

KUDUMBASHREE PROCESS HANDLING SYSTEM

Project Report Submitted by

Vijayalakshmi PB
Reg. No: LAJC18MCA060

*In Partial fulfilment for the award of
the degree Of*

MASTER OF COMPUTER APPLICATIONS (MCA)
APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY



AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPPALLY

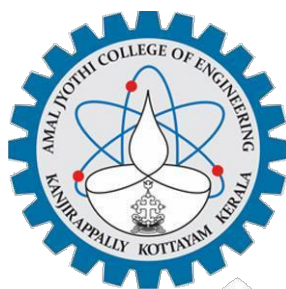
[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE,
Accredited by NAAC with 'A' grade. Koovappally, Kanjirappally, Kottayam, Kerala -686518]

2020-2021

AMAL JYOTHI COLLEGE OF ENGINEERING

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE,
Accredited by NAAC with 'A' grade. Koovappally, Kanjirappally, Kottayam, Kerala -686518]

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS



CERTIFICATE

This is to certify that the project entitled **“KUDUMBASHREE PROCESS HANDLING SYSTEM”** is a bonafide record of the work done by **VIJAYALAKSHMI PB LAJC18MCA060**, during the academic year **2020-2021** carried out under our supervision. It is certified that all corrections/suggestions indicated for assessment have been incorporated in the report. The work report has been approved as it satisfies the academic requirements in respect of the project work prescribed by the university for the Master of Computer Applications Degree. Certified further, that to the best of our knowledge the exact work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this to any other candidate.

Fr. Rubin Thottupurathu Jose
Head of the Department

Sr. Elsin Chakkalackal S.H
Project Coordinator

Ms. Paulin Paul
Project Supervisor

Expert from dept. of Computer Science and Engineering

Amal Jyothi College of Engineering, Kanjirappally

External Expert appointed by the university

DECLARATION

I hereby declare that the project report “**Kudumbashree Process Handling System**” is a bonafide work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Degree of Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2020-2021.

Date.....

Vijayalakshmi PB

KANJIRAPPALLY

Reg. No: LAJC18MCA060

ACKNOWLEDGEMENT

First and foremost, I thank Almighty God for his gracious guidance through the project. I take this opportunity to express my gratitude to all those who have helped me in completing the project successfully

It has been said that gratitude is the memory of the heart. I acknowledge my deep sense of gratitude to our manager **Rev. Fr. Dr. Mathew Paikatt** for providing all the infrastructural facilities for us, our Principal **Dr. Z. V. Lakaparampil** for providing good faculty for guidance.

I take the immense pleasure in expressing my thanks to Head of the Department of Master of Computer Applications, Fr. Rubin Thottupurathu Jose, for his kind patronages in making this project a successful one. I would like to extend my sincere thanks to our coordinator and my project guide Ms. Paulin Paul for her guidance and cooperation, without which this would not have been a success.

I am indebted to my beloved teachers whose cooperation and suggestions throughout the project which helped me a lot. I also thank all my friends and classmates for their interest, dedication and encouragement shown towards the project. I convey hearty thanks to parents for the moral support, suggestion and encouragement to make this venture a success.

Vijayalakshmi P B

ABSTRACT

Kudumbashree is the poverty eradication and women empowerment program implemented by the State Poverty Eradication Mission (SPEM) of the Government of Kerala. Kudumbashree is essentially a community network that covers the entire State of Kerala. It consist of a three tier structure with Neighborhood Groups (NHGs) as primary level units, Area Development Societies (ADS) at the ward level, and Community Development Societies (CDS) at the local government level. It is arguably one of the largest women's networks in the world. While the community network is formed around the central themes of poverty eradication and women empowerment, its main features include democratic leadership, and support structures formed from the 'Kudumbashree family'.

The aim of this project is to develop a software to handle the process of a kudumbashree unit. It is related to Kerala Govt. Each Kudumbashree unit has a set of members. There are different activities conducted by each unit. It include weekly meetings, production and sales of kudumbashree products etc. Govt of Kerala allot grant for each unit. This software is related to one Kudumbashree office. Each unit has a president, secretary and ruling members. It is a desktop application to handle the purchase, stock, damage and sales of Kudumbashree products. The different products are by the Kudumbashree members. Nowadays the Kudumbashree products are sale throw direct home delivery. It is a very difficult task. To overcome the difficulty, we introduce a new shop for Kudumbashree products. Therefore from this shop we can purchase quality products produced by the members of the unit. The purchased products move to the stock. The stock has damage and sales. Any registered customer or the member itself can visit the shop and purchase products. The software mainly concentrates the above process of the organization.

CONTENT

SI. No	Topic	Page No
	Introduction	1
1	Using Git as a version control system	2
1.1	Introduction to GitHub	3
1.2	Working with Git	4
2	Project Documentation	7
2.1	Introduction	8
2.1.1	Project Overview	8
2.1.2	Project Specification	8
2.2	System Study	9
2.2.1	Introduction	9
2.2.2	Existing System	10
2.2.3	Proposed System	11
2.3	Requirement Analysis	14
2.3.1	Feasibility Study	14
2.3.1.1	Economic Feasibility	15
2.3.1.2	Technical Feasibility	15
2.3.1.3	Behavioral Feasibility	16
2.4	Requirement Modeling	16
2.4.1	UML Diagram	16
2.5	System Specification	22
2.5.1	Hardware Specification	22
2.5.2	Software Specification	22
2.6	Software Description	22
2.6.1	Python	22
2.6.2	MySQL	23
2.7.	System Design	25
2.7.1	Architectural Diagram	25
2.7.2	Module Design	26
2.7.3	Data Base Design	27

2.8	System Testing	35
2.8.1	Introduction	35
2.8.2	Test Plan	36
2.8.2.1	Unit Testing	36
2.8.2.2	Integration Testing	37
2.8.2.3	Validation Testing	37
2.8.2.4	User Acceptance	38
2.8.3	Test Case	39
2.9	Implementation	41
2.9.1	Implementation Procedure	41
2.9.2	User Training	42
2.9.3	Operational Document	42
2.9.4	System Maintenance	43
2.10	Conclusion & Future Enhancement	43
2.10.1	Future Enhancement	43
2.10.2	Conclusion	43
2.11	Bibliography	44
2.12	Appendix	45
2.12.1	Sample Code	45
2.12.2	Screen Shots	55

LIST OF ABBREVIATIONS

IDE - Integrated Development Environment

HTML - Hyper Text Markup Language

CSS - Cascading Style Sheet

SQL - Structured Query Language

UML - Unified Modelling Language

INTRODUCTION

Kudumbashree is the poverty eradication and women empowerment program implemented by the State Poverty Eradication Mission (SPEM) of the Government of Kerala. Kudumbashree is essentially a community network that covers the entire State of Kerala. It consist of a three tier structure with Neighborhood Groups (NHGs) as primary level units, Area Development Societies (ADS) at the ward level, and Community Development Societies (CDS) at the local government level. It is arguably one of the largest women's networks in the world. While the community network is formed around the central themes of poverty eradication and women empowerment, its main features include democratic leadership, and support structures formed from the 'Kudumbashree family'. The aim of this project is to develop a software to handle the process of a kudumbasree unit. It is related to Kerala Govt. Each Kudumbashree unit has a set of members. There are different activities conducted by each unit. It include weekly meetings, production and sales of kudumbashree products etc. Govt of Kerala allot grant for each unit. This software is related to one Kudumbashree office. Each unit has a president, secretary and ruling members.

The version control System used during the development time was Git. Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

1. USING GIT AS A VERSION CONTROL SYSTEM

1.1 Introduction to GitHub

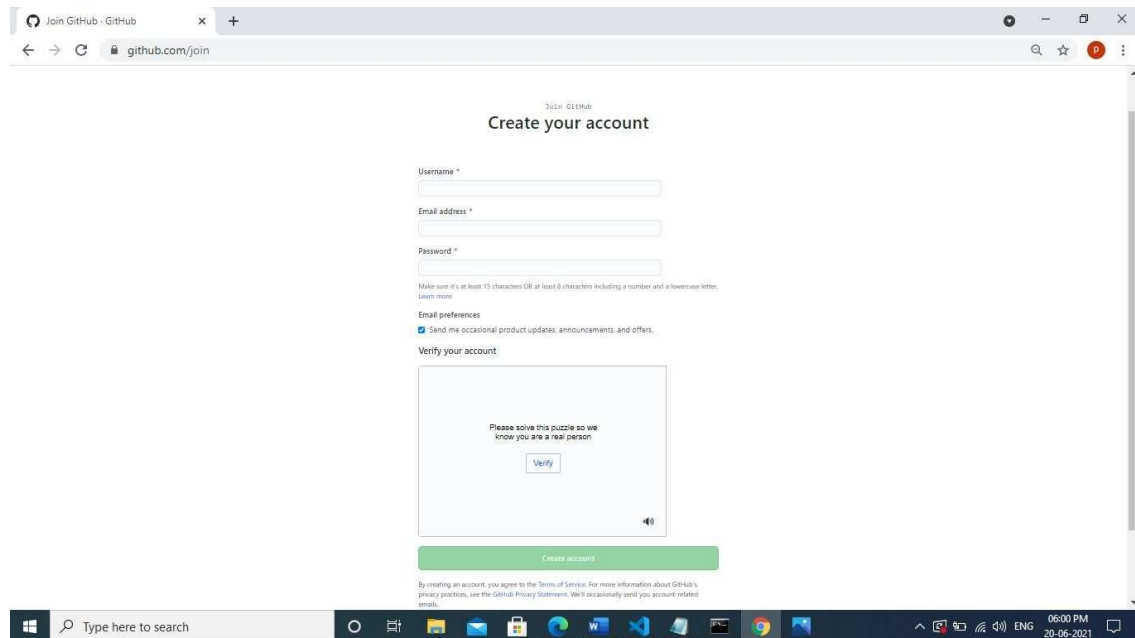
GitHub is a web-based version-control and collaboration platform for software developers. GitHub, which is delivered through a software-as-a-service (SaaS) business model, was started in 2008 and was founded on Git, an open-source code management system created by Linus Torvalds to make software builds faster. And it is used to store the source code for a project and track the complete history of all changes to that code. It allows developers to collaborate on a project more effectively by providing tools for managing possibly conflicting changes from multiple developers. GitHub allows developers to change, adapt and improve software from its public repositories for free, but it charges for private repositories, offering various paid plans. Each public or private repository contains all of a project's files, as well as each file's revision history. Repositories can have multiple collaborators and can be either public or private. GitHub facilitates social coding by providing a web interface to the Git code repository and management tools for collaboration. GitHub can be thought of as a serious social networking site for software developers. Members can follow each other, rate each other's work, receive updates for specific projects and communicate publicly or privately.

GitHub products and features

GitHub offers an on-premises version in addition to the well-known SaaS product. GitHub Enterprise supports integrated development environments and continuous integration tool integration, as well as a litany of third-party apps and services. It offers increased security and auditable than the SaaS version.

1.2 Working with Git

Click on sign up button and create an account in GitHub.com



Join GitHub

Create your account

Username *

Email address *

Password *

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more](#)

Email preferences

☒ Send me occasional product updates, announcements, and offers.

Verify your account

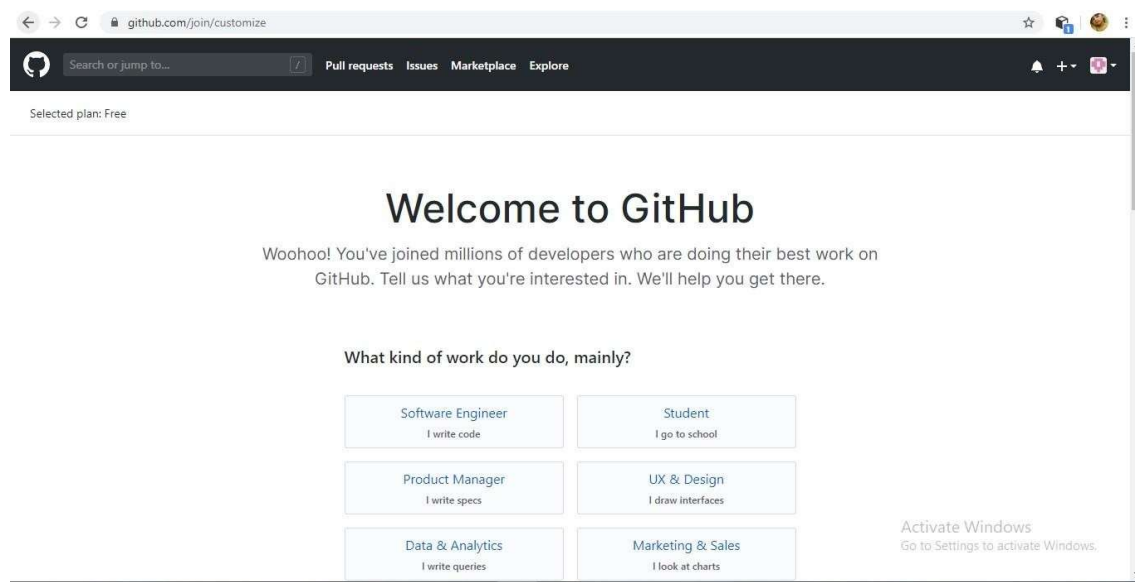
Please solve this puzzle so we know you are a real person

Verify

Create account

By creating an account you agree to the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#). We'll occasionally send you account related emails.

Once successfully sign in, set up personal account, and choose your plan by selecting the options given below based on our purpose.



github.com/join/customize

Search or jump to...

Pull requests Issues Marketplace Explore

Selected plan: Free

Welcome to GitHub

Woohoo! You've joined millions of developers who are doing their best work on GitHub. Tell us what you're interested in. We'll help you get there.

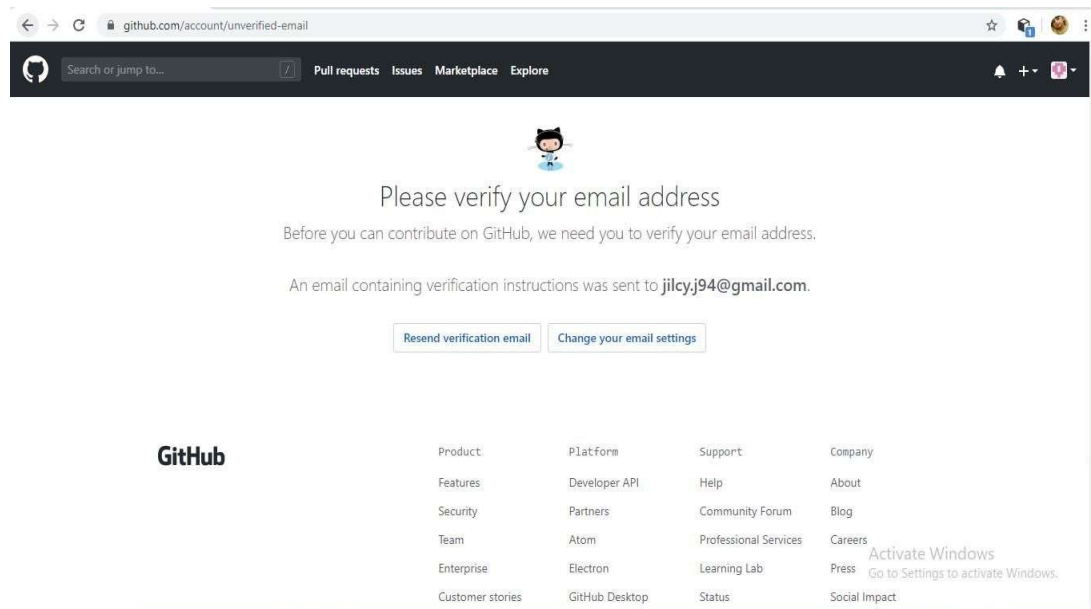
What kind of work do you do, mainly?

Software Engineer I write code	Student I go to school
Product Manager I write specs	UX & Design I draw interfaces
Data & Analytics I write queries	Marketing & Sales I look at charts

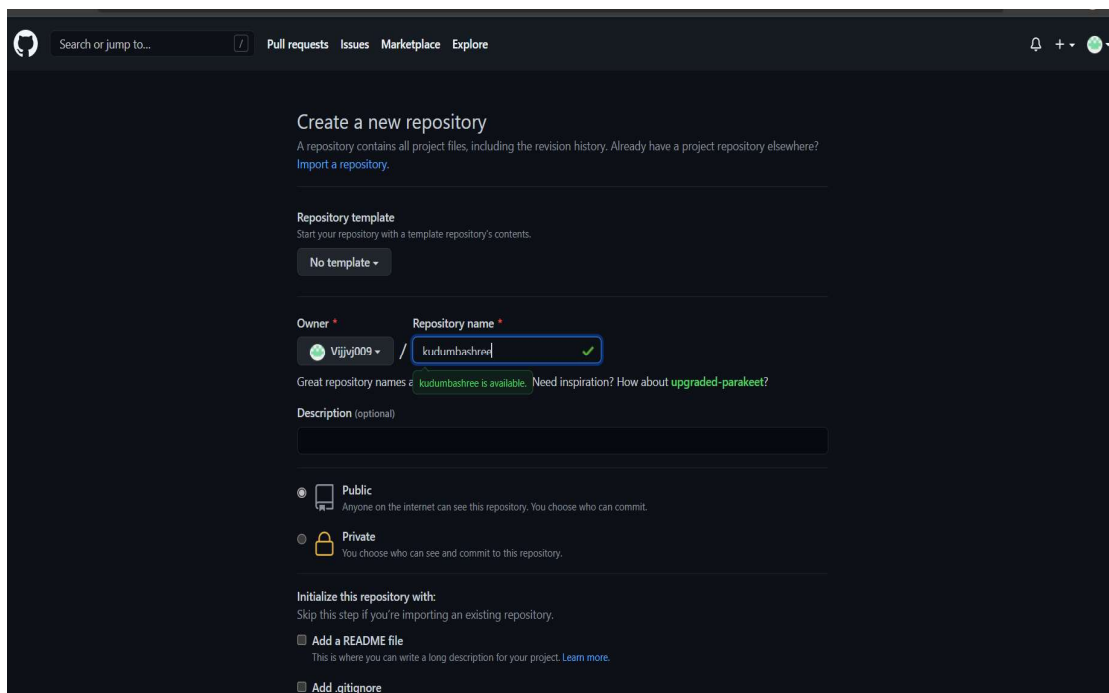
Activate Windows
Go to Settings to activate Windows.

Kudumbashree Process Handling System

Once this step is completed then verify the email address and you can access the home page



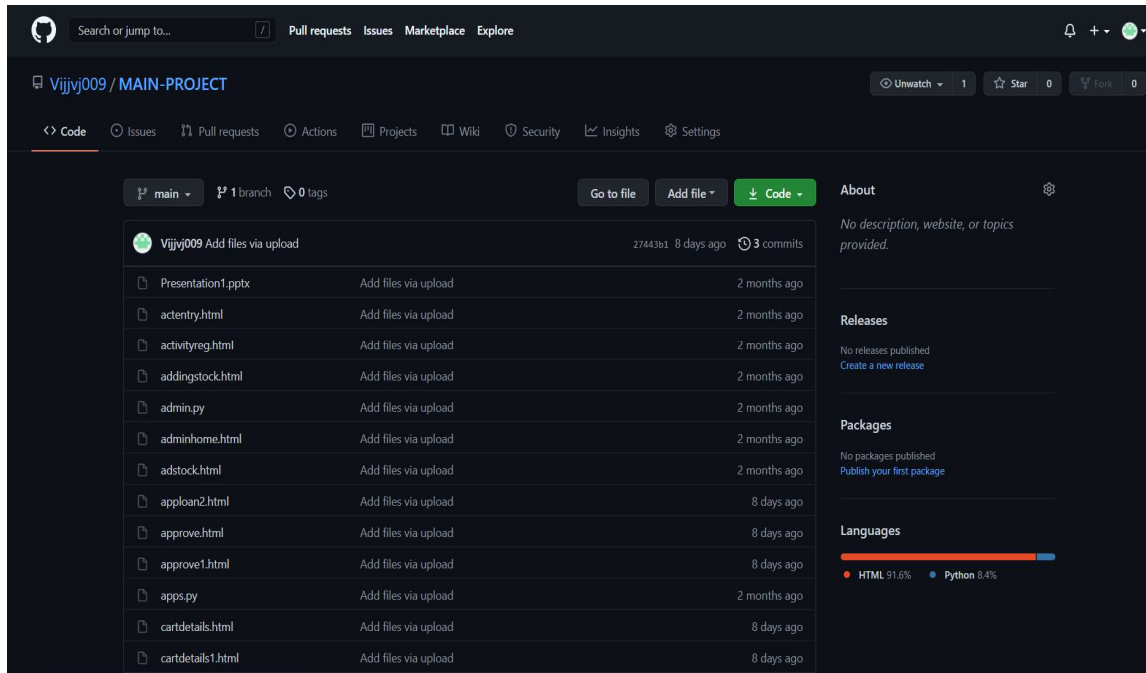
Create a repository in GitHub



Kudumbashree Processs Handling System

Push the local repository to origin

Refresh GitHub.com repository to fetch commits



2. PROJECT DOCUMENTATION

2.1 INTRODUCTION

2.1.1 Project Overview

The aim of this project is to develop a software to handle the process of a Kudumbashree unit. It is related to Kerala Govt. Each Kudumbashree unit has a set of members. There are different activities are conducted by each unit. It includes weekly meeting, production and sales of Kudumbashree products etc.

It is a desktop application to handle the purchase, stock, damage and sales of Kudumbashree products. The different products are by the Kudumbashree members. Nowadays the Kudumbashree products are sale throw direct home delivery. It is a very difficult task. To overcome the difficulty. The purchased products move to the stock. The stock has damage and sales. Any registered customer or the member itself can visit the cite and purchase products. The software mainly concentrates the above process of the organization.

2.1.2 Project Specification

The main objective of this project is to develop a software to handle the overall process of a Kudumbashree unit .The member details, collected grant from Govt., activities, products production and distribution etc. are controlled by the system. In general, this software is very helpful for the Kudumbashree unit. User must has a unique ID and Password. Using that ID and Password that user can login to the site and perform different process for the organization. The modules in the system are: -

- 1)Registration
- 2)Product Management
- 3)Loan Management
- 4)Views and Reports

The aim of this project is to develop a software to handle the process of a single kudumbashree unit. It's a desktop application to handle the purchase, sales, stock of a kudumbashree unit. Both the members and customers can buy the products. Members have an additional functionality of applying for loans, which has to be approved by the admin. Product registration is done by the admin. The main objective of this project is to develop a software to handle the overall process of a Kudumbashree unit. The member details, collected grant from Govt., activities, products production and distribution etc. are controlled by the system. In general, this software is very helpful for the Kudumbashree unit. User must have a unique ID and Password. Using that ID and Password that user can login to the site and perform different process for the organization.

2.2 SYSTEM STUDY

2.2.1 Introduction

System analysis is a process of gathering and interpreting facts, diagnosing problems and the information to recommend improvements on the system. It is a problem-solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is studied to the minute's detail and analysed. The system analyst plays the role of the interrogator and dwells deep into the working of the present system. The system is viewed as a whole and the input to the system are identified. The outputs from the organizations are traced to the various processes. System analysis is concerned with becoming aware of the problem, identifying the relevant and decisional variables, analysing and synthesizing the various factors and determining an optimal or at least a satisfactory solution or program of action. A detailed study of the process must be made by various techniques like interviews, questionnaires etc. The data collected by these sources must be scrutinized to arrive to a conclusion. The conclusion is an understanding of how the system functions. This system is called the existing

system. Now the existing system is subjected to close study and problem areas are identified. The designer now functions as a problem solver and tries to sort out the difficulties that the enterprise faces. The solutions are given as proposals. The proposal is then weighed with the existing system analytically and the best one is selected. The proposal is presented to the user for an endorsement by the user. The proposal is reviewed on user request and suitable changes are made. This is loop that ends as soon as the user is satisfied with proposal. Preliminary study is the process of gathering and interpreting facts, using the information for further studies on the system. Preliminary study is problem solving activity that requires intensive communication between the system users and system developers. It does various feasibility studies. In these studies, a rough figure of the system activities can be obtained, from which the decision about the strategies to be followed for effective system study and analysis can be taken.

2.2.2 EXISTING SYSTEM

Once the existing system has been analyzed thoroughly, alternate solutions are studied and system that satisfies our objectives most cost efficiently is selected as the proposed system. In the physical design of the proposed system detailed specification for every aspect of the system is created. This includes form designers; file organizations, procedural steps and so on. The existing system of this project is manual. At present, there is no software applicable for this. Now all the operations are manual.

The record keeping is also manual. The registrations, updates and calculations are very difficult. Therefore lots of human works are needed for a particular operation or processes. The primary information gathering tool are on-site observation, documentation, interview, and questionnaires. For this the system analyst should develop an alternative solution and evaluate each on terms of cost, benefit and feasibility.

Drawbacks of existing system

The following are its drawbacks

- Time consuming
- Not reliable
- Very slow
- Record keeping is in different books
- Human error may occur
- Less accurate

2.2.3 PROPOSED SYSTEM

The proposed system fully convert the manual system to computerized system. All the process are controlled by the software. It is fully controlled by the server machine. The user can access the system and all process in it very fast. Above all managing a document manually requires a lot of time and effort. If the process is computerized time and human effort can be reduced considerably. There is a security feature built in to the system since all information in the database is confidential and is therefore accessed only by the authorized users the access is given only to a correct password entry. Error handling has been done efficiently. Errors that could happen during data entries are informed to user. It is ensured that only valid data can be entered. Computerization can make the management of documents easier as it would be easy to add a new document, view all documents etc. This is the ultimate aim of the project.

PRODUCT REGISTRATION: In this project it is done by the Admin. Admin registers all the available products on the site, the stock of the products are also managed by the admin. Admin is also responsible for adding the stock to the site. The registered products can be removed and edited by admin, when the product is out of stock admin can add the stock.

MEMBER REGISTRATION: Kudumbashree members can login to the system and they can buy the products and can also apply for loans.

CUSTOMER REGISTRATION: Kudumbashree customers can sign up to the site and then can login to the system and they can buy the products.

PAYMENT: The customers and members are supposed to pay the amount for the purchase of different products from the site. The payment gate way is used to accomplish the task

The users of this project are: -

Customer

- View product details.
- Update profile.
- Purchase products
- Make payments
- Send feedback
- View notifications from admin about products.

Admin

- Add and edit products to the site.
- Accept or reject loan request from the member.
- Add and update stocks of different products.
- Verify the overall sales of each product.
- Add notification about the products.

Member

- Request for loan
- View product details.
- Update profile.
- Request for loan

- Purchase products
- Make payments
- Send feedback
- View notifications from admin about products and loans.

Advantages of proposed system

The system is very simple in design and to implement. The system requires very low system resources and the system will work in almost all configurations. It has got following features:

- **Low Cost-** Low cost is spend due to the reduced manual work, no transportation fee, no paper cost.
- **Better security-** For data to remain secure measures must be taken to prevent unauthorized access. Security means that data are protected from various forms of destruction. The system security problem can be divided into four related issues: security, integrity, privacy and confidentiality. Username and password requirement to sign in ensures security. It will also provide data security as we are using the secured databases for maintaining the documents.
- **Ensure data accuracy-** The proposed system eliminates the manual errors while entering the details of the users during the registration.
- **Get updated information-** The employees get job offers on correct time. So, they don't loss their job opportunities.
- **Accessibility-** The registered employees and mates can login to the site at any time anywhere and can access the updated information properly.
- **No time consuming-** Due to the automated functionality the time required to request site, accept customer job request and list preparation, attendance marking etc are reduced.

2.3 Requirement Analysis

2.3.1 Feasibility study

Feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work, effort and the time that spend on it. Feasibility study lets the developer foresee the future of the project and the usefulness. A feasibility study of a system proposal is according to its workability, which is the impact on the organization, ability to meet their user needs and effective use of resources. Thus, when a new application is proposed it normally goes through a feasibility study before it is approved for development. The document provides the feasibility of the project that is being designed and lists various areas that were considered very carefully during the feasibility study of this project such as Technical, Economic and Operational feasibilities. The following are its features: -

2.3.1.1 Economic Feasibility

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require. The following are some of the important financial questions asked during preliminary investigation:

- The costs conduct a full system investigation
- . • The cost of the hardware and software.
- The benefits in the form of reduced costs or fewer costly errors.

The proposed system is developed as part of project work, there is no manual cost to spend for the proposed system. The requirements needed to this project is collected from the Kudumbashree units and its members who have a proper idea of this project. According to all the calculations the project was developed in a low cost. As it is completely developed using open-source software.

2.3.1.2 Technical Feasibility

The system must be evaluated from the technical point of view first. The assessment of this feasibility must be based on an outline design of the system requirement in the terms of input, output, programs and procedures. Having identified an outline system, the investigation must go on to suggest the type of equipment, required method developing the system, of running the system once it has been designed. Technical issues raised during the investigation are:

- Does the existing technology sufficient for the suggested one?
- Can the system expand if developed?

The project should be developed such that the necessary functions and performance are achieved within the constraints. The project requires High Resolution Scanning device and utilizes Cryptographic techniques. Through the technology may become obsolete after some period of time, due to the fact that newer version of same software supports older versions, the system may still be used. So, there are minimal constraints involved with this project. The system has been developed using HTML,CSS in front end and MySQL in server in back end, the project is technically feasible for development. The system has been developed using PHP in front end and MySQL in server in back end, the project is technically feasible for development. The System used was also of good performance of Processor Intel i3 core; RAM 8GB and, Hard disk 1024B.

2.3.1.3 Behavioral Feasibility

The proposed system includes the following questions:

- Is there sufficient support for the members and customers?
- Will the proposed system cause harm?

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioral aspects are considered carefully and conclude that the project is behaviorally feasible. The GUI is simple so that users can easily use it and it is simple enough so that no training is needed.

2.4 Requirement Modeling

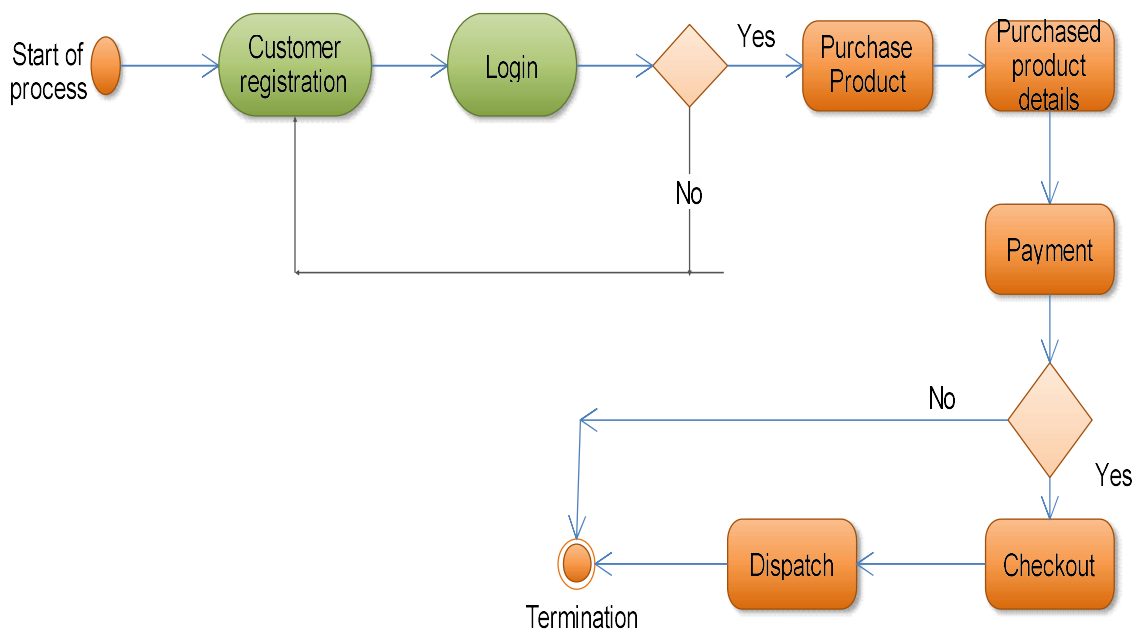
2.4.1 UML diagram

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997. UML stands for Unified Modeling Language. UML is different from the other common programming languages such as C++, Java, COBOL, etc. UML is a pictorial language used to make software blueprints. UML can be described as a general-purpose visual modeling language to visualize, specify, construct, and document software system. Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc. UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object-oriented analysis and design. After some standardization, UML has become an OMG standard. All the elements, relationships are used to make a complete UML diagram and the diagram represents a system. The visual effect of the UML diagram is the most important part of the entire process. All the other elements are used to make it complete. UML includes the following nine diagrams.

- Activity diagram
- Use case diagram
- Sequence diagram
- Collaboration diagram
- State chart diagram
- Class diagram
- Object diagram
- Component diagram
- Deployment diagram

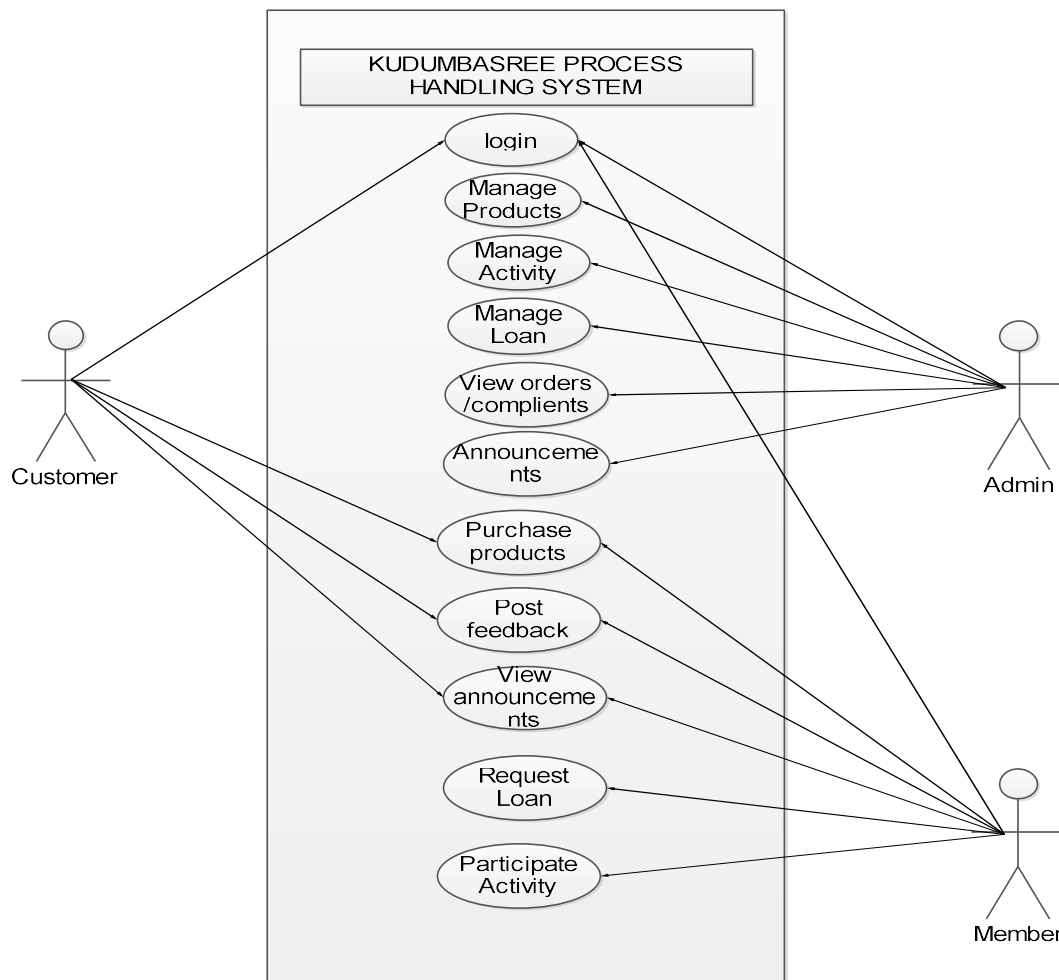
UML Activity Diagram

Activity Diagram describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc. 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.



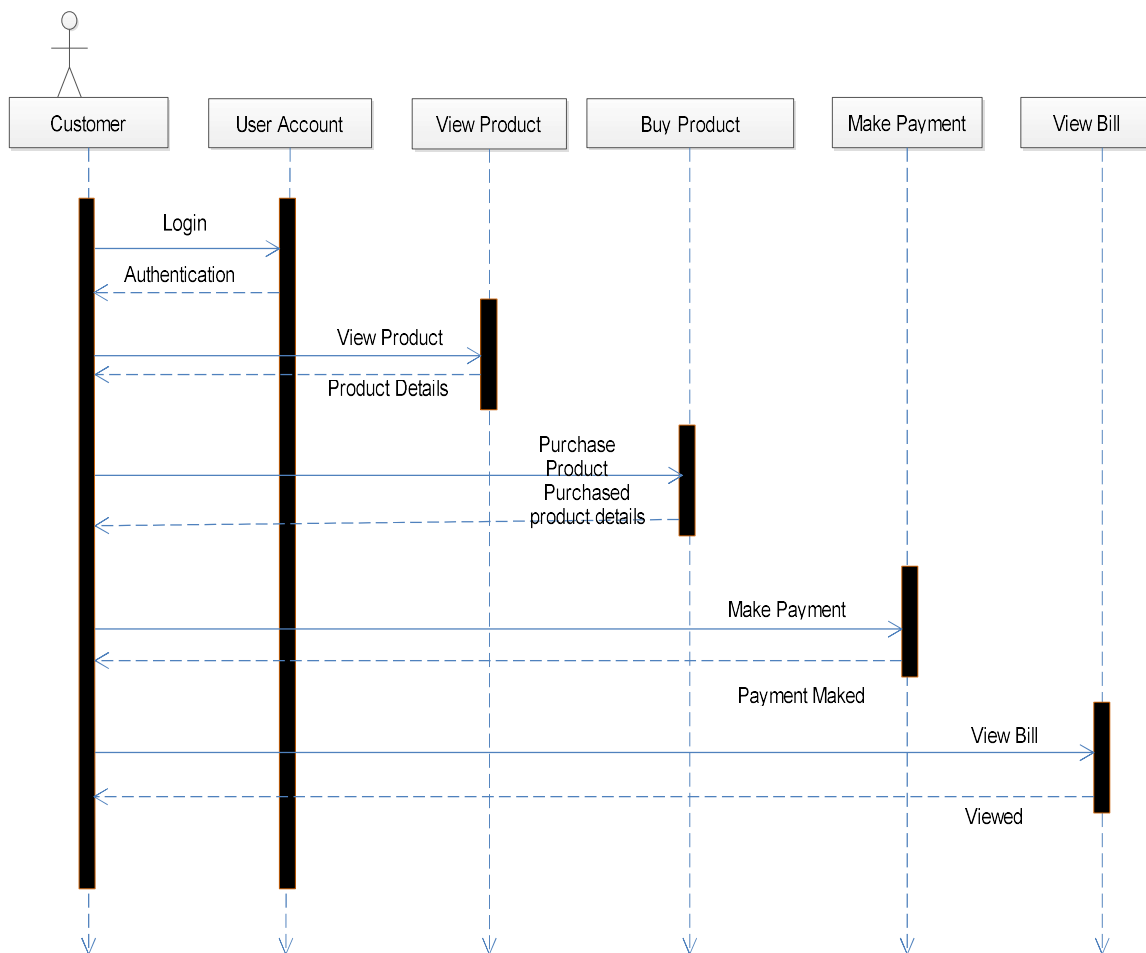
UML Use Case Diagram

A use case diagram is a graphic depiction of the interactions among the elements of a system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. In this context, the term "system" refers to something being developed or operated, such as a mail-order product sales and service Web site. Use case diagrams are employed in UML (Unified Modeling Language), a standard notation for the modeling of real world objects and systems. System objectives can include planning overall requirements, validating a hardware design, testing and debugging a software product under development, creating an online help reference, or performing a consumer-service-oriented task. For example, use cases in a product sales environment would include item ordering, catalogue updating, payment processing, and customer relations.



UML Sequence Diagram

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.



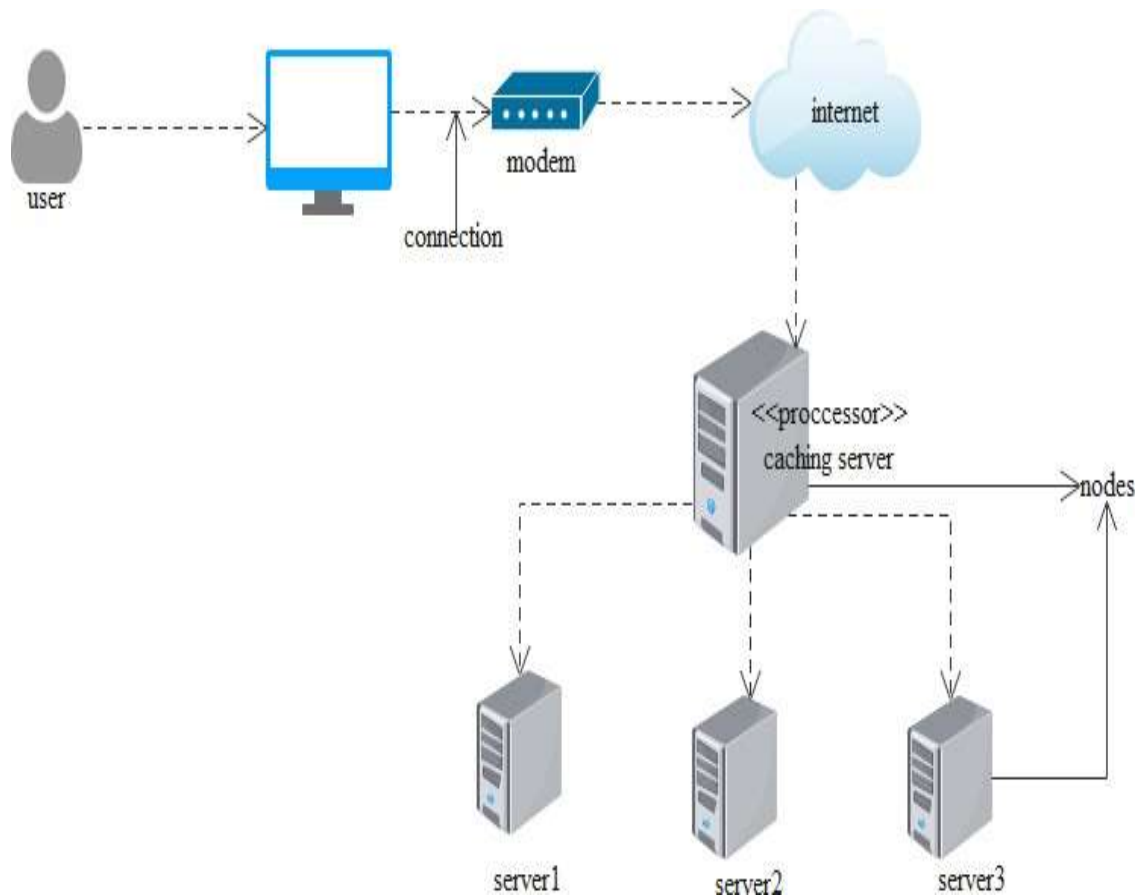
UML Collaboration Diagram

The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system. Notations of a Collaboration Diagram Following are the components of a component diagram that are enlisted below:

- **Objects:** The representation of an object is done by an object symbol with its name and class underlined, separated by a colon. In the collaboration diagram, objects are utilized in the following ways:
 - The object is represented by specifying their name and class.
 - It is not mandatory for every class to appear.
 - A class may constitute more than one object.
 - In the collaboration diagram, firstly, the object is created, and then its class is specified.
 - To differentiate one object from another object, it is necessary to name them.
- **Actors:** In the collaboration diagram, the actor plays the main role as it invokes the interaction. Each actor has its respective role and name. In this, one actor initiates the use case.
- **Links:** The link is an instance of association, which associates the objects and actors. It portrays a relationship between the objects through which the messages are sent. It is represented by a solid line. The link helps an object to connect with or navigate to another object, such that the message flows are attached to links.
- **Messages:** It is a communication between objects which carries information and includes a sequence number, so that the activity may take place. It is represented by a labelled arrow, which is placed near a link.

UML Deployment Diagram

Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed. Deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships. It ascertains how software is deployed on the hardware. It maps the software architecture created in design to the physical system architecture, where the software will be executed as a node. Since it involves many nodes, the relationship is shown by utilizing communication paths.



2.5 System specification

2.5.1 Hardware Specification

Processor - Intel core i3

RAM - 8 GB

Hard disk – 1024 GB

2.5.2 Software Specification

Front End - HTML, CSS

Backend - MySQL

Client on PC - Windows 7 and above.

Technologies used - JS, HTML5, jQuery, Python, CSS, Django

2.6 Software description

2.6.1 Python

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

does not run unmodified on Python 3. Python 2 was discontinued with version 2.7.18 in 2020. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse.

2.6.2 MySQL

MySQL, the most popular Open-Source SQL database management system, is developed, distributed, and supported by Oracle Corporation. The MySQL Web site provides the latest information about MySQL software.

MySQL is a database management system

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

MySQL databases are relational

A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed. The logical model, with objects such as databases, tables, views, rows, and columns, offers a flexible programming environment. You set up rules governing the relationships between different data fields, such as one-to-one, one-to-many, unique, required or optional, and “pointers” between different tables. The database enforces these rules, so that with a well-designed database, your application never sees inconsistent, duplicate, orphan, out-of-date, or missing data. The SQL part of “MySQL” stands for “Structured Query Language”. SQL is the most common standardized language used to access databases. Depending on your

programming environment, you might enter SQL directly (for example, to generate reports), embed SQL statements into code written in another language, or use a language-specific API that hides the SQL syntax. SQL is defined by the ANSI/ISO SQL Standard. The SQL standard has been evolving since 1986 and several versions exist. In this manual, “SQL92” refers to the standard released in 1992, “SQL: 1999” refers to the standard released in 1999, and “SQL: 2003” refers to the current version of the standard. We use the phrase “the SQL standard” to mean the current version of the SQL Standard at any time.

MySQL software is Open Source

Open Source means that it is possible for anyone to use and modify the software. Anybody can download the MySQL software from the Internet and use it without paying anything. If you wish, you may study the source code and change it to suit your needs. The MySQL software uses the GPL (GNU General Public License), to define what you may and may not do with the software in different situations. If you feel uncomfortable with the GPL or need to embed MySQL code into a commercial application, you can buy a commercially licensed version from us. See the MySQL Licensing Overview for more information.

The MySQL Database Server is very fast, reliable, scalable, and easy to use.

If that is what you are looking for, you should give it a try. MySQL Server can run comfortably on a desktop or laptop, alongside your other applications, web servers, and so on, requiring little or no attention. If you dedicate an entire machine to MySQL, you can adjust the settings to take advantage of all the memory, CPU power, and I/O capacity available.

MySQL Server works in client/server or embedded systems.

The MySQL Database Software is a client/server system that consists of a multi-threaded SQL server that supports different back ends, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces (APIs). We also provide MySQL Server as an embedded multi-threaded library that you can link into your application to get a smaller, faster, easier-to-manage standalone product.

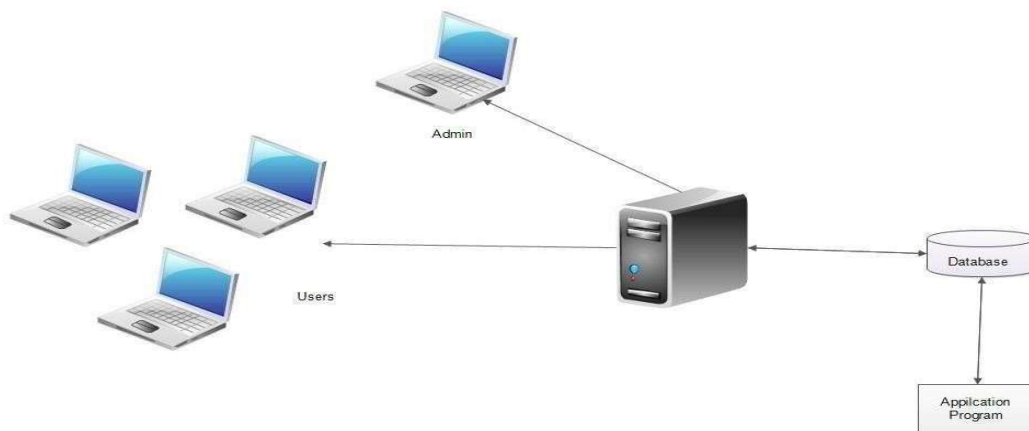
A large amount of contributed MySQL software is available.

MySQL Server has a practical set of features developed in close cooperation with our users. It is very likely that your favourite application or language supports the MySQL Database Server.

2.7 System design

Design is the first step into the development phase for any engineered product or system. Design is a creative process. A good design is the key to effective system. The term “design” is defined as “the process of applying various techniques and principles for the purpose of defining a process or a system in sufficient detail to permit its physical realization”. It may be defined as a process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permit its physical realization. Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm that is used. The system design develops the architectural detail required to build a system or product. As in the case of any systematic approach, this software too has undergone the best possible design phase fine tuning all efficiency, performance and accuracy levels. The design phase is a transition from a user-oriented document to a document to the programmers or database personnel. System design goes through two phases of development: Logical and Physical Design.

2.7.1 Architectural Design



The registered landowner, mate, employee and admin can access the website through internet using their Laptop, Smart Phone, Tablet or Desktop Computer. The System's application program processes the user's request and provides the required services by taking data from the system database.

2.7.2 Module Design

Admin Module

The administrator of the Kudumbashree unit is allowed to access all the services in the system. Admin has the overall control of the system. Admin can add or update products, notifications about loan and products etc. Admin can View all the registered members, customers, can able to approve or reject loan, mate and also can able to view all loan request from members, etc

Manage member details, Add and update new products, notifications about availability of products.	Approve/Reject loan request from the members
Add/Update stock of all products.	View the list of sold products

Registered Member Module

Member can register and login to this site. They can view list of available products. Member can apply for loan.

Register or login	View list of Products
Apply for Loan	Accept/Reject loan request
Payment for the purchased goods	View notifications

Registered Customer Module

The Customer can register and login to this site. They can view the list of products, notifications. Customer can also update the profile.

Register and login	View list of Products
Update / EditProfile	Payment for the purchased goods
Send feedback	View notifications

2.7.3 Database Design

A database is an organized mechanism that has the capability of storing information through which a user can retrieve stored information in an effective and efficient manner. The data is the purpose of any database and must be protected. The database design is a two-level process. In the first step, user requirements are gathered together and a database is designed which will meet these requirements as clearly as possible. This step is called Information Level Design and it is taken independent of any individual DBMS. In the second step, this Information level design is transferred into a design for the specific DBMS that will be used to implement the system in question. This step is called Physical Level Design, concerned with the characteristics of the specific DBMS that will be used. A database design runs parallel with the system design. The organization of the data in the database is aimed to achieve the following two major objectives.

- Data Integrity
- Data independence

Relational Database Management System (RDBMS)

A relational model represents the database as a collection of relations. Each relation resembles a table of values or file of records. In formal relational model terminology, a row is called a tuple, a column header is called an attribute and the table is called a relation. A relational database consists of a collection of tables, each of which is assigned a unique name. A row in a table represents a set of related values.

Relations, Domains & Attributes

A table is a relation. The rows in a table are called tuples. A tuple is an ordered set of n elements. Columns are referred to as attributes. Relationships have been set between every table in the database. This ensures both Referential and Entity Relationship Integrity. A domain D is a set of atomic values. A common method of specifying a domain is to specify a data type from which the data values forming the domain are drawn. It is also useful to specify a name for the domain to help in interpreting its values. Every value in a relation is atomic, that is not decomposable.

Relationships

- Table relationships are established using Key. The two main keys of prime importance are Primary Key & Foreign Key. Entity Integrity and Referential Integrity Relationships can be established with these keys.
- Entity Integrity enforces that no Primary Key can have null values.
- Referential Integrity enforces that no Primary Key can have null values.
- Referential Integrity for each distinct Foreign Key value, there must exist a matching Primary Key value in the same domain. Other key are Super Keys and Candidate Keys.

Normalization

Data are grouped together in the simplest way so that later changes can be made with minimum impact on data structures. Normalization is formal process of data structures in manners that eliminates redundancy and promotes integrity. Normalization is a technique of separating redundant fields and breaking up a large table into a smaller one. It is also used to avoid insertion, deletion, and updating anomalies. Normal form in data modelling use two concepts, keys and relationships. A key uniquely identifies a row in a table. There are two types of keys, primary key and foreign key. A primary key is an element or a combination of elements in a table whose purpose is to identify records from the same table. A foreign key is a column in a table that uniquely identifies record from a different table. All the tables have been normalized up to the third normal form. As the name implies, it denotes putting things in the normal form. The application developer via normalization tries to achieve a sensible organization of data into proper tables and columns and where names can be easily correlated to the data by the user.

First Normal Form

The First Normal Form states that the domain of an attribute must include only atomic values and that the value of any attribute in a tuple must be a single value from the domain of that attribute. In other words, 1NF disallows “relations within relations” or “relations as attribute values within tuples”. The only attribute values permitted by 1NF are single atomic or indivisible values. The first step is to put the data into First Normal Form. This can be done by moving data into separate tables where the data is of similar type in each table. Each table is given a Primary Key or Foreign Key as per requirement of the project. In this we form new relations for each nonatomic attribute or nested relation. This eliminated repeating groups of data. A relation is said to be in first normal form if only if it satisfies the constraints that contain the primary key only.

Second Normal Form

According to Second Normal Form, for relations where primary key contains multiple attributes, no non-key attribute should be functionally dependent on a part of the primary key. In this we decompose and setup a new relation for each partial key with its dependent attributes. Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it. This step helps in taking out data that is only dependent on a part of the key. A relation is said to be in second normal form if and only if it satisfies all the first normal form conditions for the primary key and every non-primary key attribute of the relation is fully dependent on its primary key alone.

Third Normal Form

According to Third Normal Form, Relation should not have a non-key attribute functionally determined by another non-key attribute or by a set of non-key attributes. That is, there should be no transitive dependency on the primary key. In this we decompose and set up relation that includes the non-key attributes that functionally determines other non-key attributes. This step is taken to get rid of anything that does not depend entirely on the Primary Key. A relation is said to be in third normal form if only if it is in second normal form and more over the non key attributes of the relation should not be depend on another non-key attributes.

TABLES**Table 1:-Login**

This table contain login details

SI No	Field Name	Type	Constraints	Description
1	Username	Numeric(10,0)	Primary Key	User id
2	Password	Varchar(15)	Not null	Password of the user
3	Status	Numeric(2)	Not null	Type of user

Table 2:-Member

This table contain member details

SI No	Field Name	Type	Constraints	Description
1	Mid	Numeric(10,0)	Primary key	Member id
2	Fname	Varchar(20)	Not null	First name of Member
3	Lname	Varchar(20)	Not null	Last name of member
4	Adharno	Numeric(20,0)	Not null	Adhar no
5	Photo	Varchar(255)	Not null	Photo of the mamber
6	Hname	Varchar(20)	Not null	House name
7	Place	Varchar(20)	Not null	Place of member
8	Pin	Numeric(6,0)	Not null	Pin code
9	Ph	Numeric(10,0)	Not null	Phone number
10	Rel	Varchar(10)	Not null	Religion of the member
11	Caste	Varchar(10,0)	Not null	Caste of the member
12	Dob	Date	Not null	Date of birth
13	Doj	Date	Not null	Date of join
14	Email	Varchar(20)	Not null	Email id of member
15	Status	Varchar(2)	Not null	Status of member

Table 3:-Customer

This table contain customer details

SI No	Field Name	Type	Constraints	Description
1	Custid	Numeric(5,0)	Primary key	Customer id
2	Cname	Varchar(20)	Not null	Customer name
3	Email	Varchar(50)	Null	Email id of customer
4	Ph	Varchar(50)	Not null	Phone no
5	Address	Varchar(100)	Not null	Address of customer
6	City	Varchar(30)	Not Null	City
7	Disrtict	Varchar(20)	Not null	District
8	Pin	Numeric(8,0)	Not null	Pin
9	Password	Varchar(20)	Not null	Password
10	Confirm password	Varchar(20)	Not null	Confirm

Table4:-Product

This table contain Product details

SI No	Field name	Type	Constraints	Description
1	Pname	Varchar(20)	Primary key	Product name
2	Srate	Numeric(10,2)	Not null	Sale rate
3	Image	Varchar(250)	Not null	Image of the product
4	Ptype	Varchar(20)	Not null	Product type
5	Unit	Varchar(10)	Not null	Unit
6	Description	Varchar(100)	Not null	Description of the product

Table5:-Pstockentry

This table contain stock entry details

SI No	Field name	Type	Constraints	Description
1	Pname	Varchar(30)	Primary key/Foreign key	Product name
2	Qty	Numeric(5,0)	Not null	Quantity
3	Edate	Date	Not null	Entry date

Table6:-Pstock

This table contain stock details

SI No	Field name	Type	Constraints	Description
1	Pname	Varchar(30)	Primary key/Foreign key	Product name
2	Cstock	Varchar(30)	Not null	Current stock

Table7:-Cart

This table contain cart details

SI No	Field name	Type	Constraints	Description
1	Pid	Varchar(30)	Foreign key	Product id
2	User	Varchar(30)	Not null	Type of user
3	Qty	Numeric(5,0)	Not null	Quantity of the product
4	Gtot	Numeric(5,0)	Not null	Grand total
5	Status	Numeric(4,0)	Not null	Status

Table8:-Loan

This table contain loan details

SI NO	Field Name	Type	Constraints	Description
1	Mid	Numeric(10,0)	Foreign Key/Primary key	Member id
2	Lamt	Numeric(10,0)	Not null	Loan amount
3	Reason	Varchar(80)	Not null	Reason for applying loan
4	Reqdate	Date	Not null	Requested date
5	Status	Numeric(3)	Not null	Status of the loan

Table9:-Order

This table contain order details

SI No	Field name	Type	Constraints	Description
1	Pid	Numeric(5,0)	Primary key	Product id
2	Mid	Numeric(10,0)	Foreign key	Member id
3	Cid	Numeric(10,0)	Foreign key	Customer id
4	Cdate	Date	Not null	Pay date
5	Qty	Numeric(5,0)	Not null	Quantity
6	Gtotal	Numeric(5,0)	Not null	Grand total
7	Status	Numeric(5,0)	Not null	Status

Table10:-Feedback

This table contain feedback details

SI No	Field name	Type	Constraints	Description
1	Fid	Numeric(5,0)	Primary key	Feedback id
2	Pid	Numeric(5,0)	Foreign key	Product id
3	Orderid	Numeric(5,0)	Foreign key	Order id
4	Mid/Custid	Numeric(20,0)	Not null	Member/customer id
5	Feedback	Varchar(60)	Not null	Feedback
6	Fdate	Date	Not null	Feedback date

2.8 System testing

2.8.1. Introduction

Software Testing is the process of executing software in a controlled manner, in order to answer the question - Does the software behave as specified? Software testing is often used in association with the term's verification and validation. Validation is the checking or testing of items, includes software, for conformance and consistency with an associated specification. Software testing is just one kind of verification, which also uses techniques such as reviews, analysis, inspections, and walkthroughs. Validation is the process of checking that what has been specified is what the user actually wanted. Other activities which are often associated with software testing are static analysis and dynamic analysis. Static analysis investigates the source code of software, looking for problems and gathering metrics without actually executing the code. Dynamic analysis looks at the behaviour of software while it is executing, to provide information such as execution traces, timing profiles, and test coverage information. Testing is a set of activity that can be planned in advanced and conducted systematically. Testing begins at the module level and work towards the integration of entire computers-based system. Nothing is complete without testing, as it vital success of the system testing objectives, there are several rules that can serve as testing objectives. They are: Testing is a process of executing a program with the intent of finding an error.

- A good test case is one that has high possibility of finding an undiscovered error.
- A successful test is one that uncovers an undiscovered error.

If a testing is conducted successfully according to the objectives as stated above, it would uncover errors in the software. Also testing demonstrate that the software function appears to be working according to the specification, that performance requirement appears to have been met. There are three ways to test program.

- For correctness
- For implementation efficiency
- For computational complexity

Test for correctness is supposed to verify that a program does exactly what it was designed to do. This is much more difficult than it may at first appear, especially for large programs.

2.8.2 Test Plan

A test plan implies a series of desired course of action to be followed in accomplishing various testing methods. The Test Plan acts as a blue print for the action that is to be followed. The software engineers create a computer program, its documentation and related data structures. The software developers is always responsible for testing the individual units of the programs, ensuring that each performs the function for which it was designed. There is an independent test group (ITG) which is to remove the inherent problems associated with letting the builder to test the thing that has been built. The specific objectives of testing should be stated in measurable terms. So that the mean time to failure, the cost to find and fix the defects, remaining defect density or frequency of occurrence and test work-hours per regression test all should be stated within the test plan. The levels of testing include:

- Unit testing Integration
- Testing Data validation
- Testing Output Testing

2.8.2.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of software design – the software component or module. Using the component level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and uncovered scope established for unit testing. The unit testing is whitebox oriented, and step can be conducted in parallel for multiple components. The modular interface is tested to ensure that information properly flows into and out of the program unit under test. The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm's execution. Boundary conditions are tested to ensure that all statements in a module have been executed at least once. Finally, all error handling paths are tested. Tests of data flow across a module interface are required before any other test is initiated. If data do not enter and exit properly, all other tests are moot. Selective

testing of execution paths is an essential task during the unit test. Good design dictates that error conditions be anticipated and error handling paths set up to reroute or cleanly terminate processing when an error does occur. Boundary testing is the last task of unit testing step. Software often fails at its boundaries.

2.8.2.2 Integration Testing

Integration testing is systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design. The entire program is tested as whole. Correction is difficult because isolation of causes is complicated by vast expanse of entire program. Once these errors are corrected, new ones appear and the process continues in a seemingly endless loop. After performing unit testing in the System all the modules were integrated to test for any inconsistencies in the interfaces. Moreover, differences in program structures were removed and a unique program structure was evolved.

2.8.2.3 Validation Testing or System Testing

This is the final step in testing. In this the entire system was tested as a whole with all forms, code, modules and class modules. This form of testing is popularly known as Black Box testing or System tests. Black Box testing method focuses on the functional requirements of the software. That is, Black Box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program. Black Box testing attempts to find errors in the following categories; incorrect or missing functions, interface errors, errors in data structures or external data access, performance errors and initialization errors and termination errors.

2.8.2.4 Output Testing or User Acceptance Testing

The system considered is tested for user acceptance; here it should satisfy the firm's need. The software should keep in touch with perspective system; user at the time of developing and making changes whenever required. This done with respect to the following points:

- Input Screen Designs
- Output Screen Designs

The above testing is done taking various kinds of test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested using that test data. While testing the system by which test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use. The system is validated by negotiating the existence and proposed system. This test evaluates the system in the real time environment with live data and finds it to be satisfied. This is done by the user. The various possibilities of the data are entered and response from the system is tested once the acceptance testing is signed off by the user.

User tests, usually performed by clients or by end-users, do not normally focus on identifying simple problems such as spelling errors or cosmetic problems, nor on showstopper defects, such as software crashes; testers and developers previously identify and fix these issues during earlier unit testing, integration testing, and system testing phases. UAT should be executed against test scenarios. Test scenarios usually differ from System or Functional test cases in the sense that they represent a "player" or "user" journey. The broad nature of the test scenario ensures that the focus is on the journey and not on technical or system-specific key presses, staying away from "click-by-click" test steps to allow for a variance in users' steps through systems.

2.8.3 Test Case

Test Case 1					
Project Name: Kumdumbhashree Process Handling System					
Login Test Case					
Test Case ID: Fun_1			Test Designed by: VIJAYALAKSHMI P B		
Test Priority (Low/Medium/High): High			Test Designed Date: 18-06-2021		
Module Name: Login Screen			Test Executed by: Ms. PAULINE PAUL		
Test Title: Verify login with valid username and password			Test Execution Date: 18-06-2020		
Description: Test the Login Page					
Pre-Condition: User has valid user name and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Navigation to Login Page		Login Page should be	Login page displayed	Pass
2	Provide Valid User name	User Name: admin	User should be able to Login	User Logged in and navigated to Admin Dashboard with records	Pass
3	Provide Valid Password	Password: dmin123			
4	Click on Login button				
5	Provide Invalid User Name or password	User Name: 123@gmail Password: admin	User should not be able to Login	Message for enter valid user or password displayed	Pass
6	Provide Null User Name or Password	User Name:null Password: null			
7	Click on Login button				
Post-Condition: User is validated with database and successfully login into account. The Account session details are logged in database					

Test Case 2

Project Name: Kumdumbashree Process Handling System					
User Registration Test Case					
Test Case ID: Fun_2			Test Designed by: VIJAYALAKSHMI P B		
Test Priority (Low/Medium/High): Medium			Test Designed Date: 18-06-2021		
Module Name: User Registration			Test Executed by: Ms. PAULINE PAUL		
Test Title: Verify new user registration			Test Execution Date: 18-06-2021		
Description: Test the User registration Page					
Pre-Condition: User should not be already registered					
Step	Test Step	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Navigation to Registration Page		Registration Page should be displayed	Registration page displayed	Pass
2	Provide null information	Name: null	Validation message should be displayed	*Mandatory field message displayed	Pass
3	Provide Valid Details of user	All the mandatory registration details of the user	User should be able to register successfully	Registration successful and home page displayed	Pass
4	Click on Register button				
Post-Condition: User is validated with database and successfully registered in the application. The Account session details are logged in database					

2.9 Implementation

Implementation is the stage of the project where the theoretical design is turned into a working system. It can be considered to be the most crucial stage in achieving a successful new system gaining the users confidence that the new system will work and will be effective and accurate. It is primarily concerned with user training and documentation. Conversion usually takes place about the same time the user is being trained or later. Implementation simply means convening a new system design into operation, which is the process of converting a new revised system design into an operational one. At this stage the main work load, the greatest upheaval and the major impact on the existing system shifts to the user department. If the implementation is not carefully planned or controlled, it can create chaos and confusion. Implementation includes all those activities that take place to convert from the existing system to the new system. The new system may be a totally new, replacing an existing manual or automated system or it may be a modification to an existing system. Proper implementation is essential to provide a reliable system to meet organization requirements. The process of putting the developed system in actual use is called system implementation. This includes all those activities that take place to convert from the old system to the new system. The system can be implemented only after thorough testing is done and if it is found to be working according to the specifications. The system personnel check the feasibility of the system. The more complex the system being implemented, the more involved will be the system analysis and design effort required to implement the three main aspects: education and training, system testing and changeover. The implementation state involves the following tasks:

- Careful planning
- Investigation of system and constraints
- Design of methods to achieve the changeover

2.9.1 Implementation Procedures

Implementation of software refers to the final installation of the package in its real environment, to the satisfaction of the intended uses and the operation of the system. In many organizations someone who will not be operating it, will commission the software development

project. In the initial stage people doubt about the software but we have to ensure that the resistance does not build up, as one has to make sure that:

- The active user must be aware of the benefits of using the new system.
- Their confidence in the software is built up.

Proper guidance is imparted to the user so that he is comfortable in using the application. Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running in the server. If the server object is not up running on the server, the actual process won't take place.

2.9.2 User Training

User training is designed to prepare the user for testing and converting the system. To achieve the objective and benefits expected from computer-based system, it is essential for the people who will be involved to be confident of their role in the new system. As system becomes more complex, the need for training is more important. By user training the user comes to know how to enter data, respond to error messages, interrogate the database and call up routine that will produce reports and perform other necessary functions.

2.9.3 Operational Document

After providing the necessary basic training on computer awareness the user will have to be trained on the new application software. This will give the underlying philosophy of the use of the new system such as the screen flow, screen design type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the ways to correct the date entered. It should then cover information needed by the specific user/ group to use the system or part of the system while imparting the training of the program on the application. This training may be different across different user groups and across different levels of hierarchy.

2.9.4 System Maintenance

Maintenance is the enigma of system development. The maintenance phase of the software cycle is the time in which a software product performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance is for it to make adaptable to the changes in the system environment. Software maintenance is of course, far more than "Finding Mistakes".

2.10 Conclusion and Future Enhancements

2.10.1 Future Enhancement

- Develop a Mobile Application for the website.
- Provide more security.
- Attendance module will be converted to online punching method.

2.10.2 Conclusion

The current system is old fashioned and totally manual and time consuming. The mate takes more effort to collect land details, declare job, prepare eligible requested employees list, marking attendance details etc. The proposed system is accessed using internet and the employees can register job according to job declared by mates. All the activities are done with the help of a computer or mobile with internet facility. I.e.; there is no need of a paper or manual work. The landowners submit the site request to village authority is made through this site. Based on the rules and conditions approval/rejection of the site is automatically get through this site. After getting approval the mate can able to view the list of new jobs and they can apply jobs as they like by checking job details, land details etc. After getting approval to start job to mate. This site is also accessible to the employees. So, they can easily apply to the job offered by the team. They can view the current updates from higher authority. So, they don't face any difficulties in knowing information, and don't waste their time. Admin broadcast the messages through this site. So, mate don't have much more effort. All is done in one click. The chance of making mistakes in list prepare is very less. The photos of each work and attendance

details are send to the village authority for verification at the end of the project. This reduces the effort of mate mostly. The data used in this site is highly secure, it will also provide data security as we are using the secured databases for maintaining the documents. The registered employees and mates can login to the site at any time anywhere and can access the updated information properly. Due to the automated functionality the time required to request site, accept job request and list preparation, attendance marking, payment etc are reduced. This site is very much useful to many employees and mates. It helps the users of this system to easily handle all the activities related to a job. This site is developing for the 16th ward of our village named kalanjoor. Due to automated functionalities of this site, it become very useful and helpful to solve the problems that faced today. All employees get job without considering any personal issues, and get proper salary for each employee as they work according to the efficient attendance list. And this site can fulfil the objectives of government that is beneficial to the peoples in rural area in a highly manner.

2.11 Bibliography Books/References:

- Gary B. Shelly, Harry J. Rosenblatt, “System Analysis and Design”, 2009.
- Roger S Pressman, “Software Engineering”, 1994.
- Pankaj Jalote, “Software engineering: a precise approach”, 2006.
- James lee and Brent ware Addison, “Open-source web development with LAMP”, 2003
- IEEE Std 1016 Recommended Practice for Software Design Descriptions

Websites:

- www.w3schools.com
- www.jquery.com
- <https://realpython.com/introduction-to-flask-part-2-creating-a-login-page>
- <https://www.geeksforgeeks.org/login-and-registration-project-using-flask-and-mysql>
- <https://youtu.be/fszxBvnlR5E>
- <https://flask.palletsprojects.com/en/0.12.x/tutorial/views/>
- <https://youtu.be/iyCEw23a10c>

2.12 APPENDIX

2.12.1 SAMPLE CODE

models.py connection code

```
from django.db import models
class Customer(models.Model):
    fname=models.CharField(max_length=30)
    lname = models.CharField(max_length=30)
    email = models.CharField(max_length=30)
    phone = models.CharField(max_length=30)
    address = models.CharField(max_length=50)
    city = models.CharField(max_length=30)
    district = models.CharField(max_length=30)
    pin = models.CharField(max_length=30)

class Login(models.Model):
    username=models.CharField(max_length=30)
    password=models.CharField(max_length=30)
    status=models.IntegerField()
class Member(models.Model):
    fname = models.CharField(max_length=30)
    lname = models.CharField(max_length=30)
    adhno = models.CharField(max_length=30)
    photo = models.FileField()
    data=models.FileField
    hname = models.CharField(max_length=30)
    place = models.CharField(max_length=30)

    pin = models.CharField(max_length=30)
    phone = models.CharField(max_length=50)
    caste = models.CharField(max_length=30)
    dob = models.CharField(max_length=30)
    doj = models.CharField(max_length=30)
    email = models.CharField(max_length=30)
    password = models.CharField(max_length=30)

class Product(models.Model):
    pname = models.CharField(max_length=30)
    srate = models.IntegerField()
    photo = models.FileField()
    data = models.FileField
    ptype = models.CharField(max_length=30)
    punit = models.CharField(max_length=30)
    dp = models.CharField(max_length=30)
    doe = models.CharField(max_length=30)
class Pstockentry(models.Model):
    pname= models.ForeignKey(Product,on_delete=models.CASCADE)
    eno=models.IntegerField(default=1)
    qty=models.CharField(max_length=30)
    edate=models.CharField(max_length=30)
class Pstock(models.Model):
    pname= models.ForeignKey(Product,on_delete=models.CASCADE)
    cstock=models.CharField(max_length=30)
```

```

class Cart(models.Model):
    pid = models.ForeignKey(Product,on_delete=models.CASCADE)
    user=models.CharField(max_length=30)
    qty=models.IntegerField()
    gtotal=models.IntegerField()
    status = models.IntegerField()
class Loan(models.Model):
    mid=models.ForeignKey(Member,on_delete=models.CASCADE)
    lamt=models.IntegerField()
    reason=models.CharField(max_length=30)
    reqdate=models.CharField(max_length=30)
    status = models.CharField(max_length=30)
class Order(models.Model):
    mid = models.ForeignKey(Member, on_delete=models.CASCADE)
    cid = models.ForeignKey(Customer, on_delete=models.CASCADE)
    pid = models.ForeignKey(Product, on_delete=models.CASCADE)
    cdate = models.CharField(max_length=30)
    qty = models.IntegerField()
    gtotal = models.IntegerField()
    status = models.CharField(max_length=30)

class Feedback(models.Model):
    mid = models.ForeignKey(Member, on_delete=models.CASCADE)
    pid = models.ForeignKey(Product, on_delete=models.CASCADE)
    orderid = models.ForeignKey(Order, on_delete=models.CASCADE)
    fback=models.CharField(max_length=100)
    fdate= models.CharField(max_length=30)

```

views.py connection code

```

from django.shortcuts import render
from django.views.decorators.csrf import csrf_exempt
from .models import
Customer,Login,Member,Activity,Product,Pstockentry,Pstock,Actentry, Cart,Loan,Order,O
rder1,Feedback
from datetime import date
import razorpay

def home(request):
    data=Product.objects.all()
    return render(request,'home.html',{'d':data})
def login(request):
    return render(request,'login.html')
def log(request):
    try:
m=Login.objects.get(username=request.POST['phone'],password=request.POST['password']
)
        if m.status == 1:
            request.session['user']=m.username
            data = Product.objects.all()
            return render(request, 'uhome.html', {'d': data})

        elif m.status == 2:

            request.session['user'] = m.username

```

```

        data = Product.objects.all()

        return render(request, 'mhome.html', {'d': data})

        return render(request, 'mhome.html')

    elif m.status == 0:

        return render(request, 'adminhome.html')
    else:
        return render(request, 'login.html', {'error': "Your Username and
Password didn't match."})
    except:
        return render(request, 'login.html', {'error': 'Invalid Uername or Password'})

def adminhome(request):
    return render(request, 'adminhome.html')
def ureg(request):
    return render(request, 'ureg.html')
def reg(request):
    fname = request.POST['fname']
    lname = request.POST['lname']
    email = request.POST['email']
    phone = request.POST['phone']
    address = request.POST['address']
    city = request.POST['city']
    district = request.POST['district']
    pin = request.POST['pin']
    password = request.POST['pass']

    data=Customer(fname=fname,lname=lname,email=email,phone=phone,address=address,city=c
ity,district=district,pin=pin)
    data.save()
    data1=Login(username=phone,password=password,status=1)
    data1.save()
    return render(request, 'login.html')
def uhome(request):
    data=Product.objects.all()
    return render(request, 'uhome.html', {'d': data})
def uprodetails(request):
    id = request.POST['id']
    data = Product.objects.get(id=id)
    stock = Pstock.objects.get(pname=data)

    st = []
    for i in range(1, int(stock.cstock) + 1):
        st.append(i)

    return render(request, 'uprodetails.html', {'i': data, 'stock': stock, 'st':
st})

def cart1(request):
    id = request.POST['id']

```

```

data3=Product.objects.get(id=id)
data = Product.objects.get(id=id)
user = request.session['user']
data1 = Customer.objects.get(phone=user)
qty = request.POST['qty']
gtotal=int(qty)*int(data.srate)
data2 = Cart(pid=data3,user=user,qty=qty,gtotal=gtotal,status=0)
data2.save()
return render(request, 'gocart.html', {'i': data,'data1':data1, 'data2': data2
})
def gocart(request):
    id = request.session['user']
    data = Cart.objects.filter(user=id)
    return render(request,'cartdetails.html',{'data':data})
def cartdetails(request):
    id = request.session['user']
    data = Cart.objects.filter(user=id)
    return render(request, 'cartdetails.html', {'data': data})
def shopping(request):
    id = request.POST['id']
    data = Cart.objects.filter(id=id)
    user = request.session['user']
    data2 = Customer.objects.get(phone=user)
    return render(request, 'shopping.html', {'data1': data,'h':data2})

def payment(request):

    user = request.session['user']
    insta = Member.objects.get(phone=user)
    pid = request.POST['pid']
    instal = Product.objects.get(id=pid)
    cdate = date.today()
    qty = request.POST['qty']
    request.session['qty'] = qty
    request.session['pid'] = pid
    gttotal = request.POST['gttotal']
    data = Order(mid=insta, pid=instal, cdate=cdate, qty=qty, gttotal=gttotal,
status=0)
    data.save()
    if request.method == 'POST':
        amount = int(gttotal) * 100
        order_currency = 'INR'
        client = razorpay.Client(auth=('rzp_test_GeBzmlv3NuniE7',
'wFxTGPyRQDWvOefUw5bm6K55'))
        payment = client.order.create({'amount': amount, 'currency': 'INR',
'payment capture': '1'})

        return render(request, 'payment.html', {'amount': amount, 'user':
insta.email})

@csrf_exempt
def success(request):

    print('success function ok')
    id = request.session['user']
    pid = request.session['pid']
    qty = request.session['qty']
    instal = Product.objects.get(id=pid)

```



```

data = Pstock.objects.get(pname=instal)

data.cstock = int(data.cstock) - int(qty)
data.save()

Cart.objects.filter(user=id, pid=instal).delete()
Order.objects.filter(pid=instal).update(status=1)
return render(request, 'success.html')
def mfbck(request):
    user = request.session['user']
    data = Member.objects.get(phone=user)
    myorder = Order.objects.filter(mid=data)
    pid = request.POST['pid']
    instal = Order.objects.get(id=pid)

    return render(request, 'mfbck.html', {'myoder':myorder, 'pid':instal})
def mfbck1(request):
    user = request.session['user']
    insta = Member.objects.get(phone=user)
    pid = request.POST['pid']
    instal = Order.objects.get(id=pid)
    orderid = request.POST['orderid']
    insta2 = Order.objects.get(id=orderid)
    fback = request.POST['fback']
    fdate = date.today()
    data = Feedback(mid=insta, pid=instal, orderid=insta2, fback=fback,
fdate=fdate)
    data.save()
    return render(request, 'mfbck.html')

def mhome(request):
    data=Product.objects.all()
    return render(request, 'mhome.html', {'d':data})
def mprodetails(request):
    id = request.POST['id']
    data = Product.objects.get(id=id)
    stock = Pstock.objects.get(pname=data)
    if stock.cstock=='0':
        data = Product.objects.all()
        return render(request, 'mhome.html', {'d': data, 's':"SELECTED ITEM IS OUTOF
STOCK"})

    st = []
    for i in range(1, int(stock.cstock) + 1):
        st.append(i)

    return render(request, 'mprodetails.html', {'i': data, 'stock': stock, 'st':
st})

def cart2(request):
    id = request.POST['id']
    data3=Product.objects.get(id=id)
    data = Product.objects.get(id=id)
    user = request.session['user']
    data1 = Member.objects.get(phone=user)

```

```

    qty = request.POST['qty']
    gtotal=int(qty)*int(data.srate)
    data2 = Cart(pid=data3,user=user,qty=qty,gtotal=gtotal,status=0)
    data2.save()
    return render(request, 'gocart1.html', {'i': data,'data1':data1, 'data2': data2
})

def gocart1(request):
    id = request.session['user']
    data = Cart.objects.filter(user=id)
    return render(request,'cartdetails1.html',{'data':data})
def cartdetails1(request):
    id = request.session['user']
    data = Cart.objects.filter(user=id)
    return render(request, 'cartdetails1.html', {'data': data})
def shopping3(request):
    id = request.POST['id']
    data = Cart.objects.filter(id=id)
    user = request.session['user']
    data2 = Member.objects.get(phone=user)
    return render(request, 'shopping3.html', {'data1': data,'h':data2})

def remove(request):
    id=request.POST['id']
    Cart.objects.filter(id=id).delete()
    id = request.session['user']
    data = Cart.objects.filter(user=id)
    return render(request, 'cartdetails.html', {'data': data})
def remove1(request):
    id=request.POST['id']
    Cart.objects.filter(id=id).delete()
    id = request.session['user']
    data = Cart.objects.filter(user=id)
    return render(request, 'cartdetails1.html', {'data': data})

def registration(request):
    return render(request,'registration.html')
def memberreg(request):
    return render(request,'memberreg.html')
def memreg(request):
    fname = request.POST['fname']
    lname = request.POST['lname']
    adhno= request.POST['adhno']
    photo = request.FILES['file']
    hname = request.POST['hname']
    place = request.POST['place']
    pin = request.POST['pin']
    phone = request.POST['phone']
    caste = request.POST['caste']
    dob = request.POST['dob']
    doj= date.today()
    email = request.POST['email']
    password = request.POST['password']
    data = Member(fname=fname, lname=lname, adhno=adhno, photo=photo, hname=hname,
place=place, pin=pin, phone=phone,caste=caste, dob=dob,doj=doj,email=email)
    data.save()
    datal = Login(username=phone, password=password, status=2)
    datal.save()
    return render(request, 'memberreg.html')

```

```

def loan(request):
    return render(request, 'loan.html')
def reqloan(request):
    return render(request, 'reqloan.html')
def rloan(request):
    user = request.session['user']
    insta=Member.objects.get(phone=user)
    lamt = request.POST['lamt']
    reason = request.POST['reason']
    reqdate=date.today()
    data = Loan(mid=insta,lamt=lamt, reason=reason, reqdate=reqdate, status=0)
    data.save()
    return render(request, 'reqloan.html')
def pending(request):
    data = Loan.objects.filter(status=0)
    return render(request, 'pending.html', {'d': data})
def approve(request):
    id=request.POST['id']
    Loan.objects.filter(id=id).update(status=1)
    data=Loan.objects.filter(status=1)
    return render(request, 'approve.html', {'d':data})
def reject(request):
    id=request.POST['id']
    Loan.objects.filter(id=id).update(status=2)
    data=Loan.objects.filter(status=2)
    return render(request, 'reject.html', {'d':data})
def status(request):
    id = request.session['user']
    user = Member.objects.get(phone=id)
    data=Loan.objects.filter(mid=user)
    return render(request, 'status.html', {'d':data})
def approve1(request):
    data=Loan.objects.filter(status=1)
    return render(request, 'approve.html', {'d':data})
def reject1(request):
    data=Loan.objects.filter(status=2)
    return render(request, 'reject.html', {'d':data})
def apploan2(request):
    data=Loan.objects.filter(status=1)
    return render(request, 'apploan2.html', {'d':data})

def productreg(request):
    return render(request, 'productreg.html')
def proreg(request):
    pname = request.POST['pname']
    srate = request.POST['srate']
    photo = request.FILES['file']
    ptype = request.POST['ptype']
    punit = request.POST['punit']
    dp = request.POST['dp']
    doe = date.today()

    data=Product(pname=pname,srate=srate,photo=photo,ptype=ptype,punit=punit,dp=dp,doe=doe)
    data.save()
    return render(request, 'productreg.html')
def show(request):
    return render(request, 'show.html')
def productdetails(request):
    data=Product.objects.all()

```

```

        return render(request, 'productdetails.html', {'d': data})

def editadminpro(request):
    id = request.POST['id']
    data = Product.objects.get(id=id)
    return render(request, 'editadminpro.html', {'data': data})

def update1(request):
    id=request.POST['id']
    data=Product.objects.get(id=id)
    data.pname=request.POST['pname']
    data.srate = request.POST['srate']
    data.photo = request.FILES['file']
    data.ptype = request.POST['ptype']
    data.punit = request.POST['punit']
    data.dp = request.POST['dp']
    data.save()
    data2=Product.objects.filter(id=id)
    return render(request, 'productdetails.html', {'d': data2})

def memberdetails(request):
    data = Member.objects.all()
    return render(request, 'memberdetails.html', {'d': data})

def memberdetails1(request):
    id = request.POST['id']
    data1 = Member.objects.filter(id=id)
    return render(request, 'memberdetails1.html', {'data1': data1})

def memberdetails2(request):
    data=Member.objects.all()
    return render(request, 'memberdetails2.html', {'d': data})

def productdetails2(request):
    data=Product.objects.all()
    return render(request, 'productdetails2.html', {'d': data})

def addingstock(request):
    data1=Product.objects.all()
    return render(request, 'addingstock.html', {'d': data1})

def adstock(request):
    pname=request.POST['id']
    data = Product.objects.get(id=pname)
    try:
        data3 = Pstock.objects.get(pname=data)
        return render(request, 'adstock.html', {'i': data, 'data3': data3})
    except:
        pass

    return render(request, 'adstock.html', {'i': data})

def adlstock(request):
    id = request.POST['id']
    instal=Product.objects.get(id=id)

    qty = request.POST['qty']
    edate = date.today()
    data2 = Pstockentry(pname=instal, qty=qty, edate=edate)
    data2.save()
    try:
        data3=Pstock.objects.get(pname=instal)
        data3.cstock=int(data3.cstock)+int(qty)
        data3.save()
    except :

```

```

        data3=Pstock(pname=instal,cstock=qty)
        data3.save()

    data3 = Pstock.objects.get(pname=instal)

    return render(request, 'adstock.html', {'i': instal,'data3':data3})
def viewcustprofile(request):
    user=request.session['user']
    data=Customer.objects.get(phone=user)
    return render(request,'viewcustprofile.html',{'i':data})
def viewmemprofile(request):
    user=request.session['user']
    data=Member.objects.get(phone=user)
    return render(request,'viewmemprofile.html',{'i':data})
def search(request):

    return render(request, 'search.html')
def sear(request):
    id=request.POST['id']
    data=Product.objects.filter(id=id)
    return render(request,'search.html',{'d':data})
def editcustprofile(request):
    user = request.session['user']
    data = Customer.objects.get(phone=user)
    return render(request, 'editcustprofile.html', {'data': data})

def update(request):
    id=request.POST['id']
    data=Customer.objects.get(id=id)
    data.fname=request.POST['fname']
    data.lname = request.POST['lname']
    data.email = request.POST['email']
    data.phone = request.POST['phone']
    data.address = request.POST['address']
    data.city = request.POST['city']
    data.district = request.POST['district']
    data.pin = request.POST['pin']
    data.save()
    return render(request, 'viewcustprofile.html',{'i':data})

def memorder(request):
    data = Order.objects.all()
    return render(request, 'memorder.html', {'d': data})

def custorder(request):
    data = Order1.objects.all()
    return render(request, 'custorder.html', {'d': data})
def myorders(request):
    user = request.session['user']
    data = Member.objects.get(phone=user)
    myorder=Order.objects.filter(mid=data,status=1)
    return render(request, 'myorders.html',{'d':myorder})
def invoice(request):
    user = request.session['user']
    data = Member.objects.get(phone=user)
    myorder=Order.objects.filter(mid=data)

    return render(request, 'invoice.html',{'d':myorder})

```

```

def invoice1(request):
    user = request.session['user']
    data = Customer.objects.get(phone=user)
    myorder=Order1.objects.filter(cid=data)

    return render(request, 'invoice1.html',{'d':myorder})
def myorders4(request):
    user = request.session['user']
    data = Customer.objects.get(phone=user)
    myorder=Order1.objects.filter(cid=data,status=1)

    return render(request, 'myorders4.html',{'d':myorder})

def psold(request):
    data = Order.objects.all()
    return render(request, 'psold.html', {'d': data})

def logout(request):
    try:
        del request.session['user']
    except KeyError:
        pass
    return render(request, 'login.html')

```

Payment.html

```

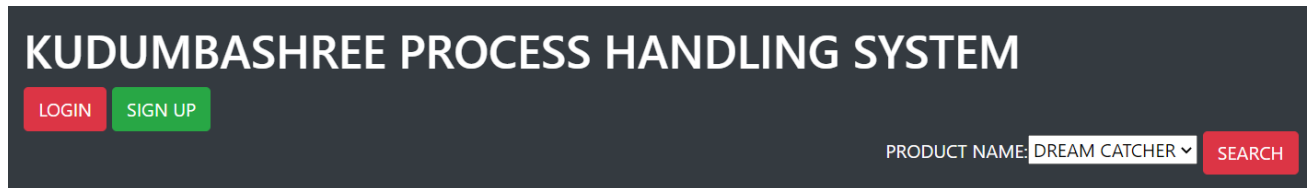
<!DOCTYPE html>
{% load static %}
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<h1>DONOT REFRESH </h1>
<form action="success" method="POST">
    {% csrf_token %}
<script
    src="https://checkout.razorpay.com/v1/checkout.js"
    data-key="rzp_test_GeBzmlv3NuniE7"
    data-amount="{{ amount }}"
    data-currency="INR"
    data-order_id="{{ payment.id }}"
    data-buttontext="Proceed payment"
    data-name="KUDUMBASHREE"
    data-description="An example from Kerala on Woman Empowerment through Local
Governance"
    data-image="{% static '20.jpg' %}"
    data-prefill.name="Gaurav Kumar"
    data-prefill.email="{{ user }}"

    data-theme.color="#F37254"
></script>
<input type="hidden" custom="Hidden Element" name="hidden"></form></body></html>

```

2.12.2 SCREENSHOTS

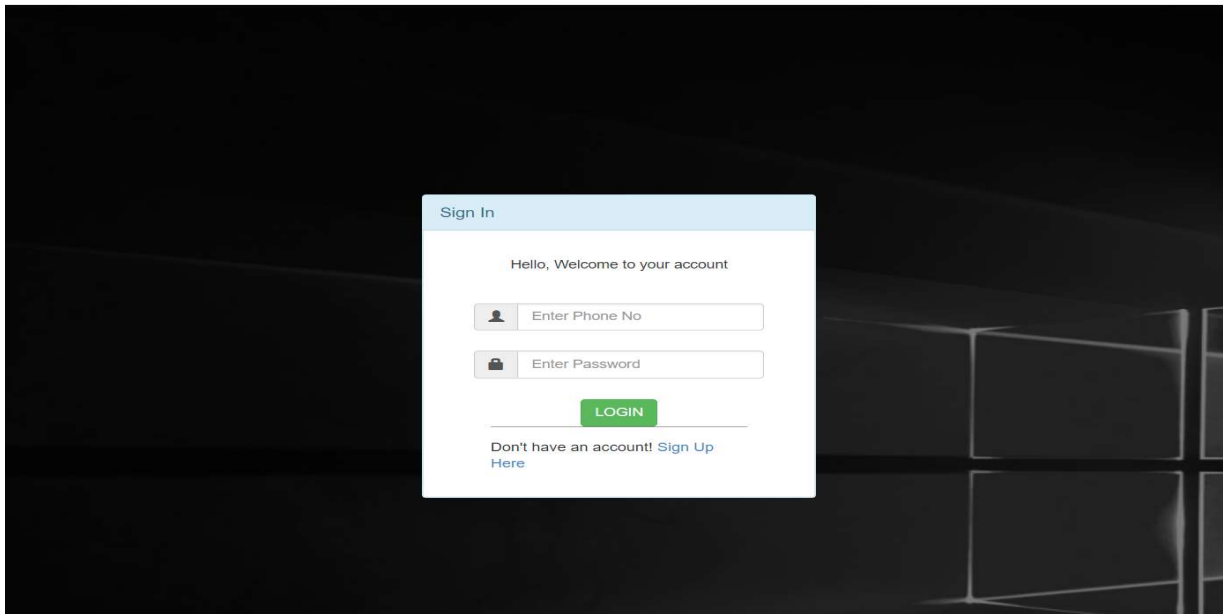
Home page



LOCAL SELF GOVERNMENT DEPARTMENT , KERALA

Kudumbashree, the Kerala State Poverty Eradication Mission was launched on 17th May 1998 inaugurated by the Prime Minister, Shri Atal Bihari Vajpayee. The Mission aims to eradicate absolute poverty within a definite time frame of 10 years under the leadership of Local Self Governments formed and empowered by the 73rd and 74th Amendments of the Constitution of India. The Mission launched by the State Government with the active support of Government of India and NABARD has adopted a different methodology in addressing poverty by organizing the poor in to community-based organizations. The Mission follows a process approach rather than a project approach. Kudumbashree, a community organization of Neighborhood Groups (NHGs) of women in Kerala, has been recognized as an effective strategy for the empowerment of women in rural as well as urban areas: bringing women together from all spheres of life to fight for their rights or for empowerment. The overall empowerment of women is closely linked to economic empowerment. Women through these NHGs work on a range of issues such as health, nutrition, agriculture, etc. besides income generation activities and seeking micro credit. Kudumbashree differs from conventional programmes in that it perceives poverty not just as the deprivation of money, but also as the deprivation of basic rights. The poor need to find a collective voice to help claim these rights.

Login page



Product registration

REGISTER PRODUCT!!!

[Home](#) [View Products](#)

PRODUCT NAME:

PRICE:

PRODUCT IMAGE:

Choose File No file chosen

PRODUCT TYPE:

HOME DECOR

PRODUCT UNIT:

PER PIECE

DESCRIPTION:

REGISTER

RESET

Stock adding

ADDING STOCK!!!

[Home](#) [View Product](#)

QUANTITY:

REGISTER


RESET

PRODUCT NAME	PRODUCT TYPE	PRODUCT UNIT	DESCRIPTION	SALE RATE	PRODUCT IMAGE	CURRENT STOCK
DREAM CATCHER	HOME DECOR	PER PIECE	DREAM CATCHER BLUE COLOR	80		5




Pending approvals

PENDING APPROVALS

[Home](#)

SI NO	PHOTO	MEMBER NAME	HOUSE NAME	AMOUNT	REASON	REQ DATE	ACTION
1		CHAITHANYA CHANDRAN	PALAKKAL (H)	5000	EDUCATIONAL PURPOSE	2021-06-15	<div>APPROVE</div> <div>REJECT</div>




Member orders

MEMBER ORDERS							
Home							
Si No	Name	Email	Product Name	Product Image	Date	Quantity	Total
1	KARTHIKA SATHEESH	connectmekarthika@gmail.com	BOTTLE ART		2021-06-15	1	100
2	ANUPAMA A	aupama89@gmail.com	MANGO PICKLE		2021-06-15	1	250
3	CHAITHANYA CHANDRAN	chaithanya123@gmail.com	SANITIZER		2021-06-15	5	900

Products sold

KUDUMBASHREE PROCESS HANDLING SYSTEM

DETAILS OF PRODUCTS SOLD






SI NO	PRODUCT NAME	PRODUCT IMAGE	PRODUCT TYPE	PRODUCT UNIT	DESCRIPTION	ORDER DATE	RATE	QUANTITY	GRAND TOTAL
1	BOTTLE ART		HOME DECOR	PER PIECE	BOTTLE ART FOR DECORATING	2021-06-15	100	1	100
2	MANGO PICKLE		PICKLE	500 gm	TASTY MANGO PICKLE	2021-06-15	250	1	250
3	SANITIZER		SANITIZER	500 ml	KILL 99.9% OF BACTERIA WITHOUT WATER	2021-06-15	180	5	900

PRINT

Member home

WELCOME TO MEMBER PAGE


[VIEW PROFILE](#) [MY CART](#) [MY ORDERS](#) [LOAN](#) [LOGOUT](#)

PRODUCT NAME	SALE RATE	PRODUCT IMAGE	PRODUCT TYPE	PRODUCT UNIT	DESCRIPTION	ACTION
DREAM CATCHER	80		HOME DECOR	PER PIECE	DREAM CATCHER BLUE COLOR	VIEW
BOTTLE ART	100		HOME DECOR	PER PIECE	BOTTLE ART FOR DECORATING	VIEW
LEMON PICKLE	200		PICKLE	500 gm	TASTY LEMON PICKLE	VIEW
MANGO PICKLE	250		PICKLE	500 gm	TASTY MANGO PICKLE	VIEW
SANITIZER	180		SANITIZER	500 ml	KILL 99.9% OF BACTERIA WITHOUT WATER	VIEW


User Cart

YOU'R CART

[Home](#)

PRODUCT NAME	PRODUCT IMAGE	PRODUCT TYPE	PRODUCT UNIT	DESCRIPTION	PRICE	QUANTITY	GRAND TOTAL	BUY	REMOVE
DREAM CATCHER		HOME DECOR	PER PIECE	DREAM CATCHER BLUE COLOR	80	2	160	BUY	REMOVE

Payment





KUDUMBASHREE
An example from Kerala on Wo...
₹ 160


×

English ▾


Country
+91 ▾

Phone
8113897993 

Email
chaithanya123@gmail.com 

 This payment is secured by Razorpay.

PROCEED

Secured by 

Customer orders

MY ORDERS							
Home							
SI NO	PRODUCT IMAGE	PRODUCT NAME	DATE	QUANTITY	GRAND TOTAL	ACTION	INVOICE
1		SANITIZER	2021-06-15	5	900	FEEDBACK	GENERATE INVOICE
2		DREAM CATCHER	2021-06-23	2	160	FEEDBACK	GENERATE INVOICE