ONLINE WASTE MANAGEMENT SYSTEM

PROJECT REPORT

Submitted by

ANJUMOL TENNYSON

Reg.No. SJC17MCA007

to

the APJ Abdul Kalam Technological University

in partial fulfillment of the requirement for the award of the degree

of

Master of Computer Applications

Department of Computer Science and Applications

St. Josephs College of Engineering and Technology

Palai

MAY 2020

DECLARATION

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

Choondacherry Anjumol Tennyson

15 May, 2020

DEPARTMENT OF COMPUTER SCIENCE AND APPLICATIONS

ST. JOSEPHS COLLEGE OF ENGINEERING AND TECHNOLOGY

Palai, Kottayam 686 579

(Approved by AICTE and affiliated to APJ Abdul Kalam Technological University)

CERTIFICATE

This is to certify that the report entitled **Online Waste Management System** submitted by **ANJUMOL TENNYSON** to the APJ Abdul Kalam Technological University in partial fulfillment of the re- quirements for the award of the Degree of Master of Computer Applications is a bonafide record of the project work carried out by her under my guidance and supervision.

Mrs. Rinu Rachel Varughese	**Mr. Alex Jose**
Asst. Professor, MCA	Asst. Professor, MCA
Internal Guide	Project Co-ordinator

Dr. T.D Jainendrakumar
Head of the Department

Viva-voce held on.......................

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

ACKNOWLEDGEMENT

First and foremost, we give all glory, honour and praise to God Almighty who gave us wisdom and enabled us to complete the project successfully. I also express sincere thanks, from the bottom of our heart, to our parents for their encouragement and support in all our endeavors and especially in this project.

I respect and thank the management of St. Josephs College of Engineering and Technology for providing me an opportunity and platform for doing the project work.

I am extremely thankful to our beloved Principal, **Dr. David J** for providing such a nice support and facilities to carry out this project.

I am extremely indebted to **Dr. Jainendrakumar T D**, Professor Head, department of Computer Science and Applications, who has been a constant source of inspiration and without his tremendous help and support this project would not have been materialized.

I owe my deep gratitude to my project guide **Mrs. Rinu Rachel Varughese**, Asst. Professor, department of Computer Science and Applications who took keen interest on my project and guided us all along, till the completion of my project work by providing all the necessary information for developing a good system.

I am thankful to and fortunate enough to get constant encouragement, support and guidance from all the faculty members of department of Computer Science and Applications, which helped us in successfully completing my project work. Also, I would like to extend my sincere esteems to all staff in laboratory for their timely support.

Anjumol Tennyson

Abstract

The proposes of this project is the establishment of a web based waste management and defines components that make up a Waste Management System and the objects that must be moved among these components. Waste Management, this will provide another way for the customer to giving the waste materials. It is an internet based web application that can be accessed throughout the internet and can be accessed by anyone who has a net connection.

This system consists of three modules.

- User Module
- Collecting System Module
- Admin Module

This system will allow registration of various users for selling waste materials. Then the users can login to the system with the help of username and a password. The system will provide sell as well as donate services. The system provides a platform to sell waste materials and buy recycled products and they can also done payments. Collecting System collect the waste materials sell by the user and they sell the recycled products to the user. The administrator of the system will manage the whole system. Admin can add the type of waste and recycled products.

Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

1.1 PROBLEM DEFINITION

The paper proposes the establishment of a web based waste management system and defines components that make up an waste management system and the objects that must be moved among these components. Waste management is an import thing in today. It includes many functionalities .This system consist of three modules:-Collecting System, User, Admin. This system helps users to sell their waste materials to a collecting system and the collecting system will collect the waste materials from user and they also have the functionality to buy the recycled products that are added by the collecting agents and also done the payments for selling and buying the materials. The collecting system is also collect the waste materials from user and also they sell the recycled products to the user and also done payment. The admin is the overall controller of the system and they provided the category of waste items and recycled products that are sell and bought by the user and the collecting system. And also it checks all the payment activities done buy the user and the collecting system .

1.2 ABOUT THE ORGANIZATION

St. Josephs College of Engineering and Technology, Palai was instituted with the objective of developing a center of professional learning with a distinct identity and character. The college aims to provide the kind of education that helps to achieve academic excellence and thereby ensures a challenging and satisfying career for the students on the successful completion of the program. With this perspective, training is organized on a regular basis for the development of personality, learning and communication skills as well as employability skills. Discipline, hard work, positive thinking, commitment to excellence and abiding faith in the Almighty are the guiding principles that propel the college to its vision of emerging as a Centre of Excellence in technical education in the country.

SJCET right from inception, has been maintaining high levels of standard in academic and extra curricular realms of activities. We offer BTECH degree courses in 6 engineering disciplines, and Masters Degree courses in Engineering, Computer Application and Business Administration. In the short span of a decade of its existence and among the six batches of students that have graduated, the college bagged several university ranks and has a remarkably high percentage of pass.

1.3 OBJECTIVE OF THE PROJECT

The main objective of the project is to create a website that helps users and collecting system in a society to sell and donate waste items in the form of waste item and recycled products. This system helps the collecting system to manage the waste items and make recycled products. Users they are sell and buy materials and products. Admin manage all the activities of user and collecting system.

Chapter 2

SYSTEM ANALYSIS

System analysis is a management technique which helps us in designing a new system or improving an existing system. System analysis is a process of gathering and interpreting facts, diagnosing problems and using the information to recommend improvement of the system. During these places, the analyst and the user come to a detailed agreement on whet function the proposed system has to perform. This one contains:

- Inputs to be supplied.
- Output to be supplied.
- Procedures to get the outputs from the given inputs.
- Data to be retained.

2.1 INITIAL INVESTIGATION

The purpose of this document is to give a clear picture of the module designs of the project Online Waste Management System . The website provide an easy way to sell and donate waste items and also buy the recycled products. Collecting agents and User will buy and sell products and make payments. This website provides an easy way to manage the waste items. This document is developed after a number of consultations with the staff and the HOD and considering the complete requirement specifications of the given project.

This project also helps to understand various functionalities of the modules in the project as well as it gives a pictorial design of how the website will look like with its functionalities working together various to achieve the requirements.

2.2 EXISTING SYSTEM

The study of the existing system is a pre-requisite for developing any software system. The study of the system reveals many features of the existing system. This gives analyst an insight into the working of the system and helps the developer to design an appropriate system, which will eliminate the many limitations present in the existing system.

Limitations of the Existing System are :-

- It is a manual system
- Process is by means of paper work
- Difficult to keep all the paper records
- The file manipulating method was not done in a centralized manner
- Document storing, accessing them takes more time
- Searching process is mainly done manually and it is difficult
- Chances of loss in document containing important details
- Difficult to find out accurate data in minimal time
- Time consuming and miss handling of reports

2.3 PROPOSED SYSTEM

The proposed system computerization is developed using SQLite3 as back-end and Python Django as front-end. The django framework is managed, type safe environment for application, developmet and execution. This system provides an easy way to manage all the waste items selled by the user and the buying of recycled products. Collecting agent is collect the waste materials from users site on the given date .Admin will manage the entire system and also control the payment activity.

2.3.1 Advantages of the Proposed System

- Give solution to the current system problems
- Less time consuming and more efficient
- Result will be very precise and accurate
- Easy to use and fast
- Simple user interface to reduce processing time
- Easy searching and storing documents
- Eliminate chances for errors and reduce effort

2.3.2 Features of the Proposed System

The various features of proposed system are as follows :-

- Access to the system and database as per user identification
- The maximum security ensured
- Integrity reliability and integrity of data
- User-friendly and flexible in all aspects
- Data entry updates is quite easy
- Effective table manipulation as facilitated by the rich SQLite3
- Good validation checking
- Easy maintenance
- Removes chances of leakage of information
- Provides a better record keeping system

All these form the major aspects and advantages of the proposed system. Provision is made for effective improvements of maintenance are needed at any stage.

2.4 FEASIBILITY STUDY

An important outcome of the preliminary investigation is the determination that the system requested is feasible. Feasibility study is carried out to select the best system that meets the performance requirements. During system analysis, a feasibility study of the proposed system was carried out to see whether it was beneficial to the organization. The main aim of the feasibility study is to determine whether it would be financially and technically feasible to develop the product. While evaluating the existing system, many advantages and disadvantages raised. Analyzing the problem thoroughly forms the vital part of the system buddy. Problematic areas are identified and information is collected.

Feasibility study is both necessary and prudent to evaluate the feasibility of the project at the earliest possible time. It involves preliminary investigation of the project and examines whether the designed system will be useful to the organization. Months or years of effort, thousand for millions of money and untold professional embarrassment can be averted if an in-conceived system is recognized early in the definition phase.

Eight steps involved in the feasibility analysis are :-

- Form a project team and appoint a project leader
- Prepare system flowcharts
- Enumerate potential proposed system
- Define and identify characteristics of proposed system
- Determine and evaluate performance and cost effective of each proposed system
- Weight system performance and cost data
- Select the best Proposed system

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

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

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

2.4.1 Technical Feasibility

According to feasibility analysis procedure the technical feasibility of the system is analyzed and the technical requirements such ad software facilities, procedure, inputs, are identified. While considering the problems of existing system, it is sufficient to implement the new system. The proposed system can be implemented to solve issues in the existing system. It includes the evaluation of and how it meets the proposed system. This system use Python Django as front-end technology and SQLite3 back-end technology.

2.4.2 Economic Feasibility

Economic feasibility deals about the economical impact faced by the organization to implement a new system. Financial benefits must equal or exceed the costs. The cost of conducting a full system, including software and hardware cost for the class of application being considered should be evaluated. Economic Feasibility in this project:

- The cost to conduct a full system investigation is possible.
- There is no additional manpower requirement.
- There is no additional cost involved in maintaining the proposed system.

Economic justification is generally the Bottom Line consideration for most systems. Economic justification includes a broad range of concerns that includes cost benefit analysis. In this we weight the cost and the benefits associated with the candidate system and if it suits the basic purpose of the organization i.e. profit making, the project is making to the analysis and design phase. The financial and the economic questions during the preliminary investigation are verified to estimate the followings:

- The cost to conduct a full system investigation.
- The cost of hardware and software for the class of application being considered.
- The benefits in the form of reduced cost.

The proposed system will give the minute information; as a result the performance is improved. This feasibility checks whether the system can be developed with the available funds. The Online-waste management system does not require enormous amount of money to be developed. This can bedone economically if planned judicially, so it is economically feasible.

2.4.3 Operational Feasibility

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

People are inherently resistant to change and computer has been known to facilitate changes. The system operation is the longest phase in the development life cycle of a system. So, Operational Feasibility should be given much importance. This system has a user-friendly interface. Thus it is easy to handle.

2.4.4 Behavioral Feasibility

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

Chapter 3

SYSTEM DESIGN

3.1 SOFTWARE AND HARDWARE REQUIREMENT SPECIFICATION

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

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

3.1.1 Hardware Requirement

- CPU - INTEL(R) Core(TM) i3-5005U
- HARD DISK SPACE -1 TB
- RAM - 4 GB
- DISPLAY - 19 STANDARD RATIO LCD MONITOR
- KEYBOARD - 101- or 102- KEYS
- CLOCK SPEED - 1.60 GHZ

3.1.2 Software Requirement

- OPERATING SYSTEM - WINDOWS 8.1
- WEB SERVER - IIS 7.5
- FRONT END – Python Django
- BACK END – SQLite3

3.2 SYSTEM DESIGN

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

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

3.2.1 Non-Functional Requirements

Performance Requirements

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

Security Requirements

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

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

The first step is to determine how the output is to be produced and in what format. Then input data and master files have to be designed as the next step and finally the impact of the candidate system on the user and organization are documented and evaluated by the management. After identifying the problem and the limitation of the existing system, a detailed design of the proposed system is conducted. Free flow personnel interview and reference to previous records prepared manually were the only methods taken to collect necessary information. At present, all organizations are on the path of computerization process.

Design is the phase that indicates the final system. It is the solution, the translation of requirements into ways of meeting them. In this phase the following elements were designed namely, data flow, data stores, processes, procedures was formulated in a manner that meet the project requirements. After logical design physical construction of the system is done.

The database tables, input screens, output screens, output reports are designed. After analyzing the various functions involved in the system the database, labels as dictionaries designed. Care is taken for the field name to be in self-explanatory form. Unnecessary fields are avoiding so as not affecting the storage system. Care must be taken to design the input screen in the most user- friendly way so as to help even the novice users to make entries approximately in the right place. This is being accomplished by the use of giving online help messages, which are brief and cleanly prompts users for appropriate action.

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

All input screens in the system are user friendly and are designed in such a way that even a layman can operate. The sizes of all screens are standardized. Reports generated in this software give the finer accepts of the required information, which helps in taking vital decision. The importance of the software design can be stated with a single word quality. Design is a place where quality is fostered in software development. Design is the only way where requirements are actually translated into a finished software product or system.

3.2.2 Modular Design

Mainly this project consists of 3 Modules:

- User Module
- Collecting System Module
- Admin Module

Administration Module

Admin is the main actor in this system. He has the entire control of the system which includes adding all the details to the system. Brief description about the functionalities performed by the admin is given below. After the admin successfully login to this website the admin can perform the functionalities including:

- **Admin Login**

  By the Username and password admin can login to the system.

- **Add Category Of Waste Products**

  Admin is responsible to add the category of waste products.

- **Add Category Of Recycled Products**

  Admin is responsible to add the category of recycled products.

- **View Registered Users**

  Admin is responsible to view the registered users who are ready to buy and sell recycled products and waste materials.

- **View Registered Collecting System**

  Admin is responsible to view all the registered collecting systems.

- **View Transaction Logs**

  Admin is responsible to view the transaction details done by user and collecting system.

Collecting System Module

Collecting System is important actor in the system. Collecting System can add recycled products that can be bought by the user, It also collect the selled waste items from user . After the Collecting System successfully login to this website the collecting system can perform the functionalities including:

- **Collecting System Registration**

  By the entering the details to register to the system.

- **Collecting System Login**

  By the Username and password collecting system can login to the system

- **Edit Profile**

  Collecting System can edit profile.

- **Add Recycled Products**

  Collecting System is responsible to add details of recycled products .

- **Add Date and Time for Collecting Waste**

  Collecting System is responsible for add date and time for collecting waste details.

- **View Selled Waste Materials**

  Collecting System can view details of the selled products and collect them.

- **Done Payments**

  Collecting System can done payments for collecting the waste material

User Module

User is an important actor in this system. User can sell waste materials and buy recycled products.. After the user successfully login to this website the user can perform the functionalities including:

- **User Registration**

  By entering all the details user can registered to the system.

- **User Login**

  By the Username and password user can login to the system.

- **User Profile Edit**

  User can edit their profile.

- **View Collecting System**

  View all the registered collecting system.

- **Sell Waste Materials**

  User can sell waste materials by entering the details of the waste materials.

- **Buy Recycled Products**

  User can buy recycled products from the collecting system.

- **View Date For Collecting Waste Materials**

  User can view the date for collecting waste materials by the collecting agents.

- **Make Payments**

  User can get payments from the collecting system for selled waste materials and also the user done payments for buying the recycled products.

3.2.3 Input Design

Input design is the process of converting user-oriented input into a computer based format. The goal of the designing input is to make data entry as easy and free from error. In Django, input to the system is entered through forms. A form is any surface on which information is to be entered, the nature of which is determined by what is already on that surface. If the data going into the system is incorrect, then processing and output will magnify these errors. So designer should ensure that form is accessible and understandable by the user.

End-users are people who communicate to the system frequently through the user interface, the design of the input screen should be according to their recommendations.

The data is validated wherever it requires in the project. This ensures only correct data is entered to the system. GUI is the interface used in input design. All the input data are validated in the order and if any data violates any condition the use is warned by a message and asks to re-enter data. If the data satisfies all the conditions then it is transferred to the appropriate tables in the database. This project uses text boxes and drop down to accept user input. If user enters wrong format then it shows a message to the user. User is never lift in confusion as to what is happening. Instead appropriate error messages and acknowledgments are displayed to the user.

3.2.4 Output Design

A quality output is one, which meets the requirement of the end user and presents the information clearly. In any system results of processing are communicated to the user and to the other systems through outputs. In the output design it is determined how the information is to be dis- played for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the systems relationship and helps user decision making.

A quality output is one, which meets the requirements of the end user and presents the information clearly. The objective of output design is to convey information about past activities, current status or projections of the future, signal important events, opportunities, problems, or warnings, trigger an action, confirm an action etc. Efficient, intelligible output design should improve the systems relationship with the user and helps in decisions making. In output design the emphasis is on displaying the output on a CRT screen in a predefined format. The primary consideration in de- sign of output is the information requirement and objectives of the end users. The major formation of the output is to convey the information and so its layout and design need a careful consideration. Two phases of the output design are:-

1. Output definition.
2. Output specification.

3.2.5 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 Database Management System (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.

Tables

Table 3.1: User
Primary key :id

FIELD	**DATA TYPE**	**DESCRIPTION**
Id	integer	Id of user table
first_name	string	First name of user
last_name	string	Last name of user
Email	email	Email of user
Username	string	Username of user
Password	string	Password of user

Table 3.2: UserRegister

Primarykey:id

Foreign key:username

FIELDS	DATA TYPE	DESCRIPTION
Id	interger	Id of table
Username	string	Username of user
Name	string	Name of user
Housenumber	int	Housenumber of user
State	string	State of user
District	string	District of user
City	string	City of user
Pin	int	Pin of user
Mobile	biginteger	Mobile of user
Usertype	string	Type of user

Table 3.3: CollectRegister

Primarykey:id

Foreignkey:username

FIELDS	DATA TYPE	DESCRIPTION
Id	integer	Table id
Username	email	Username of user
Cname	string	Name of collecting agent
State	string	State of collecting system
District	string	District of collecting system
City	string	City of collecting system
Pin	integer	Pin of collecting system
Mobile	biginteger	Mobile number of collecting system

Status	string	Status of collecting system

3.4Table Sell

Primary key : id

Foreign key :username,email,cname,wastetype

FIELD	**DATA TYPE**	**DESCRIPTION**
Id	integer	Id of table
Email	email	Email of user
Cname	string	Name of collecting agent
Wastetype	string	Type of waste
Weight	float	Weight of waste
Price	float	Price of waste
Status	string	Status of booking

3.5Table AddCategory

Primary key:id

FIELD	**DATA TYPE**	**DESCRIPTION**
Id	integer	Id of table
Categoryname	string	Category of waste type

3.6 Table AddRecycle

Primary key :id

FIELD	**DATA TYPE**	**DESCRIPTION**
Id	integer	Id of the table
Categoryname	string	Name of recycled product

Table 3.7: AddProduct

Primarykey :id

Foreignkey:productname,user

FIELD	**DATA TYPE**	**DESCRIPTION**
Id	Int	Id of the table
User	String	Name of user
Productname	String	Name of recycled product
Weight	Float	Weight of product
Price	Float	Price of product
Status	String	Status of booking
Date	Date	Date of adding product

Table 3.8: CollectDetails

Primary key:id
Foreign key:username,cname,email,wastetype

FIELDS	**DATA TYPE**	**DESCRIPTION**
Id	Integer	Id of the table
Username	String	Name of user
Email	Email	Email of user
Cname	String	Name of collecting agent
Wastetype	String	Name of wastetype
Weight	Float	Weight of waste
Price	Float	Price of weight
Date	Datetime	Date and time

Table 3.6: District Details

3.9Table AccountDetails

Primary key id

FIELD	DATA TYPE	DESCRIPTION
id	Integer	Account id
name	String	Name of the user
address	String	Address of user
accountNo	Integer	Account number
cardNo	Integer	Card number
cvv	Integer	Cvv number
transactionPassword	Integer	Transaction password

3.10Table TransactionDetails

Primary key: id

FIELD	DATA TYPE	DESCRIPTION
id	integer	Transaction id
fromAccountNo	integer	From which account its number
toAccountNo	integer	To which account its number
amount	integer	Amount to which transfer
remarks	String	Remarks

3.2.6 Data Flow diagram

Data flow diagram is the graphical representation of the system. It is a network that uses special symbols to describe the flow of data and process that transforms data throughout the system. Data flow diagram is a way of representing system requirements in a graphic form. A DFD also known as Bubble Chart has the purpose of clarifying system requirements and identifies major transformations that will become program in system design. So it is the starting point of design phase that functionally decomposes the requirements specifications down to the lowest level of details. A DFD consist of series of bubbles joined by lines. The bubbles represent data transformation and the lines represent data flow in the system.

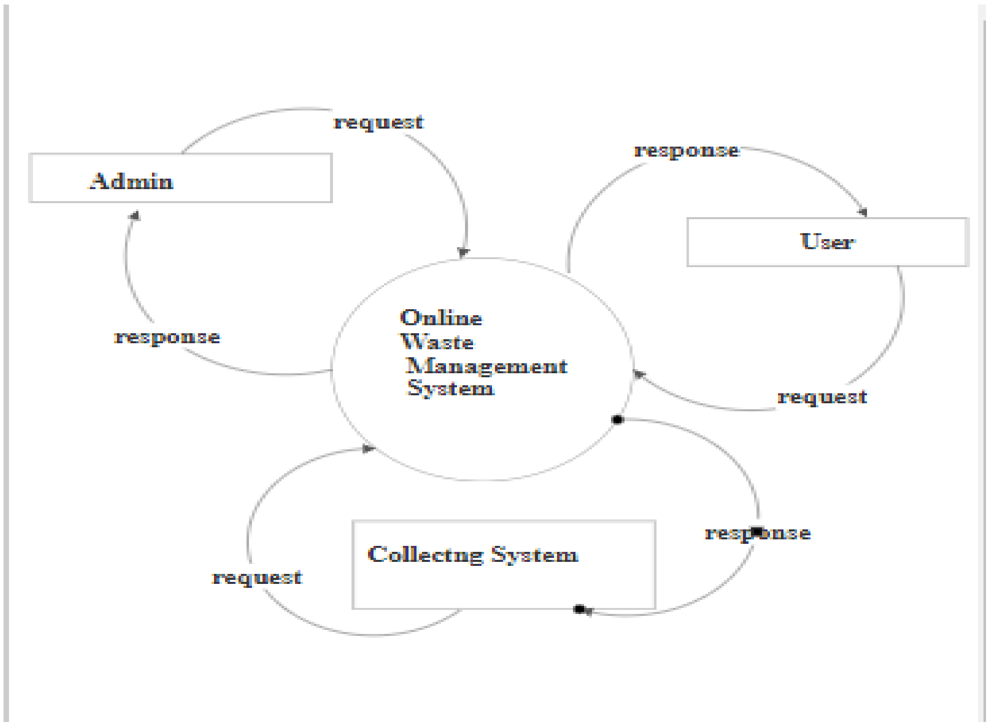


Fig. 3.1: LEVEL-0 DFD

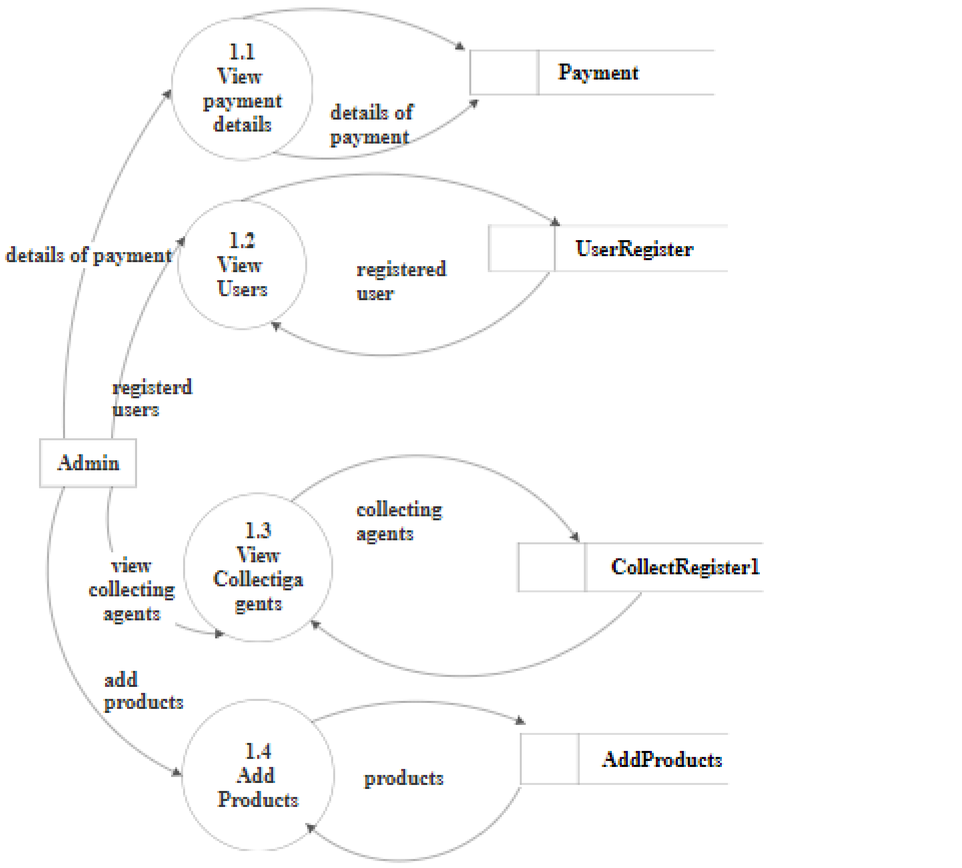


Fig. 3.2: LEVEL-1 DFD Admin

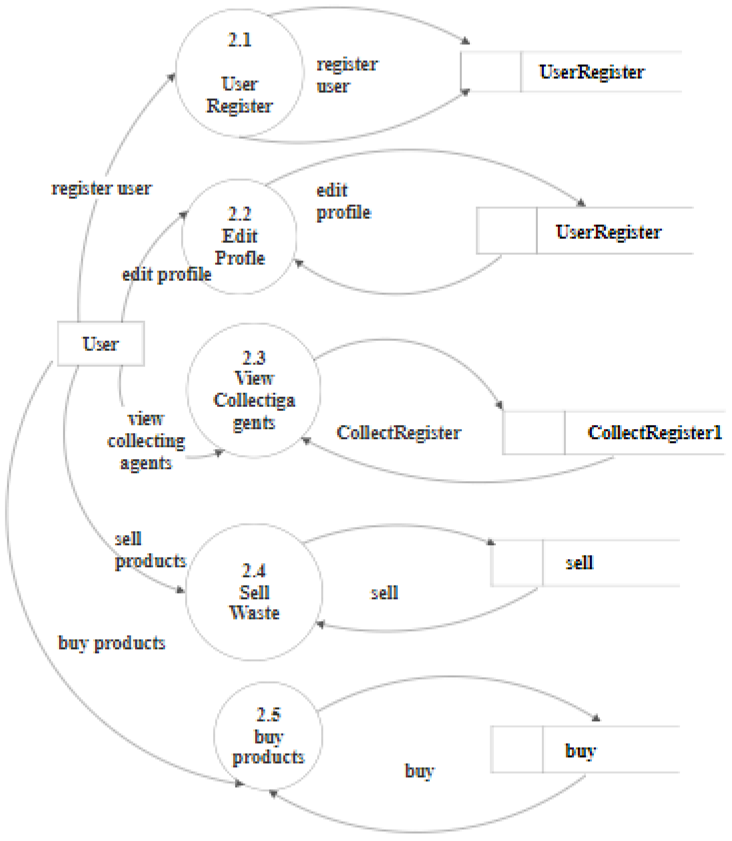


Fig. 3.3: LEVEL-1 DFD for user

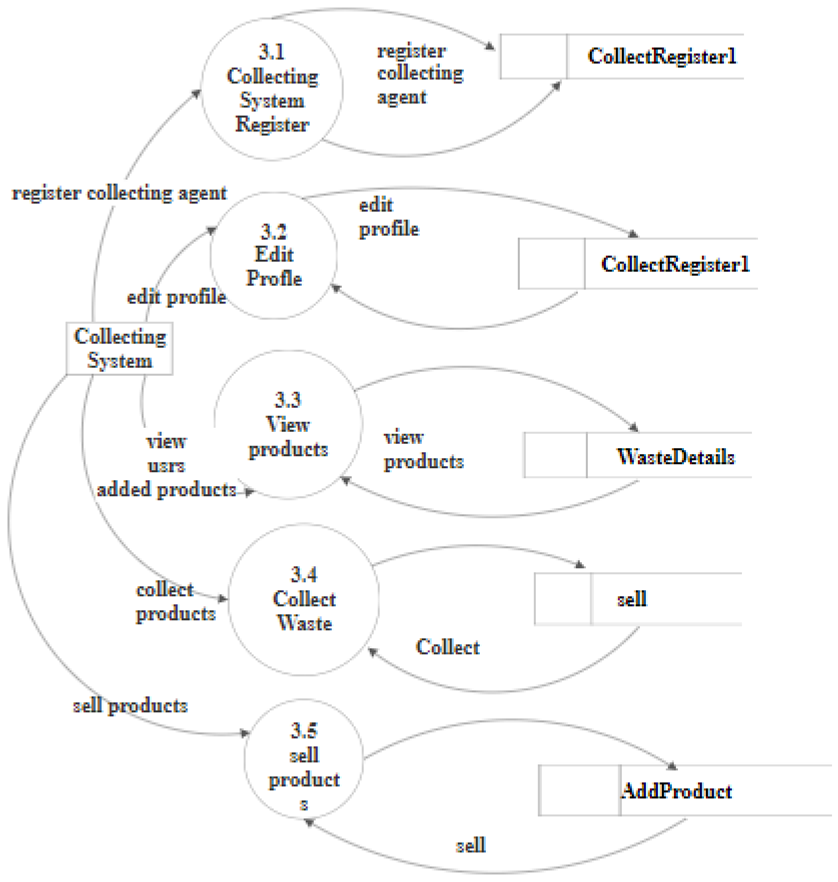


Fig. 3.4: LEVEL-1 DFD for Collecting System

3.3 UNIFIED MODELING LANGUAGE[UML]

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

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

Improved integration between structural models like class diagrams and behavior models like activity diagrams. The original UML specified nine diagrams; UML 2.x brings that number up to 13.

The four new diagrams are called: communication diagram, composite diagram, interaction overview diagram and timing diagram. It also renamed state chart diagrams to state machine diagrams , also known as state diagrams.

Types of UML diagrams

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

Structural UML diagrams

- Class diagram
- Package diagram
- Object diagram
- Component diagram
- Composite structure diagram
- Deployment diagram

Behavioral UML diagrams

– Activity diagram

– Sequence diagram

– Use case diagram

– State diagram

– Communication diagram

– Interaction overview diagram

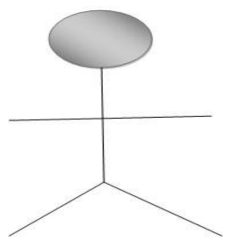
– Timing diagram

3.3.1 Use case Diagram

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

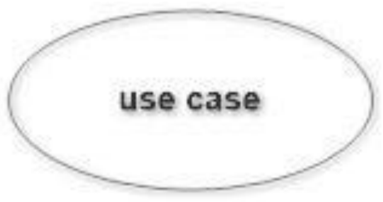
- Actor
- Use case
- Use case diagram
- System
- Package

Actor



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

Use case



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

System

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

Package

Package is another optional element that is extremely useful in complex diagrams. Similar to use class diagrams, packages are used to group together use cases.

The following is the UML diagram of this system :-

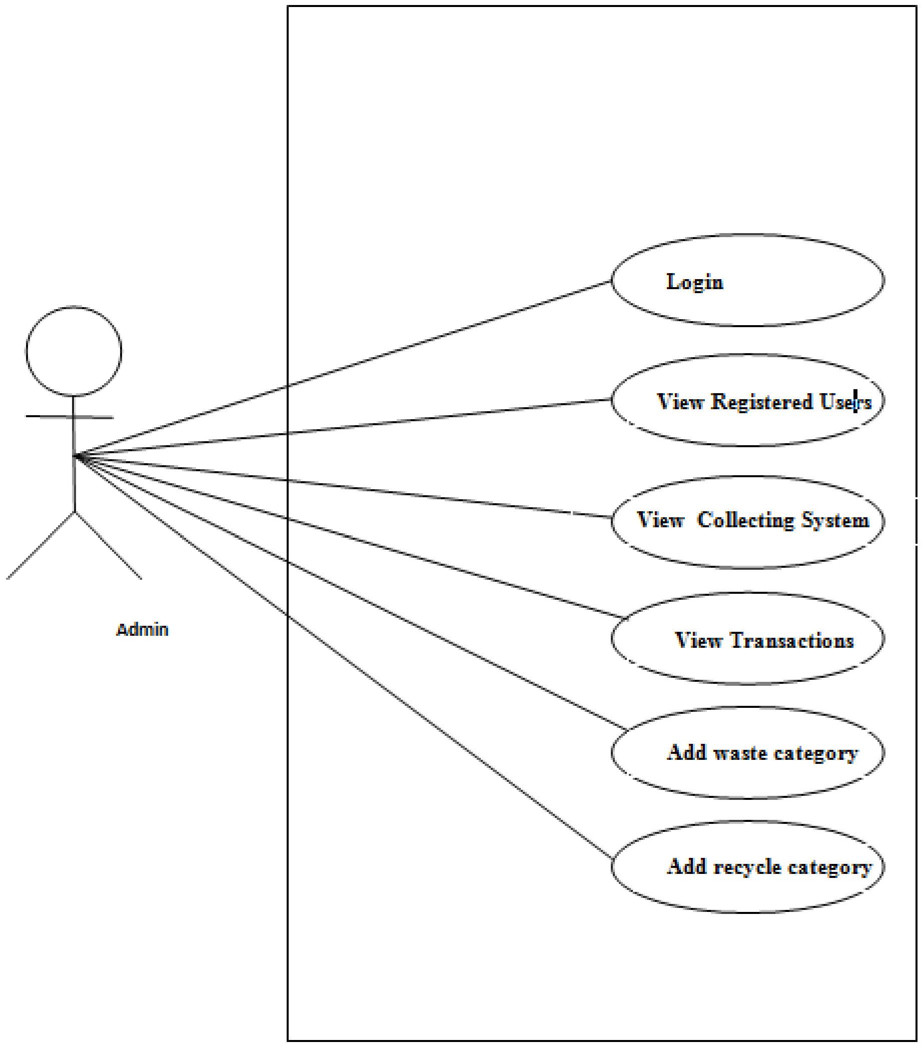


Fig. 3.5: Usecase Diagram for Admin

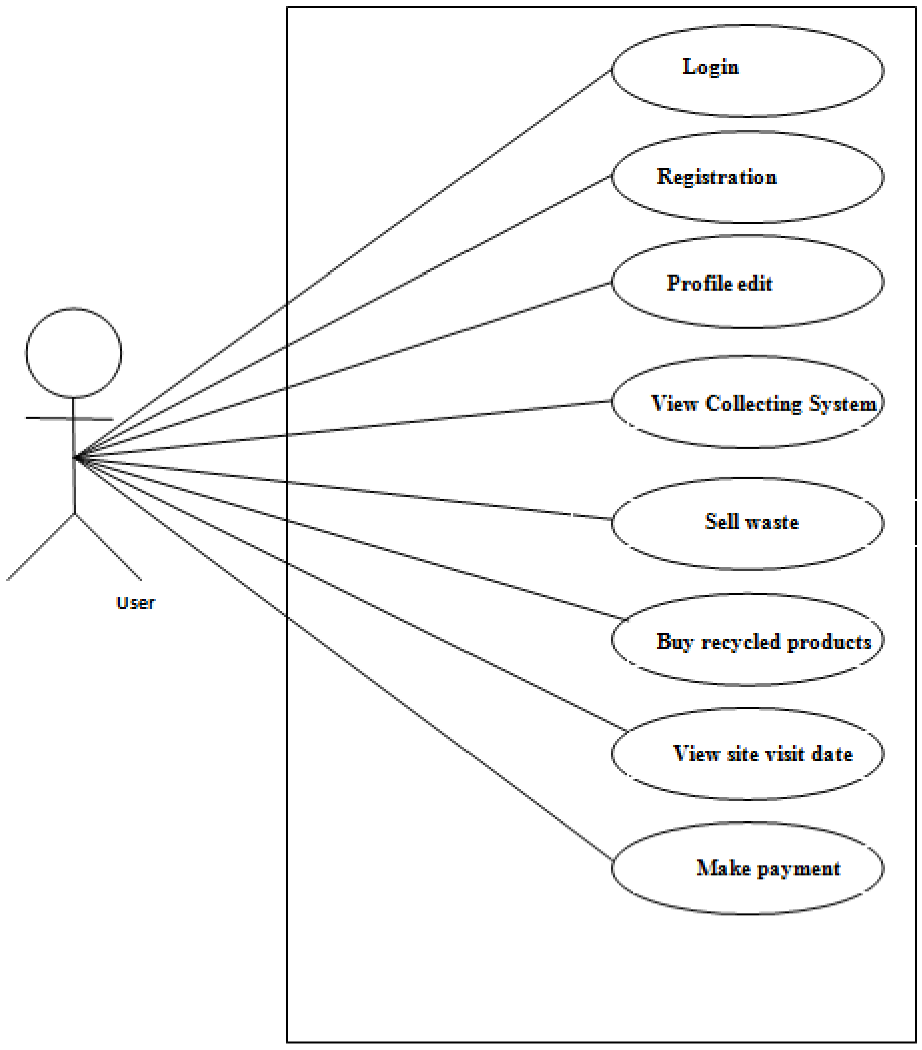


Fig. 3.6: Usecase Diagram for User

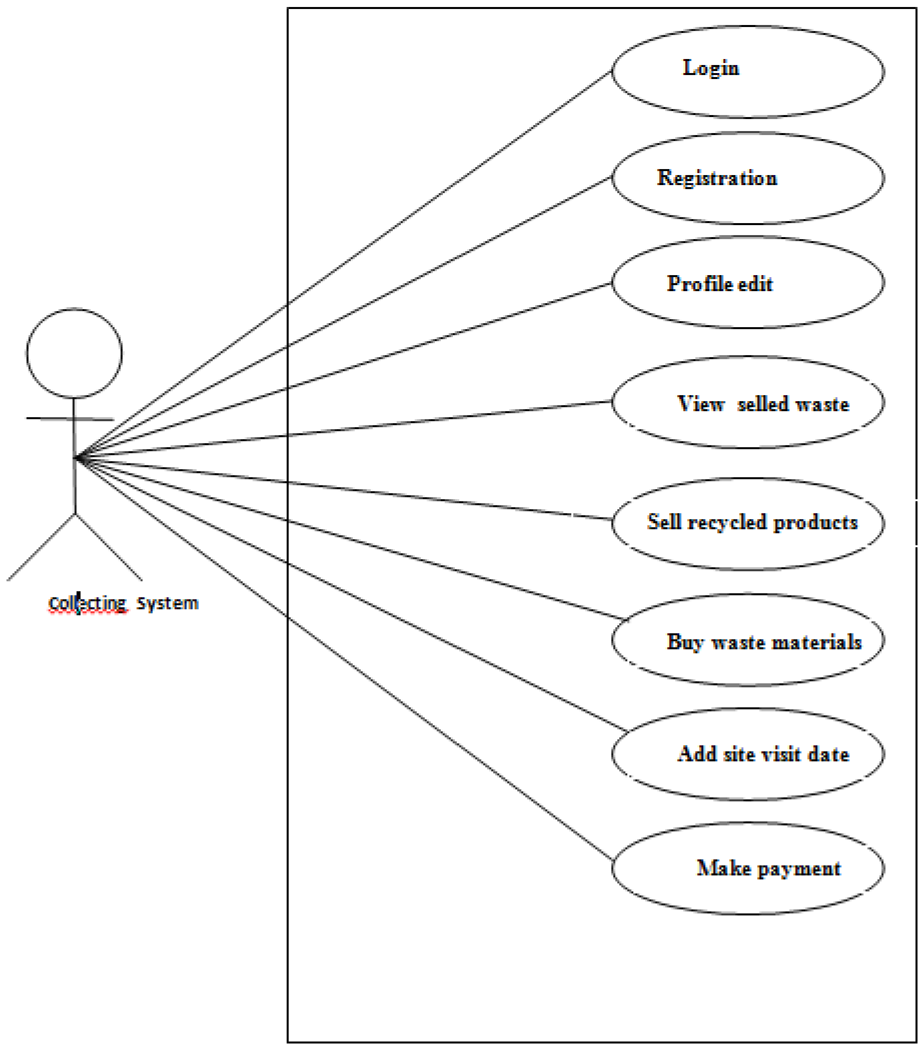


Fig. 3.7: Usecase Diagram for Collecting System

3.3.2 Sequence Diagram

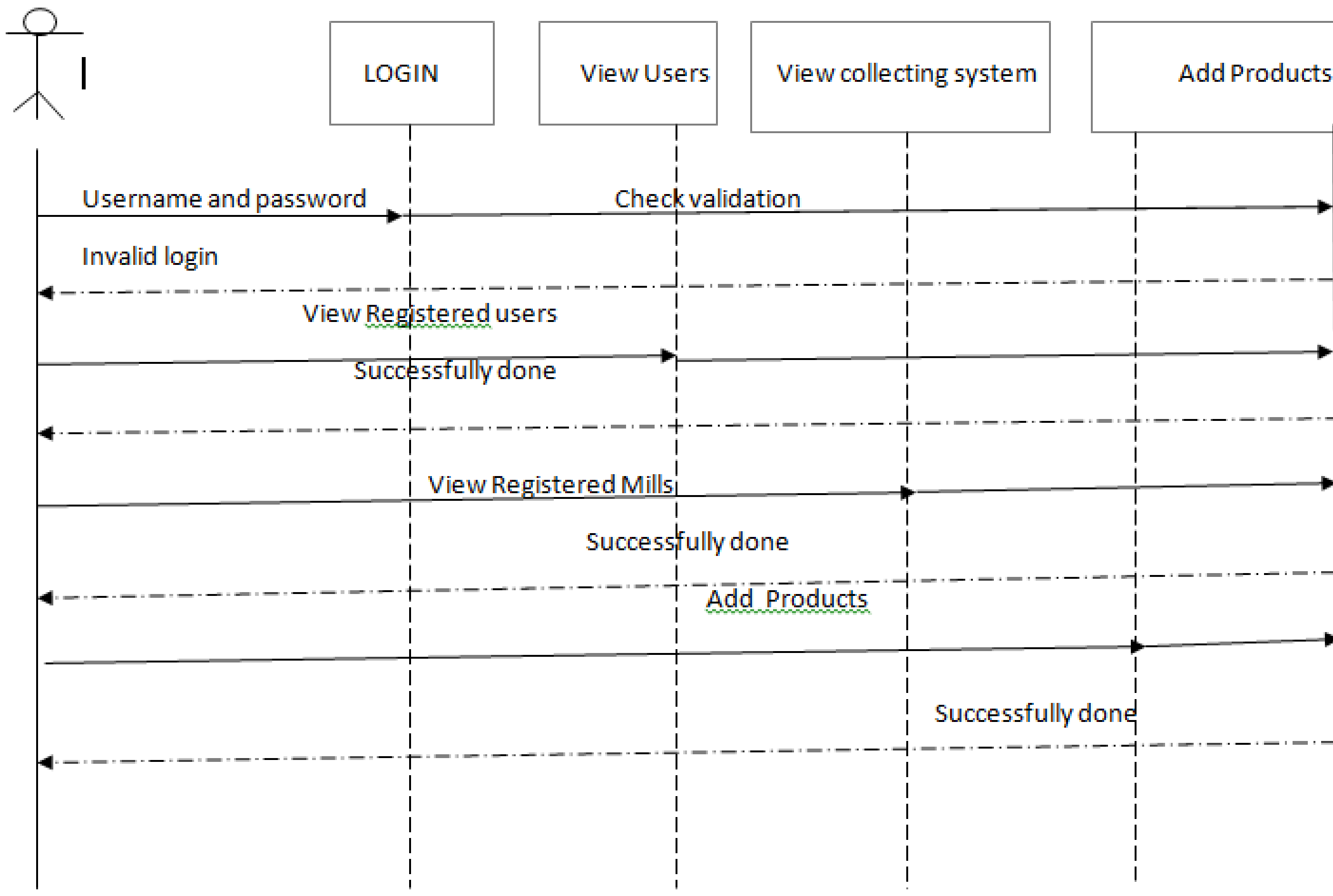
UML sequence diagrams are used to represent or model the flow of messages, events and actions between the objects or components of a system. Time is represented in the vertical direction showing the sequence of interaction of the header elements.

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

Although UML sequence diagrams are typically used to describe object-oriented software systems, they are also extremely useful as system engineering tools to design system architectures in business process, as message sequence charts and call flows for telecoms or wireless system design, and for protocol stack design and analysis.

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

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.


LOGIN
View Users
View collecting system
Add Products
Username and password
Check validation
Invalid login
View Registered users
Successfully done
View Registered Mills
Successfully done
Add Products
Successfully done

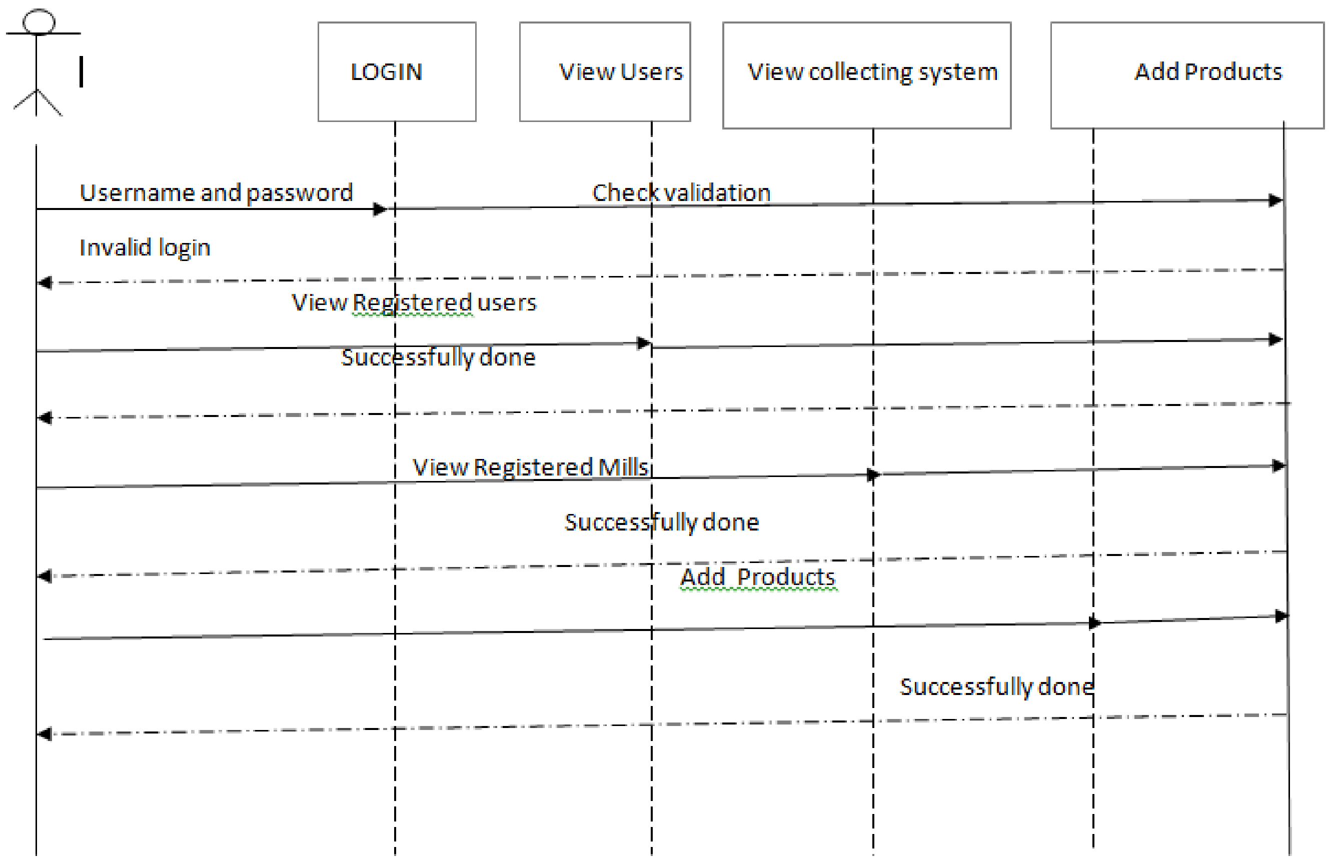


Fig. 3.8: Sequence Diagram for Admin

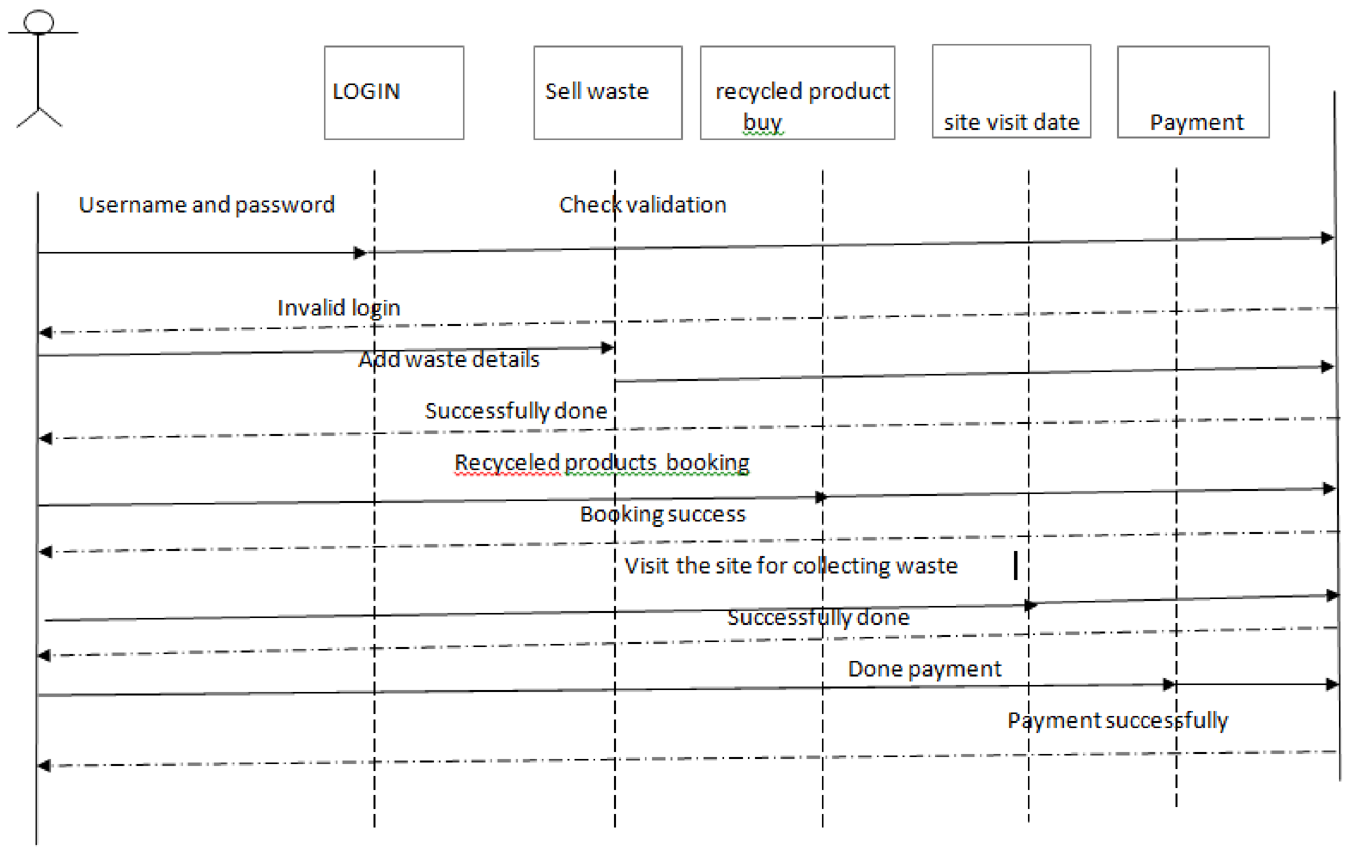


Fig. 3.9: Sequence Diagram for User

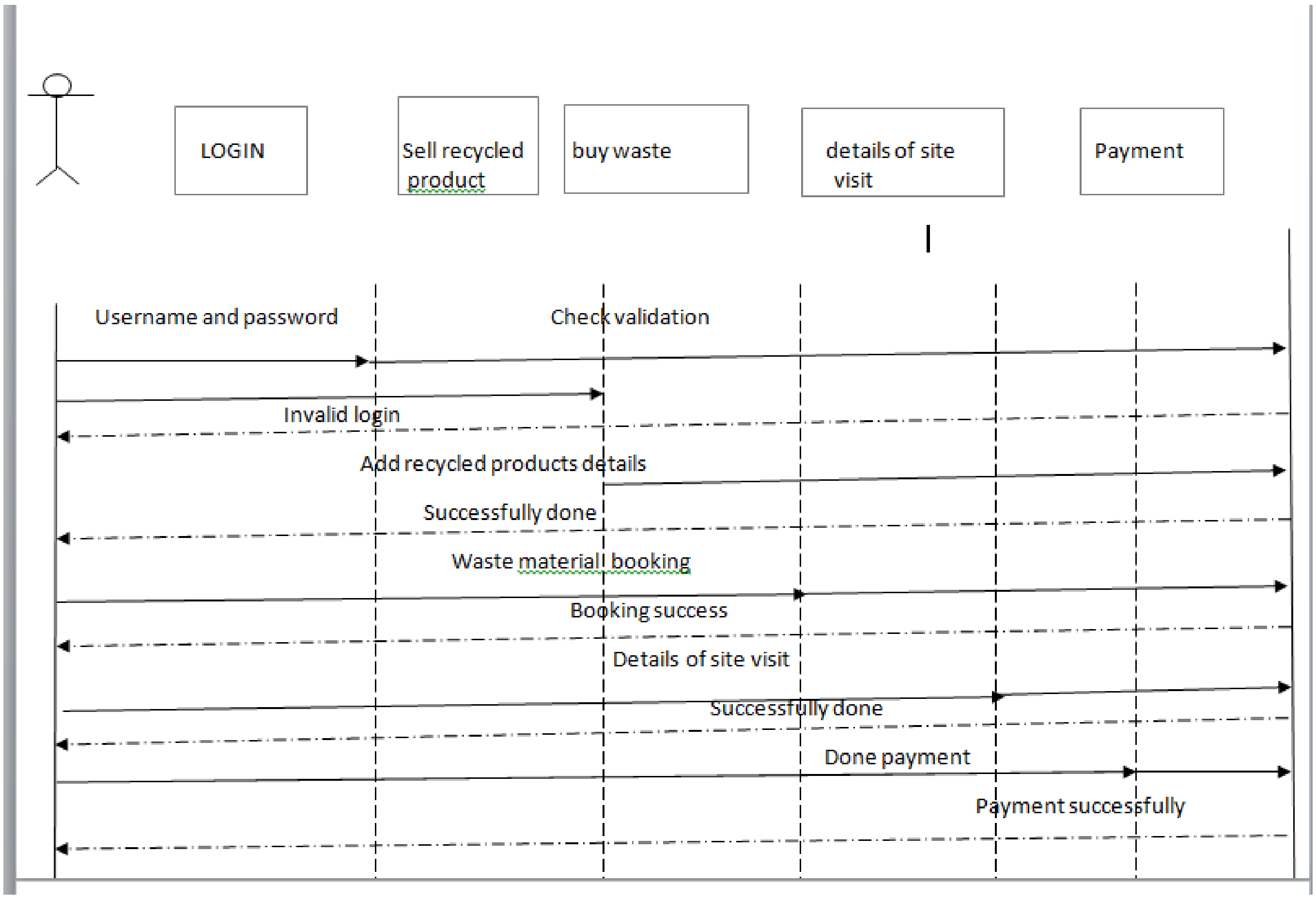


Fig. 3.10: Sequence Diagram for Collecting System

3.3.3 Activity Diagram

Activity diagram is another important diagram in UML to 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.

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

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

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

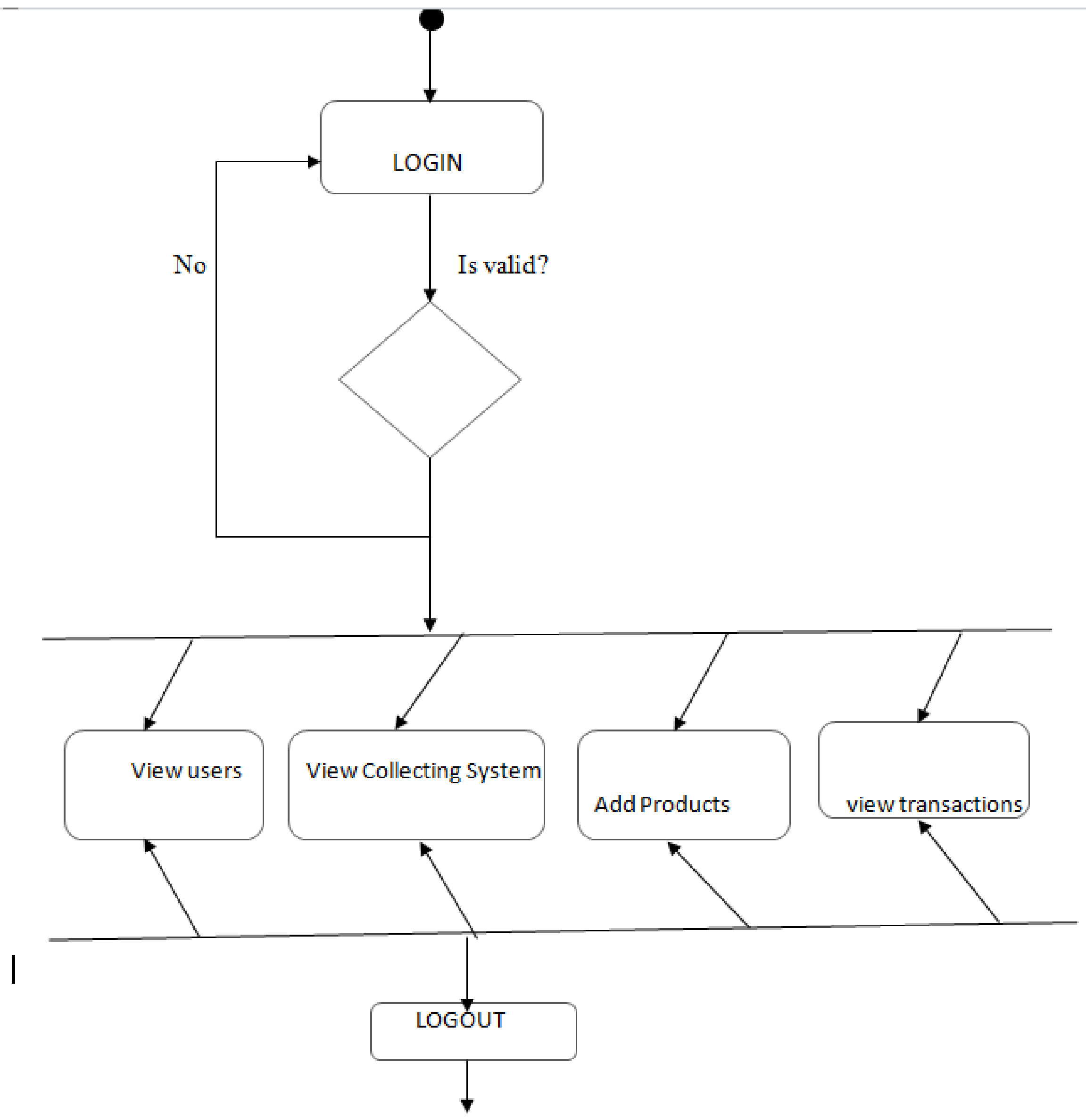


Fig. 3.11: Activity Diagram for Admin

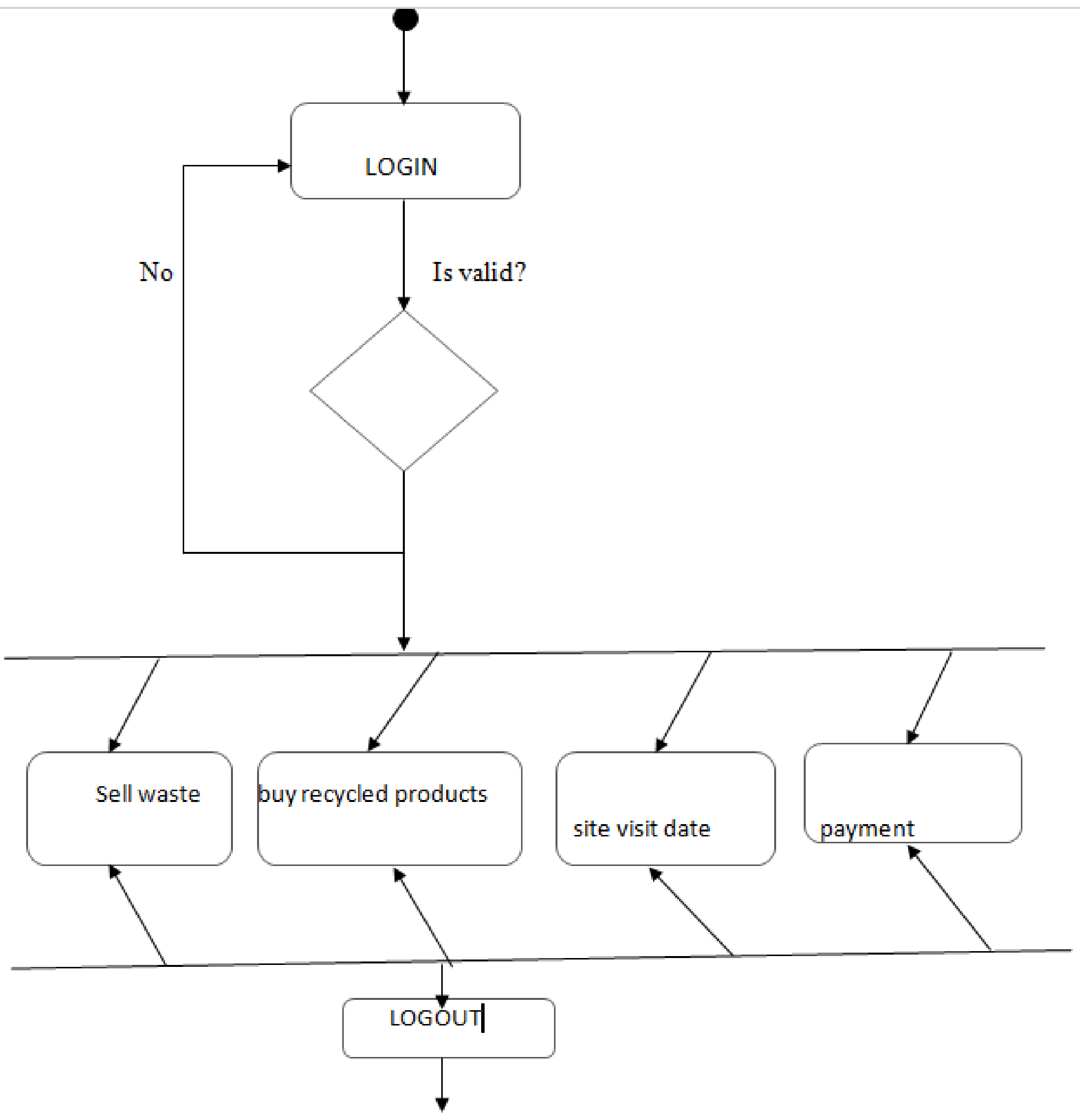


Fig. 3.12: Activity Diagram for User

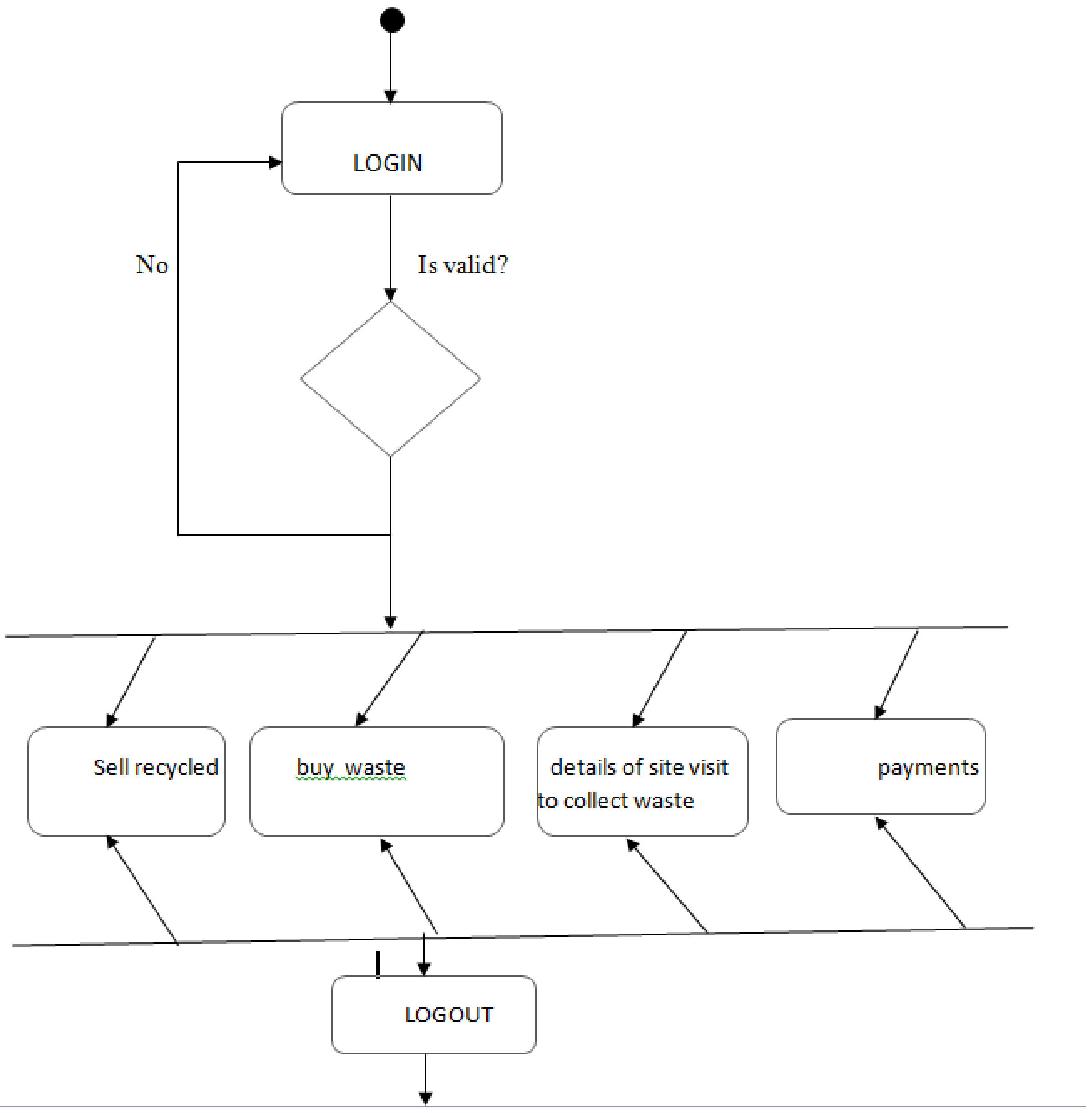


Fig. 3.13: Activity Diagram for Collecting System

3.4 TOOLS AND PLATFORMS

3.4.1 Introduction to Python

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

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

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

.

.

3.4.2 Python Environment

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

.

3.4.3 Python Language

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

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

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

.

3.4.4 Python Runtime Environment

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

.

3.4.5 Python Tools

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

3.4.6 Python Application

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

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

3.4.7 Django Framework

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

Django helps you write software that is:

Complete

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

Versatile

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

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

Secure

Django helps developers avoid many common security mistakes by providing a framework that has been engineered to "do the right things" to protect the website automatically. For example, Django provides a secure way to manage user accounts and passwords, avoiding common mistakes like putting session information in cookies where it is vulnerable (instead cookies just contain a key, and the actual data is stored in the database) or directly storing passwords rather than a password hash.

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

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

Scalable

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

Maintainable

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

Portable

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

3.4.8 SQLite3

SQLite3 is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL data base engine . The code for SQLite3 is in the public domain and is thus free for use for any purpose, commercial or private . SQLite3 is the most widely deployed database in the world with more applications than we can count, including several high-profile projects.

3.4.9 Normalization

Normalization refers how to implement the relationships and storage of data in the database tables. Keys are used to uniquely define a relationship to another instance or set of information. In first normal form each value in the database table is atomic or represented only once. In second form each instance or raw in the database table must be uniquely identifiable. The table in the third normal form wont have redundant non key information.

The SQLite3 provides the ability to create roles so that security permission granted to all members is the same. Roles are defined on a database which means that when a role is created in database its not available in other. In standard role user are assigned to the role and the role is granted. Permission on database objects. This can be significantly reducing the number of needed logins to the server to only those users who need direct access to tables.

SQLite3 is the most advanced, trusted, and scalable data platform released to date. Building on the success of the original SQLite3, SQLite3 has made an impact an impact on organizations worldwide with its groundbreaking capabilities, empowering end users through self-services business intelligent(BI),bolstering efficiency and collaboration between database administrators(DBAs) and application developers, and scaling to accommodate the most Demanding data work loads.

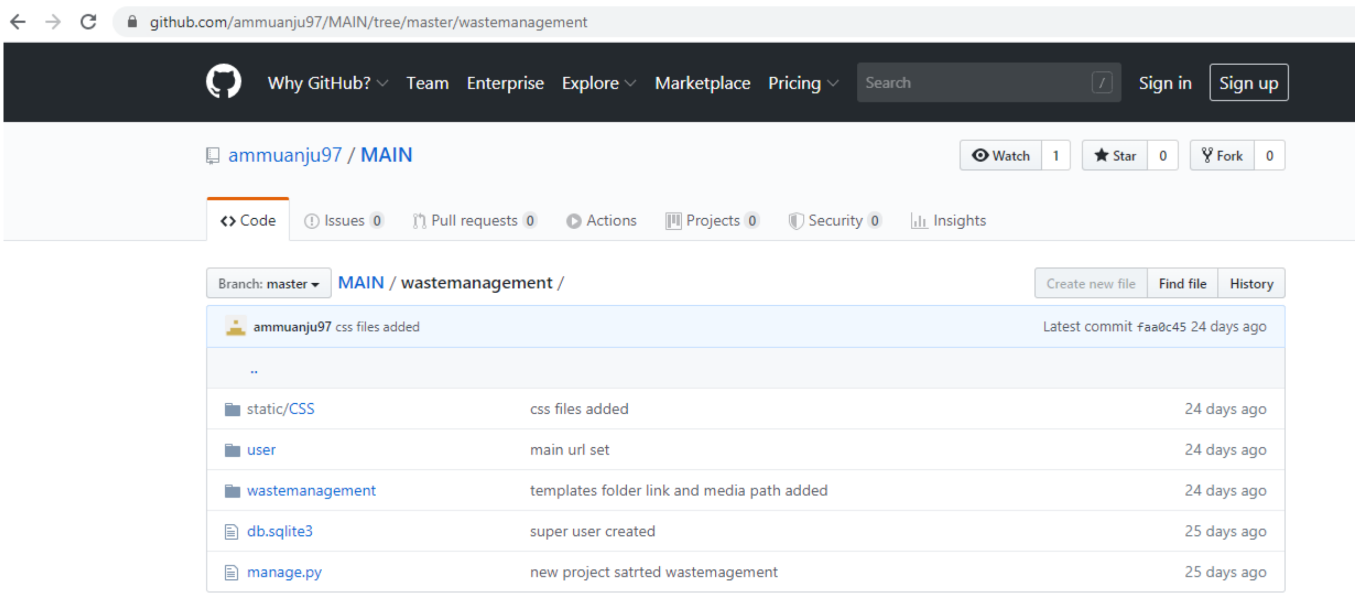
.

3.5 GITHUB

Version control is a system that manages changes to a file or files.These changes are kept as logs in a history, with detailed information on what file(s) was changed, what was changed within the file, who changed it, and a message on why the change was made. This is extremely useful, especially when working in teams.To understand how incredibly powerful version control is How many files of different versions of a manuscript or thesis do you have laying around after getting feedback from your supervisor or co-authors?

Have you ever wanted to experiment with your code or your manuscript and need to make a new file so that the original is not touched ?Have you ever deleted something and wish you hadn't ? Have you ever forgotten what you were doing on a project ? All these problems are fixed by using version control (git)!

Git Hub History



github.com/ammuanju97/MAIN/tree/master/wastemanagement

ammuanju97 / MAIN

MAIN / wastemanagement /

ammuanju97 css files added — Latest commit faa0c45 24 days ago

..		
static/CSS	css files added	24 days ago
user	main url set	24 days ago
wastemanagement	templates folder link and media path added	24 days ago
db.sqlite3	super user created	25 days ago
manage.py	new project satrted wastemagement	25 days ago

In this project **”ONLINE WASTE MANAGEMENT”**, maintained a Git Hub Repository to store the whole project details to know about the changes made. https://github.com/ammuanju97/MAIN/wastemanagement

Fig. 3.14: Git History

Chapter 4

SYSTEM TESTING

4.1 TESTING METHODOLOGIES AND STRATEGIES

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

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

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

4.1.1 Unit Testing

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

4.1.2 Integration Testing

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

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

4.1.3 System Testing

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

The perquisites for System Testing are:-

- All the components should have been successfully Unit Tested.
- All the components should have been successfully integrated.
- Testing should be completed in an environment closely resembling the production environment. When necessary iterations of System Testing are done in multiple environments.

4.1.4 User Acceptance Testing

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

Chapter 5

SYSTEM IMPLEMENTATION

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

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

Chapter 6

CONCLUSION

The project titled **”ONLINE WASTE MANAGEMENT SYSTEM”**, this system has developed for selling waste materials and buying recycled products. The project is developed as a Web Application by using Python Django as the front end and SQLite as the back end. Each user of the system has role and permission according to their role. This system helps user to sell their waste materials and buy the recycled products an also done and get payment from collecting system. Collecting system will collect the waste materials selled by user and sell the recycled products and done payments. Admin is the overall controller of the system they can add recycled and waste materials.

E-resources serve as a platform for random access to multiple users at the same time and save plenty of time. Online Waste Management has developed very fast due to internet. This project support to remove waste items from the world and recycled them then it can be again used by the user. This project helps user to manage their waste materials. So it will easy use the users to manage their waste at any time with the help of this website.

REFERENCES

1. https://docs.python.org/3/
2. https://docs.djangoproject.com/en/2.2/
3. https://django-crispy-forms.readthedocs.io/en/latest/
4. https://www.w3schools.com/
5. https://getbootstrap.com/docs/4.0/getting-started/introduction/
6. https://stackoverflow.com/

.

Appendix A

APPENDICES

A.1 SCREEN SHOTS INPUT FORM,OUTPUT FORMS

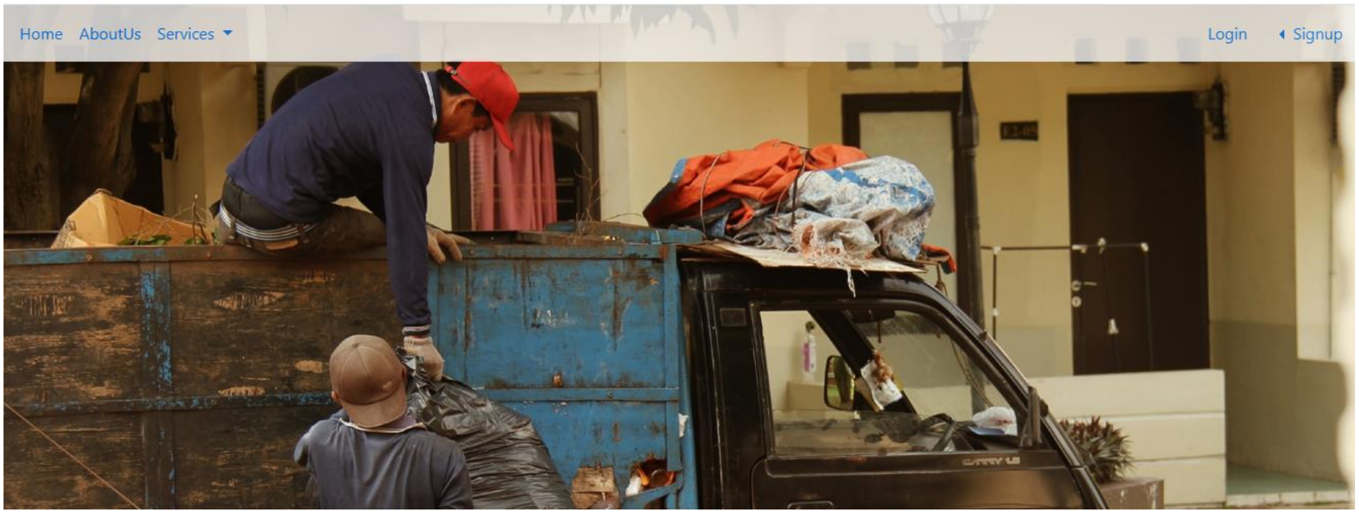
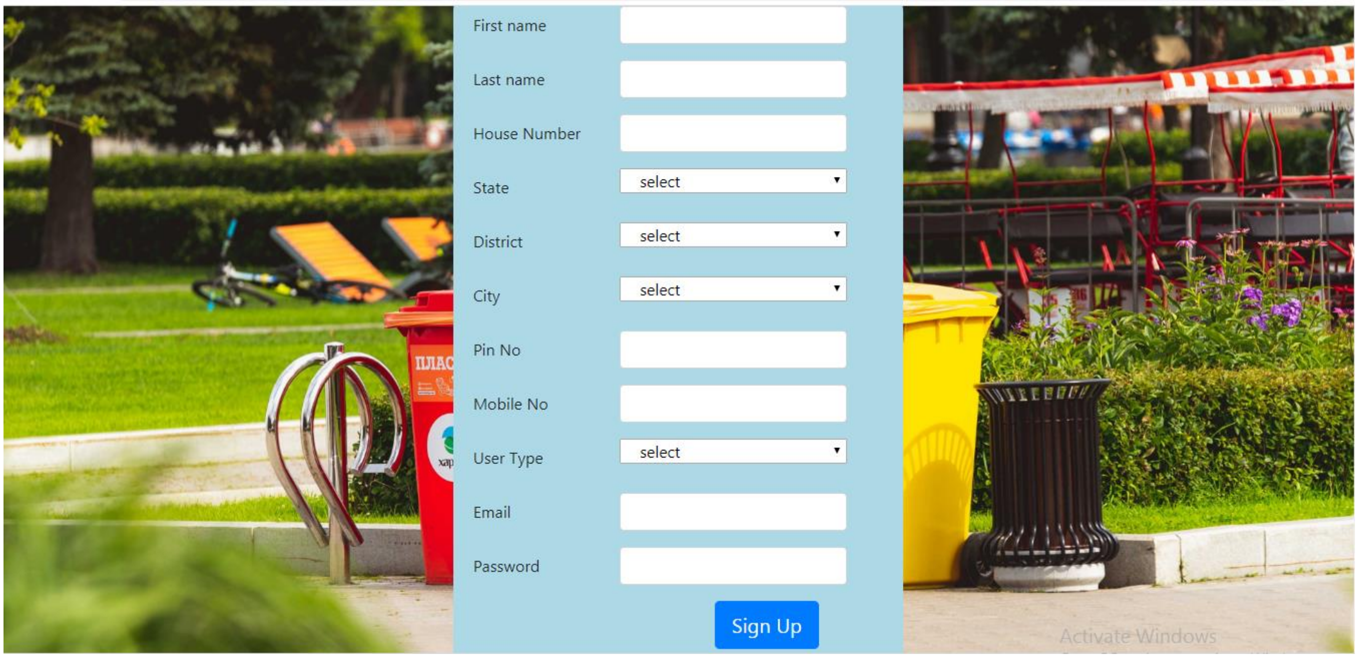
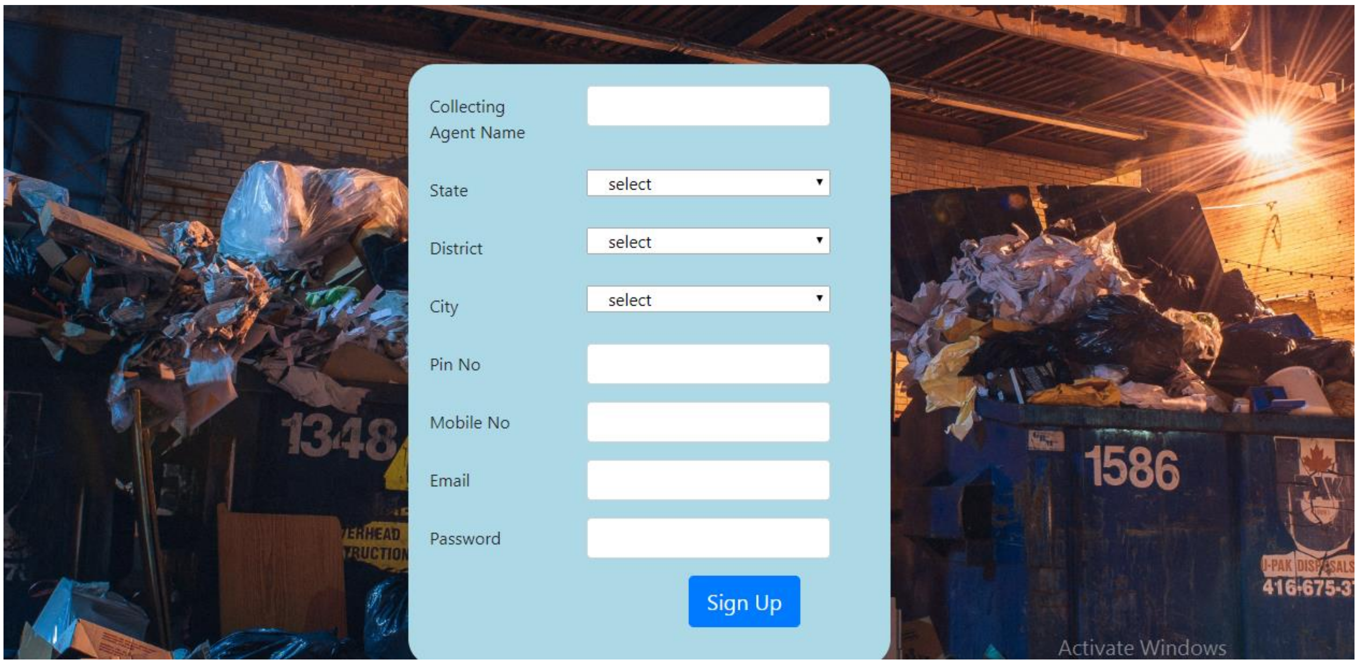


Fig. A.1: Home Page



First name	
Last name	
House Number	
State	select
District	select
City	select
Pin No	
Mobile No	
User Type	select
Email	
Password	

Fig. A.2: User Registration



Collecting Agent Name	
State	select
District	select
City	select
Pin No	
Mobile No	
Email	
Password	

Fig. A.3 Collecting System Register

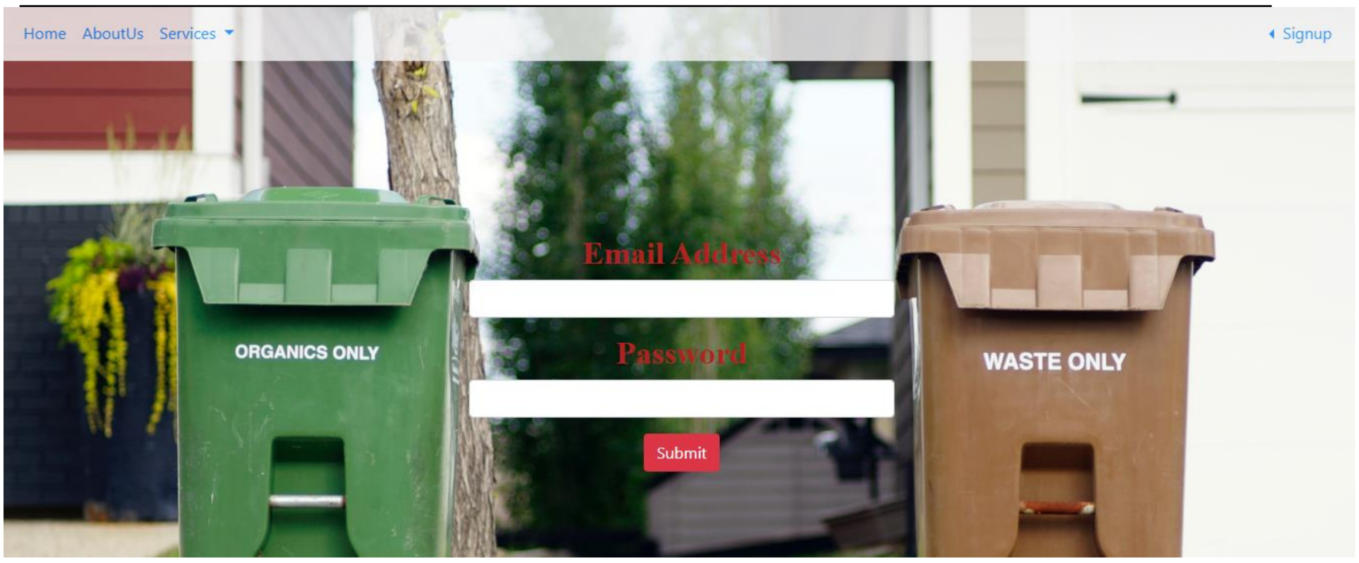


Fig. A.4 Login Page

Admin Functionalites :-

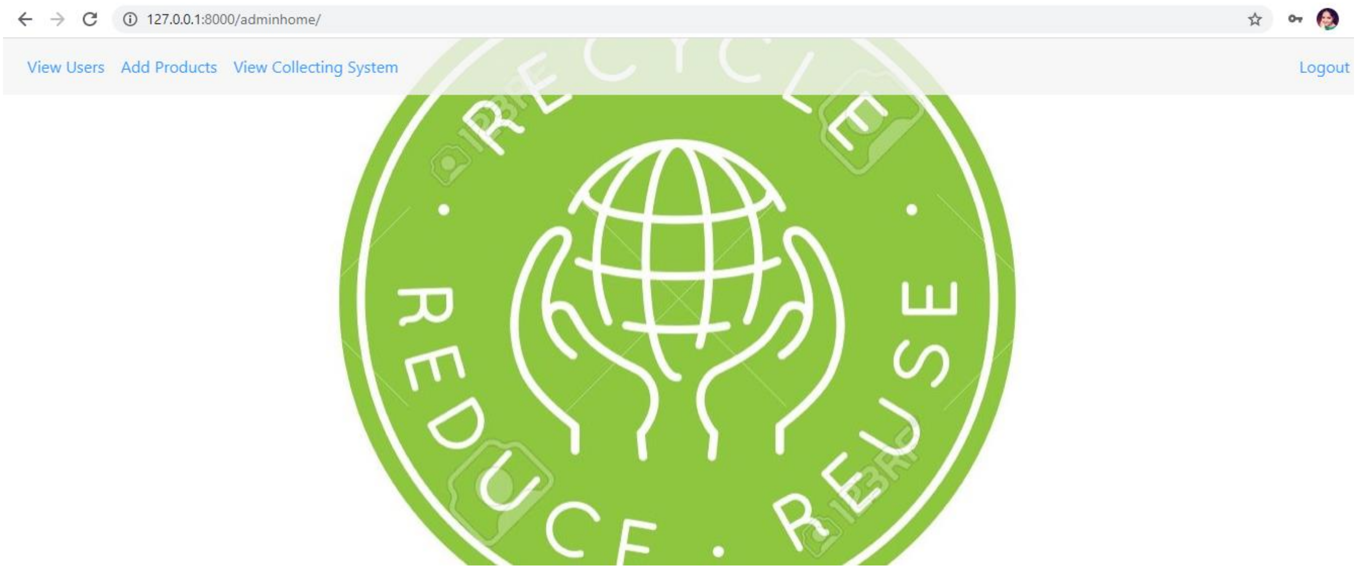
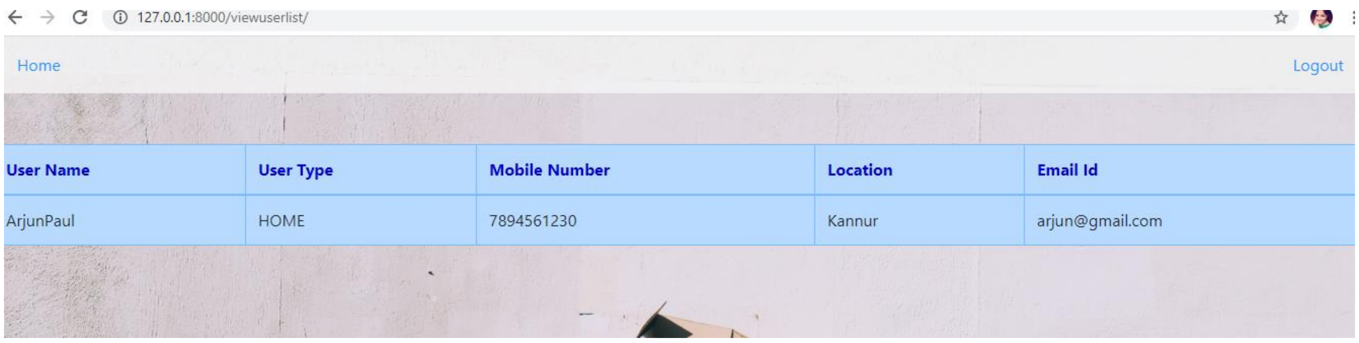
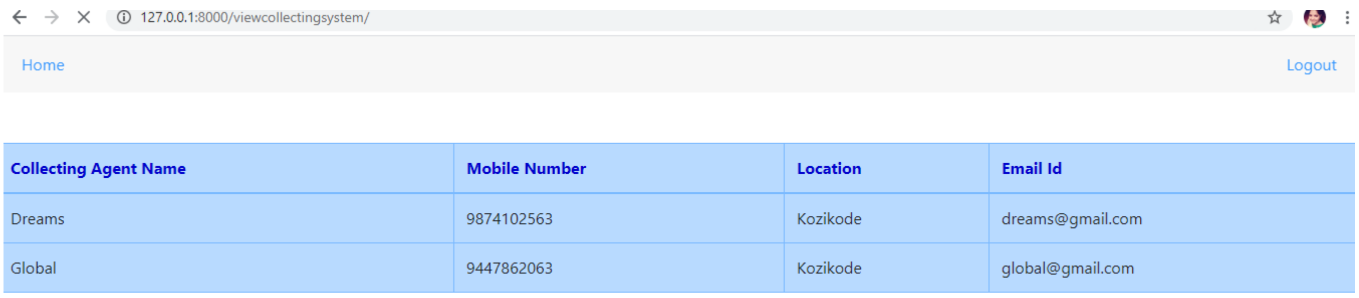


Fig. A.4: Admin Home Page



User Name	User Type	Mobile Number	Location	Email Id
ArjunPaul	HOME	7894561230	Kannur	arjun@gmail.com

Fig. A.4: View Users



Collecting Agent Name	Mobile Number	Location	Email Id
Dreams	9874102563	Kozikode	dreams@gmail.com
Global	9447862063	Kozikode	global@gmail.com

Fig. A.5: View Collecting System

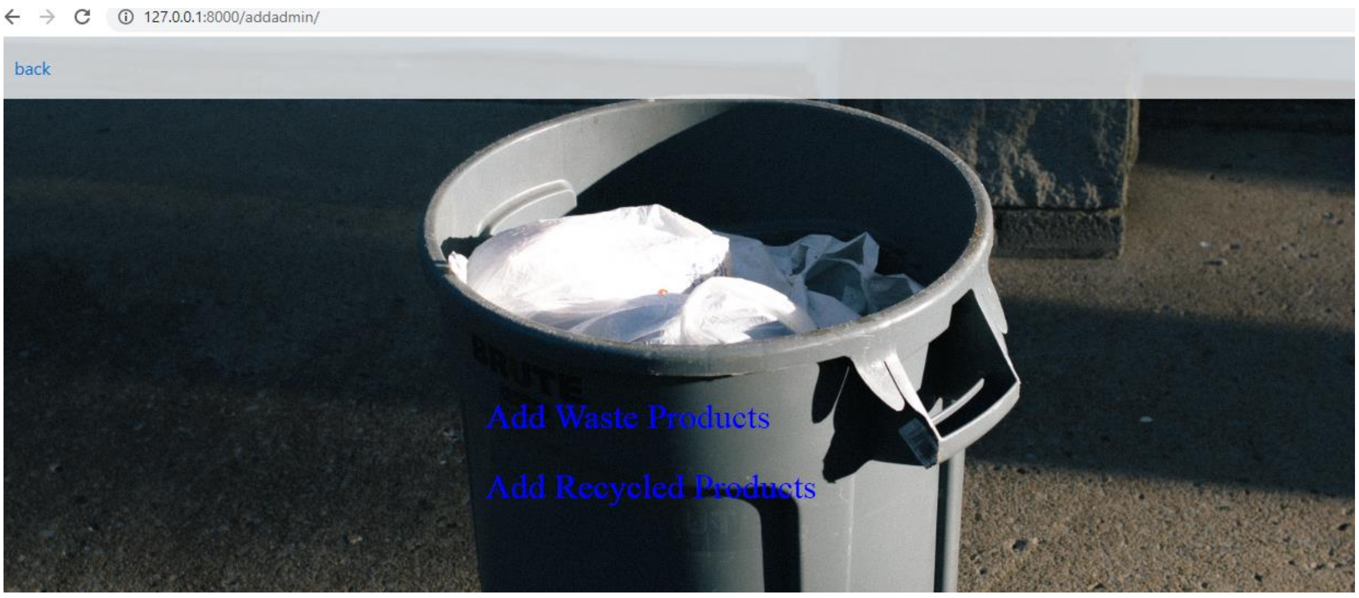


Fig.A.6 Category Add Page

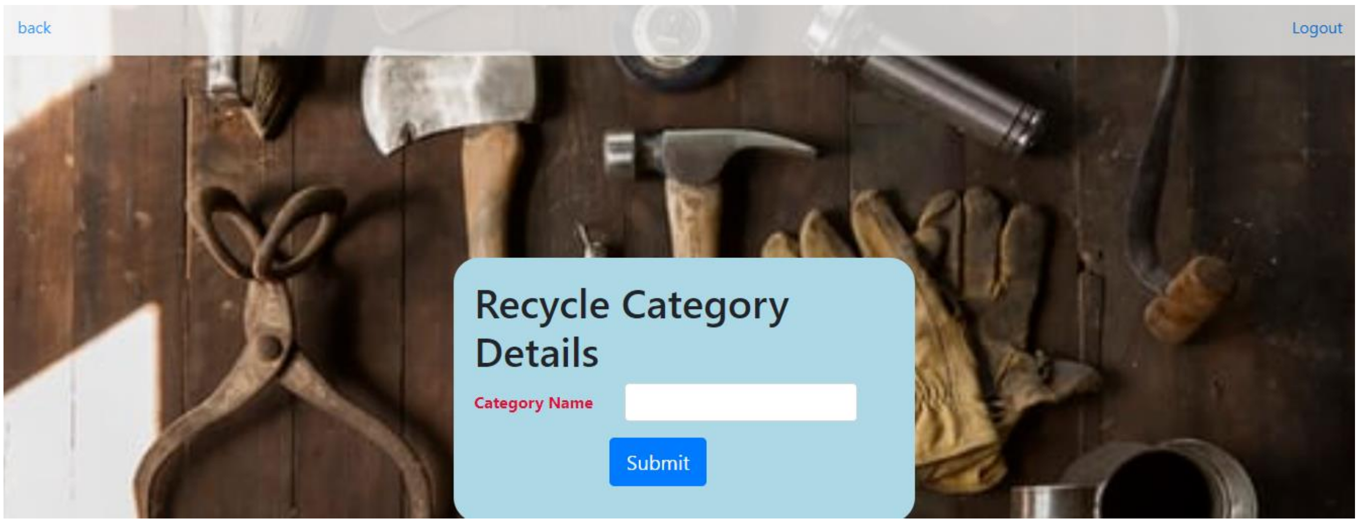


Fig.A.7 Add Recycle Products

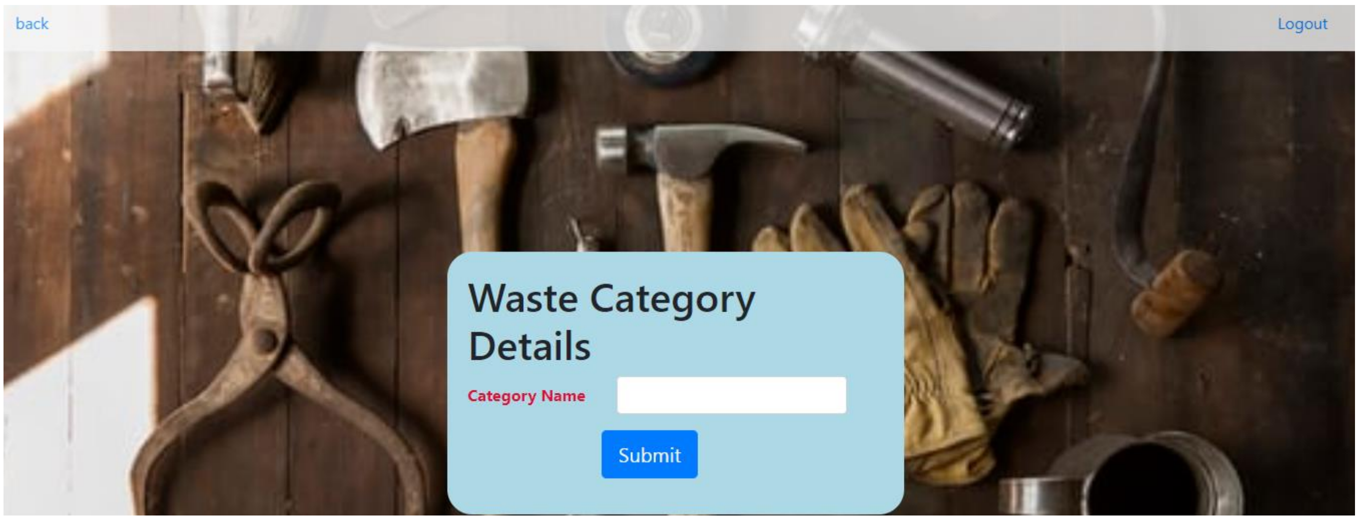


Fig.A.8 Add Waste Product

User Functionalites :-

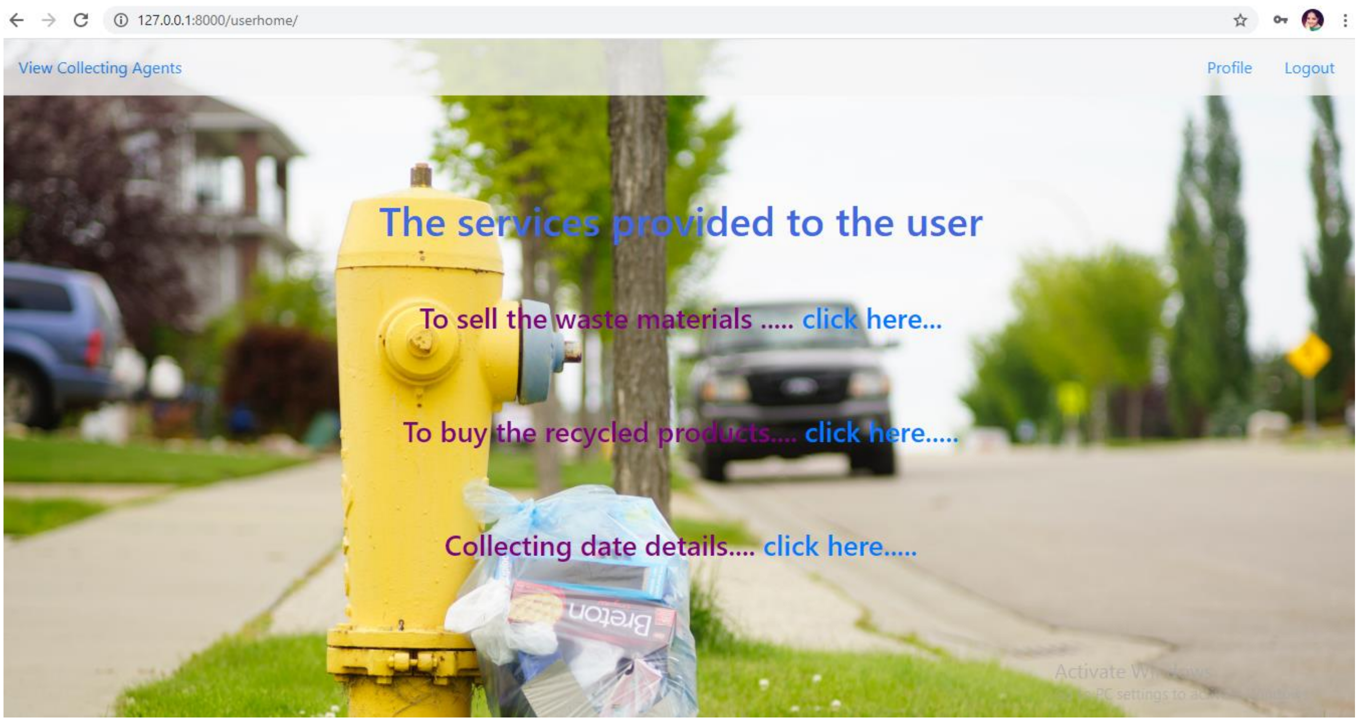

127.0.0.1:8000/userhome/
View Collecting Agents
Profile
Logout
The services provided to the user
To sell the waste materials ..... click here...
To buy the recycled products.... click here.....
Collecting date details.... click here.....

Fig.A.9 User home

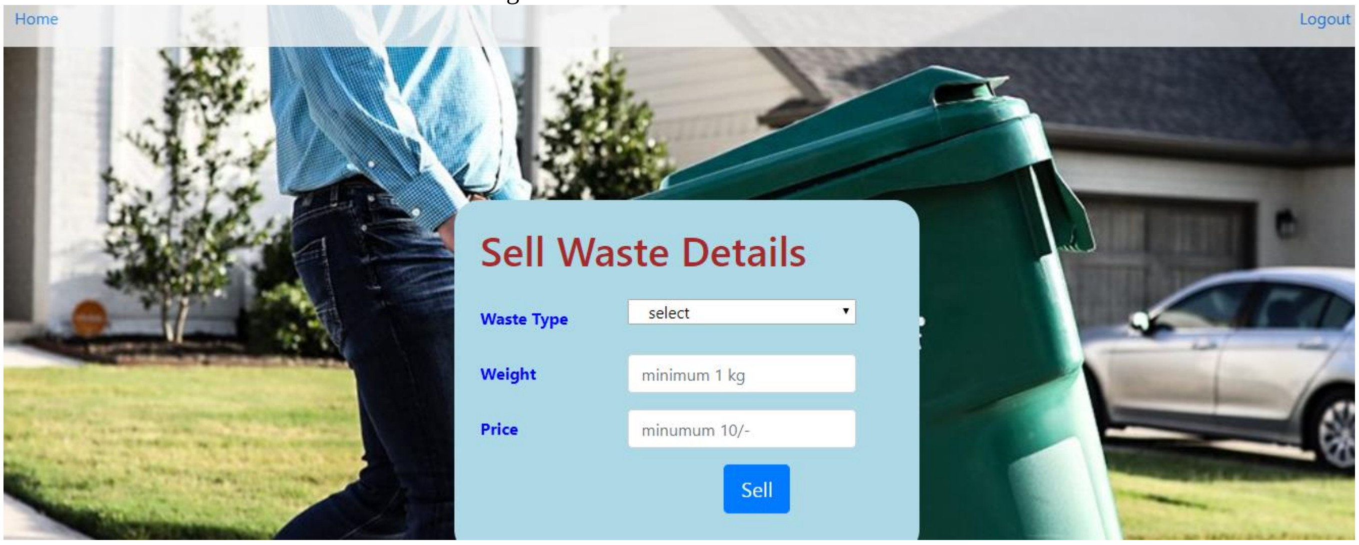
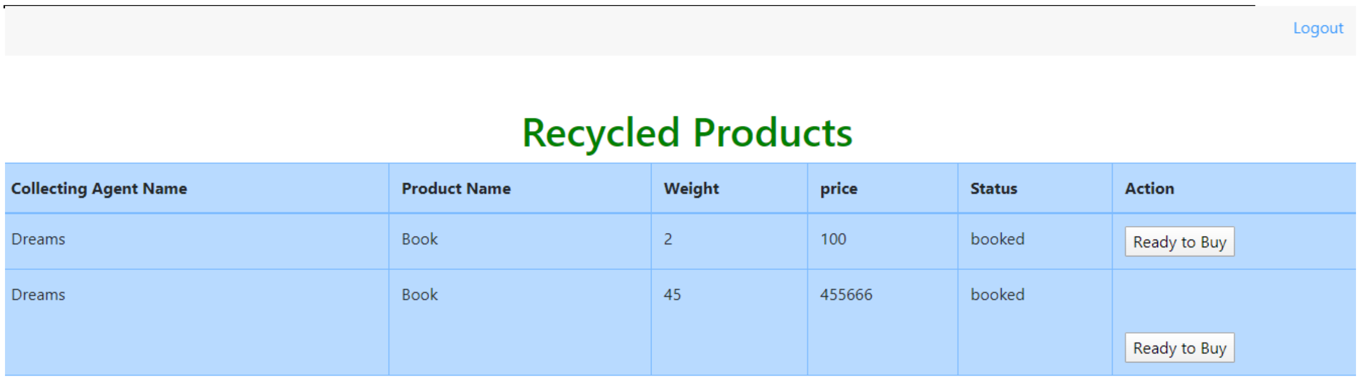


Fig.A.10 User Sell

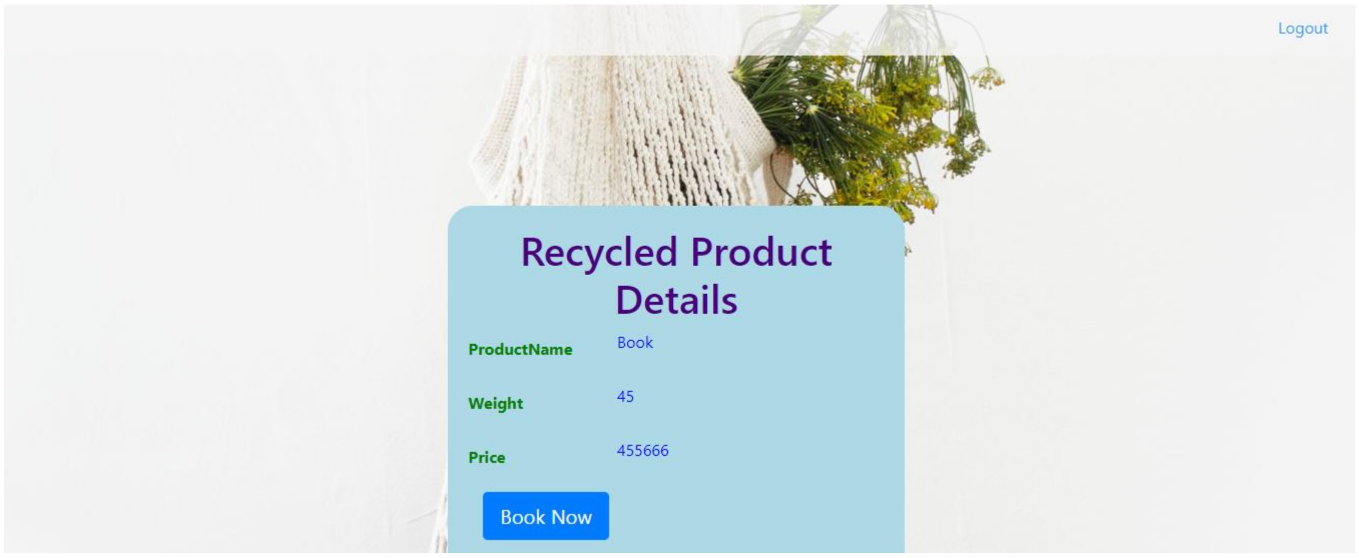


Logout

Recycled Products

Collecting Agent Name	Product Name	Weight	price	Status	Action
Dreams	Book	2	100	booked	Ready to Buy
Dreams	Book	45	455666	booked	Ready to Buy

Fig.A.11 Buy recycled Products

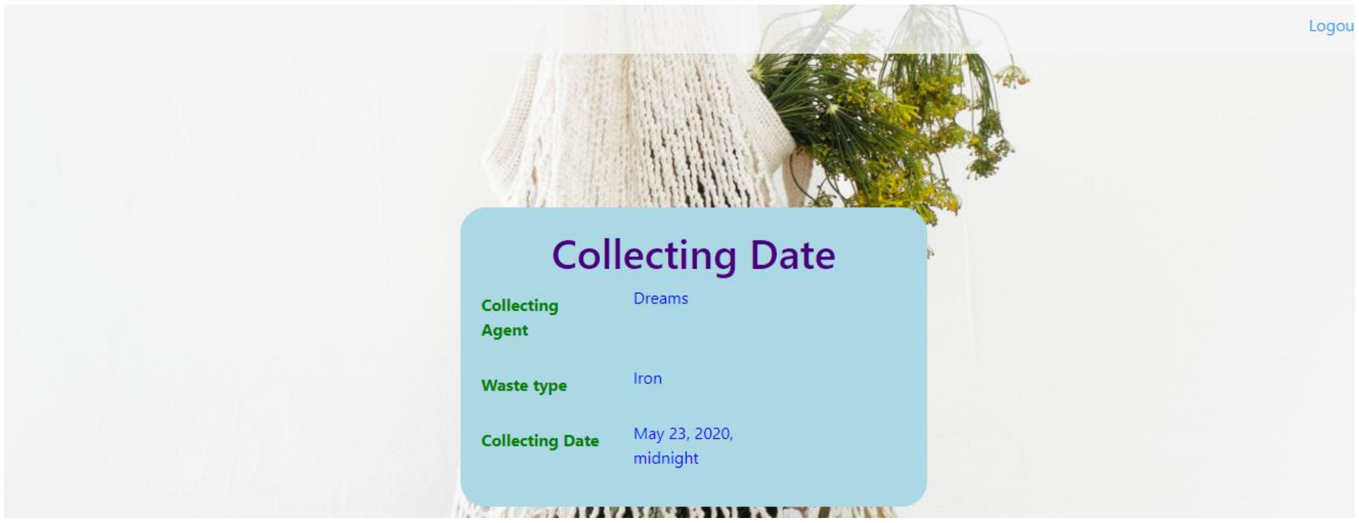


Logout

Recycled Product Details

ProductName	Book
Weight	45
Price	455666

Fig.A.12 Recycle Product Buy

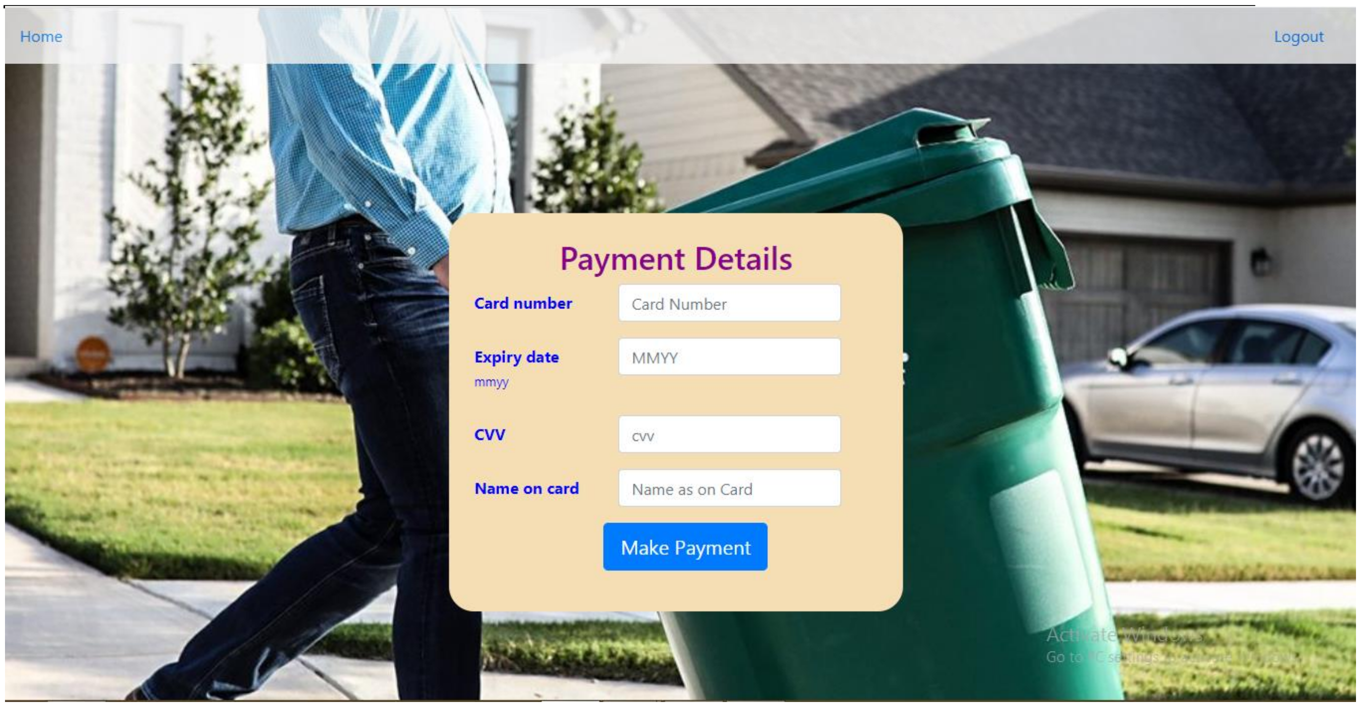


Logou

Collecting Date

Collecting Agent	Dreams
Waste type	Iron
Collecting Date	May 23, 2020, midnight

Fig A.13 View Collecting Waste Details


Home

Logout

Payment Details

Card number: Card Number

Expiry date mmyy: MMYY

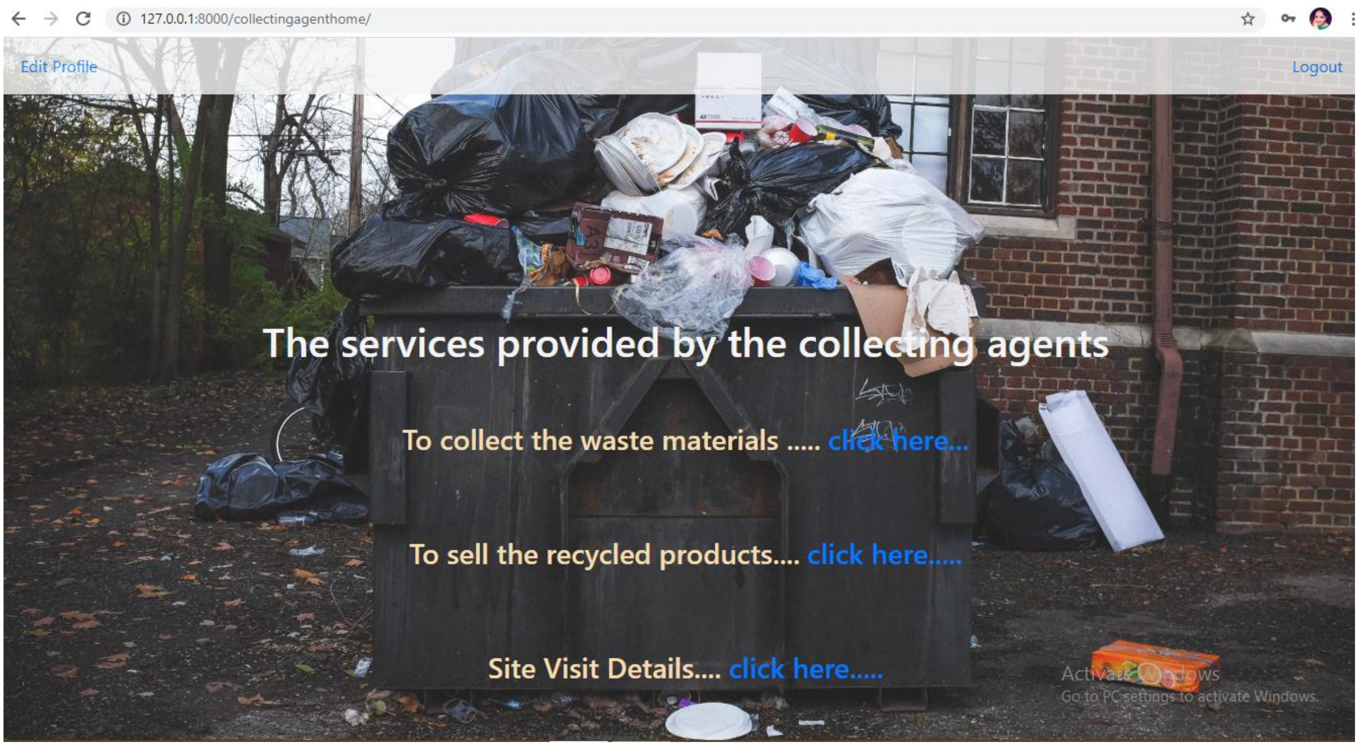
CVV: cvv

Name on card: Name as on Card

Make Payment

Fig.A14 Payment

Collecting System Functionalites :-


127.0.0.1:8000/collectingagenthome/

Edit Profile

Logout

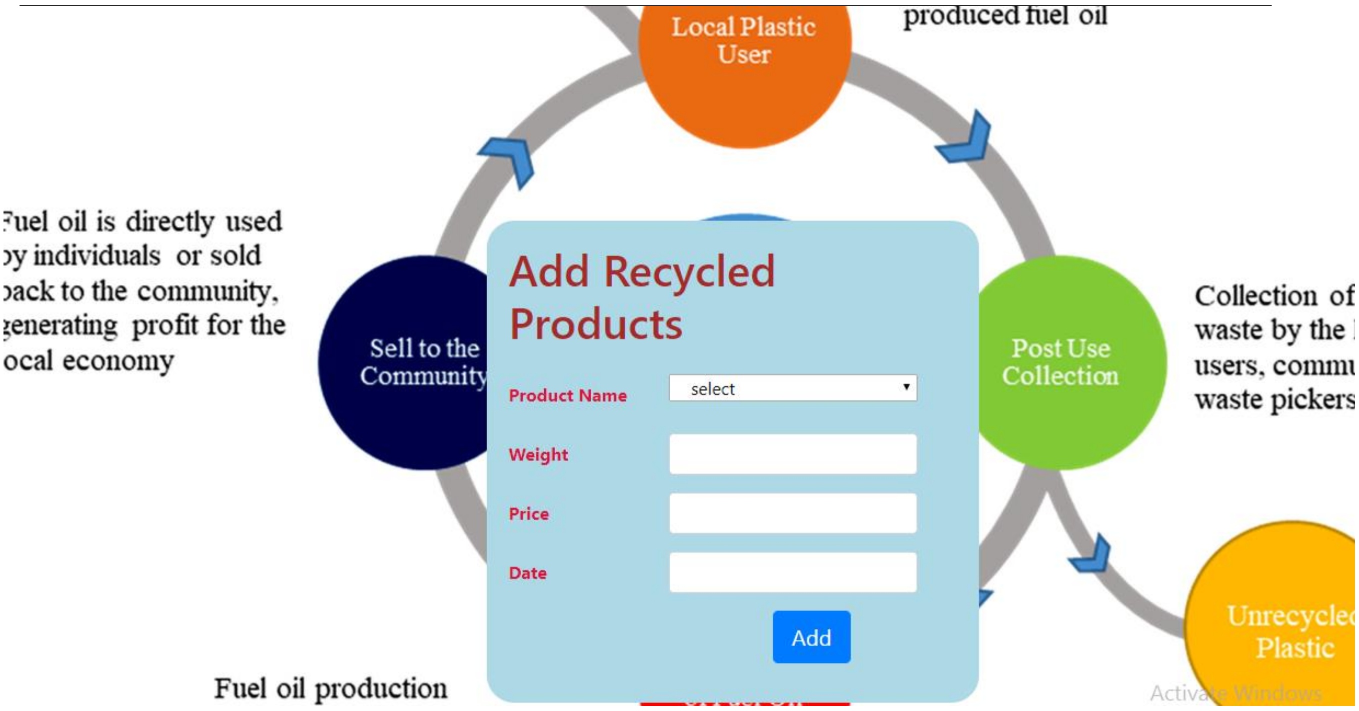
The services provided by the collecting agents

To collect the waste materials ..... click here...

To sell the recycled products.... click here.....

Site Visit Details.... click here.....

Fig.A.15 Collecting System Home

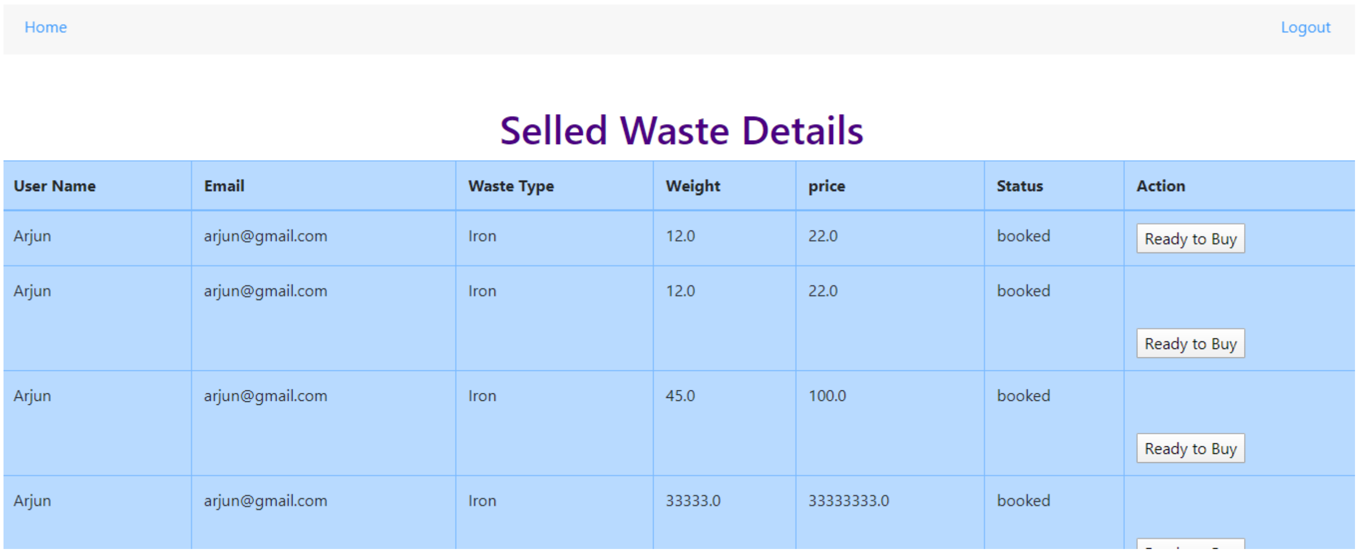

produced fuel oil
Local Plastic User
Fuel oil is directly used by individuals or sold back to the community, generating profit for the local economy
Sell to the Community
Post Use Collection
Collection of waste by the users, commu waste pickers
Unrecycled Plastic
Fuel oil production
Activate Windows

Add Recycled Products

Product Name	select
Weight	
Price	
Date	

Add

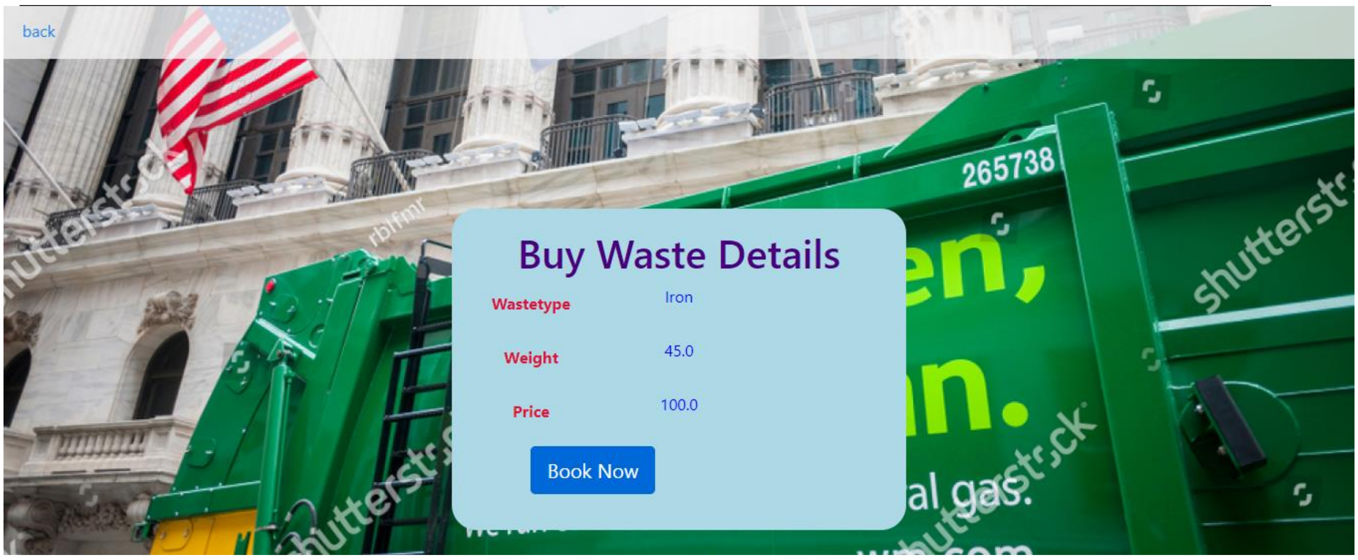
Fig.A.16 Add Recycled Products


Home Logout

Selled Waste Details

User Name	Email	Waste Type	Weight	price	Status	Action
Arjun	arjun@gmail.com	Iron	12.0	22.0	booked	Ready to Buy
Arjun	arjun@gmail.com	Iron	12.0	22.0	booked	Ready to Buy
Arjun	arjun@gmail.com	Iron	45.0	100.0	booked	Ready to Buy
Arjun	arjun@gmail.com	Iron	33333.0	33333333.0	booked	

Fig.A.17.View Selled Waste

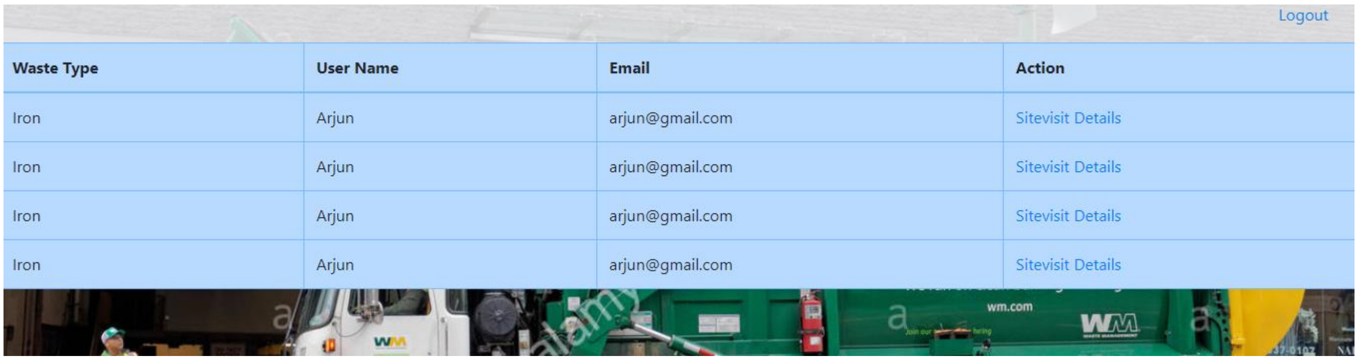


back

Buy Waste Details

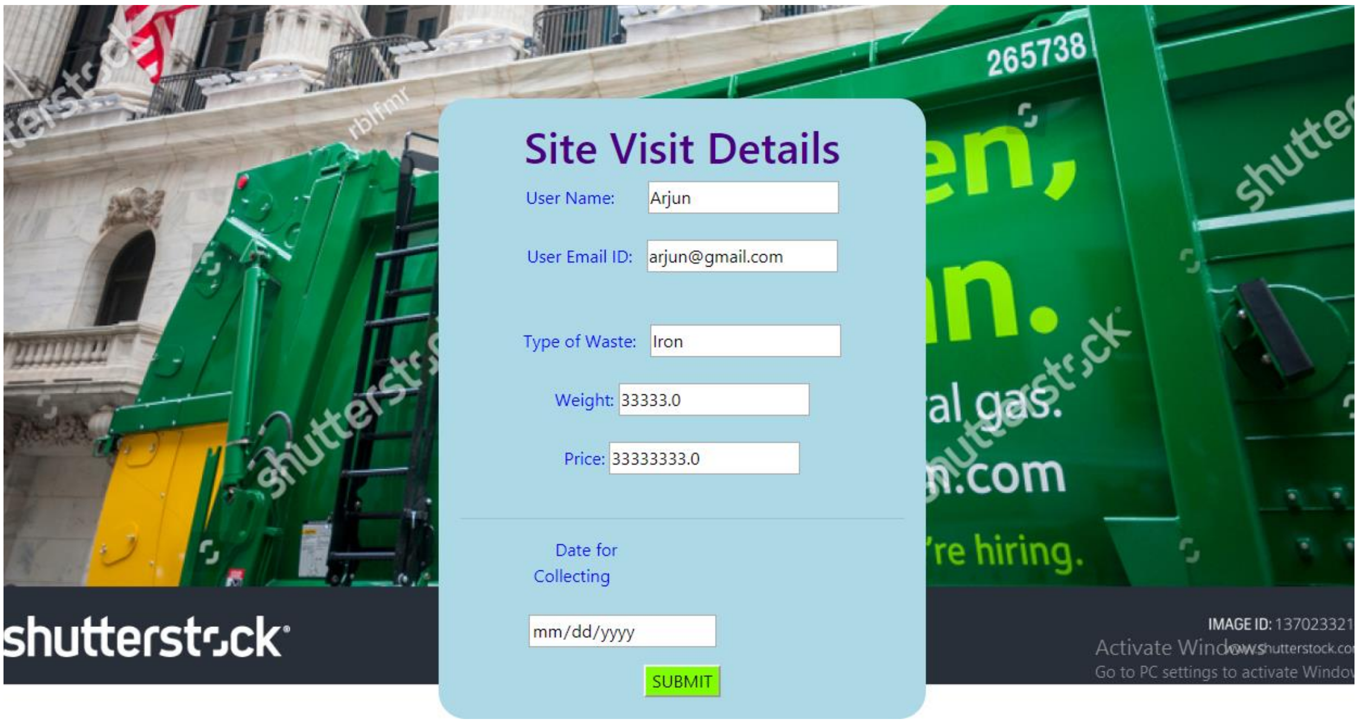
Wastetype	Iron
Weight	45.0
Price	100.0

Fig.A.18 Buy waste



Waste Type	User Name	Email	Action
Iron	Arjun	arjun@gmail.com	Sitevisit Details
Iron	Arjun	arjun@gmail.com	Sitevisit Details
Iron	Arjun	arjun@gmail.com	Sitevisit Details
Iron	Arjun	arjun@gmail.com	Sitevisit Details

Fig.A.19 Site Visitlist



Site Visit Details

User Name:	Arjun
User Email ID:	arjun@gmail.com
Type of Waste:	Iron
Weight:	33333.0
Price:	33333333.0
Date for Collecting	mm/dd/yyyy

Fig.A20.Site Visit Details

A.2 SAMPLE CODE

Views.py common app

```
from django.shortcuts import render, redirect
from django.contrib.auth.models import User, auth
from common.models import AddCategory, AddRecycle
from django.http import HttpResponse
from django.contrib import messages
from user.models import UserRegister, CollectRegister1
# Create your views here.
def category(request):
    return render(request, 'category.html')

def adminhome(request):
    return render(request, 'adminhome.html')

def addcategory(request):
    if request.method == 'POST':
        categoryname = request.POST['categoryname']

        addcat = AddCategory(categoryname=categoryname)
        addcat.save()
        return render(request, 'category.html')

def category1(request):
    return render(request, 'category1.html')

def addrecycle(request):
    if request.method == 'POST':
        categoryname = request.POST['categoryname']
        addre = AddRecycle(categoryname=categoryname)
        addre.save()
        return render(request, 'category1.html')


def adminhome1(request):
    results=request.GET["typeofusers"]
    return render(request, 'adminhome1.html',{'typeofusers':results})


def addadmin(request):
    return render(request, 'addadmin.html')
```

```
def viewuserlist(request):
    u1=UserRegister.objects.filter()
    return render(request, 'viewuserlist.html', {'u1':u1})

def viewcollect(request):
    v1=CollectRegister1.objects.filter()
    return render(request, 'viewcollect.html',{'v1':v1})

def approvcollect(request):
    v2=CollectRegister1.objects.filter()
    return render(request, 'approvcollect.html',{'v2':v2})
```

Views.py collect app

```
from django.shortcuts import render, redirect
from django.contrib.auth.models import User, auth
from collect.models import AddProduct, CollectDetails
from django.http import HttpResponse
from django.contrib import messages
from common.models import AddRecycle
from user.models import CollectRegister1, Sell
# Create your views here.


def addproducts(request):
    s=AddRecycle.objects.all()
    return render(request, 'addproducts.html',{'s':s})

def addsell(request):
    if request.method == 'POST':
        user = request.POST['name']
        productname = request.POST['productname']
        weight = request.POST['weight']
        price = request.POST['price']
        date = request.POST['date']
        add=AddProduct(user=user, productname=productname, weight=weight, price=price,
date=date, status="notbooked")
        add.save()
    return redirect('addproducts')

def userbuy(request):
    det=AddProduct.objects.filter()
    return render(request, 'userbuy.html', {'det':det})

def buyfun(request,id):
    bu=AddProduct.objects.get(id=id)
    bu.status="booked"
    bu.save()
    return render(request, 'byustatus.html' , {'bu':bu})

def byustatus(request):
    return render(request, 'byustatus.html')

def ordertable(request):
    return render(request,'ordertable.html')

def orderfun(request,id):
    if request.method=='POST':
       sta=request.POST['status']
       u=request.POST['email']
       v=request.POST['name']
       bu=AddProduct.objects.get(id=id)
       bu.status=sta
       bu.save()
       bu.user=u
       bu.save()
       bu.usname=v
       bu.save()
       return redirect('buyfun',id)
    else:
        return render(request,'byustatus.html')


def editprofile(request,cname):
    u=CollectRegister1.objects.filter(username=cname)
    return render(request, 'editprofile.html', {'u':u})

def collectprofile(request,cname):
    if request.method == 'POST':
        cstate = request.POST['state']
        cdistrict = request.POST['district']
        ccity = request.POST['city']
        cpin = request.POST['pin']
        cmobileno = request.POST['mobileno']
        u = CollectRegister1.objects.get(username=cname)
        u.state = cstate
        u.save()
        u.district = cdistrict
        u.save()
        u.city = ccity
        u.save()
        u.pin = cpin
        u.save()
        u.mobileno = cmobileno
        u.save()
        return redirect('editprofile',cname)
    else:
        return redirect('editprofile',cname)

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

def notifications(request):
    noti=Sell.objects.filter()
    return render(request, 'notifications.html', {'noti':noti})

def colfun(request,id,cname):
    cb=Sell.objects.get(id=id)
    cb.status="booked"
    cb.save()
    cb.cname=cname
    cb.save()
    return render(request, 'collect.html',{'cb':cb})

def colorder(request,id):
    if request.method == 'POST':
       sta=request.POST['status']
       u=request.POST['email']
       v=request.POST['name']
       cb=Sell.objects.get(id=id)
       cb.status=sta
       cb.save()
       cb.user=u
       cb.save()
       cb.usname=v
       cb.save()
       return redirect('colfun',id)
    else:
        return render(request,'collect.html')


def collectingagent(request,name):
    c1 = Sell.objects.filter(status="booked")
    return render(request, 'collectingagent.html',{'c1':c1})

def sitevisit(request,name,id):
    sv=Sell.objects.get(user=name,id=id)
    return render(request, 'sitevisit.html',{'sv':sv})

def collectvisit(request):
    if request.method == 'POST':
        username = request.POST['username']
        email = request.POST['email']
        cname = request.POST['cname']
        wastetype = request.POST['wastetype']
        weight = request.POST['weight']
        price = request.POST['price']
        date = request.POST['date22']
        coll=CollectDetails(username=username, email=email, cname=cname, wastetype=wast
etype, weight=weight, price=price, date=date)
        coll.save()
        return render(request, 'sitevisit.html')

def visitdate(request):
    vd=CollectDetails.objects.filter()
    return render(request,'visitdate.html',{'vd':vd})
```

Views.py user app

```
from django.shortcuts import render, redirect
from django.contrib.auth.models import User, auth
from user.models import UserRegister, CollectRegister1, Sell
from django.http import HttpResponse
from django.contrib import messages
from collect.models import AddProduct
from common.models import AddCategory
# Create your views here.
def home(request):
    return render(request, 'home.html')

def aboutus(request):
    return render(request, 'aboutus.html')

def collect(request):
    return render(request, 'collect.html')

def collectingagenthome(request):
    return render(request, 'collectingagenthome.html')

def collectservice(request):
    return render(request, 'collectservice.html')

def login(request):
    return render(request, 'login.html')
```

```
def ordertable(request):
    return render(request, 'ordertable.html')

def  productdetails(request):
    return render(request, 'productdetails.html')

def recycleservice(request):
    return render(request, 'recycleservice.html')

def recyclingagenthome(request):
    return render(request, 'recyclingagenthome.html')

def removeproducts(request):
    return render(request, 'removeproducts.html')

def sell(request):
    s=AddCategory.objects.all()
    return render(request, 'sell.html',{'s':s})

def sellservice(request):
    return render(request, 'sellservice.html')

def signup(request):
    return render(request, 'signup.html')

def userhome(request):
    return render(request, 'userhome.html')

def viewproducts(request):
    list1 = ViewProduct.objects.all()
    return render(request, 'viewproducts.html', {'list1':list1})

def wastedetails(request):
    return render(request, 'wastedetails.html')

def signup1(request):
    return render(request, 'signup1.html')



def signup(request):
    if request.method == 'POST':
        fname = request.POST['fname']
        lname = request.POST['lname']
        housenumber = request.POST['housenumber']
        state = request.POST['state']
        district = request.POST['district']
        city = request.POST['city']
        pin = request.POST['pin']
        mobileno = request.POST['mobileno']
        usertype = request.POST['usertype']
        email = request.POST['email']
        password = request.POST['password']
        uname = fname+ "" +lname

        if User.objects.filter(username=email).exists():
            print('user taken')
            return redirect('login')
        else:
            user = User.objects.create_user(username=email, password=password, email=em
ail, first_name=fname, last_name=lname)
            user.save()
            u = UserRegister(username=email, name=uname, housenumber=housenumber, state
=state, district=district, city=city, pin=pin, mobile=mobileno, usertype=usertype)
            u.save()
            print('user created')
            return redirect('login')
    else:
        return render(request, 'signup.html')


def signup1(request):
    if request.method == 'POST':

        cname = request.POST['cname']
        state = request.POST['state']
        district = request.POST['district']
        city = request.POST['city']
        pin = request.POST['pin']
        mobileno = request.POST['mobileno']
        email = request.POST['email']
        password = request.POST['password']
        if User.objects.filter(username=email).exists():
            print('username taken')
            return redirect('login')
        else:
            user = User.objects.create_user(username=email, password=password, email=em
ail, first_name=cname, last_name="Agent")
            user.save()
            u1 = CollectRegister1(username=email, cname=cname, state=state, district=di
strict, city=city, pin=pin, mobile=mobileno)
            u1.save()
            print('collecting system created')
            return redirect('login')
    else:
        return render(request, 'signup1.html')


def login(request):
    if request.method == 'POST':
        username = request.POST['email']
        password = request.POST['password']
        if username=='admin' and  password=='admin':
            return redirect('adminhome')
        else:
            user = auth.authenticate(username=username, password=password)
            if user is not None:
                auth.login(request, user)
                if User.objects.filter(username=username, last_name="Agent").exists():
                    return  redirect('collectingagenthome')
                else:
                    return redirect('userhome')
            else:
                messages.info(request, 'invalid credintail')
            return render(request,'login.html')
    else:
        return render(request, 'login.html')


def logout(request):
    auth.logout(request)
    return redirect('/')

def userprofile(request,uname):
    m = UserRegister.objects.filter(username=uname)
    return render(request, 'userprofile.html',{'m':m})

def saveprofile(request,uname):
    if request.method == 'POST':

        nhousenumber = request.POST['housenumber']
        nstate = request.POST['state']
        ndistrict = request.POST['district']
        ncity = request.POST['city']
        npin = request.POST['pin']
        nmobileno = request.POST['mobileno']

        m = UserRegister.objects.get(username=uname)

        m.housenumber = nhousenumber
        m.save()
        m.state = nstate
        m.save()
        m.district = ndistrict
        m.save()
        m.city = ncity
        m.save()
        m.pin = npin
        m.save()
        m.mobile = nmobileno
        m.save()

        return redirect('userprofile',uname)

    else:
        return redirect('userprofile',uname)

def viewcollectingsystem(request):
    m1 = CollectRegister1.objects.filter()
    return render(request, 'viewcollectingsystem.html',{'m1':m1})


def selltype(request):
    if request.method == 'POST':
        name = request.POST['name']
        email = request.POST['email']
        cname = request.POST['cname']
        wastetype =request.POST['wastetype']
        weight = request.POST['weight']

        price = request.POST['price']
        s=Sell(user=name, email=email, cname=cname, wastetype=wastetype, weight=weight,
 price=price, status="notbooked")
        s.save()
        return redirect('sell')

def usertype(request):


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