**E-CONTRACT**



***A project report submitted to Kannur University***

***In partial fulfilment of requirements for the award of***

### MASTER

***of***

### COMPUTER APPLICATION

***By***

### ANU TOMY

**REG. NO: C0GMCA2118**

**(2020-2022)**

**DEPARTMENT OF COMPUTER APPLICATIONS**

## DONBOSCO COLLEGE

**ANGADIKADAVU, KANNUR**

**DEPARTMENT OF COMPUTER APPLICATIONS**

## DONBOSCO COLLEGE

**ANGADIKADAVU, KANNUR**



**CERTIFICATE**

This is to certify that the report of the project entitled **“E-CONTRACT”** is a bonafide record of the original work done by **ANU TOMY** (**Reg.No:C0GMCA2118)** during the **fourth semester** of the year **2020-2022** in partial fulfilment of the requirements for the award of Master of Computer Applications under the Kannur university.

**Date:**

**Internal Guide Head of the Institution**

**External Examiners**

**1.**

**2.**

### DECLARATION

I ANU TOMY, fourth semester MCA, student of Don Bosco College, Angadikadavu, under Kannur University do hereby declare that the project entitled “E-CONTRACT” is the record of original work done by me under the supervision of Mrs. Shyni K A Assistant Professor, Don Bosco College, Angadikadavu towards partial fulfilment of the requirement of Master of computer applications, and no part thereof has been presented for the award of any other degree.

Date: ANU TOMY

Reg No: C0GMCA2118

Dept of MCA,

Don Bosco College, Angadikadavu, Kannur

### ACKNOWLEDGEMENT

First of all, I thank the lord Almighty for his immense grace and blessings showered on me at every stage of this work. I am greatly indebted to our Principal Fr. Fidel Thachil, Don Bosco College, Angadikadavu for providing the opportunity to take up this project as part of my curriculum.

I express my sincere thanks to Mr Kevinson Kurian (HOD, Dept. of MCA, Don Bosco College), Mrs. Shyni K A (Internal Guide), other Faculties, Management and Staff of Don Bosco College for the support throughout my MCA course.

Acknowledging the efforts of everyone, their chivalrous help in the course of the project preparation and their willingness to corroborate with the work, their magnanimity through lucid technical details lead to the successful completion of my project.

I would like to express my sincere thanks to all my friends, colleagues, parents and all those who have directly or indirectly assisted during this work.

ANU TOMY

**ABSTRACT**

The main idea of E-CONTRACT application is to make contract work available to everyone. This contract application covers all existing 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. Users can also record the feedback of contractors. In this document we present our understanding of what needs to be accomplished in the project along with the budget and time estimation.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Chapters** | **Contents** | **Page No** |
| **1** | **INTRODUCTION** | 1 |
| 1.1 | Introduction | 2 |
| 1.1 | Project review | 2 |
| **2** | **SYSTEM ANALYSIS** | 3 |
| 2.0 | System Analysis | 4 |
| 2.1 | Requirement Analysis | 4 |
| 2.2 | Existing System | 5 |
| 2.3 | Proposed System | 5 |
| 2.4 | Feasibility study | 6 |
| 2.4.1 | Technical Feasibility | 6 |
| 2.4.2 | Economic Feasibility | 6 |
| 2.4.3 | Behavioural Feasibility | 7 |
| 2.4.4 | Legal Feasibility | 7 |
| 2.5 | System Requirement Specification | 7 |
| 2.5.1 | Actor Identification | 8 |
| 2.5.2 | Use case Identification | 9 |
| 2.5.2.1 | Use case diagrams | 12 |
| 2.5.3 | Activity Diagram | 16 |
| 2.5.4 | Sequence Diagram | 19 |
| 2.6 | System Requirements | 22 |
| 2.6.1 | Hardware and software requirements | 22 |
| **3** | **SYSTEM DESIGN** | 23 |
| 3.0 | System Design | 23 |
| 3.1 | Database Design | 24 |
| 3.1.1 | Data Flow Diagram | 25 |
| 3.1.2 | ER Diagram | 30 |
| 3.1.3 | Table Design | 32 |
| 3.2 | Hierarchical Diagram | 37 |
| 3.3 | Interface Design | 38 |
| 3.3.1 | Input Design | 38 |
| 3.3.2 | Output Design | 39 |

|  |  |  |
| --- | --- | --- |
| **4** | **CODING** | 40 |
| 4.1 | About the Software Tool Used | 41 |
| 4.1.1 | Python-Flask | 41 |
| 4.2 | Coding Principle | 42 |
| 4.3 | Sample Code | 43 |
| **5** | **TESTING** | 46 |
| 5.0 | System testing | 47 |
| 5.1 | Unit Testing | 47 |
| 5.2 | Integration Testing | 49 |
| 5.3 | System Testing | 50 |
| 5.4 | Validation | 50 |
| **6** | **IMPLEMENTATION** | 51 |
| 6.1 | System Implementation | 52 |
| **7** | **CONCLUSION** | 53 |
| 7.1 | Conclusion | 54 |
| **8** | **REFERENCES** | 55 |
| 8.0 | References | 56 |
| 8.2 | Website | 56 |
| **9** | **APPENDIX** | 57 |
|  | Sample screen with valid data | 58 |

# CHAPTER 1 INTRODUCTION

**1. INTRODUCTION**

* 1. **PROJECT OVERVIEW**

The main idea of E-CONTRACT application is to make contract work available to everyone. This contract application covers all existing 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. Users can also record the feedback of contractors. In this document we present our understanding of what needs to be accomplished in the project along with the budget and time estimation.

E-CONTRACT is an android web application which mainly enhances communication among contractors and users. So the user can easily search contractors with their needs. It is an interactive web enabled application. Admin can block the contractors from site if they are not good at works. If the user have any complaint about works they can inform to the admin through this site. The

main website feature is admin can monitor all functions and provide contractors for users. The contractors can register to this site with their technical skills. The user can easily communicate with contractor to the communication platform. User can buy the necessary materials for construction from contractors and user can transfer money through this application itself. By this can save time and money. It is simple user interface. This allows user to select their contractors and the contractor can take over the users work. Contractors add vacancy and user view vacancy and apply job. It help the user to get a job. The main idea of this application to make contract work available to everyone.

**1.2. OBJECTIVE**

The main objective of this application is to ensure the communication between contractors and users. In this application the contractor can keep track of the users works, user can view vacancy in the contractor site and apply job, checking the efficiency of nearby contractor, works document sharing and download chat each other.

# CHAPTER-2 SYSTEM ANALYSIS

**2. SYSTEM ANALYSIS**

System analysis is the process of collecting and interpreting facts, understanding problems and using the information to suggest improvements to the system. This will help to comprehend the existing system and decide how computers make its operation more adequate. This analysis aims to collect detailed information on the system and the feasibility study of the proposed system. This analysis focuses on the flow of the system module by module and the efficiency of each. To design the proposed system we need the exact processing logic as well as the extended features of the existing system such as reliability, consistency, storage capacity, etc. This report will discuss the advantages and drawbacks/disadvantages of the existing system and the modifications and enhancements that can be done. This analysis will concentrate on the information gathering for an efficient, user-friendly, and reliable system, which will carry forward the features of the existing system.

**2.1. REQUIREMENT ANALYSIS**

Requirements analysis results in the specification of software’s functional features meaning the software’s interface with other system components and specifies limitations that software must meet. Requirements analysis allows you to elaborate on basic requirements specified during the inception, elicitation, and negotiation tasks that are a factor of Requirements engineering.

**REQUIREMENT GATHERING**

The requirement gathering can be done in the following ways.

* Interview and site visit
* Questionnaire
* Website Visit

For this project, I have used the Interview method, Site visit, and Questionnaire

* Interview with the Contractor.
* Interview with the Customers.

**2.2 EXISTING SYSTEM**

Currently, there are no similar concepts like this system. The existing system is a manually maintained system. Now a days, people find a contractor by going directly and looking for them. A lot of time is waisted in this process. Now there’s no application to find a particular work-related contractor from home. Users need to personally go to contractors and check their work and book the contractor.

**2.3 PROPOSED SYSTEM**

In the proposed system propose to computerize the above-mentioned activities. In existing system, all data processing is done manually. All the files and records are replaced by the system software. When there are a lot of issues such as retrieval and storage information, reporting etc. and keeping track of them becomes tedious task. By implementing a computerized system, the limitation in the present system will be reduced. Man power can be reduced to a great extend and efficiency and accuracy can be increased to manifold. More over consumption of time can be traduced to far greater extend by the implementation of the proposed system. The main objective of this application is to make contract work available to everyone. This contract application covers all existing contract work in the community. This will benefit both job seekers and customers. The Simple user interface. This allows customer to select their contractors and the contractor can take over the customers works. easy to use.

**2.4 FEASIBILITY STUDY**

The feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work, effort, and time spent on it. The feasibility study lets the developer predict the future of the project and its benefit. The feasibility study is a test of the system proposed regarding its workability, impact on the organization, ability to meet the needs, and effective use of resources. Thus when a new project is proposed,it normally goes through a feasibility study before it’s authorized for development. The document provides the feasibility of

the project that is being developed and documents different areas that were considered very carefully during the feasibility study of this project such as technical,economical,and behavioral feasibilities. The proposed system is theoretically analyzed to check its feasibility and found that they are more reliable and reliable in the cases given below. There are three aspects in the feasibility study part of the primary analysis.

* Technical Feasibility
* Economical Feasibility
* Behavioural Feasibility
* Legal Feasibility

The proposed system must be estimated from a technical point of view first, and if technologically feasible the impact on the organization must be assessed. If compatible, the operating system can be devised. Then they must be tested for economic feasibility.

**2.4.1 TECHNICAL FEASIBILITY**

The technical feasibility of the proposed system deals with the technology used in the system. It deals with the hardware and software used in the system whether they are of the latest technology or not and if after a system is prepared, a new technology arises the user wants the system based on that technology. This system uses the windows platform, SQL for the database, Python as the language. Thus E-CONTRACT is technically feasible.

Technical feasibility centers on the current computer system and to what extent it can support the proposed addition. If the budget is a serious constraint, then the project is judged not feasible. Technical feasibility speaks about the existing hardware and the software that we are using and the deviations that we have to make from the existing one. So we a say that this application is technically feasible as there is no change in the configuration more it is cost-effective.

**2.4.2. ECONOMIC FEASIBILITY**

Economic analysis is the most often used method for estimating the effectiveness of a new system. More commonly known as cost/benefit analysis. Python flask, HTML, and SQL databases,Android studio are easily available on the internet. This technique is to determine the advantages and savings that are expected from a candidate system and compare them with costs. If the advantages outweigh the costs, then the conclusion is made to design and implement the

system. Otherwise, further explanations or alterations in the proposed system will have to be made if it is to have a possibility of being approved. This is an ongoing effort that enhances accuracy at each phase of the system life cycle. Economic feasibility speaks about the uses that we get from this project. Here with the introduction of this online process we are not only decreasing the time taken for the registration of the entrants. We even reduce the load on the administrator. As this project is not only reducing the time but also the workload of the user we say that this product is economically feasible.

Top of Form

**2.4.3. BEHAVIORAL FEASIBILITY**

The “E-CONTRACT” is designed in a user-friendly manner and we need not provide any special training for the persons using this software. Those who have a simple browsing experience, they can easily use this software. It does not have any operational barriers, so there is no need to provide any special training for using this software. Hence it is behaviorally feasible

**2.4.4. LEGAL FEASIBILITY**

This website will not produce any illegal problems and it is not harmful to your machine (Hardware & Software). It will not create or spread the virus and it will not violate any government cyber rules and regulations. So E-CONTRACTis legally feasible.

**2.5. SYSTEM REQUIREMENT SPECIFICATION**

System requirements are expressed in a software requirement document. The Software requirement specification (SRS) is the official statement of what is required of the system developers. This requirement document includes the requirements definition and the requirement specification. The software requirement document is not a design document. It should set out what the system should do without specifying how it should be done. The requirement set out in this document is complete and consistent. The software specification document satisfies the following:-

* It specifies the external system behaviour.
* It specifies constraints on the implementation.
* It is easy to change.
* It serves as a reference tool for system maintainers.
* It records forethought about the life cycle of the system.
* It characterizes the acceptable response to undesired events.

# 2.5.1 ACTOR IDENTIFICATION

An actor specifies a role played by a user or the other system that interacts with the system. It may represent roles played by human users, external hardware, or other subjects. Actors are always outside the system and interact directly with it by initiating a use case, providing input to that, and/or receiving outputs from it. Asking certain questions as detailed below can identify the actors of the system.

The following questions will need to identify the actors.

Top of Form

|  |  |  |
| --- | --- | --- |
| **1.** | Who will use the main functionality of the system? | Administrator, Contractors,  Users |
| **2.** | Who will lead support from the system and do their daily tasks? | Administrator, Contractors,  Users. |
| **3.** | Who will maintain and administrate the system? | Admin |
| **4.** | Which hardware devices does the system need to handle? | Tablet, Computer, Mobile phone, etc |
| **5.** | With which other systems, does this system need to  interact? | Database |
| **6.** | Who was interested in the result produced by the system? | Administrator,Contractors  Users. |

As per the above answers, we can conclude the actors. They are;

* Admin
* Contractor
* Customer

**2.5.2 IDENTIFICATION OF USE CASES**

A use case represents the functionality of an actor. It is defined as a set of actions performed

a system, which causes an observable result. An ellipse containing its name inside the ellipse or

below it represents it. It is placed inside the system boundary and connected to an actor with an association. This shows how the use cases and the actor interact.

To find out the use cases, ask the following questions to each of the actors:

* Which functions does the actor need from the system? What does the actor need to do?
* Does the actor need to read, create, destroy, modify or store some kind of information in the system?
* Could the actor’s daily work be simplified or made more efficient by adding new functions to the system?

# Use case identification:

# Use case for the actor Administrator

|  |  |  |
| --- | --- | --- |
| **1** | Which functions does the Administrator require from the system? What does the Administrator need to do? | The Administrator requires the following functionalities from the system such as;  Login, Approve Contractors, Block Contactors, View job vacancy. View contract request,  Feedback view. View complaints and reply. |
| **2** | Does the Administrator need to read, create, destroy, modify or store some kind of information in the system? | Yes. Administrators must create, view, approve, delete and edit the data if required. |
| **3** | Could the Administrator’s work be simplified by adding new functions to the system? | Yes, the system can reduce his/her daily work. |

# The above questions give the following use cases for the actor Administrator.

* + Approve Contactors.
  + Block Contractors.
  + View job vacancy.
  + View contract request
  + Feedback view.
  + View complaints and reply.

# Use case for the actor contractor.

|  |  |  |
| --- | --- | --- |
| **1** | Which functions do the contractor require from the system? What does the user need to do? | Contractor requires the following functionalities from the system such as;  Registration, Add and manage products, Add & manage vacancy request. View customers Job request & update status, Sent feedback. Add & manage works (share documents, videos and images about works),Add features and mange details (skills, experience…etc.)Communication platform, Contractor groups, View ordered product details. |
| **2** | Does the User need to read, create, destroy, modify or store some kind of information in the system? | Yes. contractor need to create, view and delete the data if required. |
| **3** | Could the User work be simplified by adding new functions to the system? | Yes, the system can reduce his/her work. |

# The above questions give the following use cases for the actor Contractor.

* + Registration.
  + Add and manage products.
  + Add & manage vacancy request.
  + View customers Job request & update status.
  + Sent feedback.
  + Add & manage works (share documents, videos and images about works).
  + Add features and mange details (skills, experience… etc.)
  + Communication platform.
  + Contractor groups.
  + View ordered product details.

# Use case for the actor customer.

|  |  |  |
| --- | --- | --- |
| **1** | Which functions do the customer require from the system? What does the user need to do? | Customer requires the following functionalities from the system such as;  Registration. Search Nearby contractors & sent request. View request status. Buy products. Ordered products. Communication platform. View vacancy & Apply job. Complaints and reply. Send feedback. |
| **2** | Does the User need to read, create, destroy, modify or store some kind of information in the system? | Yes. customer need to create, view and delete the data if required. |
| **3** | Could the User work be simplified by adding new functions to the system? | Yes, the system can reduce his/her work. |

# The above questions give the following use cases for the actor Customer.

* + Registration.
  + Search Nearby contractors & sent request.
  + View request status.
  + Buy products.
  + Ordered products.
  + Communication platform.
  + View vacancy & Apply job.
  + Complaints and reply
  + Send feedback.

# 2.5.2.1 USE CASE DIAGRAMS

Use case diagrams to model behaviour within a system and helps the developers understand what the user requires. The stick man represents what’s called an actor. Use case diagrams can be useful for getting an overall view of the system and clarifying what can do and more importantly, what they can’t do. Use case diagrams consist of use cases and actors, and show the interaction between the use case and the actor.

* The purpose is to show the interaction between the use case and the actor.
* To represent the system requirements from the user’s perspective.
* An actor could be the end user of the system or an external system.

A use case is the description of a set of sequences of actions. Graphically it is generated as an ellipse with a solid line including only its name. A use case diagram is a behavioural diagram that delivers a set of use cases and actors and their relationships. It is an association between the use cases and the actor. An actor defines a real-world object. Primary actor - sender, secondary actor – receiver.

# USECASE FOR ADMIN

# 

# 

# USECASAE FOR CONTRACTOR

# 

# USECASE FOR CUSTOMER

# 

# 2.5.3 ACTIVITY DIAGRAMS

The activity diagram supplements the use case by providing a graphical representation of the flow of interaction within a specific scenario. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes. Activity diagrams show the overall flow of control.

Activity diagrams are constructed from a limited number of shapes, connected with arrows. The most important shape types:

* rounded rectangles represent actions;
* diamonds represent decisions;
* bars represent the start (split) or end (join) of concurrent activities;
* a black circle represents the start (initial state) of the workflow;
* an encircled black circle represents the end (final state).

Arrows run from the start to the end and represent the order in which activities happen. Hence, they can be regarded as a form of a flowchart. Typical flowchart approaches lack constructs for describing concurrency, however the join and split symbols in activity diagrams only decide this for simple cases; the meaning of the model is not evident when they are arbitrarily combined with decisions or loops.

# ACTIVITY DIAGRAM FOR ADMIN

# ACTIVITY DIAGRAM FOR CONTRACTOR

# ACTIVITY DIAGRAM FOR CUSTOMER

# 

# 2.5.4 SEQUENCE DIAGRAM

System sequence diagrams are visual summaries of individual use cases. All systems are treated as a black box; the diagram emphasizes events that cross the system boundary from actors to systems. A system sequence diagram should be done for the main success scenario of the use case, and frequent or complex alternative scenarios. A system sequence diagram should specify and show the following:

* External actors
* Messages (methods) invoked by these actors
* Return values (if any) associated with previous messages
* Indication of any loops or iteration area

Professionals, in developing a project, often use system sequence diagrams to illustrate how certain tasks are done between users and the system. These tasks may include repetitive, simple, or complex tasks. The purpose is to illustrate the use case in a visual format. To construct a system sequence diagram, you need to be familiar with the unified modelling language (UML).

These models show the logic behind the actors (people who affect the system) and the system in performing the task. Reading a sequence diagram begins at the top with the actor(s) or the system(s) (which is located at the top of the page). Under each actor or system, there are long dotted lines called lifelines, which are attached to them. Actions are performed with lines that extend between these lifelines. When an action line is connected to a lifeline it shows the interaction between the actor or system. Messages will often appear at the top or bottom of a system sequence diagram to illustrate the action in detail. For example, the actor could request to login, this would be represented by login (username, password). After each action is performed, the response or next action is located under the previous one. As you read down the lines you will see in detail how certain actions are performed in the system.

# .

# 2.6. SYSTEM REQUIREMENTS

# 2.6.1. HARDWARE REQUIREMENTS

The hardware required for the application depends on the following:

* Determining size and capacity requirements
* Computer evaluation and measurements
* Financial factors
* Maintenance and support.

Processor: Intel Core i3

Ram: 4GB

Keyboard: Entry-level quiet key keyboard, ps/2

Mouse: PS/2, 2 buttons with scroll

Monitor: 15-inch monitor

HDD: 40 GB

**2.6.2. SOFTWARE REQUIREMENTS**

The software required for the application depends on the following factors:

* The flexibility of the software
* Software contracts
* Limitations of the software
  + - * Operating system : Windows 7 or above, Android, Linux
      * Technology Used : Python
      * IDE : PyCharm , Eclipse/Android Studio
      * Framework : Flask
      * Database : MySQL

# CHAPTER-3

# SYSTEM DESIGN

**3.0 SYSTEM DESIGN**

Design is a meaningful engineering representation of something that is to be built. It is an iterative process through which requirements are translated into a blueprint for constructing the software. The goal of the design phase is to plan a solution to the problem specified by the requirements document.

Major activities during the design phase are:

* Data Base Design
* Architectural Design
* Interface Design
* Modular Design

**3.1. DATABASE DESIGN**

A database is a collection of interrelated data stored with minimum redundancy to serve many users quickly and efficiently. In database design data independence, accuracy, privacy, and security are given higher priority. Database design is an integrated approach to file design. This activity deals with the design of the physical database. All entries and attributes have been identified while creating the database. The database design deals with the grouping of data into several tables to reduce the duplication of data, minimize storage space, and retrieve the data efficiently.

Guidelines for designing a database:

* Design a relational schema so that it is easy to explain its meaning. Do not combine attributes from multiple entities and relationship types into a single relation.
* Design the database schema so that no insertion, deletion or modification anomalies are present in the relation.
* As far as possible, avoid placing attributes in a base relation whose values may frequently be null.
* Design relation schemas so that they can be joined with equality conditions on attributes that are either primary keys or foreign keys in a way that no spurious tuples are generated.

**Advantage**

* Ease of use
* Data independence
* Accuracy and integrity
* Avoiding inordinate delays
* Recovery from failure
* Privacy and security.

**3.1.1 DATA FLOW DIAGRAM**

A data flow diagram is a graphical technique that depicts data flow and transforms that are applied as data move from input to output. The DFD is used to represent increasing information flow and functional details. A Level 0 DFD also called a fundamental system model or context model represents the entire software elements as a single bubble with input and output indicated by incoming and outgoing arrows respectively. Additional process and information flow parts are represented in the next level i.e., Level 1 DFD. Each of the processes represented at level 1 is a sub-function of the overall system depicted in the context model.

Data flow diagram symbol:

Source/Destination of Data

Data Flow

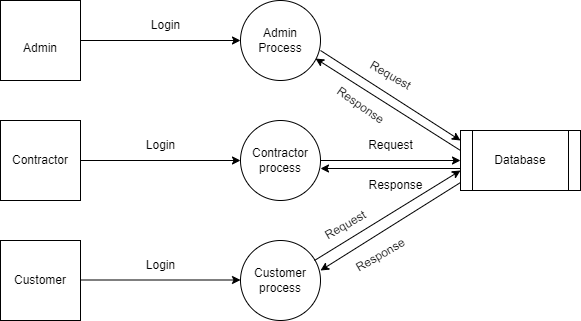
Process

Storage

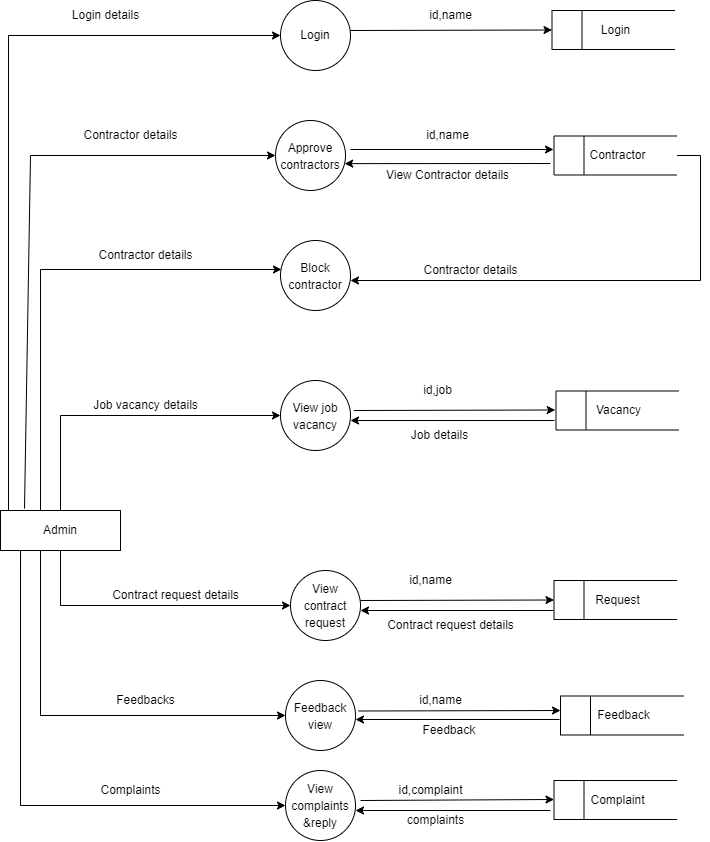
**LEVEL-0**

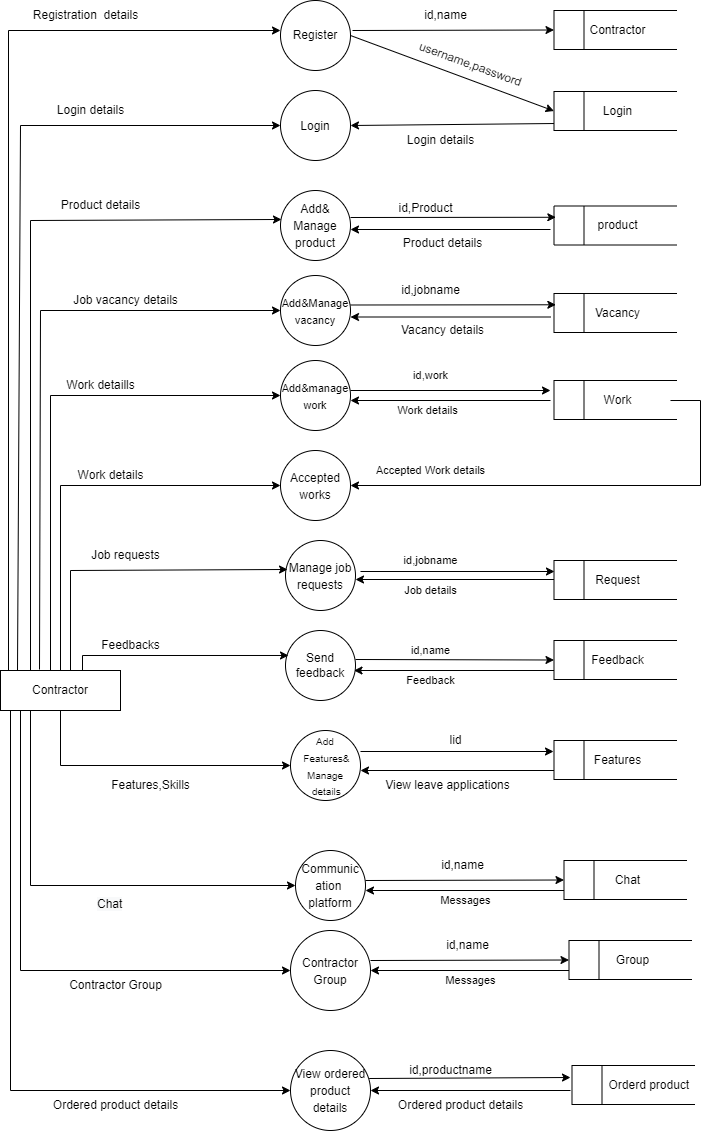
****

**LEVEL-1**

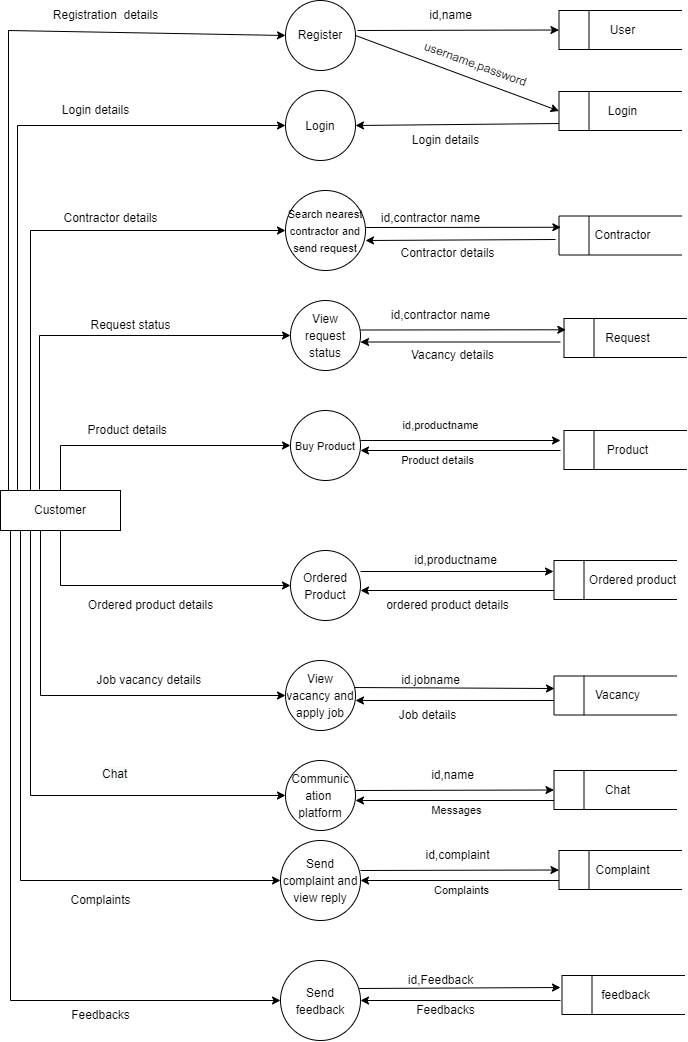
****

**LEVEL 2:ADMIN**

****

**LEVEL 2:CONTRACTOR**

**LEVEL 2:CUSTOMER**

****

**3.1.2 E-R DIAGRAM**

An entity-relationship diagram is a data modelling technique that creates a graphical representation of the entities, and the relationship between entities, within an information system.

There are three basic elements in ER models:

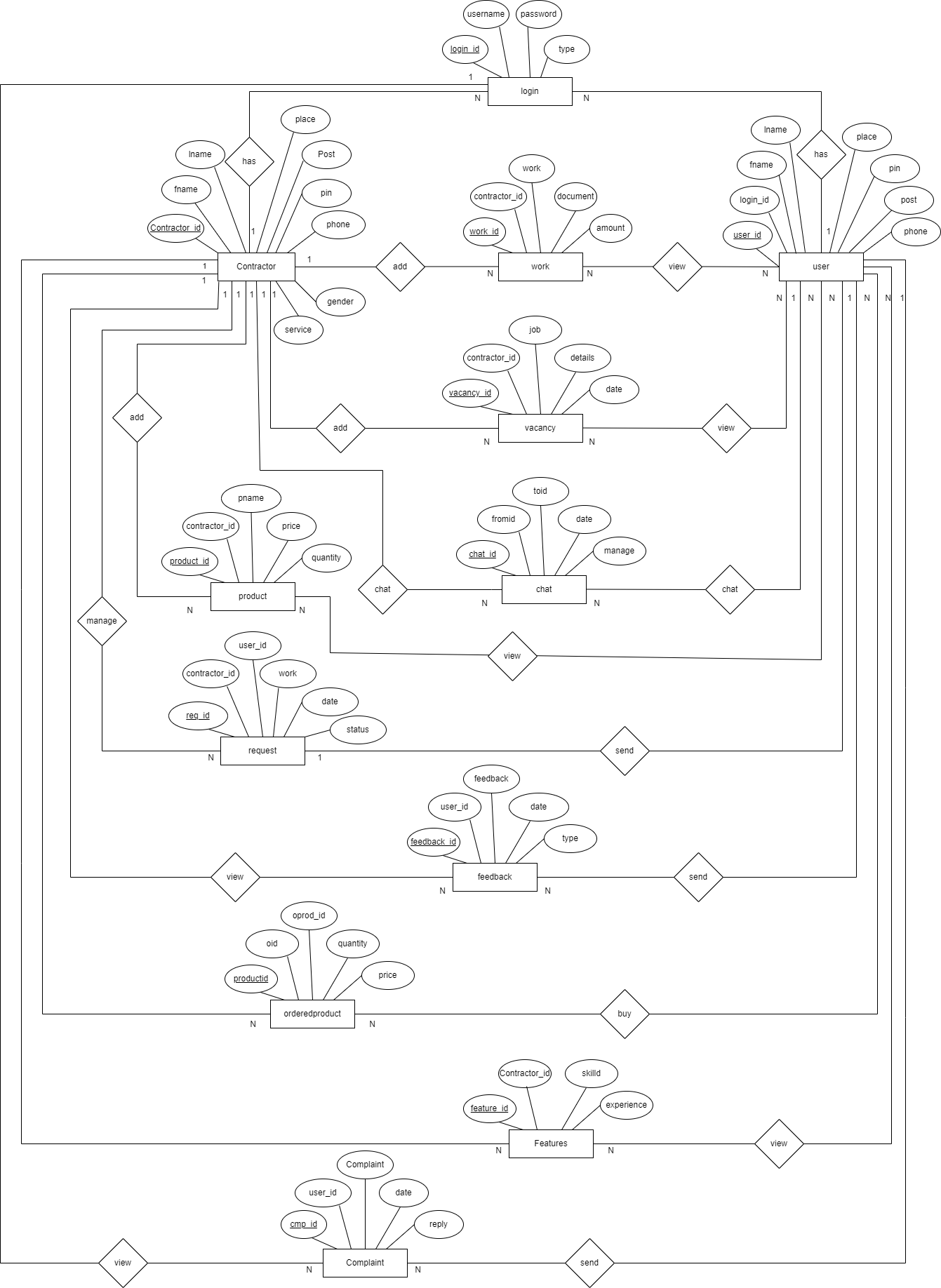
* Entities are the “things” about which we seek information
* Attributes are the data we collect about entities.
* Relationships provided the structure needed to draw information from multiple entities.

E-R Diagram Symbols:

Entity

Attributes

Relation



**3.1.3 TABLE DESIGN**

In the database all the information is stored in the form of tables. A table is simply a way of storing data in rows and columns. In the system data is stored in many tables.

#### Login

|  |  |  |  |
| --- | --- | --- | --- |
| Sl**.**NO | Field Name | Data Type | Constraints |
| 1 | Login\_Id | Int | Primary Key |
| 2 | User\_Name | Varchar | Not Null |
| 3 | Password | Varchar | Not Null |
| 4 | Type | Varchar | Not Null |

1. **Contractor**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl**.**NO | Field Name | Data Type | Constraints |
| 1 | Contractor\_Id | Int | Primary Key |
| 2 | First Name | Varchar | Not Null |
| 3 | Last Name | Varchar | Not Null |
| 4 | Gender | Varchar | Not Null |
| 5 | Place | Varchar | Not Null |
| 6 | Pin | Int | Not Null |
| 7 | Post | Varchar | Not Null |
| 8 | Phone | Int | Not Null |
| 9 | Service | Varchar | Not Null |

#### User

|  |  |  |  |
| --- | --- | --- | --- |
| Sl**.**NO | Field Name | Data Type | Constraints |
| 1 | User\_Id | Int | Primary Key |
| 2 | Login\_Id | Int | Foreign\_Key |
| 3 | First Name | Varchar | Not Null |
| 4 | Last Name | Varchar | Not Null |
| 5 | Place | Varchar | Not Null |
| 6 | Pin | Int | Not Null |
| 7 | Post | Varchar | Not Null |
| 8 | Phone | Int | Not Null |

1. **Vaccancy**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl**.**NO | Field Name | Data Type | Constraints |
| 1 | Vaccancy\_Id | Int | Primary Key |
| 4 | Job | Varchar | Not Null |
| 3 | Details | Varchar | Not Null |
| 4 | Date | Date | Not Null |
| 5 | Contractor\_Id | Int | Foreign\_Key |

#### Feedback

|  |  |  |  |
| --- | --- | --- | --- |
| Sl**.**NO | Field Name | Data Type | Constraints |
| 1 | Feedback\_Id | Int | Primary Key |
| 2 | Feedback | Varchar | Not Null |
| 3 | User\_Id | Int | Foreign\_Key |
| 4 | Date | Date | Not Null |
| 5 | Type | Varchar | Not Null |

1. **Request**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl**.**NO | Field Name | Data Type | Constraints |
| 1 | Request\_Id | Int | Primary Key |
| 2 | User\_Id | Int | Foreign\_Key |
| 3 | Contractor\_Id | Int | Foreign\_Key |
| 4 | Work | Varchar | Not Null |
| 5 | Date | Date | Not Null |
| 6 | Status | Varchar | Not Null |

#### 7.Complaint

|  |  |  |  |
| --- | --- | --- | --- |
| Sl**.**NO | Field Name | Data Type | Constraints |
| 1 | Complaint\_Id | Int | Primary Key |
| 2 | Complaint | Varchar | Not Null |
| 3 | Date | Date | Not Null |
| 4 | User\_Id | Int | Foreign\_Key |
| 5 | Reply | Varchar | Not Null |

1. **Product**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl**.**NO | Field Name | Data Type | Constraints |
| 1 | Product\_Id | Int | Primary Key |
| 2 | Product Name | Varchar | Not Null |
| 3 | Quantity | Int | Not Null |
| 4 | Price | Int | Not Null |
| 5 | Contractor\_Id | Int | Foreign\_Key |

#### Work

|  |  |  |  |
| --- | --- | --- | --- |
| Sl**.**NO | Field Name | Data Type | Constraints |
| 1 | User\_Id | Int | Primary Key |
| 2 | Contractor\_Id | Int | Foreign\_Key |
| 3 | Work | Varchar | Not Null |
| 4 | Document | Varchar | Not Null |
| 5 | Amount | Int | Not Null |

1. **Features**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl**.**NO | Field Name | Data Type | Constraints |
| 1 | Feature\_Id | Int | Primary Key |
| 2 | Contractor\_Id | Int | Foreign\_Key |
| 3 | Skills | Varchar | Not Null |
| 4 | Experience | Varchar | Not Null |

#### Chat

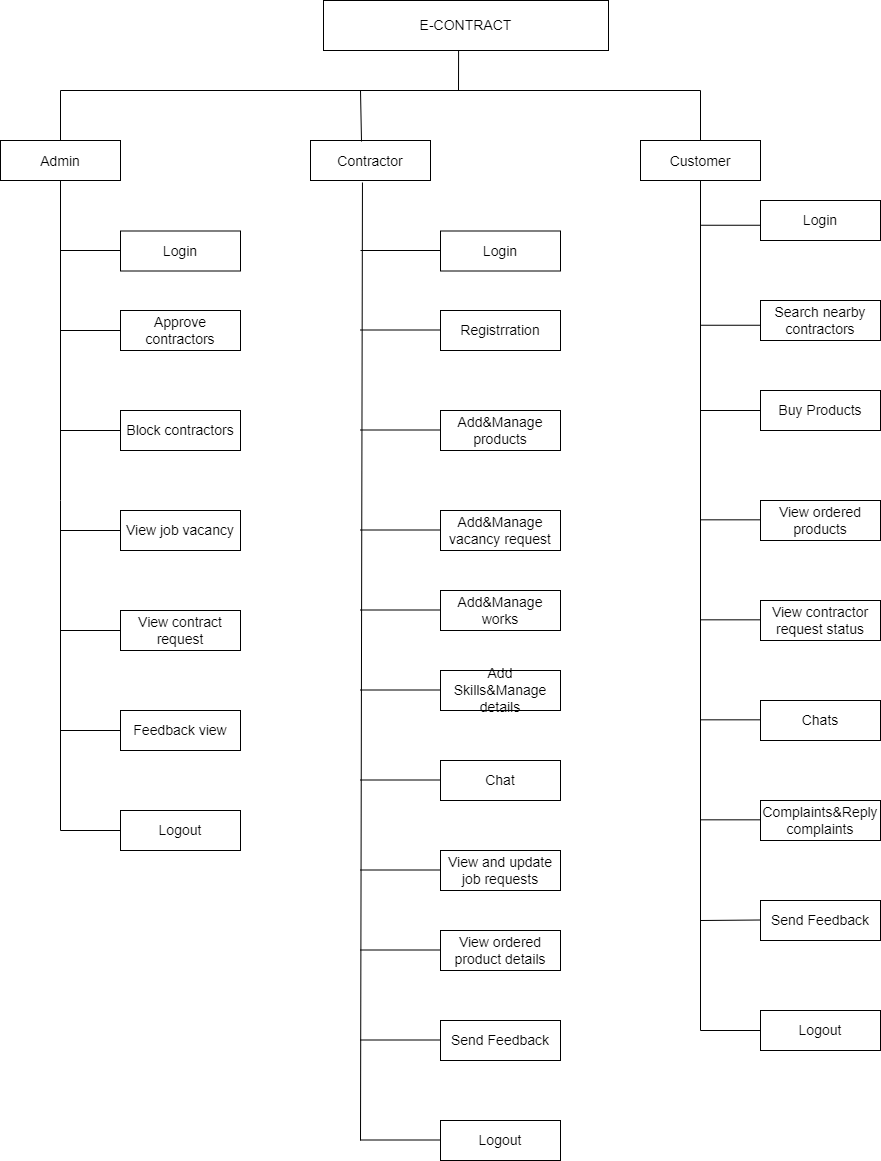
|  |  |  |  |
| --- | --- | --- | --- |
| Sl**.**NO | Field Name | Data Type | Constraints |
| 1 | Chat\_Id | Int | Primary Key |
| 2 | Fromid | Varchar | Not Null |
| 3 | Toid | Varchar | Not Null |
| 4 | Manage | Varchar | Not Null |
| 5 | Date | Date | Not Null |

1. **OrderedProduct**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl**.**NO | Field Name | Data Type | Constraints |
| 1 | Oproduct\_id | Int | Primary Key |
| 2 | Oid | Int | Foreign\_Key |
| 3 | Product\_Id | Int | Foreign\_Key |
| 4 | Quantity | Int | Not Null |
| 5 | Price | Int | Not Null |

**3.2 HIERARCHICAL DIAGRAM**

The hierarchical diagram is a technique for representing the modules of a system as a hierarchy and for documenting each module. It was used to develop requirements, construct the design, and support the implementation of an expert system to demonstrate and verify the system. Structure charts can be used to display several types of information.

****

**3.3 INTERFACE DESIGN**

Interface design elements for the software tell how information flows into and out of the system and how it is communicated among the components as part of the architecture.

**3.3.1 INPUT DESIGN**

Input design is the link between the information system and users and those steps that are necessary to put transaction data into a usable form for processing data entry. Instructing the computer to read data from a written printed document can active the activity of putting data into the computer for processing or it can occur by keying data directly into the system. The design of input focuses on controlling errors, avoiding delay, and keeping the process simple. The system analyst decides the following input design details

* What data to input?
* What medium to use?
* How the data is arranged and coded?

In my project named E-CONTRACT, I tried to include the following design constrains provided in software engineering.

**Avoid scattering of fields in the forms**

In all forms of the software the text boxes (which are provided to input some data), labels (which label the text boxes), combo boxes (list a set of values) etc all are arranged in a neat and well format. It provides a simple look to the pages. The buttons are placed at the bottom of the page and are easily accessible to the user. The menus are arranged below the heading and at a minimum level menu are arranged with pages. The menu provides continuity to the pages.

**User only needs to enter a minimum amount of data**

All forms contain a minimum amount of data, but most essentials. No page provides or wanted bulky data. It provides more easiness to the user. It creates more software for the end user. Also, the operation continues with a single click.

**Avoid confusion in the forms**

All forms have a well-defined menu and each menu name indicates its purpose. So, the user can easily access various forms without confusion. Each form and its subforms are well-labelled. So, the user can easily identify the forms and work on them.

**3.3.2 OUTPUT DESIGN**

Designing a computer should proceed in a well-thought-out manner. The term output means any information produced by the information system whether printed or displayed. Output design is a process that involves designing necessary output that has to be used by various users according to requirements. The efficient intelligent output design should remove the system’s relationship with the users and help in decision-making.

When designing the output, the system analyst must accomplish the following:

* Determine the information present.
* Decide whether to print, display the information and select the output medium.
* Arrange information in an acceptable format.

In my project, the outputs are in the form of reports. They are well formatted and it provides the output in a correct and neat format.

**CHAPTER-4 CODING**

**4. CODING**

**4.1 ABOUT THE SOFTWARE TOOLS USED**

### 4.1.1 Python

Python is a general purpose, dynamic, [high-level](https://www.javatpoint.com/classification-of-programming-languages), and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures. Python is easy to learn yet powerful and versatile scripting language, which makes it attractive for Application Development. Python's syntax and dynamic typing with its interpreted nature make it an ideal language for scripting and rapid application development. Python supports multiple programming pattern, including object-oriented, imperative, and functional or procedural programming styles. Python is not intended to work in a particular area, such as web programming. That is why it is known as multipurpose programming language because it can be used with web, enterprise, 3D CAD, etc. We don't need to use data types to declare variable because it is dynamically typed so we can write a=10 to assign an integer value in an integer variable. Python makes the development and debugging fast because there is no compilation step included in Python development, and edit-test-debug cycle is very fast.

### Flask-FrameWork

Flask is a web framework, it’s a Python module that lets you develop web applications easily. It’s has a small and easy-to-extend core: it’s a microframework that doesn’t include an ORM (Object Relational Manager) or such features. Flask is a web application framework written in Python. It was developed by Armin Ronacher, who led a team of international Python enthusiasts called Poocco. Flask is based on the Werkzeg WSGI toolkit and the Jinja2 template engine. Both are Pocco projects.

### HTML

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript it forms a triad of cornerstone technologies for the World Wide Web. Web browsers receive HTML documents

from a web server or from local storage and render them into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

### ANDROID

The android system supports background processing, provides a rich user interface library, supports 2-D and 3-D graphics using the OpenGL-ES (short OpenGL) standard and grants access to the file system as well as an embedded SQLite database.Android Software Development Kit (Android SDK) contains the necessary tools to create,compile and package Android applications. Most of these tools are command line based. The primary way to develop Android application is based on java programming language.Most of Android’s configuration files are based on XML. In this case these editors allow you to switch between the XML representation of file and a structured user interface for entering the data.

#### Database Servers

A database server is used to store data in a database. Users can access the data and manipulate it. There are many types of databases. The most popular among them is the Relational Database Management System (RDBMS).

#### MYSQL

MySQL is an open source relational database and it includes advanced data types. MySQL operates using client/server architecture in which the server runs on the machine containing the database and client connect to the server over the network. MySQL run on all platforms supported by MySQL and provides the most direct means of interacting with the server, so it's the logical client to begin with.

* You need to have the MySQL software installed.
* You need a MySQL account so that you can connect to the server.
* You need a database to work with.

**4.2 CODING PRINCIPLE**

The input to the coding phase is the design document. During the coding phase, modules identified in the design document are coded according to the module specification. The objectives of the coding phase are, to transform the design into code and unit test the code.

**Coding Guidelines**

* The code should be easy to understand.
* Don’t take pride in cryptic code.
* The code should be well documented.
* Comments should be present.
* Functions should be small.
* Do not use the Go-to statement.
* Do not use the same variable for multiple purposes

**4.3 SAMPLE CODE**

**Login.html**

{% extends 'index.html'%}  
{% block body %}  
<**body**>  
<**form id="form1" name="form1" method="post" action="login"**>  
 <**table width="10" border="1" class="table table-dark table-striped" id="table"**>  
 <**tr** >  
 <**th scope="row"**>username</**th**>  
 <**td**><**label**>  
 <**input type="text" name="textfield" required class="form-control"**/>  
 </**label**></**td**>  
 </**tr**>  
 <**tr**>  
 <**th scope="row"**>password</**th**>  
 <**td**><**label**>  
 <**div align="center"**>  
 <**input type="password" name="textfield2" class="form-control"** />  
 </**div**>  
 </**label**></**td**>  
 </**tr**>  
 <**tr align="center"**>  
 <**th colspan="2" scope="row"**><**label**>  
 <**input type="submit" name="Submit" class="btn btn-secondary" value="login"** />  
 *<!--<a href="contractorregister">signup</a></label></th>-->* </**tr**>  
 </**table**>  
</**form**>  
</**body**>  
*<!--</html>-->*{% endblock %}

**Project.py**

**from** flask **import**\*  
app=Flask(\_\_name\_\_)  
**from** src.database **import**\*  
**import** os  
**from** werkzeug.utils **import** secure\_filename

**import** functools  
**def** login\_required(func):  
 @functools.wraps(func)  
 **def** secure\_function():  
 **if "lid" not in** session:  
 **return** redirect (**"/"**)  
 **return** func()  
 **return** secure\_function  
app.secret\_key=**"123"**@app.route(**'/logout'**)  
**def** logout():  
 session.clear()  
 **return** render\_template(**"login.html"**)  
  
@app.route(**'/'**)  
**def** start\_page():  
 **return** render\_template(**'login.html'**)  
@app.route(**'/login'**,methods=[**'post'**])  
**def** login():  
 username=request.form[**'textfield'**]  
 password=request.form[**'textfield2'**]  
 qry=**"select\*from login where username=%s and password=%s"** val=(username,password)  
 print(val,**"========"**)  
 res=selectone(qry,val)  
 print(**"res"**,res)  
 **if** res **is None**:  
 **return'''<script>alert("invalid");window.location="/"</script>'''  
 elif** res[3]==**'admin'**:  
 session[**'lid'**] = res[0]  
 **return '''<script>alert("successfully login");window.location="adminhome"</script>'''  
 elif** res[3]==**'contractor'**:  
 session[**'lid'**]=res[0]  
 **return '''<script>alert("successfully login");window.location="contractrhome"</script>'''  
 else**:  
 **return'''<script>alert("invalid");window.location="/"</script>'''**@app.route(**'/adminhome'**)  
@login\_required  
**def** adminhome():  
 **return** render\_template(**'adminhome.html'**)  
@app.route(**'/approveuser'**)  
@login\_required  
  
**def** approveuser():  
 qry=**"SELECT `register`.\*,`login`.`type` FROM `register` JOIN `login` ON `login`.`lid`=`register`.`lid` where login.type!='pending'"** res=selectall(qry)  
 print(res)  
 **return** render\_template(**'Approve users.html'**,val=res)  
@app.route(**'/useraprove'**,methods=[**'post'**])  
@login\_required

**webservice.py**

**from** flask **import** \*  
**from** src.database **import** \*  
  
app = Flask(\_\_name\_\_)  
  
@app.route(**'/login'**,methods=[**'post'**])  
**def** login():  
 unam = request.form[**"user\_name"**]  
 pswd = request.form[**"password"**]  
 qry = **"SELECT** *\** **FROM `login` WHERE `username`=%s AND `password`=%s"** val = (unam,pswd)  
 print(val)  
 res = selectone(qry,val)  
 print(res)  
 **if** res **is None**:  
 **return** jsonify({**'task'**:**'invalid'**})  
 **else**:  
 id = res[0]  
 **return** jsonify({**'task'**:**'valid'**,**'id'**:id})  
  
@app.route(**'/registration'**,methods=[**'post'**])  
**def** registration():  
 **try**:  
 fname = request.form[**'firstname'**]  
 lname = request.form[**'lastname'**]  
 email = request.form[**'email'**]  
 gender = request.form[**'gender'**]  
 place = request.form[**'place'**]  
 post = request.form[**'post'**]  
 pin = request.form[**'pin'**]  
 district = request.form[**'district'**]  
 phone = request.form[**'phone'**]  
 uname = request.form[**'username'**]  
 pswd = request.form[**'password'**]  
 qry = **"insert into login values(null,%s,%s,'customer')"** val = (uname,pswd)  
 id = iud(qry,val)  
 qry = **"INSERT INTO `user` VALUES(NULL,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"** val = (id,fname,lname,email,gender,place,post,pin,district,phone)  
 iud(qry,val)  
 **return** jsonify({**'task'**:**'valid'**})  
 **except** Exception **as** e:  
 print(e)  
 **return** jsonify({**'task'**:**'failed'**})  
  
@app.route(**'/search\_contractor'**,methods=[**'post'**])  
**def** search\_contractor():  
 print(request.form)  
 service = request.form[**'service'**]  
 latitude = request.form[**'latitude'**]  
 longitude = request.form[**'longitude'**]  
 print(latitude,longitude)  
 qry=**"SELECT `register`.\*,`location`.\*, (3959 \* ACOS ( COS ( RADIANS(%s) ) \* COS( RADIANS( `location`.lattitude) ) \* COS( RADIANS( `location`.longitude ) - RADIANS(%s) ) + SIN ( RADIANS(%s) ) \* SIN( RADIANS( `location`.lattitude ) ))) AS user\_distance FROM`location` JOIN `register` ON `location`.user\_id=`register`.lid WHERE `register`.service=%s HAVING user\_distance < 31.068"** *#qry="SELECT `register`.\*,`location`.\*, (3959 \* ACOS ( COS ( RADIANS(%s) ) \* COS( RADIANS( `location`.lattitude) ) \* COS( RADIANS( `location`.longitude ) - RADIANS(%s) ) + SIN ( RADIANS(%s) ) \* SIN( RADIANS( `location`.lattitude ) ))) AS user\_distance FROM`location` JOIN `register` ON `location`.user\_id=`register`.lid WHERE `register`.service=%s HAVING user\_distance"* val=(latitude,longitude,latitude,service)  
 res = androidselectall(qry,val)  
 print(res)  
 **return** jsonify(res)

# CHAPTER-5

# TESTING

**5. SYSTEM TESTING**

For software that is newly developed, primary importance is given to testing the system. It is the last opportunity for the developer to detect possible errors in the software before handing over it to the customer. Testing is the process by which the developer will generate a set of data, which gives the maximum probability of finding all types of errors that can occur in the software.

The various steps in testing the system can be listed below:

1. Running the program to identify any errors that might have occurred while feeding the program into the system.
2. Applying the screen formats to regulate users to an extent, so that the screens are comprehensible to the user.
3. Presenting the formats to the administration to obtain approval and check if any modification has to be done. Obtaining feedback from users and analyzing the scope for improvement.
4. Checking the data accessibility from the data server and whether any improvement is needed or not.

**The following are the types of Testing:**

* Unit Testing
* Integration Testing
* System Testing
* Validation Testing

**5.1 UNIT TESTING**

Unit testing is carried out screen-wise, with each screen being identified as an object. Attention is diverted to individual modules, independently to one another to locate in coding and logic.

In unit testing,

* The module interface is tested to ensure that information properly flows into and out of the program under test.
* Local data structures are examined to ensure that data stored temporarily maintains its integrity during all steps in algorithm execution.
* The boundary condition is tested to ensure that the module operates properly at boundaries established to limit or restrict processing.
* All independent paths through the control structures are executed to ensure that all statements in the module have been executed at least once.
* Error handling paths are also tested.

**TEST CASES**

**Login form**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test scenario** | **Expected result** | **Observed result** | **Result** |
| 1. | Enter the wrong user name and password | Display login form again with an invalid message. | Message displayed. | Pass |
| 2. | Enter the correct user name and wrong password | Display login form again with invalid message. | Message displayed. | Pass |
| 3. | Enter correct user name and correct password of administrator or user. | Administrator or user can login to the system. | Login successful is returned | Pass |
| 4. | Press login button without filling the user name and password. | Display warning message to fill the details. | Warning message displayed | Pass |

**Student Registration form**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No:** | **Test Scenario** | **Expected Result** | **Observed Result** | **Result** |
| 1. | Form displayed. | Display the registration form. | Form loaded | Pass |
| 2. | Enter the name in integers. | Display an invalid message. | Invalid message displayed. | Pass |
| 3. | Enter the mobile number in characters. | Display an invalid message. | Invalid message displayed. | Pass |
| 4. | Enter the mobile number more than and less than 10  integers. | Display an invalid message. | Invalid message displayed. | Pass |
| 5. | Click the register  button without  filling the details. | Display a warning  message to fill the details. | Warning message displayed. | Pass |
| 6. | Click on register button with filled fields. | Accept the details. | Registration successfully done. | Pass |
| 7. | Click reset button | Clear all fields to blank. | All fields cleared. | Pass |

**5.2 INTEGRATION TESTING**

Integration testing is a symmetric technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. Unit-tested modules were taken and a single program structure was built that has been dictated by and tested in small segments, where errors were easy to locate and rectify. Each database or table manipulation operation was written as a single program and was tested again with numerous test data to check for its functionality.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Input/procedure** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1. | Check the value pass between different forms are the appropriate  format | Appropriate operations of different forms. | Same as expected. | Pass |

**5.3 SYSTEM TESTING**

System testing is used to test the entire system (Integration of all modules). It also tests to find the discrepancies between the system and the original objective, current specification and system documentation. The entire system is checked to correct deviations to achieve correctness

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Input/procedure** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1. | Check whether indented output is  obtained. | All operations are carried out  properly. | Same as expected. | Pass |

**5.4 VALIDATION TESTING**

After integration testing, the software is completely assembled as a package, interfacing errors have been uncovered and corrected and a final series of software tests begin validation test has been conducted one of the two possible conditions exists. One is the function or performance characteristics confirm to specification and are accepted and the other is a deviation from the specification is uncovered and a deficiency list is created.

**CHAPTER-6 IMPLEMENTATION**

**6.1 SYSTEM IMPLEMENTATION**

System implementation is the stage where the theoretical design is turned into a working system. The system can be implemented only after thorough testing is done and if it is found to work according to specifications. The following methods were undergone.

* Testing developed programs with updating.
* Correction of errors identified.
* Creating the tables of the system with actual data.
* Making necessary changes with actual data.
* Doing a parallel run of the system to find out any errors identified and to correct them.
* Training of user personnel.

The implementation method used to implement the “E-CONTRACT” is a parallel run. That is, the new system will work parallel to the existing system. The new system will replace the existing system completely. E-CONTRACT is developed as a web application, as usual, some web development technologies are used in the implementation of the project. The language selected to program this software in Python. The reason for selecting this language is a simple and powerful language that was specially developed to create a web application.

# CHAPTER-7 CONCLUSION

**7.1 CONCLUSION**

The system that I developed was implemented and tested with real data and were found to be error free. Also it is found the system will work successfully. I tried to make the system maximum user friendly. System is protected from any unauthorized access by giving user name and password during login process.

All the necessary validations are carried out in this project, so that any kind of users can make use of this software. In this document we present our understanding of what needs to be accomplished in the project along with the budget and time estimation.

**CHAPTER-8 REFERENCES**

**8 REFERENCES**

**8.1. REFERENCE BOOKS**

* Software engineering 5th edition by Roger S Pressman, McGraw-Hill publication 1987.

**8.2. WEBSITE**

* [www.stackoverflow.com](http://www.stackoverflow.com)
* [www.w3school.com](http://www.w3school.com)
* <https://www.codeproject.com>
* https://www.tutorialspoint.com
* www.youtube.com
* www.github.co

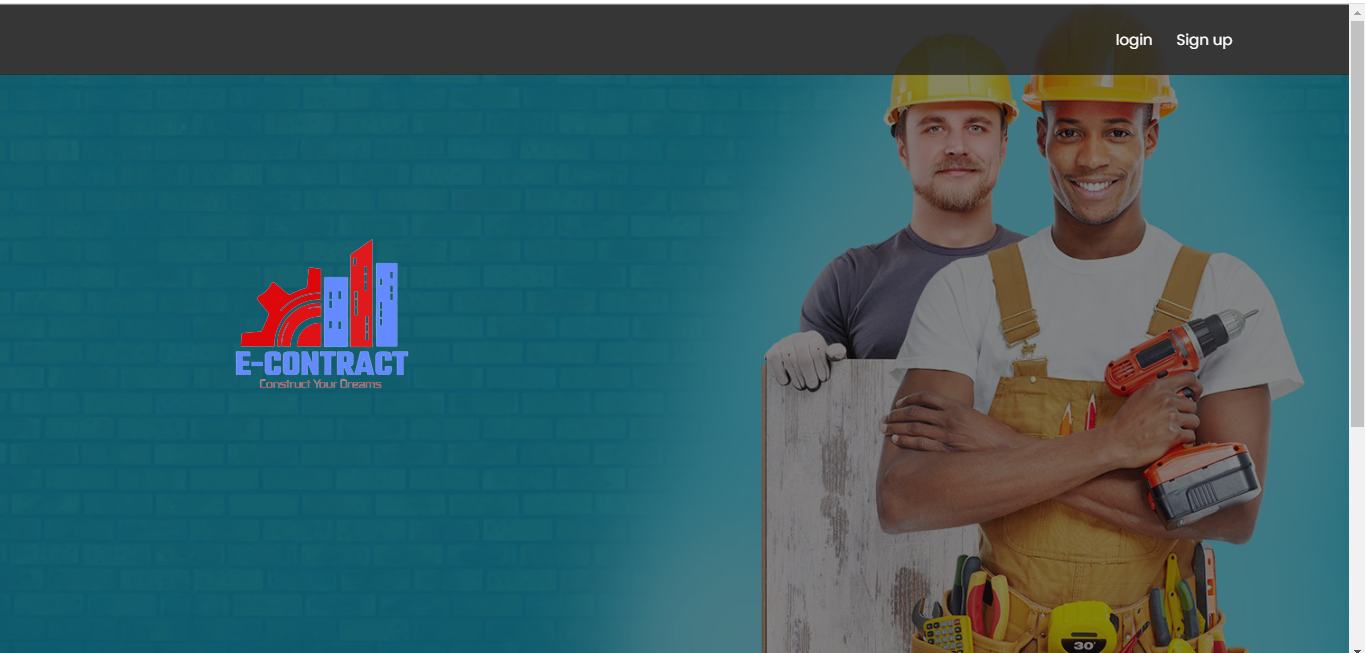
**CHAPTER-9**

**APPENDIX**

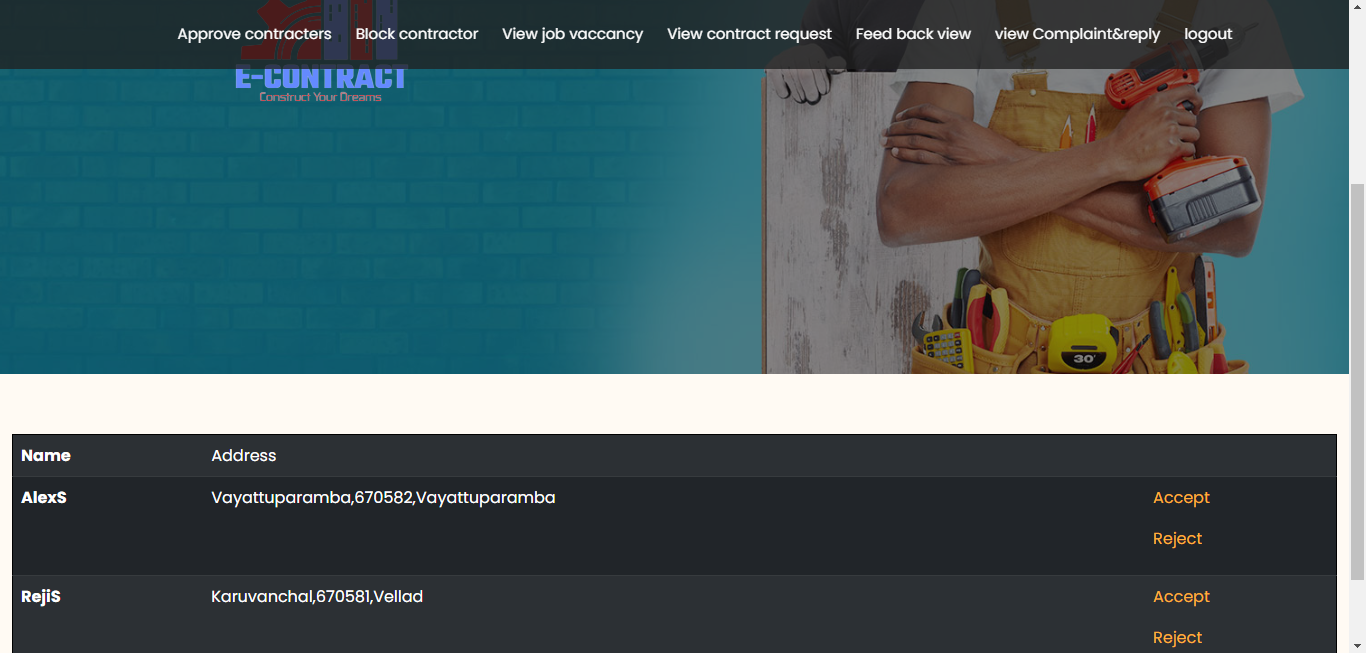
# 

# APPENDIX

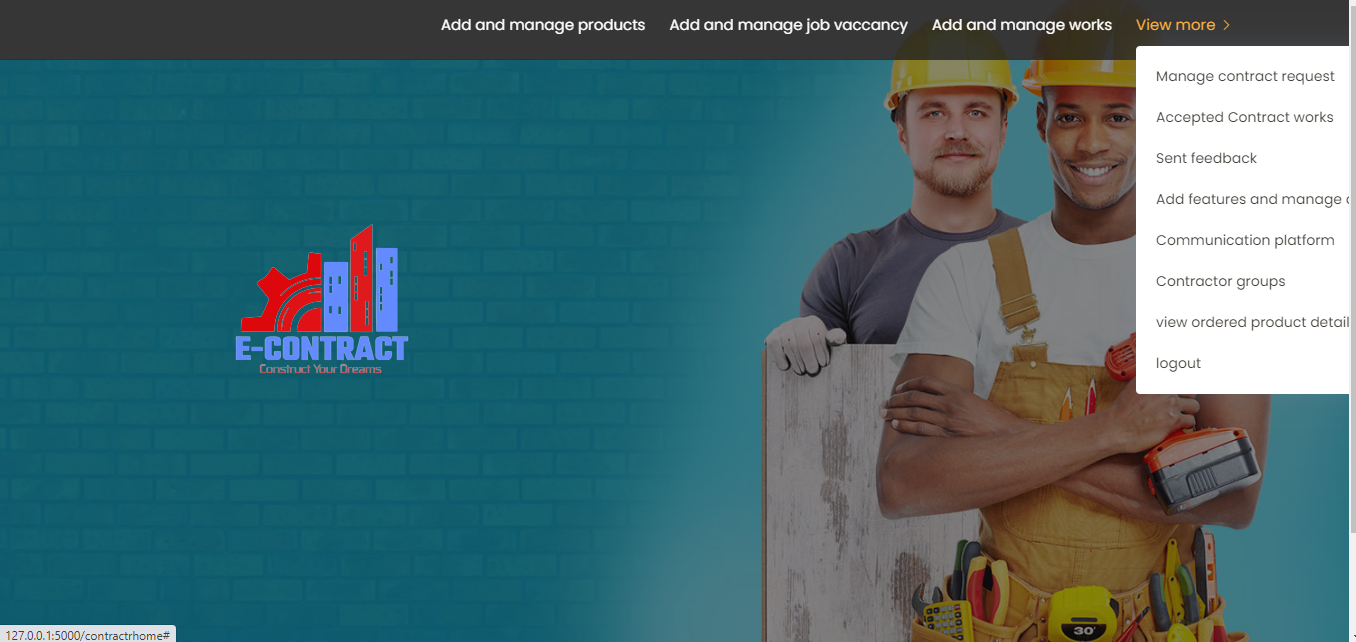
##### **Login page**

****

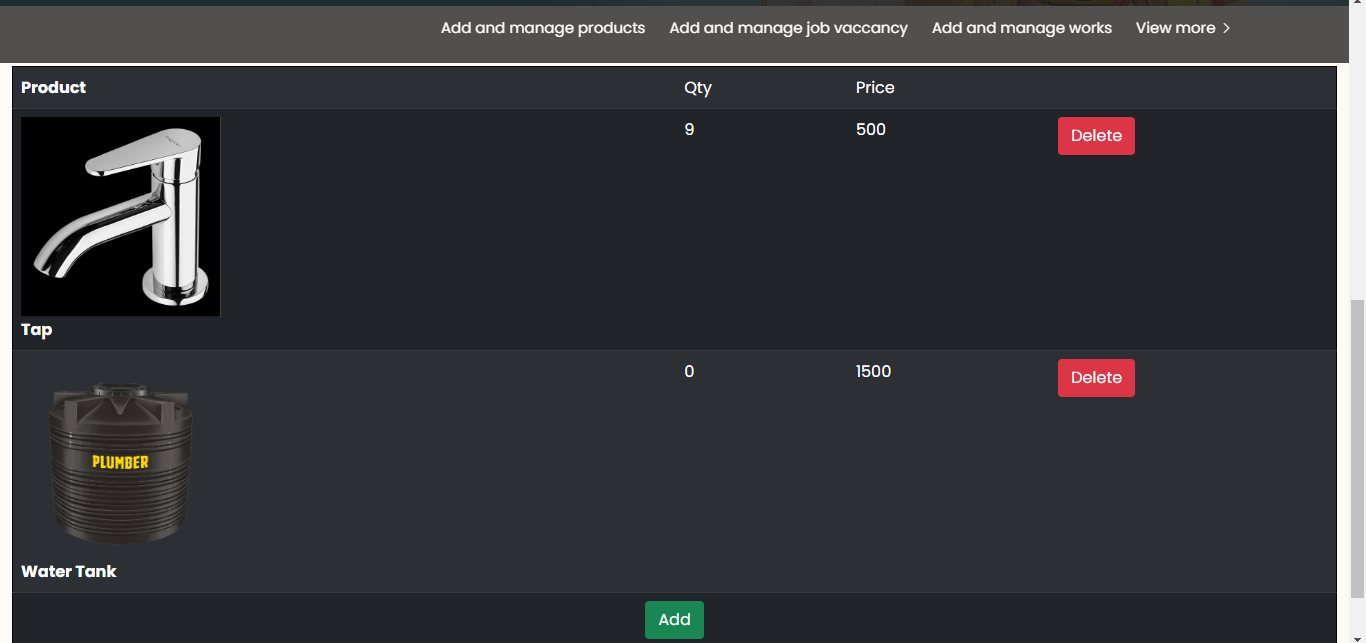
**Approve Contractors**

****

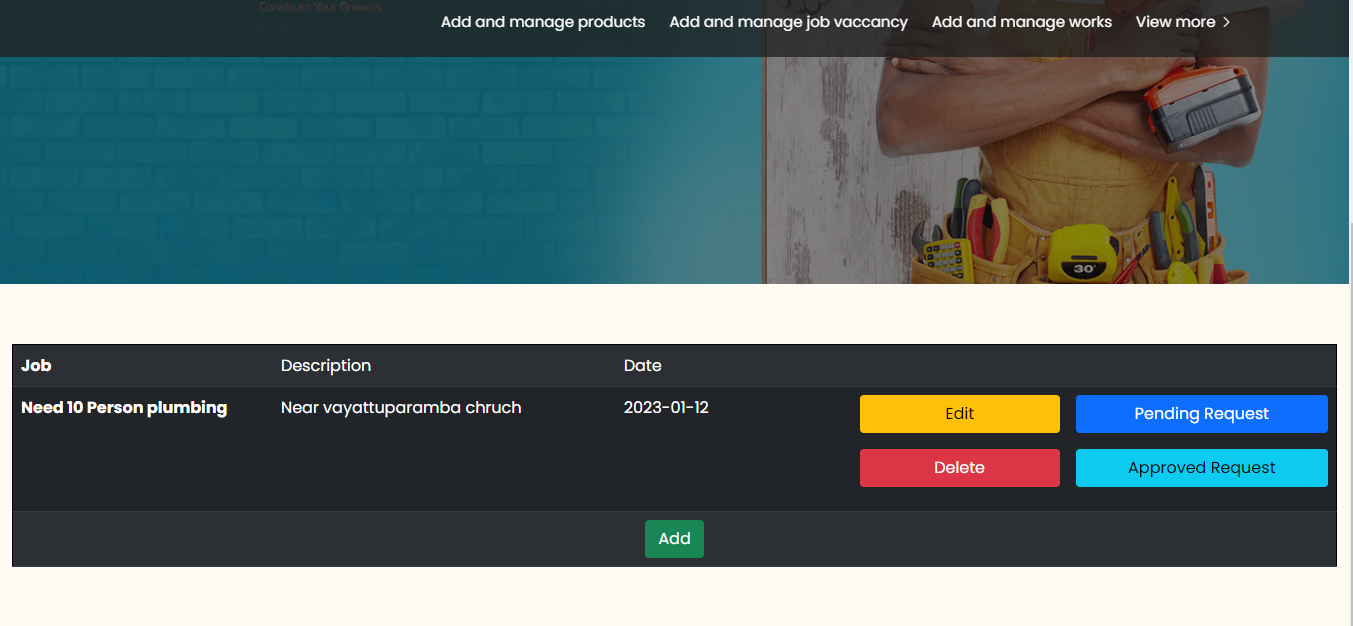
**Contractor home page**

****

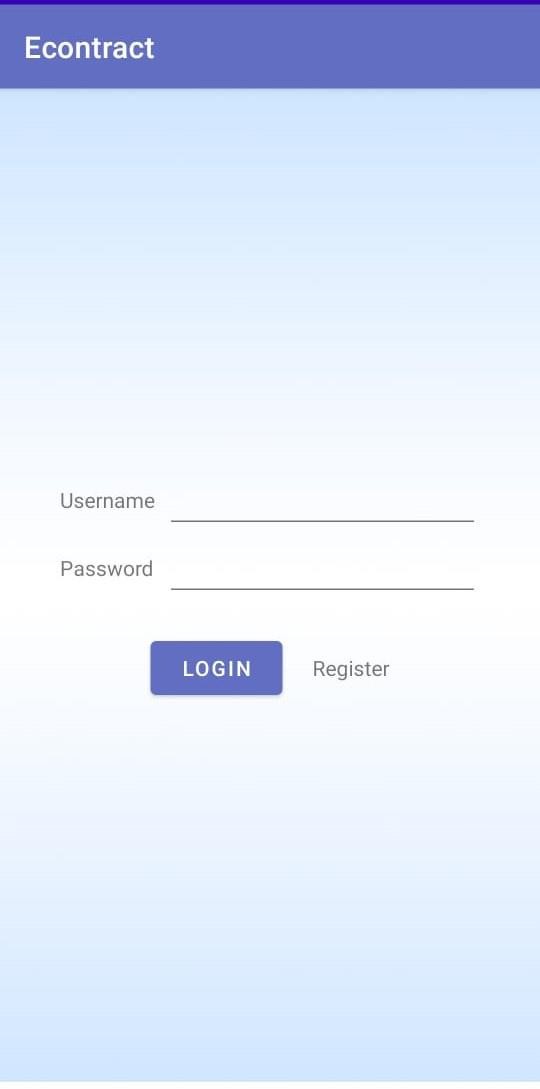
**Add and manage product**

****

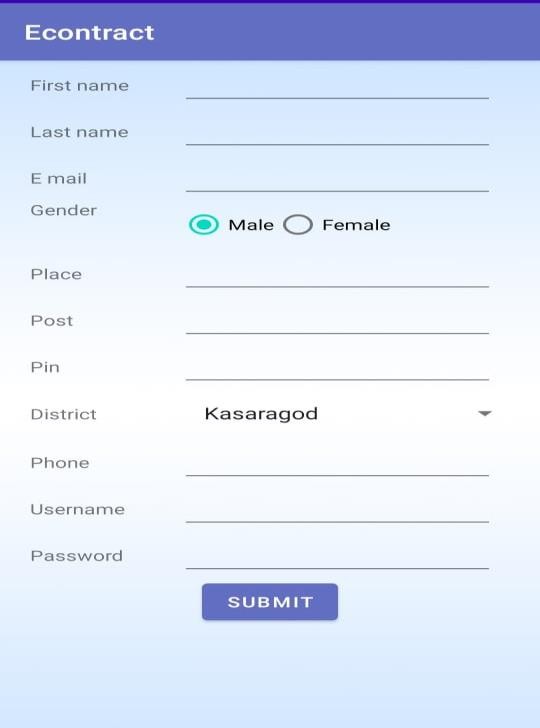
**Add &Manage Job vacancy**



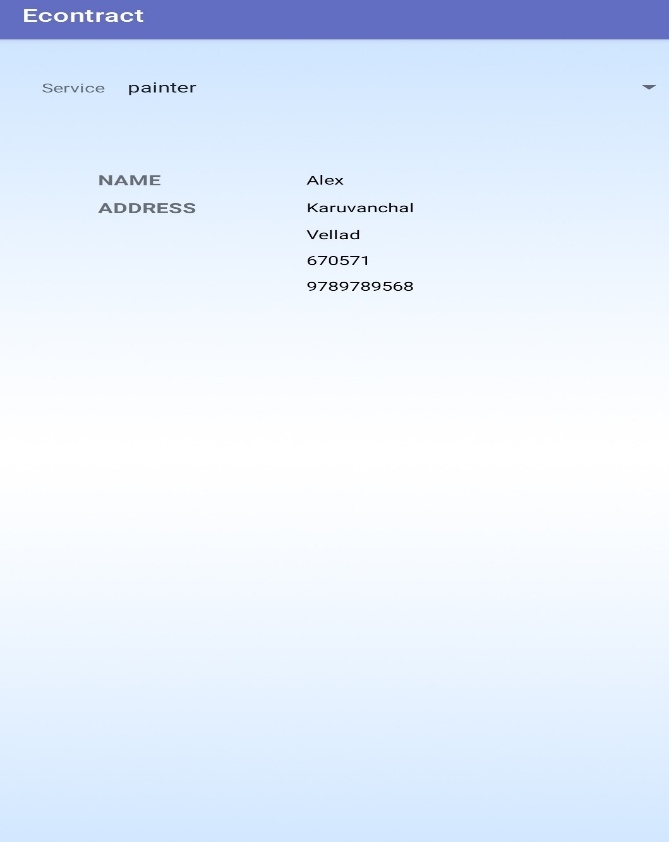
**User login**



##### **User registration**



**Search nearest contractor**

****

**View vacancy and apply job**

****