ONLINE TIMBER MILL

A PROJECT REPORT

Submitted By

ANJANA VIJAYAN

Reg.No:SJC17MCA006

to

the APJ Abdul Kalam Technological University
in partial fulfillment of the requirements for
the award of the degree
of

MASTER OF COMPUTER APPLICATIONS

Department of Computer Science and Applications
ST.JOSEPH'S COLLEGE OF ENGINERRING AND TECHNOLOGY
Bharananganam Pravithanam Road, Kottayam, Palai,
Choondacherry, Kerala 686579

May, 2020

DECLARATION

I undersigned hereby declare that the project report "ONLINE TIMBER MILL", submitted for partial fulfilment of the requirements for the award of degree of Master of Computer Applications of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under supervision of Mrs. Rinu Rachel Varughese . This submission represents my ideas in my own words and where ideas or words of hers have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place: Palai   **ANJANA VIJAYAN**
Date: **Reg.No: SJC17MCA006**

DEPARTMENT OF COMPUTER SCIENCE AND APPLICATIONS
ST.JOSEPH'S COLLEGE OF ENGINEERING AND TECHNOLOGY
Palai, Choondacherry Kottayam, 686579
(*Approved by AICTE and affiliated to APJ Abdul Kalam Technological University*)

CERTIFICATE

This is to certify that the report entitled **" Online Timber Mill "** submitted by **"ANJANA VIJAYAN , Reg.No:SJC17MCA006"** to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications is a bonafide record of the project work carried out by her under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Mrs.Rinu Rachel Varughese   **Mr.Alex Jose**

Asst. Professor MCA   Asst.Professor MCA

Internal Guide   Project Co-ordinator

Dr.T.D.Jainendrakumar
Head of the Department

Viva-voce held on:...

External Examiner 1:   **External Examiner 2:**

ACKNOWLEDGEMENT

ABSTRACT

Timber is an important material for building construction as well as furniture construction. ONLINE TIMBER MILL is a website that is used for buying and selling of woods in an online manner. This project consists of three module :- Admin, Mill and User. The admin has the entire control over the website. Mill's as well as user's registration are approved by the admin. Admin sets a premium amount for both users and mills and both of them must pay the amount before booking.User can add their wood details in the site. Users can sell and buy the woods for their needs. They can buy the woods by checking the wood details provided by different mills. Each mill can sell and buy the woods depending upon the notifications of the user. After booking a sitevisit is also allowded to both parties for ensuring the quality of the material . The main aim of this project is a developed website that is simple, user friendly, and provide easy access for all users.
Technology Used: Python (Django)

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 INTRODUCTION

The project entitled as "ONLINE TIMBER MILL" is a simple website for selling and buying of different types of woods in an online manner. By using this site peoples can sell their woods in an easiest way. The mill owners have the same provision for selling their material in anywhere in the country.And we also provide the provision for visiting the woods for ensuring the quality of the product.

I.2 ABOUT THE ORGANIZATION

The college was founded by a group of well known educators. They are pioneering educators, having unmatched experience in the field of education with a belief that the continuous search for knowledge is the sole path to success. The Primary focus of the institution is to expose the young minds to be world of technology, instilling in them confidence and fortitude to face new challenges that enables them to excel in their chosen field. The college inculcates the development of all facets of the mind culminating in an intellectual and balanced personality. Our team of dedicated and caring faculty strives to widen the students horizon of learning thereby achieving excellent results for every student. St. Joseph's college of Engineering & Technology (SJCET), has always been in the forefront with a wide spectrum of distinct features and facilities. The institution is a leader in the academia and its culture has set a benchmark in the region of quality in education. SJCET, Right from inception, has been maintaining high levels of standard in academic and extracurricular realms of activities. We offer BTEC degree courses in 6 engineering disciplines, and Masters Degree courses in Engineering, Computer Application and Business Administration. In the short span of a decade of its existence and among the six batches of students that have graduated, the college bagged several university ranks and has a remarkably high percentage of pass. The college is also the venue of national and

state level seminars and symposiums and has emerged as the hub of technical education in the state. The placement scenario is also quite commendable, with several premier industries visiting ST. Joseph's college of engineering & technology for placement and recruitment.

CHAPTER II
SYSTEM ANALYSIS

II.1 INITIAL INVESTIGATION

Preliminary investigation is a problem solving activity that requires interactive communication between the system user and system developer. It does various feasibility studies. In these studies, a rough figure system activities can be obtained from the decision about the strategies to be followed for effective system study and analysis can be taken.

In the preliminary investigation an initial picture about the system working is got from this study and data collection methods are identified. To launch a system investigation we need a master plan detailing the steps to be taken, the people to be questioned and the outcome expected. The scope of preliminary investigation may vary from a brief one person effort to an extensive series of activities requiring the participation may vary from a brief one person effort to an extensive series of activities requiring the participation of many individuals.

II.2 EXISTING SYSTEM

Online Timber Mill is an already existing system. In this system the peoples get information about the details of a wood.The user can buy woods through this system as well as they can sell their woods.

II.3 PROPOSED SYSTEM

As the proposed system is a web-based application where the users can perform all the arrangements easily and efficiently. This system provides a searching facility to the user based on their demands.

To overcome the drawbacks of the existing system, the proposed system has been evolved. The system provides with the best user interface. The efficient reports can be generated by using this proposed system. In this system there are mainly three logins, admin, users, and mills. The system allows us to post our requirements and buy it .Also we can test our soil. The user will get

more information about the available loans and schemes. And they can choose depends on there eligibility. The users will get more information about the market price details of crops across different regions accurately. And through this system farmers can directly contact with the customers. The new system developed includes provision for future expansion.

.

II.3.1 Advantages of the Proposed System

- The proposed system provides accurate data.
- The proposed system is very much faster than existing issue, searching is taking small amount of computerized time.
- Less time consuming and more efficient.
- Simple user interface to reduce processing
- Eliminate chances for errors and reduce effort

II.3.2 Features of the Proposed System

The various features of proposed system are as follows:

- Access to the system and database as per user identification the maximum security ensured.
- Integrity reliability and integrity of data User friendly and flexible in all aspects
- Data entry updates is quite easy
- Effective table manipulation as facilitated by the rich SQL Good validation checking
- Easy maintenance
- Removes chances of leakage of information. Provides a better record keeping system
- All these forms the major aspects and advantages of the proposed system. Provision is made for effective improvements of maintenance are needed at any stage.
- All these form the major aspects and advantages of the proposed system. Provision is made for effective improvements of maintenance are needed at any stage.

II.4   FEASIBILITY STUDY

During system analysis, a feasibility study of the proposed system was carried out to see whether it was beneficial to the organization. The main aim of the feasibility study is to determine whether it would be financially and technically feasible to develop the product. While evaluating the existing system, many advantages and disadvantages raised. Analyzing the problem thoroughly forms the vital part of the system study. Problematic areas are identified and information is collected.

The benefits of this site are users can easily interact and get the services without much complexity. It helps to make it possible that more users can interact with the site at a time. Feasibility study is to determine whether the proposed system is technically, economically and behaviourally feasible in all respects.

The main aim of feasibility study is to evaluate alternative site and propose the most feasible and desirable site for development. If there is no loss for the organization then the proposed system is considered financially feasible. A feasibility study is carried out to select the best system that meets performance requirements.

The feasibility study activity involves the analysis of the problem and collection of all relevant information relating to the product such as the different data items which would be input to the system, the processing required to be carried out on these data, the output data required to be produced by the system as well as various constraints on the behaviour of the system.

In this scenario, problems are identified. Essential data are being gathered for the existing problems. It is necessary that this analysis familiarizes the designer with objectives, activities, and the function of the organization in which the system is to be implemented. The feasibility study was divided into four:- Technical, Economical, Operational and Behavioural. It is summarized below:-

II.4.1 TECHNICAL FEASIBILITY

According to feasibility analysis procedure the technical feasibility of the system is analyzed and the technical requirements such as software facilities, procedure, inputs, are identified. While considering the problems of existing system, it is sufficient to implement the new system. The proposed system can be implemented to solve issues in the existing system. It includes the evaluation of and how it meets the proposed system. This system use Python Django framework and Mysql as back end technology.

II.4.2 ECONOMIC FEASIBILITY

Economic analysis is most frequent used for evaluating of the effectiveness of the candidate system. More commonly known as cost/benefit analysis the procedure is to determine the benefit and saving that are expected from a candidate system and compare them with the existing system. Except for the initial capital amount and the amount after each financial year, no other huge amount is needed. The expenses can be handles by any participants. So, the system is economically feasible.

This feasibility involves some questions such as whether the firm can afford to build the system, whether its benefits should substantially exceed its costs, and whether the project has higher priority and profits than other projects that might use the same re- sources. Here there is no problem. This firm has fully equipped hard ware, and fully fledged software, so no need to spend money on these issues. And as the client and the developer are one, there is no further problem in economic issues.

II.4.3 OPERATIONAL FEASIBILITY

Methods of processing and presentation are all according to the needs of clients since they can meet all user requirements here. The proposed system will not cause any problem under any circumstances and will work according to the specifications mentioned. Hence the proposed system is operationally feasible.

People are inherently resistant to change and computer has been known to facilitate changes. The

system operation is the longest phase in the development life cycle of a system. So, Operational Feasibility should be given much importance. This system has a user-friendly interface. Thus it is easy to handle.

II.4.4 BEHAVIORAL FEASIBILITY

In today's world, computer is an inevitable entity. As per the definition of behaviour design, many valid points are recognized in this study. This system behaviour changes according to different environment. In order to ensure proper authentication and authorization and security of sensitive data of the admin or employers, login facilities are provided. These are the main feasibility studies tested in this application.

CHAPTER III
SYSTEM ANALYSIS AND DESIGN

III.1 SOFTWARE REQUIREMENT SPECIFICATION

The primary goal of the system analyst is to improve the efficiency of the existing system. For that study of specification of the requirement is very essential. For the development of the new system, a preliminary survey of the existing system will be conducted. An investigation is done whether the up gradation of the system into an application program could solve the problems and eradicate the inefficiency of the existing system. This gives an idea about the system specifications required to develop and install the project "ONLINE TIMBER MILL".

The System Requirements Specification is based on the System Definition. The requirement specifications are primarily concerned with functional and performance aspect of a software product and emphasis are placed on specifying product characteristics implying how the product will provide those characteristics. One of the most difficult tasks is selecting software, once the system requirement is find out then we have to determine whether a particular software package fits for those system requirements. This selection summarizes the application requirement.

III.1.1 HARDWARE REQUIREMENT

- CPU – INTEL CORE i3
- HARD DISKSPACE - 500 GB
- RAM - 2GB
- DISPLAY - 19 STANDARD RATIO LCDMONITOR
- KEYBOARD - 99-104 KEYS
- CLOCK SPEED - 1.99 GHZ

III.1.2 SOFTWARE REQUIREMENT

- OPERATING SYSTEM – WINDOWS 10
- WEB SERVER - WSGI PYTHON DJANGO SERVER
- FRONT END – PYTHON DJANGO FRAMEWORK
- BACK END – MYSQL

III.2 SYSTEM DESIGN

Designing the system in an effective way leads to the smooth working of any software's. System design is the process of developing specification for a candidate system that meet the criteria established in the system analysis. Major step in the system design is the preparation of the input forms and output reports in a form applicable to the user. The main objective of the system design is to use the package easily by any computer operator. System design is the creative act of invention, developing new inputs, and database, off-line files, method, procedure and output for processing business to meet an organization objective. System design builds information gathered during the system analysis. This system is designed neatly so that user will never get ambiguity while using the system.

III.2.1 NON-FUNCTIONAL REQUIREMENTS

Performance Requirements

For the efficient performance of the application, network must have high bandwidth so that the task of centralized management does not lead to network jam. Also the hard disk capability must be high so that data can be effectively stored and retrieved.

Security Requirements

Security requirements of this application involves authentication using user name and password so that invalid users are restricted from data access. For the security of data, periodic database backups must be performed so that we can recover data in the case of data loss.

III.3 UNIFIED MODELING LANGUAGE [UML]

UML is a way of visualizing a software program using a collection of diagrams. The notation has evolved from the work of Grady Booch, James Rumbaugh, Ivar JAcobson and the RationalSoftware Corporation to be used for object-oriented design, but it has since been extended to cover a wider variety of software engineering projects. Today, UML is accepted by the Object Management Group (OMG) as the standard for modelling software development.

UML stands for Unified Modeling Language.UML 2.0 helps extend the original UML specification to cover a wider portion of software development efforts including agile practices.

Improved integration between structural models like class diagrams and behavior models like activity diagrams.

The original UML specified nine diagrams; UML 2.x brings that number up to 13. The four new diagrams are called: communication diagram, composite diagram, interaction overview diagram and timing diagram. It also renamed state chart diagrams to state machine diagrams, also known as state diagrams.

Types of UML diagrams

The current UML standards call for 13 different types of diagrams: class, activity, object, use case, sequence, package, state, component, communication, composite structure, interaction overview, timing and deployment. These diagrams are organized into two distinct groups: structural diagrams and behavioral or interaction diagrams.

Structural UML diagrams

- Class diagram

- Package diagram

- Object diagram

- Component diagram

- Composite structure diagram

- Deployment diagram

Behavioral UML diagrams

- Activity Diagram

- Sequence diagram

- Use case diagram

- State diagram

- Communication diagram

- Interaction overview diagram

- Timing diagram

- **III.3.1   Use case Diagram**

To model a system the most important aspect is capture the dynamic behaviour. To modify a bit in details, dynamic behaviour of the system when it is running or operating. So only behaviour is not sufficient to model a system rather dynamic behaviour is more important than static behaviour. In UML there are five diagrams available to model dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction. These internal and external agents are known as actors. So use case diagram consists of actors, use case and their relationships. The diagram is used to model the system of an application. A single use case diagram captures a particular functionality of a system.

Use case Diagram objects:

- Actor

- Use case

- System

- Package Actor
  Actor
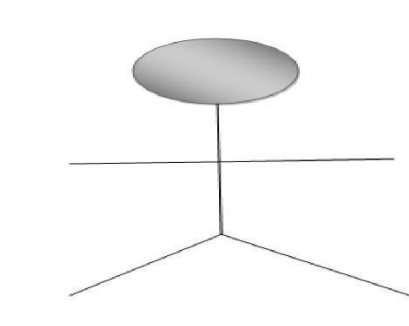
Figure III.1: Actor

Actor is a use case diagram in an entity that performs a role in one given system. This could be a person, organization or an external system usually drawn like skeleton.

Use case

A use case represents a function or an action within the system. It's drawn as an oval and named with the function.

Figure III.2: Use Case

System

System is used to define the scope of the use case and drawn as a rectangle. This is an optional element but useful when your visualizing large systems. For example you can create all the use cases and then use the system object to define the scope covered by your project. Or you can even use it to show the different areas covered in different releases.

Package

Package is another optional element that is extremely useful in complex diagrams. Similar to use class diagrams, packages are used to group together use cases.
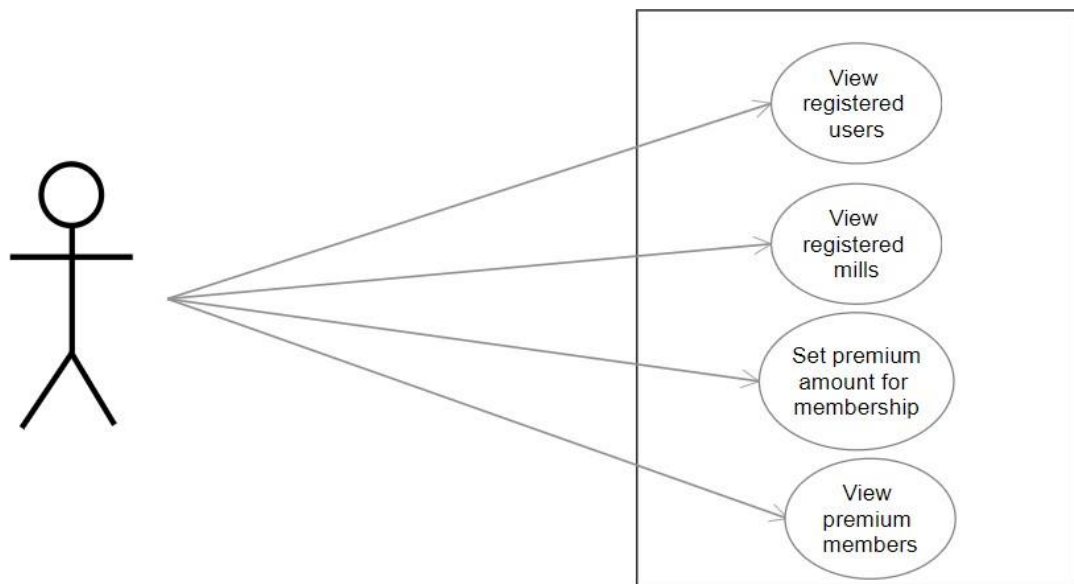
Fig11.1: Use case diagram for admin
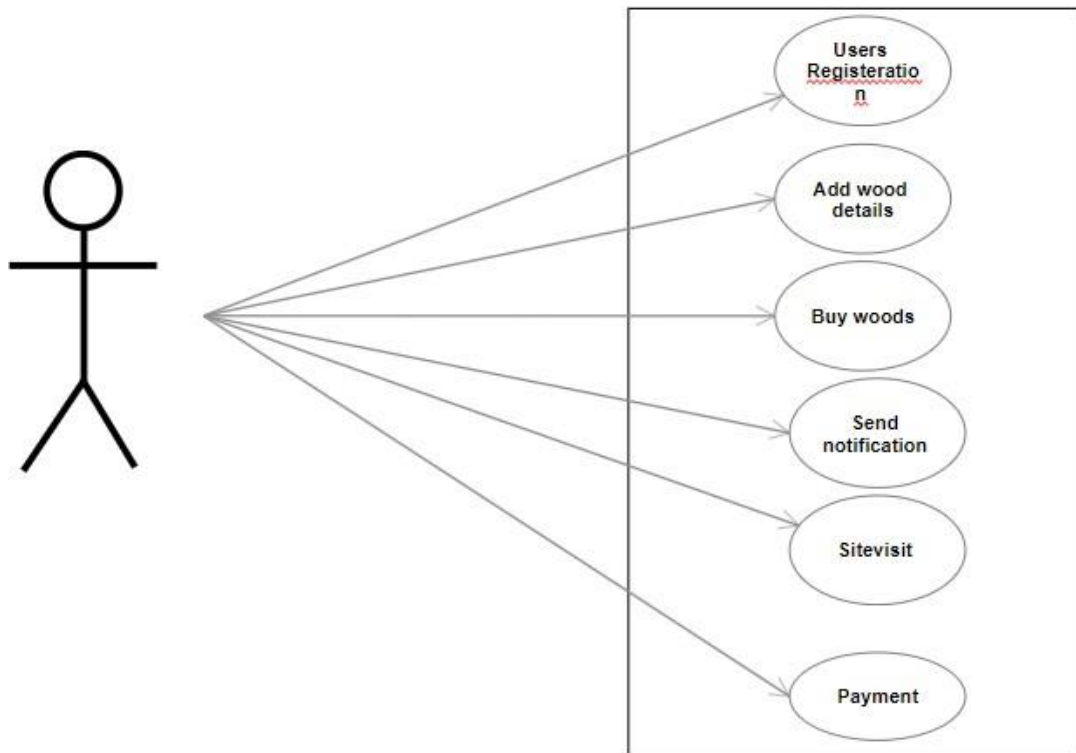
Fig11.2: Use case diagram for User
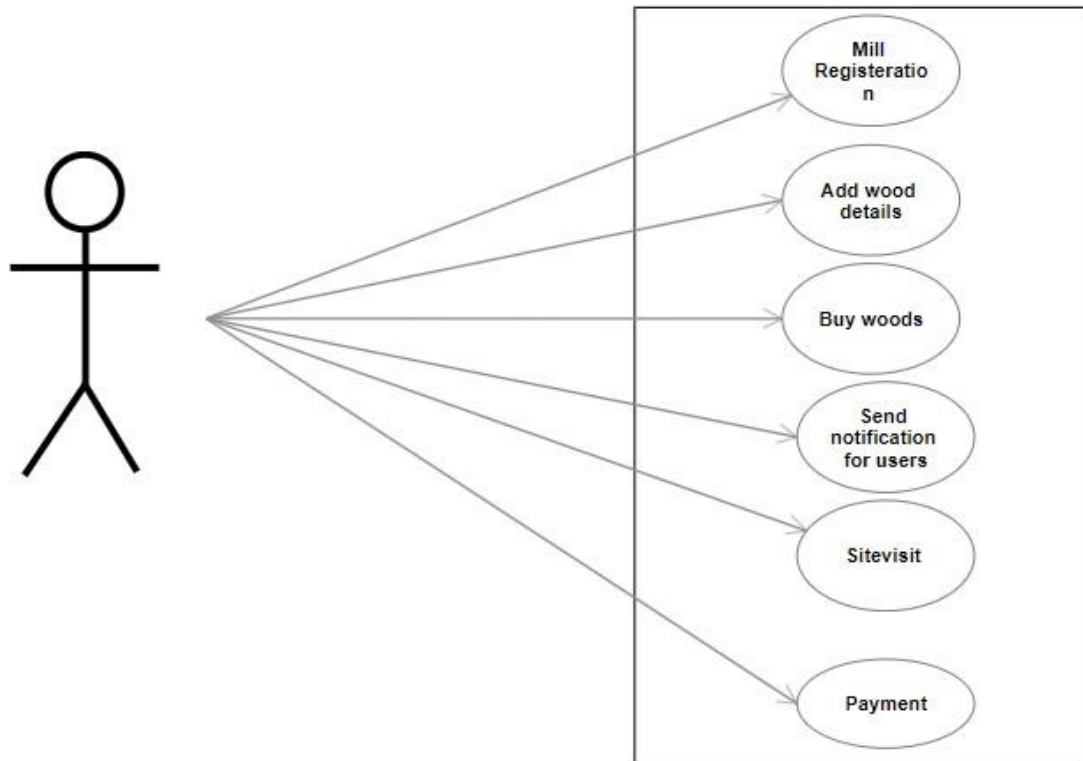
Fig11.3: Use case diagram for Mill

III.3.2 Sequence Diagram

UML sequence diagrams are used to represent or model the flow of messages, events and actions between the objects or components of a system. Time is represented in the vertical direction showing the sequence of interaction of the header elements.

Sequence Diagrams are used primarily to design, document and validate the architecture, interfaces and logic of the system by describing the sequence of actions that need to be performed to complete a task. UML sequence diagrams are useful design tools because they provide a dynamic view of the system behavior which can be difficult to extract from static diagrams or specifications.

Although UML sequence diagrams are typically used to describe object-oriented software

16

systems, they are also extremely useful as system engineering tools to design system architectures in business process, as message sequence charts and call flows for telecoms or wireless system design, and for protocol stack design and analysis.

A sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a message sequence chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence Diagrams are typically associated with use case realizations in the logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.
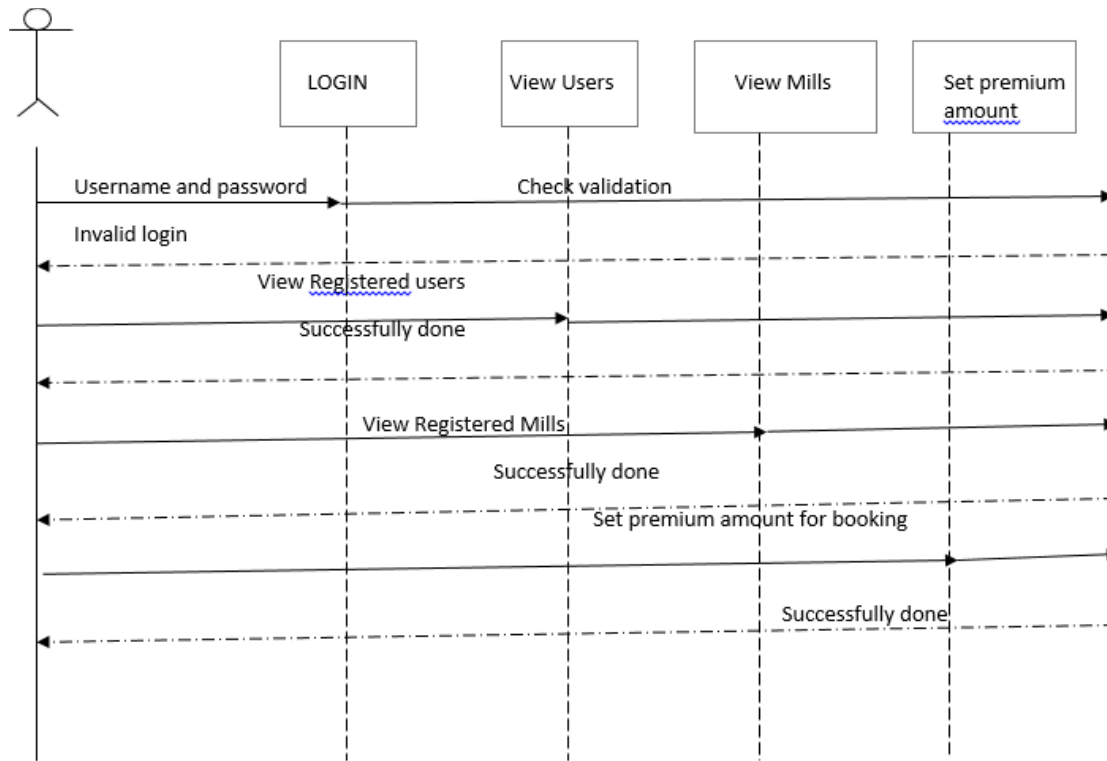
Figure III.1: Sequence diagram for Admin
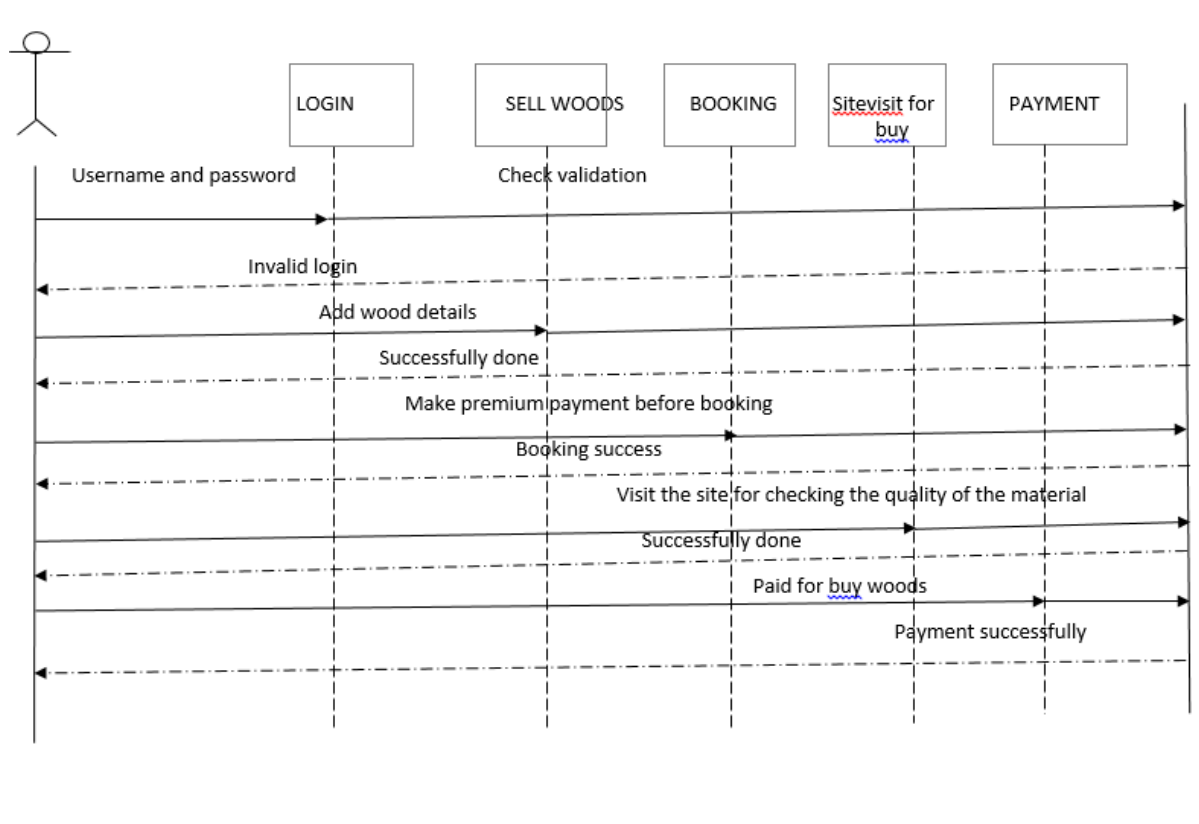
Figure III.2: Sequence diagram for User
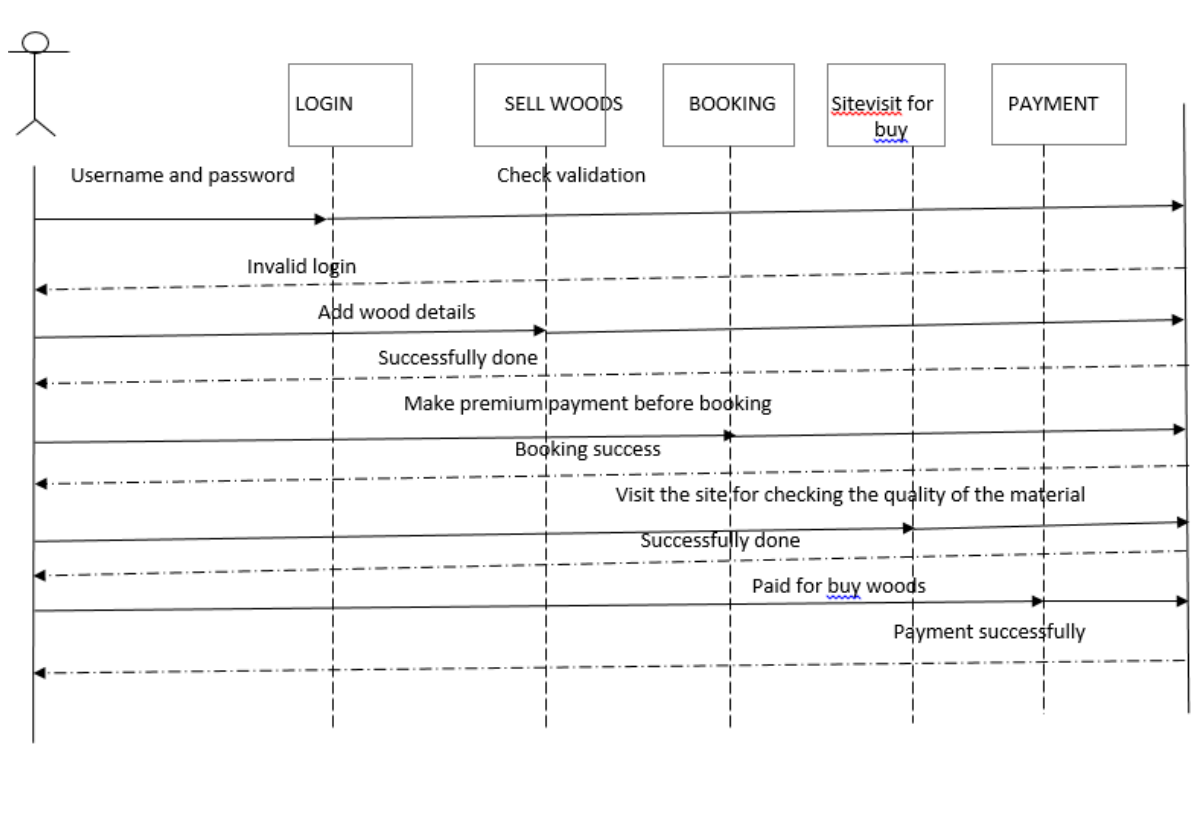
Figure III.3: Sequence diagram for Mill

III.3.3 Activity Diagram

The basic purposes of activity diagrams are similar to other four diagrams. It captures the dynamic behaviour of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.

Activity diagrams are mainly used as a flowchart that consists of activities performed by the system. Activity diagrams are not exactly flowcharts as they have some additional capabilities. These additional capabilities include branching, parallel flow, swimlane, etc.

Before drawing an activity diagram, we must have a clear understanding about the elements used in activity diagram. The main element of an activity diagram is the activity itself. An activity is a function performed by the system. After identifying the activities, we need to understand how they are associated with constraints and conditions.

Before drawing an activity diagram, we should identify the following elements –
- Activities
- Association
- Conditions
- Constraints

Figure III.5: Activity diagram for Admin
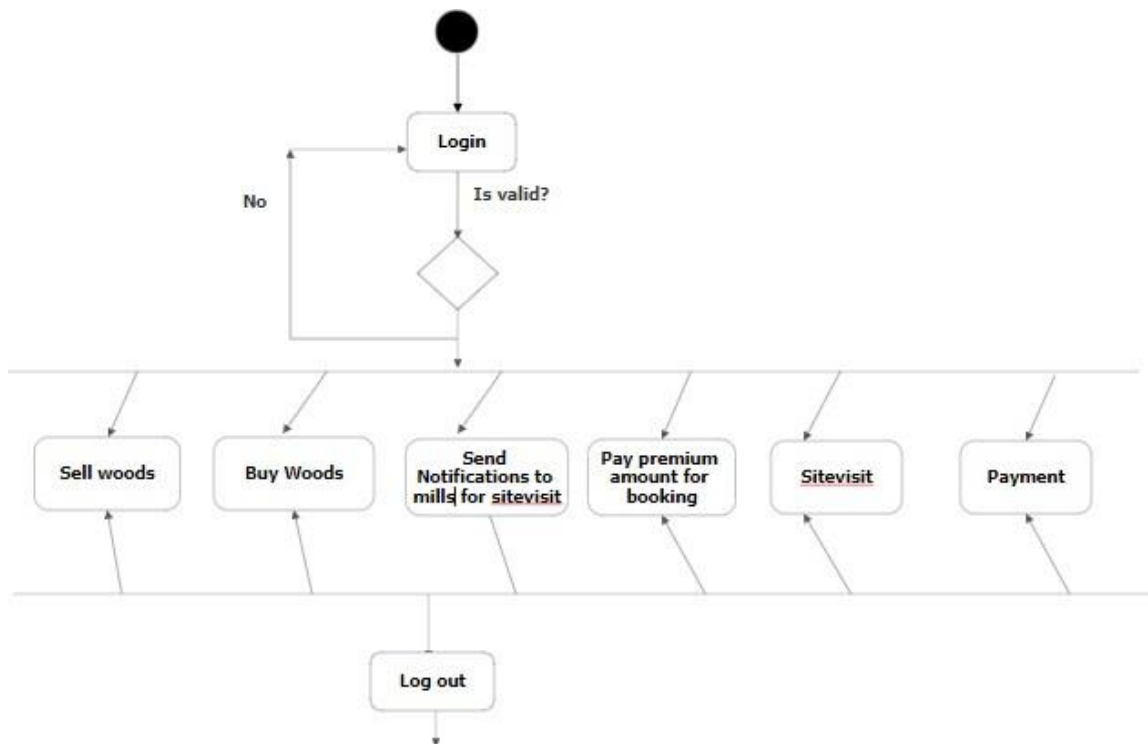
Figure III.6: Activity diagram for User
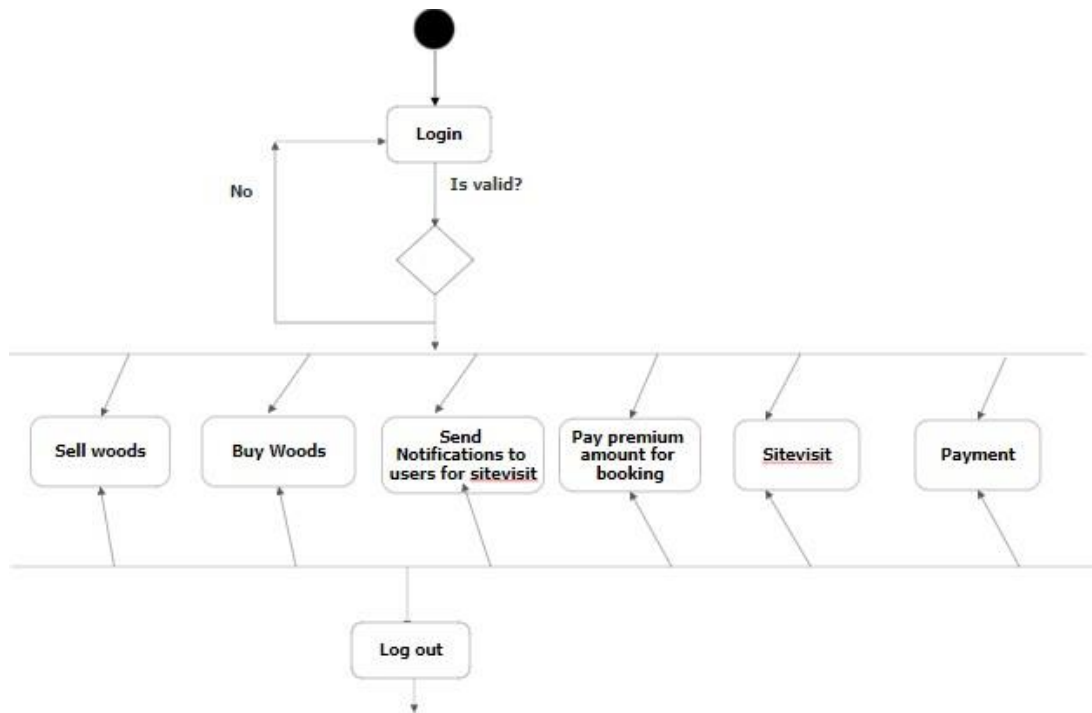
Figure III.7: Activity diagram for Mill

III.4 SYSTEM DESIGN

The most creative and challenging phase of the system life cycle is the system design. The term design describes a final system and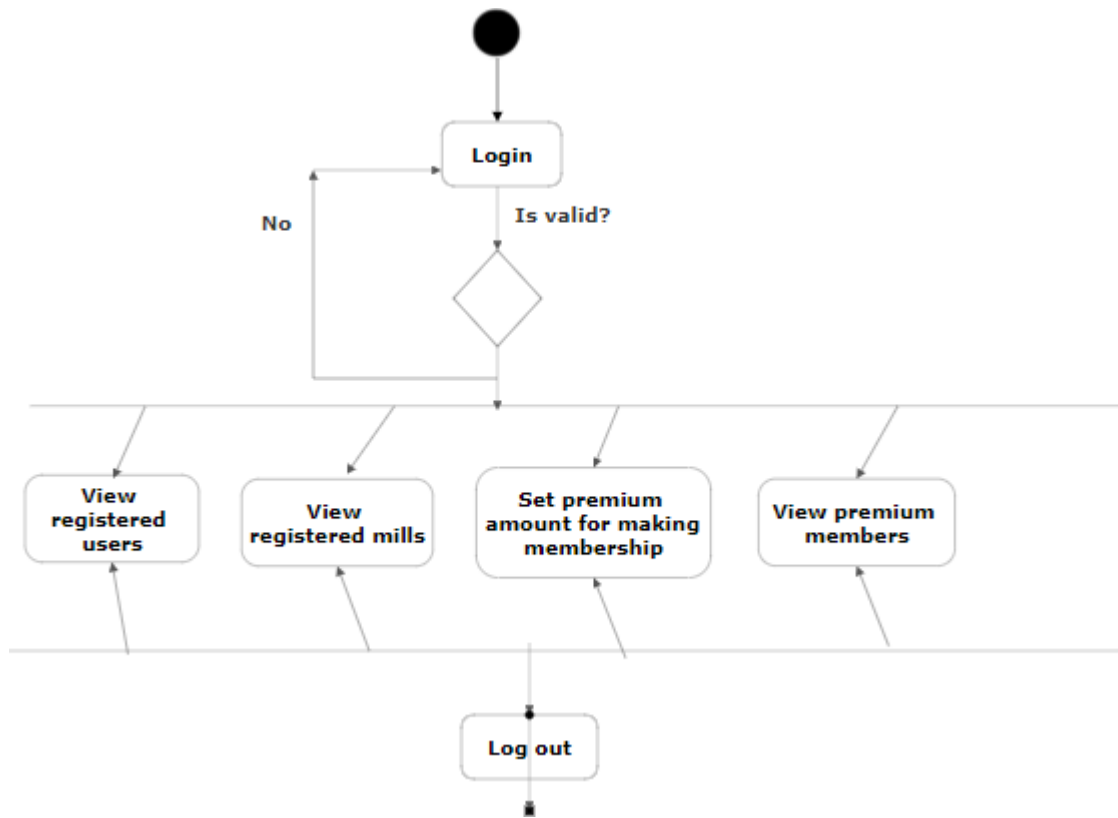 the process by which it is developed. It refers to the technical specification that will be applied in implementing the candidate system. In system design, we move from the logical to the physical aspects of the life cycle.

The first step is to determine how the output is to be produced and in what format. Then input data and master files have to be designed as the next step and finally the impact of the candidate system on the user and organization are documented and evaluated by the management. After identifying the problem and the limitation of the existing system, a detailed design of the proposed system is conducted.

Free flow personnel interview and reference to previous records prepared manually were the only methods taken to collect necessary information. At present, all organizations are on the path of computerization process.

Design is the phase that indicates the final system. It is the solution, the translation of requirements into ways of meeting them. In this phase the following elements were designed namely, data flow, data stores, processes, procedures was formulated in a manner that meet the project requirements. After logical design physical construction of the system is done.The database tables, input screens, output screens, output reports are designed. After analysing the various functions involved in the system the database, labels as dictionaries designed. Care is taken for the field name to be in self-explanatory form. Unnecessary fields are avoiding so as not affecting the storage system.

Care must be taken to design the input screen in the most user-friendly way so as to help even the novice users to make entries approximately in the right place. This is being accomplished by the use of giving online help messages, which are brief and cleanly prompts users for appropriate

action. Design is the only way that we can accurately translate a customer's requirements into a finished software product or system. Without design, risk of building an unstable system exist one that will fail when small changes are made, one that will be difficult to test.

All input screens in the system are user friendly and are designed in such a way that even a layman can operate. The sizes of all screens are standardized.

Reports generated in this software give the finer accepts of the required information, which helps in taking vital decision.

The importance of the software design can be stated with a single word quality. Design is a place where quality is fostered in software development. Design is the only way where requirements are actually translated into a finished software product or system.

Mainly this project consists of 3 Modules:

- **Admin**
- **Mill**
- **Users**

Admin Module

Administrator is the main actor in this system. He has the entire control of the system which includes accepting or rejecting the registration of mill as well as users. Brief description about the functionalities performed by the admin is given below. After the admin successfully login to this website the admin can perform the functionalities including:

Admin Login

- Registration
- Login
- View registered mills
- View registered users
- Issue a premium amount for both mills and users
- View premium members

User

- Registration
- Login
- Add wood details for selling their woods.
- Issue site visit details for booked customers
- Make a premium membership for booking
- Buy woods from different mills
- Site visit
- Payment

Mill

- Registration
- Login
- Add wood details
- Check notifications from the users
- Sell woods
- Pay a premium amount for making payments
- Buy woods from different users

- Send site visit notifications to the users
- Site visit
- Payment

III.4.1 Input Design

Input design is the process of converting a user oriented description of the inputs to a computer based business system into a programmer oriented specification. The design decision for handling input specify how data are accepted for computer processing. Input design is a part of overall design that needs careful attention. The collection of input data is considered to be the most expensive part of the system design. Since the inputs have to be planned in such a way so as to get the relevant information, extreme care is taken to obtain the pertinent information. If the data going into the system is incorrect then the processing and outputs will magnify these errors. The goal of designing input data is to make data entry as easy, logical and free from errors as possible. The following are the objectives of input design:

- To produce a cost effective method of input.
- To ensure validation

Effort has been made to ensure that input data remains accurate from the stage at which it is recorded and documented to the stage at which it is accepted by the computer. Validation procedures are also present to detect errors in data input, which is beyond control procedures. Validation procedures are designed to check each record, data item or field against certain criteria.

In my proposed system Image Compression, data has to be accurate and complete. If not, error messages are displayed to the user and he is unable to proceed to the next stage of action unless he corrects his data. Duplicate entries are not allowed. The data validation, a procedure of the

proposed system, provides program checks for the completeness, consistency, reasonableness and sequence of the system.

Maximum care has been taken to ensure that user types in only minimum data into the system, as all he/she will have to do is to move and click the mouse or strike a key to select the desired data at the desired position.

The screens are designed in such a way that the user can find the needed like options, actions etc. with ease of use. The needed columns, where interaction is needed, like labels, buttons are also simple. The related data columns are clubbed together as groups, so that the user can understands the related data easily.

The input design is the link between the information system and the user. It comprises developing specifications and procedures for data preparation and those steps that are necessary to put input data into a usable form for processing data entry. The design of inputs focuses on controlling the amount of inputs required, controlling errors, avoiding delay, avoiding extra steps and keeping the process simple.

III.4.2 Data Flow Diagram

Data flow diagram is the graphical representation of the system. It is a network that uses special symbols to describe the flow of data and process that transforms data throughout the system.
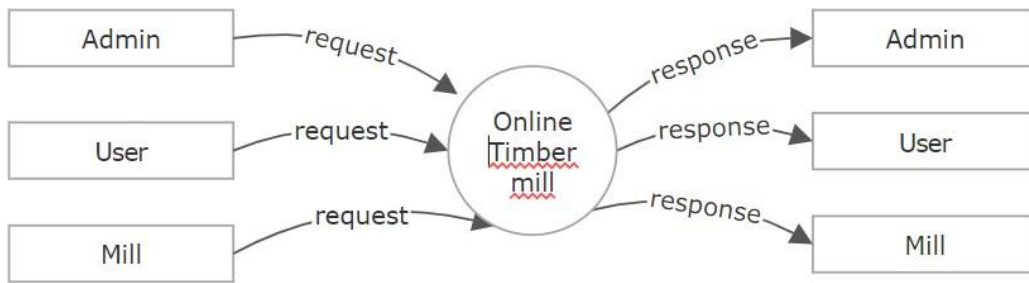
Figure III.10: Level 0 DFD
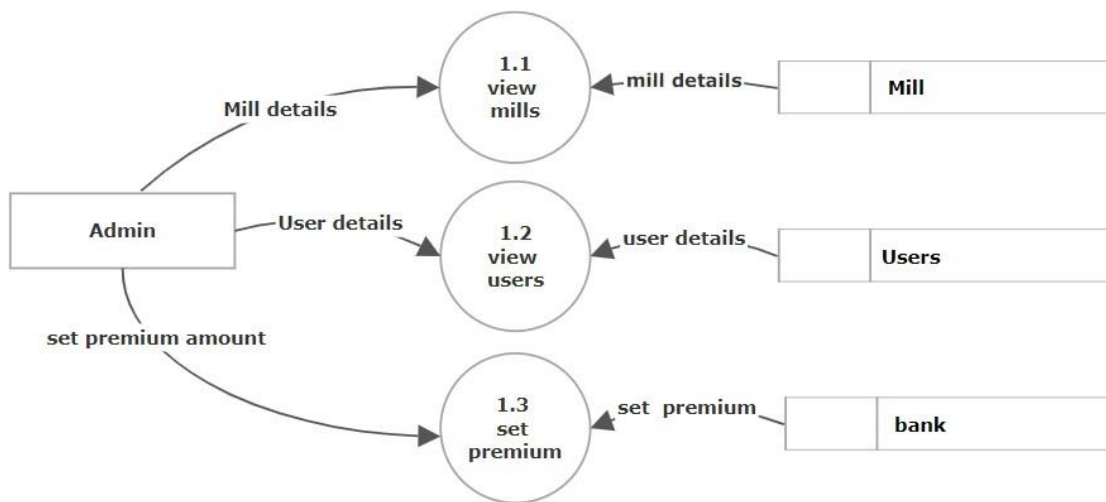
Figure III.11: Level 1 DFD for Admin
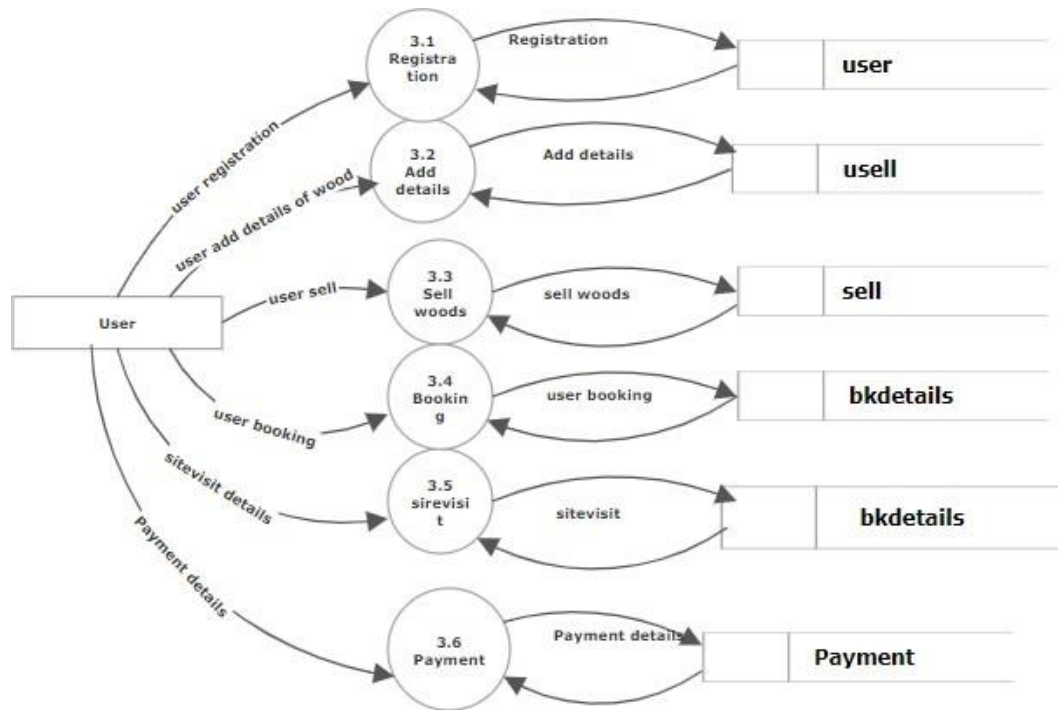
Figure III.12: Level 1 DFD for User
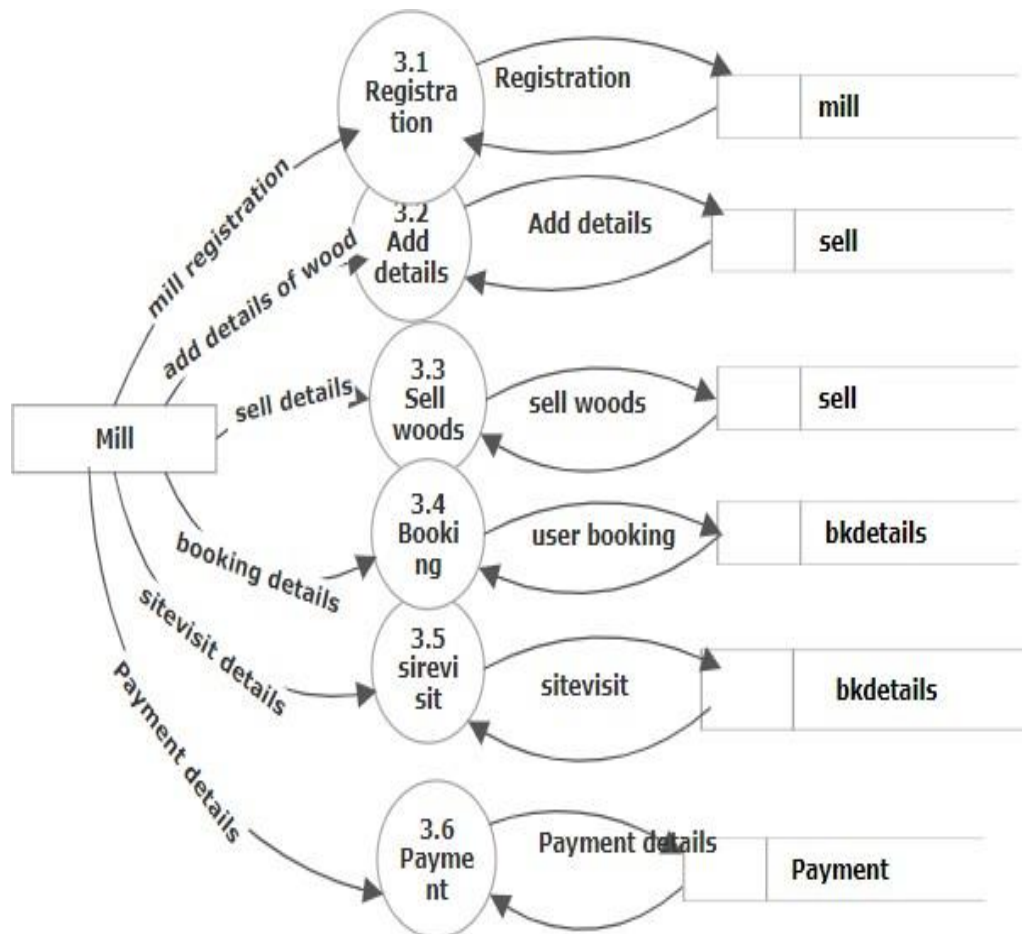
Figure III.13: Level 1 DFD for Mills

III.4.3 Output Design

The output design phase of the system design is concerned with the conveyance of information to the end users in user friendly manner. The output design should be efficient, intelligible so that the system relationship with the end user is improved and thereby enhancing the process of decision making. The output design is an ongoing activity almost from the beginning of the project, efficient and well defined output design improves the relation of the system and the user. The primary considerations in the design of the output are the requirement of the information and the objective of the end user. There are various types of outputs required by most of the

systems, but outputs of Image Compression are purely interactive outputs which involve the user in communicating with the computer.

The system output may be of any of the following
- A report
- A document
- A message

The output design specification is made in such a way that it is unambiguous and comprehensive. The approach to output design is very dependent on the type of output and nature of data. Special attention has to be made to data editing. The choice of appropriate output medium is also an important task. The output designed must be specified and documented, data items have to be accurately defined and arranged for clarity. The layout of the output will be normally specified on a layout chart. The final design layout must be approved by the user, communicated in detailed to the programmer. The user's requirements are quite different from that of the programmer. Before preparing a specification for the programmer, it is prudent to ensure that the design is acceptable to the user.

III.4.4 Database Design

TABLE1:USER

Primary key:Uid

FIELD	DATA TYPE	DESCRIPTION
Uid	Integer	Id of user table
First_name	String	First name of user
Last_name	String	Last name of user
Email	String	Email id of user
Mob	BigInteger	Mobile number of the user
Username	String	Username of the user
Password	String	Password of the user

TABLE 2: MILL
Primary key:mid

Field	Data type	Description
Fname	String	First name
Lname	String	Last name
Email	String	Email id
Pwd	String	Password

TABLE 3: sell

Primary key: sid
Foreignkey : uid

FIELD	DATA TYPE	DESCRIPTION
Sid	Integer	Id of the table
uid	Integer	Id of User table
Wood_type	String	Type of the wood
Length	Integer	Length of the wood
Size	Integer	Size of the wood
Rate	Integer	Rate of the wood

TABLE5: Premium

Primary key: pid

FIELD	DATA TYPE	DESCRIPTION
Pid	Integer	ID of the table
Mamt	Float	Membership Amount
status	String	Payment status

TABLE 6: bkdetails

Primary key: bid
Foreign Key:sid

FIELD	DATA TYPE	DESCRIPTION
Bid	Integer	Id of the table
Sid	Integer	Id of sell table
Wo_id	String	Wood id
User	String	Username of mill
Uemail	String	Email id of user
Mill	String	Name of mill
Memail	String	Email id of mill
Timber	String	Type of wood
Length	Integer	Length of the wood
Cirf	Integer	Circumference of the wood
Price	Integer	Rate of the wood
Svdate	Datefield	Date for site visit
Svtime	String	Time for site visit

TABLE:7 PAYMENT

Primary key:ptid

Foreign Key:Uid,Mid

Field	Data type	Description
Ptid	Integer	Id of the table
Uid	Integer	User id
Mid	Integer	Mill id
amt	String	Amount
status	String	Status

III.5   TOOLS AND PLATFORMS

III.5.1 Python

Python is an interpreted high level programming language for general purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community after 30 years.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community based development model, as do nearly all of Python's other implementations. Python andCPython are managed by the non profit Python Software Foundation.

Python Environment:

A Python environment is a child/offshoot of a parent python distribution which allows you to use the packages in the parent distribution as well as to install packages that are only visible to the child distribution.

Python Language:

Python is a powerful high level, object oriented programming language created by Guido van Rossum. It has simple easy to use syntax, making it the perfect language for someone trying to

learn computer programming for the first time.

Python is a general purpose language. It has wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D).

The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

Python Runtime Environment:

The runtime environment used to execute the code. It is made up of the Python language and Python interpreter. It is portable and it is platform neutral.

Python tools:

It is used by the developers to create Python code. They include Python compiler, Python interpreter, classes, libraries etc.

Python Application:

Applications are programs written in Python to carry out certain tasks on standalone local computer. Python source code is automatically compiled into Python byte code by the CPython interpreter. Compiled code is usually stored in PYC (or PYO) files, and is regenerated when the source is updated, or when otherwise necessary.

To distribute a program to people who already have Python installed, you can ship either the PY files or the PYC files. In recent versions, you can also create a ZIP archive containing PY or PYC files, and use a small "bootstrap script" to add that ZIP archive to the path.

III.5.2 Django Framework

Django is a high level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid for support.

Django helps you write software that is:

Complete

Django follows the "Batteries included" philosophy and provides almost everything developers might want to do "out of the box". Because everything you need is part of the one "product", it all works seamlessly together, follows consistent design principles, and has extensive and up to date documentation.

Versatile

Django can be (and has been) used to build almost any type of website — from content management systems and wikis, through to social networks and news sites. It can work with any client side framework, and can deliver content in almost any format (including HTML, RSS feeds, JSON, XML, etc). The site you are currently reading is based on Django!

Internally, while it provides choices for almost any functionality you might want (e.g. several popular databases, templating engines, etc.), it can also be extended to use other components if needed.

Secure

Django helps developers avoid many common security mistakes by providing a framework that has been engineered to "do the right things" to protect the website automatically. For example,

Django provides a secure way to manage user accounts and passwords, avoiding common mistakes like putting session information in cookies where it is vulnerable (instead cookies just contain a key, and the actual data is stored in the database) or directly storing passwords rather than a password hash.

A password hash is a fixed length value created by sending the password through a cryptographic hash function. Django can check if an entered password is correct by running it through the hash function and comparing the output to the stored hash value. However due to the "one way" nature of the function, even if a stored hash value is compromised it is hard for an attacker to work out the original password.

Django enables protection against many vulnerabilities by default, including SQL injection, cross-site scripting, crosssite request forgery and clickjacking (see Website security for more details of such attacks).

Scalable

Django uses a component based "shared-nothing" architecture (each part of the architecture is independent of the others, and can hence be replaced or changed if needed). Having a clear separation between the different parts means that it can scale for increased traffic by adding hardware at any level: caching servers, database servers, or application servers. Some of the busiest sites have successfully scaled Django to meet their demands (e.g. Instagram and Disqus, to name just two).

Maintainable

Django code is written using design principles and patterns that encourage the creation of maintainable and reusable code. In particular, it makes use of the Don't Repeat Yourself (DRY) principle so there is no unnecessary duplication, reducing the amount of code. Django also promotes the grouping of related functionality into reusable "applications" and, at a lower level, groups related code into modules (along the lines of the Model View Controller (MVC) pattern).

Portable

Django is written in Python, which runs on many platforms. That means that you are not tied to any particular server platform, and can run your applications on many flavours of Linux, Windows, and Mac OS X. Furthermore, Django is well supported by many web hosting providers, who often provide specific infrastructure and documentation for hosting Django sites.

III.5.3 Python - Spacy

Spacy is a relatively new package for "Industrial strength NLP in Python" developed by Matt Honnibal at Explosion AI. It is designed with the applied data scientist in mind, meaning it does not weigh the user down with decisions over what esoteric algorithms to use for common tasks and it's fast. Incredibly fast (it's implemented in Cython). If you are familiar with the Python data science stack, spacy is your numpy for NLP – it's reasonably low level, but very intuitive and performant.

Spacy provides a one stop shop for tasks commonly used in any NLP project, including:

- Tokenisation
- Lemmatisation
- Part-of-speech tagging
- Entity recognition
- Dependency parsing
- Sentence recognition
- Word-to-vector transformations
- Many convenience methods for cleaning and normalizing text

III.5.4 Django – Crispy – Forms

Django-crispy-forms is a Django application that lets you easily build, customize and reuse forms using your favorite CSS framework, without writing template code and without having to take care

of annoying details. Django-crispy-forms implements a class called FormHelper that defines the form rendering behavior. Helpers give you a way to control form attributes and its layout, doing this in a programatic way using Python. This way you write as little HTML as possible, and all your logic stays in the forms and views files.

III.5.5 Python – xhtml2pdf

xhtml2pdf is a HTML/XHTML/CSS to PDF converter written in Python and based on Reportlab Toolkit, pyPDF, TechGame Networks CSS Library and HTML5lib. The primary focus is not on generating perfect printable webpages but to use HTML and CSS as commonly known tools to generate PDF files within Applications. For example generating documentations (like this one), generating invoices or other office documents etc.xhtml2pdf enables users to generate PDF documents from HTML content easily and with automated flow control such as pagination and keeping text together. The Python module can be used in any Python environment, including Django.

xhtml2pdf is a html2pdf converter using the ReportLab Toolkit, the HTML5lib and pyPdf. It supports HTML 5 and CSS 2.1 (and some of CSS 3). It is completely written in pure Python so it is platform independent.The main benefit of this tool that a user with Web skills like HTML and CSS is able to generate PDF templates very quickly without learning new technologies.xhtml2pdf was previously developed as "pisa".

III.5.6 Pycharm IDE 3.4

JetBrains has developed PyCharm as a cross-platform IDE for Python. In addition to supporting versions 2.x and 3.x of Python, PyCharm is also compatible with Windows, Linux, and macOS. At the same time, the tools and features provided by PyCharm help programmers to write a variety of software applications in Python quickly and efficiently. The developers can even customize the PyCharm UI according to their specific needs and preferences. Also, they can extend

43

the IDE by choosing from over 50 plugins to meet complex project requirements.

Important Features and Tools Provided by Visual studio code

Code Editor

The intelligent code editor provided by PyCharm enables programmers to write high quality Python code. The editor enables programmers to read code easily through colour schemes, insert indents on new lines automatically, pick the appropriate coding style, and avail context-aware code completion suggestions. At the same time, the programmers can also use the editor to expand a code block to an expression or logical block, avail code snippets, format the code base, identify errors and misspellings, detect duplicate code, and auto generate code. Also, the editor makes it easier for developers to analyze the code and identify the errors while writing code.

Code Navigation

The smart code navigation options provided by PyCharm help programmers to edit and improve code without putting extra time and effort. The IDE makes it easier for programmers to go to a class, file and symbols, along with the go to declarations invoked from a reference. The user can even find an item in the source code, code snippet, UI element, or user action almost immediately. They can further locate usage of various symbols, and set bookmarks in the code. At the same time, the developers can even take advantage of the code navigation feature to scrutinize the code thoroughly in the lens mode.

Refactoring

PyCharm makes it easier for developers to implement both local and global changes quickly and efficiently. The developers can even take advantage of the refactoring options provided

by the IDE while writing plain Python code and working with Python frameworks. They can avail the rename and move refactoring for files, classes, functions, methods, properties, parameters, and local/global variables. Likewise, they can improve code quality by extracting variables, fields, constants, and parameters. Also, PyCharm allows programmers to break up longer classes and methods through extract method.

Support for Popular Web Technologies

PyCharm makes it easier for programmers to write various web applications in Python supporting widely used web technologies like HTML, CSS, JavaScript, TypeScript and CoffeeScript. The web developers can use the live editing preview option provided by the IDE to view a single web page simultaneously in the editor and browser. At the same time, the live edit feature provided by the IDE enables programmers to see the changes made to the code instantaneously on a web browser. PyCharm further allows developers to avail a JavaScript debugger as well as CoffeeScript and TypeScript editors. It even simplifies isomorphic web application development by supporting both AngularJS and NodeJS.

Support for Popular Python Web Frameworks

In addition to supporting commonly used web technologies, PyCharm also provides first class support for a robust Python web framework like Django. The developers can use the IDE to avail code completion suggestions for Django tags, filters, parameters, and template variables. Also, they can gather additional information about tags and filters by referring to the quick documentation. The Python IDE even helps web developers to debug Django templates, format the code, verify the code, and manage .py consoles. At the same time, PyCharm also supports widely used Python web frameworks like Pyramid and Web2Py. It provides code completion and navigation options specific to Pyramid. Likewise, it allows web developers to avail code completion and navigation options while working with Web2Py.

Support for Python Scientific Libraries

PyCharm further helps programmers to use Python more efficiently in big data and data science projects. It supports some of the widely used scientific libraries for Python—NumPy, Anaconda and Metplotlib. The developers can work efficiently with these scientific libraries by availing the interactive graphs, deep code insight, and array viewers provided by the IDE. They can even run the REPL Python console provided by PyCharm to avail robust features like on the fly syntax check and code inspection. At the same time, the programmers can also integrate the IDE seamlessly with IPython Notebook to create innovative solutions without putting extra time and effort.

Database Tools

In addition to supporting various Python libraries and frameworks, PyCharm allows developers to work with a number of relational databases including Oracle, SQL Server, MySQL and PostgreSQL. The developers can further use the IDE to run queries, edit SQL code, browse data, alter table data, and alter/analyze schemas. PyCharm further supports SQLAlchemy library and inject SQL code into code written in various programming languages. The professional edition of the IDE further makes it easier for developers to handle large volumes of data efficiently through data grids.

Visual Debugger

The visual debugger provided by the IDE helps programmers to debug Python, JavaScript, and Django code. The developers can use the inline debugger to see live debugging data directly on the editor. Likewise, they can debug multiple Python processes simultaneously and step through the code bypassing libraries. PyCharm further creates reusable and customizable configuration for each test script or debugger execution. The users even have option to facilitate remote debugging by integrating the visual debugger with remote interpreters.

Built-in Terminal

PyCharm comes with local terminals for Windows, Linux, and macOS. The built-in terminal enables programmers to continue coding and testing without leaving the IDE. Also, the programmers can use the IDE to run Python files and configure custom Python environments according to precise project requirements. At the same time, they can run interactive Python or Django consoled directly in the IDE. The console provides useful features like code completion, automatic braces matching, and dynamic syntax change. The programmers even have option to integrate the console with both local and remote interpreters.

Support for Major Version Control Systems

PyCharm allows developers to work with widely used version control systems like Git, Mercurial, Perforce and SVN. It even performs complex tasks like adding, removing, and deleting files automatically. The developers even have option to avail a number of features provided by the IDE regardless of their choice of version control system—grouping individual changes into multiple change lists, setting aside the changes to be restored, monitor changes made to the code repository by various users, and check the changes made to the code before being integrated into the local copy.

Software Testing

Like other IDEs, PyCharm also comes with features and tools to simplify Python application testing. It allows developers to perform unit testing through popular Python testing frameworks like Nose, Attest and Doctests. The testers even have option to run individual or multiple test files and test classes. They can further integrate the IDE with Coverage.py to measure code coverage while testing the applications. While testing multi-threaded applications, the testers

can use the thread concurrency visualization option provided by the IDE to control the application fully and efficiently.

CHAPTER IV

SYSTEM TESTING

IV.1 TESTING METHODOLOGIES AND STRATEGIES

Software testing is an integral part of to ensure software quality, some software organizations are reluctant to include testing in their software cycle, because they are afraid of the high cost associated with the software testing .There are several factors that attribute the cost of software testing. Creating and maintaining large number of test cases is a time consuming process. Furthermore, it requires skilled and experienced testers to develop great quality test cases.

Even with the wide availability of automation tools for testing, the degree of automation mostly remains at the automated test script level and generally significant amount of human intervention is required in testing. In addition data collected, as testing is conducted provides a good indication of software quality as a while. The debugging process is the most unpredictable part of testing process. Testing begins at the module level and work towards the integration of entire computer based system. No testing is completed without verification and validation part.

The goal of verification and validation activities are to access and improve the quality of work products generated during the development and modification of the software. Testing plays a vital role in determining the reliability and efficiency of the software and hence is very important stage in software development. Tests are to be conducted on the software to evaluate its performance under a number of conditions. Ideally, it should do so at the level of each module and also when all of them are integrated to form the completed system.

In the project "Online Timber Mill" the testing has been successfully handled with the modules. The test data was given to each and every module in all respect and got the desired output. Each module that has been tested is found working properly.

IV.1.1 Unit Testing

Here we test each module individually and integrated the overall system. Unit testing focuses verification efforts even in the smallest unit of software design in each module. This is known as "module testing". The modules of the "Online Timber Mill" are tested separately. This testing is carried out in the programming style itself. In this testing each module is focused to work satisfactorily as regard to expected output from the module. There are some validation checks for the fields. Unit testing gives stress on the modules of "Online Timber Mill" independently of one another, to find errors. Different modules are tested against the specifications produced during the design of the modules. Unit testing is done to test the working of individual modules with test servers. Program unit is usually small enough that the programmer who developed it can test it in a great detail. Unit testing focuses first on that the modules to locate errors. These error are verified and corrected and so that the unit perfectly fits to the project.

IV.1.2 Integration Testing

Data can be lost across an interface, one module can have an adverse effect on the other sub-functions, when combined they may not perform the desired functions. Integrated testing is the systematic testing to uncover the errors within the interface. This testing is done with simple data and the developed system has run successfully with this simple data. The need for integrated system is to find the overall system performance. The Modules of this project are connected and tested.

After splitting the programs into units, the units were tested together to see the defects between each module and function. It is testing to one or more modules or functions together with the intent of finding interface defects between the modules or functions. Testing completed at as part of unit or functional testing, integration testing can involve putting together of groups of modules and functions with the goal of completing and verifying meets the system requirements.

IV.1.3 system Testing

System testing focuses on testing the system as a whole. System Testing is a crucial step in Quality Management Process. In the Software Development Life Cycle, System Testing is the first level where the System is tested as a whole. The System is tested to verify whether it meets the functional and technical requirements. The application/System is tested in an environment that closely resembles the production environment where the application will be finally deployed.

The perquisites for System Testing are:-
- All the components should have been successfully Unit Tested.
- All the components should have been successfully integrated.

- Testing should be completed in an environment closely resembling the production environment. When necessary iterations of System Testing are done in multiple environments.

IV.1.4 User Acceptance Testing

The system was tested by a small client community to see if the program met the requirements defined the analysis stage. It was fond to be satisfactory. In this phase, the system is fully tested by the client community against the requirements defined in the analysis and design stages, corrections are made as required, and the production system is built. User acceptance of the system is key factor for success of the system.

Test Case

Login form

No.	Test scenario	Expected result	Observed result	result
1.	Enter wrong user name and password	Display login form again with invalid message.	Message displayed.	Pass
2.	Enter correct user name and wrong password	Display login form again with invalid message.	Message displayed.	Pass
3.	Enter correct user name and correct password of admin or mill or user	Admin or user or mill can login to the system.	Login to their pages.	Pass
4.	Press login button without filling the user name and password.	Display warning message to fill the details.	Warning message displayed	Pass

User register

No.	Test scenario	Expected result	Observed result	result
1	Form displayed	Display form with all controls.	Form loaded with all controls.	Pass
2	Click register button without data	Display warning message to fill the details.	Warning message displayed	Pass
3	Click register button with data	Display message saved success fully and clear the fields	Message displayed and all fields cleared	Pass
4	Click cancel button	Exit the form	Exited	Pass

CHAPTER V

SYSTEM IMPLEMENTATION

The implementation is one phase of software development. Implementation is that stage in the project where theoretical design is turned into working system. Implementation involves placing the complete and tested software system into actual work environment. Implementation is concerned with translating design specification with source code. The primary goal of implementation is to write the source code to its specification that can be achieved by making the source code clear and straight forward as possible. Implementation means the process of converting a new or revised system design into operational one. The three types of implementation are:-implementation of a computerized system to replace a manual system, implementation of a new system to replace existing one and implementation of a modified system to replace an existing one.

The implementation is the final stage and it is an important phase. It involves the individual programming; system testing, user training, and the operational running of developed proposed system that constitute the application subsystem. The implementation phase of the software development is concerned with translating design specification in the source code. The user tests the developed system and the changes are according to the needs. Before implementation, several tests have been conducted to ensure no errors encountered during the operation. The implementation phase ends with an evaluation of the system after placing it into operation of time. The validity and proper functionality of all the modules of the developed application is assured during the process of implementation. Implementation is the process of assuring that the information system is operational and then allowing user to take over its operation for use and evaluation. Implementation is the stage in the project where the theoretical design is turned into a working system. The implementation phase constructs, installs and operated the new system. The most crucial stage in achieving a new successful system is that it works effectively and efficiently.

CHAPTER VI

CONCLUSION

This project is an initial proposal to show that this kind of information system is a real benefit to peoples to sell and buy their woods in an effective manner and its also helps to identify the price of woods for peoples. This system can be used in future by manipulating backup servers in each research Institute for efficient on line service.

This system is web-based application system with several features. All the suggestions forwarded in the software proposal have successfully been completed and the final threshold of the application has been crossed. The system has been designed in such a way that it can be modified with very little effort when such a need arises in the future.

This system is a web-based application that works on any available platform. The system is found to be work efficiently and effectively. New features can be added with slight modification of the software, which make it easy to expand the scope of the system. When we compare the proposed system with existing system, we found that the proposed system is more reliable, flexible and secure in comparison with existing system. Also, the system is capable for handling the problems. It also reduces the cost and saves the time.

REFERENCES

1. https://docs.python.org/3/

2. https://docs.djangoproject.com/en/2.2/

3. https://spacy.io/api/doc

4. https://xhtml2pdf.readthedocs.io/en/latest/

5. https://www.w3schools.com/

6. https://simpleisbetterthancomplex.com/

7. https://stackoverflow.co

APPENDIX A

APPENDICES

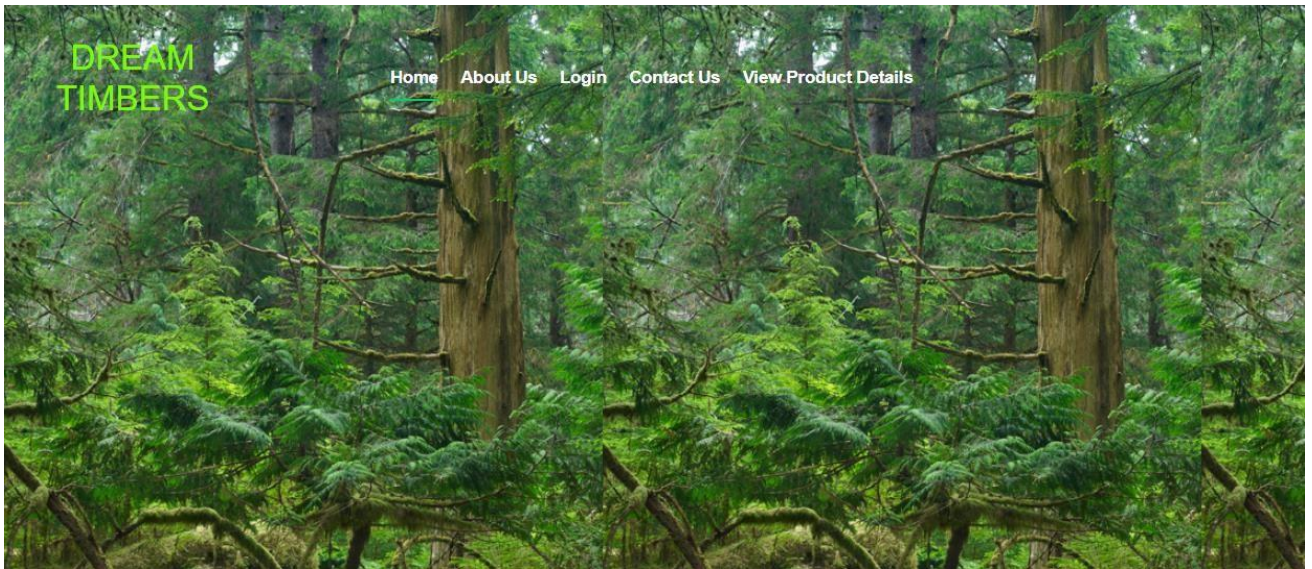## A.1 SCREEN SHOTS INPUT FORM, OUTPUT FORMS

A.1   SCREEN SHOTS

Figure A.1: Home Page
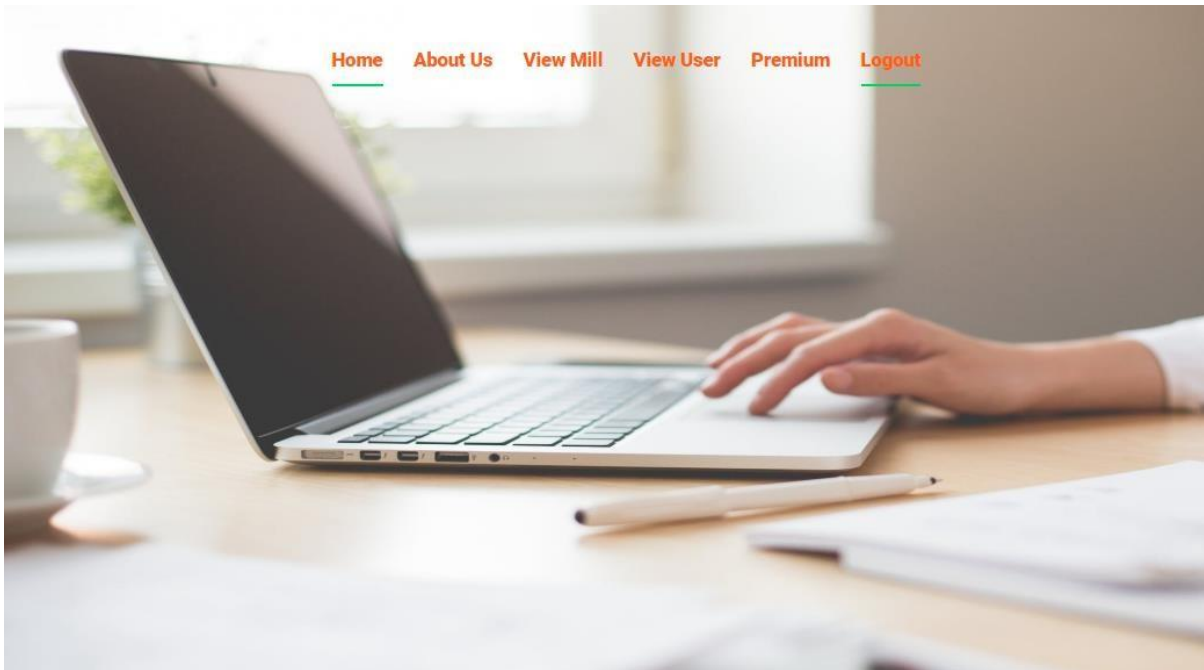
Figure A.2: Admin Home Page
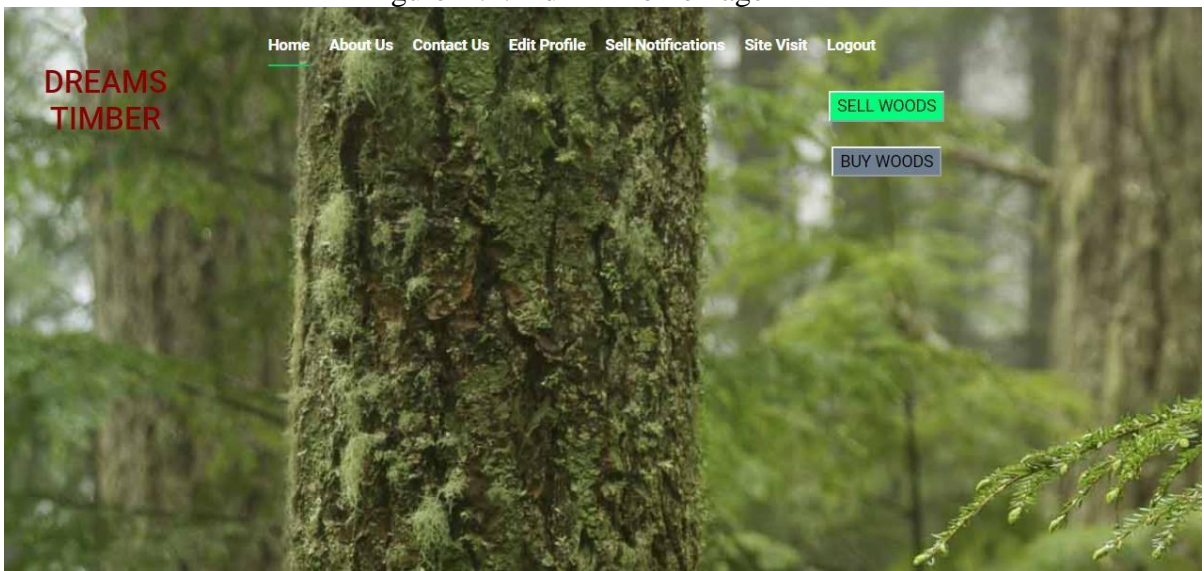

Figure A.3: User Home

Figure A.4: Mill Home Page

Figure A.5: User Add details

Figure A.6: Membership

Figure A.7: Premium payment

Figure A.8: Mill Add details

Figure A.9 Sitevisit details

61

Figure A.10: Booking

A.2 SAMPLE CODE

views.py in interactive app

```python
from django.shortcuts import render

from django.contrib import messages

from django.http import HttpResponse

from django.shortcuts import render, redirect

from django.contrib.auth.models import User, auth

from .models import mill,User1,sell,usell,timber,bank,bkdetails,usdetail


def home(request):

    return render(request, 'index.html')

def login1(request):

    return render(request, 'login1.html')

def login(request):


def user_sign(request):

    return render(request, 'usersign.html')


def usersign(request):

    return render(request, 'user_sign1.html')


def loginuser(request):


        uname =request.POST['email']

        pswd = request.POST['pass']
```

```python
        if uname=='anjana' and pswd=='1234':

            return redirect('adminhome')

        else:


            user = auth.authenticate(username=uname, password=pswd)

            if user is not None:

                auth.login(request, user)

                if User.objects.filter(username=uname,last_name="mill").exists():

                    return render(request, 'homemill.html')

                else:

                    return render(request, 'homeuser.html')

            else:

                messages.info(request, 'Invalid username/password')

            return render(request, 'login.html')

    else:

        return render(request, 'login.html')



def homeuser(request):

    if request.method == 'POST':

        fname = request.POST['fname']

        lname = request.POST['lname']

        username = request.POST['email']

        email = request.POST['email']

        mob = request.POST['mob']

        loc = request.POST['loc']

        pass1 = request.POST['pass1']
```

```python
        pass2 = request.POST['pass2']


        if(pass1 == pass2):
            if User.objects.filter(username=username).exists():

                messages.info(request, 'username taken')

                return render(request, 'user_sign1.html')

            else:

                user = User.objects.create_user(username=username, password=pass1
, email=email, first_name=fname, last_name=lname)

                user.save()

                myuser = User1(fname=fname,lname=lname,email=email,mob=mob,loc=lo
c)

                myuser.save()

                return render(request, 'homeuser.html')

        else:

            messages.info(request, 'Password Missmatch')

            return render(request, 'user_sign1.html')

            #return render(request, 'homeuser.html')

    else:

        return render(request, 'user_sign1.html')


def mill_reg(request):
    if request.method == 'POST':

        mill_name = request.POST['millname']

        username = request.POST['email']

        email = request.POST['email']
```

65

```python
        password1 = request.POST['pass1']

        password2 = request.POST['pass2']

        mill_loc = request.POST['millloc']

        mob = request.POST['millmob']

        if(password1 == password2):

            if User.objects.filter(username=username).exists():

                messages.info(request, 'username taken')

                return render(request, 'mill_sign.html')

            user = User.objects.create_user(username=username,password=password1,
email=email,first_name=mill_name,last_name="mill")

            user.save()

            mill1=mill(muname=email,mill_name=mill_name,mill_loc=mill_loc,mob=mob
,email=email )

            mill1.save()

            return render(request, 'login.html')


        else:

            messages.info(request,'Password Missmatch')

            return render(request, 'mill_sign.html')



def selectmill(request):

    u = User.objects.filter(last_name="mill")

    return render(request, 'selectmill.html',{'u':u})

def adddetails(request,username):

    u=mill.objects.get(email=username)

    return render(request, 'adddetails.html',{'u':u})
```

66

```python
def sell_mill(request):
    if request.method == 'POST':


        uname = request.POST['uname']

        name=request.POST['name']

        wid=request.POST['wid']

        type_wd = request.POST['t1']

        length = request.POST['t2']

        width = request.POST['t3']

        rate = request.POST['t4']
    msell=sell(user="null",usname="null",uname=uname,m_name=name,wood_id=wid,type
_of_wood= type_wd,length=length,width=width,rate=rate,status="not booked")
    msell.save()
    return render(request, 'adddetails.html')


def mill1(request):
    if request.method == 'POST':
        mill = request.POST['mill']
        loc = request.POST['loc']


        m = mill.objects.filter(uname=mill)
        return render(request, 'mill1.html',{'m': m})


    else:
        return render(request,'selectmill.html')
```

67

```python
def mill_list(request):

    m = mill.objects.filter()

    return render(request,'mill_list.html',{'m':m})


def millhome(request,muname):

    m = mill.objects.filter(uname=muname)

    return render(request,'mill1.html',{'m': m})

def expression(request):

    a=int(request.GET['t2'])

    b=int(request.GET['t3'])

    c=b*b*a/2304

    return render(request,'rate.html',{'result':c})

def rate(request):

     return render(request, 'rate.html')


def usersell1(request):

    return render(request, 'usersell1.html')

def ursell(request):

    if request.method == 'POST':

        uname = request.POST['u1']

        umail = request.POST['u2']

        type_wd = request.POST['s2']

        length = request.POST['t2']

        width = request.POST['t3']

        rate = request.POST['t14']
```

```python
        usersell=usell(mill="null",mname="null",user=uname,umail=umail,wood=type_wd,l
ength=length,width=width,rate=rate,status="Not booked")
        usersell.save()
        return render(request, 'usersell1.html')


def uname(request):
        srv=sell.objects.filter( status="mill")
        return render(request, 'userbuy.html',{'srv':srv})
def sell_user(request):
        if request.method == 'POST':


            type_wd = request.POST['t1']
            length = request.POST['t2']
            width = request.POST['t3']
            rate = request.POST['t4']
        usell=sell(type_of_wood= type_wd,length=length,width=width,rate=rate,status="u
ser")
        usell.save()
        return render(request, 'usersell.html')


def mill_sign(request):
        return render(request, 'mill_sign.html')
def homemill(request):
        return render(request, 'homemill.html')
def userhome(request):
        return render(request, 'homeuser.html')
```

```python
def elements(request):

    return render(request, 'elements.html')

def elements1(request):

    return render(request, 'elements1.html')


def contact1(request):

    return render(request, 'contact1.html')

def request(request):

    return render(request, 'request.html')

def index(request):

    return render(request, 'index.html')

def contact(request):

    return render(request, 'contact.html')

def contact11(request):

    return render(request, 'contact11.html')

def contactmill(request):

    return render(request, 'contactmill.html')

def elements11(request):

    return render(request, 'elements11.html')

def aboutmill1(request):

    return render(request, 'aboutmill1.html')

def editprofile(request,muname):

    m = mill.objects.filter(muname=muname)

    return render(request, 'editprofile.html',{'m':m})
```

```python
def saveprofile(request,muname):

    if request.method == 'POST':

        mname = request.POST['t1']

        loc = request.POST['t2']

        mmob = request.POST['t3']

        memail = request.POST['t4']

        m = mill.objects.get(muname = muname)

        m.mill_name = mname

        m.save()

        m.mill_loc = loc

        m.save()

        m.mob = mmob

        m.save()

        m.email =memail

        m.save()

        return redirect('editprofile',muname)


    else:

        return redirect('editprofile',muname)


def view(request):

    m=sell.objects.filter()

    return render(request,'one.html',{'m':m})

def view1(request):

    m=sell.objects.filter()

    return render(request,'one1.html',{'m':m})
```

71

```python
def newfun(request,id):

    s=sell.objects.get(id=id)

def status1(request,id):

    if request.method=='POST':

        sta=request.POST['status']

        u=request.POST['email']

        v=request.POST['name']

        s=sell.objects.get(id=id)

        s.status=sta

        s.save()

        s.user=u

        s.save()

        s.usname=v

        s.save()


        messages.info(request,'success')

        return redirect('newfun',id)


    else:

        return render(request,'status.html')


def notification(request):

    no = usell.objects.filter()

    return render(request, 'noti.html',{'no':no})

def notification1(request):

    no = usell.objects.filter()
```

```python
        return render(request, 'noti1.html',{'no':no})
def notific(request):

    no = usell.objects.filter()

    return render(request, 'notifi.html',{'no':no})
def millfun(request,id):

    s = usell.objects.get(id=id)

    return render(request, 'mstatus.html',{'s': s})
def mstatus(request,id):

    if request.method=='POST':

        sta=request.POST['status']

        u=request.POST['email']

        v=request.POST['name']

        s=usell.objects.get(id=id)

        s.status=sta

        s.save()

        s.mill=u

        s.save()

        s.mname=v

        s.save()

        messages.info(request,'success')

        return redirect('millfun',id)

    else:

        return render(request,'mstatus.html')
def payment(request):

    # s=sell.objects.get(id=id)

    return render(request, 'pay.html')
```

```python
def membership(request):

    a=bank.objects.filter()

    return render(request, 'membership.html',{'a':a})

def membership2(request):

    a=bank.objects.filter()

    return render(request, 'membership2.html',{'a':a})

def credit(request):

    return render(request, 'credit.html')

def debit(request):

    return render(request, 'debit.html')

def nbank(request):

    return render(request, 'nbank.html')

def paypal(request):

    return render(request, 'paypal.html')

def product(request):

    m=sell.objects.filter()

    return render(request,'product.html',{'m':m})

def adminhome(request):

    return render(request, 'adminhome.html')

def mview(request):

    srv=User.objects.filter(last_name="mill")

    return render(request, 'mview.html',{'srv':srv})

def uview(request):

    srv=User1.objects.filter()

    return render(request, 'uview.html',{'srv':srv})

def pview(request):
```

```python
        srv=sell.objects.filter(status="booked")

        return render(request, 'pview.html' ,{'srv':srv})

def premium(request):

    if request.method == 'POST':

        amt = request.POST['amt']

        a=request.POST['amt1']

        amount=bank(amt=amt,status=a)

        amount.save()

    return render(request, 'premium.html')




def notifi(request,uname):

    s = sell.objects.filter(uname=uname,status="booked")

    return render(request, 'notification.html',{'s':s})

def svisit(request,user,id):

    m=sell.objects.get(user=user,id=id)

    return render(request, 'sitevisit.html',{'m':m})

def site(request):

    if request.method == 'POST':

        wid = request.POST['a1']

        user = request.POST['d1']

        umail=request.POST['d2']

        mill=request.POST['d3']

        mmail=request.POST['d4']

        timber=request.POST['d5']

        lenn=request.POST['d6']
```

```python
        cir=request.POST['d7']

        rate=request.POST['d8']

        sdate=request.POST['date22']

        stime=request.POST['d10']

        site1=bkdetails(wo_id=wid,user=user,uemail=umail,mill=mill,memail=mmail,t
imber=timber,leng=lenn,cirf=cir,price=rate,svdate=sdate,svtime=stime)

        site1.save()

    return render(request, 'sitevisit.html')


def usersite(request,name):

    m = bkdetails.objects.filter(uemail=name)

    return render(request, 'usersite.html',{'m':m})



def urnoti(request,uname):

    s = usell.objects.filter(umail=uname,status="booked")

    return render(request, 'urnoti.html',{'s':s})

def ursite(request,u,id):

    m=usell.objects.get(mill=u,id=id)

    return render(request, 'sitevisit1.html',{'m': m})

def site1(request):

    if request.method == 'POST':


        user = request.POST['d3']

        umail=request.POST['d4']

        mill=request.POST['d1']

        mmail=request.POST['d2']
```

```python
            timber=request.POST['d5']

            lenn=request.POST['d6']

            cir=request.POST['d7']

            rate=request.POST['d8']

            sdate=request.POST['date22']

            stime=request.POST['d10']

            sitee=usdetails(user=user,uemail=umail,mill=mill,memail=mmail,timber=timb
er,leng=lenn,cirf=cir,price=rate,svdate=sdate,svtime=stime)

            sitee.save()

    return render(request, 'sitevisit1.html')




def millsite(request,name):

    m = usdetails.objects.filter(memail=name)

    return render(request, 'millsite.html',{'m':m})

def ueditprofile(request,email):

    m = User1.objects.filter(email=email)

    return render(request, 'ueditprofile.html',{'m':m})




def usaveprofile(request,email):

    if request.method == 'POST':

        fname  =  request.POST['t1']

        lname  =  request.POST['t2']

        memail = request.POST['t3']

        mmob = request.POST['t4']
```

```python
        mloc = request.POST['t5']

        m = User1.objects.get(email = email)

        m.fname = fname

        m.save()

        m.lname = lname

        m.save()

        m.email = memail

        m.save()

        m.mob = mmob

        m.save()

        m.loc = mloc

        m.save()

        return redirect('ueditprofile',email)


    else:

        return redirect('ueditprofile',email)




def premiumpay(request):

    if request.method == 'POST':

        ano=request.POST['d1']

        mon=request.POST['d2']

        year=request.POST['d3']

        cvv=request.POST['d4']

        book=timber(cno=ano,month=mon,year=year,cvv=cvv,status="null")

        book.save()
```

```python
    return render(request, 'credit.html')
```