A PROJECT REPORT ON

ASSISTANT 360

Dissertation Submitted by

Akshaya J H Pillai (Reg.No. - 32020894002)

Fathima Ajith Rukhsana (Reg.No.- 32020894003)

In partial fulfillment of the requirement for the award of the Degree of

BACHELOR OF COMPUTER SCIENCE

of the

UNIVERSITY Of KERALA

College of Applied Science, Kalanjoor

March 2023

CERTIFICATE

This is to certify that the project work entitled **"ASSISTANT 360"** submitted to **University of Kerala** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Computer Science**. This is a record of original work done by **Akshaya J H Pillai (Reg.No. - 32020894002), Fathima Ajith Rukhsana (Reg.No.- 32020894003),** during the period of study in **IHRD Kalanjoor,** under my supervision and guidance . This project report or part thereof was not submitted earlier for any other similar Degree / Diploma/ Associate ship / Fellowship.

Head of the Department **Internal Guide**

Submitted for University Examination held on ……...

Internal Examiner **External Examiner**

DECLARATION

We**, Akshaya J H Pillai (Reg.No. - 32020894002), Fathima Ajith Rukhsana (Reg.No.- 32020894003),** declare that the project on "**ASSISTANT 360**" is the result of original work done by us and to the best of our knowledge, a similar work has not been submitted earlier to the **University Of Kerala** or any other Institution , for fulfillment of the requirement of a course of study .

This project report is submitted on partial fulfillment of the requirement for the degree of **Bachelor of Computer Science** of University of Kerala

Akshaya J H Pillai

Fathima Ajith Rukhsana

ACKNOWLEDGEMENT

We would like to thank the Almighty God for showering all His blessings up on us for the successful completion of our project ASSISTANT 360. We are thankful to our principal Mrs. Thara K S (Principal, College of Applied Science, Kalanjoor) for providing us all the facilities during our course of study. We express our heartily thanks to Mrs. Bindu S (HOD- Computer Department, College of Applied Science, Kalanjoor) for her valuable guidance and support throughout the project work. We express our heartily thanks to internal guide Mrs. Remya Vijayan (Internal Guide, College of Applied Science, Kalanjoor) who has guided us with a great deal of patience. We wish to express our deep sense of gratitude and sincere thanks to my parents and all well-wishers who have directly and indirectly contributed a lot of towards our project work.

ABSTRACT

Our modern world time have an imported role in every once life, no one have time to waste. But for some task like finding a proper assistance, services, personals on time have become difficult and more time consuming and also it need good contact on a special location .these all factor had led to more complexity and ambiguity .it can be managed with the help of our purposing solution “ASSISTANT 360”

ASSISTANT 360 is a web based solution that was managed with the help of internet. As internet is as well familiar technique that used in our today it is more applicable and cost less to implement. Our proposing solution has to manage details such as assistance, services, and personals with accuracy. This system will help to maintain a feasible atmosphere communication among a user and a service provider.

TABLE OF CONTENTS

SOURCE CODE

1. INTRODUCTION

1.1 ABOUT THE PROJECT

Today in the fast moving environment it is important to the role of local workers in our day to day life. But due to the shortage of skillful persons there is a bit harder to find them for our uses.

Assistant 360 is an online portal which assist a user in all manner. It helps users to find out persons that are required in daily life. Persons such as electricians, plumbers, Masons, carpenter, Painters, Baby care taker, home nurses, welding workers, excavator operators etc. Staffs can register in to this website by providing their details. Users can hire them according to their needs which can be for a single day or more than one day. Assistant 360 provides all kinds of assistance need in daily life .Users can register in to the system and look for workers for their needs.

Another section in assistant 360 includes a Cab section which we provide you car for your travelling. The main thing in this area is that we provide female driven cars as a part of women empowerment and safety. This ensures safer journey for lady passengers. This cab services can be accessible at any time. The real-time tracking of these are possible through GPS.

Also we provide a section for local transportation. Local transportations like Autorikshaws can register into this website and users can call them at any time for travelling. Here we also include time chart of buses and trains through the place you are at. In this system payment can be made by cash or cards.

The project is developed using Python as designing tool and MySQL as database. Python is a powerful tool for web programming From Microsoft and is the front end of this project with MySQL as backend.

2. SYSTEM ANALYSIS

System analysis works with users to identify goals and build systems to achieve them. System analysis is the important phase of any system development process. The system is studied to the minutest details and analyzed. The system analyst plays the role of an interrogator and dwells deep into the working of the present system. In analysis, a detailed study of these operations performed by a system and their relationship within and outside of the system is done. A key question considered here is "what must be done to solve the problem? One aspect of analysis is defining the boundaries of the system and determining whether or not the candidate system should be considered.

2.1 EXISTING SYSTEM

The major activities in the system study are studying the ways an organization currently retrieving and processing data to produce information it involves the study of various operation performed by the system and their relationship within and outside the system. The system study is usually done on the basic approaches of understanding the problem or user requirements. To understand the user requirements the analyst communicates with the user or customer and studies the basic requirements of the customer. The study of the existing system was necessary to carry out the preliminary investigation.

Disadvantages:

- Hard to find skillful local workers.
- Can't afford their payment sometimes.
- Lack of jobs for local workers.
- People in rural areas have no medium for giving their vehicles for fleet purposes
- No existing system to buy favorite food from any hotel through online system.

- No existing system for bringing back the item we have accidentally left somewhere we know.
- No way to find autorikshaw services through online.
- No way to ensure safety for alone traveling lady passengers.
- No way to find local transport such as bus and train from a specific place.

2.2. PROPOSED SYSTEM

Here the existing system is manual and the proposed system is the automated version of existing system. Proposed system is designed based on the objectives prepared to implement and fulfill existing system draw back. Proposed system is software to be developed for the computerization of existing system and also help in inventory management. This system will overcome the disadvantages of existing system. The main aim of this proposed system is to remove the limitations of existing system through the computerization and increase the efficiency.

Advantages:

- Can find skillful local workers such as home nurses, carpenters, masons, painters, welders etc.
- Provision for chat with them regarding their payment and work details.
- Jobs for professional workers are easily available.
- The ideal solution for Fleet Management, is specially designed for Cab, Truck and Pickup Operators who operate a fleet of vehicles for hire. The vehicles maybe Cars, pickup vans, buses, etc. FLEET-CAB enables you to manage demand-supply position of vehicles, keep a tab on the mileage and reduce maintenance expenditure and operating costs. It automates your operations like accepting customer bookings, allocation of vehicles, tracking your vehicles etc.
- Buy your favourite food from your favourite hotel. And pay from your credit or debit card.
- Bring the items that you where accidently left in your friends home or shops.

- Can book for taxi or autorishaws by sitting in your home.
- Alone traveling lady passengers can now travel safely by SHE TAXIs.
- Find local transport such as buses or trains from anywhere.
- Saves user time, cost and efforts.
- Maintenance is easy and performance is good
- The website is flexible and user-friendly

2.3 FEASIBILITY ANALYSIS

Preliminary investigations examine project feasibility, the like hood of the system to be useful to the user community. Feasibility study is the major step in the system development life cycle. The main objective of feasibility study is to test the economic, technical and operational feasibility developing a system.

This analysis is done by investigating the existing system in the area under investigation and generating an idea about the new system. Mainly three key considerations are involved in the feasibility analysis of our system. And they are as follows:

- Economic feasibility
- Technical feasibility
- Operational feasibility

2.3.1 Economical feasibility

The economic feasibility was aimed at determining whether the cost creating the system so great that the project must be undertaken. The existing resources are enough for implementing the system and also for further development. The fully computerized system will reduce the human effort and valuable time by simplifying day –to-day activities. In this, economic or financial feasibility is considered. The basic resources to be considered are:

- Time spend by analysis team
- Cost of doing the full system study
- Estimated cost of hardware
- Estimated cost of software and software development

E-Recruitment Portal Service uses ASP.NET for summing up the projects main features and taking SQL Express server as the backend. Both front end and back end are open source. When compared to the advantage obtained from implementing the system its cost is affordable. Modification can be done without much work. Since the proposed system is financially beneficial, it is judged to be economically feasible.

2.3.2 Technical feasibility

Technical feasibility centers around the existing computer system and to what extend it can support the proposed addition. The main consideration is to be given at the study available resources of the organizations where the project is to be developed and implemented. Here the system analyst evaluates the technical merits of the given system emphasis on the performance, reliability, maintainability, and productivity.

E-Recruitment Portal Service reduces a significant amount of speed and reliability. It is very user friendly. There is no doubt that once the system is implemented it will reduce the precious time of the users considerably and enhance system performance. The system is module based which can be used to expand the system in future. So the system is technically feasible.

2.3.3 Operational feasibility

Once it is determined that the proposed system is both economically and technically feasible, it has to be seen if it is operationally feasible. Operational feasibility is depended upon determining human resources for the project. It refers to protecting whether the system will operate and be used once it is installed.

The organization is convinced of the benefit resulting from implementing of the system and hence the proposed system is operationally feasible. As a whole, the system will help the organization to achieve a high degree of performance. Hence the system is totally a feasible one.

3. REQUIREMENT ANALYSIS &SPECIFICIFICATION

3.1 REQUIREMENT ANALYSIS

Requirement analysis is the process of analyzing the requirements with the potential goal of improving or modifying it. Mainly it contains the analyzing phase of the existing system and its features and also the proposed system. Analyzing its advantages and disadvantages the proposed system can be designed which can avoid all the complexities, inabilities and the disadvantages of the existing system. The new system requirements are defined during this phase. The requirements of the desired software product are extracted.

To design a new system we need the requirements of the system and the description of the system. Based on the business scenario the software Requirement Specification document is prepared in this phase. The functional requirement of the software that is to build is specified here. The specifications are intended to guide the group through the development process. It explains all the process, activities, relationships and all other organizational objectives.

Requirement analysis is done in order to understand the problem with which the software system to solve. For example, the problem could be automating an existing manual process or developing a completely new automated system or the combination of the two. For large systems that have large number of features and need to perform many different tasks, understanding the requirements of the system is a major task. The emphasis in requirement analysis is on identifying what is needed from the system and not how the system will achieve its goal. This task is complicated by the fact that there are often at least two parties involved in software development, a client and a developer. There are two major activities in the phase-problem understanding or analysis and requirement specification. In problem analysis the analyst has to understand the problem and its context. Such analysis typically requires a thorough understanding of the existing system and a part of which must be automated. With the analysis of the current system the analyst can understand the reason for automation and what affects the automation system might have. The goal of this activity is to understand the requirements of the new system to be developed. Requirement analysis understands the users requirement within the framework of the

organization's objectives and the environment in which the system is being installed. Consideration is given to the user's resources as well as finance. User's requirements have been identified as follows.

❖ **Pre -defined Questions**

It allows analysts to collect information about the various aspects of the system from large number of persons the use of standardized question format can yield more reliable data than other technique.

❖ **Interview**

Analysts use interviews to collect information from individuals or from groups. The respondents are generally current users of the existing system. In News portal interviews with users, correspondents etc. are conducted to collect information.

3. Record Interview

In record review analysts examine information that has been recorded about the system and the users. Record inspection can be performed at the beginning of the study as an introduction, or later in the study as a basis for comparing actual operations with what the records indicate should behappening. In news portal record review or inspection is performed at the beginning of its study as an introduction.

3.2 SYSTEM REQUIREMENT SPECIFICATION

A System Requirements Specification (abbreviated SRS when need to be distinct from a Software Requirements Specification SRS) is a structured collection of information that embodies the requirements of a system.

A business analyst sometimes titled system analyst, is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the systems development life cycle domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers.

3.2.1Functional Requirements

3.2.2 Non-Functional Requirements

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. Non-functional requirements are “system shall be requirement ". Non-functional requirements are often called qualities of a system. Other terms for non-functional requirements are "constraints", "quality attributes”, “quality goals", "quality of service requirements" and "non-behavioral requirements. Some of the non-functional requirements are mentioned below

1.Usability: The system shall have a clean interface with only needed features, clear terminology and tool tips wherever necessary. Warnings or alerts shall be specified in clear way.

2. **Efficiency:** The system shall respond to different searches being conducted like searching particular job, search applicant, etc. in a very fast way.

3.Interoperability: The system shall be able to interact with other systems. The system should able to be supported at least one software which has a relationship with recruitment process

4.**Portability:**The system shall be independent of the specific technological platform used to implement it.

5.Reliability: Reliability defined as a measure of the time between failures occurring in a system (measure show frequently the system fails), so that the system shall operate without any failure for a particular period of time

6.Availability: Availability measures the percentage of time the system is in its operational state so that the system shall be available for use 24 hours per day and 365days per year.

3.3 Hardware and Software Requirements

- **Hardware Specification**

Operating System : Windows 10
Front End : Python
Back End : MySQL
Web Server : IIS
Web Browser : Google chrome

- **Software Specification**

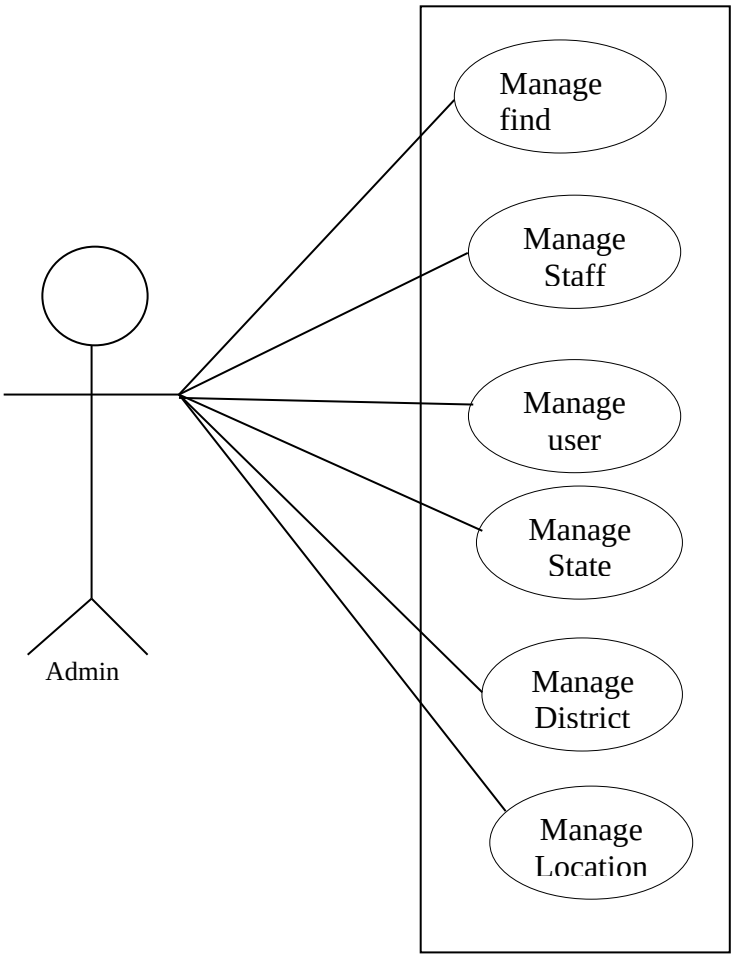
Processor : i3 or above
Processor Speed : 3.4 GHz
Hard Disk Space : 500 GB
Main Memory : 4 GB
Mouse : 3 Button
Key Board : 104 Keys Multimedia Keyboard

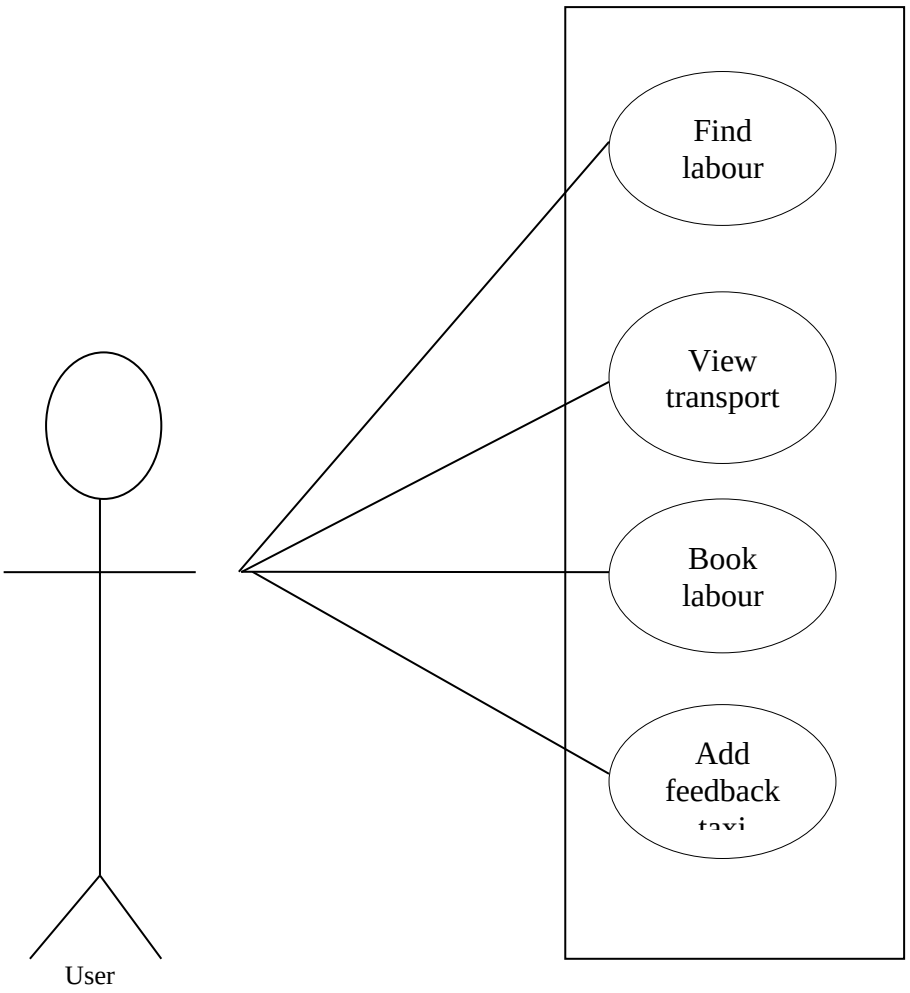
3.4 USECASE DIAGRAM

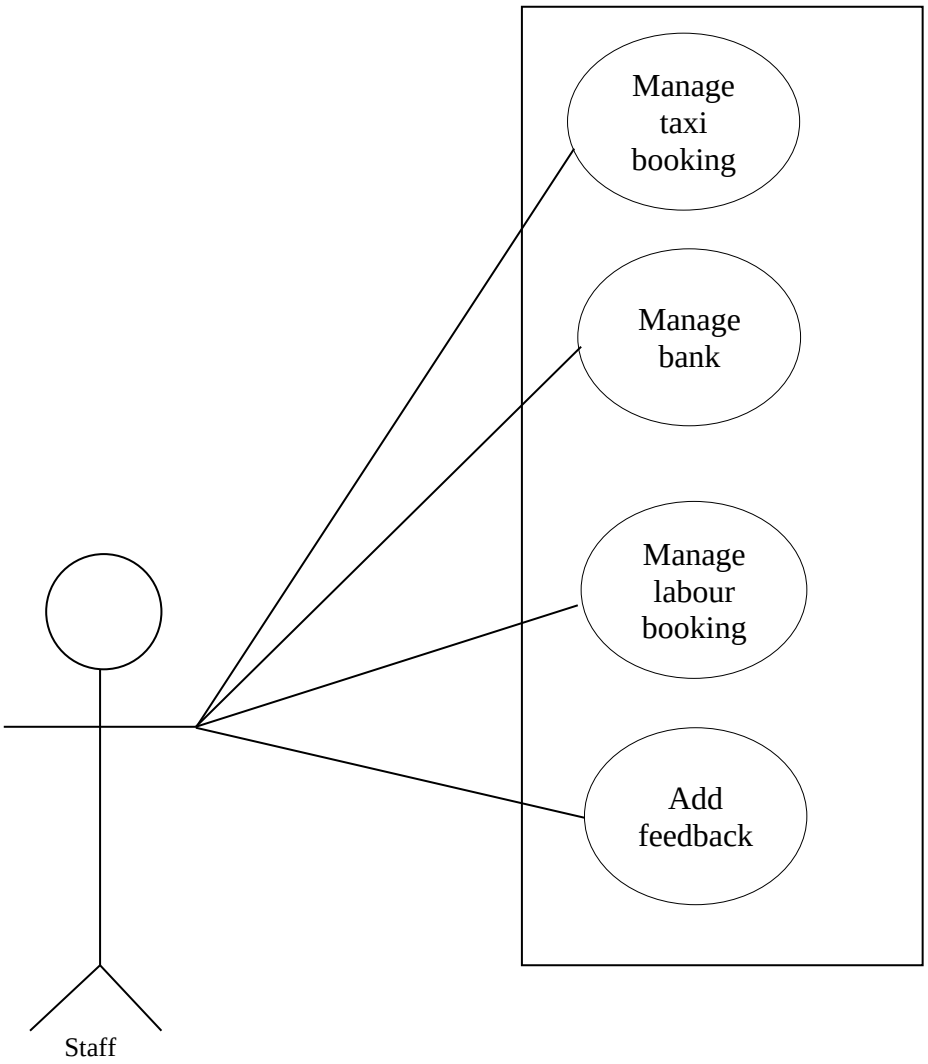
Use case diagram comprises of use cases and actors such that there would be various kinds of relationships among the use cases and the actors. A use case diagram shows all the actions that a particular actor needs to perform throughout the system at any point of time. There would be only one use case diagram per each system.A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

The purpose of use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose. Because other four diagrams (activity, sequence, collaboration and State chart) are also having the same purpose. So we will look into some specific purpose which will distinguish it from other four diagrams.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analyzed to gather its functionalities use cases are prepared and actors are identify




Find labour
View transport
Book labour
Add feedback taxi
User


Manage taxi booking
Manage bank
Manage labour booking
Add feedback
Staff

4. SYSTEM DESIGN

System design is a reduction of an entire system by studying the various operations performed and their relationships within the system and the requirements of its success. One aspect of design is defining the boundaries of the system and determining whether or not the candidate system should consider other related system. System can be defined, as an orderly grouping of interdependent components can be simple or complex. The idea of the systems has been most practical and necessary in computerizing the interrelationships and integration of operations, especially when using computers. Thus it's a way of thinking organizations and their problems.

An organization consists of several interrelated and interlocking components. Anticipated effects are the high quality information, include performance etc, and the anticipated effects include the possible threat to employees in their works and the feeling of intimidation by the users who have limited training in the new computer. Since all jobs are done manually, it became a time consuming and lengthy process. If you want to get a certificate, then the staff has to check all the registers and books. Also the details will be in different register. It takes a long time to verify the register and then issuing the certificate. This is not an easy task when the data is bulk. Once the design is completed, the analyst has a firm understanding of what is to be done.

The most creative and challenging phase of the system life cycle is system design. The term design describes a final system and the process by which it is developed .It refers to the technical specifications that will be applied in implementing the candidate system .It also includes the construction of programs and program testing. The first step in the system design is to determine how the output is to be produced and in what format. Samples of the output and the inputs are also presented .In the second step, input data and master files are to be designed to meet requirement of the proposed output .The processing phases are handled through program construction and testing, including a list of the programs needed to meet the system's objectives and complete documentation. Finally details related to justification of the system and an estimate of the impact of the candidate system on the user and organization are documented and evaluated by management as a step towards implementation. The final report prior to the implementation phase includes procedure flow chart, record

lay outs, and a workable plan for implementing the candidate system. System design has two phases:

- Logical
- Physical

The logical design reviews the present physical system, prepares the input and output and also prepares a logical design walk- through. We have to deal with how to take entries required and whether and how to process the user data. Also we have to deal with how to present the data in an informative and appealing format .This design also involves the methodology to store, modify and retrieve data from the data base as per the requirement. Physical design maps out the details of the physical system, plans the system implementation, devices a test and implementation plan and new hardware and software. We have to decide how and where to store the input data and how to process it so as to present it to the user in an easy, informative and attractive manner .A major step in the design is the preparation of input and output report in a form acceptable to the user .In this a data entry operator can feed the relevant details asked by the system for a particular task as input.

4.1 INPUT DESIGN

Input designing is the basic theory to be considered during system study. The input media used in the system is the keyboard. Details are entered in the system through different data entry screens. The system is designed in a user-friendly manner. Appropriate error messages are displayed when a false data is entered. Design of the system is web-oriented and is highly interactive to the users. The user interface design is very important for any application. The interface design defines how the software communicates within itself, to system that interpreted with it and with human who use it. The interface design is very good; the user will fall into an interactive software application.

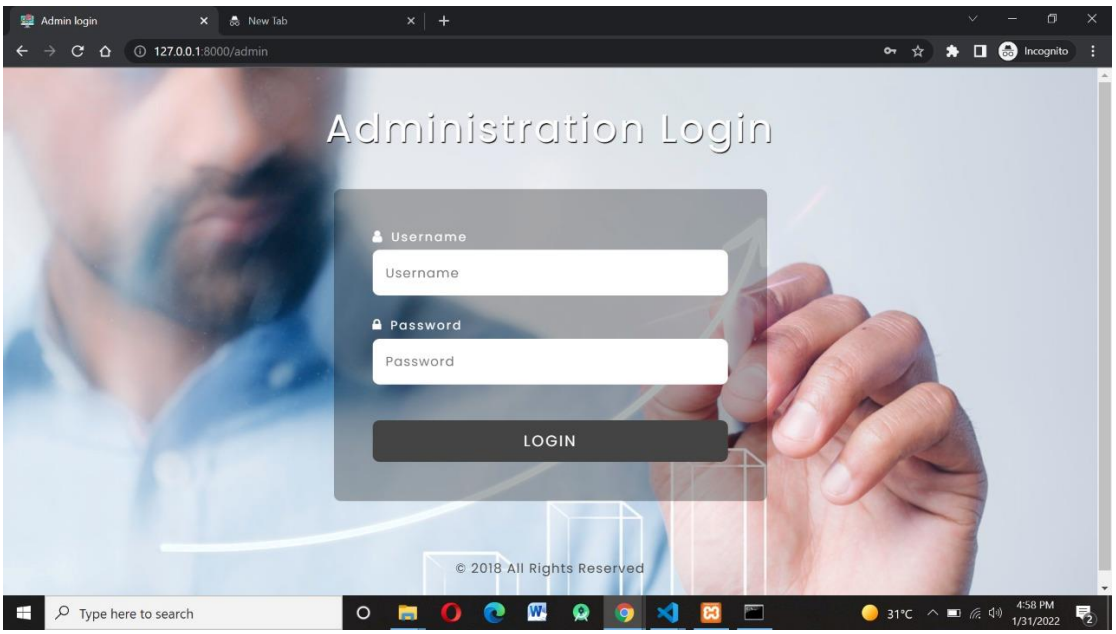
The input design is the process of converting the user-oriented description of inputs into a programmer-oriented specification. The objective of input design is to create an input layout that is easy to follow and prevents the user from committing errors. It covers all phases of input, right from the creation of initial databases to the actual data entry into the system. The input design is the link that ties the system into the world of its users. Hence, lays its importance in the design phase. The input design makes sure that while entering data, the end-users understand the format in which the data is to be

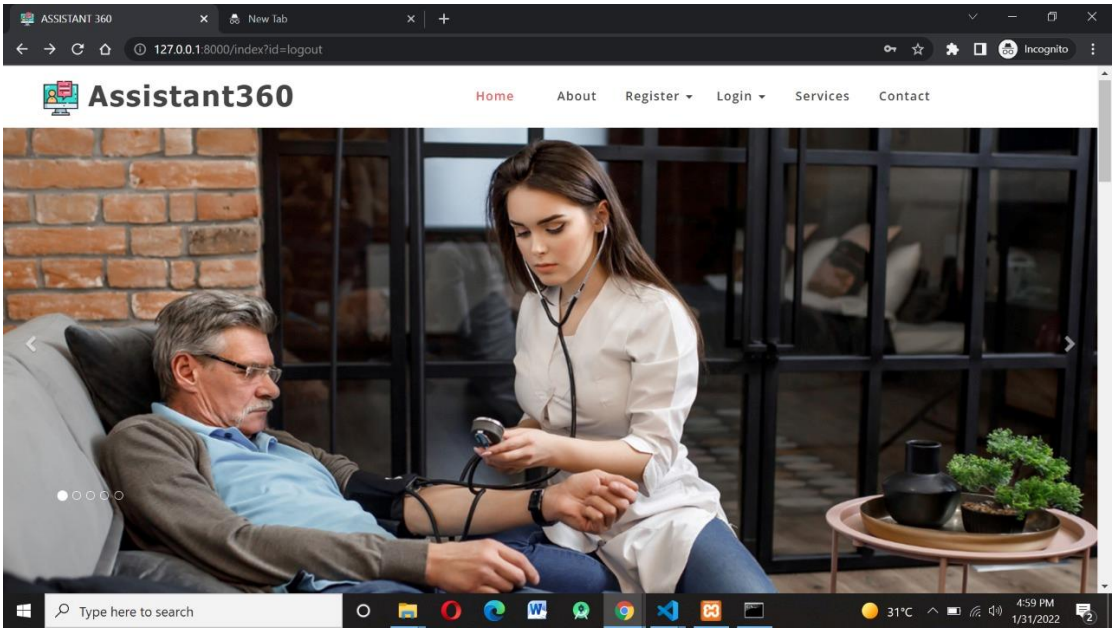
entered so that it is accepted by the system, the data values that are mandatory for the system to function, the order in which transactions need to be processed etc.

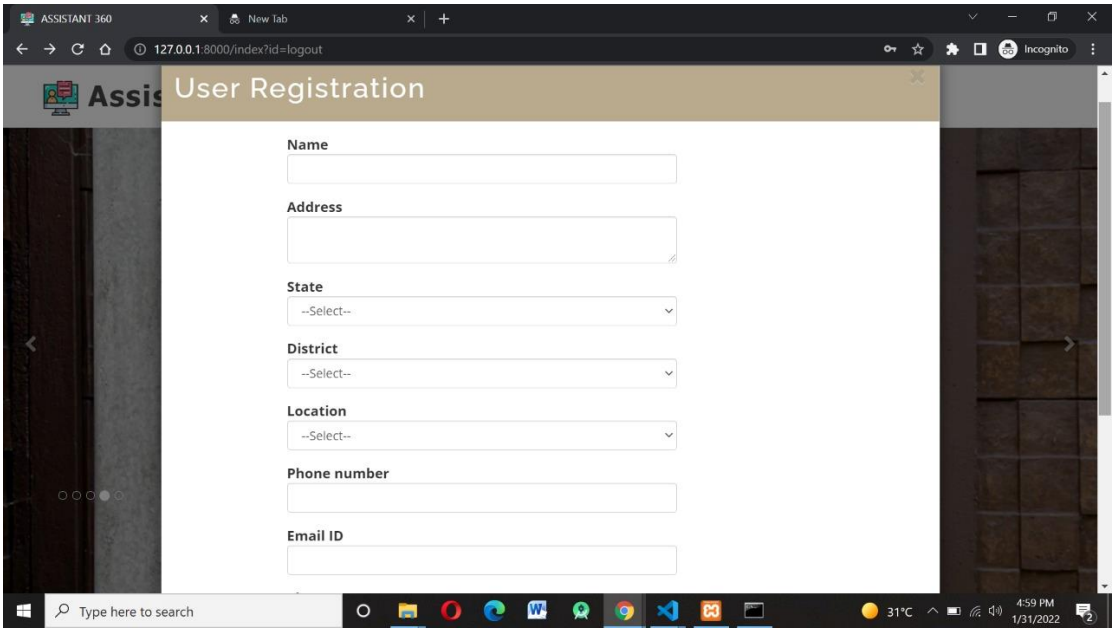
The goal designing input data is to make the automation as easy and free from errors as possible. For providing a good input design for the application easy data input and selection feature and adopted. The input design requirements such as user friendliness, consistent format and interactive dialogue for giving the right message and help for the user at right time are also considered for the development of this project. Input design, involves determining the record media, method of input, speed of capture and entry to the system.

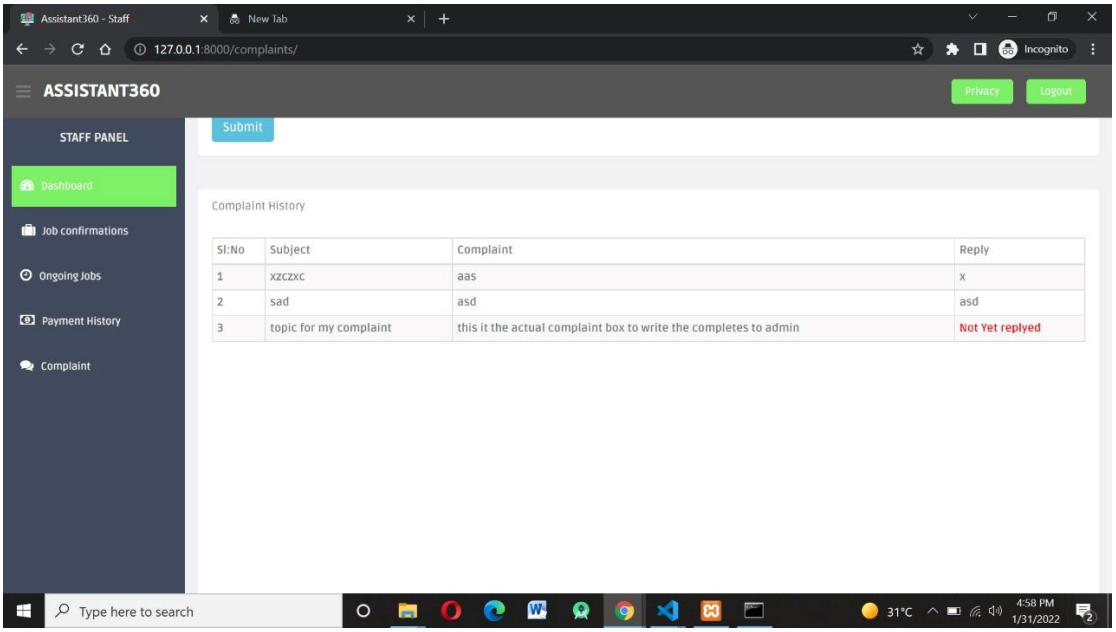
4.2 OUTPUT DESIGN

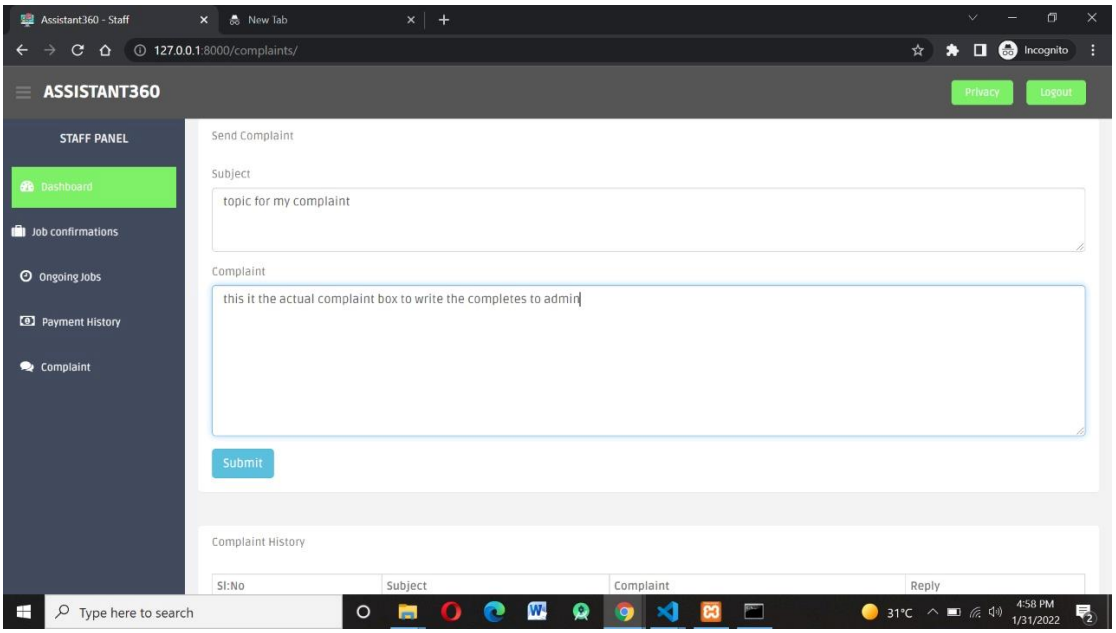
It is the most important and direct source of information to the user. Efficient, intelligible output design improves the system relationship with the user and helps the decision making. Computer outputs are the most important and direct source to the user. An efficient output of the system improves the interaction of the system with the user and it provides his/her required information. The output can be displayed on the screen or copied. In our system, hard copies are preferred because a document for further references. Careful considerations have been given while developing the output reports as it helps in decision making. Other than the remote system desktops visuals, received messages from other systems etc. are also outputted.

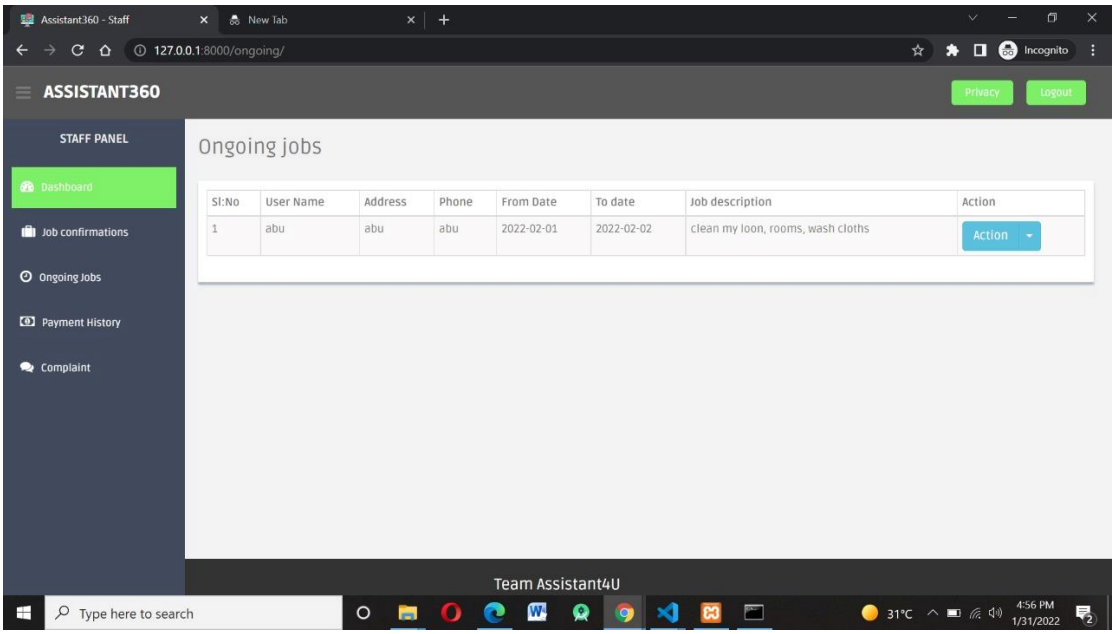


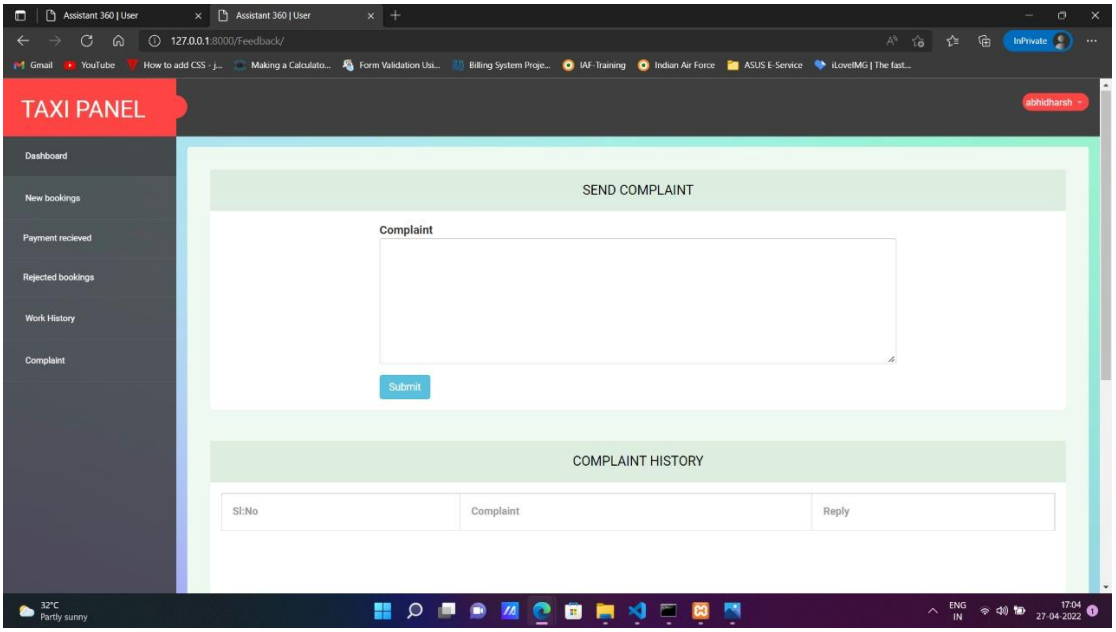

ASSISTANT 360
New Tab
127.0.0.1:8000/index?id=logout
Incognito
Assistant360
Home
About
Register
Login
Services
Contact
Type here to search
31°C
4:59 PM
1/31/2022

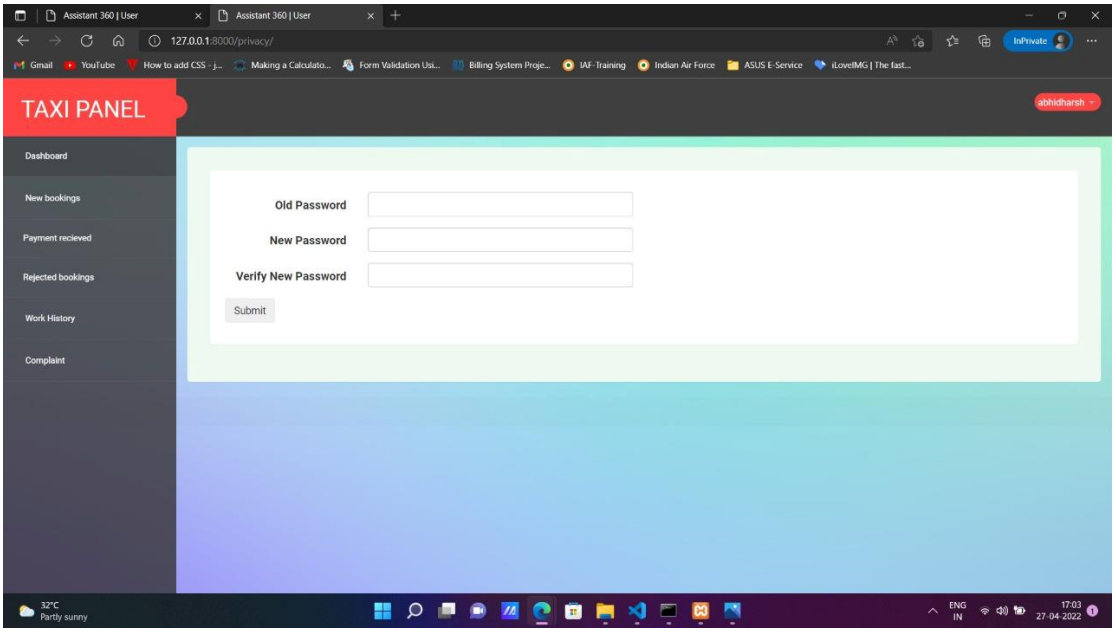

ASSISTANT 360
New Tab
127.0.0.1:8000/index?id=logout
Incognito
User Registration
Name
Address
State
--Select--
District
--Select--
Location
--Select--
Phone number
Email ID
Type here to search
31°C
4:59 PM
1/31/2022

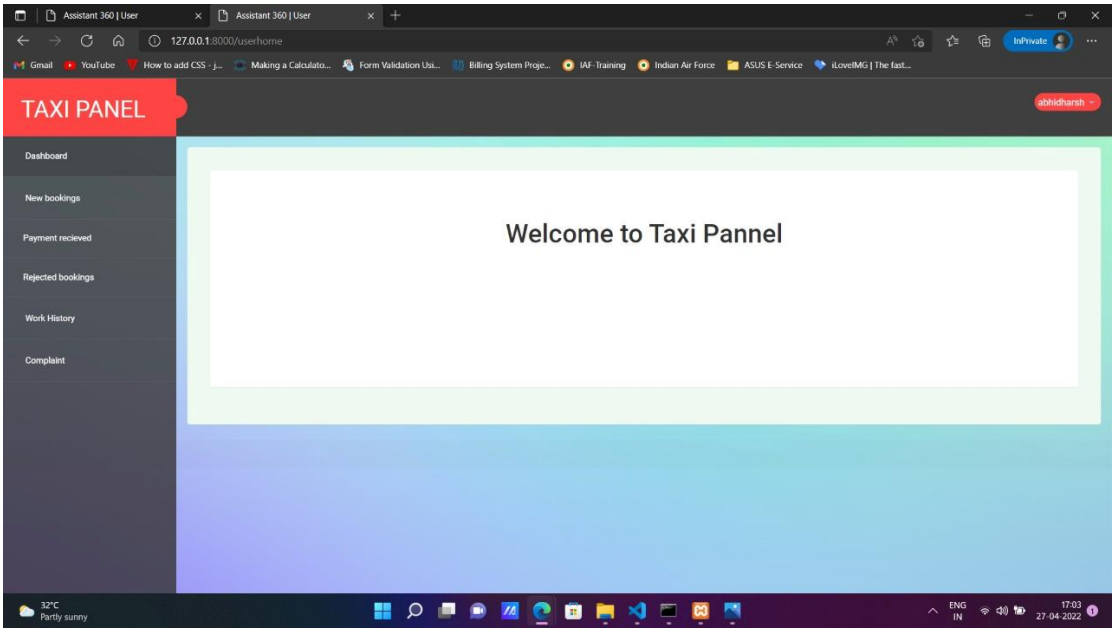

Assistant360 - Staff
New Tab
127.0.0.1:8000/complaints/
Incognito
ASSISTANT360
Privacy
Logout
STAFF PANEL
Submit
Dashboard
Job confirmations
Ongoing Jobs
Payment History
Complaint
Complaint History
Sl:No
Subject
Complaint
Reply
1
xzczxc
aas
x
2
sad
asd
asd
3
topic for my complaint
this it the actual complaint box to write the completes to admin
Not Yet replyed
Type here to search
31°C
4:58 PM
1/31/2022

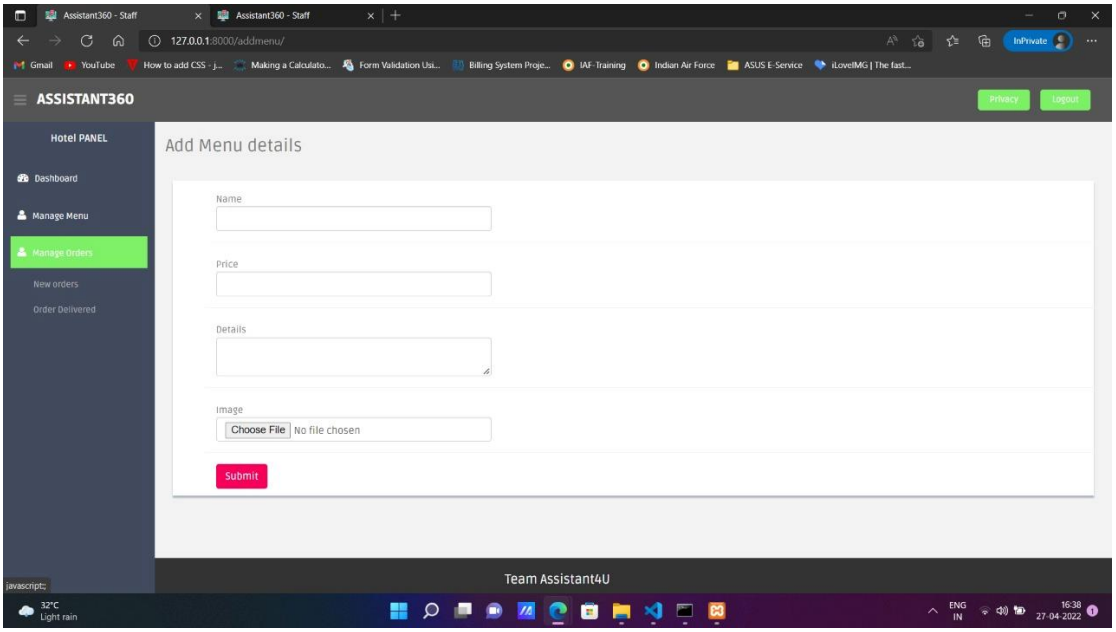

Assistant360 - Staff
New Tab
127.0.0.1:8000/complaints/
Incognito
ASSISTANT360
Privacy
Logout
STAFF PANEL
Send Complaint
Subject
topic for my complaint
Complaint
this it the actual complaint box to write the completes to admin
Submit
Dashboard
Job confirmations
Ongoing Jobs
Payment History
Complaint
Complaint History
Sl:No
Subject
Complaint
Reply
Type here to search
31°C
4:58 PM
1/31/2022

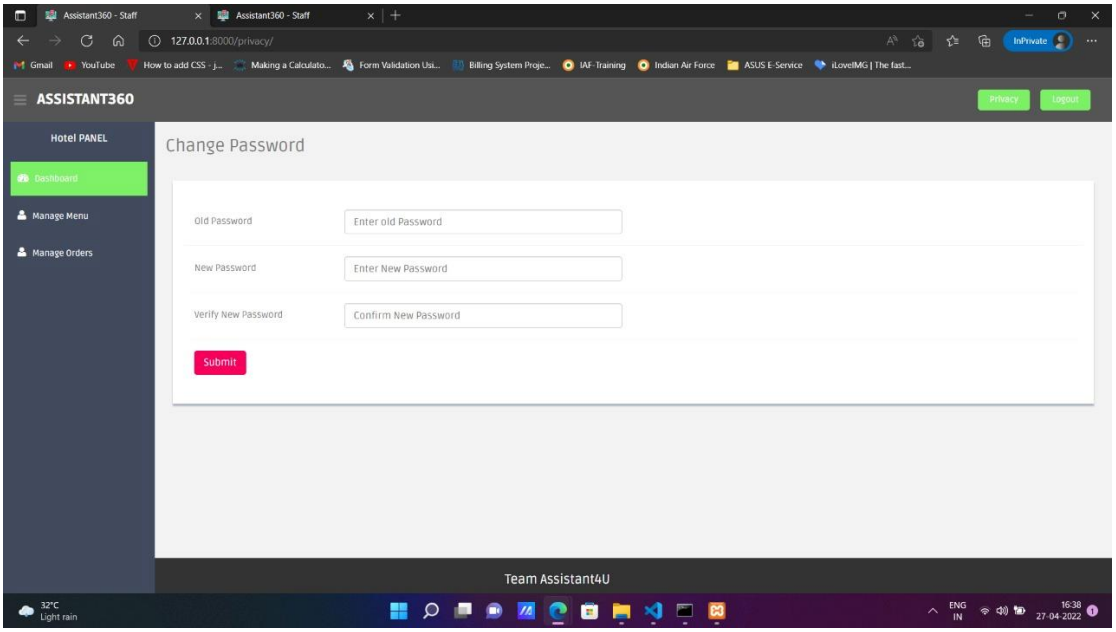

Assistant360 - Staff
New Tab
127.0.0.1:8000/ongoing/
Incognito
ASSISTANT360
Privacy
Logout
STAFF PANEL
Dashboard
Job confirmations
Ongoing Jobs
Payment History
Complaint
Ongoing jobs
Sl:No
User Name
Address
Phone
From Date
To date
Job description
Action
1
abu
abu
abu
2022-02-01
2022-02-02
clean my loon, rooms, wash cloths
Action
Team Assistant4U
Type here to search
31°C
4:56 PM
1/31/2022


Assistant 360 | User
Assistant 360 | User
127.0.0.1:8000/Feedback/
InPrivate
Gmail
YouTube
How to add CSS - j...
Making a Calculato...
Form Validation Usi...
Billing System Proje...
IAF-Training
Indian Air Force
ASUS E-Service
iLoveIMG | The fast...
TAXI PANEL
abhidharsh
Dashboard
New bookings
Payment recieved
Rejected bookings
Work History
Complaint
SEND COMPLAINT
Complaint
Submit
COMPLAINT HISTORY
Sl:No
Complaint
Reply
32°C
Partly sunny
ENG
IN
17:04
27-04-2022


Assistant 360 | User
Assistant 360 | User
127.0.0.1:8000/privacy/
InPrivate
Gmail
YouTube
How to add CSS - j...
Making a Calculato...
Form Validation Usi...
Billing System Proje...
IAF-Training
Indian Air Force
ASUS E-Service
iLoveIMG | The fast...
TAXI PANEL
abhidharsh
Dashboard
New bookings
Payment recieved
Rejected bookings
Work History
Complaint
Old Password
New Password
Verify New Password
Submit
32°C
Partly sunny
ENG
IN
17:03
27-04-2022


Assistant 360 | User
Assistant 360 | User
127.0.0.1:8000/userhome
InPrivate
Gmail
YouTube
How to add CSS - j...
Making a Calculato...
Form Validation Usi...
Billing System Proje...
IAF-Training
Indian Air Force
ASUS E-Service
iLoveIMG | The fast...
TAXI PANEL
abhidharsh
Dashboard
New bookings
Payment recieved
Rejected bookings
Work History
Complaint
Welcome to Taxi Pannel
32°C
Partly sunny
ENG
IN
17:03
27-04-2022


Assistant360 - Staff
127.0.0.1:8000/addmenu/
ASSISTANT360
Privacy
Logout
Hotel PANEL
Dashboard
Manage Menu
Manage Orders
New orders
Order Delivered
Add Menu details
Name
Price
Details
Image
Choose File
No file chosen
Submit
Team Assistant4U


Assistant360 - Staff
127.0.0.1:8000/privacy/
ASSISTANT360
Privacy
Logout
Hotel PANEL
Dashboard
Manage Menu
Manage Orders
Change Password
Old Password
Enter old Password
New Password
Enter New Password
Verify New Password
Confirm New Password
Submit
Team Assistant4U

4.3 DATA FLOW DIAGRAM

Data Flow Diagram (DFD) is an important tool used by system analyst. DFD provide an overview of what data a system would process, What transformation of data are done, what files are used and where the results flow. The graphical representation of the system makes it a good communication tool between the user and the analyst.

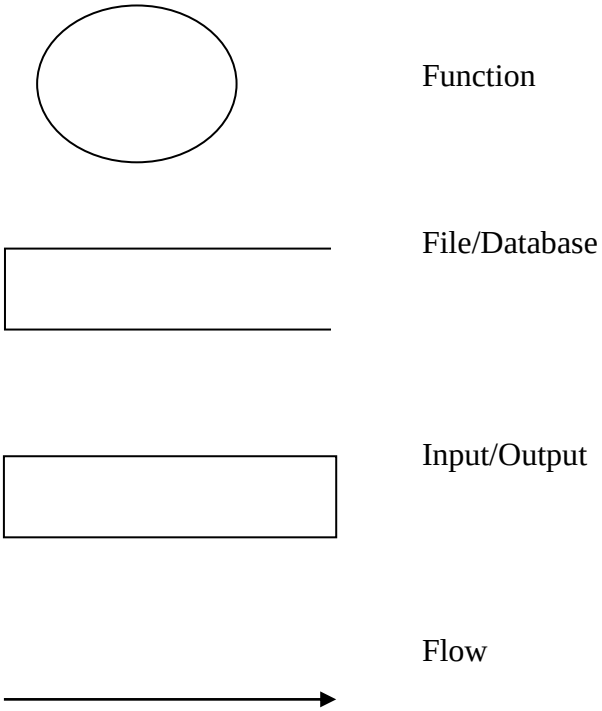
Analysis model help us to understand the relationship between different components in the design. Analysis model shows the user clearly how a system will function. This is the first technical representation of the system.

The analysis modeling must achieve three primary objectives

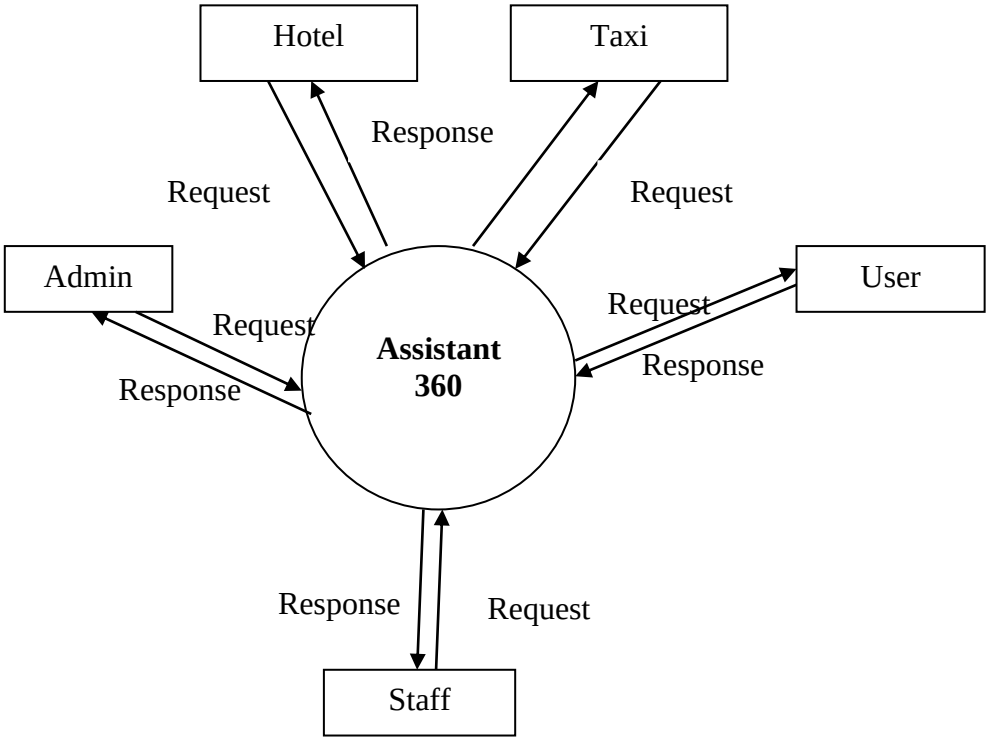
- To establish a basis for creation of software design
- To describe what the user requires
- To define set of requirements that can be validated once the software us build.

A data flow diagram is a graphical technique tat depicts information flow and transforms that are applied as data move from input to output. The DFD is used to represent increasing information flow and functional details. A level 0 DFD also called fundamental system model represents the entire software elements as single bubble with input and output indicated by incoming and outgoing arrow respectively.

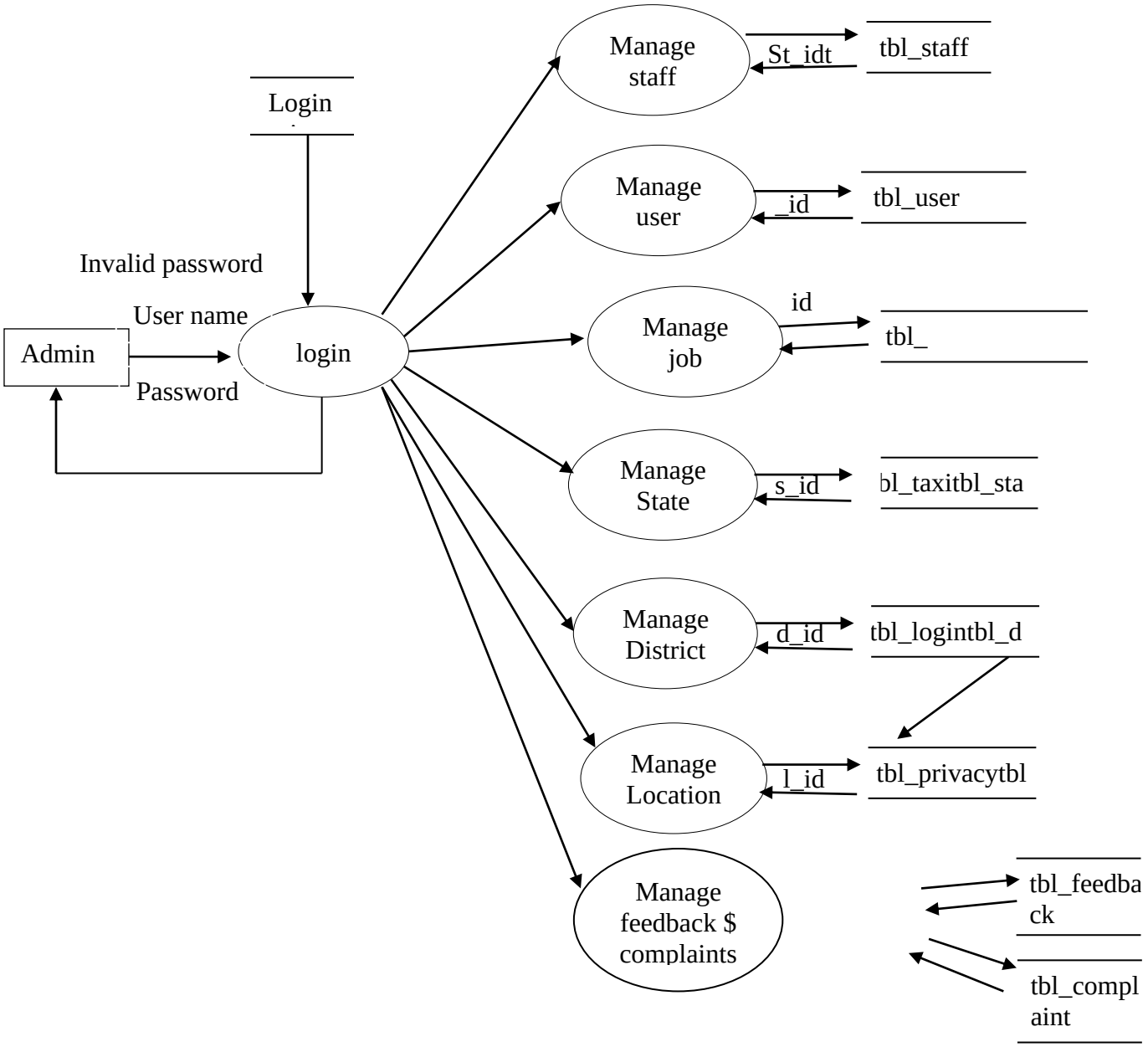
Components of Data Flow Diagram



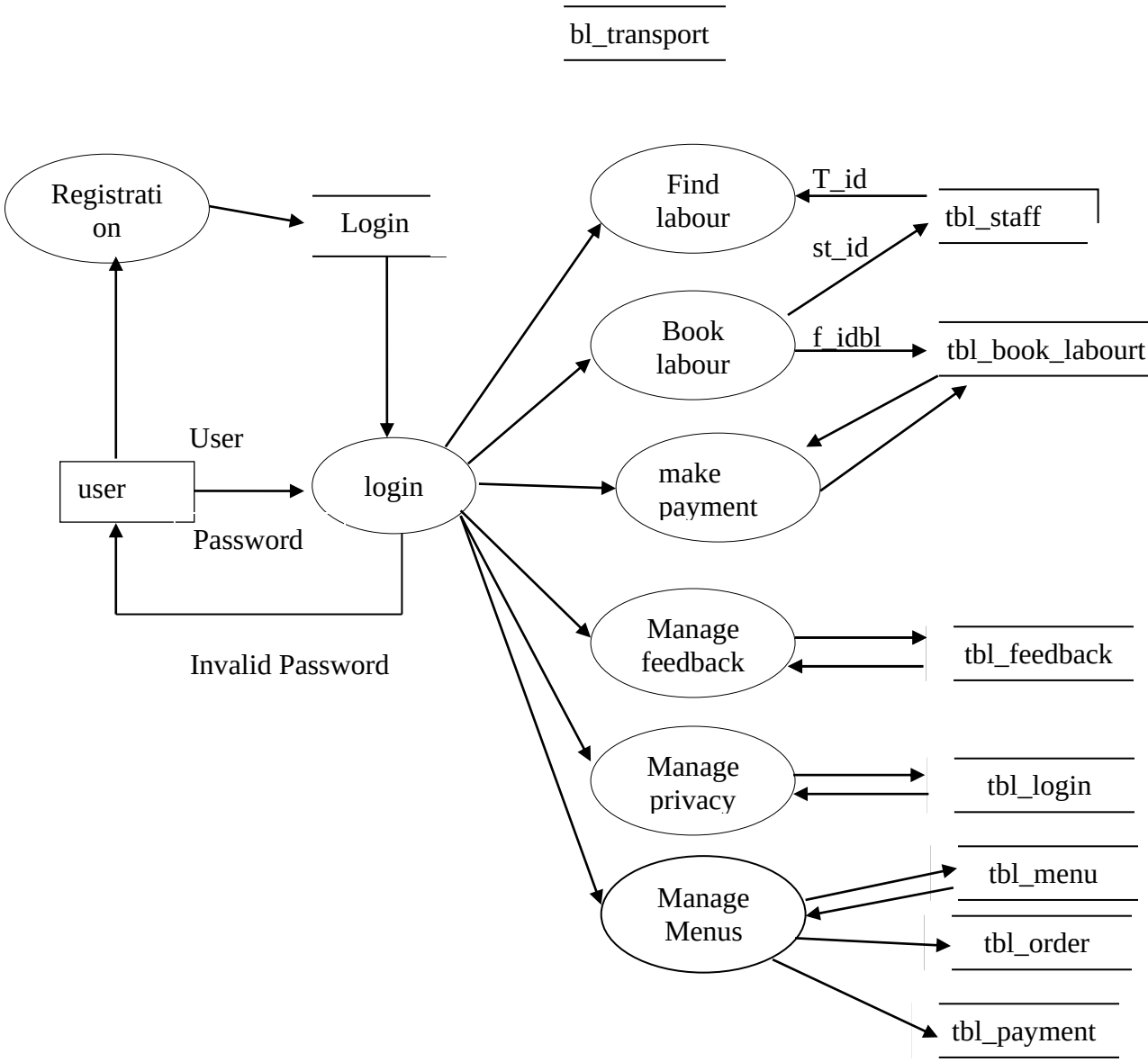
Level 0



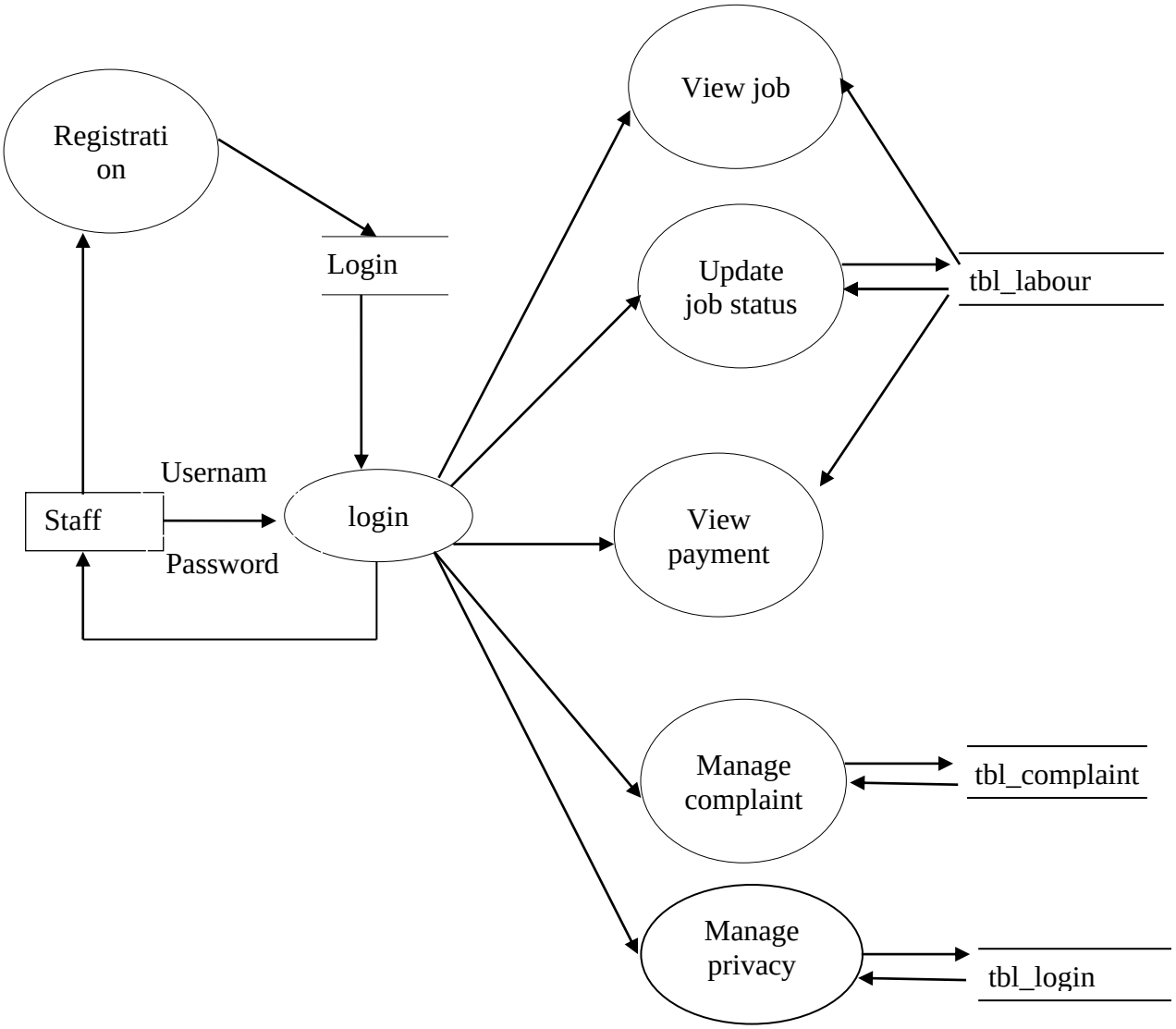
Level 1
Admin



User Level


bl_transport
Registrati
on
Login
Find
labour
T_id
tbl_staff
st_id
Book
labour
f_idbl
tbl_book_labourt
User
user
login
make
payment
Password
Invalid Password
Manage
feedback
tbl_feedback
Manage
privacy
tbl_login
tbl_menu
Manage
Menus
tbl_order
tbl_payment

Level 1


Registrati
on
Login
Staff
Usernam
Password
login
View job
Update
job status
View
payment
Manage
complaint
Manage
privacy
tbl_labour
tbl_complaint
tbl_login

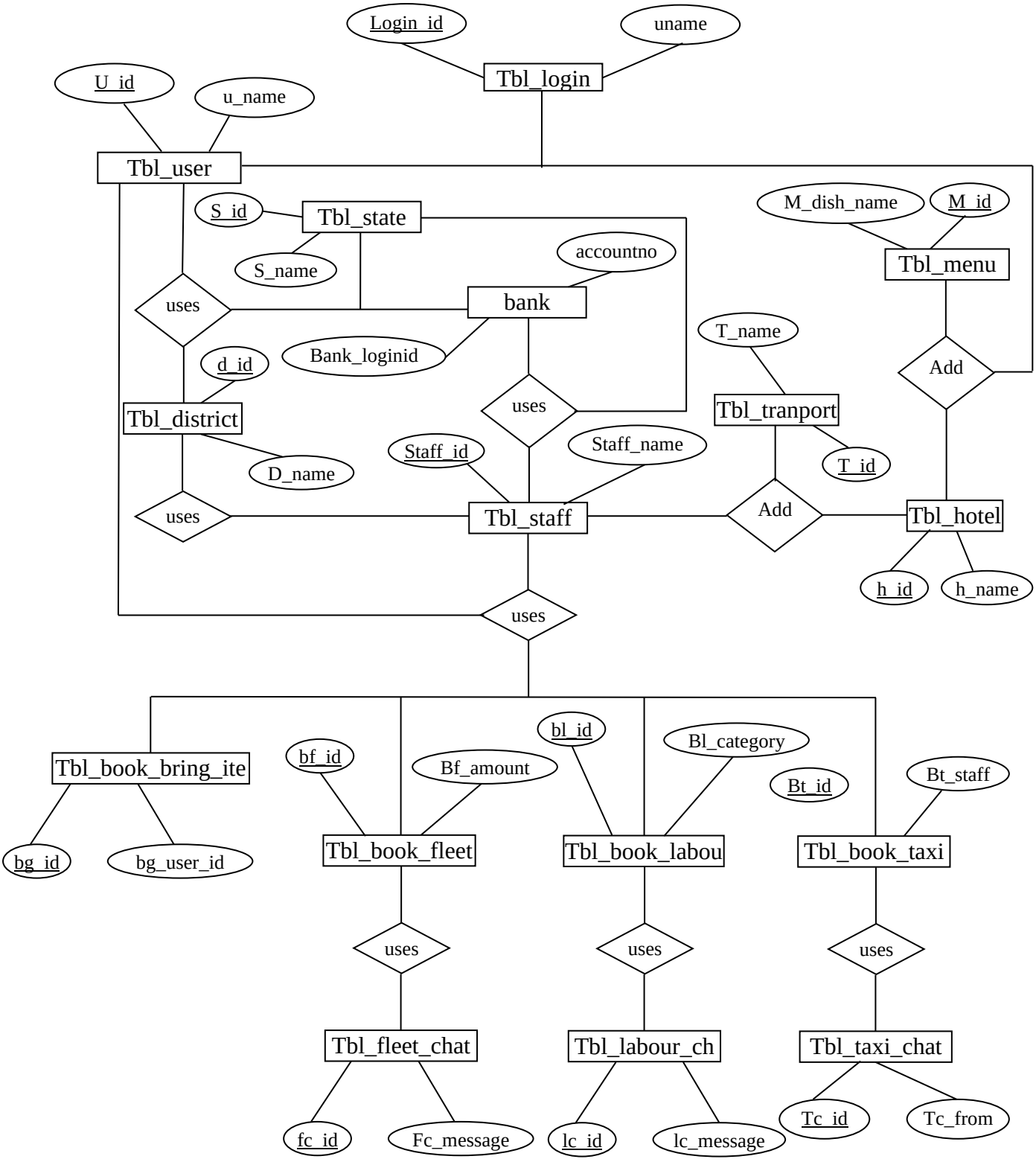
4.3 ER DIAGRAM

ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships.

ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.

At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique. The purpose of ER Diagram is to represent the entity framework infrastructure.

The following diagram describe ER diagram of Assistant 360:


Login_id
uname
Tbl_login
U_id
u_name
Tbl_user
S_id
Tbl_state
S_name
accountno
bank
uses
Bank_loginid
d_id
Tbl_district
D_name
uses
Staff_id
Staff_name
Tbl_staff
uses
M_dish_name
M_id
Tbl_menu
Add
T_name
Tbl_tranport
T_id
Add
Tbl_hotel
h_id
h_name
uses
Tbl_book_bring_ite
bg_id
bg_user_id
bf_id
Bf_amount
Tbl_book_fleet
bl_id
Bl_category
Bt_id
Tbl_book_labou
Bt_staff
Tbl_book_taxi
uses
uses
uses
Tbl_fleet_chat
Tbl_labour_ch
Tbl_taxi_chat
fc_id
Fc_message
lc_id
lc_message
Tc_id
Tc_from

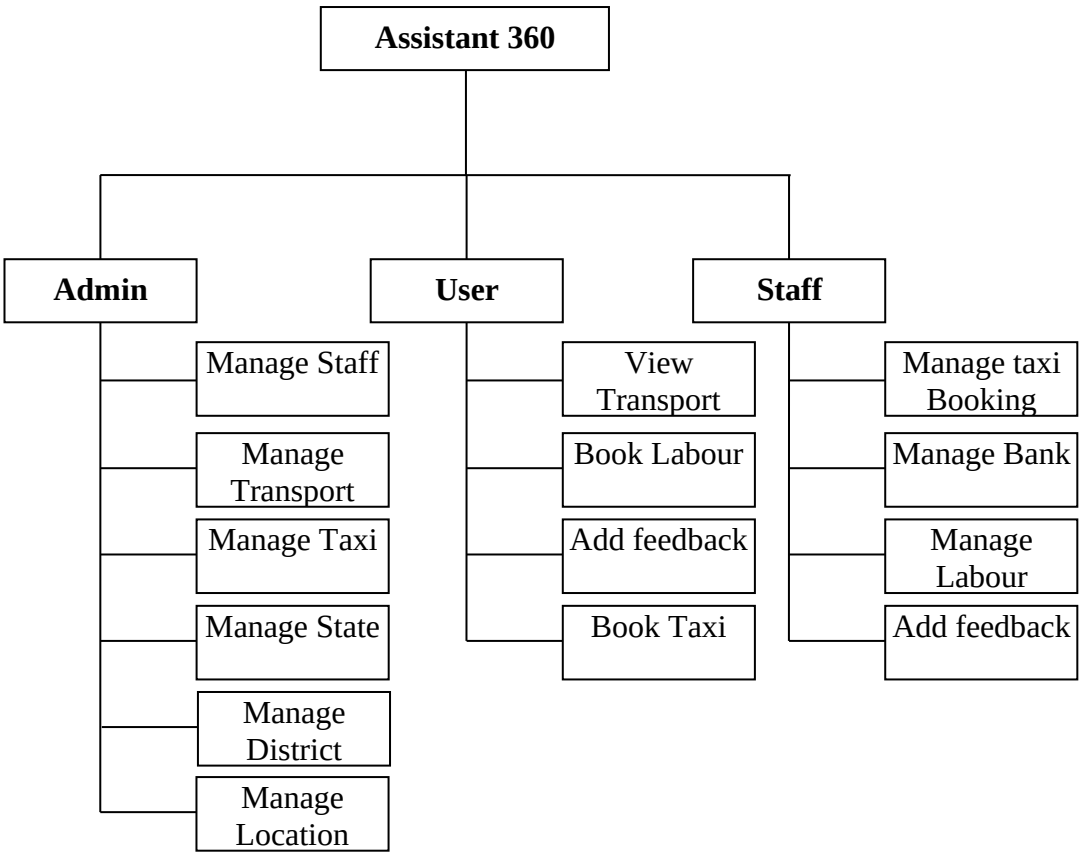
4.4 SOFTWARE DESIGN

4.4.1 ARCHITECTURAL DESIGN

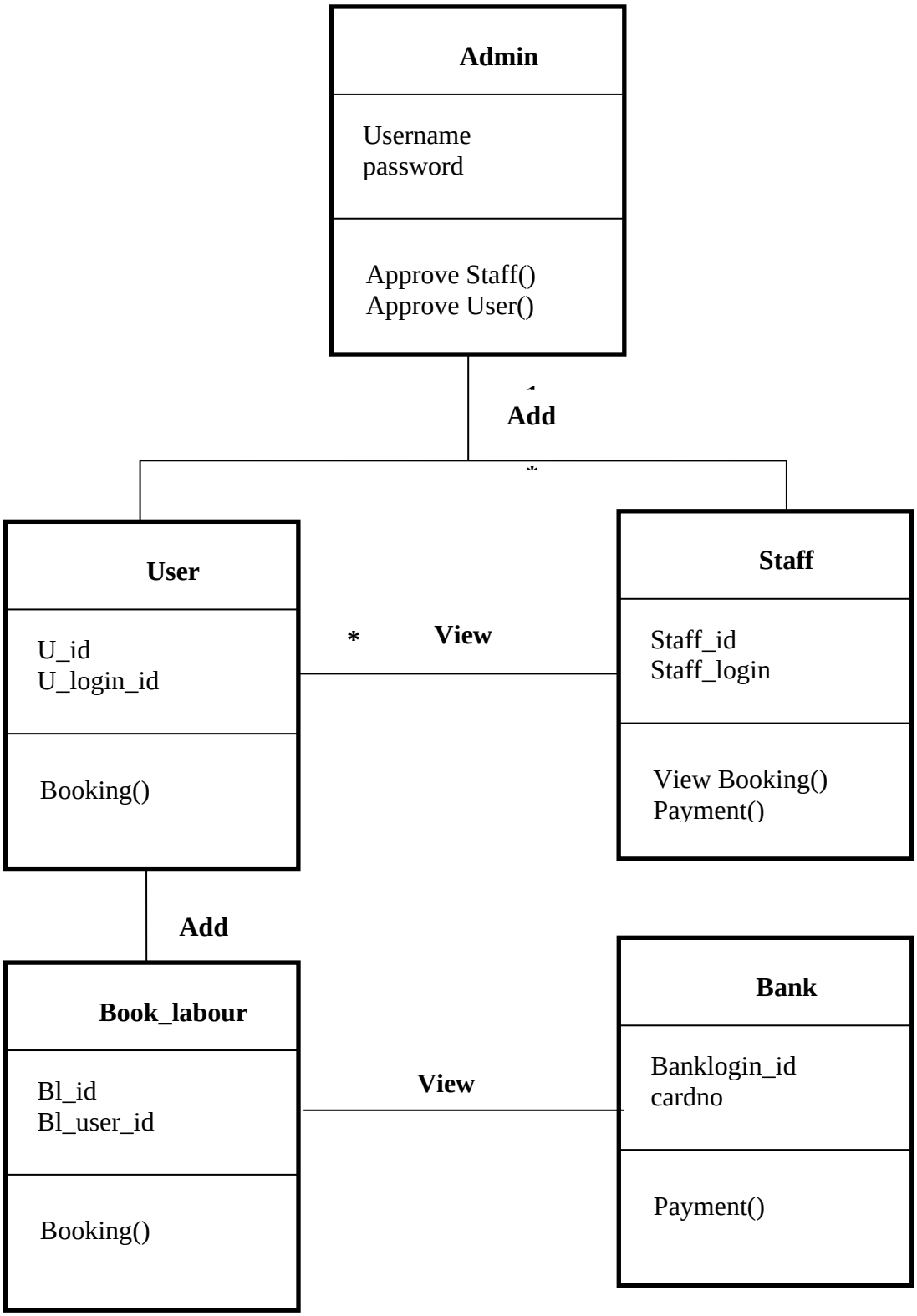
Structure Chart

Structure Chart represent hierarchical structure of modules. It breaks down the entire system into lowest functional modules, describe functions and sub-functions of each module of a system to a greater detail. Structure Chart partitions the system into black boxes (functionality of the system is known to the users but inner details are unknown). Inputs are given to the black boxes and appropriate outputs are generated.

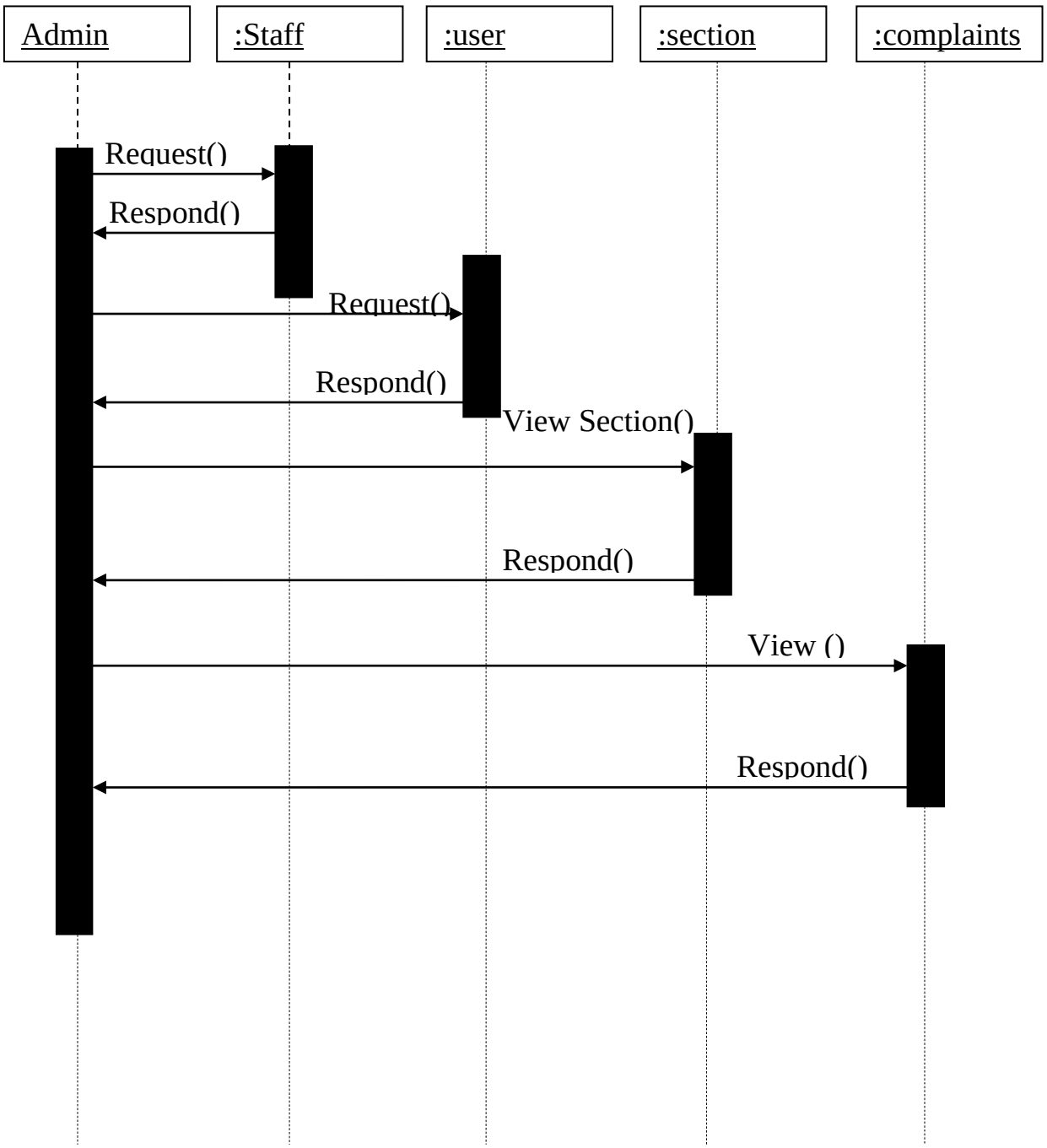
Modules at top level called modules at low level. Components are read from top to bottom and left to right. When a module calls another, it views the called module as black box, passing required parameters and receiving results.



Class Diagram



Sequence Diagram



4.4.2 DATABASE DESIGN

Database design forms an important part of every project. The management of data involves both the definition of structure for the storage of information and provision of mechanisms for manipulation of information. The database system must provide safety for the information stored; despite system crashes or attempts of unauthorized access the database used in this project is MYSQL.

Database constructed using relation is termed as relational database. The row of a table is referred as tuple .Each column in a table has column names. Column or a field is called an attribute. The number of tuples in a relation is called cardinality and the number of attribute is called degree. Every table must have some column or combination of columns that uniquely identify each row in a table .It is called primary key. A column in one table with a value that matches the primary key in some other table is called a foreign key. Together a primary key and foreign key create a parent-child relationship between tables.

The table has the following properties:

- Each entry in a table represents one data item.
- They are column homogeneous i.e, in any column all items are of same kind.
- Each column is assigned a distinct name.
- All rows are distinct.

The database design is made up of three levels

- Conceptual level(High level)
- Physical level(Low level)
- View level(Representation level)

Conceptual Level

Conceptual level describes the essential features of the system data just like a DFD for system. It uses symbols and is called Entity-Relationship analysis. An entity is a conceptual representation of an object. Relationship between entities used to make the database structure.

1. A one-to-one relationship is an association between two entities.

2. A one-to-many relationship describes an entity that may have two or more entities related to it.

3. A many-to-many relationship describes entities that may have many relationships in both directions.

Physical Level

In this level the data is stored physically. That is an internal schema describes the physical storage structure of the database.

View Level

This level is used to describe how the user views the records or objects in the database.

4.5 DATA DICTIONARY

TABLES

bank

Field Name	Data Type	Constraints	Description
Accountno	bigint(11)	Not Null	Account Number
cvv	varchar(3)	Not Null	Curriculum vitae
exmonth	varchar(10)	Not Null	Exam month
exyear	year(4)	Not Null	Exam year
Cardtype	varchar(10)	Not Null	Card type
Bankname	varchar(25)	Not Null	Bank name
balance	varchar(20)	Not Null	balance
bank_loginid	int(10)	Primary Key	Bank login id
cardno	varchar(16)	Not Null	Card Number

tbl_district

Field Name	Data Type	Constraints	Description
d_id	int(11)	Primary Key	***District id***
d_state_id	int(11)	Foreign Key	District state id
d_name	varchar(30)	Not Null	District name

tbl_book_labour

Field Name	Data Type	Constraints	Description
bl_id	int(11)	Primary Key	***Book Labour id***
bl_user_id	int(11)	Foreign Key	Book Labour user id
bl_staff	int(11)	Not Null	Book Labour staff
bl_from_date	date	Not Null	Book Labour from date
bl_to_date	date	Not Null	Book Labour to date

bl_category	varchar(30)	Not Null	Book Labour category
bl_amt	int(11)	Not Null	Book Labour amount
bl_reject_reason	text	Not Null	Book Labour reject reason
bl_status	int(11)	Not Null	Book Labour status

tbl_book_taxi

Field Name	Data Type	Constraints	Description
bt_id	int(11)	Primary Key	***Book Taxi id***
bt_user_id	int(11)	Foreign Key	Book Taxi user id
bt_staff	int(11)	Not Null	Book Taxi staff
bt_from_date	date	Not Null	Book Taxi from date
bt_to_date	date	Not Null	Book Taxi to date
bt_category	varchar(30)	Not Null	Book Taxi category
bt_reject_reason	text	Not Null	Book Taxi reject reason
bt_status	int(11)	Not Null	Book Taxi status

tbl_feedback

Field Name	Data Type	Constraints	Description
f_id	int(11)	Primary Key	***Feedback id***
f_user_id	int(11)	Foreign Key	Feedback user id
f_feedback	text	Not Null	Feedback Details

tbl_labour_chat

Field Name	Data Type	Constraints	Description
lc_id	int(11)	Primary Key	***Labour Chat id***
lc_labour_id	int(11)	Foreign Key	Labour Chat labour id
lc_from	int(11)	Not Null	Labour Chat from
lc_to	int(11)	Not Null	Labour Chat to
lc_message	text	Not Null	Labour Chat message
lc_time	varchar(60)	Not Null	Labour Chat time

tbl_location

Field Name	Data Type	Constraints	Description
l_id	int(11)	Primary Key	***Location id***
l_state_id	int(11)	Foreign Key	Location state id
l_district_id	int(11)	Foreign Key	Location district id
l_location_name	varchar(40)	Not Null	Location name

tbl_login

Field Name	Data Type	Constraints	Description
login_id	int(20)	Primary Key	***Login id***
uname	varchar(20)	Not Null	User Name

pwd	text	Not Null	pwd
role	int(20)	Not Null	role
role_des	varchar(20)	Not Null	role_des

tbl_state

Field Name	Data Type	Constraints	Description
s_id	int(11)	Primary Key	***State id***
s_name	varchar(30)	Not Null	State Name

tbl_stop

Field Name	Data Type	Constraints	Description
st_id	int(11)	Primary Key	***Stop id***
s_transport_id	int(11)	Foreign Key	Stop transport id
s_name	varchar(20)	Not Null	Stop name
s_arrival_time	timestamp	Not Null	Stop arrival time
s_departed_time	timestamp	Not Null	Stop departed time

tbl_transport

Field Name	Data Type	Constraints	Description
t_id	int(11)	Primary Key	***Transport id***
t_type	varchar(20)	Not Null	Transport type
t_name	varchar(25)	Not Null	Transport name
t_details	text	Not Null	Transport details
t_departed_time	timestamp	Not Null	Transport departed time
t_departed_state	int(11)	Not Null	Transport departed state
t_departed_district	int(11)	Not Null	Transport departed district
t_departed_location	int(11)	Not Null	Transport departed location
t_arrival_time	timestamp	Not Null	Transport arrival time
t_arrival_state	int(11)	Not Null	Transport arrival state
t_arrival_district	int(11)	Not Null	Transport arrival district
t_arrival_location	int(11)	Not Null	Transport arrival location

tbl_taxi_chat

Field Name	Data Type	Constraints	Description
tc_id	int(11)	Primary Key	***Taxi Chat id***
tc_taxi_id	int(11)	Foreign Key	Taxi Chat taxi id
tc_from	int(11)	Not Null	Taxi Chat from
tc_to	int(11)	Not Null	Taxi Chat to
tc_chat_message	text	Not Null	Taxi Chat message
tc_time	varchar(60)	Not Null	Taxi Chat time

tbl_user

Field Name	Data Type	Constraints	Description
u_id	int(11)	Primary Key	***User id***
u_login_id	int(11)	Foreign Key	User login id
u_name	varchar(30)	Not Null	User name
u_address	text	Not Null	User address
u_age	int(11)	Not Null	User age
u_phone	varchar(20)	Not Null	User phone
u_adhar	varchar(20)	Not Null	User aadhar
u_email_id	varchar(30)	Not Null	User email id
u_state	int(11)	Not Null	User state
u_district	int(11)	Not Null	User district
u_location	int(11)	Not Null	User location
u_photo	varchar(40)	Not Null	User photo
u_status	int(11)	Not Null	User status

tbl_vehicle

Field Name	Data Type	Constraints	Description
v_id	int(11)	Primary Key	***Vehicle id***
v_staff_id	int(11)	Foreign Key	Vehicle staff id
v_model	varchar(40)	Not Null	Vehicle model
v_number	varchar(20)	Not Null	Vehicle number
v_insurance	text	Not Null	Vehicle insurance
v_rate	int(11)	Not Null	Vehicle rate
v_vehicle	text	Not Null	Vehicle Details

tb_staff

Field Name	Data Type	Constraints	Description
staff_id	int(11)	Primary Key	***Staff id***
staff_login	int(11)	Foreign Key	Staff login
staff_name	varchar(30)	Not Null	Staff name
staff_address	text	Not Null	Staff address
staff_age	int(11)	Not Null	Staff age
staff_adhar_no	varchar(20)	Not Null	Staff adhar no
staff_phone_no	varchar(20)	Not Null	Staff phone no
staff_email	varchar(30)	Not Null	Staff email
staff_state	int(11)	Not Null	Staff state
staff_district	int(11)	Not Null	Staff district
staff_location	int(11)	Not Null	Staff location
staff_licence_no	varchar(20)	Not Null	Staff licence no
staff_category	varchar(30)	Not Null	Staff category
staff_basic_salary	varchar(20)	Not Null	Staff basic salary
staff_photo	varchar(30)	Not Null	Staff photo
s_status	int(11)	Not Null	Staff status

App_orderitem

Field Name	Data Type	Constraints	Description
order_itemid	int(11)	Primary Key	Item id
quantity	varchar(100)	Not Null	quantity
menu_id	int(11)	Foreign Key	Manu id
order_id	int(11)	Foreign Key	Order id

Tbl_menu

Field Name	Data Type	Constraints	Description
menuid	int(11)	Primary Key	Menu ID
name	varchar(100)	Not Null	Name
price	varchar(100)	Not Null	Price
details	varchar(100)	Not Null	Details
image	varchar(100)	Not Null	Image

App_orders

Field Name	Data Type	Constraints	Description
order_id	int(11)	Primary Key	Order id
order_status	varchar(100)	Not Null	status
menu_id	int(11)	Foreign Key	Manu id
user_id	int(11)	Foreign Key	User id

5. SYSTEM IMPLEMENTATION

5.1. MODULE DESCRIPTION

ADMIN

Admin is the overall authority of the system. His functions are

- View users and staffs registered in website
- View request from users and assign corresponding staffs/workers.
- View users and staffs complaint
- Logistics Section: View request and assign staffs, view previous requests.
- Manage labour.
- Add required staff posts.

USER

The functions of users are..

- View professionals and Make request for hiring professionals.
- Make complaints
- Book labor.
- Manage feedback
- Update profile

STAFF/WORKERS

- View request from users assigned by admin.
- Add real-time status.
- Payment

- Add Complaints
- Contact admin
- Update profile
- Upload job status

Hotel

The functions of users are..

- Manage menu
- Manage delivery
- Manage privacy
- Update profile

Taxi

The functions of users are..

- View booking
- Update booking
- View completed work
- Manage privacy
- Update profile

5.2. IMPLEMENTATION DETAILS

The implementation is the final and important phase. It involves user training, system testing in order to ensure successful running of the proposed system. Once the system design phase is over, the next stage is to implement and monitors the operation of the system to ensure that it continues the work effectively and efficiently.

The three main phases in implementation take place in series. These are the initial installation, the test of the system as a whole and evaluation maintenance and

control of the system. The implementation plan and action to implement should be bound closely together. The implementation plan is a function of line management at least as far as key decision or alternative plans are concerned.

The implementation plan was to convert the existing clerical files to the computer. The implementation plan listed all sub tasks so that individuals in the organization may be assigned specific responsibilities.

The installation of the new system that is bound to replace the current one may require a major revision of computer facilities as well as completely new after space. Space planning took into account the space occupied by the people, space by equipment and the movement of people and equipment in the working investment. After conduction the initial testing the system is loaded on the client office's computer. Some of the user employees in this case are selected. These users are trained first and they run the system. A detailed documentation is prepared to this set of employees. There may be slight modifications to meet the organization.

After all modifications specified by the users in the documentation are made, the computer system is run along with manual system. Even though this kind of parallel run make extra burden to the employees and management, the system is run in parallel for the sake of checking reliability and efficiency. After this document, which compares the result of the manual system with those of the computerized is prepared. If there is any modifications are made as needed.

A procedure is developed for delivering instructions and forms to supervisors for coordination and integrating the proposal with other parts of the organization, and for working out of problems with people involved. This procedure also helped for evaluation of hardware and software. A program was developed to emphasis the nature and goals of the new system on the management and the support personnel and train operation personnel in their new tasks.

In the case of management many of whom participated in the development of the system short seminars were given. Particular attention was paid to the training of end users. The training sessions were aimed at giving the user staff the specific skills required in their new jobs. They were given practical training to have a thorough understanding of what the new system is like and how it behaves.

Education involved creating the right atmosphere and motivation of user staff. It explained the need for changes and helped to overcome the resentment caused by the feeling that computers took away the responsibility from individual departments.

Various measures have been taken by department officials in order to find suitable solutions by the following issues:

- About the skill to be acquired.
- Reduction of man power in department
- About the new form having all required option.

Installation Procedure

To install the system, the primary need is web based environments without which the system will not have a proper utilization. To install the system it is must to setup a centralized server which can hold social networking website including the user's information database. The database is accessed through web pages using browser at the client end. In order to have the server setup for the device information system, the following components are needed at the server end.

1. Python

2. MySQL

3. A preferable operating system like Windows 10

Copy all the required files to the web server root folder and create the database.

In the field of computer software, the term software build refers either to the process of converting source code file in to stand alone software artifacts that can be run on a computer, or the result of doing so.

6. TESTING

6.1. TEST CASES

Test cases are the key to the process because they identify and communicate the conditions that will be implemented in test and are necessary to verify successful and acceptable implementation of the product requirement. They are all about making sure that the product fulfils the requirements of the system.

Sl.No	Test Case	Test Procedure	Precondition	Expected Result	Passed/ Failed (Yes/No)
1	Login Page	To check whether the control from the login screen goes to the main menu.	Enter a valid user name and password on the login screen	The control should go to the home page	yes
2	User Registration	To check whether the control goes to the user registration screen when the user select the registration from homepage	Select registration from the home page	The control should go to registration page	Yes
3	Booking	There is a option in the users home screen for booking staff. Here the user can book staff .	Select booking page from the users homepage	The control should go to booking page	Yes
4	Make payments	After booking a particular staff the user is responsible for making payment .	Select payment option from user page	The control should go to payment page	yes

5	complaints	Here the users can register their complaints about the problems they where noticed in the system	Select complaint option from the user home page	The control should go to complaint page	Yes
6	Staff registration	To check whether the control goes to the staff registration screen when the staff select the registration from homepage	Select the staff registration option from the home page	The control should go to staff registration page	Yes
7	View bookings and view cancellations	Here the admin can view the booking status and cancellation status of staff	Select view booking and view cancellation pages in the admin home page	The control should go to the corresponding pages	Yes
9	Approval of Staff and user	Admin can only have the permission to approve the staff and user	Select view user and staff details from the homepage	The control should go to the list of staff and users	Yes
10	View staff details	User can check the currently booked staff	Select the staff	The control should go to the list of Staff	yes

6.2. UNIT, INTEGRATION, USER ACCEPTANCE, OUTPUT, VALIDATION

For any software that is newly developed, first and foremost preference is given to the testing of the system. It is developer‘s last chance to detect and correct the errors. That may occur possibly in the software. The programmers will generate a set of test data, which will give the maximum possibility of finding all most all types of errors that can occur in the system.

Unit Testing

The primary goal of unit testing is to take the smallest piece of testable software in the application, isolate it from the remainder of the code, and determine whether it behaves exactly as you expect. Each unit is tested separately before integrating them into modules to test the interfaces between the modules. Unit testing has proven its value in that a large percentage of defects are identified during its use.

The most common approach to unit testing requires drivers and stubs to be written. The driver simulates a calling unit and the stub simulates a called unit. The investment of developer time in this activity sometimes results in demoting unit testing to a lower level of priority and that is almost always a mistake. Even though the drivers and stubs cost time and money, unit testing provides some undeniable advantages. It allows for automation of the testing process, reduces difficulties of discovering errors contained in more complex pieces of the application, and test coverage is often enhanced because attention is given to each unit.

Integration Testing

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

The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items. These–design items‖ i .e. assemblages(or groups of units) are exercised through their interfaces using Black box testing, success and error

cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test that all components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individua lmodules, i.e .unit testing. The over all idea is a‒building block‖ approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages.

Validation Testing

Data validation is the process of testing the accuracy of data. A set of rule we can apply to a control to specify the type and range of data that can enter. It can be used to display error alert when users enter incorrect values in to a form. Now performing validation testing in system Centralized Social Welfare by undergoing validation for each tools and the validation succeeded when the software function in a manner that can be reasonably accepted, by the user.

User Acceptance Testing

User acceptance of a system is a key factor for the success of any system. The system under consideration is tested for the user acceptance by constantly, keeping in touch with the prospective system user at the time of developing and making changes whenever required.

7. CONCLUSION

The project was successfully completed within the time span allotted every effort has been made to present the system in more user friendly manner. The new system has overcome most of the limitations of the existing system and works according to the design specification given. The developed systems dispense the problem and meet the needs of by providing reliable and comprehensive information. All the requirements projected by the user have been met by the system. The newly developed system consumes less processing time and all the details are updated and processed immediately. Since the screen provides online help messages and is very user-friendly, any user will get familiarized with its usage. Modules are designed to be highly flexible so that any failure requirements can be easily added to the modules without facing many problems.

7.1 FUTURE ENHANCEMENT

Enhancement means adding, modifying or developing the code to support the changes in the specification. It is the process of adding new capabilities such as report, new interface without other systems and new features such as better screen or report layout. Every module in the system is being developed carefully such that the future enhancements do not affect the basic performance of the system. In future we can add any links or services to the System very easily. Moreover, due to limited time allotted for the project, there are features, which I couldn't implement. Thus the system offers the scope of future enhancement. As this software is reliable to use, any modification in accordance with the necessity of the user can be done for the future use. Any additional feature can be implemented very easily. So what we call this software also a user friendly.

8. BIBILIOGRAPHY

BOOKS:

[1] “Beginning Python and MySQL From Novice to Professional” by W Jason Gilmore

[2] “Build Your Own Database Driven Web Site Using Python& MySQL” by Kevin Yank

[3] “Eloquent JavaScript: A Modern Introduction to Programming” by Marijn Haverbeke

[4] “HTML & CSS”: The Complete Reference, Fifth Edition

[5] “JavaScript: The Definitive Guide” by David Flanagan

[6] “JavaScript: The Good Parts” by Douglas Crockford

[7] “Learning Python5” by David Sklar

[8] “Modern Python: New Features and Good Practices” by Josh Lockhart

[9] “Python for the Web: Visual QuickStart Guide” by Larry Ullman

WEB SITES

- Python.net/manual/en/tutorial .Python
- https://www.w3schools.com/Python/
- https://www.awwwards.com/websites/Python/
- https://en.wikipedia.org/wiki/Python

9. APPENDIX

SOURCE CODE

```
from django.db import models


# Create your models here.


class login(models.Model):
logid = models.AutoField(primary_key=True)
username = models.CharField("username",max_length=100)
password =  models.CharField("password",max_length=100)
role=models.CharField('role',max_length=10)
    #logid,username,password,role


class state(models.Model):
    state_id = models.AutoField(primary_key=True)
state=models.CharField("state",max_length=100)


class district(models.Model):
    district_id=models.AutoField(primary_key=True)
district=models.CharField("district",max_length=100)
state=models.ForeignKey(state, on_delete=models.CASCADE, null=True)


class locations(models.Model):
    location_id=models.AutoField(primary_key=True)
location=models.CharField("location",max_length=100)
distict=models.ForeignKey(district, on_delete=models.CASCADE, null=True)
```

```
    @property
def getalldist(self):
data=district.objects.filter(state=self.distict.state).all()
return data


class staff(models.Model):
    staff_id= models.AutoField(primary_key=True)
login=models.ForeignKey(login,on_delete=models.CASCADE,null=True)
name=models.CharField("staffname",max_length=100)
address=models.CharField("address",max_length=500)
    aadhaar_no=models.CharField("aadhaar",max_length=100)
    phone_no=models.CharField("phone_no",max_length=100)
email=models.CharField("email",max_length=100)
state=models.ForeignKey(state,on_delete=models.CASCADE, null=True)
district=models.ForeignKey(district, on_delete=models.CASCADE, null=True)
location=models.ForeignKey(locations, on_delete=models.CASCADE, null=True)
category=models.CharField("category",max_length=100)

exp=models.CharField("experience",max_length=100)
    basic_salary=models.CharField("basic_salary",max_length=100)
photo=models.FileField("photo:",max_length=100,upload_to="images/")
status=models.CharField("status:",max_length=100)

#staff_id,login,name,address,aadhaar_no,phone_no,email,state,district,location,category,license,exp,b
asic_salary,photo,status
class user(models.Model):
    user_id=models.AutoField(primary_key=True)
login=models.ForeignKey(login,on_delete=models.CASCADE,null=True)
```

```
username=models.CharField("username",max_length=100)
useraddress=models.CharField("address",max_length=500)
phoneno=models.CharField("Phone_no",max_length=100)
useremail=models.CharField("email",max_length=100)
state=models.ForeignKey(state,on_delete=models.CASCADE, null=True)
district=models.ForeignKey(district, on_delete=models.CASCADE, null=True)
location=models.ForeignKey(locations, on_delete=models.CASCADE, null=True)
    Photo=models.FileField("photo:",max_length=100,upload_to="images/")
status=models.CharField("status:",max_length=100)
    #user_id,login,username,useraddress,phoneno,useremail,state,district,location,Photo,status


class labour(models.Model):
    labour_id=models.AutoField(primary_key=True)
userid=models.ForeignKey(user,on_delete=models.CASCADE,null=True)
staff=models.ForeignKey(staff,on_delete=models.CASCADE,null=True)
    from_date=models.CharField("from date",max_length=100)
    to_date=models.CharField("to date",max_length=100)
category=models.CharField("category",max_length=100)
desc=models.CharField("description",max_length=500)
amount=models.CharField("amount",max_length=100)
reject=models.CharField("reject",max_length=100)
status=models.CharField("status",max_length=100)
    #userid,staff,from_date,to_date,category,desc,amount,reject,status
class feedback(models.Model):
    feedback_id=models.AutoField(primary_key=True)
userid=models.ForeignKey(user,on_delete=models.CASCADE,null=True)
feedback=models.CharField("feedback",max_length=500)
```

```
reply=models.CharField("reply",max_length=500)
    #userid,feedback,reply
class complaint(models.Model):
    complaint_id=models.AutoField(primary_key=True)
staff=models.ForeignKey(staff,on_delete=models.CASCADE,null=True)
sub=models.CharField("sublect",max_length=200)
msg=models.CharField("message",max_length=500)
reply=models.CharField("reply",max_length=500)
    #staff,sub,msg,reply


class bank(models.Model):
     bank_id=models.AutoField(primary_key=True)
holder=models.CharField("holder",max_length=100)
card=models.CharField("card",max_length=100)
cvv=models.CharField("cvv",max_length=100)
exp=models.CharField("exp",max_length=100)
bal=models.CharField("bal",max_length=100)
#bank_id,holder,card,cvv,exp,bal


from django.shortcuts import render,HttpResponse,redirect
from .models import login as log,state as st,district as dt,locations as loc
from .models import staff as stf,user as usr,feedback as fd, complaint as cm,labour as lb
from .models import bank as bnk
# Create your views here.

def index(request):
role=request.session.get("role")
if role == "admin":
return redirect("/adminhome")
```

```
elif role == "staff":
return redirect("/staffhome")
elif role == "user":
return redirect("/userhome")

datast=st.objects.all()
return render(request,"index.html",{"datast":datast})
def admin(request):
role=request.session.get("role")
if role == "admin":
return redirect("/adminhome")
elif role == "staff":
return redirect("/staffhome")
elif role == "user":
return redirect("/userhome")
if request.POST:
user = request.POST["username"]
password = request.POST["password"]

datac = log.objects.filter(username=user, password=password,role="admin").count()
if datac==1:
data=log.objects.get(username=user, password=password,role="admin")
request.session['username'] = data.username
request.session['role'] = data.role
request.session['id'] = data.logid
response = redirect('/adminhome')
return response
else:
return render(request,"adminlog.html",{"msg":"invalid username or password"})
else:
return render(request,"adminlog.html",{"msg":""})

def adminhome(request):
role=request.session.get("role")
```

```
if role != "admin":
return redirect("/index")

return render(request,"adminhome.html")
def privacy(request):
    msg=""
if request.POST:
        t1=request.POST["t1"]
        t2=request.POST["t2"]
id=request.session['id']

log.objects.filter(logid=id).update(username=t1,password=t2)



returnpage="adminhead.html"
    pg="Privacy.html"
if(request.session.get('role', ' ')=="staff"):
returnpage="staffhead.html"
elif(request.session.get('role', ' ')=="user"):
returnpage="userhead.html"
        pg="Privacy1.html"
return render(request,pg,{"role":returnpage,"msg":msg})
def getDistrict(request):
id=request.GET["id"]
datast=st.objects.get(state_id=id)
datadt=dt.objects.filter(state=datast).all()
res="<option value="">-select-</option>"
for d in datadt:
        res+="<option value="'+str(d.district_id)+'">"+d.district+"</option>"
return HttpResponse(res)
def get_location(request):
id=request.GET["id"]
datadt=dt.objects.get(district_id=id)
datalt=loc.objects.filter(distict=datadt).all()
```

```
res="<option value="">-select-</option>"
for d in datalt:
    res+="<option value="'+str(d.location_id)+'">"+d.location+"</option>"
return HttpResponse(res)
def Logout(request):
try:
del request.session['id']
del request.session['role']
del request.session['username']
response = redirect("/index?id=logout")
return response
except:
response = redirect("/index?id=logout")
return response
def stafflogin(request):
if request.POST:
user = request.POST["username"]
password = request.POST["password"]
try:
datac = log.objects.filter(username=user, password=password).count()
if datac==1:
data=log.objects.get(username=user, password=password)
if data.role=="staff":
request.session['username'] = data.username
request.session['role'] = data.role
request.session['id'] = data.logid
response = redirect('/staffhome')
return response
elif data.role=="user":
request.session['username'] = data.username
request.session['role'] = data.role
request.session['id'] = data.logid
response = redirect('/userhome')
return response
```

```
else :
response = redirect('/index?msg=invalid access')
return response


else:
response = redirect('/index?msg=invalid username or password')
return response

except:
response = redirect('/index?msg=something went wrong')
return response
else:
response = redirect('/index')
return response
def staffreg(request):
name=request.POST["name"]
addr=request.POST["addr"]
aadhar=request.POST["aadhar"]
category=request.POST["category"]
    Experience=request.POST["Experience"]
salary=request.POST["salary"]
state=request.POST["state"]
district=request.POST["district"]
location=request.POST["location"]
phone=request.POST["phone"]
mail=request.POST["mail"]
files=request.FILES["files"]
username=request.POST["username"]
password=request.POST["password"]
datast=st.objects.get(state_id=state)
datadt=dt.objects.get(district_id=district)
datalc=loc.objects.get(location_id=location)
log.objects.create(username=username,password=password,role="staff")
```

```
datal=log.objects.last()

stf.objects.create(login=datal,name=name,address=addr,aadhaar_no=aadhar,phone_no=phon
e,email=mail,state=datast,district=datadt,location=datalc,category=category,exp=Experience,
basic_salary=salary,photo=files,status="waiting")
response = redirect('/index')
return response
def userreg(request):
name=request.POST["name"]
addr=request.POST["addr"]
state=request.POST["state"]
district=request.POST["district"]
location=request.POST["location"]
phone=request.POST["phone"]
mail=request.POST["mail"]
files=request.FILES["file"]
username=request.POST["username"]
password=request.POST["password"]
datast=st.objects.get(state_id=state)
datadt=dt.objects.get(district_id=district)
datalc=loc.objects.get(location_id=location)
log.objects.create(username=username,password=password,role="user")
datal=log.objects.last()
usr.objects.create(login=datal,username=name,useraddress=addr,phoneno=phone,useremail=
mail,
state=datast,district=datadt,location=datalc,Photo=files,status="waiting")

response = redirect('/index')
return response
def staffhome(request):
role=request.session.get("role")
if role != "staff":
return redirect("/index")
return render(request,"staffhome.html")
```

```
def userhome(request):
role=request.session.get("role")
if role != "user":
return redirect("/index")
return render(request,"userhome.html")
def Add_state(request):
    msg=""
if request.POST:
        t1=request.POST["state"]
st.objects.create(state=t1)
msg="inserted successfully"
return render(request,"add_state.html",{"msg":msg})
def delete_state(request):
id=request.POST["s_id"]
st.objects.filter(state_id=id).delete()
response = redirect("/list_state")
return response
def edit_state(request):
id=request.POST["s_id"]
state=request.POST["state"]
st.objects.filter(state_id=id).update(state=state)
response = redirect("/list_state")
return response
def list_state(request):
datalst=st.objects.all()
return render(request,"state_list.html",{"data":datalst})

def add_location(request):
    msg=""
data=st.objects.all()
if request.POST:

        t2=request.POST["district"]
        t3=request.POST["location"]
```

```
datadt=dt.objects.get(district_id=t2)
loc.objects.create(location=t3,distict=datadt)
msg="inserted successfully"
return render(request,"add_location.html",{"msg":msg,"data":data})

def list_location(request):
data=loc.objects.all()
datas=st.objects.all()
return render(request,"list_location.html",{"data":data,"datas":datas})

def edit_location(request):
id=request.POST["l_id"]
state=request.POST["location"]
sid=request.POST["state"]
did=request.POST["district"]

datadt=dt.objects.get(district_id=did)
loc.objects.filter(location_id=id).update(location=state,distict=datadt)
response = redirect("/list_location")
return response

def delete_location(request):
id=request.POST["l_id"]
st.objects.filter(location_id=id).delete()
response = redirect("/list_location")
return response

def user_feed(request):
if request.POST:
    t1= request.POST["t1"]
    t2= request.POST["t2"]
fd.objects.filter(feedback_id=t1).update(reply=t2)
data=fd.objects.all()
return render(request,"user_feed.html",{"data":data})
```

```
def staff_complaints(request):
if request.POST:
        t1= request.POST["t1"]
        t2= request.POST["t2"]
cm.objects.filter(complaint_id=t1).update(reply=t2)
data=cm.objects.all()
return render(request,"staff_feed.html",{"data":data})

def complaints(request):
id=request.session['id']
datal=log.objects.get(logid=id)
datas=stf.objects.get(login=datal)
if request.POST:
        t1=request.POST["subject"]
        t2=request.POST["msg"]
cm.objects.create(staff=datas,sub=t1,msg=t2,reply="")
data=cm.objects.filter(staff=datas).all()
return render(request,"complaints.html",{"data":data})
def feedback(request):
id=request.session['id']
datal=log.objects.get(logid=id)
datau=usr.objects.get(login=datal)
if request.POST:
        t1=request.POST["feedback"]
fd.objects.create(userid=datau,feedback=t1,reply="")
data=fd.objects.filter(userid=datau).all()
return render(request,"feedback.html",{"data":data})

def find_labour(request):
datas= st.objects.all()
data=stf.objects.all()
datc=stf.objects.count()

if request.POST:
```

```
        t1=request.POST["state"]
        t2=request.POST["district"]
        t3=request.POST["location"]
        t4=request.POST["category"]
datast=st.objects.get(state_id=t1)
datadt=dt.objects.get(district_id=t2)
datalc=loc.objects.get(location_id=t3)
data=stf.objects.filter(state=datast,district=datadt,location=datalc).all()
datc=stf.objects.filter(state=datast,district=datadt,location=datalc).count()



return render(request,"find_labour.html",{"datas":datas,"data":data,"datc":datc})

def book_labour(request):
staff=request.GET["staff"]
state=request.GET["state"]
district=request.GET["district"]
location=request.GET["location"]
category=request.GET["category"]
datastf=stf.objects.get(staff_id=staff)
if request.POST:
        t1=request.POST["fdate"]
        t2=request.POST["todate"]
        t3=request.POST["category"]
        t4=request.POST["staff"]
        t5=request.POST["aboutjob"]
        t6=request.POST["amt"]
id=request.session['id']
datal=log.objects.get(logid=id)
datau=usr.objects.get(login=datal)

datas=stf.objects.get(staff_id=t4)
```

```
lb.objects.create(userid=datau,staff=datas,from_date=t1,to_date=t2,category=t3,desc=t5,amo
unt=t6,reject="",status="waiting")
return redirect("/find_labour/")

return
render(request,"book_labour.html",{"datastf":datastf,"staff":staff,"state":state,"district":distri
ct,"location":location,"category":category})



def staff_ongoing(request):
datag=lb.objects.filter(status="approved").exclude(reject="rejected").all()
return render(request,"staff_ongoing.html",{"data":datag})

def staff_rejected(request):
datag=lb.objects.filter(reject="rejected").all()
return render(request,"staff_rejected.html",{"data":datag})

def staff_completed(request):
datag=lb.objects.filter(status="stfcompleted").all()
return render(request,"staff_completed.html",{"data":datag})
def payment_completed(request):
datag=lb.objects.filter(status="completed").all()
return render(request,"payment_completed.html",{"data":datag})

def request_payment(request):
lid=request.POST["lid"]
lb.objects.filter(labour_id=lid).update(status="paymentrequested")
return redirect("/staff_completed")

def waiting_payment(request):
datag=lb.objects.filter(status="paymentrequested").all()
return render(request,"waiting_payment.html",{"data":datag})

def payment_request(request):
```

```
id=request.session['id']
datal=log.objects.get(logid=id)
datau=usr.objects.get(login=datal)
   msg=""
if request.POST:
      t1=request.POST["lid"]
      t2=request.POST["holder"]
      t3=request.POST["card"]
      t4=request.POST["cvv"]
      t5=request.POST["exp"]
      t6=int(request.POST["amt"])
bcc=bnk.objects.filter(holder=t2,card=t3,cvv=t4,exp=t5).count()
if bcc==1:
datb=bnk.objects.get(holder=t2,card=t3,cvv=t4,exp=t5)
bal=int(datb.bal)
if bal < t6 :
msg="insufficient Balance"
else:
bmt=bal-t6
bnk.objects.filter(holder=t2,card=t3,cvv=t4,exp=t5).update(bal=bmt)
lb.objects.filter(labour_id=t1).update(status="completed")
msg="payment successfull"
else :
msg="invalid account details"


datag=lb.objects.filter(status="paymentrequested",userid=datau).all()
datac=lb.objects.filter(status="paymentrequested",userid=datau).count()
return render(request,"payment_request.html",{"data":datag,"datac":datac,"msg":msg})

def payment_history(request):
datag=lb.objects.filter(status="completed").all()
return render(request,"payment_history.html",{"data":datag})
```

```
def ongoing(request):
id=request.session['id']
datal=log.objects.get(logid=id)
datau=stf.objects.get(login=datal)
datag=lb.objects.filter(status="approved",reject="confirm",staff=datau).all()
return render(request,"ongoing.html",{"data":datag})

# praveeen

def schedule(request):
datasb=lb.objects.exclude(status="completed").exclude(status="paymentrequested").exclude(
reject="rejected").all()
return render(request,"schedule.html",{"datakk":datasb})



def delete_labour(request):
id=request.POST["labour_id"]
lb.objects.filter(labour_id=id).delete()
response = redirect("/schedule")
return response

def complete(request):
dataty=lb.objects.filter(status="completed")
return render(request,"complete.html",{"datatt":dataty})

def rejected(request):
datatm=lb.objects.filter(reject="rejected").all()
return render(request,"rejected.html",{"datamm":datatm})



#/* amitha */
def add_district(request):
    msg=""
```

```
data=st.objects.all()
if request.POST:
    t1=request.POST["state"]
    t2=request.POST["district"]
datast=st.objects.get(state_id=t1)
dt.objects.create(state=datast,district=t2)
msg="inserted successfully"
return render(request,"add_district.html",{"msg":msg,"data":data})

def list_district(request):
data=dt.objects.all()
dataldt=st.objects.all()
return render(request,"list_district.html",{"data":data,"datas":dataldt})

def edit_district(request):
id=request.POST["d_id"]
state=request.POST["district"]
sid=request.POST["state"]
state=st.objects.get(state_id=sid)
dt.objects.filter(district_id=id).update(district=state)
response = redirect("/list_district")
return response

def delete_district(request):
id=request.POST["d_id"]
st.objects.filter(district_id=id).delete()
response = redirect("/list_district")
return response

def list_user(request):
datausr=usr.objects.filter(status="approved").all()
return render(request,"list_user.html",{"data":datausr})
def delete_user(request):
id=request.POST["user_id"]
```

```
usr.objects.filter(user_id=id).delete()
response = redirect("/list_user")
return response

def list_staff(request):
datast=stf.objects.filter(status="approved").all()
return render(request,"list_staff.html",{"data":datast})
def delete_staff(request):
id=request.POST["staff_id"]
stf.objects.filter(staff_id=id).delete()
response = redirect("/list_staff")
return response
def approve_staff(request):
datastf=stf.objects.filter(status="waiting").all()
return render(request,"approve_staff.html",{"data":datastf})

def approved_staff(request):
id=request.POST["staff_id"]
stf.objects.filter(staff_id=id).update(status="approved")
response = redirect("/approve_staff")
return response

def approve_user(request):
datauser=usr.objects.filter(status="waiting").all()
return render(request,"approve_user.html",{"data":datauser})

def approved_user(request):
id=request.POST["user_id"]
usr.objects.filter(user_id=id).update(status="approved")
response = redirect("/approve_user")
return response
def newjob(request):
data=lb.objects.filter(status="waiting").all()
return render(request,"newjob.html",{"data":data})
```

```
def job_confirm(request):
dataff=lb.objects.filter(status="approved",reject="").all()
return render(request,"job_confirm.html",{"datavb":dataff})

def confirm_labour(request):
id=request.POST["labour_id"]
lb.objects.filter(labour_id=id).update(reject="confirm")
return redirect("/job_confirm")

def reject_labour(request):
id=request.POST["labour_id"]
lb.objects.filter(labour_id=id).update(reject="rejected")
return redirect("/job_confirm")

def newjob_approve(request):
id=request.POST["lid"]
lb.objects.filter(labour_id=id).update(status="approved")
return redirect("/newjob")

def task_complete(request):
id=request.POST["lid"]
lb.objects.filter(labour_id=id).update(status="stfcompleted")
return redirect("/ongoing")
```