**SYNOPSIS**

**ABSTRACT**

**“E-CONTRACT"** is a web application for the contract work that available to everyone. This contract application covers all exiting contract work in the community. This will benefit both job seekers and customers. Users have the option to select services as required. Notification will be given the respective contractors in the nearest area, when the users select and submit the contract. The contract can be viewed for other contractor’s cases. Communication platform is available for users and contractors. The contractors have the option to record their features, that way we can select contractors that are user friendly. You can transfer money through this application itself. Users can also record the feedback of contractors.

**INTRODUCTION**

**INTRODUCTION**

“**E-CONTRACT** " web application is for contracts and is for contractor seekers. Normal contract work goes to the contractor, whether by people or by someone preferred, Maybe not getting a good contractor. But the Contractors are available through the website. In this we can understand their quality we like and talk to them and assign the job. That's why Contractors It can provide jobs both small and large so that people can look for work .work available to everyone. This contract application covers all exiting contract work in the community. When looking job we can look at places near us and choose.

**“E-CONTRACT "** is a web application for contractor work seekers as well as contractors. The 'E-CONTRACT' web application is a way for contractors to select acontractor based on the quality of their work and loyalty. With this web application it is possible to enter jobs small and large as well as find job opportunities. When we find a job, we can then choose jobs in places near us.

**OBJECTIVES OF THE PROJECT**

**OBJECTIVES OF THE PROJECT**

The purpose of this project is to When choosing a contractor, this contractor is a publication that will easily find a contractor based on the quality of the work and their reliability, as well as those looking for jobs small and large.

**SYSTEM ANALYSIS**

**FEASIBILITY STUDY**

All projects are feasible when given unlimited resources and infinite time. It is both unnecessary and prudent to evaluate the feasibility of a project at the earliest possible time. A feasibility study is not warranted for systems in which economic justification is obvious, technical risk is low, few legal problems are expected and no reasonable alternative exists. An estimate is made of whether the identified user needs may be satisfied using current software and hardware technologies. The study will decide if the proposed system will be cost effective from the business point of view and if it can be developed in the given existing budgetary constraints. The feasibility study should be relatively cheap and quick. The result should inform the decision of whether to go ahead with a more detailed analysis. Before any user request is to be accepted, it is mandatory to check whether the new system is feasible or not. The major purpose of the analysis is to see that the development is technically and operationally helpful to the organization or not. Thus when a new application is proposed it normally goes through a feasibility study before it is approved for developed. The key considerations involved in the feasibility are:

1. Technical Feasibility

2.Operational Feasibility

3.Economical Feasibility

4.Behavioral Feasibility

1. **TECHNICAL FEASIBILITY**

A study of function, performance and constraints may improve the ability to create an acceptable system. Technical feasibility is frequently the most difficult to achieve at the stage of product engineering process. Considering that are normally associated with the technical feasibility include:

1. Development Risk

2.Resource Availability

3. Technology

Technical feasibility study deals with the hardware as well as software requirements. The scope was whether the work for the project is done with the current equipment and the existing software technology has to be examined in the feasibility study. The outcome was found to be positive.

The proposed system **"E-CONTRACT"** is technically feasible because the software required are easy to deploy and handle. The necessary software platform is already there. Connection available to all the systems anyways. So, the web application is quite easy to handle

1. **OPERATIONAL FEASIBILITY**

Operational feasibility is a measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development. To ensure success, desired operational outcomes must be imparted during design and development.

These include such design-dependent parameters such as reliability maintainability, usability supportability, disposability, sustainability, affordability and others. Therefore operational feasibility is a critical aspect of systems engineering that needs to be an integral part of the early design phases.operational feasibility is a critical aspect of systems engineering that needs to bean integral part of the early design phases.

The system is operationally feasible since software requirements and installation can easily handle and there is no additional effort is to be made to train and educate the user son the new way of the system. The proposed system "E-CONTRACT" requires minimum effort fordevelopment and easily operable. So I can say that in above aspects my proposed system fit and match is operationally feasible

1. **ECONOMICAL FEASIBILITY**

Justification of any capitals outlay is that it will reduce expenditure or improve the quality of service, which in turn may be expected to provide increased profits and reputation. The technique of cost benefit analysis is often used as a basis for assessing economic feasibility .Cost benefit feasibility includes both hardware and software feasibility

In this system no initial Investment is needed. The proposed system **"E-CONTRACT"** was developed with the available resources. Since cost input is almost nil the output of the software is always profit. Hence the system is economically feasible. The purpose of the economic feasibility assessment is to determine the positive economic. Benefits to the society that the proposed system will provide. It is very essential because the main goal of the proposed system includes quantification and identification of all the benefits expected. This assessment typically involves a cost/benefits analysis, long term cooperative income strategies and cost of resources needed for development.

1. **BEHAVIOURAL FEASIBILITY**

People are inherently resistant to changes and computer is known for facilitating changes. An estimate should be made of how strongly the user staffs react towards the developments of the computerized system.

In the proposed system the manpower is reduced so unnecessary burden is reduced .Therefore the system is behaviorally feasible.

**EXISTING SYSTEM**

Actually there is no specific application for contractor work seekers as well as contractors. Selecting and Selecting a Contractor One No One Publication There are job-giving websites but web pages for small jobs do not exist today.

**PROPOSED SYSTEM**

The proposed system is an integration of the website application. Communicates with the system using the contractor web application..The web application is a way for contractors to select a contractor based on the quality of their work and loyalty.

**SCOPE OF THE SYSTEM**

## The **'E-CONTRACTOR'** web application is a way for contractors to select a contractor based on the quality and reliability of their work, which can eliminate confusion. With this web application it is possible to enter jobs small and large as well as find job opportunities. When we find a job, we can then choose jobs in places near us

## **PURPOSE OF THE SYSTEM**

* Provide works to contractors
* easily find out contractors
* easily find out jobs

**SYSTEM REQUIREMENT SPECIFICATION**

**SYSTEM REQUIREMENTS**

## **SOFTWARE REQUIREMENTS**

For the proposed system to work properly, it is necessary that following software are installed and running on the server / client.

* IDE : JETBRAINS PYCHARM
* OS : Windows 7 or above / Linux New Versions
* Languages : Python
* DATABASE : Mysql, SQLyog
* Other Tools : Adobe Dream Weaver
* WEB BROWSER : GOOGLE CHROME

## **HARDWARE REQUIREMENTS**

It is recommended that for optimal performance, the following minimum hardware are installed,

* Processor : 1.9 gigahertz (GHz) x86- or x64-bit dual core processor with SSE2 instruction set
* Monitor : Min. 14
* RAM : 2 GB
* Display : Super VGA with a resolution of 1024 x 768
* Hard Disk : 160 GB,80 GB
* Keyboard : Standard 104 Keys
* Mouse : Serial mouse

**DATABASE DETAILS**

**DATABASE DETAILS**

Database design is the process of producing a detailed data model of a database. This logical 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. The term database design can be used to describe many different parts of the design of an overall database system. Principally, and most correctly, it can be thought of as the logical design of the base data structures used to store the data. In the relational model these are the tables and views. In an object database the entities and relationships map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the base data structures, but also the forms and queries used as part of the overall database application within the database management system. The process of doing database design generally consists of a number of steps which will be carried out by the database designer.

Usually, the designer must

* Determine the relationships between the different data elements
* Superimpose a logical structure upon the data on the basis of these relationships

The following factors are to be considered while designinga database

* Organize data into columns
* Decide the primary key
* Normalization.

**NORMALISATION**

Normalization is the process of decomposing a set of relations with anomalies to produce smaller and well structured relations that contain minimum redundancy. It is a formal process of deciding which attributes should be grouped together in a relation.

**First Normal Form**

First Normal form (INF) is now considered to be part of the formal definition of relational model. INF is designed to disallow multivalve attribute, composite attributes, and their combinations. It states that the domain of an attribute must include only atomic values. A domain is atomic, if elements of the domain are considered to be indivisible units. We say that a relational schema R is in INF if the domain of all attributes of R' is atomic.

**Second Normal Form**

Second Normal form (2NF) is based on the concept of functional dependency. A relation R is in 2NF if it is in INF and every non key attribute A of R is fully dependent on the primary key. That is, relation is said to be in 2NF if each attribute Ain R meets one of the following criteria:

(a) It appears in the primary key.

(b) It is fully functionally dependent on the primary key

**Third Normal Form**

Third Normal form (3NF) is based on the concept of transitive dependency. A relation is said to be in 3NF if it is in 2NF and has no transitive dependencies. That is all the non key attribute should be functionally determined by the primary key. In the proposed system all attributes of tables are fully depends on the primary key only that is all non key attributes are mutually independent.

**DATABASE TABLES**

**LOGIN**

|  |  |  |  |
| --- | --- | --- | --- |
| **COLUMN NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| loginid | INT | 11 | PRIMARY KEY |
| username | INT | 11 |  |
| password | VARCHAR | 50 |  |
| usertype | VARCHAR | 50 |  |

**CONTRACTOR**

|  |  |  |  |
| --- | --- | --- | --- |
| **COLUMN NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| cid | INT | 11 | PRIMARY KEY |
| loginid | INT | 11 | FOREIGN KEY |
| fname | VARCHAR | 50 |  |
| lname | VARCHAR | 50 |  |
| gender | VARCHAR | 20 |  |
| place | VARCHAR | 50 |  |
| Post | VARCHAR | 40 |  |
| Pin | INT | 20 |  |
| phone | BIGINT | 10 |  |
| service | VARCHAR | 50 |  |

**USERS**

|  |  |  |  |
| --- | --- | --- | --- |
| **COLUMN NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| userid | INT | 11 | PRIMARY KEY |
| fname | VARCHAR | 11 |  |
| lname | VARCHAR | 50 |  |
| place | VARCHAR | 50 |  |
| post | VARCHAR | 20 |  |
| pin | INT | 11 |  |
| Phonenumber | BIGINT | 10 |  |
| loginid | INT | 11 | FOREIGN KEY |

**FEATURES**

|  |  |  |  |
| --- | --- | --- | --- |
| **COLUMN NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| featureid | INT | 11 | PRIMARY KEY |
| contractid | INT | 11 | FOREIGN KEY |
| skills | VARCHAR | 50 |  |
| lname | VARCHAR | 50 |  |
| experiance | VARCHAR | 20 |  |
| work | VARCHAR | 50 |  |

**FEEDBACK**

|  |  |  |  |
| --- | --- | --- | --- |
| **COLUMN NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| feedid | INT | 11 | PRIMARY KEY |
| feedback | VARCHAR | 11 |  |
| userid | INT | 50 | FOREIGN KEY |
| date | DATE |  |  |
| cid | INT | 20 | FOREIGN KEY |

**LOCATION**

|  |  |  |  |
| --- | --- | --- | --- |
| **COLUMN NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| locationid | INT | 11 | PRIMARY KEY |
| userid | INT | 11 | FOREIGN KEY |
| type | VARCHAR | 50 |  |
| latitude | DOUBLE | 50 |  |
| longitude | DOUBLE | 20 |  |

**REQUEST**

|  |  |  |  |
| --- | --- | --- | --- |
| **COLUMN NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| requestid | INT | 11 | PRIMARY KEY |
| userid | INT | 11 | FOREIGN KEY |
| conid | INT | 50 | FOREIGN KEY |
| work | VARCHAR | 50 |  |
| date | DATE |  |  |
| status | VARCHAR | 50 |  |

**VACANCY**

|  |  |  |  |
| --- | --- | --- | --- |
| **COLUMN NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| vaccid | INT | 11 | PRIMARY KEY |
| job | VARCHAR | 11 |  |
| details | VARCHAR | 50 |  |
| contractid | INT | 50 | FOREIGN KEY |
| date | DATE |  |  |
| vacancy | VARCHAR | 50 |  |

**WORKS**

|  |  |  |  |
| --- | --- | --- | --- |
| **COLUMN NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| workid | INT | 11 | PRIMARY KEY |
| loginid | INT | 11 | FOREIGN KEY |
| work | VARCHAR | 50 |  |
| document | VARCHAR | 50 |  |
| status | DATE |  |  |

**WORKDETAILS**

|  |  |  |  |
| --- | --- | --- | --- |
| **COLUMN NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| wid | INT | 11 | PRIMARY KEY |
| workdetails | VARCHAR | 11 |  |
| file | VARCHAR | 50 |  |
| cid | INT | 50 | FOREIGN KEY |

**APPLYJOB**

|  |  |  |  |
| --- | --- | --- | --- |
| **COLUMN NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| applyid | INT | 11 | PRIMARY KEY |
| vaccancyid | INT | 11 | FOREIGN KEY |
| userid | INT | 50 | FOREIGN KEY |
| date | DATE |  |  |
| status | VARCHAR | 100 |  |
| resume | VARCHAR | 100 |  |

**CHAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **COLUMN NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| chatid | INT | 11 | PRIMARY KEY |
| fromid | INT | 11 |  |
| toid | INT | 11 |  |
| message | VARCHAR | 50 |  |
| date | DATE |  |  |

**COMPLAINT**

|  |  |  |  |
| --- | --- | --- | --- |
| **COLUMN NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| complaintid | INT | 11 | PRIMARY KEY |
| complaint | VARCHAR | 11 |  |
| date | DATE |  |  |
| userid | INT | 50 | FOREIGN KEY |
| replay | VARCHAR | 50 |  |
| conid | INT | 50 | FOREIGN KEY |

**SYSTEM DESIGN**

**SYSTEM DESIGN**

**USERS OF THE SYSTEM**

1. Admin
2. Contractor
3. Users

### **Functions of Admin**

* View contractor
* View User
* Feedback view
* View complaints and replays
* Block users and contractors

Admin can manage the details of contractor and user that are registered. Admin add and manage the details of contractor and also admin view the details of users. Admin can login using username and password.

### **Functions of Contractor**

* Signup
* Login
* Add/manage job vacancy
* View job request
* Add features and details
* Communication platform
* View work
* Work request
* View feedback

**Functions of User**

* Add work and send work request
* View contractor work
* View work request status
* Sent complaints
* View job vacancy and apply job
* View and send feedback
* View replay
* Chat contractor

**Architecture Design**

Architecture is an overall structure of a system. Architecture takes into consideration the overall working of the system, Large system can be decomposed into sub-systems that provide some related set of services. The initial design process of identifying these sub-systems and establishing a framework for sub-systems control and communication is called architecture design.

Architecture design usually comes before detailed system specification. Architecture decomposition is necessary to structure and organize the specification. There is no generally accepted process depends on application knowledge and on the skill and intuition of the system architecture.

**ENTITY RELATIONSHIP DIAGRAM**

An ER diagram can express the overall structure of the database graphically. ER diagrams are simple and clear. ER diagram often use symbols to represent three different types of information. Boxes are commonly used to represent entities. Diamonds are normally used to represent relationships and ovals are used to represent attributes.

**Entity Relationship Diagram Notations**

**Entity:**

An entity is an object or concept about which you want to store information

**Entity**

**Attribute:**

Each entity has attribute, or particular properties that describe the entity.

**Attribute**

**Key Attribute:**

A key attribute is the unique, distinguishing characteristics of the entity.

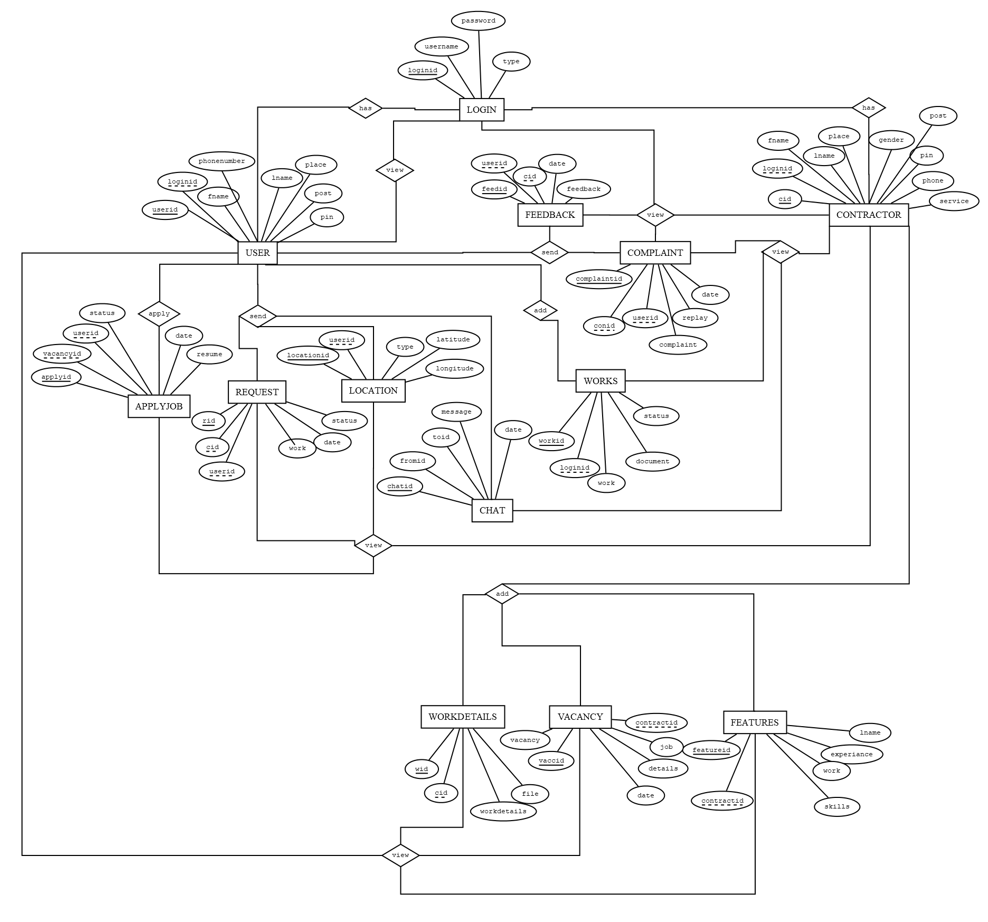
**Key Attribute**

**Relationships:**

Relationships illustrate how two entities share information in the database structure.

**Relationship**

**ER Diagram:**

****

**DATA FLOW DIAGRAM**

**DFD DESIGN NOTATIONS**

Data Flow Diagram is the graphical description of the system's data and how the processes transform the data. Data Flow Diagram depicts information flow, and the transforms that are applied as data move from the input to output. It is the starting point of the design phase that functionally decomposes the requirement specifications down to the lowest level of details. Thus a DFD describes what data flows (logical) rather than how they are processed. Unlike detailed flowchart, Data Flow Diagrams don't supply detailed description of the modules but graphically describes a system's data and how the data interacts with the system.

In DFD, there are four main symbols:

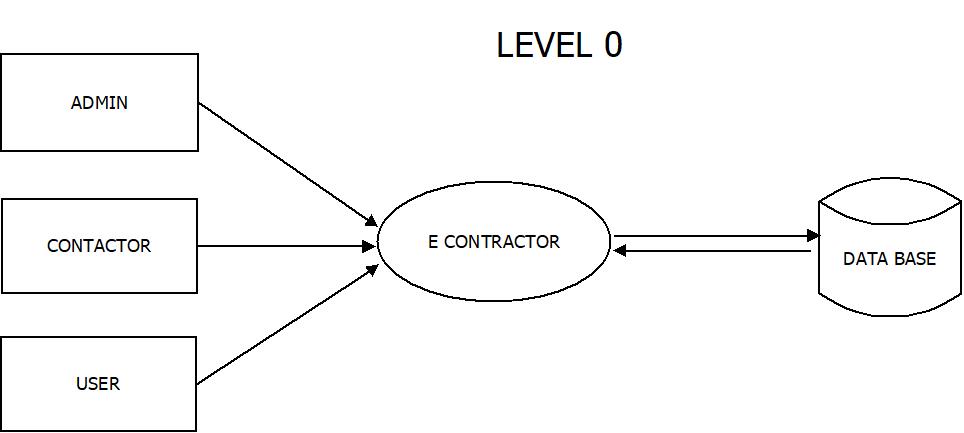
* Source or Destination of data

* Flow of Data

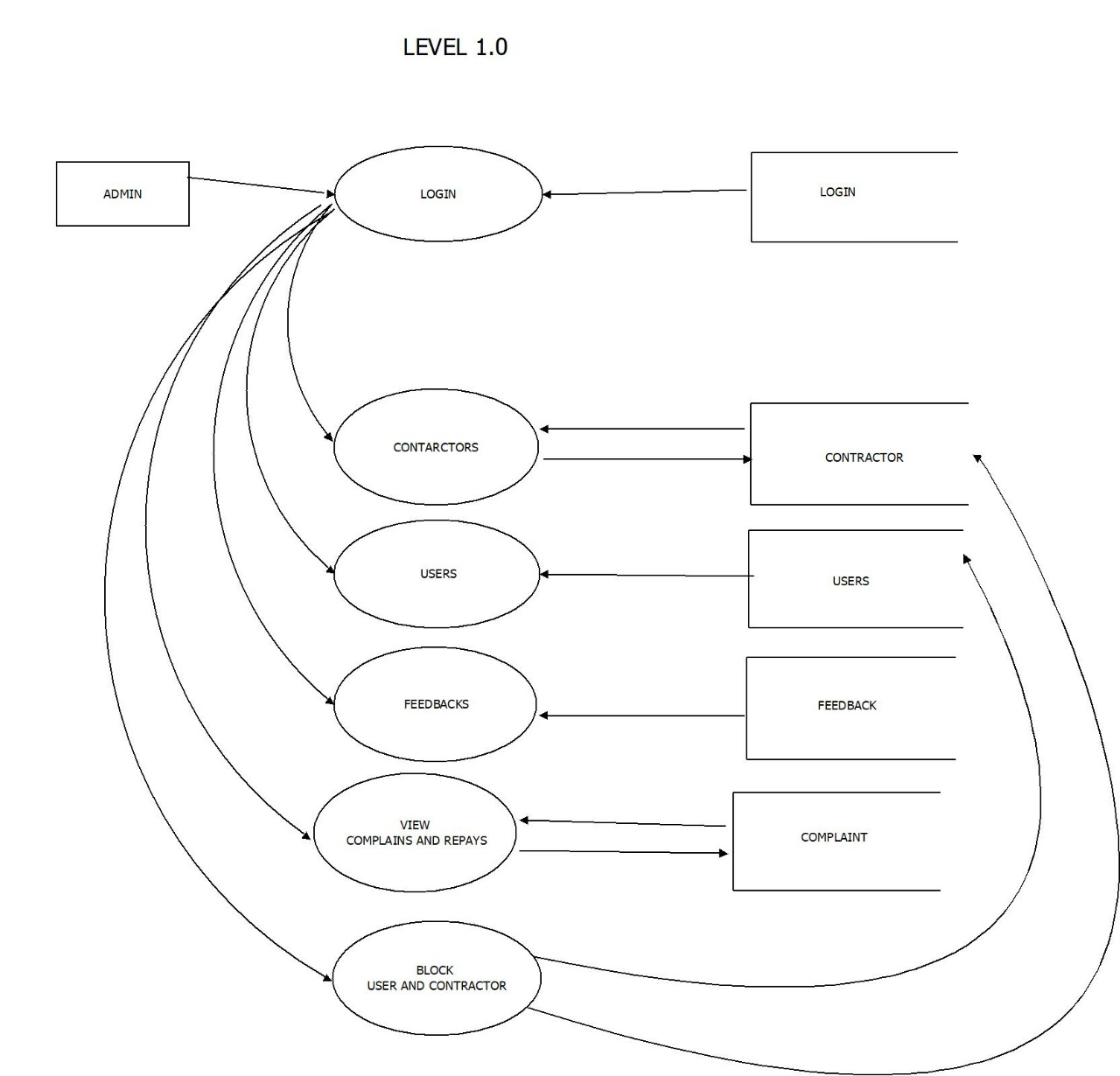
-Process transforming data

-Temporary repository of data

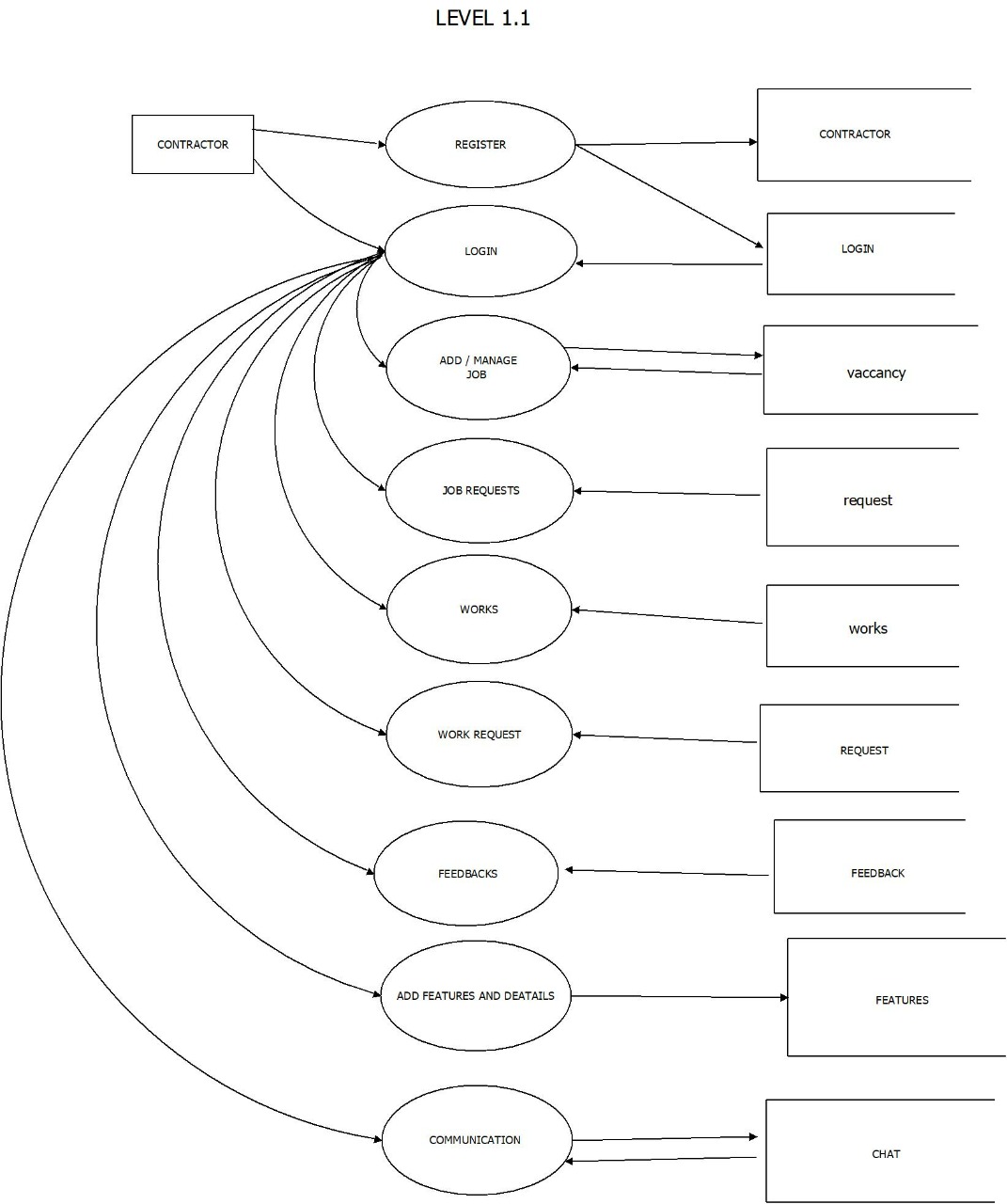
**LEVEL: 0**



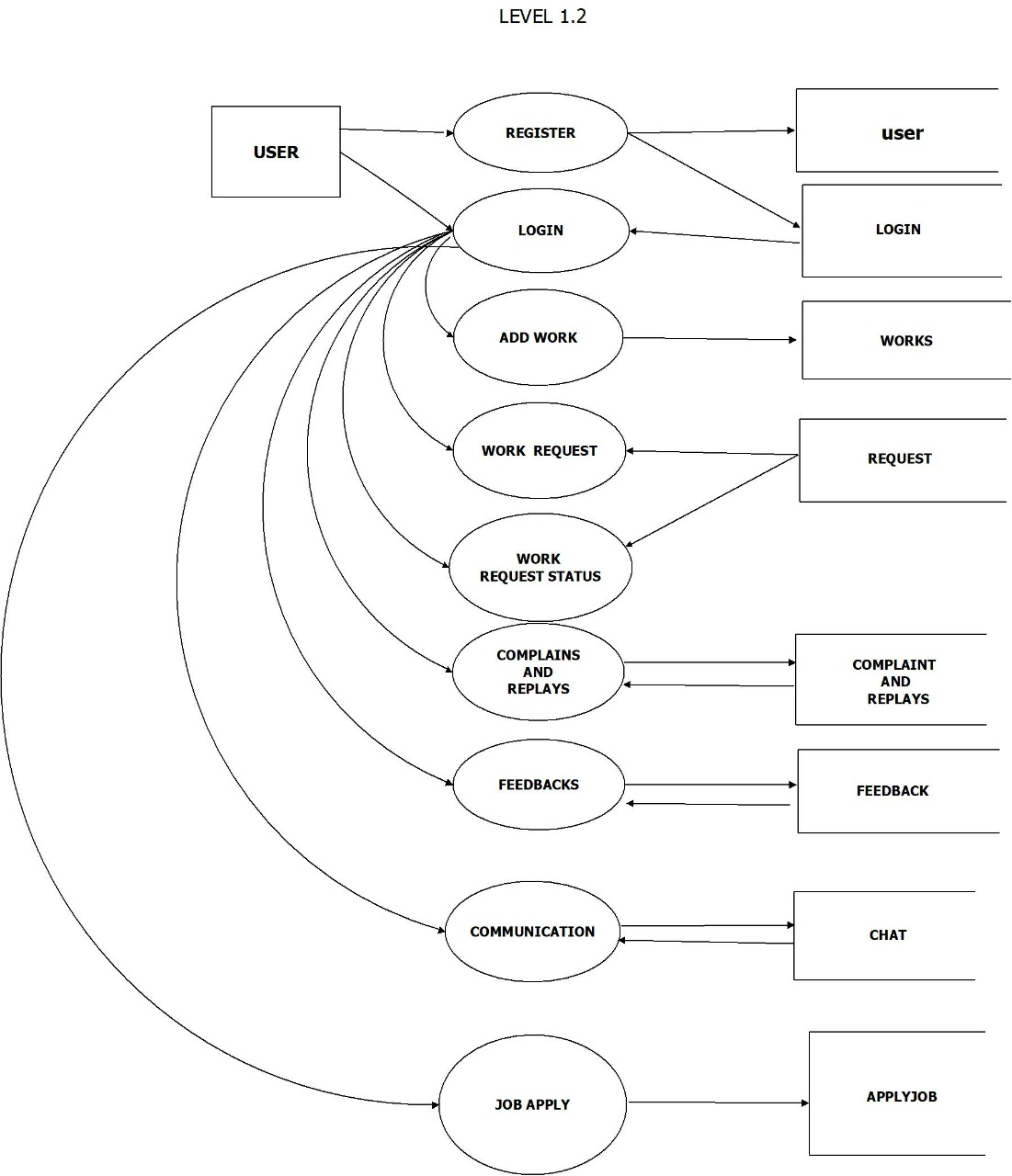
**LEVEL: 1**

****

**LEVEL: 2**



**LEVEL :3**



**INPUT DESIGN**

Input design is the link that ties the information system into the world of its user. Inaccurate input data are the most common cause of errors in data processing. Error entered by data entry operators can be controlled by input design. It is the process of converting user originated input to a computer based format. The input design involves determining what the inputs are, how the data should be performed, how to validate data, how to minimize data entry and how to provide a multi-user facility. The input data are collected and organized into groups of similar data. In input data design, the designing of source document that capture the data and then selected the media used to enter them into the computers. The source document may be entered into the system through punch cards, dise or even directly from keyboard.

**OUTPUT DESIGN**

Computer output is the most important and direct source of implementation of the user. Efficient intelligible, output design should improve the system's relationship with the user and help in decision making It is very important phase because the output needs to be in an attractive manner. The success or failure of software is decided by the integrity and correctness output that is produced from the system. One of the objectives behind the automation of business system itself is the fast and prompt generation of reports in a short period of time. In today's competitive world of business it is very important for companies to keep themselves up to date about the happening in the business. Prompt and reliable reports are considered to be the lifetime of every business today So the output report generated by the software system is of paramount importance.

**OVERVIEW OF THE SOFTWARE**

**TECHNOLOGY**

**FRONT END SOFTWARE**

**HTML, CSS, JavaScript and Bootstrap**

HTML provides the basic structure of sites, which is enhanced and modified by other technologies like CSS and JavaScript. CSS is used to control presentation, formatting, and layout.JavaScript is used to control the behavior of different elements.

HTML stands for HyperText Markup Language. "Markup language" means that, rather than using a programming language to perform functions, HTML uses tags to identify different types of content and the purposes they each serve to the webpage.

Quickly design and customize responsive mobile-first sites with Bootstrap, the world’s most popular front-end open source toolkit, featuring Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful JavaScript plugins.Bootstrap is the most popular CSS Framework for developing responsive and mobile-first websites. Bootstrap 5 is the newest version of Bootstrap. Bootstrap is the most popular HTML, CSS and JavaScript framework for developing a responsive and mobile friendly website.

**HTML**

Markup languages work in the same way as you just did when you labeled those content types, except they use code to do it -- specifically, they use HTML tags, also known as "elements." These tags have pretty intuitive names: Header tags, paragraph tags, image tags, and so on.Every web page is made up of a bunch of these HTML tags denoting each type of content on the page. Each type of content on the page is "wrapped" in, i.e. surrounded by, HTML tags

**CSS**

CSS stands for Cascading Style Sheets. This programming language dictates how the HTML elements of a website should actually appear on the frontend of the page.HTML provides the raw tools needed to structure content on a website. CSS, on the other hand, helps to style this content so it appears to the user the way it was intended to be seen. These languages are kept separate to ensure websites are built correctly before they're reformatted.

**JAVASCRIPT**

JavaScript is a more complicated language than HTML or CSS, and it wasn't released in beta form until 1995. Nowadays, JavaScript is supported by all modern web browsers and is used on almost every site on the web for more powerful and complex functionality.JavaScript is a logic-based programming language that can be used to modify website content and make it behave in different ways in response to a user's actions. Common uses for JavaScript include confirmation boxes, calls-to-action, and adding new identities to existing information.

**BOOTSTRAP**

Quickly design and customize responsive mobile-first sites with Bootstrap, the world’s most popular front-end open source toolkit, featuring Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful JavaScript plugins.Bootstrap is the most popular CSS Framework for developing responsive and mobile-first websites.

Bootstrap 5 is the newest version of Bootstrap

Bootstrap is the most popular HTML, CSS and JavaScript framework for developing a responsive and mobile friendly website.

* It is absolutely free to download and use.
* It is a front-end framework used for easier and faster web development.
* It includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many others.
* It can also use JavaScript plug-ins.
* It facilitates you to create responsive designs.

**Why use Bootstrap**

Following are the main advantage of Bootstrap:

* It is very easy to use. Anybody having basic knowledge of HTML and CSS can use Bootstrap.
* It facilitates users to develop a responsive website.
* It is compatible on most of browsers like Chrome, Firefox, Internet Explorer, Safari and Opera etc.

A website is called responsive website which can automatically adjust itself to look good on all devices, from smart phones to desktops etc.

**BACK-END SOFTWARE**

**PyCharm**

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for Python language.

* It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes) and supports web development with flask
* Coding assistance and analysis, with code completion, syntax and error highlighting, linter integration, and quick fixes
* Project and code navigation: specialized project views, file structure views and quick jumping between files, classes, methods and usages

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming Python is often described as a "batteries included" language due to its comprehensive standard library

**DATABASE**

**MySQL**

MySQL is free and open-source software under the terms of the GNU General Public License, and is also available under a variety of proprietary licenses. MySQL was owned and sponsored by the Swedish company MySQL AB, which was bought by Sun Micro systems now Oracle Corporation). In 2010, when Oracle acquired Sun, Widenius forked the open-source MYSQL project to create MariaDB. MYSQL is a component of the LAMP web application software stack(and others), which is an acronym for Linux, Apache, MySQL, Perl/PHP/Python. MySQL is used by many database-driven web applications including Drupal, Joomla, phpBB and WordPress.

MySQL is also used by many popular websites, including Google (though not for searches),

Facebook, Twitter, Flickr, and YouTube.

Cross-platform support

* Stored procedures, using a procedural language that closely adheres to SQL/PSM
* Triggers
* Cursors
* Updatable views
* Online Data Definition Language (DDL) when using the In no DB Storage Engine
* Information schema
* Performance Schema that collects and aggregates statistics about server execution and query performance for monitoring purposes.
* Unicode support

**Backup software**

Mysql dump is a logical backup tool included with both community and enterprise editions of MySQL. It supports backing up from all storage engines. MYSQL Enterprise Backup is a hot backup utility included as part of the MySQL Enterprise subscription from Oracle, offering native InnoDB hot backup, as well as backup for other storage engines.

**High availability software**

Oracle MySQL offers nowadays an high availability solution with a mix of tools including the MySQL router and the MySQL Shell all based on Group Replication, open source tools.

**Cloud deployment**

Main article : Cloud database

MYSQL can also be run on cloud computing platforms such as Microsoft Azure Amazon EC2, Oracle Cloud Infrastructure.

**SYSTEM IMPLEMENTATION**

**SYSTEM IMPLEMENTATION**

Implementation phase is the phase, which involves the process of converting a new system design into an operational one. It is the key stage in achieving a successful new system. Implementation is the stage of the project, where the theoretical design is turned into a working system. At this stage, the workload, the greatest up heal and major impact on controlled carefully,

It can cause chaos. The implementation stage is a system project in its own right. It involves careful planning, investigation of the current system and its constraints on the implementation, design methods the implementation plan consists of the following steps:

* Testing the developed system within the sample data
* Detection and correctness of errors
* Making necessary changes in system
* Training and involvement of user personnel
* Installation of software utilities

**IMPLEMENTATION PROCEDURES**

The implementation phase is less creative than system design. A system project may be dropped at any time prior to the implementation although it becomes more difficult when it goes to the design phase. The final report to the implementation phase includes procedural charts, record layout and a workable plan for implementing the candidate system. Implementation is used to the process of converting a new or revised system design into an operational one. Conversation is one aspect of implementation. Several procedure are unique to the implementation phase Conversation begins with a review of the project plan, the system test documentation, and implementation plan.

**SYSTEM TESTING**

**SYSTEM TESTING**

System testing is actually a series of different testing's whose primary purpose is to fully exercise the computer based system. Software testing is critical element of software quality assurance and represents the ultimate review of the specification, design and coding. System testing makes a logical assumption that all the part of the system is correct the goal will be successfully achieved. Testing is the final verification and validation activity within the organization itself. During testing, the major activities are concerned on the examinations and modifications of the source code.

Testing is a process of executing a program with the intend of finding an error. A good test is one that uncovers an as yet undiscovered error

Testing Objectives are:

* Unit Testin
* Integration Testing
* Validation Testing
* Output Testing

**UNIT TESTING**

The first level of testing is called as unit testing, here the different modules are tested and the specification produced during design for the modules. Unit testing is essential for verification of the goal and to test the internal logic of the modules. Unit testing is conducted to different modules of the project. Errors were noted down and corrected down immediately and the program clarity was increased. The testing was carried out during the programming stage itself carried out during the programming stage itself. In this step each modules found to be working satisfactory us regard to be expected out from the module

For example, the Login page is tested against three different states that positive input, a negative input and a 0 input. Testing with a positive input, and with a negative input will behaves expected. Testing with a 0 input however will yield a surprise. This is just one example of why it makes sense to are son testing the different states of your code.

**INTEGRATIONTESTING**

The second level of testing includes integration testing. It is systematic testing of constructing structure. At the same time tests are conducted to uncover errors with the interface. It need not to be the case that software whose modules when run individually showing results will also show perfect results when run as a whole.

The individual modules are tested again and the results are verified. The goal is to see if the modules integrated between the modules. This testing activity can be considered as testing the design and emphasizes on testing modules interaction.

**VALIDATION TESTING**

The next level of testing is validation testing. Here the entire software is tested. The reference document for this process is the requirement and the goal is to see if the software meets its requirements.

The requirement document reflects and determines whether the software functions as the user expected. At culmination of integration testing software completely assembled as package and corrected and a final series of software test validation test begins. The proposed system under construction has been tested by using validation testing and found to be working satisfactory.

Data validation checking is done to see whether the corresponding entries made in different tables are done correctly. Proper validation checks are done in case of insertion and updating of tables, in order to see that no duplication of data has occurred. If any such case arises proper warning message will be displayed. Double configuration is done before the administrator deletes a data in order to get positive results and to see that 0 data have been deleted by accident

**OUTPUTTESTING**

The output of the software should be acceptable to the system user. The output of requirement is defined during the system analysis. Testing of the software system is done against the output and the output testing was completed with success

**USER ACCEPTANCE TESTING**

An acceptance test has the objective of selling the user on the validity and reliability of the system. It verifies that the system procedures operate to system specification and the integrity of the vital data is maintained.

**SYSTEM MAINTENANCE**

**SYSTEM MAINTENANCE**

Maintenance corresponds to restoring something to original condition covering a wide range of activity, including correcting coding, design errors, updating user support Better the system design, easier to maintain the system. Maintenance is performed most often to improve the existing software rather than to respond to a crisis or system failure According to user needs and operational environment change maintenance should be done in parallel, otherwise the system could fail. Provision must be made for environmental changes, which may affect either the computer or other parts of a computer-based system such as activity are normally called maintenance. It includes both improvement of system functions and the correction of fungus that arise during the operation of a system Maintenance activity may require the continuing involvement of a large proportion of computer department resources. Most changes arise in two ways.

As part of the normal running of the system when errors are found, users ask for improvement of external requirements change. As the results of specific investigation and review of the systems performance, maintenance involves the software industry captive. typing of system resources. It means restoring something to its original condition Maintenance was done after the successful implementation. Maintenance is continued till the product is re-engineered or deployed to another platform. Maintenance is also done based on fixing the problem reported, changing the interface with other software or hardware. enhancing the software. Any system developed should be seeurod and protected against possible hazards. Security measures are provided to prevent thorized access of the database at various levels: An uninterrupted power supply should be provided so that power failure or voltage fluctuations will not erase the data in the files.

**FUTURE ENHANCEMENT**

**FUTURE ENHANCEMENT**

As future enhancement **“EContractor”** It intends to launch another mobile app in the near future. This will make it easier for contractors to select, add more information and make it easier for users.

**CONCLUSION**

**CONCLUSION**

As per the requirement specified proposed system is successfully implemented and tested it has been a great pleasure for me to work on this existing and challenging project. This project proved good for me as it provided practical knowledge of programming in and python. The Project achieved entire goal the practice of “E-CONTRACT” is to help to select a contractor based on the quality of their work and loyalty.

**BIBLIOGRAPHY**

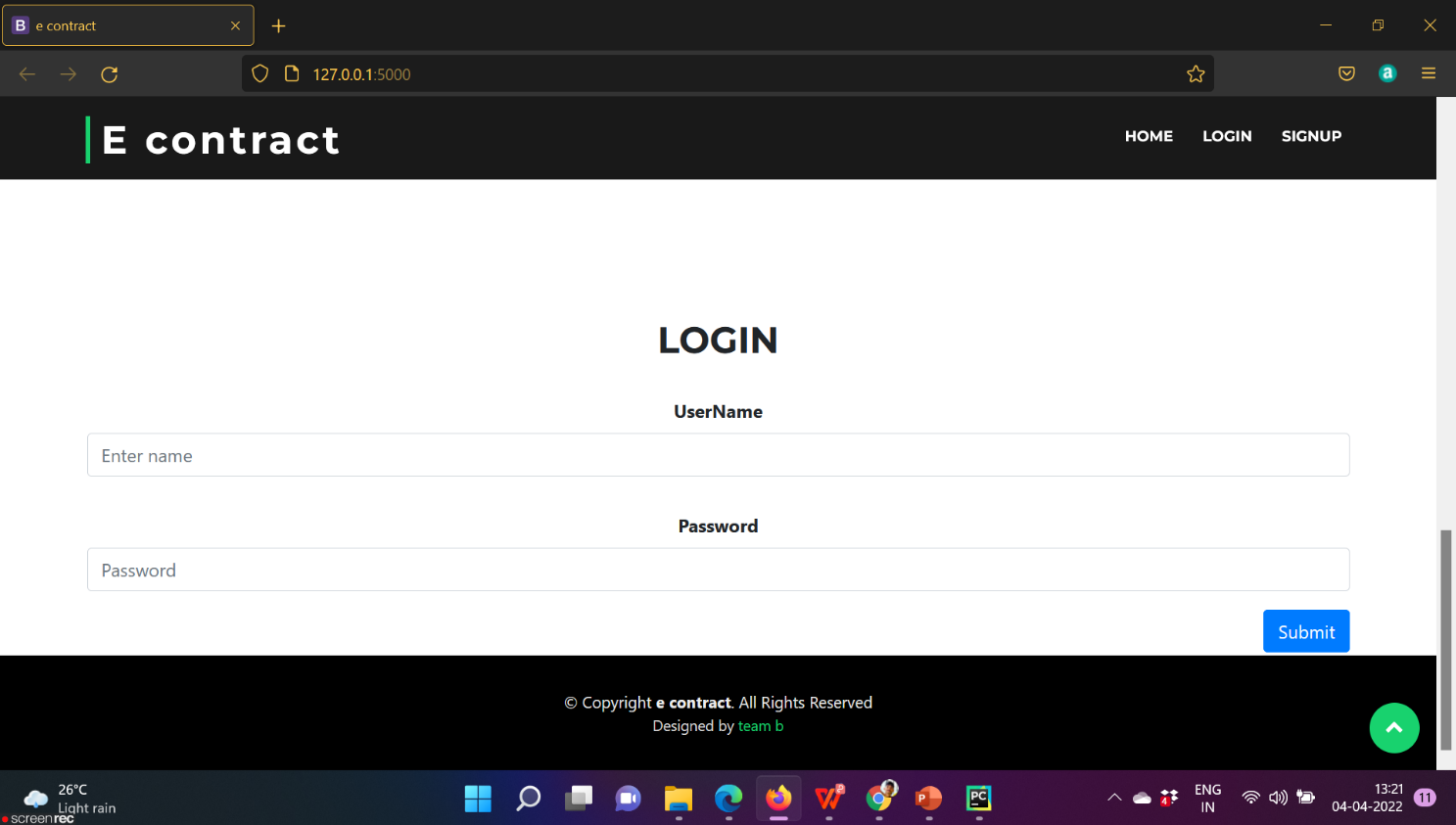
**BIBLIOGRAPHY**

**ONLINE RERERENCE**

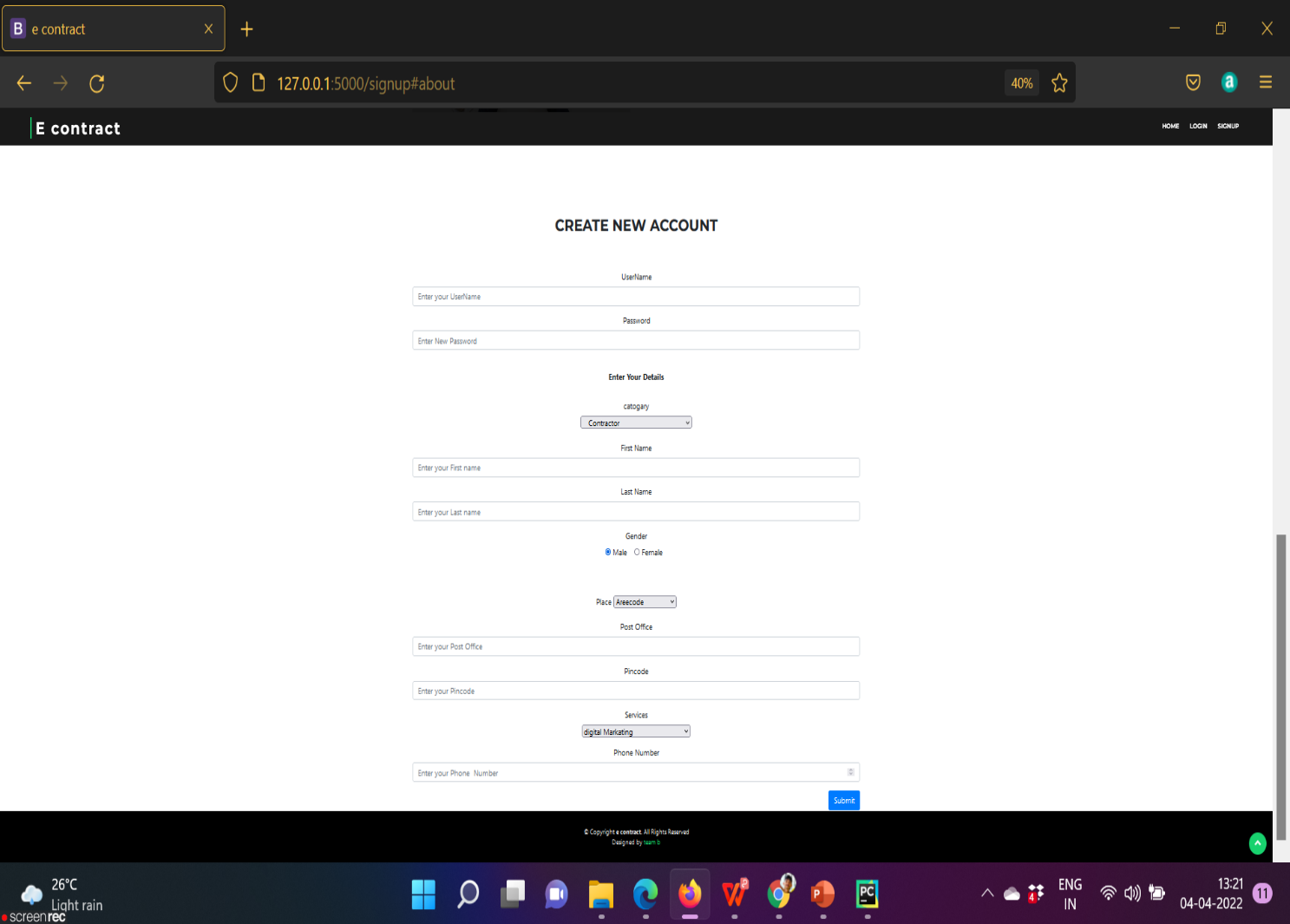
1. [**https://www.w3schools.com/**](https://www.w3schools.com/)
2. [**https://getbootstrap.com**](https://getbootstrap.com)
3. [**https://www.javascript.com**](https://www.javascript.com)
4. [**https://www.w3.org**](https://www.w3.org)

**SCREENSHOT**

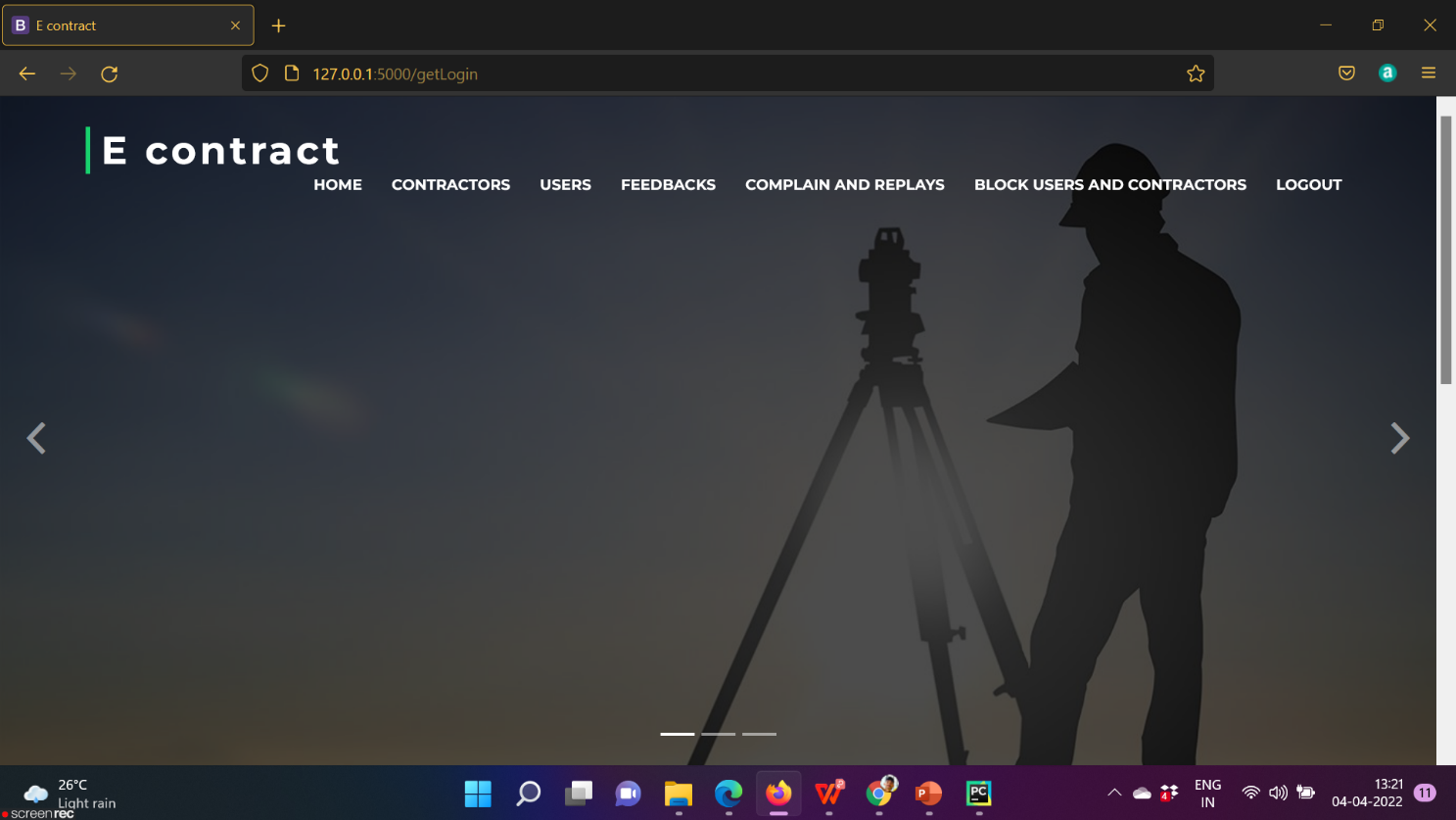
**LOGIN**

****

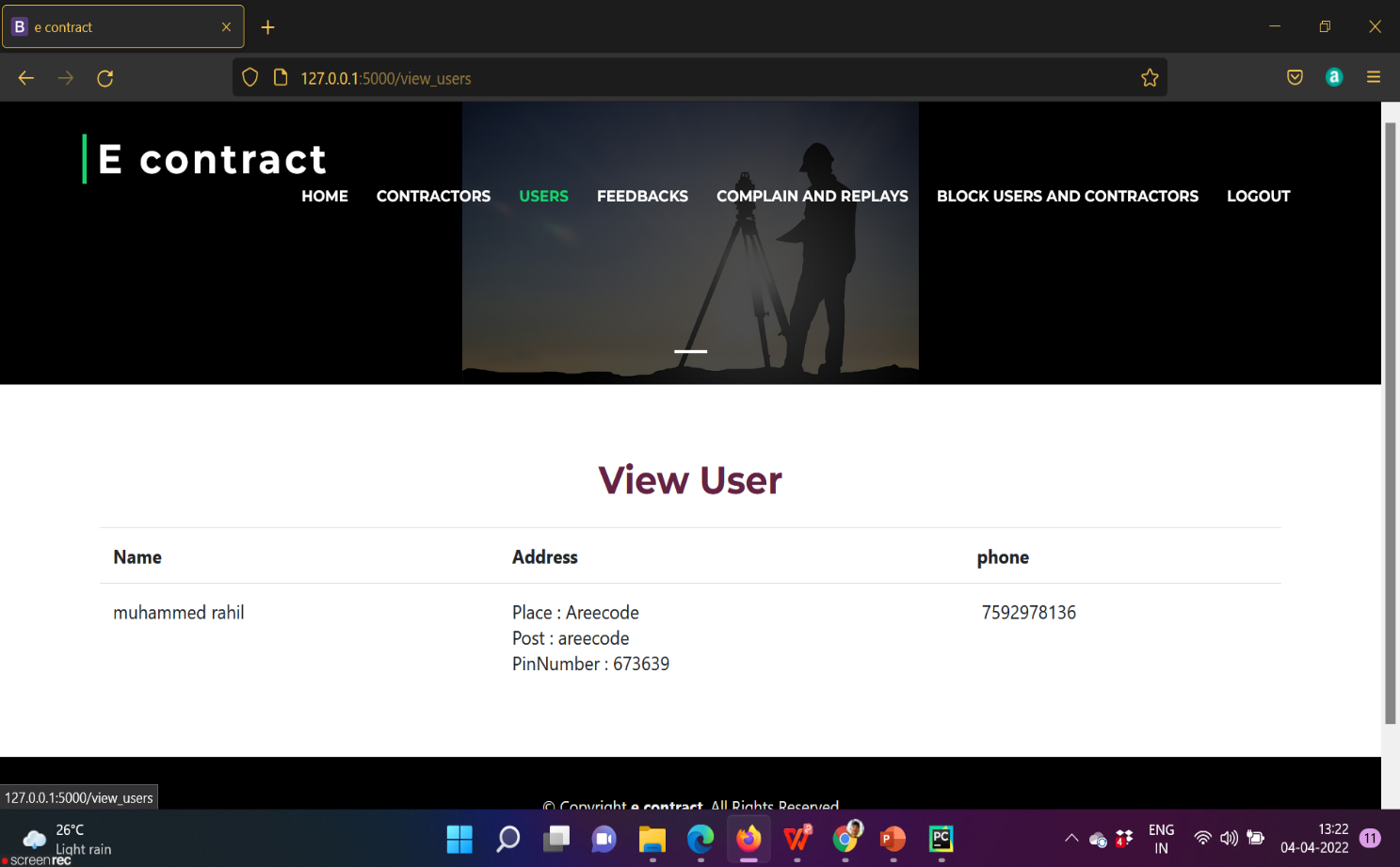
**SIGN UP**

****

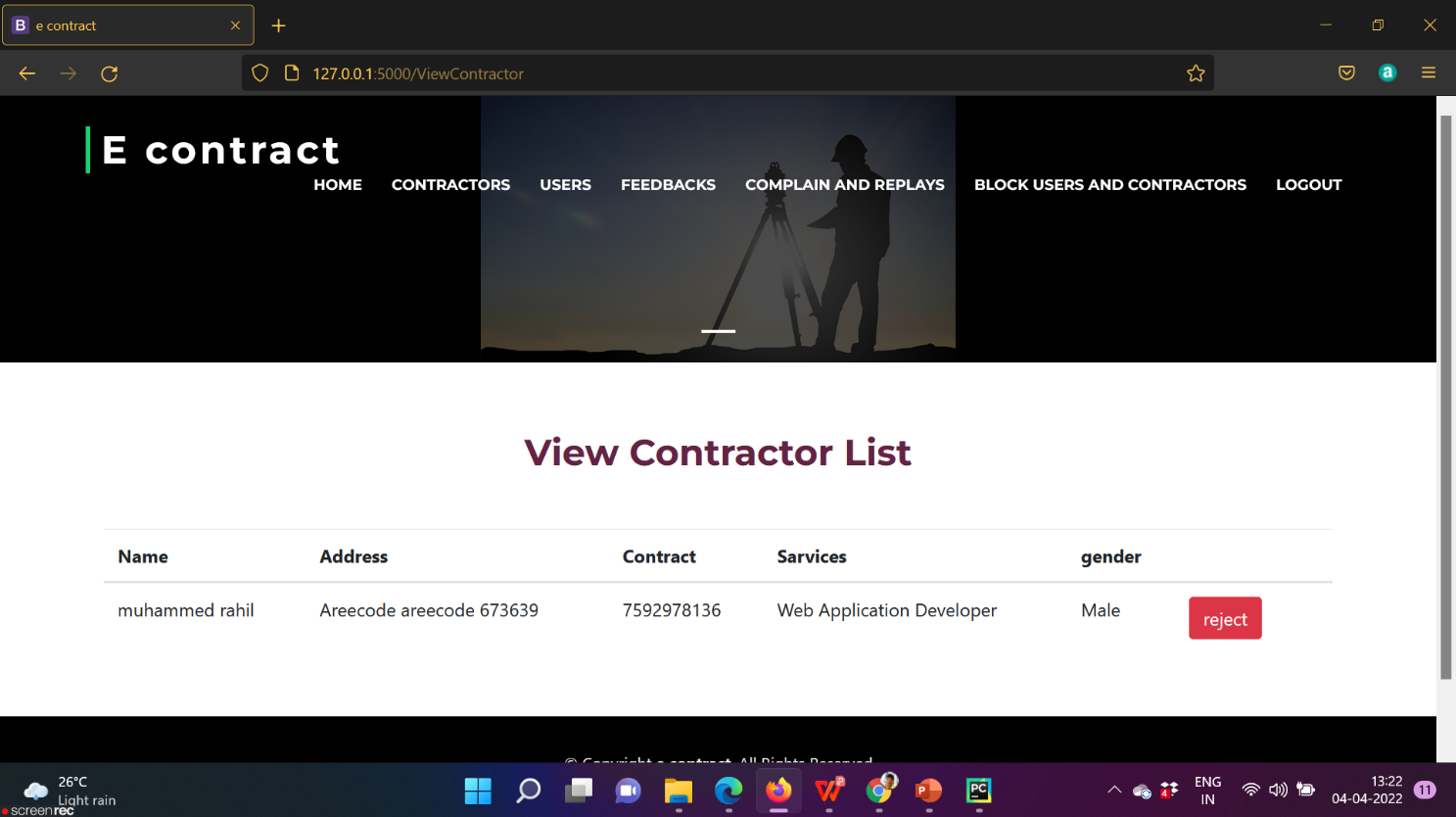
**ADMIN HOME**

****

**VIEW USER**

****

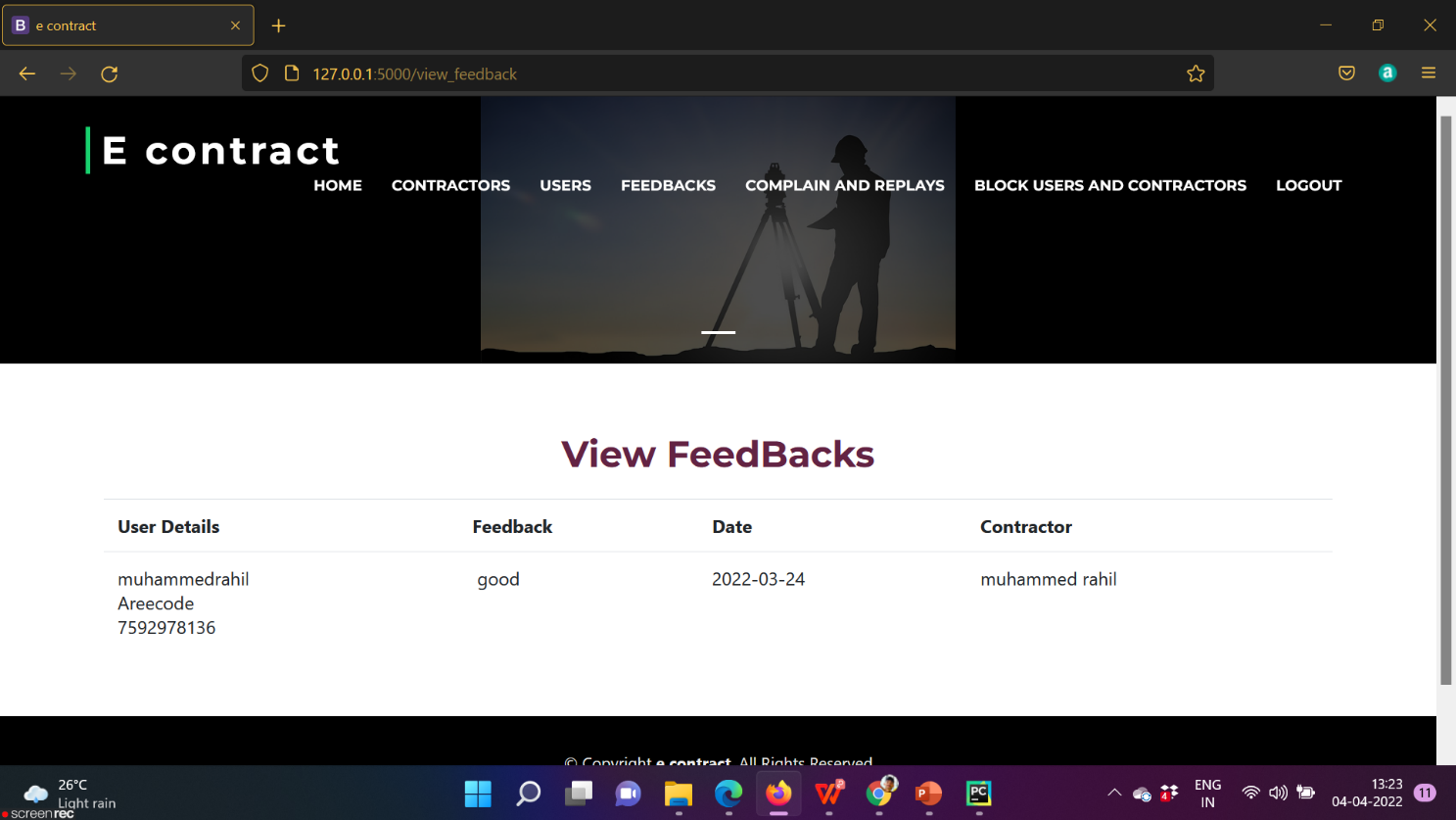
**VIEW CONTRACTOR**

****

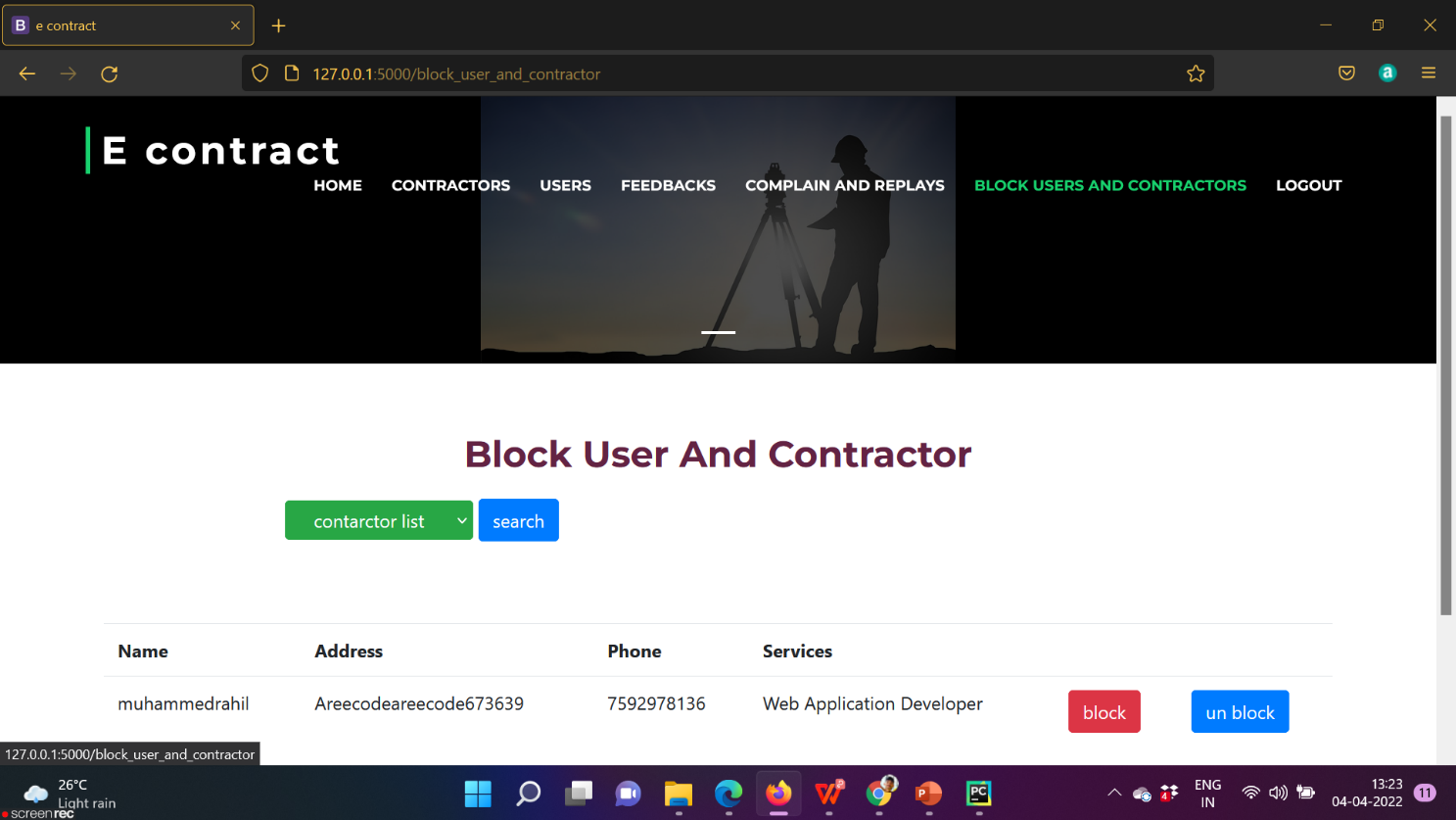
**VIEW COMPLAINTS**

****

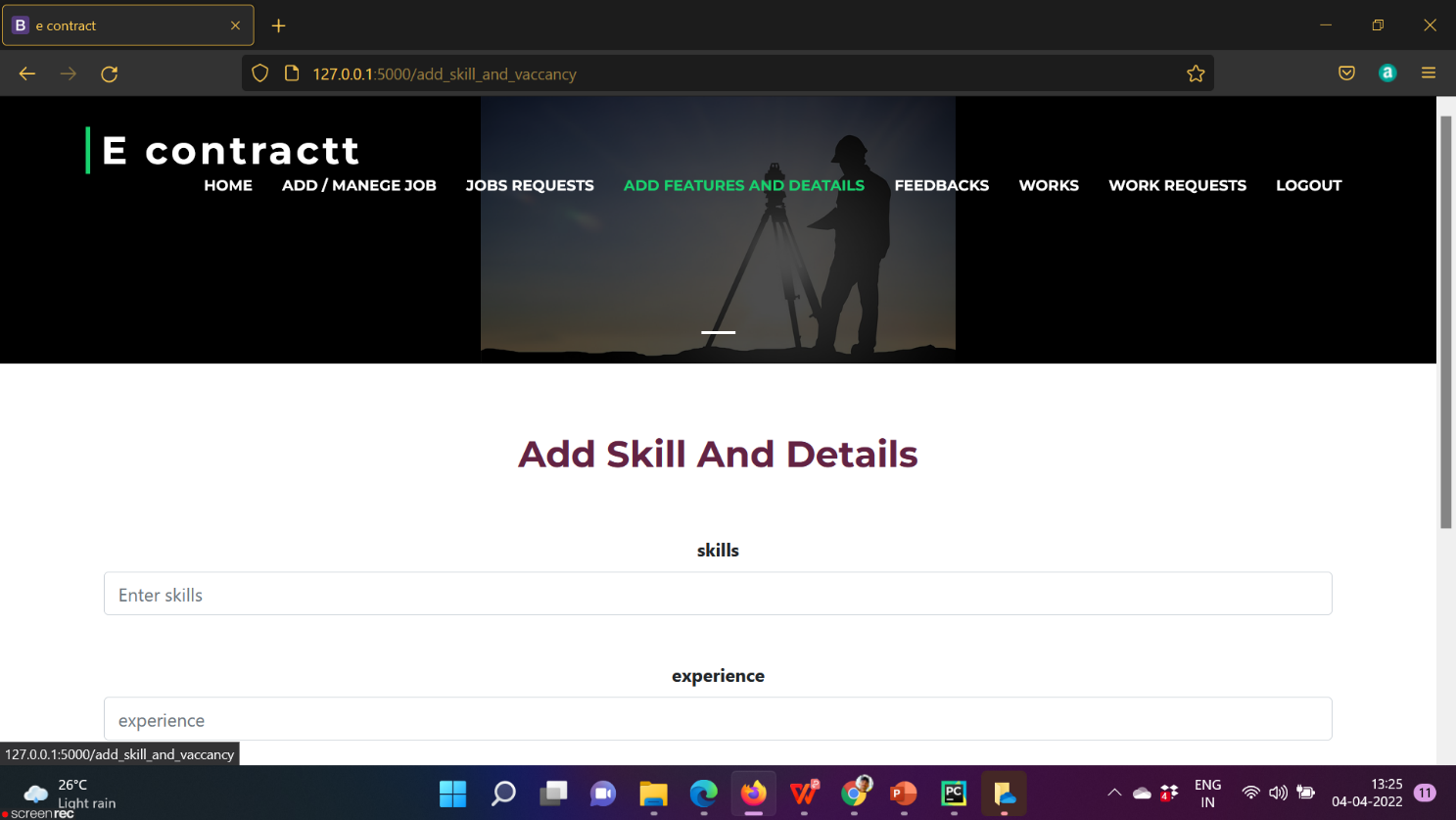
**VIEW FEEDBACK**

****

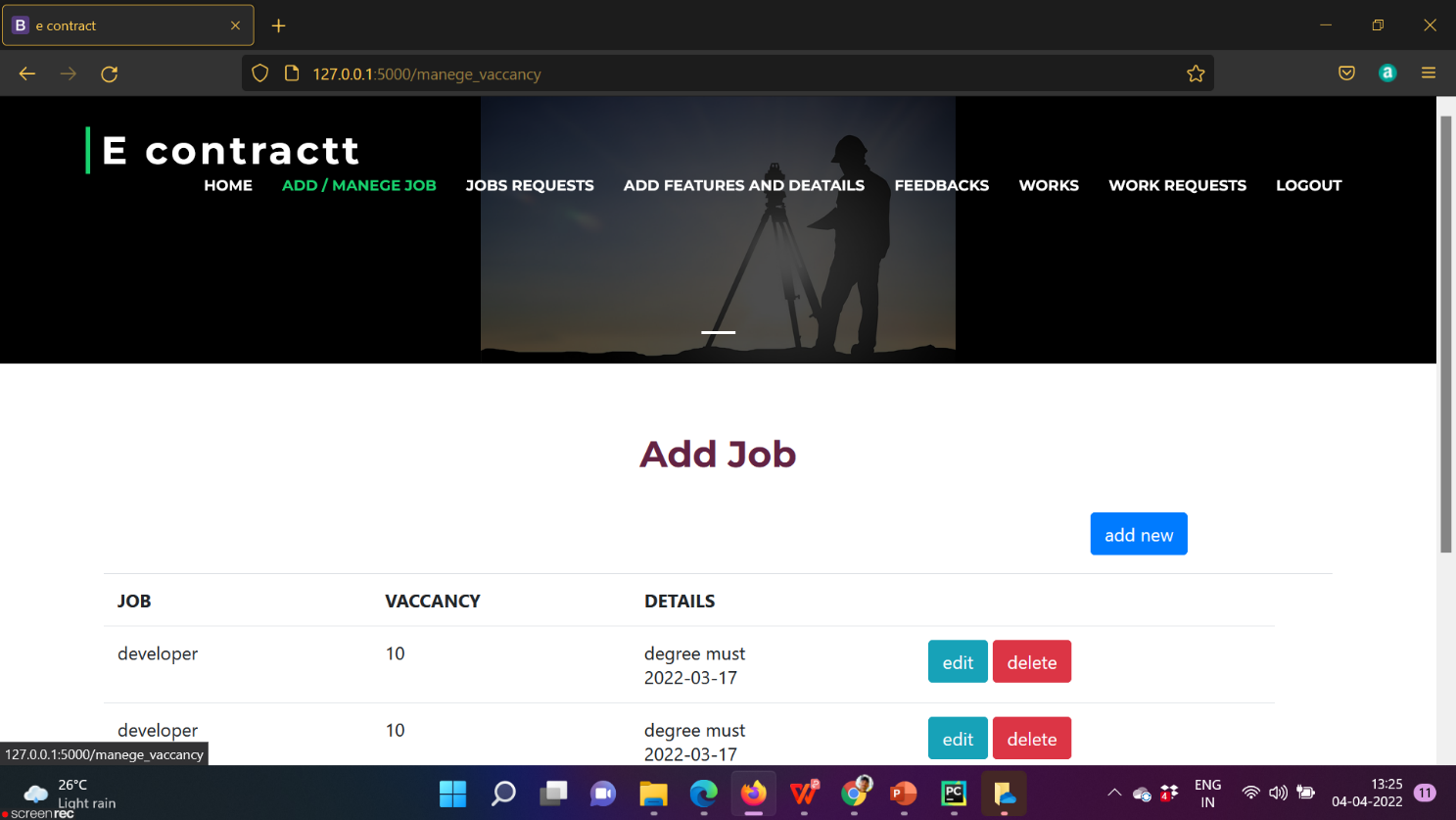
**BLOCK USER AND CONTRACTOR**

****

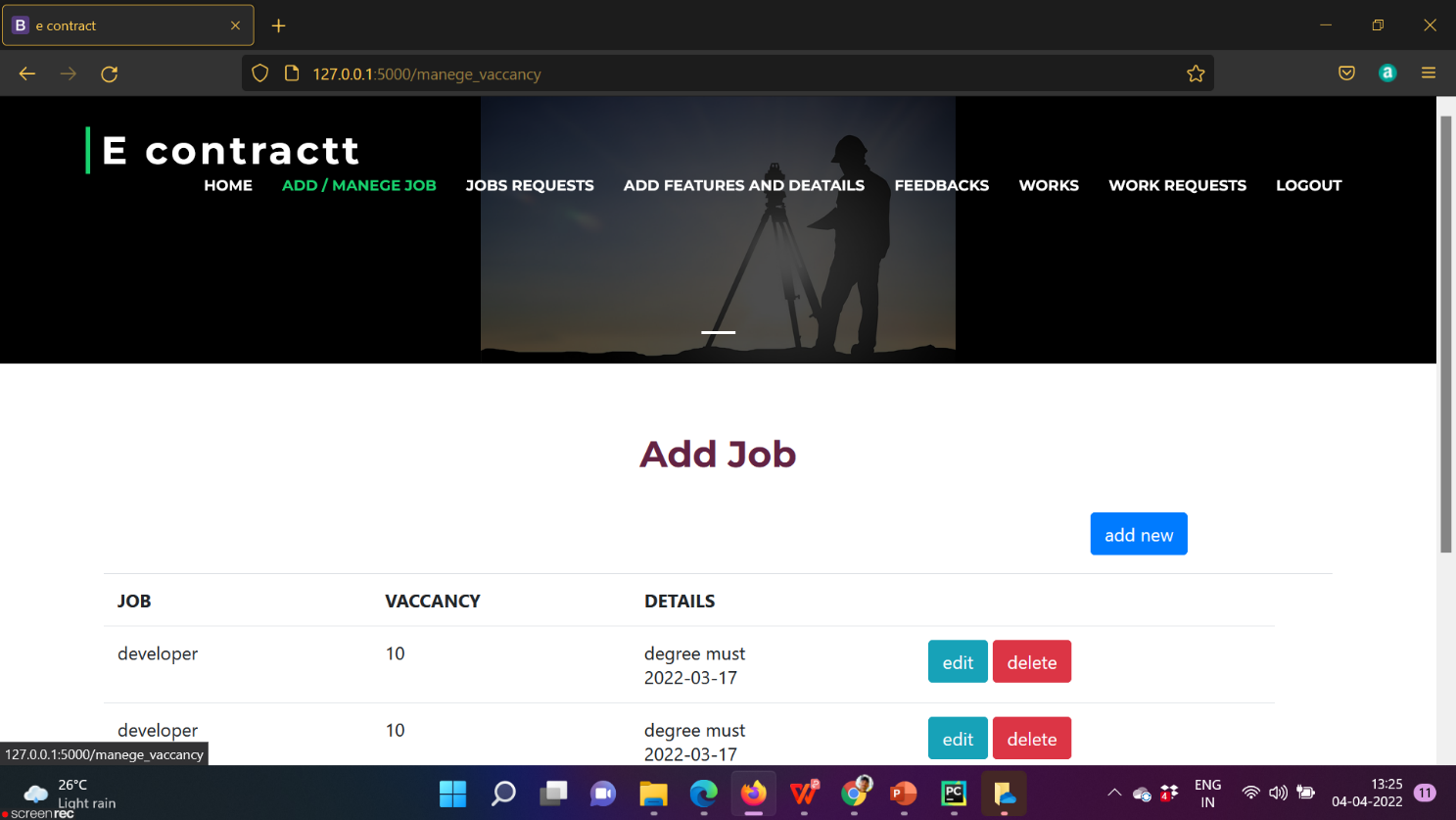
**ADD SKILLS AND DETAILS**

****

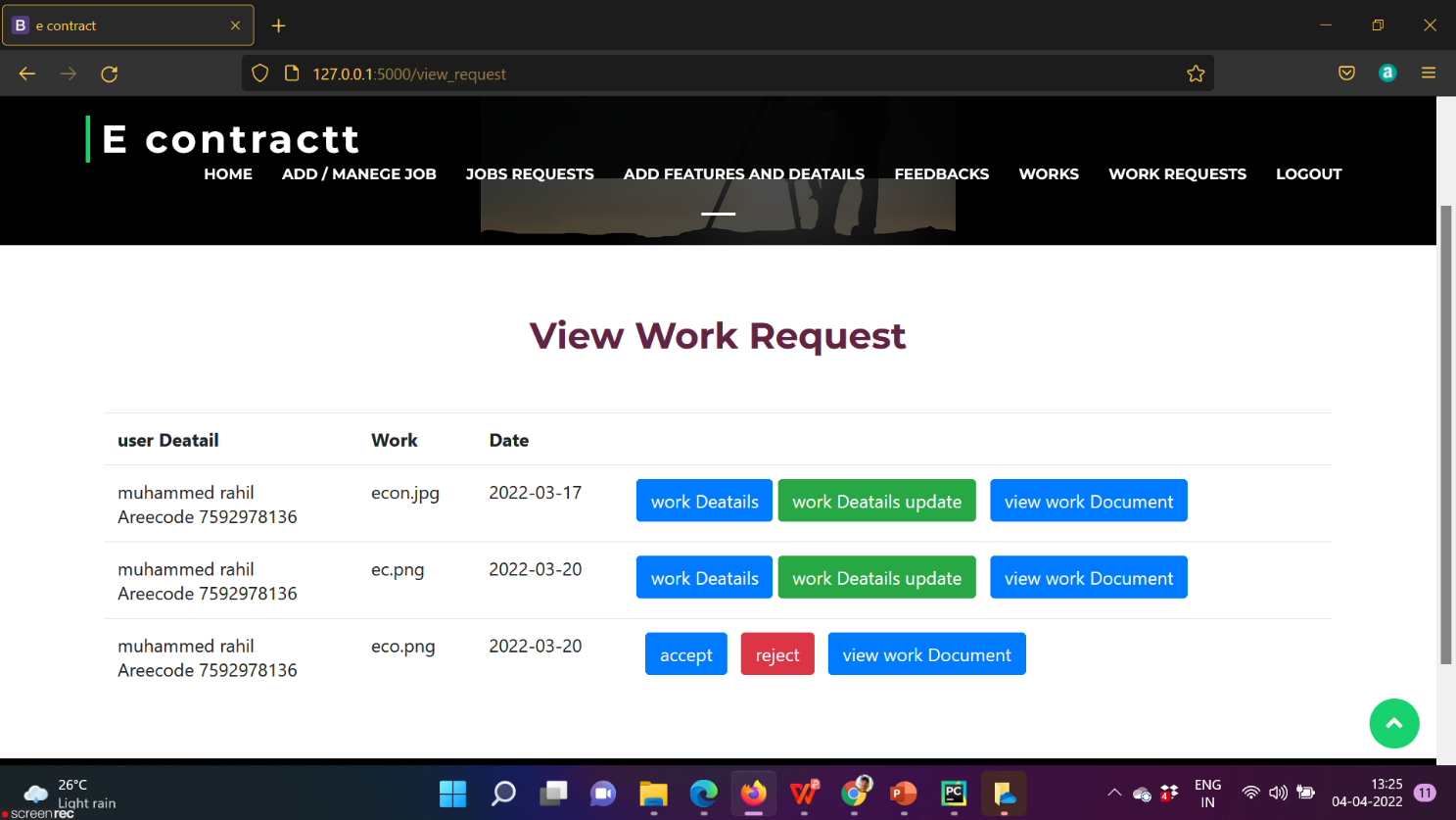
**ADD JOBS**

****

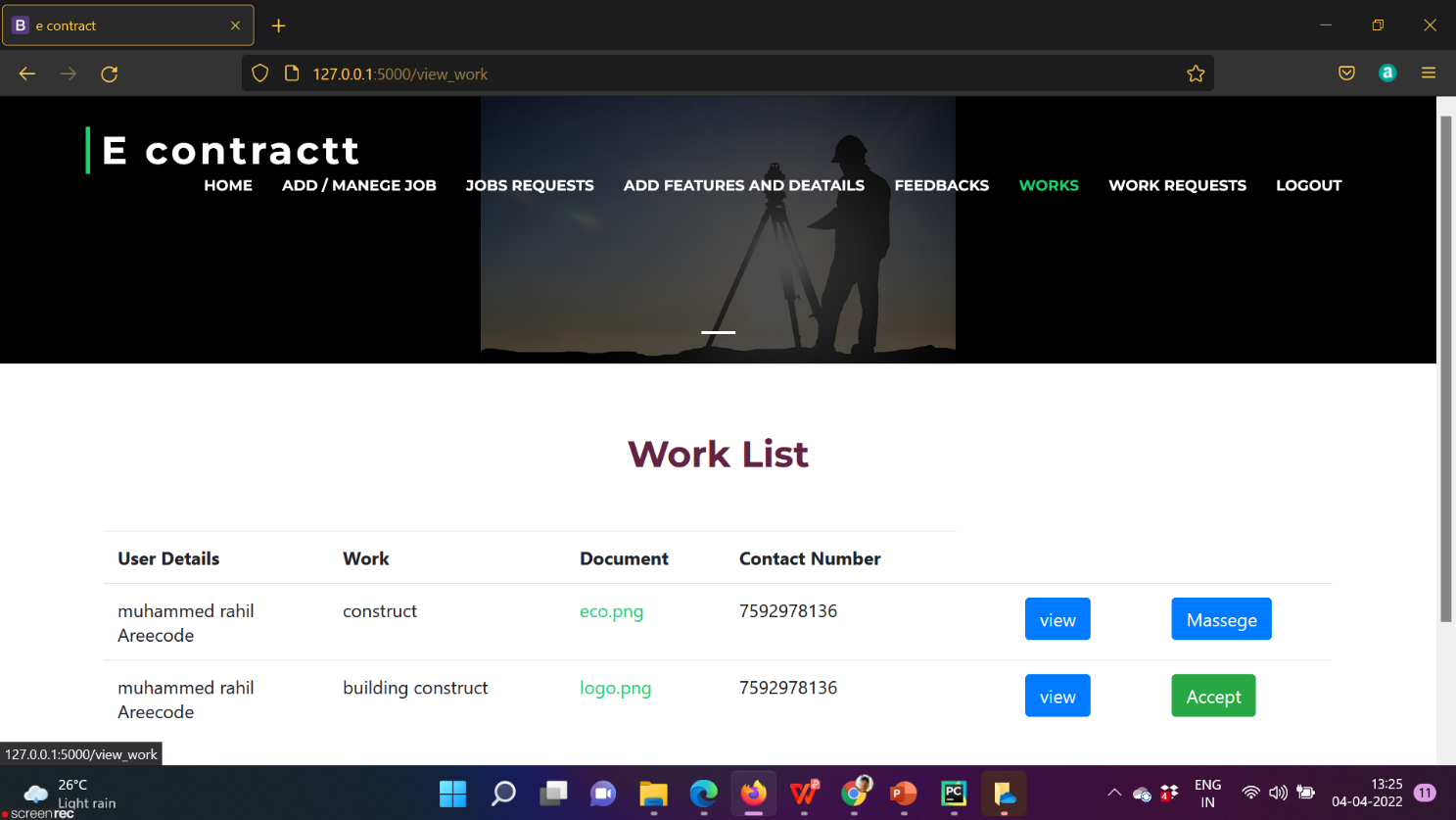
**MANAGE JOB**

****

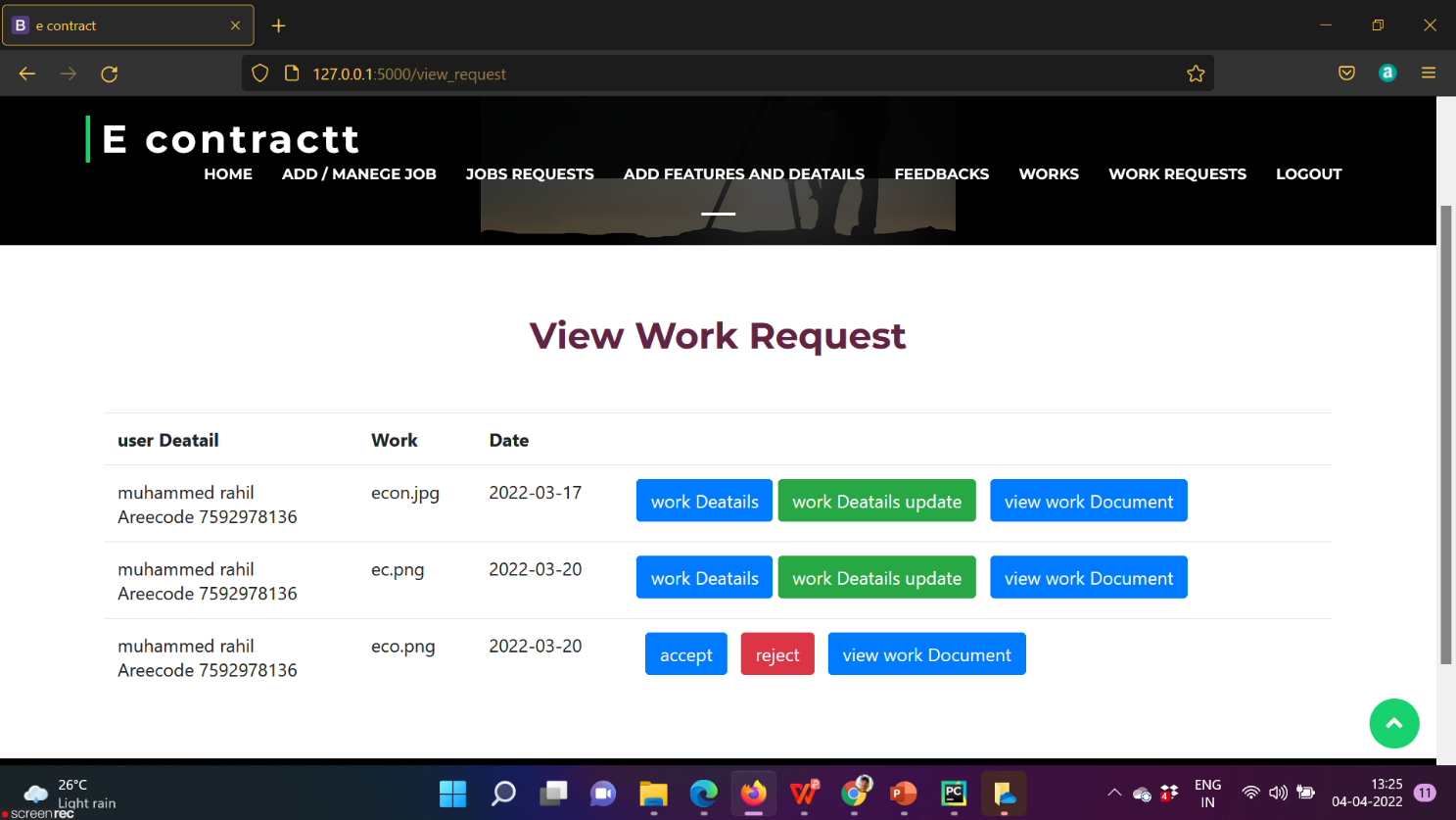
**VIEW JOB REQUEST**

****

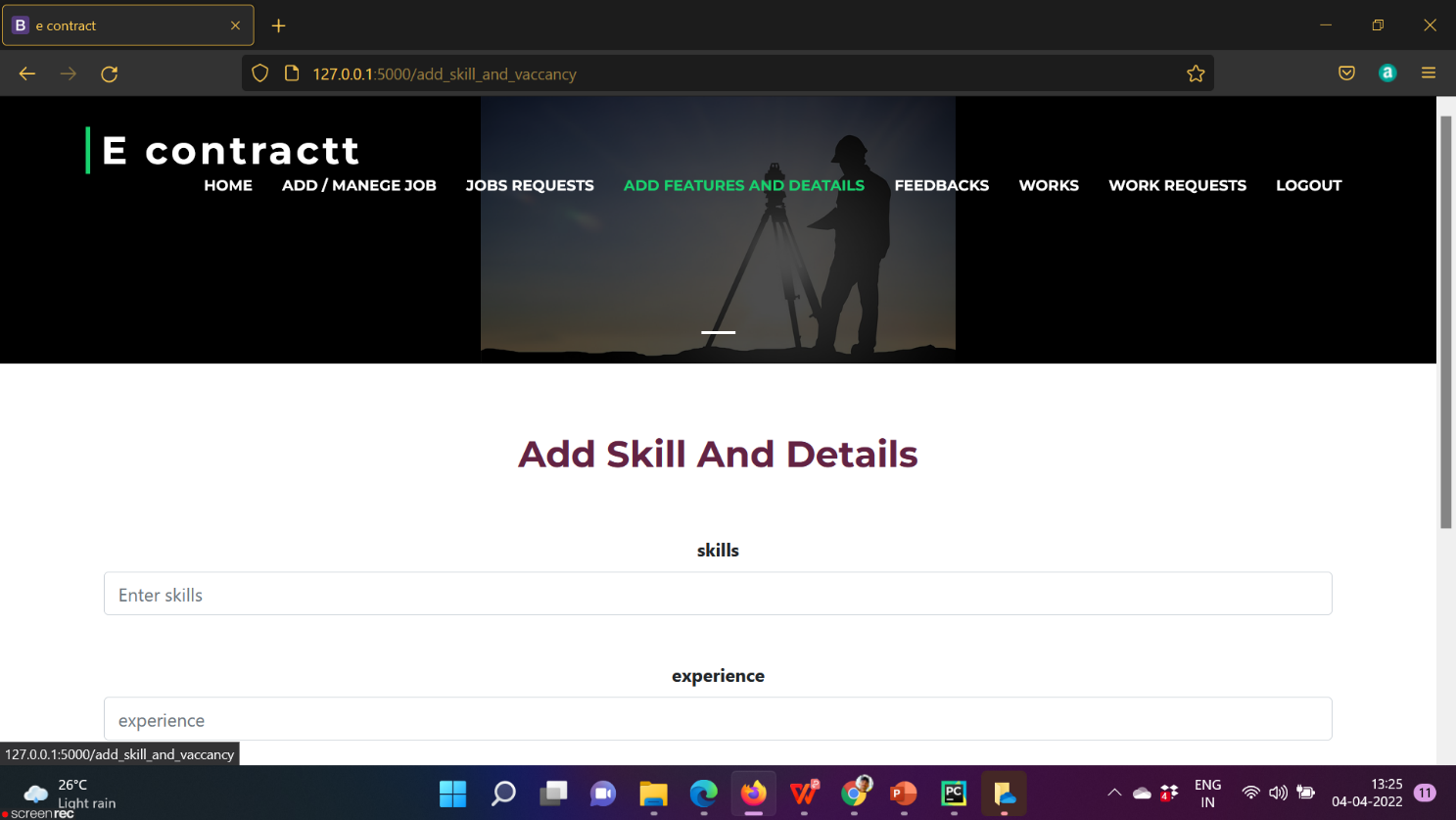
**WORK LIST**

****

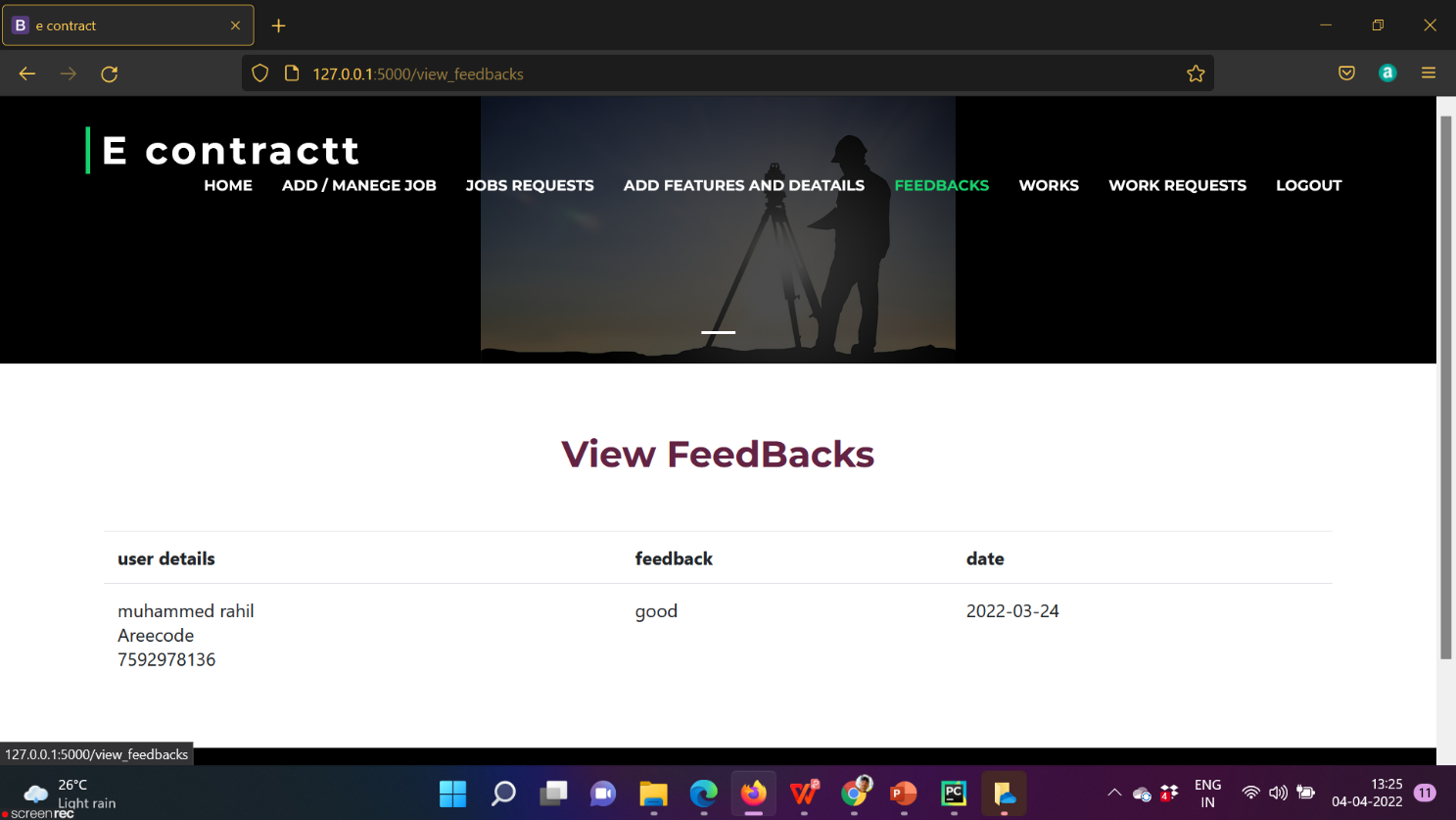
**VIEW WORK REQUEST**

****

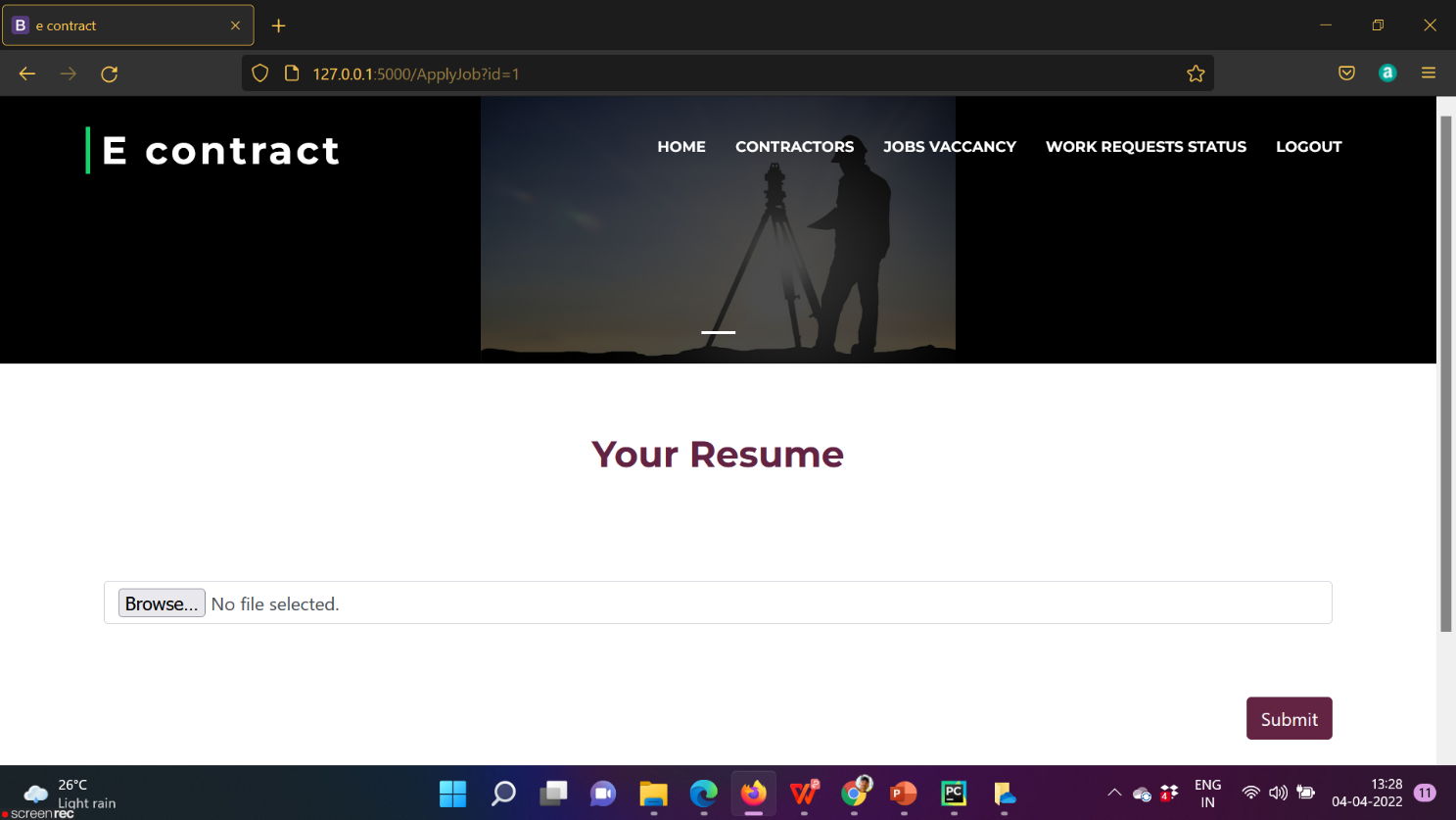
**ADD WORK DETAILS**

****

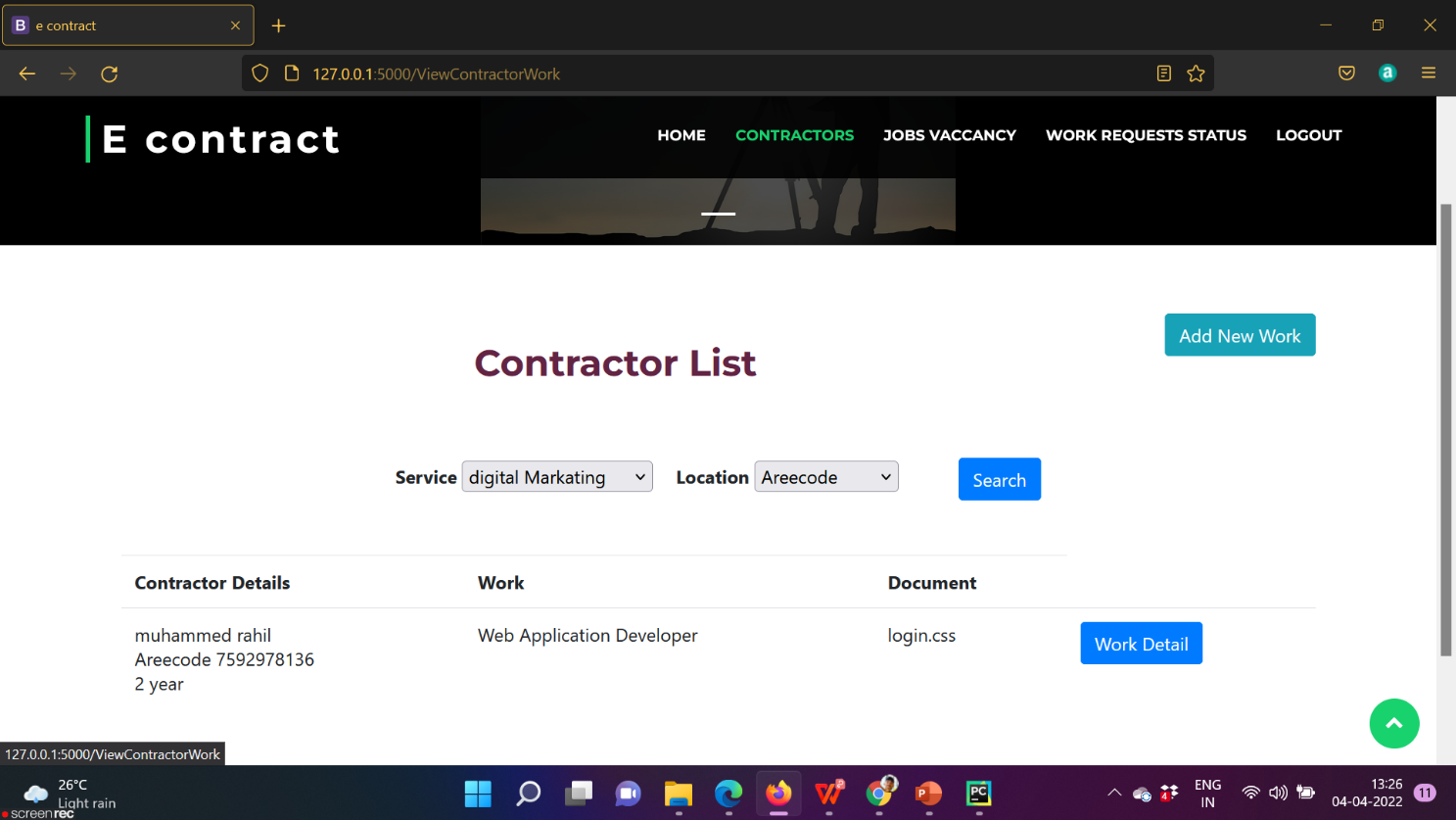
**VIEW FEEDBACK**

****

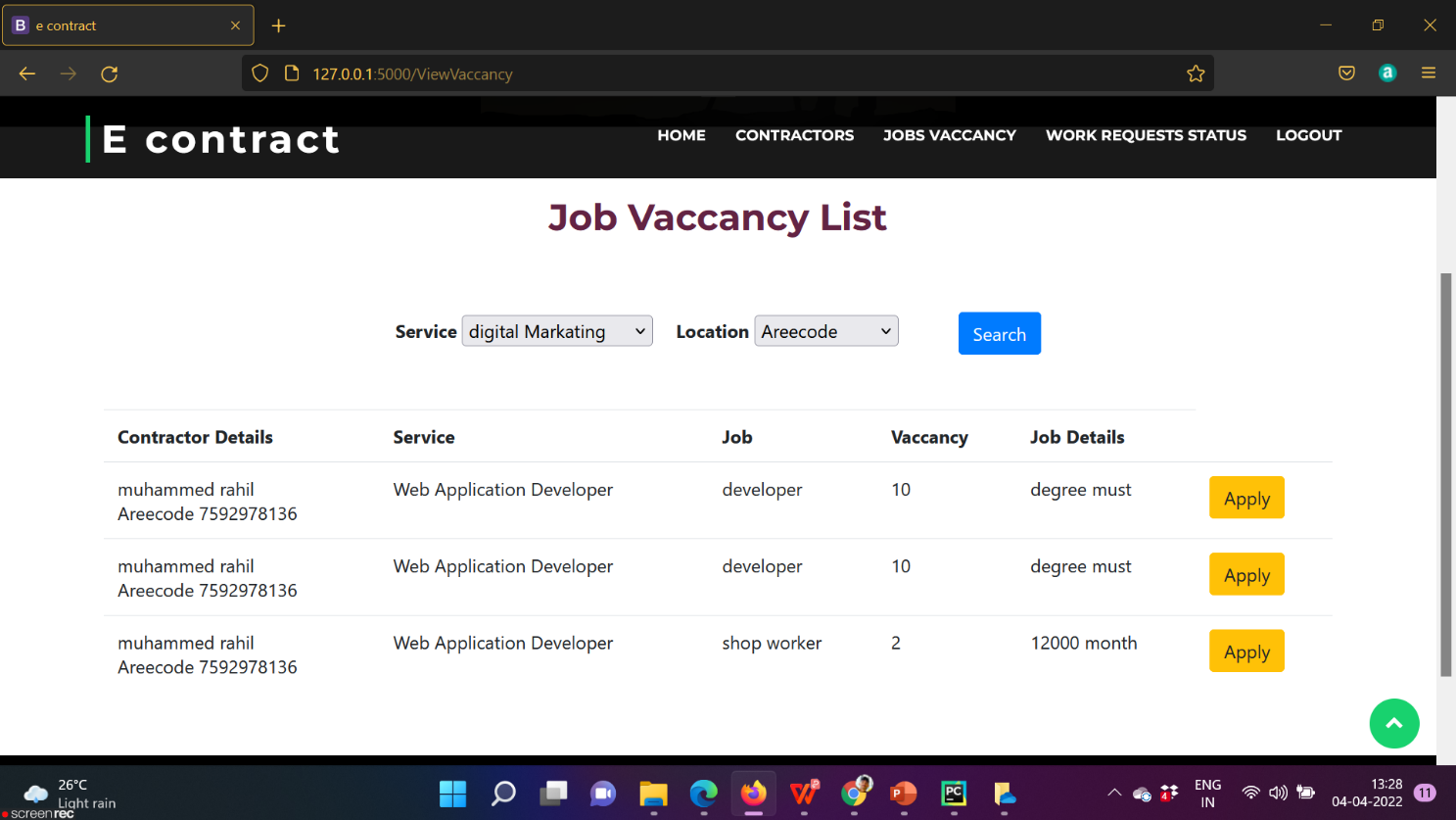
**ADD WORK**

****

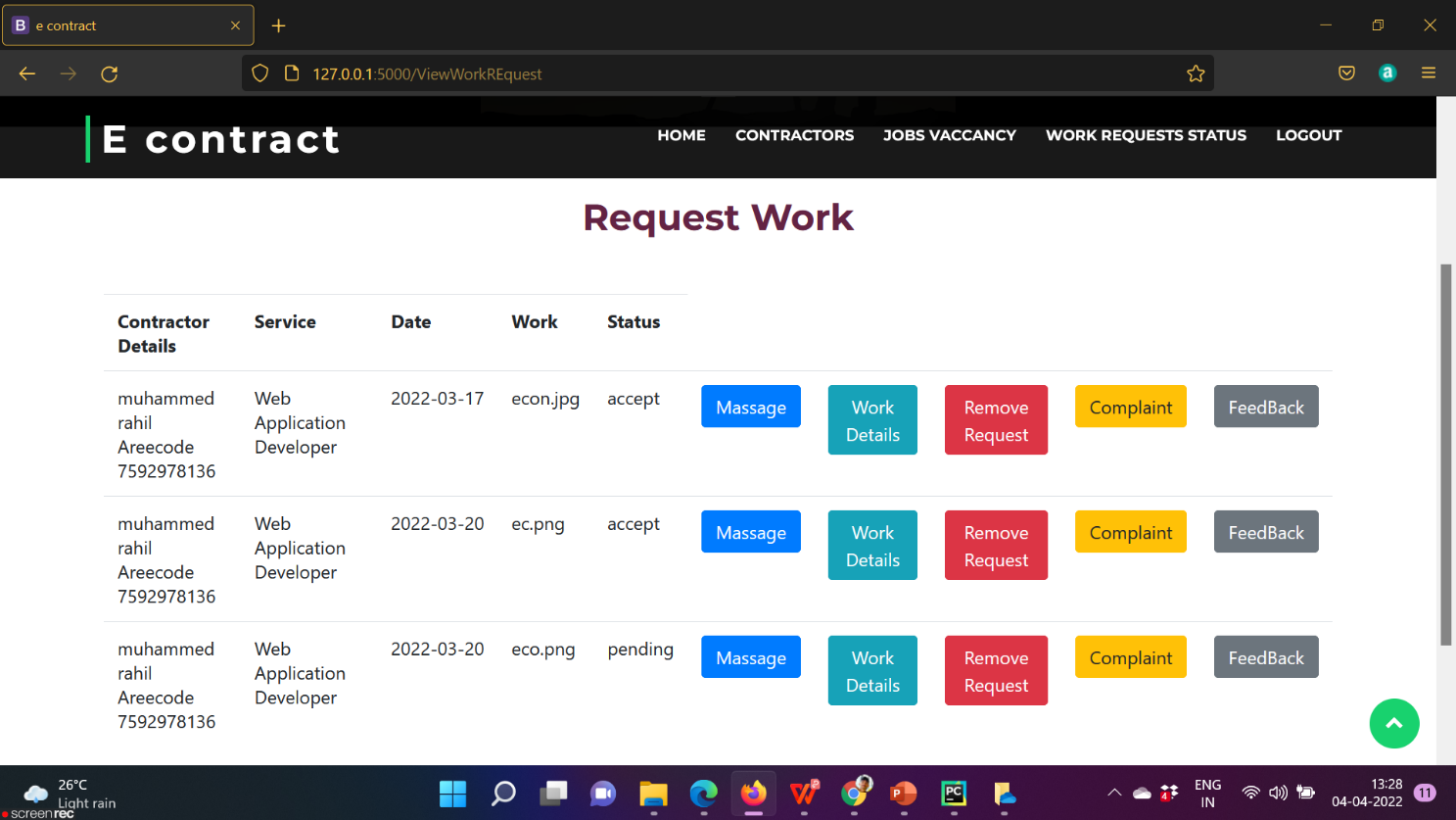
**FIND CONTRACTOR**

****

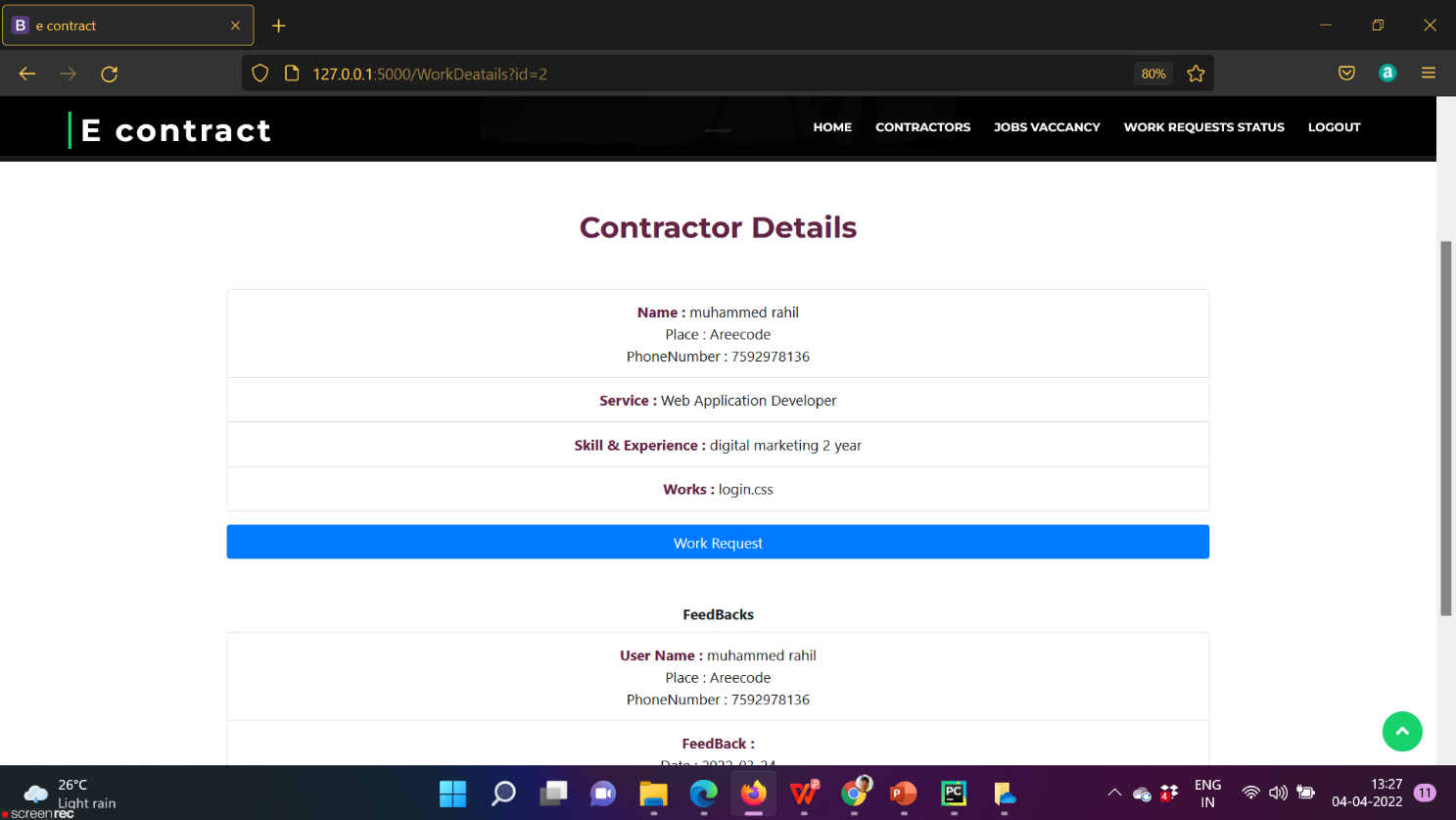
**VIEW JOB VACANCY**

****

**WORK STATUS**

****

**CONTRACTOR DETAILS**

****

**SOURCE CODE**

**SOURCE CODE**

**PYTHON CODE**

import os

from flask import\*

from werkzeug.utils import secure\_filename

app = Flask (\_\_name\_\_)

app.secret\_key="key"

from src.dbconnection import \*

@app.route("/")

def login():

    return render\_template("admin/login.html")

@app.route("/getLogin",methods=['post','get'])

def getLogin():

    uname = request.form['textfield']

    pwd = request.form['textfield2']

    qry = "select \* from login where username =%s and password=%s"

    vals =(uname,pwd)

    result= selectone(qry,vals)

    if result is None:

        return '''<script>alert ("no result");window.location="/"</script>'''

    elifresult[3] =='admin':

        return render\_template("admin/layout.html")

    elifresult[3] =="contractor" :

        session['lid']= result[0]

        return  render\_template("CONTRACTOR/layout.html")

    elifresult[3] =="user" :

        session['uid']= result[0]

        return  render\_template("user/layout.html")

    else :

        return '''<script>alert("sorry invalid");window.location="/"</script>'''

@app.route("/signup")

def signup():

    return render\_template("admin/signup.html")

@app.route("/Adminlaoyt")

def Adminlaoyt():

return render\_template("admin/layout.html")

@app.route("/add\_signup",methods=['post'])

def add\_signup():

    catogary = request.form['catogary']

    print(catogary)

    if(catogary == 'Contractor'):

                 username = request.form['user']

                 password = request.form['password']

                 fname = request.form['textfield']

                 lname = request.form['textfield1']

                 gender = request.form['textfield3']

                 place = request.form['textfield4']

                 post = request.form['textfield5']

                 pincode = request.form['textfield6']

                 phone = request.form['textfield7']

                 service = request.form['textfield8']

                 logqry = "INSERT INTO `login` VALUES (NULL,%s,%s,'contractor')"

                 logval=(username,password)

                 lid=iud(logqry,logval)

                 qry = "INSERT INTO `contractor` VALUES (NULL,%s,%s,%s,%s,%s,%s,%s,%s,%s)"

                 val = (fname, lname, gender, place,post,pincode,phone,service,str(lid))

                 iud(qry,val)

   return '''<script>alert("signup successfully");window.location="/contracters"</script>'''

    else :

                 username = request.form['user']

                 password = request.form['password']

                 fname = request.form['textfield']

                 lname = request.form['textfield1']

                 place = request.form['textfield4']

                 post = request.form['textfield5']

                 pincode = request.form['textfield6']

                 phone = request.form['textfield7']

                 logqry = "INSERT INTO `login` VALUES (NULL,%s,%s,'user')"

                 logval=(username,password)

                 lid=iud(logqry,logval)

                 print(lid)

                 qry = "INSERT INTO `users` VALUES (NULL,%s,%s,%s,%s,%s,%s,%s)"

                 val = (fname, lname, place,post,pincode,phone,str(lid))

                 iud(qry,val)

                 return '''<script>alert("signup successfully");window.location="/user"</script>'''

@app.route("/logout")

def logout():

    return redirect('/')

# @app.route("/contractor")

# def contractor():

#     qry="SELECT contractor.\* FROM login JOIN contractor ON login.loginid=contractor.loginid WHERE login.usertype ='pending'"

#     res=select(qry)

#     return render\_template("admin/contractor.html",val=res)

# @app.route("/AcceptContractor")

# def AcceptContractor():

#     id=request.args.get('id')

#     accept='contractor'

#     qry="UPDATE `login` SET `usertype`=%s WHERE `loginid`=%s"

#     val=(accept,id)

#     iud(qry,val)

#     return '''<script>alert("accept successfully");window.location="/contractor"</script>'''

# @app.route("/RejectContractor")

# def RejectContractor():

#     id=request.args.get('id')

#     qry="DELETE FROM`login`WHERE `loginid`=%s"

#     iud(qry,str(id))

#     return '''<script>alert("reject successfully");window.location="/contractor"</script>'''

@app.route("/ViewContractor")

def ViewContractor():

    qry="SELECT \* FROM contractor "

    res=select(qry)

    return render\_template("admin/ViewContractor.html",val=res)

@app.route("/ApproveRejectContractor")

def ApproveRejectContractor():

    id=request.args.get('id')

    qry="DELETE FROM`contractor`WHERE `loginid`=%s"

    iud(qry,str(id))

    return '''<script>alert("reject successfully");window.location="/ViewContractor"</script>'''

@app.route("/view\_users")

def view\_users():

    qry="SELECT users.\* FROM login JOIN users ON login.loginid = users.loginid "

    res=select(qry)

    return render\_template("admin/view users.html",val=res)

# @app.route("/view\_job\_vacancy")

# def view\_job\_vacancy():

#     qry = "SELECT \* FROM `contractor`"

#     res=select(qry)

#     return render\_template("admin/view\_job\_vacancy.html",val=res)

# @app.route("/view\_job\_vacancynew",methods=['post'])

# def view\_job\_vacancynew():

#     service =request.form['service']

#     qry = "SELECT `contractor`.\*,`vaccancy`.\* FROM `vaccancy` JOIN `contractor` ON `contractor`.`loginid`=`vaccancy`.`contractid` WHERE `contractor`.`sevice`=%s"

#     val=(service)

#     res=selectall(qry,val)

#     return render\_template("admin/view\_job\_vacancy.html",val1=res)

@app.route("/view\_feedback")

def view\_feedback():

    qry ="SELECT  `users`. \*,feedback.\*,`contractor`.\* FROM  feedback JOIN users ON `users`.`loginid`=`feedback`.`userid` join `contractor` on `contractor`.`loginid`=`feedback`.`cid`"

    res=select(qry)

    return render\_template("admin/view feedback.html",val=res)

@app.route("/view\_complains")

def view\_complains():

    qry= "SELECT users.\*,complaint.\*,`contractor`.\* FROM `complaint` JOIN `users` ON `users`.`loginid`=`complaint`.`userid` JOIN `contractor` ON `contractor`.`loginid`=`complaint`.`conid` WHERE `complaint`.`replay`='pending'"

    res=select(qry)

    return render\_template("admin/view complains.html",val=res)

@app.route("/sent\_replynew")

def sent\_replynew():

    id = request.args.get('id')

    session['id']=id

    return render\_template("admin/sent reply.html" )

@app.route("/sent\_reply",methods=['post'] )

def sent\_reply():

    replay = request.form['textarea']

    qry ="UPDATE `complaint` SET `replay` = %s WHERE `complaintid`=%s"

    val=(replay,session['id'])

    iud(qry,val)

    return '''<script>alert("reply successfully");window.location="view\_complains"</script>'''

@app.route("/admins")

def admins():

    return render\_template("admin/layout.html")

@app.route("/block\_user\_and\_contractor")

def block\_user\_and\_contractor():

    qry = "SELECT \* FROM `contractor`"

    res = select(qry)

    return render\_template("admin/block user and contractor.html",val=res)

@app.route("/manege\_Vaccancy")

def manege\_Vaccancy():

    return render\_template("admin/manege\_Vaccancy.html")

@app.route("/view\_contractor\_vacancy")

def view\_contractor\_vacancy():

    return render\_template("admin/view contractor vacancy.html")

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@app.route("/contracters")

def contracters():

    return render\_template("CONTRACTOR/layout.html")

@app.route("/add\_job\_vacancy")

def add\_job\_vacancy():

    return render\_template("CONTRACTOR/ADD JOB VACCANCY.html")

@app.route("/add\_skill\_and\_vaccancy")

def add\_skill\_and\_vaccancy():

    return render\_template("CONTRACTOR/ADD SKILL AND DETAILS.html")

@app.route("/new\_add\_skill\_and\_vaccancy",methods=['post'])

def new\_add\_skill\_and\_vaccancy():

    conid=session['lid']

    skill = request.form['textfield1']

    exprience = request.form['textfield2']

    work = request.form['Document']

    qry="INSERT INTO `features` VALUES (NULL,%s,%s,%s,%s)"

    val=(conid,skill,exprience,work)

    iud(qry,val)

    return '''<script>alert("added successfully");window.location="/add\_skill\_and\_vaccancy"</script>'''

@app.route("/add\_new")

def add\_new():

    return render\_template("CONTRACTOR/addnew.html")

@app.route("/get\_add\_new" ,methods=['post'])

def get\_add\_new():

    job=request.form['textfield1']

    vaccancy = request.form['textfield2']

    details = request.form['textfield3']

    qry="INSERT INTO `vaccancy` VALUES (NULL,%s,%s,%s,CURDATE(),%s)"

    val=(job,details,session['lid'],vaccancy)

    iud(qry,val)

    return '''<script>alert("added successfully");window.location="/manege\_vaccancy"</script>'''

@app.route("/delete", methods=['post','get'])

def delete():

    id = request.args.get('id')

    qry="DELETE FROM `vaccancy` WHERE  `vaccid`=%s "

    iud(qry,id)

    return '''<script>alert("delete successfully");window.location="/manege\_vaccancy"</script>'''

@app.route("/edit\_new")

def edit\_new():

    id = request.args.get('id')

    session['editid']=id

    qry="SELECT \* FROM `vaccancy` WHERE `vaccid`=%s"

    res=selectone(qry,id)

    return render\_template("CONTRACTOR/editnew.html",val=res)

@app.route("/edit\_new\_vacancy", methods=['post'])

def edit\_new\_vacancy():

    job=request.form['textfield1']

    vaccancy = request.form['textfield2']

    details = request.form['textfield3']

    qry="UPDATE `vaccancy` SET `job` =%s ,`details`=%s,`vacancy`=%s where vaccid=%s "

    val=(job,vaccancy,details,session['editid'])

    iud(qry,val)

return '''<script>alert("edit successfully");window.location="/manege\_vaccancy"</script>'''

@app.route("/manege\_vaccancy")

def manege\_vaccancy():

    qry="SELECT \* FROM `vaccancy` "

    res=select(qry)

    return render\_template("CONTRACTOR/MANEGE VACCANCY.html",val=res)

@app.route("/view\_job\_request")

def view\_job\_request():

    qry="SELECT users.\*,`vaccancy`.\*,`applyjob`.\* FROM `vaccancy` JOIN `applyjob` ON `vaccancy`.`vaccid`=`applyjob`.`vacancyid` JOIN `users` ON `users`.`loginid`=`applyjob`.`userid` WHERE `applyjob`.`status`='pending'"

    res=select(qry)

    return render\_template("CONTRACTOR/VIEW JOB REQUEST.html",val=res)

@app.route("/AcceptJobRequest")

def AcceptJobRequest():

    accept='accept'

    id = request.args.get('id')

    qry="UPDATE applyjobSET  status = %s WHERE applyid=%s"

    val=(accept,id)

    iud(qry,val)

    return '''<script>alert("Accept successfully");window.location="/view\_job\_request"</script>'''

@app.route("/RejectJobRequest")

def RejectJobRequest():

    Reject='Reject'

    id = request.args.get('id')

    qry="UPDATE applyjobSET  status = %s WHERE applyid=%s"

    val=(Reject,id)

    iud(qry,val)

    return '''<script>alert("Accept successfully");window.location="/view\_job\_request"</script>'''

@app.route("/view\_feedbacks")

def view\_feedbacks():

    qry="SELECT users.\*,`feedback`.\* FROM `feedback` JOIN `users` ON `users`.`loginid`=`feedback`.`userid` "

    res=select(qry)

    return render\_template("CONTRACTOR/view feedback.html",val=res)

@app.route("/view\_request")

def view\_request():

    qry="SELECT users.\*,`request`.\* FROM `request` JOIN `users` ON `users`.`loginid`=`request`.`userid` "

    res=select(qry)

    return render\_template("CONTRACTOR/view request.html",val=res)

@app.route("/accept\_request")

def accept\_request():

    accept = "accept"

    id = request.args.get('id')

    session['id'] = id

    qry = "UPDATE `request` SET `status` = %s WHERE requestid =%s"

    val = (accept, session['id'])

    iud(qry, val)

    return '''<script>window.location="/view\_request"</script>'''

@app.route("/reject\_request")

def reject\_request():

    id = request.args.get('id')

    session['id'] = id

    qry = "DELETE FROM request WHERE requestid =%s"

    iud(qry,id)

    return '''<script>window.location="/view\_request"</script>'''

@app.route("/view\_request\_Deatails")

def view\_request\_Deatails():

    return render\_template("CONTRACTOR/workDeatails.html")

@app.route("/Get\_view\_request\_Deatails",methods=['post'])

def Get\_view\_request\_Deatails():

    details = request.form['Deatail']

    file = request.files['file']

    filename = secure\_filename(file.filename)

    file.save(os.path.join("static",filename))

    qry="INSERT INTO `workdeatails` VALUES (NULL,%s,%s,%s)"

    val=(details,filename,session['lid'])

    iud(qry,val)

    return '''<script>alert("Add successfully");window.location="/view\_request"</script>'''

@app.route("/status")

def status():

    return render\_template("CONTRACTOR/status.html")

@app.route("/view\_work")

def view\_work():

    qry="SELECT `users`.\*,`works`.\* FROM `users` JOIN `works` ON  `users`.`loginid`=`works`.`loginid` "

    res=select(qry)

    return render\_template("CONTRACTOR/view work.html",val=res)

@app.route('/uploads/<path:filename>', methods=['GET', 'POST'])

def download(filename):

    print(filename)

    print(app.root\_path)

    full\_path = os.path.join(app.root\_path, "static")

    print(full\_path)

    return send\_from\_directory(full\_path, filename)

@app.route("/Massege\_view\_work")

def Massege\_view\_work():

    accept='your work accept'

    id = request.args.get('id')

    qry="UPDATE works SET  status = %s WHERE workid =%s"

    val=(accept,id)

    iud(qry,val)

    return '''<script>alert("Request successfully");window.location="/view\_work"</script>'''

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@app.route("/user")

def user():

    return render\_template("user/layout.html")

@app.route("/ViewContractorWork")

def ViewContractorWork():

    qry="SELECT `contractor`.\*,`features`.\* FROM `contractor` JOIN `features` ON `contractor`.`loginid`=`features`.`contractid` "

    res=select(qry)

    return render\_template("user/ViewContractorWork.html",val=res)

@app.route("/GetViewContractorWork",methods=['post'])

def GetViewContractorWork():

    servis =request.form['catogary']

    place =request.form['place']

    qry="SELECT `contractor`.\*,`features`.\* FROM `contractor` JOIN `features` ON `contractor`.`cid`=`features`.`contractid` WHERE `contractor`.`sevice`=%s OR `contractor`.`place`=%s"

    val1=(servis,place)

    res=selectall(qry,val1)

    return render\_template("user/GetViewContractorWork.html",val=res)

@app.route("/Add\_work")

def Add\_work():

    return render\_template("user/AddWork.html")

@app.route("/Get\_Add\_work",methods=['post'])

def Get\_Add\_work():

     Work =request.form['Work']

     file = request.files['Document']

     filename = secure\_filename(file.filename)

     file.save(os.path.join("static",filename))

     qry="INSERT INTO `works` VALUES (NULL,%s,%s,%s,'pending')"

     val=(session['uid'],Work,filename)

     iud(qry,val)

     return '''<script>alert("Add successfully");window.location="/Add\_work"</script>'''

@app.route("/ViewVaccancy")

def ViewVaccancy():

    qry="SELECT `contractor`.\*,`vaccancy`.\* FROM `contractor` JOIN `vaccancy`  ON `contractor`.`loginid`=`vaccancy`.`contractid`"

    res=select(qry)

    return render\_template("user/ViewVaccancy.html",val=res)

@app.route("/GetViewVaccancy",methods=['post'])

def GetViewVaccancy():

    servis =request.form['catogary']

    place =request.form['place']

    qry="SELECT `contractor`.\*,`vaccancy`.\* FROM `contractor` JOIN `vaccancy`  ON `contractor`.`cid`=`vaccancy`.`contractid`  WHERE `contractor`.`sevice`=%s OR `contractor`.`place`=%s"

    val1=(servis,place)

    res=selectall(qry,val1)

    return render\_template("user/GetViewVaccancy.html",val=res)

@app.route("/ApplyJob")

def ApplyJob():

    id = request.args.get('id')

    session['rid'] = id

    return render\_template("user/ApplyJob.html")

@app.route("/GetApplyjob",methods=['post'])

def GetApplyjob():

    file =request.form['file']

    qry="INSERT INTO `Applyjob` VALUES (NULL,%s,%s,CURDATE(),'pending',%s)"

    val=(session['rid'],session['uid'],file)

    iud(qry,val)

return '''<script>alert("Apply successfully");window.location="/ViewVaccancy"</script>'''

@app.route("/WorkDeatails")

def WorkDeatails():

    id = request.args.get('id')

    session['cid'] = id

    qry="SELECT `contractor`.\*,`features`.\* FROM `contractor` JOIN `features` ON `contractor`.loginid=`features`.`contractid` WHERE `contractor`.`loginid`=%s"

    res=selectone(qry,id)

    qry1="SELECT `users`.\*,`feedback`.\* FROM `users` JOIN `feedback` ON `users`.`loginid`=`feedback`.`userid` WHERE `feedback`.`cid`=%s"

    res1=selectall(qry1,id)

    return render\_template("user/WorkDeatail.html",val=res,val1=res1)

@app.route("/Work\_Request")

def Work\_Request():

    return render\_template("user/WorkDeitailFile.html")

@app.route("/Get\_Work\_Request",methods=['GET', 'POST'])

def Get\_Work\_Request():

    fil =request.files['files']

    filename = secure\_filename(fil.filename)

    fil.save(os.path.join("static",filename))

    qry="INSERT INTO `request` VALUES (NULL,%s,%s,%s,CURDATE(),'pending')"

    val=(session['uid'],session['cid'],filename)

    iud(qry,val)

    return '''<script>alert("Work Request successfully");window.location="/ViewContractorWork"</script>'''

@app.route("/ViewWorkREquest")

def ViewWorkREquest():

    qry="SELECT `contractor`.\*,`request`.\* FROM `request` JOIN `contractor` ON `contractor`.`loginid`=`request`.`conid` WHERE `request`.`status`='pending' OR `request`.`status`='accept'"

    res=select(qry)

    return render\_template("user/ViewWorkREquest.html",val=res)

@app.route("/RemoveViewWorkREquest")

def RemoveViewWorkREquest():

    id = request.args.get('id')

    qry="DELETE FROM request WHERE requestid =%s"

    iud(qry,id)

    return '''<script>alert("Delete successfully");window.location="/ViewWorkREquest"</script>'''

@app.route("/Sent\_Complaints")

def Sent\_Complaints():

    id = request.args.get('id')

    session['Sid'] = id

    qry="SELECT \* FROM `complaint` WHERE `conid`=%s"

    res=selectone(qry,id)

    return render\_template("user/SentComplaint.html",val=res)

@app.route("/Get\_Sent\_Complaints",methods=['post'])

def Get\_Sent\_Complaints():

    Complaint =request.form['Complaint']

    qry="INSERT INTO `complaint` VALUES (NULL,%s,CURDATE(),%s,'pending',%s)"

    val=(Complaint,session['uid'],session['Sid'])

    iud(qry,val)

    return '''<script>alert(" successfully");window.location="/ViewWorkREquest"</script>'''

@app.route("/View\_WOrk\_Deatails\_Request\_Work")

def View\_WOrk\_Deatails\_Request\_Work():

    id = request.args.get('id')

    session['contraid'] = id

    qry="SELECT \* FROM `workdeatails` WHERE `cid`=%s"

    res=selectone(qry,id)

    return render\_template("user/View\_WOrk\_Deatails\_Request\_Work.html",val=res)

@app.route("/FeedBack")

def FeedBack():

    id = request.args.get('id')

    session['feedid'] = id

    return render\_template("user/FeedBack.html")

@app.route("/Get\_FeedBack",methods=['post'])

def Get\_FeedBack():

    FeedBack =request.form['FeedBack']

    qry="INSERT INTO `feedback` VALUES (NULL,%s,%s,CURDATE(),%s)"

    val=(FeedBack,session['uid'],session['feedid'])

    iud(qry,val)

    return '''<script>alert("Feedback add successfully");window.location="/ViewWorkREquest"</script>'''

# @app.route('/Location')

# def Location():

#   return render\_template('Location.html',apikey=api\_key,latitude=latitude,longitude=longitude)

if \_\_name\_\_ == "\_\_main\_\_":

    app.run(debug=True)

**DATABASE CODE**

mportpymysql

def iud(qry,val):

con=pymysql.connect(

host='localhost',

user='root',

port=3306,

password="",

db='econtractor'

)

cmd=con.cursor()

cmd.execute(qry,val)

id=cmd.lastrowid

con.commit()

con.close()

return id

def select(qry):

con=pymysql.connect(

host='localhost',

user='root',

port=3306,

password="",

db='econtractor'

)

cmd=con.cursor()

cmd.execute(qry)

res=cmd.fetchall()

con.commit()

con.close()

return res

def selectall(qry,val):

con=pymysql.connect(

host='localhost',

user='root',

port=3306,

password="",

db='econtractor'

)

cmd=con.cursor()

cmd.execute(qry,val)

res=cmd.fetchall()

con.commit()

con.close()

return res

def selectone(qry,val):

con=pymysql.connect(

host='localhost',

user='root',

port=3306,

password="",

db='econtractor'

)

cmd=con.cursor()

cmd.execute(qry,val)

res=cmd.fetchone()

con.commit()

con.close()

return res