

CONTENTS

Abstract	8
List of Figures	9
List of Tables	10
1. Introduction.....	11
1.1 Background.....	12
1.2 Project Profile.....	12
1.3 Environmental characteristics.....	12
1.3.1 Hardware	12
1.3.2 Software.....	12
2. Problem Definition and Methodology.....	13
2.1 Problem Definition.....	14
2.2 Objectives.....	14
2.3 Methodology.....	14
2.4 Scope of the project.....	15
2.5 Constraints.....	15
3. Requirement Analysis and Specification.....	16
3.1 Requirement Analysis/Literature Review.....	17
3.2 Existing System.....	17
3.3 Proposed System.....	18
3.4 Requirement Specification.....	18
3.4.1 Functional Requirements.....	18
3.4.2 Non-Functional Requirements.....	19
3.5 Feasibility Study.....	20
3.5.1 Technical Feasibility.....	20
3.5.2 Economical Feasibility.....	21
3.5.3 Operational Feasibility.....	21
4. System Design.....	22
4.1 Users of the System.....	24
4.2 Modularity Criteria.....	24

4.3 Architecture Diagram.....	25
4.3.1 Data Flow Diagram.....	25
4.4 Database Design.....	29
4.4.1 List of Entities and Attributes.....	32
4.4.2 E-R Diagram.....	38
5. Development	41
5.1 Tools.....	42
5.2 Modules.....	46
6. Validation Criteria and Implementation.....	48
6.1 Test Plans.....	49
6.2 Classifications of Testing.....	49
6.3 Implementation.....	50
Conclusion.....	51
Code.....	52
User Interface.....	121

ABSTRACT

One of the problems that we face nowadays is that we cannot complete a construction of a building without middleman involved. We all search for architects, designers and contractors though the means of family friends and relatives. This is a problem. There is very limited source to find the architects, designers and contractors. This leads to the build-up of extra amount to the construction cost, which is already expensive. Not everybody would be financially stable in constructing a building, or a home. The main objective of this project is to build a platform where the user can find everything that is required to construct a building, from architects to contractors, under one roof.

LIST OF FIGURES

SL No:	Figures	Page no.
1	Level 0 DFD	27
2	Level 1 DFD	27
3	Level 2	28
4	Database Design	31
5	ER diagram of the system	40

LIST OF TABLES

I. List of entities and attributes of database

1. tbllogin.....	32
2. tblallocation.....	32
3. tblarchitect.....	33
4. tblcontractor.....	33
5. tblcustomer.....	34
6. tbldesigner.....	34
7. tbldesignrequest.....	35
8. tblplan.....	35
9. tblrequirement.....	36
10. tblvideo.....	37
11. tblwork.....	37

INTRODUCTION

1.INTRODUCTION

1.1Background

RAD – Realistic Architectural Design is a project that allow people to search Architects, Designers, and Contractors for the construction. All the interactions are done remotely through the users browser window. Users can use their email and password to register to the site. They can assign architects, designers, and contractors. The admin approve the Architects, Designers, and Contractors based on their proof and qualification. Users can see the Architects, Designers and Contractors after submitting their requirements. The plans, videos and work status are updated in the site for the user. RAD is a useful project for people looking to cut cost and also a platform for architects, designers, and contractors to get more customers.

1.2 Project Profile

We create a platform that enable the user to search and find the required people for construction. It also creates a platform for contractors and architectures to get more customers based on their rating and quality of previous works. If the user is further interested on viewing how the Elevation looks like in the 3D version the work can be forwarded to a 3D designer. This reduce the cost of middleman and time spent to find the suitable people for the construction project.

1.3 Environmental Characteristics

1.3.1 Hardware Configuration

Processor	: Intel Core i5 or above.
RAM	: 4 GB
HDD	: 100GB
Key Board/Mouse	: Any standard Key Board/Mouse
Monitor	: Any standard Monitor

1.3.2 Software Configuration

Front End	: HTML, PHP
Back End	: MySQL
Operating System	: Windows 7 OR above.
Browser	: Google Chrome or Microsoft Edge.

PROBLEM DEFINITION AND METHODOLOGY

2. PROBLEM DEFINITION AND METHODOLOGY

2.1 Problem Definition

Currently there are individual sites for Contractor, Architect, and Designers in western countries, and very few in India. Right now there's no site that provide all the facility combined together. Visiting an architect for plan is expensive and also may not be always productive, this leads to build up in cost and also in wastage of time.

2.2 Objectives

The main objective of this project is to build a platform where the user can find everything that is required to construct a building, from architects to constructors, under one roof. This also creates a platform for designers, contractors and architectures to get more customers based on their rating and quality of previous works. This also reduce the cost of middle men and time spent to find the suitable people for the construction project.

2.3 Methodology

The working methodology of this project is that a registered user can request to get an elevation of a house based on his wish. User can filter out the architects based on the nearby location, rating, and experience. The user can select any architect and provide them instruction. The architect work on the instruction and create a blueprint of the work. If the user is further interested on viewing how the Elevation looks like in the 3D version the work can be forwarded to a 3D designer. The selection of the designer can also be based on the above mentioned filtering criteria. After viewing the design the user can call for an estimate from the contractors. After analysing the blueprint and the instruction from the user the contractors sent an estimate to the user. The estimate is secured using encryption. The public key for decrypting will be handed over to the requester only. From the estimate the user can create a contract with a contractor only with the knowledge of the System Administrator. The contract generated will also be highly secured.

There are different modules in our project:

1. User Management Module.

Admin and other Users' login to the site using a common login page. Users use the username and password provided at the time of registration to login to the site.

2. Requirement Module.

In this module the user can fill out the requirement details of the house. The requirement is then send to the architect and designer for further planning process. After analysing the requirements, architect and designer construct plans based on the user needs.

3. Plan Module.

In this module with the data provided in requirement module, the architect builds a plan and upload to the user.

4. 3D Video module.

In this Module the Designer create a 3D model based on the Architectural plan.

5. Work Modul.

In this module the user view and approve or reject the contractors work.

2.4 Scope of the Project

This project would help a lot of people to find the right people to construct a building, and have an estimate of the construction. This site also acts as a platform for the freelance workers to find a job. The contribution this project would make is that it would eliminate the need of middle men in the construction field, also reduce the cost of construction, they also can find the architects, civil engineer, 3D designers and contractors under a single roof.

2.5 Constraints

A registered user can request to get an elevation of a house based on his wish. User can filter out the architects based on the nearby location, rating, and experience. The user can select any architect and provide them instruction. The architect work on the instruction and create a blueprint of the work. If the user is further interested on viewing how the Elevation looks like in the 3D version, the work can be forwarded to a 3D designer. After viewing the design the user can call for an estimate from the contractors. After analysing the blueprint and the instruction from the user the contractors sent an estimate to the user. From the estimate the user can create a contract with a contractor only with the knowledge of the System Administrator.

REQUIREMENT ANALYSIS AND SPECIFICATION

3. REQUIREMENT ANALYSIS AND SPECIFICATION

3.1 Requirement Analysis/Literature review

Requirement analysis results in the specification of software's operational characteristics; indicates software's interface with other system elements and establishes constraints the software must meet. Requirement analysis allows the software engineer to elaborate on basic requirements established during earlier requirement engineering tasks and build models that depict user scenarios, functional activities, problem classes and their relationships, system and class behavior and flow of data as it is transformed. Requirement analysis provides the software designer with representation of information, function and behavior that can be translated to architectural, interface and component level designs. Finally the analysis model and the requirement specification provide the development and the customs with the means to access quality once the software is built. The main requirement of this project is to eliminate the communication gap between a person who wants to build a home and the people needed for the process.

3.2 Existing System

Currently there are individual sites for Contractor, Architect, and Designers in western countries, and very few in India. Right now there's no site that provide all the facility combined together. Visiting an architect for plan is expensive and also may not be always productive, this leads to build up in cost and also in wastage of time. The user may not get the plan they like and seeing architects after architects is a tiring process, not all would be able to afford appointments with number of architects.

Limitations of existing system

1. It takes so much time for meeting various architects and coming down to a plan.
2. Consumption of working days.
3. It leads in cost and the overall cost of the construction increases.
4. Does not provide all the usefull people from one source.
5. Multiple source for different people.
6. Not easy to keep track of workers and their payment.

3.3 Proposed system

The proposed system is designed to eliminate all the disadvantages of the existing system. It is designed for keeping in mind all the drawbacks of the present system. This system allows the user to meet with the Architect, Designer and Contractor under a single website and even contact them if needed through email or contact number. This eliminates the process of going over to meet the Architects, Designer and wasting time, or face the peer pressured recommendations, also can assign a contractor for the construction business. All of the process happen under one roof, this all comes to this platform, this saves time and money of the customers, also this platform open opportunity for Architects, Designer and Contractors to get more customers and improve their career.

Advantages of proposed system

1. Saves time.
2. Saves money.
3. Reduce the stress of meeting a bunch of architects and designers.
4. The basic needed people of construction under one roof.

3.4 Requirement Specification

3.4.1 Functional Requirement

In software engineering and system engineering, a functional requirement defines a function of a system or its component. A function is described as a set of inputs, the behaviour, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioural requirements describing all the cases where the system uses the functional requirements are captured in use cases. Functional requirements are supported by non-functional requirement (also known as quality requirements), which impose constraints on the design or implementation (such as performance requirements, security, or reliability). Generally, functional requirements are expressed in the form "system must do <requirement>", while non-functional requirements are "system shall be <requirement>". The plan for implementing functional requirements is detailed in the system design. The plan for implementing non-functional requirements is detailed in the system architecture.

The main functional requirements are:

1. The administrator will be given more power than other users.
2. The site can be accessed any time. The users use email and the password provided at the time of registration. By entering email and password, users can access the system any time.
3. Users can easily login and can update details, view notification and can perform any doubts.

3.4.2 Non-Functional Requirement

Non-functional requirements are sometimes defined in terms of metrics (something that can be measured about the system) to make them more tangible. Non-functional requirements may also describe aspects of the system that don't relate to its execution, but rather to its evolution over time (e.g. maintainability, extensibility, documentation, etc). A functional requirement describes what a software system should do, while non-functional requirements place constraints on how the system will do so. Non-functional requirements - can be divided into two main categories: Execution qualities, such as security and usability, which are observable at run time. Evolution qualities, such as testability, maintainability, extensibility and scalability, which are embodied in the static structure of the software system. And it defines the system properties and constraints that arise through user needs, because of the need of the budgeted constraints or organizational policies, or because of the need for interoperability with other software or due to the external factors such as safety regulation privacy registration and so on.

The non-functional requirement of RAD are:

1. Functionality – It provides good functionality.
2. Usability – Requirements can be used effectively.
3. Software should be user friendly.
4. Maintainability – It should be maintainable and able to add new features

3.5 Feasibility Study

Feasibility analysis is an analysis of possible alternative solution to a problem and a recommendation on the best alternative. Before the request is to be approved, it has to be checked whether the system is feasible or not with respect to the following areas. It is a test of a system proposal according to its workability, impact on the organization, ability to meet user needs, and effective use of resources. The objective of feasibility study is not to solve the problem but to acquire a sense of its scope.

Feasibility analyses involve 8 steps:

1. From a project team and appoint a project leader.
2. Prepare system flow charts.
3. Enumerate potential design systems.
4. Describe and identify characteristics of design systems.
5. Determine and evaluate performance and cost effectiveness of each design system.
6. Weigh system performance and cost data.
7. Select the best design system.
8. Repair and report final project directive to management.

Three key considerations are involved in the feasibility analysis:

1. Technical
2. Economical
3. Operational

3.5.1 Technical Feasibility

It deals with hardware as well as software requirements. In technical feasibility it understands the organization ability to construct the proposed system. The system needs computer with all the latest equipment. The system should be supported with efficient storage mechanisms and high speed computers. The software tools is also needed, these are not available with the current system. With the current infrastructure of the institution and availability of the equipment in the current scenario, the project can be considered technically feasible. RAD is developed by HTML and PHP so it is technically feasible.

3.5.2 Economical Feasibility

Economic analysis is the most frequently used method for evaluating the effectiveness of the software. Its more commonly known as cost benefit analysis, the procedure to determine the benefits and savings that are expected from a project system and compare them with costs. The purpose of accessing economical feasibility is to identify the financial benefits and cost associated with the development of the project. This project aims at reducing the retrieval time, efforts and cost. The RAD is economically feasible because this system uses less expensive for storing the user details. So the system is economically feasible.

3.5.3 Operational Feasibility

In operational feasibility it is check if the system will work. Operational feasibility is reviewed in the early stages of project planning. The project or software which is developed has a interactive, user friendly interface. So operational difficulties are almost eliminated. The user interface is designed in such a way that it gets comfortable and understandable for non-technical person to operate easily. And RAD is operationally feasible.

SYSTEM DESIGN

4. SYSTEM DESIGN

The system design is the second of the four system development life cycle phase. It is the phase in which the detailed design of the system selected in the study phase accomplished and the users oriented performance specification is converted into a technical design specification. The principle activity performed during the design phase includes the general system design and the design of all outputs, input design phase reported and user review. At this point in the system development life cycle, the problems has been identified, alternative solutions have been studied, and also a management review has been ended with an authorization to be proceed with a recommended solution. The first design step, the system design involves the selection of those system functions to be performed by either the people in the system, the equipment, or computer programs. These functions are usually identified by decomposing the data flow diagrams into low levels, more detailed DFD's, or drawing expanded system flow charts in the course of the design phase, the performance specification is expanded into design specification.

The goal of the design process is to produce a model which can be used later to develop system. Produced model is called design of the system. The user oriented base line document prepared in the study phase becomes a base line document oriented to the needs of the programmers and other professional personals, which are actually develop the system. A smooth traction for the study phase to the design phase is necessary because the design phase continue the activities begun in the earlier phase. However, the project becomes enlarged in scope, and the personal are added. The system design phase provides details on how to meet the requirements.

This phase converts the alternative solutions into logical and then physical system solution. In this phase must design all the aspect from input and output to reports, database and computer process. There are two types of design possible. They are logical design and physical design. In logical design, concentrate on business aspect of system to be obtained to high level of specificity. In physical system, analyst will design the various part of the system to perform physical operations. The analyst must design the following elements of a system, that are files and database, input design, output design, forms and reports, dialogue an interface, system and program structure, distributed system.

System design consists of two major steps:-

1. Initiation of the detailed design by allocating system function between manual tasks, the equipment functions and computer program functions
2. The identification of the test requirements for the system and each of the part.

4.1 Users of the system

The main users of this system are admin and users. Our project is mainly intended for the people who would like to build a simple homes with minimal design. The main objective of our project is to reduce the time user spends on finding the right people for the work.

4.2 Modularity criteria

In software engineering, modularity refers to the extent to which a software/Web application may be divided into smaller modules. Software modularity indicates that the numbers of application modules are capable of serving a specified business domain. Software engineering modularity allows typical applications to be divided into two modules, as well as integration with similar modules, which helps developers use pre written code. Modules are divided based on functionality, and programmers are not involved with the functionalities of other modules. Thus, new functionalities may be easily programmed in separate modules. Software constructed of assemblies of small pieces. Each piece encompasses the data and operations necessary to do one task well.

4.2.1 Criteria for modularity

A modular design method must satisfy:

1. Decomposability: Break a problem into sub-problems connected by simple structures minimize communication between sub problems. Permit further work to proceed separately on each sub problem.
2. Composability: Produce software from reusable plug and play modules Composed software is itself a reusable module .Reusable modules work in environments different from the ones in which they were developed
3. Understandability: Minimize need to understand module context.
Know or examine as few other modules as possible very important for maintenance.
4. Continuity: The smaller the change in specification, the fewer the number of modules that must be changed (edited) and if possible complied .Understandability and continuity related to coupling and cohesion

5. **Protection:** Confine abnormal run time errors to one or a very few modules . Avoid Propagation of error condition to neighboring modules .Exceptions in languages like C++ and Java can be used in an undisciplined manner leading to violations of protection. Exceptions raised in one part of the system should not be handled by a remote part of the system. Without these, we cannot produce modular software.

4.3 Architecture diagram

Software architecture involves the high level structure of software system abstraction, by using decomposition and composition, with architectural style and quality attributes. A software architecture design must conform to the major functionality and performance requirements of the system, as well as satisfy the non-functional requirements such as reliability, scalability, portability and availability.


4.3.1 Data flow diagram (DFD)

Data flow diagrams are used widely for modeling the requirements.DFDs show the flow of data through a system. The system may be a company, an organizational set of procedures, a computer hardware system, a software system or any combination of the proceedings.

The DFD also known as a data flow graph or a bubble chart. The following observations about DFDs are important:

1. All names should be unique. This makes easier to refer items in the DFD. Remember that a DFD is not a flow chart.
2. Arrows in a flow chart represent the order of events; arrows in DFD represent flowing data. A DFD doesn't imply any order of events.
3. Suppress logical decisions (A diamond shape box is used in flow chart to represent decision points with multiple exit paths of which only one is taken).This implies an ordering of events, which makes no sense in DFD.

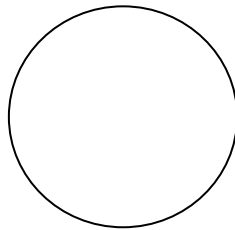
Standard symbols:

1. Data Flow 

Used to connect process to each other, to sources of sinks; the arrow head indicates direction of data flow.

- 2.

Process



Performs some transformation of input data to yield output data.

- 3.

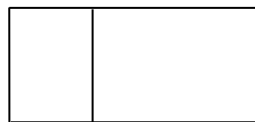
Source/Sink (external entity)



A source of system inputs or sinks of system output.

- 4.

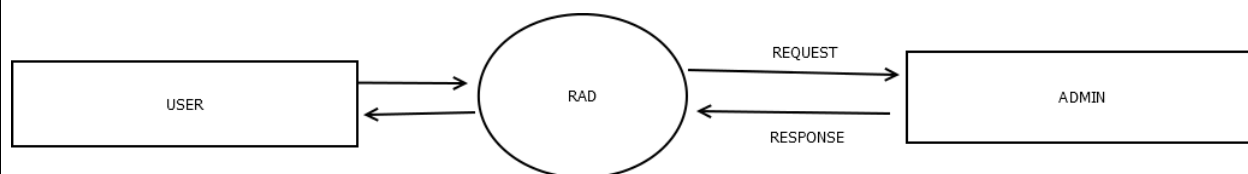
Data store



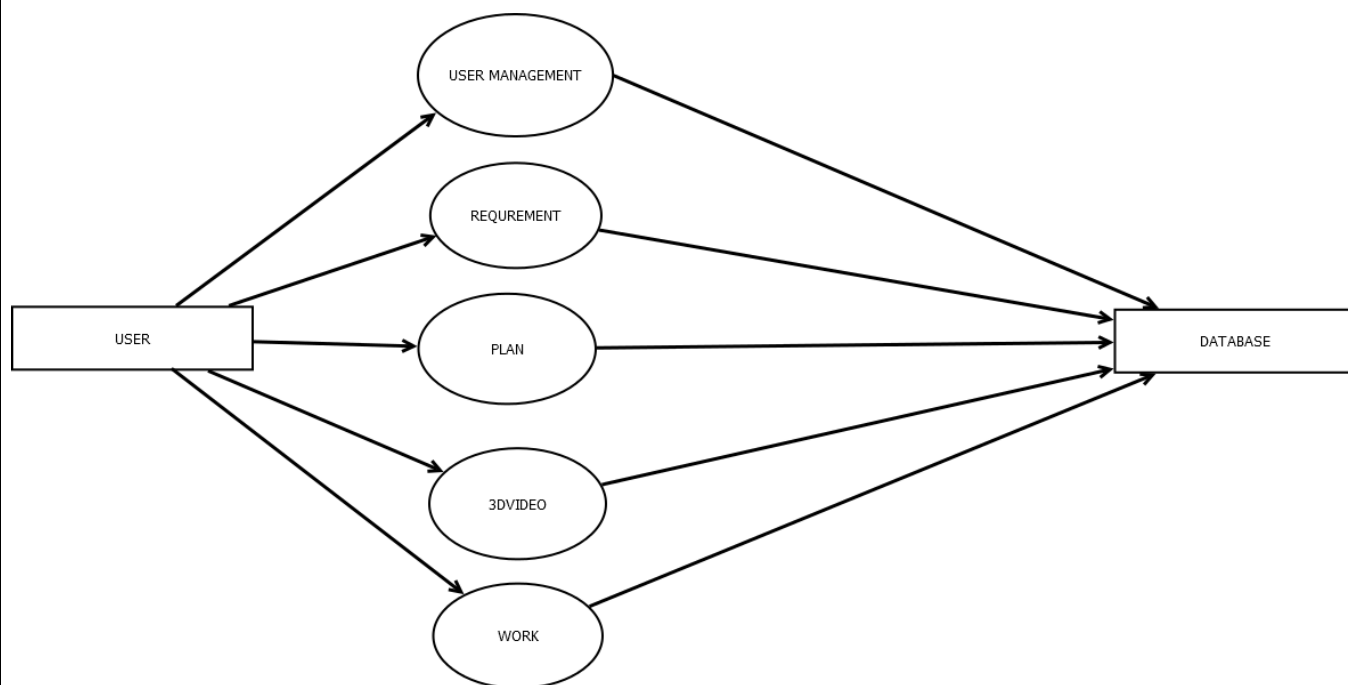
A logical file in which it is used to store table details.

Data-Flow Diagram

Level 0

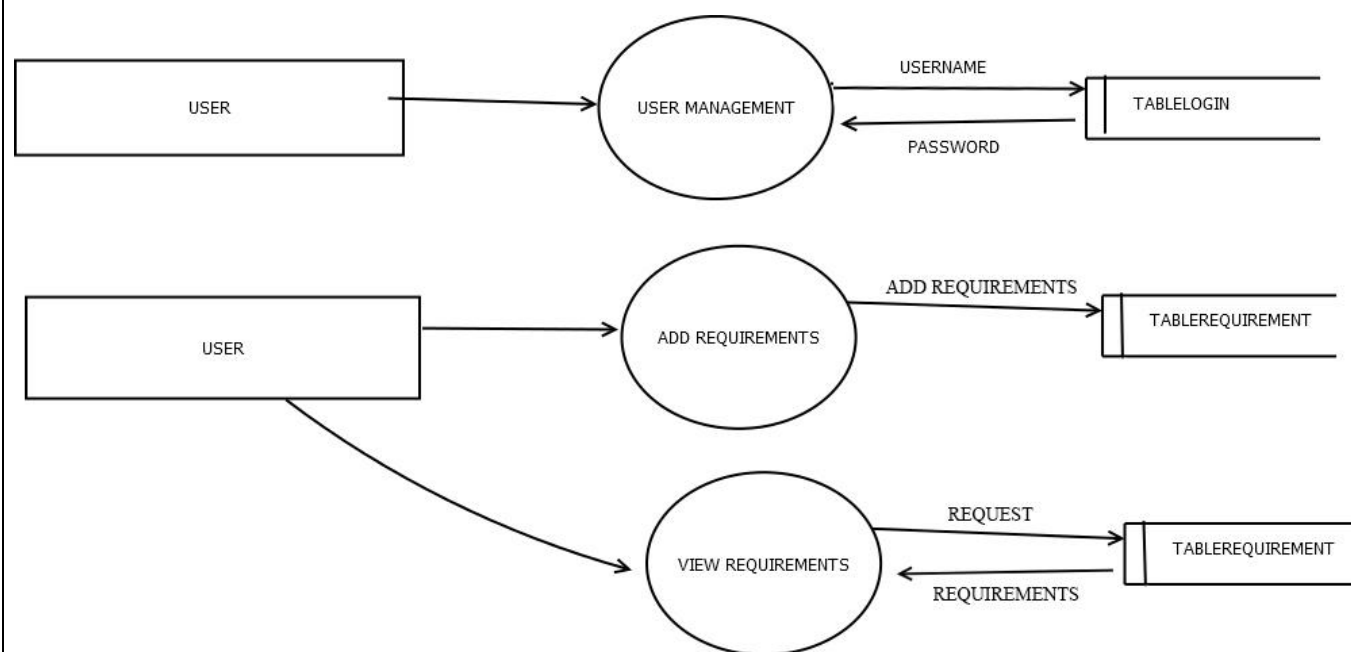


Level 1

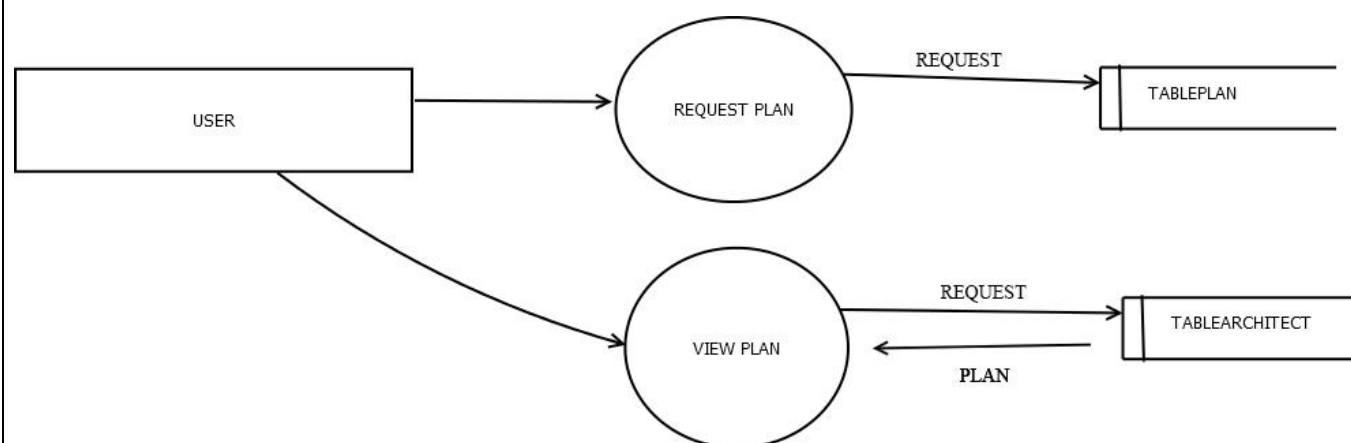


Level 2

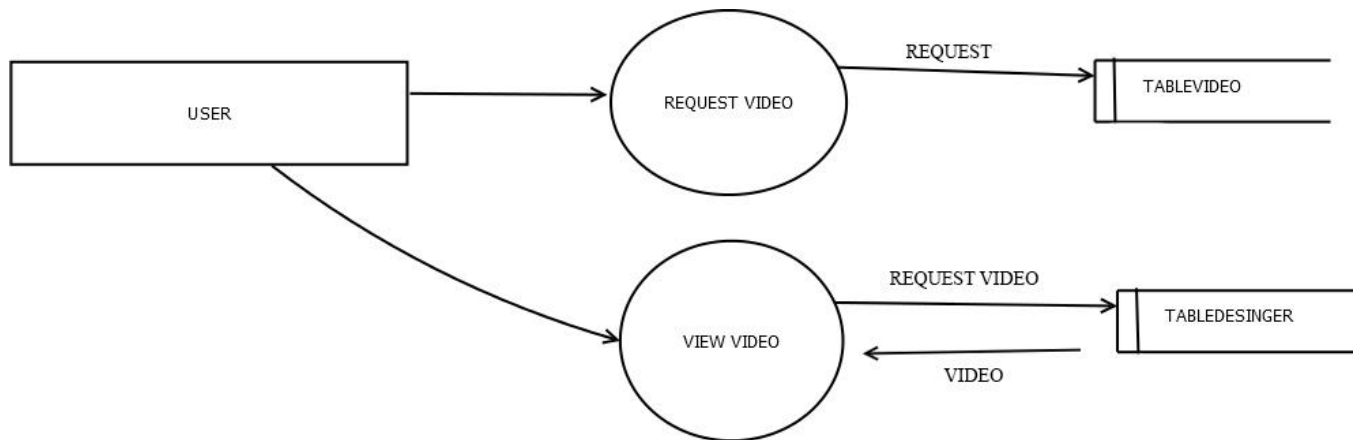
Requirement Filling



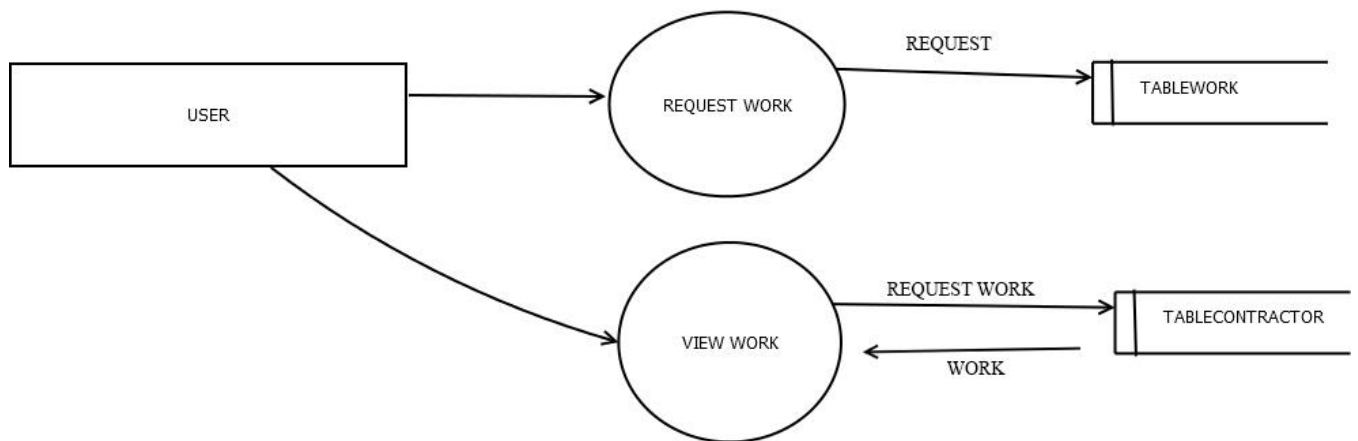
Plan Requesting



Requesting Design / Video



Requesting Work



4.4 Database design

Database design is the process of producing a detailed data model of database. This data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a data definition language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity. 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 (DBMS). Database Management is the key to effective handling of data within an organization and between different functional entities. The storage of all the data in a standardized and streamlined manner is important.

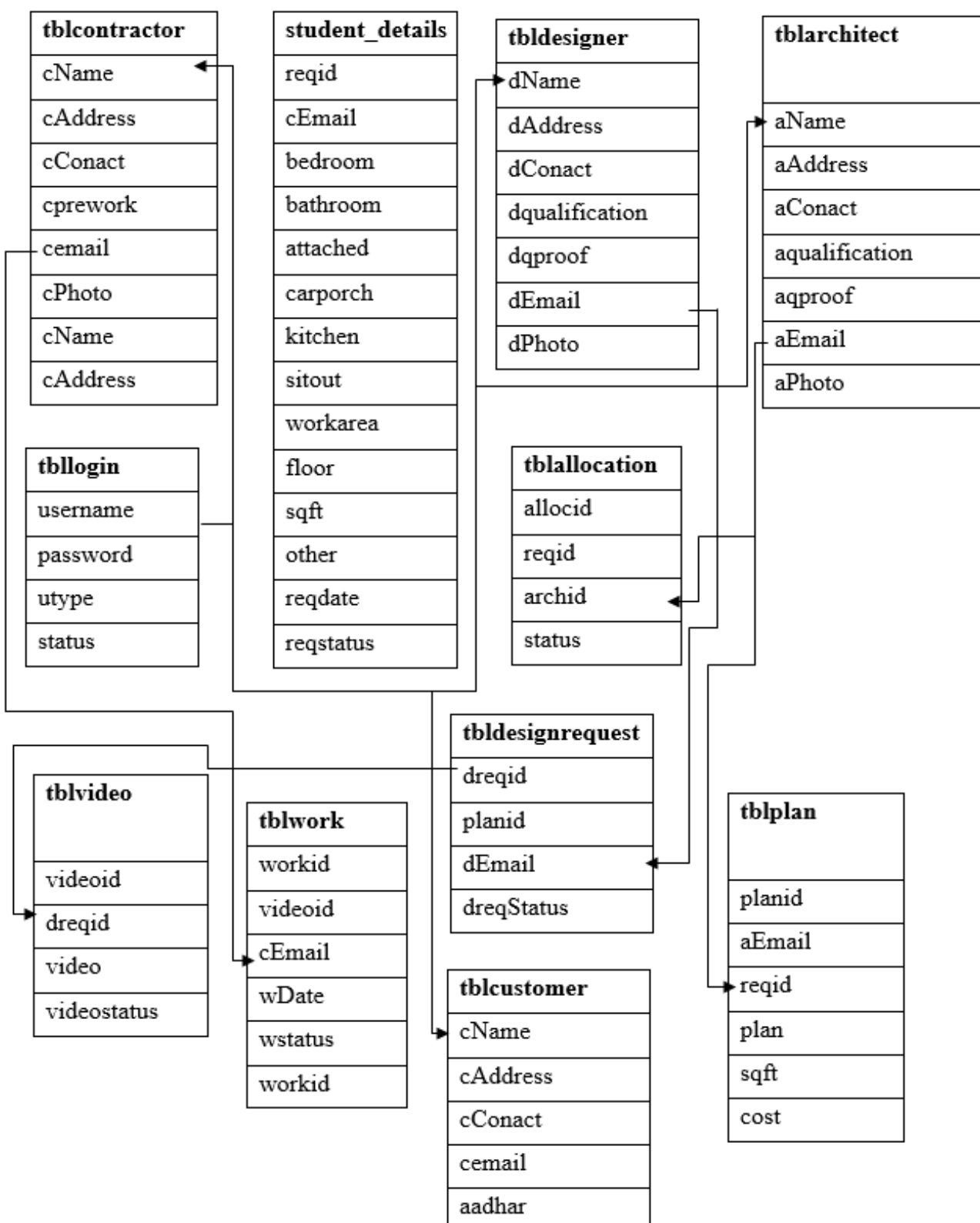
The process of doing database design generally consists of a number of steps which will be carried out by the database designer. Usually, the designer must:

1. Determine the data to be stored in the database.
2. Determine the relationships between the different data elements.
3. Superimpose a logical structure upon the data on the basis of these relationships.

Benefits of a good database design include :

1. Efficient support for complex and interrelated business processes .
2. Lower cost of database ownership.
3. Consistent availability of data to support business operations and decision making.
4. Reduction in redundant data storage.
5. Increased productivity at work.
6. Avoidance of inconsistent data .

Database Name: db_realistic



4.5.1 List of entities and attributes

Table No:1

Table name: **tbllogin**

Table Description: To store logindetails

Column name	Data type	Size	Constraints	Description
username	Varchar	50	primary key	Unique id for users
password	Varchar	50	not null	To store password
utype	Varchar	50	not null	To store usertype
status	Varchar	50	not null	To store current status

Table No: 2

Table name: **tblallocation**

Table Description: To storeallocation details

Column name	Data type	Size	Constraints	Decription
allocid	Integer	11	Primary key	Unique id for allocation
reqid	Integer	11	Not null	To store request id
archid	Varchar	30	Not null	To store architect id
status	Varchar	50	Not null	To store current status

Table No: 3

Table name: **tblarchitect**

Table Description: To store Architects details

Column name	Data type	Size	Constraints	Description
aName	Varchar	50	Not null	To store Architect's name
aAddress	Varchar	100	Not null	To store Architect's address
aConact	Varchar	50	Not null	To store Architect's contact
aqualification	Varchar	50	Not null	To store Architect's qualification
aaproof	Varchar	100	Not null	To store Architect's proof
aEmail	Varchar	50	Primary key	To store Architect's email
aPhoto	Varchar	100	Not null	To store Architect's photo

Table No: 4

Table name: **tblcontractor**

Table Description: To store contractor's details

Column name	Data type	Size	Constraints	Description
cName	Varchar	50	Not null	To store Contractor's name
cAddress	Varchar	100	Not null	To store Contractor's address
cConact	Varchar	50	Not null	To store Contractor's contact
cprework	Varchar	50	Not null	To store Contractor's previous work
cemail	Varchar	100	Primary key	To store Contractor's email
cPhoto	Varchar	100	Not null	To store Contractor's photo

Table No: 5

Table name: **tblcustomer**

Table Description: To store Customer's details

Column name	Data type	Size	Constraints	Description
cName	Varchar	50	Not null	To store Customer's name
cAddress	Varchar	100	Not null	To store Customer's address
cConact	Varchar	50	Not null	To store Customer's contact
cemail	Varchar	100	Primary key	To store Customer's email
aadhar	Varchar	100	Not null	To store Customer's profile

Table No: 6

Table name: **tbldesigner**

Table Description: To store Designer's details

Column name	Data type	Size	Constraints	Description
dName	Varchar	50	Not null	To store Designer's name
dAddress	Varchar	100	Not null	To store Designer's address
dConact	Varchar	50	Not null	To store Designer's contact
dqualification	Varchar	50	Not null	To store Designer's qualification
dqproof	Varchar	100	Not null	To store Designer's proof
dEmail	Varchar	50	Primary key	To store Designer's email
dPhoto	Varchar	100	Not null	To store Designer's photo

Table No: 7

Table name: **tblDesignrequest**

Table Description: To store Designer's request details

Column name	Data type	Size	Constraints	Description
dreqid	Integer	11	Primary key	To store request id
planid	Integer	11	Not null	To store plan id
dEmail	Varchar	50	Not null	To store designer's email
dreqStatus	Varchar	50	Not null	To store designer's request status

Table No: 8

Table name: **tblplan**

Table Description: To store plan details

Column name	Data type	Size	Constraints	Description
planid	Integer	11	Primary key	To store plan id
aEmail	Varchar	100	Not null	To store architect email
reqid	Integer	11	Not null	To store request id
plan	Varchar	100	Not null	To store plan details
sqft	Integer	11	Not null	To store square feet details
cost	Bigint	20	Not null	To store cost details
planstatus	Varchar	50	Not null	To store plan status

Table No:9

Table name: **tblrequirement**

Table Description: To storerequirementdetails

Column name	Data type	Size	Constraints	Description
reqid	Varchar	50	Primary key	To store request id
cEmail	Varchar	50	Not null	To store customer email
bedroom	Varchar	50	Not null	To store bedroom details
bathroom	Varchar	50	Not null	To store bathroom details
attached	Varchar	50	Not null	To store attached details
carporch	Varchar	50	Not null	To store carporch details
kitchen	Varchar	50	Not null	To store kitchen details
sitout	Varchar	50	Not null	To store sitout details
workarea	Varchar	50	Not null	To store workarea details
floor	Varchar	50	Not null	To store floor details
sqft	Varchar	50	Not null	To store sqft details
other	Varchar	50	Not null	To store other details
reqdate	datetime	-	Not null	To store requestdate details
reqstatus	Varchar	50	Not null	To store request status details

Table No:10

Table name: **tblvideo**

Table Description: To storerequirementdetails

Column name	Data type	Size	Constraints	Description
videoid	Integer	11	Primary key	To store video id
dreqid	Integer	11	Not null	To store desinger's request id
video	Varchar	100	Not null	To store video details
videostatus	Varchar	50	Not null	To store video status

Table No:11

Table name: **tblwork**

Table Description: To store work details

Column name	Data type	Size	Constraints	Description
workid	Integer	11	Primary key	To store work id
videoid	Integer	11	Not null	To store video id
cEmail	Varchar	50	Not null	To store contractor email
wDate	date	-	Not null	To store week date
wstatus	Varchar	50	Not null	To store work status

4.5.2 E R diagram

E-R Diagram is a logical representation of the data for an organization and uses 3 main constraints. That is, data entities, relationships and their associated attribute.

Entities:

An Entity is a fundamental thing of an organization about which data may be maintained. An entity has its own identity, which distinguishes it from each other entity. An entity type is the description of all entities to which a common definition and common relationships attributes applied.

Relationship:

A relationship is a reason for associating two entity types. These relationships are some-times called binary relationships. Because they involve two entity types. Some forms of data model allow more than two entity types to be associated.

Degree of Relationship:

The degree of a relationship is a number of entities that participate in that relationship.

The 3 most common relationships in E-R models are:

1. Unary (Degree 1)
2. Binary (Degree 2)
3. Ternary (Degree 3)

Higher degree relationships are possible but they are rarely encountered in practice.

1. Unary Relationship

This is also called recursive relationship. It is a relationship between the instances of one entity type. An instance is a single occurrence of an entity type. There may be many instances of an entity type.

2. Binary Relationship

It is a relationship between instances of two entity types and is a most common type of relationship encountered in E-R diagram.

3. Ternary Relationship

It is a simultaneous relationship between instances of 3 entity types. Relationship may have provided the association of 3 entities. All the 3 entities are many to many participants. There may be one or many participants in a ternary relationship.

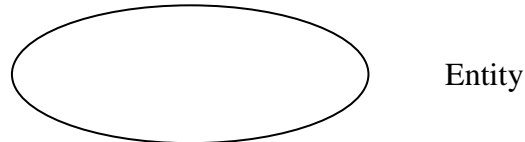
Cardinalities and Optional ties

Suppose that there are two entity types A and b, that are connected by a relationship. The cardinality of the relationship is the number of instances of entity B that can be associated with each instance of entity A. The minimum cardinality of a relationship is the minimum number of instances of an entity B that may be associated with each instance of entity A.

Attributes:

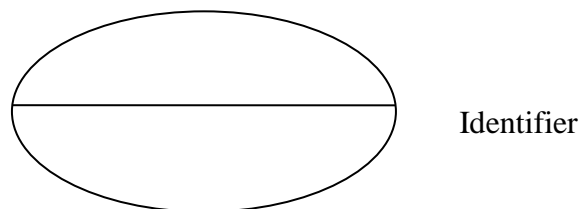
Each entity type has a set of attributes associated with it. An attribute is a property or characteristic of an entity that is of interest to the organization. We use an initial capital letter, followed by lowercase letters, and nouns in naming an attribute. In E-R diagram, we can visually represent an attribute by placing its name as an ellipse with a line connecting it to the associated entity.

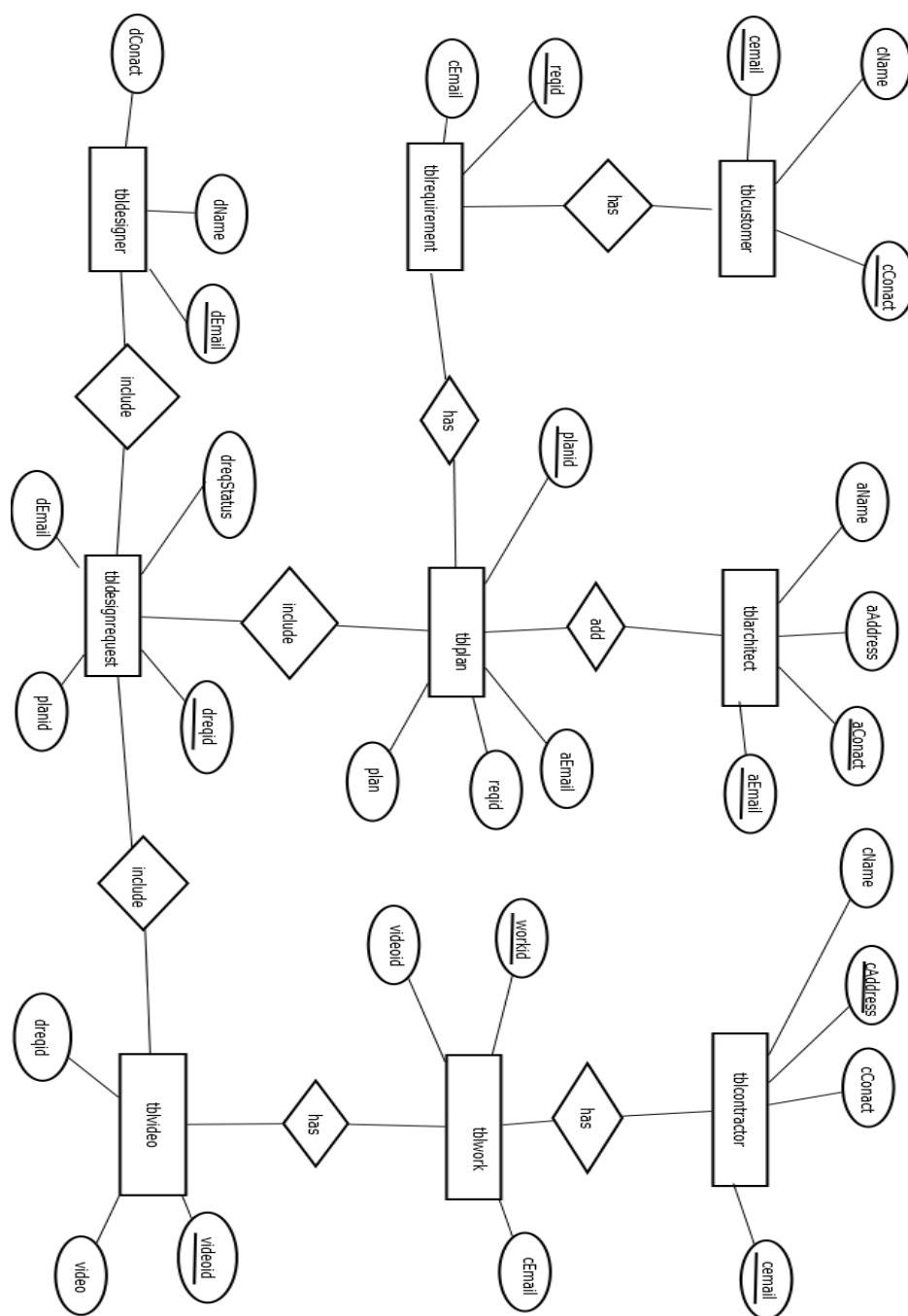
Notation for attribute is:

**Candidate Key and Identifier:**

Every entity type must have an attribute and a set of attributes that distinguish one in-stance from other instances of the same type. A candidate key is an attribute (or a combination of attributes) that uniquely identifies each instance of an entity type. If there is more than one candidate key, the designer must choose one of the candidate keys as the identifier. An identifier is a candidate key that has been selected to be used as the unique characteristic for an entity type.

Notation for an identifier:



ER diagram

DEVELOPMENT AND IMPLEMENTATION

4. DEVELOPMENT

Software development is the process of computer programming, documenting, testing, and bug fixing involved in creating and maintaining applications and frameworks resulting in a software product. Software development is a process of writing and maintaining the source code. Therefore, software development may include research, new development, prototyping, modification, reuse, re-engineering, maintenance, or any other activities that result in software products. Software can be developed for a variety of purposes, the three most common being to meet specific needs of a specific client/business (the case with custom software), to meet a perceived need of some set of potential users (the case with commercial and open source software), or for personal use (e.g. a scientist may write software to automate a mundane task). Embedded software development, that is, the development of embedded software such as used for controlling consumer products, requires the development process to be integrated with the development of the controlled physical product. System software underlies applications and the programming process itself, and is often developed separately.

5.1 Tools

Easy PHP Deserver:

EasyPHPDeserver is a ready to use development environment with PHP, Python, Ruby, Perl, MySQL, PhpMyAdmin, Apache, Nginx, modules (Virtual Hosts manager, Xdebug manager, Codiad IDE Editor, Webgring, PHP Code Sniffer...), tools and more... Devserver installs a complete and ready-to-use development environment. Devserver is portable, modular, fully con-figurable and easy to update and extend. Webserver turns your computer into a ready-to-use per-sonal web hosting server. You can host whatever you want directly on your computer and share it on internet like any website. Your computer acts like a web hosting service and allows you to make your website / application / demo accessible via internet. The server is fully configurable, modular and easy to update and extend. Webserver turns your computer into a ready-to-use per-sonal web hosting server. You can host whatever you want directly on your computer and share it on internet like any website. Your computer acts like a web hosting service and allows you to make your website / application / demo accessible via internet.

5.2 Language used

Backend: -

MySQL:

MySQL (officially pronounced as /maɪ ˈɛskjuːˈɛl/ "My S-Q-L") is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius' daughter, and "SQL", the abbreviation for Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. For proprietary use, several paid editions are available, and offer additional functionality. MySQL is written in C and C++. Its SQL parser is written in yacc, but it uses a home-brewed lexical analyzer. MySQL works on many system platforms, including AIX, BSDi, FreeBSD, HP-UX, eComStation, i5/OS, IRIX, Linux, macOS, Microsoft Windows, Net-BSD, Novell NetWare, OpenBSD, OpenSolaris, OS/2 Warp, QNX, Oracle Solaris, Symbian, SunOS, SCO OpenServer, SCO UnixWare, Sanos and Tru64. A port of MySQL to OpenVMS also exists. MySQL has received positive reviews, and reviewers noticed it "performs extremely well in the average case" and that the "developer interfaces are there, and the documentation (not to mention feedback in the real world via Web sites and the like) is very, very good. It has also been tested to be a "fast, stable and true multi-user, multi-threaded sql database server".

Frontend:-

HTML:

Hypertext Mark-up Language (HTML) is the standard mark-up 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 webserver 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. HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects, such as interactive forms, may be embedded into the rendered page. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by *tags*, written using angle brackets. Tags such as and <input /> introduce content into the page directly. Others such as <p>...</p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page. HTML can embed programs written in a scripting language such as JavaScript which affect the behavior and content of web pages. Inclusion of

CSS defines the look and layout of content. The World Wide Web Consortium (W3C), maintainer of both the HTML and the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

JavaScript:

JavaScript is a high-level, dynamic, untyped, and interpreted programming language. It has been standardized in the ECMAScript language specification. Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content production; the majority of websites employ it, and all modern Web browsers support it without the need for plug-ins. JavaScript is prototype-based with first-class functions, making it a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles. It has an API for working with text, arrays, dates and regular expressions, but does not include any I/O, such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded. Although there are strong outward similarities between JavaScript and Java, including language name, syntax, and respective standard libraries, the two are distinct languages and differ greatly in their design. JavaScript was influenced by programming languages such as Self and Scheme. JavaScript is also used in environments that are not Web-based, such as PDF documents, site-specific browsers, and desktop widgets. Newer and faster JavaScript virtual machines (VMs) and platforms built upon them have also increased the popularity of JavaScript for server-side Web applications. On the client side, developers have traditionally implemented JavaScript as an interpreted language, but more recent browsers perform just-in-time compilation. Programmers also use JavaScript in video-game development, in crafting desktop and mobile applications, and in server-side network programming with run-time environments such as Node.js.

Cascading style sheet(CSS):

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a mark-up language. Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL, and is applicable to rendering in speech, or on other media. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user in-terfaces for web applications, and user interfaces for many mobile applications. CSS is designed primarily to enable the separation of document content from document presentation, including aspects such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .CSS file, and reduce complexity and repetition in the structural content. Separation of formatting and con-tent

makes it possible to present the same mark-up page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. It can also display the web page differently depending on the screen size or viewing device. Readers can also specify a different style sheet, such as a CSS file stored on their own computer, to override the one the author specified. Changes to the graphic design of a document (or hundreds of documents) can be applied quickly and easily, by editing a few lines in the CSS file they use, rather than by changing mark-up in the documents. The CSS specification describes a priority scheme to determine which style rules apply if more than one rule matches against a particular element. In this so-called cascade, priorities (or weights) are calculated and assigned to rules, so that the results are predictable. The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type)text/CSS is registered for use with CSS by RFC 2318 (March 1998). The W3C operates a free CSS validation service for CSS documents.

Scripting language:-

PHP:

PHP is a server-side scripting language designed primarily for web development but also used as a general-purpose programming language. Originally created by RasmusLerdorf in 1994, the PHP reference implementation is now produced by The PHP Development Team. PHP originally stood for Personal Home Page, but it now stands for the recursive acronym PHP: Hypertext Preprocessor. PHP code may be embedded into HTML or HTML5 code, or it can be used in combination with various web template systems, web content management systems and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as Common Gateway Interface (CGI) executable. The web server combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code may also be executed with a command-line interface (CLI) and can be used to implement standalone graphical applications. The standard PHP interpreter, powered by the Zend Engine, is free software released under the PHP License. PHP has been widely ported and can be deployed on most web servers on almost every operating system and platform, free of charge. The PHP language evolved without a written formal specification or standard until 2014, leaving the canonical PHP interpreter as a de facto standard. Since 2014 work has gone on to create a formal PHP specification.

Python:

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting and was discontinued with version 2.7.18 in 2020. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible and much Python 2 code does not run unmodified on Python 3.

5.2 Modules

1. User Management Module

Admin and other Users' login to the site using a common login page. Users use the username and password provided at the time of registration to login to the site

2. Requirement Module

this module the user can fill out the requirement details of the house. The requirement is then send to the architect and designer for further planning process. After analysing the requirements, architect and designer construct plans based on the user needs.

3. Plan Module

In this module with the data provided in requirement module, the architect builds a plan and upload to the user. The user can either approve or reject the plan

4. 3D Video module

In this Module the Designer create a 3D model based on the Architectural plan. The user can either approve or reject the video

5. Work Module

In this Module the contractor can give an estimate based on the 3D video and Architectural plan.

VALIDATION CRITERIA AND IMPLIMENTATION

6. VALIDATION CRITERIA AND IMPLIMENTATION

6.1 Test plans

A test plans describes how testing would be accomplished. It is a document that specifies the purpose, scope and method of software testing. It determines the testing tasks and the persons involved in executing those tasks, test items and the features to be tested. It also describes the environment for testing, the test design and the measurement techniques to be used. A properly defined test plan is an agreement between testers and users describing the role of testing in software. A test plan helps the people who are not involved in test group to understand why product validation is needed and how it is to be performed. However if the test plan is not complete, it might not be possible to check how the software operates when installed on different operating systems or when used with other software. A carefully developed test plan facilities effective execution, proper analysis of errors and preparation of error report.

6.2 Classifications of testing

Black box testing– Internal system design is not considered in this type of testing. Tests are based on requirements and functionality.

White box testing– This testing is based on knowledge of the internal logic of an application's code. Also known as Glass box Testing. Internal software and code working should be known for this type of testing. Tests are based on coverage of code statements, branches, paths, conditions.

Unit testing – Testing of individual software components or modules. Typically done by the programmer and not by testers, as it requires detailed knowledge of the internal program design and code may require developing test driver modules or test harnesses.

Incremental integration testing – Bottom up approach for testing i.e. continuous testing of an application as new functionality is added; Application functionality and modules should be independent enough to test separately done by programmers or by testers.

Integration testing – Testing of integrated modules to verify combined functionality after integration. Modules are typically code modules, individual applications, client and server applications on a network, etc. This type of testing is especially relevant to client/server and distributed systems.

Functional testing – This type of testing ignores the internal parts and focus on the output is as per requirement or not. Black-box type testing geared to functional requirements of an application.

System testing– Entire system is tested as per the requirements. Black-box type testing that is based on overall requirements specifications, covers all combined parts of a system.

End-to-end testing– Similar to system testing, involves testing of a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate.

Sanity testing– Testing to determine if a new software version is performing well enough to accept it for a major testing effort. If application is crashing for initial use then system is not stable enough for further testing and build or application is assigned to fix.

Regression testing– Testing the application as a whole for the modification in any module or functionality. Difficult to cover all the system in regression testing so typically automation tools are used for these testing types.

Acceptance testing -Normally this type of testing is done to verify if system meets the customer specified requirements. User or customer does this testing to determine whether to accept application.

Load testing– It's a performance testing to check system behavior under load. Testing an application under heavy loads, such as testing of a web site under a range of loads to determine at what point the system's response time degrades or fails.

Stress testing– System is stressed beyond its specifications to check how and when it fails. Performed under heavy load like putting large number beyond storage capacity, complex database queries, continuous input to system or database load.

Performance testing– Term often used interchangeably with „stress“ and „load“ testing. To check whether system meets performance requirements. Used different performance and load tools to do this.

Usability testing– User-friendliness check. Application flow is tested, Can new user understand the application easily, Proper help documented whenever user stuck at any point. Basically system navigation is checked in this testing.

Install/uninstall testing– Tested for full, partial, or upgrade install/uninstall processes on different operating systems under different hardware, software environment.

Recovery testing– Testing how well a system recovers from crashes, hardware failures, or other catastrophic problems.

Security testing– Can system be penetrated by any hacking way. Testing how well the system protects against unauthorized internal or external access. Checked if system, database is safe from external attacks.

Compatibility testing– Testing how well software performs in a particular hardware/software/operating system/network environment and different combinations of above.

Comparison testing– Comparison of product strengths and weaknesses with previous versions or other similar products.

Alpha testing– In house virtual user environment can be created for this type of testing. Testing is done at the end of development. Still minor design changes may be made as a result of such testing.

Beta testing– Testing typically done by end-users or others. Final testing before releasing application for commercial purpose.

6.3 Implementation

Implementation is the stage in the project where the theoretical design is turned in to a working system and is giving confidence and effectiveness. It involves careful planning investigation of the current system and its constraints on implementation design of methods to achieve the change, over an evaluation of change over methods. A part from planning major tasks of preparing the implementation is education and training of the users. The more complex system is being implemented the more evolved will be the system analysis and design efforts required for the implementation. An implementation coordinating committee based on policies of the individual organization has been appointed. The implementation process begins with preparing a plan for the implementation of the system. According to this plan the activities are to be carried out, discussions made regarding the equipment has to be acquired to implement the system. Implementation is the final and the most important phase. The most critical stage in achieving a successful new system and in giving the users confidence is only after through testing is done and if it is found to work according to the specification. This method also offers the greatest security since the old system can take over if the errors are found or inability to handle certain type of transactions while using the new system.

CONCLUSION

To conclude, this project would help a lot of people to find the right people to construct a building, and have an estimate of the construction. This site also acts as a platform for the freelance workers to find a job. The contribution this project would make is that it would eliminate the need of middle men in the construction field, also reduce the cost of construction, they also can find the architects, civil engineer, 3D designers and contractors under a single roof. This project aim to help the users find more architects, designers, and constructors based on what they like and also act as a platform for the Architects, Designers and Constructors to find more customers and grow their business. This also helps the customers to find the people needed for the construction in a quicker method and do not need to ask around for finding new architects. This site helps the Architects, Designers and Constructors those are new in this field find customers and grow their business, this also help the customers to find the Architects, Designers and Constructors for low cost and thus reduce the construction cost.

Code**Views.py**

```

from django.shortcuts import render
from django.http import HttpResponseRedirect
from django.http import HttpResponseRedirect
from django.core.files.storage import FileSystemStorage
import MySQLdb
import webbrowser

db=MySQLdb.connect("localhost","root","","dbrealistic")
c=db.cursor()

#####
#                LOAD INDEX PAGE
#####

def index(request):
    """
    The function to load index page of the project.
    -----
    Parameters:
    HTTP request

    Returns:
    html page
    """
    return render(request,"index.html")

#####
#                LOAD LOGIN PAGE
#####

def login(request):

```

"""

The function for login process

Parameters:

HTTP request

Returns:

html page

"""

msg="""

if(request.POST):

email=request.POST.get("txtEmail")

pwd=request.POST.get("txtPassword")

s="select count(*) from tbllogin where username='"+email+"'"

c.execute(s)

i=c.fetchone()

if(i[0]>0):

s="select * from tbllogin where username='"+email+"'"

c.execute(s)

i=c.fetchone()

if(i[1]==pwd):

request.session['email'] = email

if(i[3]=="1"):

if(i[2]=="admin"):

return HttpResponseRedirect("/adminhome")

elif(i[2]=="architect"):

return HttpResponseRedirect("/architecthome")

elif(i[2]=="designer"):

return HttpResponseRedirect("/designerhome")

elif(i[2]=="contractor"):


```

return HttpResponseRedirect("/contractorhome")

elif(i[2]=="customer"):
return HttpResponseRedirect("/customerhome")
else:
msg="You are not authenticated to login"
else:
msg="Incorrect password"
else:
msg="User doesnt exist"
return render(request,"login.html",{ "msg":msg })

#####
#                CUSTOMER REGISTRATION
#####

def customer(request):
    """
    The function for customer registration
    -----
    Parameters:
    HTTP request

    Returns:
    html page
    """
    msg=""
    if(request.POST):
        name=request.POST.get("txtName")
        address=request.POST.get("txtAddress")
        contact=request.POST.get("txtContact")
        email=request.POST.get("txtEmail")
        pwd=request.POST.get("txtPassword")

```

```

aadhar=request.FILES["txtaadhar"]
fs=FileStorage()
filename=fs.save(aadhar.name,aadhar)
uploaded_file_url=fs.url(filename)
s="select count(*) from tbllogin where username='"+str(email)+"'"
c.execute(s)
i=c.fetchone()
if(i[0]>0):
    msg="User already registered"
else:
    s="insert into tblcustomer(cName,cAddress,cContact,cEmail,aadhar)
    values('"+str(name)+"','"+str(address)+"','"+str(contact)+"','"+str(email)+"','"+str(uploaded_file_url)+"'"
    try:
        c.execute(s)
        db.commit()
    except:
        msg="Sorry registration error"
    else:
        s="insert into tbllogin (username,password,utype,status)
        values('"+str(email)+"','"+str(pwd)+"','customer','1')"
        try:
            c.execute(s)
            db.commit()
        except:
            msg="Sorry login error"
        else:
            msg="Registration successful"
    return render(request,"customer.html",{ "msg":msg})

#####

#                ARCHITECT REGISTRATION

#####

```

```

def architect(request):
    """
    The function for architect registration
    -----
    Parameters:
    HTTP request

    Returns:
    html page
    """
    msg=""
    if(request.POST):
        name=request.POST.get("txtName")
        address=request.POST.get("txtAddress")
        contact=request.POST.get("txtContact")
        quali=request.POST.get("txtquali")

        qproof=request.FILES["proof"]
        fs=FileSystemStorage()
        filename=fs.save(qproof.name,qproof)
        uploaded_file=fs.url(filename)

        email=request.POST.get("txtEmail")
        pwd=request.POST.get("txtPassword")

        img=request.FILES["txtFile"]
        fs=FileSystemStorage()
        fileimg=fs.save(img.name,img)
        uploaded_file_url=fs.url(fileimg)

```

```

s="select count(*) from tbllogin where username='"+str(email)+"'"
c.execute(s)
i=c.fetchone()
if(i[0]>0):
    msg="User already registered"
else:
    s="insert into tblarchitect(aName,aAddress,aContact,aqualification,aqproof,aEmail,aPhoto)
    values('"+str(name)+"','"+str(address)+"','"+str(contact)+"','"+str(quali)+"','"+str(uploaded_file)+"','"+str
    (email)+"','"+str(uploaded_file_url)+"'"
    try:
        c.execute(s)
        db.commit()
    except:
        msg="Sorry registration error"
    else:
        s="insert into tbllogin (username,password,utype,status)
        values('"+str(email)+"','"+str(pwd)+"','architect','0')"
        try:
            c.execute(s)
            db.commit()
        except:
            msg="Sorry login error"
        else:
            msg="Registration successfull"
    return render(request,"architect.html",{ "msg":msg})

#####

#                DESIGNER REGISTRATION

#####

def designer(request):
    """
    The function for designer registration

```

Parameters:

HTTP request

Returns:

html page

"""

msg="""

if(request.POST):

name=request.POST.get("txtName")

address=request.POST.get("txtAddress")

contact=request.POST.get("txtContact")

quali=request.POST.get("txtquali")

qproof=request.FILES["txtproof"]

fs=FileSystemStorage()

filename=fs.save(qproof.name,qproof)

uploaded_file=fs.url(filename)

email=request.POST.get("txtEmail")

pwd=request.POST.get("txtPassword")

img=request.FILES["txtFile"]

fs=FileSystemStorage()

filename=fs.save(img.name,img)

uploaded_file_url=fs.url(filename)

s="select count(*) from tbllogin where username='"+str(email)+"'"

c.execute(s)

i=c.fetchone()

if(i[0]>0):

msg="User already registered"

else:

```

s="insert into tbldesigner(dName,dAddress,dContact,dqualification,dqproof,dEmail,dPhoto)
values('"+str(name)+"','"+str(address)+"','"+str(contact)+"','"+str(quali)+"','"+str(uploaded_file)+"','"+str
(email)+"','"+str(uploaded_file_url)+"')
try:
c.execute(s)
db.commit()
except:
msg="Sorry registration error"
else:
s="insert into tbllogin (username,password,utype,status)
values('"+str(email)+"','"+str(pwd)+"','designer','0')
try:
c.execute(s)
db.commit()
except:
msg="Sorry login error"
else:
msg="Registration successfull"
return render(request,"designer.html",{ "msg":msg })

#####
#                CONTRACTOR REGISTRATION
#####

def contractor(request):
    """
    The function for contractor registration
    -----
    Parameters:
    HTTP request

    Returns:
    html page
  
```

```

"""
msg=""
if(request.POST):
    name=request.POST.get("txtName")
    address=request.POST.get("txtAddress")
    contact=request.POST.get("txtContact")
    pw=request.FILES["txtpw"]
    fs=FileSystemStorage()
    filename=fs.save(pw.name,pw)
    uploaded_file=fs.url(filename)

    email=request.POST.get("txtEmail")
    pwd=request.POST.get("txtPassword")

    img=request.FILES["txtFile"]
    fs=FileSystemStorage()
    filename=fs.save(img.name,img)
    uploaded_file_url=fs.url(filename)

    s="select count(*) from tbllogin where username='"+str(email)+"'"
    c.execute(s)
    i=c.fetchone()
    if(i[0]>0):
        msg="User already registered"
    else:
        s="insert into tblcontractor(cName,cAddress,cContact,cprework,cEmail,cPhoto)
        values('"+str(name)+"','"+str(address)+"','"+str(contact)+"','"+str(uploaded_file)+"','"+str(email)+"','"+str(uploaded_file_url)+"'"
        try:
            c.execute(s)
            db.commit()

```

```

except:

msg="Sorry registration error"

else:

s="insert into tbllogin (username,password,utype,status)
values('"+str(email)+"','"+str(pwd)+"','contractor','0')"

try:

c.execute(s)

db.commit()

except:

msg="Sorry login error"

else:

msg="Registration successfull"

return render(request,"contractor.html",{ "msg":msg})

#####

#                ADMIN HOME

#####

def adminhome(request):

"""

The function for adminhome

-----

Parameters:

HTTP request

Returns:

html page

"""

return render(request,"adminhome.html")

#####

#                ADMIN ARCHITECT

#####

def adminarchitect(request):

```



```
"""
```

The function for architect details for admin

```
-----
```

Parameters:

HTTP request

Returns:

html page

```
"""
```

```
s="select * from tblarchitect where aEmail in(select username from tbllogin where status='0')"
```

```
c.execute(s)
```

```
data=c.fetchall()
```

```
s="select * from tblarchitect where aEmail in(select username from tbllogin where status='1')"
```

```
c.execute(s)
```

```
data1=c.fetchall()
```

```
return render(request,"adminarchitect.html",{ "data":data,"data1":data1 })
```

```
#####
```

```
#                ADMIN DESIGNER
```

```
#####
```

```
def admindesigner(request):
```

```
"""
```

The function for designer details for admin

```
-----
```

Parameters:

HTTP request

Returns:

html page

```
"""
```

```
s="select * from tbldesigner where dEmail in(select username from tbllogin where status='0')"
```

```

c.execute(s)

data=c.fetchall()

s="select * from tbldesigner where dEmail in(select username from tbllogin where status='1')"

c.execute(s)

data1=c.fetchall()

return render(request,"admindesigner.html",{ "data":data,"data1":data1 })

#####

#                ADMIN CONTRACTOR

#####

def admincontractor(request):

    """

    The function for contractor details for admin

    -----

    Parameters:

    HTTP request


    Returns:

    html page

    """

    s="select * from tblcontractor where cEmail in(select username from tbllogin where status='0')"

    c.execute(s)

    data=c.fetchall()

    s="select * from tblcontractor where cEmail in(select username from tbllogin where status='1')"

    c.execute(s)

    data1=c.fetchall()

    return render(request,"admincontractor.html",{ "data":data,"data1":data1 })

#####

#                ADMIN CUSTOMER

#####

def admincustomer(request):

```

"""

The function for customer details for admin

Parameters:

HTTP request

Returns:

html page

"""

s="select * from tblcustomer where cEmail in(select username from tbllogin where status='1')"

c.execute(s)

data=c.fetchall()

return render(request,"admincustomer.html",{ "data":data})

#####

ADMIN APPROVE CUSTOMER

#####

def adminapproveuser(request):

"""

The function to approve users

Parameters:

HTTP request

Returns:

html page

"""

email=request.GET.get("id")

status=request.GET.get("status")

url=request.GET.get("url")

s="update tbllogin set status='"+status+"' where username='"+email+"'"

```

c.execute(s)
db.commit()
c.execute("select utype from tbllogin where username='"+email+"'")
k=c.fetchone()
if(k[0]=='architect'):
n="Select aContact from tblarchitect where aEmail='"+email+"'"
c.execute(n)
d=c.fetchone()
contact=d[0]
msg="Your registration is approved"
sendsms(contact,msg)
if(k[0]=='designer'):
n="Select dContact from tbldesigner where dEmail='"+email+"'"
c.execute(n)
d=c.fetchone()
contact=d[0]
msg="Your registration is approved"
sendsms(contact,msg)
if(k[0]=='contractor'):
n="Select cContact from tblcontractor where cEmail='"+email+"'"
c.execute(n)
d=c.fetchone()
contact=d[0]
msg="Your registration is approved"
sendsms(contact,msg)
if(k[0]=='customer'):
n="Select cContact from tblcustomer where cEmail='"+email+"'"
c.execute(n)
d=c.fetchone()
contact=d[0]

```

```

msg="Your registration is approved"

sendsms(contact,msg)

return HttpResponseRedirect(url)

#####

#                CUSTOMER HOME

#####

def customerhome(request):
    """
    The function for customer home

    -----

    Parameters:
    HTTP request

    Returns:
    html page

    """
    email=request.session["email"]
    s="select * from tblcustomer where cEmail='"+email+"'"
    c.execute(s)
    data=c.fetchall()
    return render(request,"customerhome.html",{"data":data})

#####

#                CUSTOMER REQUIREMENT

#####

def customerrequirement(request):
    """
    The function for customer requirement

    -----

    Parameters:
    HTTP request

```

Returns:

html page

""

email=request.session["email"]

msg=""

if(request.POST):

bed=request.POST["txtBed"]

bath=request.POST["txtBath"]

attached=request.POST["txtAttached"]

car=request.POST["txtCar"]

kitchen=request.POST["txtKitchen"]

sitout=request.POST["txtSitout"]

work=request.POST["txtWork"]

floor=request.POST["txtFloor"]

sqft=request.POST["txtSqft"]

other=request.POST["txtOther"]

s="insert into

tblrequirement(cEmail,bedroom,bathroom,attached,carporch,kitchen,sitout,workarea,floor,sqft,other,req Date,reqStatus)

values('"+email+"','"+bed+"','"+bath+"','"+attached+"','"+car+"','"+kitchen+"','"+sitout+"','"+work+"','"+f
loor+"','"+sqft+"','"+other+"',(select sysdate()),'requested')"

try:

c.execute(s)

db.commit()

except:

msg="Sorry some error occured"

else:

msg="Requirement submitted"

s="Select * from tblrequirement where cEmail='"+email+"' and tblrequirement.reqStatus<>'plan approved'"

c.execute(s)

```

data=c.fetchall()

return render(request,"customerrequirement.html",{ "msg":msg,"data":data })

#####

#                CUSTOMER PLANS

#####

def customerplan(request):
    """
    The function to load plan status

    -----

    Parameters:

    HTTP request


    Returns:

    html page

    """
    email=request.session["email"]
    rid=request.GET.get("id")

    s="select
tblrequirement.reqId,tblarchitect.aName,tblplan.sqft,tblplan.cost,tblplan.planStatus,tblplan.planId,tblplan.plan from tblrequirement,tblplan,tblarchitect where tblarchitect.aEmail=tblplan.aEmail and
tblplan.reqId=tblrequirement.reqId and tblplan.reqId='"+str(rid)+"' and
tblrequirement.cEmail='"+str(email)+"'"

    c.execute(s)
    data=c.fetchall()
    return render(request,"customerplan.html",{ "data":data })

#####

#                CUSTOMER VIEW PLANS

#####

def customerviewplan(request):
    """
    The function to view plan

    -----

```

Parameters:

HTTP request

Returns:

html page

"""

plan=request.GET.get("id")

return render(request,"customerviewplan.html",{"plan":plan})

#####

CUSTOMER PLAN UPDATE

#####

def customerplanupdate(request):

"""

The function to update plan status

Parameters:

HTTP request

Returns:

html page

"""

pid=request.GET.get("id")

status=request.GET.get("status")

url=request.GET.get("url")

rid=request.GET.get("rid")

s="update tblplan set planStatus='"+status+"' where planId='"+pid+'"

try:

c.execute(s)

db.commit()

except:


```

pass
else:
if(status=="approved"):
status1="rejected"
elif(status=="rejected"):
status1="approved"
s="update tblplan set planStatus='"+status1+"' where reqId='"+rid+"' and planId<> '"+pid+"'"
try:
c.execute(s)
db.commit()
except:
pass
else:
if(status=="approved"):
status="plan approved"
s="update tblrequirement set reqStatus='"+status+"' where reqId='"+rid+"'"
try:
c.execute(s)
db.commit()
except:
pass
else:
return HttpResponseRedirect("/") + url + "?id="+pid)
else:
return HttpResponseRedirect("/") + url)

#####
#                CUSTOMER APPROVED PLANS
#####

def customerapprovedplan(request):
"""

```

The function to load plan status

Parameters:

HTTP request

Returns:

html page

"""

```
email=request.session["email"]
```

```
s="select
tblrequirement.reqId,tblarchitect.aName,tblplan.sqft,tblplan.cost,tblplan.planStatus,tblplan.planId,tblpla
n.plan from tblrequirement,tblplan,tblarchitect where tblarchitect.aEmail=tblplan.aEmail and
tblplan.reqId=tblrequirement.reqId and tblrequirement.cEmail='"+email+"' and
(tblplan.planStatus='approved' or tblplan.planStatus='3D requested')"
```

```
c.execute(s)
```

```
data=c.fetchall()
```

```
return render(request,"customerapprovedplan.html",{"data":data})
```

```
#####
```

```
#                CUSTOMER VIEW DESIGNER
```

```
#####
```

```
def customerviewdesigner(request):
```

"""

The function to load designers

Parameters:

HTTP request

Returns:

html page

"""

```
request.session["planid"]=request.GET.get("id")
```

```
s="select * from tbldesigner where dEmail in (select username from tbllogin where status='1')"
```

```

c.execute(s)

data=c.fetchall()

return render(request,"customerviewdesigner.html",{ "data":data})

#####

#                CUSTOMER PASS PLAN

#####

def customerpassplan(request):
    """
    The function to pass work to designer
    -----
    Parameters:
    HTTP request

    Returns:
    html page
    """
    email=request.GET.get("id")
    planid=request.session["planid"]
    s="insert into tbldesignrequest(planId,dEmail,dreqStatus)
    values('"+str(planid)+"','"+str(email)+"','requested')"
    try:
        c.execute(s)
        db.commit()
    except:
        pass
    else:
        s="update tblplan set planStatus='3D requested' where planId='"+str(planid)+"'"
        try:
            c.execute(s)
            db.commit()
        except:

```

```

pass
else:
return HttpResponseRedirect("/customerdesignrequest")

#####

#          CUSTOMER DESIGN REQUEST
#####

def customerdesignrequest(request):
    """
    The function for request for design
    -----

    Parameters:
    HTTP request

    Returns:
    html page
    """
    email=request.session["email"]

    s="select
tblrequirement.reqId,tbldesigner.dName,tblplan.sqft,tblplan.cost,tbldesignrequest.dreqId,tbldesignrequest.dreqStatus from tblrequirement,tbldesigner,tblplan,tbldesignrequest where
tblrequirement.reqId=tblplan.reqId and tblplan.planId=tbldesignrequest.planid and
tbldesigner.dEmail=tbldesignrequest.dEmail and tblrequirement.cEmail='"+email+"'"

    c.execute(s)
    data=c.fetchall()
    return render(request,"customerdesignrequest.html",{ "data":data})

#####

#          CUSTOMER VIDEOS
#####

def customervideo(request):
    """
    The function for request for design
    -----

```

Parameters:

HTTP request

Returns:

html page

"""

email=request.session["email"]

s="select

tbldesignrequest.dreqId,tbldesigner.dName,tblvideo.video,tblvideo.videoId,tbldesignrequest.dreqStatus
from tbldesignrequest,tbldesigner,tblvideo where tbldesignrequest.dreqId=tblvideo.dreqId and
tbldesignrequest.dEmail=tbldesigner.dEmail and tbldesignrequest.planId in(select planId from tblplan
where reqId in(select reqId from tblrequirement where cEmail="'+email+'")) and
tbldesignrequest.dreqStatus='video uploaded'"

c.execute(s)

data=c.fetchall()

s="select

tbldesignrequest.dreqId,tbldesigner.dName,tblvideo.video,tblvideo.videoId,tbldesignrequest.dreqStatus
from tbldesignrequest,tbldesigner,tblvideo where tbldesignrequest.dreqId=tblvideo.dreqId and
tbldesignrequest.dEmail=tbldesigner.dEmail and tbldesignrequest.planId in(select planId from tblplan
where reqId in(select reqId from tblrequirement where cEmail="'+email+'")) and
tbldesignrequest.dreqStatus='video approved'"

c.execute(s)

data1=c.fetchall()

return render(request,"customervideo.html",{"data":data,"data1":data1 })

#####

CUSTOMER VIDEO UPDATE

#####

def customervideoupdate(request):

"""

The function to update video status

Parameters:

HTTP request

Returns:

html page

"""

pid=request.GET.get("id")

request.session["vid"]=pid

status=request.GET.get("status")

url=request.GET.get("url")

s="update tblvideo set videoStatus='"+status+"' where dreqId='"+pid+"'"

try:

c.execute(s)

db.commit()

except:

pass

else:

if(status=="approved"):

status="video approved"

s="update tbldesignrequest set dreqStatus='"+status+"' where dreqId in(select dreqId from tblvideo where videoId='"+pid+"')"

try:

c.execute(s)

db.commit()

except:

pass

else:

return HttpResponseRedirect(url)

else:

return HttpResponseRedirect(url)

#####

CUSTOMER VIEW VIDEO

#####

def customerview3D(request):

```
"""
```

The function for request for design

```
-----
```

Parameters:

HTTP request

Returns:

html page

```
"""
```

```
video=request.GET.get("id")
```

```
return render(request,"customerview3D.html",{"video":video})
```

```
#####
```

```
#                CUSTOMER SELECT CONTRACTOR
```

```
#####
```

```
def customerselectcontractor(request):
```

```
"""
```

The function for request for architect

```
-----
```

Parameters:

HTTP request

Returns:

html page

```
"""
```

```
s="select * from tblcontractor where cEmail in(select username from tbllogin where status='1')"
```

```
c.execute(s)
```

```
data=c.fetchall()
```

```

return render(request,"customerselectcontractor.html",{ "data":data})

#####

#                CUSTOMER ASSIGN WORK

#####

def customerassignwork(request):
    """
    The function for request for contractor
    -----

    Parameters:
    HTTP request

    Returns:
    html page
    """
    cemail=request.GET.get("id")
    vid=request.session["vid"]

    s="insert into tblwork (videoId,cEmail,wDate,wStatus) values('"+str(vid)+"','"+str(ceemail)+"',(select
    sysdate()),'assigned')"
    c.execute(s)
    db.commit()
    return HttpResponseRedirect("/customerwork")

#####

#                CUSTOMER SELECT CONTRACTOR

#####

def customerwork(request):
    """
    The function to view current works
    -----

    Parameters:
    HTTP request

```


Returns:

html page

"""

email=request.session["email"]

s="select tblwork.workId,tblcontractor.cName,tblplan.sqft,tblwork.wStatus from
tblwork,tblcontractor,tblplan,tblvideo,tbldesignrequest where tblwork.videoId=tblvideo.videoId and
tblvideo.dreqId=tbldesignrequest.dreqId and tbldesignrequest.planId=tblplan.planId and
tblwork.cEmail=tblcontractor.cEmail and tblplan.reqId in(select reqId from tblrequirement where
cEmail='"+email+"') "

c.execute(s)

data=c.fetchall()

return render(request,"customerwork.html",{ "data":data})

#####

ARCHITECT HOME

#####

def architecthome(request):

"""

The function for request for architect

Parameters:

HTTP request

Returns:

html page

"""

email=request.session["email"]

s="select * from tblarchitect where aEmail='"+email+"'"

c.execute(s)

data=c.fetchall()

return render(request,"architecthome.html",{ "data":data})

#####

```

#                                ARCHITECT REQUEST
#####

def architectrequest(request):
    """
    The function for request for architect
    -----

    Parameters:
    HTTP request

    Returns:
    html page
    """
    email=request.session['email']

    s="select tblrequirement.*,tblcustomer.cName from tblcustomer,tblrequirement,tblallocation,tblarchitect
    where tblrequirement.cEmail=tblcustomer.cEmail and tblallocation.Status='assigned' and
    tblallocation.archid='"+str(email)+"' and tblallocation.archid=tblarchitect.aEmail and
    tblallocation.requid=tblrequirement.reqId and tblrequirement.reqStatus='requested'"

    c.execute(s)

    data=c.fetchall()

    return render(request,"architectrequirement.html",{ "data":data })
#####

#                                ARCHITECT ADD PLAN
#####

def architectaddplan(request):
    """
    The function to add plan
    -----

    Parameters:
    HTTP request

    Returns:

```

```

html page
"""
msg=""
email=request.session["email"]
rid=request.GET.get("id")
if(request.POST):
img=request.FILES["txtFile"]
fs=FileSystemStorage()
filename=fs.save(img.name,img)
uploaded_file_url=fs.url(filename)
sqft=request.POST["txtSqft"]
cost=request.POST["txtCost"]

s="insert into tblplan (aEmail,reqId,plan,sqft,cost,planStatus)
values('"+email+"','"+rid+"','"+uploaded_file_url+"','"+sqft+"','"+cost+"','submitted')"
try:
c.execute(s)
db.commit()
except:
msg="Sorry some error occured"
else:
s="update tblrequirement set reqStatus='plan uploaded' where reqId='"+rid+"'"
try:
c.execute(s)
db.commit()
except:
msg="Sorry error"
else:
msg="Plan added"
return render(request,"architectaddplan.html",{ "msg":msg })

#####
#                ARCHITECT PLANS

```

```
#####
```

```
def architectplan(request):
```

```
    """
```

The function to load plan status

```
-----
```

Parameters:

HTTP request

Returns:

html page

```
    """
```

```
email=request.session["email"]
```

```
s="select tblrequirement.reqId,tblcustomer.cName,tblplan.sqft,tblplan.cost,tblplan.planStatus from
tblrequirement,tblplan,tblcustomer where tblcustomer.cEmail=tblrequirement.cEmail and
tblplan.reqId=tblrequirement.reqId and tblplan.aEmail='"+email+"'"
```

```
c.execute(s)
```

```
data=c.fetchall()
```

```
return render(request,"architectplan.html",{ "data":data })
```

```
#####
```

```
#                DESIGNER HOME
```

```
#####
```

```
def designerhome(request):
```

```
    """
```

The function for designer home

```
-----
```

Parameters:

HTTP request

Returns:

html page

```
    """
```

```

email=request.session["email"]

s="select * from tbldesigner where dEmail='"+email+"'"

c.execute(s)

data=c.fetchall()

return render(request,"designerhome.html",{ "data":data})

#####

#                DESIGNER REQUEST

#####

def designerrequest(request):

    """

    The function to view all design request

    -----

    Parameters:

    HTTP request


    Returns:

    html page

    """

    email=request.session["email"]

    s="select
tblcustomer.cName,tblcustomer.cAddress,tblcustomer.cContact,tblplan.plan,tblplan.sqft,tbldesignrequest.dreqId from tbldesignrequest,tblcustomer,tblplan,tblrequirement where
tbldesignrequest.dEmail='"+str(email)+"' and tbldesignrequest.planId=tblplan.planId and
tblplan.reqId=tblrequirement.reqId and tblrequirement.cEmail=tblcustomer.cEmail and
tbldesignrequest.dreqStatus='requested'"

    c.execute(s)

    data=c.fetchall()

    return render(request,"designerrequest.html",{ "data":data})

#####

#                DESIGNER ADD VIDEO

#####

def designeraddvideo(request):

```

"""

The function to view all design request

Parameters:

HTTP request

Returns:

html page

"""

msg="""

dreqid=request.GET.get("id")

if(request.POST):

img=request.FILES["txtFile"]

fs=FileSystemStorage()

filename=fs.save(img.name,img)

uploaded_file_url=fs.url(filename)

s="insert into tblvideo (dreqId,video,videoStatus) values('"+dreqid+"','"+uploaded_file_url+"','video uploaded')"

try:

c.execute(s)

db.commit()

except:

msg="Sorry some error occurred"

else:

s="update tbldesignrequest set dreqStatus='video uploaded' where dreqId='"+dreqid+"'"

try:

c.execute(s)

db.commit()

except:

msg="Sorry som error"

else:

```

msg="Video added successfully"

return render(request,"designeraddvideo.html",{ "msg":msg})

#####

#                CONTRACTOR HOME

#####

def contractorhome(request):
    """
    The function for contractor home

    -----

    Parameters:

    HTTP request


    Returns:

    html page

    """
    email=request.session["email"]
    s="select * from tblcontractor where cEmail='"+email+"'"
    c.execute(s)
    data=c.fetchall()
    return render(request,"contractorhome.html",{ "data":data})

#####

#                CONTRACTOR REQUEST

#####

def contractorrequest(request):
    """
    The function to view contractor request

    -----

    Parameters:

    HTTP request

```

Returns:

html page

"""

email=request.session["email"]

s="SELECT tblcustomer.cName, tblcustomer.cAddress, tblcustomer.cContact, tblplan.plan, tblplan.sqft, tblvideo.video, tblvideo.videoId, tblcontractor.cEmail FROM tblcustomer, tblrequirement, tblplan, tblvideo, tblwork, tbldesignrequest, tblcontractor WHERE tblcustomer.cEmail = tblrequirement.cEmail AND tblrequirement.reqId = tblplan.reqId AND tblplan.planId = tbldesignrequest.planId AND tbldesignrequest.dreqId = tblvideo.dreqId AND tblvideo.videoId = tblwork.videoId AND tblwork.cEmail = '"+str(email)+"' AND tblcontractor.cEmail = tblwork.cEmail"

c.execute(s)

data=c.fetchall()

n="Select cContact from tblcustomer where cEmail='"+emai+"'"

c.execute(n)

d=c.fetchone()

contact=d[0]

msg="Your contractor request approved."

sendsms(contact,msg)

return render(request,"contractorrequest.html",{ "data":data})

#####myself#####33

def customerviewplans(request):

email=request.session["email"]

s="Select * from tblrequirement where cEmail='"+email+"' and tblrequirement.reqStatus<>'plan approved'"

c.execute(s)

data1=c.fetchall()

print(data1)

return render(request,"customerviewplans.html",{ "data1":data1 })

#####

def assignarchitect(request):

msg=""

data=""


```

if(request.POST):
    msg=""

    reqid=request.GET.get('reqid')
    aid=request.POST.get('archid')
    m="insert into tblallocation(reqid,archid,status) values('"+str(reqid)+"','"+str(aid)+"','assigned')"
    c.execute(m)
    db.commit()

    msg="Assigned successfully"

    n="select * from tblarchitect,tbllogin where tbllogin.status='1' and
tblarchitect.aEmail=tbllogin.username "
    c.execute(n)
    data1=c.fetchall()
    print(data1)
    data=showarchitect()
    return render(request,"assignarchitect.html",{ "data":data,"msg":msg,"data1":data1 })

def showarchitect():

    data=""
    c.execute("select * from tblarchitect where aEmail in(select username from tbllogin where status='1')")

    data=c.fetchall()
    return data

def contractorapprove(request):
    if(request.GET):

        email=request.GET.get("cemail")

```

```
m=("update tblwork,tblcontractor set tblwork.wStatus='Approved' where
tblwork.cEmail='"+str(email)+"' and tblcontractor.cEmail=tblwork.cEmail and
tblwork.wStatus='assigned'")
```

```
c.execute(m)
```

```
print(m)
```

```
db.commit()
```

```
return render(request,"contractorrequest.html")
```

index.html

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<!-- Meta tags -->
```

```
<title>RAD</title>
```

```
<meta name="keywords" content="Realbuild a Realestate Responsive web template, Bootstrap Web
Templates, Flat Web Templates, Android Compatible web template,
```

```
Smartphone Compatible web template, free webdesigns for Nokia, Samsung, LG, SonyEricsson,
Motorola web design" />
```

```
<meta charset="utf-8">
```

```
<meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

```
<!-- stylesheets -->
```

```
<link rel="stylesheet" href="../static/css/bootstrap.min.css">
```

```
<link rel="stylesheet" href="../static/css/font-awesome.css">
```

```
<link rel="stylesheet" href="../static/css/style.css">
```

```
<!-- google fonts -->
```

```
<link href="//fonts.googleapis.com/css?family=Open+Sans" rel="stylesheet">
```

```
<link href="//fonts.googleapis.com/css?family=Raleway:300,400,500,600,700,700i,800,900"
rel="stylesheet">
```

```
<!-- scripts -->
```

```

<script src="../../static/js/jquery.min.js"></script>

</head>

<body>
<div class="video-responsive">
<video class="video" muted="muted" loop="loop" autoplay="autoplay">
<source src="../../static/video/real2.mp4" type="video/mp4">
Your browser does not support HTML5 video.
</video>

<canvas class="canvas"></canvas>

<div id="over_video">
<div class="bg-mask">
<nav class="navbar w3-navbar">
<div class="navigation-overlay">
<div class="container-fluid">
<div class="nav-top">
<div class="w3-contact">
<!-- <a> <span class="fa fa-volume-control-phone" aria-hidden="true"> </span>+692 527 6524</a>
<a href="mailto:abcd@yoursite.com"><span class="fa fa-envelope-o"
aria-hidden="true"></span>admin@realbuild.com</a> -->
</div>
<div class="w3-socials">
<!-- <ul>
<li>
<a href="#"><span class="fa fa-facebook"></span></a>
</li>
<li>
<a href="#"><span class="fa fa-vk"></span></a>

```

```

</li>
<li>
<a href="#"><span class="fa fa-pinterest-p"></span></a>
</li>
<li>
<a href="#"><span class="fa fa-twitter"></span></a>
</li>
</ul> -->
</div>
</div>

<div class="row">

<div class="navbar-header">
<button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
<span class="sr-only">Toggle navigation</span>
<span class="icon-bar"></span>
<span class="icon-bar"></span>
<span class="icon-bar"></span>
</button>

<!-- Logo -->
<div class="logo-container">
<div class="logo-wrap">
<a href="#home" class="scroll">
RAD
</a>
</div>
</div>
</div> <!-- end navbar-header -->

```

```

<div class="col-md-8 col-xs-12 nav-wrap">
<div class="collapse text-center navbar-collapse w3ls-nav navbar-collapse">

<ul class="nav navbar-nav w3ls-nav1 text-center">

<li class="active">
<a href="/" >Home</a>
</li>
<li>
<a href="/login" >Login</a>
</li>
<li>
<a href="/architect" >Architect</a>
</li>
<li>
<a href="/designer" >Designer</a>
</li>
<li>
<a href="/contractor">Contractor</a>
</li>
<li>
<a href="/customer" >Customer</a>
</li>

</ul>
</div>
</div> <!-- end col -->
</div> <!-- end row -->

```

```

</div> <!-- end container -->

</div> <!-- end navigation -->

</nav> <!-- end navbar -->

<div class="heading">
<h1>New Way to Design is Online</h1>
<h3 align="center">One of the biggest achievements in life is considered to be building
one's own house. <br>Whether you are building a home on a plot or
have recently purchased an apartment flat, RAD.com helps
you with all your planning and design needs without needing a time
consuming and expensive visit to an engineer or a designer.</h3>

<!-- bootstrap-pop-up -->
<div class="modal video-modal fade" id="property0" tabindex="-1" role="dialog">
<div class="modal-dialog" role="document">
<div class="modal-content">
<div class="modal-header">
RAD
<button type="button" class="close" data-dismiss="modal" aria-label="Close"><span
aria-hidden="true">&times;</span></button>
</div>
<section>
<div class="modal-body">

<p>One of the biggest achievements in life is considered to be building one's own house.
Whether you are building a home on a plot or have recently purchased an apartment flat,
RAD.com helps you with all your planning and design needs without needing a time
consuming and expensive visit to an engineer or a designer.</p>
</div>
</section>

```

```

</div>
</div>
</div>
<!-- //bootstrap-pop-up -->
</div>

</div>
</div>
</div>
<style>
.video-responsive {
padding-bottom: 750px;
position: relative;
width: 100%;
}

.canvas,
.video {
left: 0;
position: absolute;
top: 0;
background: #000;
z-index: 5;
overflow: hidden;
width: 100%;
height: 750px;
object-fit: cover;
}

#over_video {

```

```
position: absolute;
width: 100%;
height: 100%;
text-align: center;
top: 0;
z-index: 10;
color: #FFF;
}
@media screen and (max-width: 1280px) {
.video-responsive {
padding-bottom: 650px;
}
.canvas,
.video {
height: 650px;
}
}
@media screen and (max-width: 1080px) {
.video-responsive {
padding-bottom: 600px;
}
.canvas,
.video {
height: 600px;
}
}
@media screen and (max-width: 568px) {
.video-responsive {
padding-bottom: 550px;
}
```



```

.canvas,
.video {
height: 550px;
}
}
</style>

<script src="../../static/js/canvas-video-player.js"></script>

<script>
var isIOS = /iPad|iPhone|iPod/.test(navigator.platform);

if (isIOS) {

var canvasVideo = new CanvasVideoPlayer({
videoSelector: '.video',
canvasSelector: '.canvas',
timelineSelector: false,
autoplay: true,
makeLoop: true,
pauseOnClick: false,
audio: false
});

} else {

// Use HTML5 video
document.querySelectorAll('.canvas')[0].style.display = 'none';

```

```

}
</script>

<!-- About us -->
<div class="agile-about" id="about">
<h3 class="center">About Us</h3>
<div class="container">
<p size="30" align="center">RAD is a team of architects, designers and contractors who love designing,
who love construction, who
love seeing a idea turn into a beautiful home.<br>
The team at RAD wishes you good luck with your design and constructions process.<br><br>

Happy Building!<br><br><br></p>
<div class="clearfix"></div>
</div>

</div>
<!-- //About us -->
<!-- Partners -->

<!-- //Partners -->
<!-- Gallery Property -->

<!-- //Gallery Property-->
<!-- our guides -->

<!-- //our guides -->
<!-- testimonials -->

```

```

<!-- //testimonials -->

<!-- contact -->

</div>

<!-- //contact -->

<!-- scripts -->

<script src="../../static/js/bootstrap.min.js"></script>

<!-- smooth scrolling -->
<script src="../../static/js/SmoothScroll.min.js"></script>
<script type="text/javascript" src="../../static/js/move-top.js"></script>
<script type="text/javascript" src="../../static/js/easing.js"></script>

<!-- here stars scrolling icon -->
<script type="text/javascript">
$(document).ready(function () {
/*
var defaults = {
containerID: 'toTop', // fading element id
containerHoverID: 'toTopHover', // fading element hover id
scrollSpeed: 1200,
easingType: 'linear'
};
*/

$.UItoTop({
easingType: 'easeOutQuart'
});

```

```

});
</script>
<!-- //here ends scrolling icon -->
<!-- smooth scrolling -->

<!-- scrolling script -->
<script type="text/javascript">
jQuery(document).ready(function ($) {
$(".scroll").click(function (event) {
event.preventDefault();
$('html,body').animate({
scrollTop: $(this.hash).offset().top
}, 1000);
});
});
</script>
<!-- //scrolling script -->

</body>

</html>

```

commonbase.html

```

<!DOCTYPE html>
<html lang="en">
<head>
<!-- Meta tags -->
<title>RAD</title>

<meta name="keywords" content="Realbuild a Realestate Responsive web template, Bootstrap Web
Templates, Flat Web Templates, Android Compatible web template,

```

Smartphone Compatible web template, free webdesigns for Nokia, Samsung, LG, SonyEricsson, Motorola web design" />

```
<meta charset="utf-8">
```

```
<meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

```
<!-- stylesheets -->
```

```
<link rel="stylesheet" href="../static/css/bootstrap.min.css">
```

```
<link rel="stylesheet" href="../static/css/font-awesome.css">
```

```
<link rel="stylesheet" href="../static/css/style.css">
```

```
<!-- google fonts -->
```

```
<link href="//fonts.googleapis.com/css?family=Open+Sans" rel="stylesheet">
```

```
<link href="//fonts.googleapis.com/css?family=Raleway:300,400,500,600,700,700i,800,900" rel="stylesheet">
```

```
<!-- scripts -->
```

```
<script src="../static/js/jquery.min.js"></script>
```

```
</head>
```

```
<body>
```

```
<div>
```

```
<video class="video" muted="muted" loop="loop" autoplay="autoplay" style="height:250px;">
```

```
<source src="../static/video/real2.mp4" type="video/mp4">
```

Your browser does not support HTML5 video.

```
</video>
```

```
<canvas style="max-height:250px;" class="canvas"></canvas>
```

```
<div id="over_video" style="max-height:250px;">
```

```
<div class="bg-mask">
```

```
<nav class="navbar w3-navbar">
```

```
<div class="navigation-overlay">
```

```

<div class="container-fluid">
<div class="nav-top">
<div class="w3-contact">

</div>
<div class="w3-socials">

</div>
</div>

<div class="row">

<div class="navbar-header">
<button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
<span class="sr-only">Toggle navigation</span>
<span class="icon-bar"></span>
<span class="icon-bar"></span>
<span class="icon-bar"></span>
</button>

<!-- Logo -->
<div class="logo-container">
<div class="logo-wrap">
<a href="#home" class="scroll">
RAD
</a>
</div>
</div>
</div> <!-- end navbar-header -->

```

```

<div class="col-md-8 col-xs-12 nav-wrap">
<div class="collapse text-center navbar-collapse w3ls-nav navbar-collapse">

<ul class="nav navbar-nav w3ls-nav1 text-center">

<li>
<a href="/" >Home</a>
</li>
<li>
<a href="/login" >Login</a>
</li>
<li>
<a href="/architect" >Architect</a>
</li>
<li>
<a href="/designer" >Designer</a>
</li>
<li>
<a href="/contractor">Contractor</a>
</li>
<li>
<a href="/customer" >Customer</a>
</li>

</ul>
</div>
</div> <!-- end col -->
</div> <!-- end row -->
</div> <!-- end container -->

```

```
</div> <!-- end navigation -->
```

```
</nav> <!-- end navbar -->
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<style>
```

```
.video-responsive {  
padding-bottom: 750px;  
position: relative;  
width: 100%;  
}
```

```
.canvas,  
.video {  
left: 0;  
position: absolute;  
top: 0;  
background: #000;  
z-index: 5;  
overflow: hidden;  
width: 100%;  
  
object-fit: cover;  
}
```

```
#over_video {  
position: absolute;
```



```
width: 100%;  
height: 100%;  
text-align: center;  
top: 0;  
z-index: 10;  
color: #FFF;  
}  
@media screen and (max-width: 1280px) {  
.video-responsive {  
padding-bottom: 650px;  
}  
  
}  
@media screen and (max-width: 1080px) {  
.video-responsive {  
padding-bottom: 600px;  
}  
  
}  
@media screen and (max-width: 568px) {  
.video-responsive {  
padding-bottom: 550px;  
}  
  
}  
</style>  
  
<script src="../../static/js/canvas-video-player.js"></script>
```

```

<script>
var isIOS = /iPad|iPhone|iPod/.test(navigator.platform);

if (isIOS) {

var canvasVideo = new CanvasVideoPlayer({
videoSelector: '.video',
canvasSelector: '.canvas',
timelineSelector: false,
autoplay: true,
makeLoop: true,
pauseOnClick: false,
audio: false
});

} else {

// Use HTML5 video
document.querySelectorAll('.canvas')[0].style.display = 'none';

}
</script>

<div style="margin-top: 300px;">
{ % block content % }

{ % endblock % }
</div>
<!-- //contact -->

```

```

<!-- scripts -->

<script src="../../static/js/bootstrap.min.js"></script>

<!-- smooth scrolling -->
<script src="../../static/js/SmoothScroll.min.js"></script>
<script type="text/javascript" src="../../static/js/move-top.js"></script>
<script type="text/javascript" src="../../static/js/easing.js"></script>

<!-- here stars scrolling icon -->
<script type="text/javascript">
$(document).ready(function () {
/*
var defaults = {
containerID: 'toTop', // fading element id
containerHoverID: 'toTopHover', // fading element hover id
scrollSpeed: 1200,
easingType: 'linear'
};
*/

$.UItoTop({
easingType: 'easeOutQuart'
});

});

</script>

<!-- //here ends scrolling icon -->

<!-- smooth scrolling -->

```

```

<!-- scrolling script -->
<script type="text/javascript">
jQuery(document).ready(function ($) {
$(".scroll").click(function (event) {
event.preventDefault();
$('html,body').animate({
scrollTop: $(this.hash).offset().top
}, 1000);
});
});
</script>
<!-- //scrolling script -->

</body>

</html>

```

Login.html

```

{% extends 'commonbase.html' %}
{% block content %}
<style>
th,td{
padding: 10px;
}
</style>
<div style="margin:100px 50px 50px 50px;">
<h1 style="margin: 50px;">Login</h1>
<center><form method="POST">
{% csrf_token %}
<table style="margin: 50px;" >

```

```

<tr>
<td>Username</td>
<td><input type="email" class="form-control" name="txtEmail" required></td>
</tr>
<tr>
<td>Password</td>
<td><input type="password" class="form-control" name="txtPassword" required></td>
</tr>
<tr>
<td></td>
<td><input type="submit" class="btn btn-primary" value="Login"></td>
</tr>
</table>
</form></center>
</div>
<script>
var msg="{ { msg} }"
if(msg!="")
alert(msg)
</script>
{% endblock %}

```

Adminbase.html

```

<!DOCTYPE html>
<html lang="en">

<head>
<!-- Meta tags -->
<title>Realistic 3D Architectural Design</title>

```

```

<meta name="keywords" content="Realbuild a Realestate Responsive web template, Bootstrap Web
Templates, Flat Web Templates, Android Compatible web template,
Smartphone Compatible web template, free webdesigns for Nokia, Samsung, LG, SonyEricsson,
Motorola web design" />

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- stylesheets -->

<link rel="stylesheet" href="../static/css/bootstrap.min.css">
<link rel="stylesheet" href="../static/css/font-awesome.css">
<link rel="stylesheet" href="../static/css/style.css">


<!-- google fonts -->
<link href="//fonts.googleapis.com/css?family=Open+Sans" rel="stylesheet">
<link href="//fonts.googleapis.com/css?family=Raleway:300,400,500,600,700,700i,800,900"
rel="stylesheet">

<!-- scripts -->
<script src="../static/js/jquery.min.js"></script>

</head>

<body>
<div>
<video class="video" muted="muted" loop="loop" autoplay="autoplay" style="height:250px;">
<source src="../static/video/real2.mp4" type="video/mp4">
Your browser does not support HTML5 video.
</video>

<canvas style="max-height:250px;" class="canvas"></canvas>

<div id="over_video" style="max-height:250px;">
<div class="bg-mask">

```

```

<nav class="navbar w3-navbar">
<div class="navigation-overlay">
<div class="container-fluid">
<div class="nav-top">
<div class="w3-contact">

</div>
<div class="w3-socials">

</div>
</div>

<div class="row">

<div class="navbar-header">
<button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
<span class="sr-only">Toggle navigation</span>
<span class="icon-bar"></span>
<span class="icon-bar"></span>
<span class="icon-bar"></span>
</button>

<!-- Logo -->
<div class="logo-container">
<div class="logo-wrap">
<a href="#home" class="scroll">
RAD
</a>
</div>
</div>

```

```

</div> <!-- end navbar-header -->

<div class="col-md-8 col-xs-12 nav-wrap">
<div class="collapse text-center navbar-collapse w3ls-nav navbar-collapse">

<ul class="nav navbar-nav w3ls-nav1 text-center">

<li>
<a href="/adminhome" >Home</a>
</li>

<li>
<a href="/adminarchitect" >Architect</a>
</li>

<li>
<a href="/admindesigner" >Designer</a>
</li>

<li>
<a href="/admincontractor" >Contractor</a>
</li>

<li>
<a href="/admincustomer" >Customer</a>
</li>

<li>
<a href="/login" >Logout</a>
</li>
</ul>
</div>
</div> <!-- end col -->

```



```
</div> <!-- end row -->
</div> <!-- end container -->
</div> <!-- end navigation -->
</nav> <!-- end navbar -->
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<style>
```

```
.video-responsive {
padding-bottom: 750px;
position: relative;
width: 100%;
}
```

```
.canvas,
.video {
left: 0;
position: absolute;
top: 0;
background: #000;
z-index: 5;
overflow: hidden;
width: 100%;

object-fit: cover;
}
```

```
#over_video {  
position: absolute;  
width: 100%;  
height: 100%;  
text-align: center;  
top: 0;  
z-index: 10;  
color: #FFF;  
}  
  
@media screen and (max-width: 1280px) {  
.video-responsive {  
padding-bottom: 650px;  
}  
  
}  
  
@media screen and (max-width: 1080px) {  
.video-responsive {  
padding-bottom: 600px;  
}  
  
}  
  
@media screen and (max-width: 568px) {  
.video-responsive {  
padding-bottom: 550px;  
}  
  
}  
  
</style>  
  
<script src="../../static/js/canvas-video-player.js"></script>
```

```
<script>
var isIOS = /iPad|iPhone|iPod/.test(navigator.platform);

if (isIOS) {

var canvasVideo = new CanvasVideoPlayer({
videoSelector: '.video',
canvasSelector: '.canvas',
timelineSelector: false,
autoplay: true,
makeLoop: true,
pauseOnClick: false,
audio: false
});

} else {

// Use HTML5 video
document.querySelectorAll('.canvas')[0].style.display = 'none';

}
</script>

<div style="margin-top: 300px;">
{ % block content % }

{ % endblock % }
</div>
```

```

<!-- //contact -->

<!-- scripts -->

<script src="../../static/js/bootstrap.min.js"></script>

<!-- smooth scrolling -->
<script src="../../static/js/SmoothScroll.min.js"></script>
<script type="text/javascript" src="../../static/js/move-top.js"></script>
<script type="text/javascript" src="../../static/js/easing.js"></script>

<!-- here stars scrolling icon -->
<script type="text/javascript">
$(document).ready(function () {
/*
var defaults = {
containerID: 'toTop', // fading element id
containerHoverID: 'toTopHover', // fading element hover id
scrollSpeed: 1200,
easingType: 'linear'
};
*/

$.UItoTop({
easingType: 'easeOutQuart'
});

});

</script>

<!-- //here ends scrolling icon -->

<!-- smooth scrolling -->

```

```

<!-- scrolling script -->
<script type="text/javascript">
jQuery(document).ready(function ($) {
$(".scroll").click(function (event) {
event.preventDefault();
$('html,body').animate({
scrollTop: $(this.hash).offset().top
}, 1000);
});
});
</script>
<!-- //scrolling script -->

</body>

</html>

```

Adminhome.html

```

{% extends 'adminbase.html' %}
{% block content %}

<style>
th,td{
padding: 10px;
}
</style>

<div style="margin:100px 50px 50px 50px;">
<h1 style="margin: 50px;">Welcome Admin</h1>

</div>

```

```

<script>
var msg="{{ msg }}"
if(msg!="")
alert(msg)
</script>

{% endblock %}

```

Customerhome.html

```

{% extends 'customerbase.html' %}

{% block content %}

<style>
th,td{
padding: 10px;
}
</style>

<div style="margin:100px 50px 50px 50px;">
<h1 style="margin: 50px;">Profile</h1>
<center><form method="POST" enctype="multipart/form-data">
{% csrf_token %}

<table style="margin: 50px;" >
{% for d in data %}

<tr>

<td>Name</td>

<td><input type="text" readonly class="form-control" value="{{ d.0 }}" name="txtName" required
pattern="[a-zA-Z ]+"></td>

</tr>

<tr>

<td>Address</td>

<td><textarea name="txtAddress" class="form-control" readonly required>{{ d.1 }}</textarea></td>

</tr>

<tr>

```

```

<td>Contact</td>

<td><input type="text" class="form-control" readonly value="{{ d.2 }}" name="txtContact" required
pattern="[6789][0-9]{9}"></td>

</tr>

<tr>

<td>Email</td>

<td><input type="text" class="form-control" readonly value="{{ d.3 }}" name="txtEmail" required
></td>

</tr>

{% endfor %}

</table>

</form></center>

</div>

<script>

var msg="{{ msg }}"

if(msg!="")

alert(msg)

</script>

{% endblock %}

```

Designerhome.html

```

{% extends 'designerbase.html' %}

{% block content %}

<style>

th,td{

padding: 10px;

}

</style>

<div style="margin:100px 50px 50px 50px;">

<h1 style="margin: 50px;">Designer</h1>

<center><form method="POST" enctype="multipart/form-data">

```

```

{% csrf_token %}

<table style="margin: 50px;" >

{% for d in data %}

<tr>

<td colspan="2"></td>

</tr>

<tr>

<td>Name</td>

<td><input type="text" class="form-control" name="txtName" value="{{d.0}}" required pattern="[a-zA-Z]+"></td>

</tr>

<tr>

<td>Address</td>

<td><textarea name="txtAddress" class="form-control" required>{{d.1}}</textarea></td>

</tr>

<tr>

<td>Contact</td>

<td><input type="text" class="form-control" name="txtContact" value="{{d.2}}" required pattern="[6789][0-9]{9}"></td>

</tr>

<tr>

<td>Email</td>

<td><input type="text" class="form-control" value="{{d.3}}" name="txtEmail" required ></td>

</tr>

{% endfor %}

</table>

</form></center>

</div>

<script>

var msg="{{msg}}"

if(msg!="")

```



```
alert(msg)
```

```
</script>
```

```
{% endblock %}
```

Architecthome.html

```
{% extends 'architectbase.html' %}
```

```
{% block content %}
```

```
<style>
```

```
th,td{
```

```
padding: 10px;
```

```
}
```

```
</style>
```

```
<div style="margin:100px 50px 50px 50px;">
```

```
<h1 style="margin: 50px;">Profile</h1>
```

```
<center><form method="POST" enctype="multipart/form-data">
```

```
{% csrf_token %}
```

```
<table style="margin: 50px;" >
```

```
{% for d in data %}
```

```
<tr>
```

```
<td>Name</td>
```

```
<td><input type="text" readonly class="form-control" value="{{ d.0 }}" name="txtName" required  
pattern="[a-zA-Z ]+"></td>
```

```
</tr>
```

```
<tr>
```

```
<td>Address</td>
```

```
<td><textarea name="txtAddress" class="form-control" readonly required>{{ d.1 }}</textarea></td>
```

```
</tr>
```

```
<tr>
```

```
<td>Contact</td>
```

```
<td><input type="text" class="form-control" readonly value="{{ d.2 }}" name="txtContact" required  
pattern="[6789][0-9]{9}"></td>
```

```

</tr>
<tr>
<td>Email</td>
<td><input type="text" class="form-control" readonly value="{{ d.3 }}" name="txtEmail" required
></td>
</tr>
{% endfor %}
</table>
</form></center>
</div>
<script>
var msg="{{ msg }}"
if(msg!="")
alert(msg)
</script>
{% endblock %}

```

Contractorhome.html

```

{% extends 'contractorbase.html' %}
{% block content %}
<style>
th,td{
padding: 10px;
}
</style>
<div style="margin:100px 50px 50px 50px;">
<h1 style="margin: 50px;">Contractor</h1>
<center><form method="POST" enctype="multipart/form-data">
{% csrf_token %}
<table style="margin: 50px;" >
{% for d in data %}

```

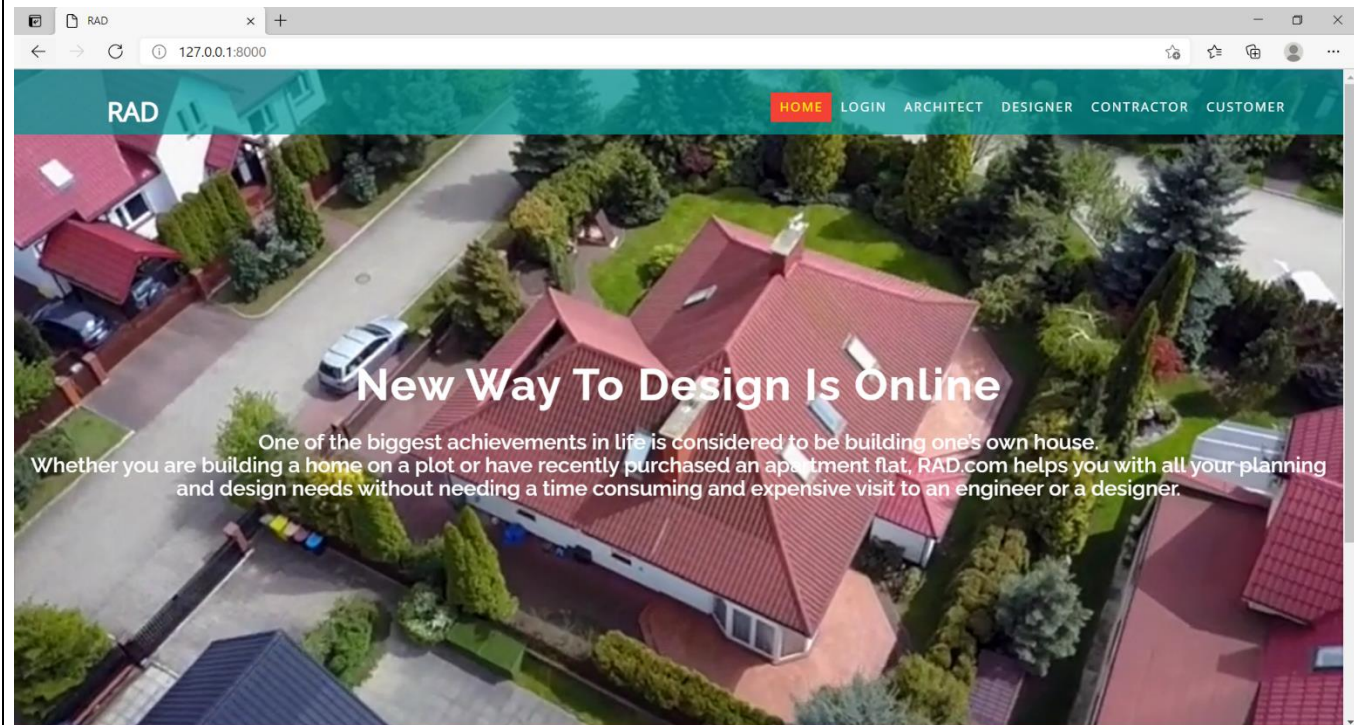
```

<tr>
<td>Name</td>
<td><input type="text" class="form-control" name="txtName" value="{{ d.0 }}" required pattern="[a-zA-Z ]+"></td>
</tr>
<tr>
<td>Address</td>
<td><textarea name="txtAddress" class="form-control" required>{{ d.1 }}</textarea></td>
</tr>
<tr>
<td>Contact</td>
<td><input type="text" class="form-control" name="txtContact" value="{{ d.2 }}" required
pattern="[6789][0-9]{9}"></td>
</tr>
<tr>
<td>Email</td>
<td><input type="text" class="form-control" name="txtEmail" value="{{ d.3 }}" required ></td>
</tr>
{% endfor %}
</table>
</form></center>
</div>
<script>
var msg="{{ msg }}"
if(msg!="")
alert(msg)
</script>
{% endblock %}

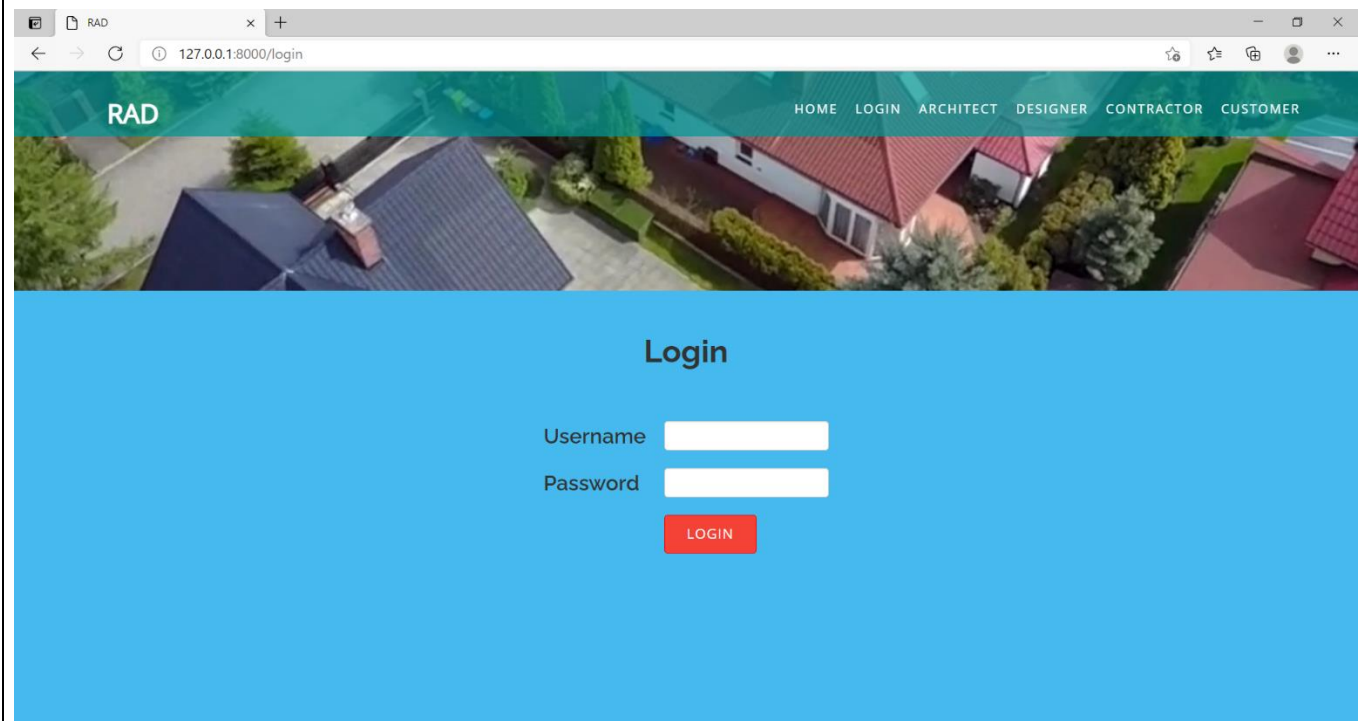
```

User Interface

Home Page



Login Page



Registration page

Customer

Name

Address

Contact

Email

Password Password Pattern - A combination of Alphabets, Special Characters and Numbers

Upload Photo No file chosen

Admin Page

Realistic 3D Architectural Design

127.0.0.1:8000/admincontractor

RAD

HOMEARCHITECTDESIGNERCONTRACTORCUSTOMERLOGOUT

Contractor

NAME	ADDRESS	CONTACT	EMAIL	PREVIOUS WORK	PROFILE
Durga Dyal	Niranjan, Juhu Tara Rd, Marine Lines, Goa	9228089191		durga@gmail.com	
Jaish Kulkarni	Siddiambur Bazar, Hyderabad	9412364111		jaish@gmail.com	
Sonam Tella	Geeta Mandir, Pratapnagar Road, Pratapnagar	9112741998		sonam@gmail.com	

Architect Home Page

Architect Home Page

Browser tabs: RAD, localhost / localhost / dbrealisti: |

Address bar: 127.0.0.1:8000/architecthome

Page Header: RAD HOME REQUEST MY PLANS LOGOUT

Profile

Name: Milton Bell

Address: 9, Main Market, Moti Nagar, Delhi, 110015

Contact: 8052740996

Email: B-ARCH

Contractor Home Page

Contractor Home Page

Browser tabs: RAD, localhost / localhost / dbrealisti: |

Address bar: 127.0.0.1:8000/contractorhome

Page Header: RAD HOME VIEW WORK LOGOUT

Contractor

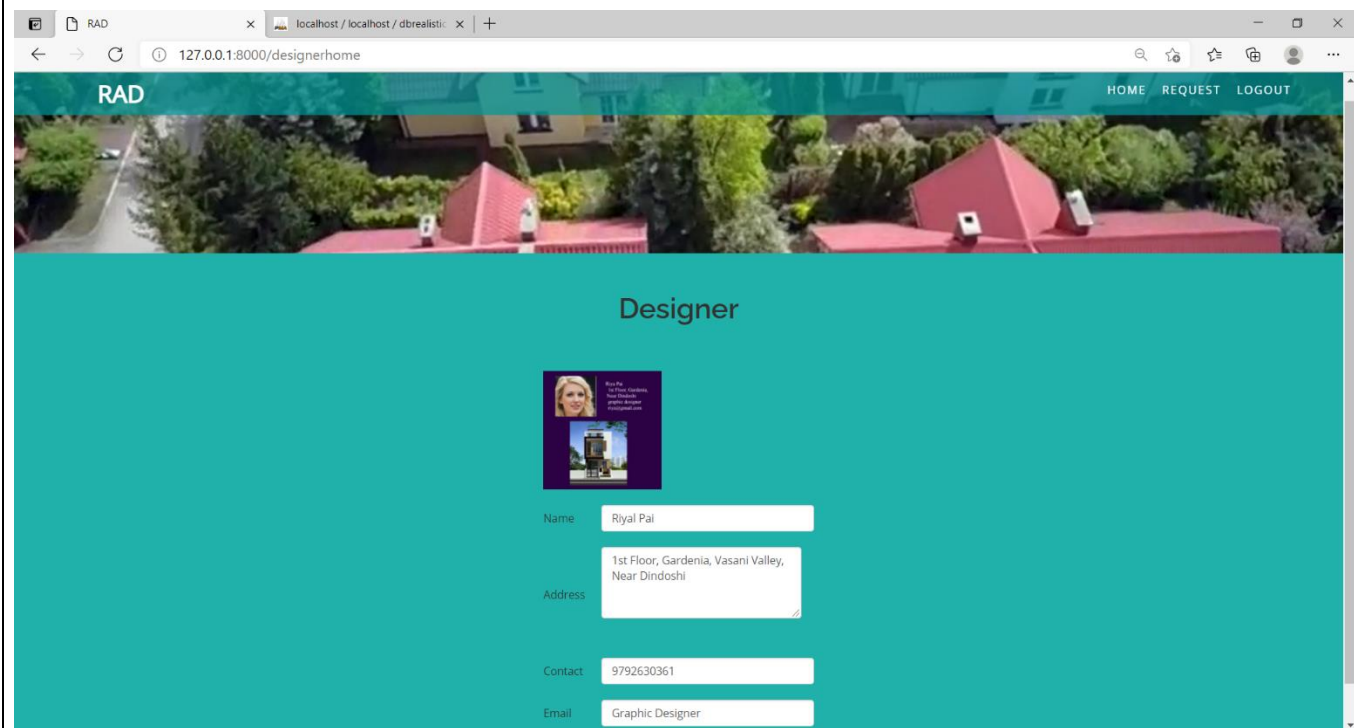
Name: Sonam Tella

Address: Geeta Mandir, Pratapnagar Road, Pratapnagar

Contact: 9112741998


Email: /media/4d9dd658da8a62f5f9726c903af2be98.jpg

Designer Home Page



RAD HOME REQUEST LOGOUT

Designer



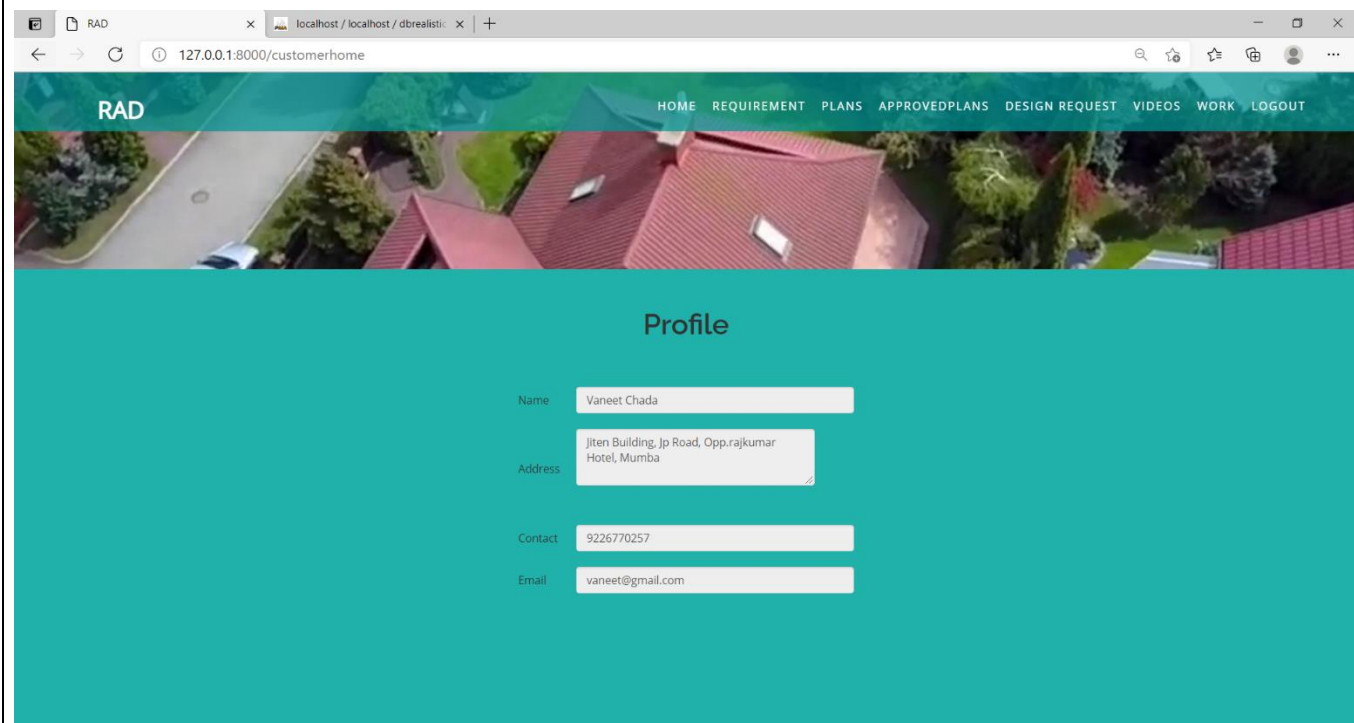
Name

Address

Contact

Email

Customer Home Page



RAD HOME REQUIREMENT PLANS APPROVEDPLANS DESIGN REQUEST VIDEOS WORK LOGOUT

Profile

Name

Address

Contact

Email