducing the rate by which users create false identity or commit fraud.

**REFERENCE**

[1] S. Tams and M. B. Arthur “New directions for Boundaryless careers: Agency and Interdependence in a Changing World”. *Journal of Organizational Behaviour,* vol 5, p.311-322,2010.

[2] E.M. Akhmetshin, K.E. Kovalenko, J.E Mueller, A.K. Khakimov, A.V. Yumashev and A.D Khairullina “freelancing As A Type of Entrepreneurship: Advantages, Disadvantages and Development Prospects”. *Journal of Enterprenuership Education*, vol 21, p.2, 2018.

[3] W. Sonam, P. Tashi, R. Kailash, T. Tshewang and D. Nima “G-Task:Proposal for Development of a Freelancing Android Based Mobile Application in Bhutan”. *International Journal on Recent and Innovation Trends in Computing and Communication*, vol 5, p.566-570, June 2017. [Online]. Available: http://www.ijritcc.org

[4] A. Mwathi and B.S. Kasamani, “Design and Implementation of a collaborative Freelancers’ Sourcing Platform”. *International Journal of Scientific Research and Innovative Technology*, vol 4, no 10, p.178-182, October 2017.

[5] C. Dissanayake, D. Rathnayake, N. Dilhani, U. Kumarasiri, N. Kumari, K.A. Dilhini and T. Kulawansa, “Freelancing System”, 2020 Proceedings from the eighth International Conference on Advances in Computing, Electronics and Computing, ACEC 2019, pp.59-62

[6] P.P. Tushar, A. Sharma. and A. Mishra “ROZGAAR: Android App for Freelancing”. *International Journal of Emerging Trends in Engineering and Development*, vol 5, p.4-6, Aug-Sept 2020. [Online]. Available: http://rspublication.com/ijeted/ijeted\_index.htm

[7] P. Deshmukh, S. Kalwaghe, A. Appa and A. Pawar, “Decentralized Freelancing using Ethereum Blockchain”,2020 International Conference and Signal Processing (ICCSP), 2020, pp. 881-883, doi:10. 1109ICCSP48568.2020.9182127.

[8] S. Ramkrishnan, “System Analysis and Design”. *International Journal of Information Technology and software Engineering,* p.1, 2012.

APPENDIX

LOGIN PAGE

<?php

require\_once("includes/DB.php");

require\_once("includes/sessions.php");

require\_once("includes/functions.php");

if(isset($\_POST['loginfreelancer'])){

$Email = $\_POST['email'];

$Password = $\_POST['pass'];

if(empty($Email) || empty($Password)){

echo "<script> alert('All feilds must be filled out')</script>";

// Redirect\_to("login.php");

}else{

$found\_Account=Login\_Freelancer\_Attempt($Email);

if($found\_Account){

if (password\_verify($Password,$found\_Account["pass"])) {

$\_SESSION['Freelancer\_ID']=$found\_Account["id"];

$\_SESSION['FreelancerName']=$found\_Account["firstname"];

$\_SESSION["SuccessMessage"]= "Welcome to your dashboard";

Redirect\_to('freelancerdashboard.php');

} else {

$\_SESSION["SuccessMessage"]= "Incorrect password";

Redirect\_to('login.php');

}

}else{

$\_SESSION["ErrorMessage"]= "incorrect email/password";}}}

<!DOCTYPE html>

<html>

<head>

<title>login</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" type="text/css" href="bootstrap/dist/css/bootstrap.css">

<link href="https://fonts.googleapis.com/css?family=Lora&display=swap" rel="stylesheet">

<link rel="stylesheet" type="text/css" href="font-awesome-4.7.0/font-awesome-4.7.0/css/font-awesome.css">

<link rel="stylesheet" type="text/css" href="css-files/register.css">

<link rel="stylesheet" type="text/css" href="css-files/login.css">

</head>

<body>

<div class="container">

<div class="row">

<div class="col-md-6 loginformspace">

<div class="mycontainhidereg loginchoice container" id="loginchoice">

<?php echo ErrorMessage();?>

<?php echo SuccessMessage();?>

<div class="row">

<div class="col-md-6"><pclass="loginasfreelancer" onclick="openTabs(event, 'loginformfreelancer')"> login as freelancer</p> </div>

<div class="col-md-6"><p class="loginasclient" onclick="openTabs(event, 'loginformclient')" name="clientloginbtn">login as client</p></div></div></div>

<div class="mycont mycontainhidereg" id="loginformfreelancer"><div class="container"><form method="POST" action="login.php" class="text-center">

<input type="email" name="email" placeholder="Enter Email" class="inputs"> <br>

<input type="password" name="pass" placeholder="Enter password" class="inputs"><br>

<input type="submit" name="loginfreelancer" value="Login" class="reg">

<p><a href="javascript:void(0)" data-toggle="modal" data-target="#freelancerPassword">Forget password </a></p>

<p>Don't have an account? <a href="registrationpage.php">Register </a></p></form></div> </div>

<div class="mycont mycontainhidereg" id="loginformclient"><div class="container">

<form method="POST" action="login.php" class="text-center">

<input type="email" name="email" placeholder="Enter Email" class="inputs"> <br>

<input type="password" name="pass" placeholder="Enter password" class="inputs"><br>

<input type="submit" name="loginclient" value="Login" class="reg">

<p><a href="javascript:void(0)" data-toggle="modal" data-target="#clientPassword">Forget password </a></p>

<p>Don't have an account? <a href="registrationpage.php">Register </a></p>

</form></div></div></div>

<div class="col-md-6"><img src="images/image3.jpg"></div></div></div>

FREELANCER DASHBOARD: ADD WORK

<?php

if (isset($\_POST['AddGigToList'])) {

$proposal = $\_POST['proposal'];

$workType = $\_POST['workType'];

$imageOne = $\_FILES["Image1"]["name"];

$URLlink = $\_POST['workLink'];

$TargetOne = "images/freelancer\_workImages/".basename($\_FILES["Image1"]["name"]);

$imageTwo = $\_FILES["Image2"]["name"];

$TargetTwo = "images/freelancer\_workImages/".basename($\_FILES["Image2"]["name"]);

$amountCharged = $\_POST['amountCharged'];

global $ConnectingDB;

$sql = "INSERT INTO freelancergigform(freelancecategory,workType, Proposal, workLink, pictureOne, pictureTwo, amount,freelancer\_reg\_id)";

$sql .= "VALUES(:freelancecategorY,:workTypE, :proposaL, :worklinK, :pictureonE, :picturetwO, :amounT, :postidformurl)";

$stmt = $ConnectingDB->prepare($sql);

$stmt->bindValue(':freelancecategorY',$ExistingCategory);

$stmt->bindValue(':workTypE',$workType);

$stmt->bindValue(':proposaL',$proposal);

$stmt->bindValue(':worklinK',$URLlink);

$stmt->bindValue(':pictureonE',$imageOne);

$stmt->bindValue(':picturetwO',$imageTwo);

$stmt->bindValue(':amounT',$amountCharged);

$stmt->bindValue(':postidformurl',$UserId);

$Execute=$stmt->execute();

if($Execute){

move\_uploaded\_file($\_FILES["Image1"]["tmp\_name"],$TargetOne);

move\_uploaded\_file($\_FILES["Image2"]["tmp\_name"],$TargetTwo);

$\_SESSION["SuccessMessage"]= "Gig created successfully :)";

}else{$\_SESSION["ErrorMessage"]= "A problem occured. Please try again";}}?>

<div class="container mycontainhide " id="addinggigs">

<div class="backarrow" onclick="openTabs(event, 'Dashboard')"> &leftarrow; Back</div>

<p class="pt-3"><em>please add your gigs below.</em></p>

<div class="applicationformmainflex"><div class="applicationformmain addinggigsmain">

<form class="text-center" action="freelancerdashboard.php" method="POST" enctype="multipart/form-data"><p class="text-left"> Skillset: <span class="lead"><?php echo $ExistingCategory; ?></span></p><input type="text" name="workType" placeholder="title of what you are offering." class="amountofcharge"><textarea rows="13" placeholder="please write your proposal here" name="proposal" class="amountofcharge p-3"></textarea>

<input type="text" name="amountCharged" placeholder="amount charged" class="amountofcharge"><input type="text" name="workLink" placeholder="Please Place a link to your recent website here" class="amountofcharge"><div class="form-group">

<div class="custom-file">

<input class="custom-file-input" type="File" name="Image1" value="" id="imageselect">

<label for="imageselect" class="custom-file-label">Select Image</label>

</div>

</div>

<div class="form-group">

<div class="custom-file">

<input class="custom-file-input" type="File" name="Image2" value="" id="imageselect">

<label for="imageselect" class="custom-file-label">Select Image</label>

</div>

</div>

<div class="text-right">

<input type="submit" name="AddGigToList" value="post work" class="applicationformapplybtn"></div>

</form>

</div>

</div></div>

PAGE FOR VIEWING FREELANCERS WORK THAT ARE AVAILABLE

<div class="row" id="workSearch">

<?php

if(!diplayAvailbleFlncrJob()){

echo "no freelancers match";

}else{

$ConnectingDB;

$sql = "SELECT \* FROM freelancergigform ORDER BY id desc";

$stmtfreelancergigs = $ConnectingDB->query($sql);

while ($DataRows = $stmtfreelancergigs->fetch()){

$freelancer\_id = $DataRows['freelancer\_reg\_id'];

$fetch = $ConnectingDB->query("SELECT COUNT(id) AS counter FROM ongoing\_completed\_work WHERE (gig\_status='ONGOING' OR gig\_status='UNDER REVIEW') AND freelancer\_id='$freelancer\_id'");

$fetchRow = $fetch->fetch();

$counter = $fetchRow['counter'];

if($counter<3){

$detail\_work\_id = $DataRows['id'];

$workType = $DataRows['workType'];

$GigProposal = $DataRows['Proposal'];

$GigPictureOne = $DataRows['pictureOne'];

$GigPictureTwo = $DataRows['pictureTwo'];

$GigAmount = $DataRows['amount'];

$freelancecategory = $DataRows['freelancecategory'];

?>

<div class="col-md-4 mb-4">

<div class="availbletab-item">

<div class="availabletabimage">

<img src="images/freelancer\_workImages/<?php echo $GigPictureOne;?>">

<div><h6 class="freelancername"><?php echo $workType; ?></h6></div></div>

<p class="worktitle">category:<span><?php echo $freelancecategory;?></span></p>

<p class="priceofwork">Amount: N <span class="priceofworkmoney"><?php echo number\_format($GigAmount);?></span></p>

<div class="availbletab-details mb-3">

<p>

<?php

if(strlen($GigProposal)> 20){

$GigProposal = substr($GigProposal,0,20)."...";} ?>

<?php echo $GigProposal; ?>

</p>

</div>

<a href="work\_detail\_byFreelancer.php?detail\_of\_work=<?php echo $DataRows['id'];?>"><div class="available-apply text-center"> View </div></a>

</div>

</div>

<?php } } }?></div></div>

PAGE FOR SELECTING FREELANCER FOR A JOB

<?php

require\_once("includes/DB.php");

require\_once("includes/sessions.php");

require\_once("includes/functions.php");

$PropId = $\_GET['propId'];

$gigId = $\_GET['InitialpropId'];

if (isset($PropId) || isset($\_POST['chooseFreelancer']) || isset($gigId)) {

// $myattri = $\_POST['chooseFreelancer'];

if(isset($\_POST['chooseFreelancer'])){

$activeOn = "ON";

$freelancerName = $\_POST['freelancerName'];

$freelancerStack = $\_POST['freelancerStack'];

$freelancer\_id = $\_POST['freelancer\_id'];

$gigid = $\_POST['gigId'];

global $ConnectingDB;

$sqlInsert = "INSERT INTO ongoing\_completed\_work(freelancer\_name, freelancer\_stack, freelancer\_id, client\_gig\_id)

VALUES(:freelancerName, :freelancerStack, :freelancerId, :clientGigId)";

$stmt = $ConnectingDB->prepare($sqlInsert);

$stmt->bindValue('freelancerName', $freelancerName);

$stmt->bindValue('freelancerStack', $freelancerStack);

$stmt->bindValue('freelancerId', $freelancer\_id);

$stmt->bindValue('clientGigId', $gigid);

// var\_dump($freelancer\_id); die;

$sqlDelete = "DELETE FROM freelancers\_proposals WHERE client\_job\_id='$gigid'";

// The code below is used to update the active column in clientsgigsform table

$sqlUpdate = "UPDATE clientsgigsform SET active='$activeOn' WHERE id='".$gigid."'";

$ExecuteInsert = $stmt->execute();

$ExecuteDelete = $ConnectingDB->query($sqlDelete);

$ExecuteUpdate = $ConnectingDB->query($sqlUpdate);

if ($ExecuteInsert) {

if($ExecuteDelete){

if($ExecuteUpdate){

$\_SESSION["SuccessMessage"]= "Congratulations. You have picked a frelancer for your work.";

Redirect\_to('companydashboard.php');

}else{

$\_SESSION["ErrorMessage"]= "problem occured. could not update";

Redirect\_to('companydashboard.php');

}}else{

$\_SESSION["ErrorMessage"]= "problem occured. could not delete";

Redirect\_to('companydashboard.php');

}}else{

$\_SESSION["ErrorMessage"]= "problem occured. could not inser";

Redirect\_to('companydashboard.php');

}}}else{

$\_SESSION["ErrorMessage"]= "could not get page";

Redirect\_to('index.php');

}

?>

<!DOCTYPE html>

<html>

<head>

<title>Select Proposal</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" type="text/css" href="bootstrap/dist/css/bootstrap.css">

<link href="https://fonts.googleapis.com/css?family=Lora&display=swap" rel="stylesheet">

<link rel="stylesheet" type="text/css" href="css-files/proposalInformation.css">

</head>

<body>

<div class="container main-container mt-5">

<?php

global $ConnectingDB;

$sql = "SELECT \* FROM freelancers\_proposals WHERE id ='".$PropId."'"; $stmtfreelancergigs = $ConnectingDB->query($sql);

while ($DataRows = $stmtfreelancergigs->fetch()){

$ProposalId =$DataRows['id'];

$freelancerName =$DataRows['freelancerName'];

$proposaldetails =$DataRows['Proposal'];

$freelancerStack =$DataRows['freelancerStack'];

$freelancer\_id =$DataRows['freelancer\_id'];

$client\_job\_id =$DataRows['client\_job\_id'];

$datesubmitted =$DataRows['date'];

}

?>

<div class="mycontainer">

<form action="proposalselection.php" method="POST">

<div class="freelancerDet-flex m-3">

<h4><em><?=$freelancerName;?></em></h4>

<p>Stack: <span><em><?=$freelancerStack;?></em></span></p>

</div>

<div class="options p-1"><p><input type="submit" name="chooseFreelancer" class="yes-btn" value="yes"></p>

<p class="No-btn">NO</p></div>

<input type="hidden" name="freelancerName" value="<?=$freelancerName;?>">

<input type="hidden" name="freelancerStack" value="<?=$freelancerStack;?>">

<input type="hidden" name="freelancer\_id" value="<?=$freelancer\_id;?>">

<input type="hidden" name="gigId" value="<?=$gigId;?>">

</form></div></div>

</body>

</html>