EcoRecover

Mini Project Report

Submitted by

Joyel Joy

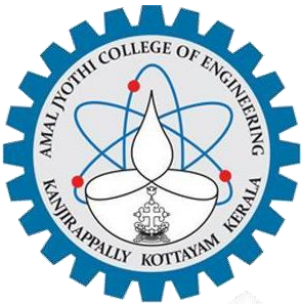
Reg. No.: AJC19MCA-I033

In Partial fulfillment for the Award of the Degree of

INTEGRATED MASTER OF COMPUTER APPLICATIONS

(INMCA)

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]

2023-2024

DEPARTMENT OF COMPUTER APPLICATIONS

AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

CERTIFICATE

This is to certify that the Project report, "**EcoRecover"** is the bona fide work of **Joyel Joy (Regno: AJC19MCA-I033)** in partial fulfillment of the requirements for the award of the Degree of Integrated Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2023-24.

Mr. Jinson Devis

Internal Guide

Ms. Meera Rose Mathew

Coordinator

Rev. Fr. Dr. Rubin Thottupurathu Jose

Head of the Department

DECLARATION

I hereby declare that the project report **"EcoRecover"** is a bona fide work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2023-2024.

Date: **Joyel Joy**

KANJIRAPPALLY **Reg: AJC19MCA-I033**

ACKNOWLEDGEMENT

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our Manager **Rev. Fr. Dr. Mathew Paikatt** and Principal **Dr. Lillykutty Jacob** for providing good faculty for guidance.

I owe a great depth of gratitude towards our Head of the Department **Rev.Fr.Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinator **Meera Rose Mathew** for his valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my guide **Jinson Devis** for her inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

Joyel Joy

ABSTRACT

EcoRecover, an innovative online platform, is dedicated to addressing waste management challenges and promoting sustainability through the sale of recycled products. This multifaceted system seeks to establish a circular economy by efficiently managing waste disposal and encouraging the purchase of eco-friendly products.

The platform comprises several integral modules. User Authentication ensures that only authorized personnel have access to sensitive data and functionalities. The Waste Collection Module empowers Waste Generators to book waste bins for collection and enables Waste Collectors to efficiently manage waste collection orders, offering features for bin booking, order creation, rescheduling, and collection date scheduling. An Online Payment and Order Management system securely handles customer purchases from the Recycled Store, complete with order tracking and management features. The Database Design and Management module organizes user information, product data, waste collection orders, and other pertinent data.

One of the platform's most exciting applications is Waste Type Recognition and Disposal Method Using Image Classification. This employs computer vision and machine learning to identify different types of waste from images and subsequently suggests appropriate disposal methods based on the waste category. Furthermore, users can book events such as Recycling Drives, Workshops, Seminars, Waste Collection, and Disposal Events, Green Fairs, and Expos.

For Waste Generators, the platform offers functionalities including registration, login, user profile management, bin booking, rescheduling, payment processing, feedback provision, order tracking, and campaign booking. Administrators can access the system to manage user accounts and details, oversee bin orders, and authorize waste disposal processes. Event Managers can register, login, add, update, or delete event details, ensuring the seamless organization of eco-friendly events. EcoRecover serves as an integrated solution that not only efficiently manages waste but also actively promotes recycling, sustainability, and the purchase of eco-friendly products, making it a comprehensive waste management and recycling platform.

CONTENT

List of Abbreviation

IDE - Integrated Development Environment
HTML - Hyper Text Markup Language.
CSS - Cascading Style Sheet
SQL - Structured Query Language
UML - Unified Modelling Language
JS - JavaScript
AJAX - Asynchronous JavaScript and XML Environment

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

EcoRecover is a groundbreaking online platform that tackles waste management challenges while promoting sustainability through the sale of recycled products. The primary objective of this innovative system is to create a circular economy by efficiently managing waste disposal and encouraging the adoption of eco-friendly products. One of the foundational modules of EcoRecover is user authentication, ensuring that only authorized personnel have access to sensitive data and functionalities. This secure access sets the stage for the platform's other core functions. The Waste Collection Module, for instance, empowers Waste Collectors to efficiently manage waste collection orders by offering features like waste bin booking, order creation, rescheduling, and precise collection date and time scheduling.

EcoRecover also incorporates an Online Payment and Order Management system, allowing customers to make secure purchases from the Recycled Store and providing tools for order tracking and management. Meanwhile, the Database Design and Management module ensures the structured storage of crucial data, including user information, product details, waste collection orders, and other relevant information. A standout feature of EcoRecover is its use of computer vision and machine learning for waste type recognition and disposal method recommendations based on image classification. Users can easily identify different waste types from images and receive guidance on the most appropriate disposal methods, contributing to eco-conscious decision-making. Furthermore, EcoRecover supports the booking of various eco-friendly events such as recycling drives, workshops, seminars, green fairs, and expos. This module encourages active participation in sustainability initiatives, promoting a greener planet.

EcoRecover caters to the needs of Event Managers as well, simplifying event management through streamlined registration and login processes, the ability to add and update event details, and the option to remove outdated events. In essence, EcoRecover is a comprehensive platform that unites waste management and sustainability, offering a user-friendly solution that not only addresses waste management challenges but actively promotes recycling, sustainability, and eco-friendly choices. It stands as a pioneering force in the realm of waste management and sustainability, striving to make the world a more eco-conscious and sustainable place for all.

1.2 PROJECT SPECIFICATION

1. Customer (Waste Generators):

For Waste Generators, EcoRecover offers a comprehensive set of user-friendly features. Users can begin their journey by creating accounts on the platform, allowing them to access a range of services. Registered users can seamlessly log in to their accounts and manage their profiles, ensuring a personalized experience. They have the convenience of booking waste bins for efficient collection and even the flexibility to reschedule if needed. The platform streamlines payments for services and products, making transactions hassle-free. Furthermore, users are encouraged to provide feedback on their experiences, ensuring continuous improvement. They can easily track the progress of their waste collection orders and actively participate in recycling campaigns and events through convenient booking options.

2. Admin:

Administrators play a pivotal role in EcoRecover, benefiting from a range of essential tools and responsibilities. They are granted access through a secure login system to manage their accounts effectively. One of their core functions includes the meticulous management of user accounts and associated details, ensuring a smooth and organized user experience. Administrators also oversee the crucial process of waste bin order management, ensuring that waste collection is efficient and well-coordinated. Additionally, they hold the responsibility of authorizing waste disposal processes, an essential task to maintain the integrity of the waste management system. These key functionalities empower administrators to maintain the platform's efficiency and sustainability, making EcoRecover a reliable and responsive solution in waste management and recycling.

3. Event Managers:

Event managers are equipped with a user-friendly suite of features within ECORECOVER to efficiently handle and manage recycling events. To commence their involvement, event managers can easily register and access their accounts through a straightforward login process. Once logged in, they have the authority to add comprehensive event details, providing information about recycling events that ensures transparency and accessibility to users. In the ever-evolving landscape of event management, event managers can effortlessly update event information, ensuring that the most current and accurate details are readily available to participants. Should the need arise, event managers also have the capability to delete events, providing flexibility and control in adapting to changing circumstances and maintaining a well-organized and responsive platform for recycling drives, workshops, and seminars.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

System analysis is a critical and foundational phase in the development of any system. It serves as the initial step, involving the meticulous gathering and examination of data to identify issues and provide effective solutions. Effective communication between system users and developers is paramount during this phase, ensuring a comprehensive understanding of the current system's performance. The system analyst takes on the role of an investigator, closely scrutinizing the system's inputs, outputs, processes, and their impact on organizational outcomes.

Data is collected through various means, such as surveys and interviews, with the overarching goals of comprehending system operations, pinpointing areas of concern, and proposing solutions to address organizational challenges. The responsibility of problem-solving is then transferred to the system designer, who rigorously evaluates the proposed solutions against the existing system. Upon selecting the best solution, users are provided with the opportunity to either accept or reject the recommendation, and this iterative process continues until user satisfaction is achieved based on their valuable feedback.

2.2 EXISTING SYSTEM

Current existing waste management systems have long been in need of innovation and transformation, and EcoRecover steps in as a beacon of change. Traditionally, waste management has been a challenging and often inefficient process, but with the emergence of innovative platforms like EcoRecover, there is hope for a sustainable future. In these conventional systems, the lack of user authentication has led to issues in data security, while the management of waste collection has been cumbersome and time-consuming. The absence of secure online payment systems often hinders user convenience in purchasing eco-friendly products. Moreover, the traditional approaches often fall short in efficiently categorizing and disposing of waste. EcoRecover 's advanced modules, including user authentication, waste collection management, online payment, and image-based waste recognition, are revolutionizing the way waste is handled.

2.2.1 NATURAL SYSTEM STUDIED

The study of natural systems encompasses a wide range of scientific disciplines and fields, all aimed at understanding the intricate web of relationships, processes, and phenomena that govern our natural world. It delves into the complex interactions between living organisms, their environments, and the physical forces that shape our planet. From ecology, where researchers investigate the dynamics of ecosystems and the interplay between species, to meteorology and climatology, which

explore the Earth's atmospheric systems and climate patterns, the study of natural systems is a diverse and interconnected endeavor. It also encompasses fields like geology, oceanography, and biology, where scientists seek to comprehend the Earth's geological history, the secrets of marine ecosystems, and the mysteries of life itself. In essence, the study of natural systems is a pursuit of knowledge that helps us better appreciate the marvels of the natural world and aids in making informed decisions for its preservation and sustainable coexistence with human society.

2.2.2 DESIGNED SYSTEM STUDIED

The designed system for ECORECOVER stands as a testament to innovation in waste management and sustainability. In this meticulously crafted platform, the challenges of traditional waste management systems are effectively addressed. Through thorough system analysis, the design team has introduced features like user authentication to enhance data security and streamline user access, ensuring that only authorized personnel have control over sensitive information. The Waste Collection Module offers a practical solution, allowing Waste Generators to conveniently book waste bins for collection while providing Waste Collectors with the tools to manage these orders efficiently. Online Payment and Order Management bring the convenience of secure online transactions, making eco-friendly product purchases accessible to users. The integration of database design and management ensures the robust storage of essential data, from user information to waste collection records. The system also delves into the realm of computer vision and machine learning, introducing waste type recognition and disposal methods based on image classification, a remarkable feature that has the potential to revolutionize waste categorization and management. Additionally, the platform fosters sustainability by allowing users to actively participate in recycling drives, workshops, seminars, and other eco-friendly events. The designed system for ECORECOVER offers an integrated and comprehensive solution, marking a significant leap toward more responsible and efficient waste management practices.

2.3 DRAWBACKS OF EXISTING SYSTEM

- Lack of User Authentication: Many existing systems lack robust user authentication, which poses a significant security risk by allowing unauthorized access to sensitive data and functionalities.
- Inefficient Waste Collection: The traditional waste management systems often suffer from inefficiencies in waste collection, leading to delays, missed collections, and disorganized scheduling.

- Limited Payment Options: These systems often do not provide secure online payment options, making it inconvenient for users to purchase eco-friendly products or services.
- Data Management Challenges: Data management can be challenging, resulting in a lack of organization and accessibility for essential user information and waste collection records.
- Inadequate Waste Categorization: Traditional waste management systems may not efficiently categorize waste, which can lead to improper disposal and negative environmental impacts.
- Manual Recognition of Waste Types: Without the use of advanced technologies like image classification, identifying waste types is a labor-intensive and error-prone process, hampering efficient waste management.
- Limited User Engagement: These systems often lack the tools and features to actively engage users in sustainability efforts, such as recycling drives and eco-friendly events, which can limit the promotion of responsible waste management practices.

2.4 PROPOSED SYSTEM

The proposed system for EcoRecover presents a comprehensive and transformative solution to address the drawbacks of existing waste management systems. It is designed to revolutionize waste management and promote sustainability. One of the central features of the proposed system is robust user authentication, ensuring the security and confidentiality of sensitive data. This addresses the first drawback by preventing unauthorized access. The Waste Collection Module streamlines the process of waste collection, making it efficient, organized, and user-friendly, overcoming the inefficiencies of the existing system. Secure online payment and order management features address the limited payment options in traditional systems, providing a convenient and hassle-free platform for customers to purchase recycled products. The proposed system's database design and management enhance data organization and accessibility, mitigating data management challenges. Furthermore, the introduction of waste type recognition through image classification offers a groundbreaking solution for efficient waste categorization and disposal. Finally, the system actively engages users in sustainability efforts by allowing them to participate in recycling drives, workshops, and eco-friendly events, addressing the issue of limited user engagement. The proposed system for EcoRecover signifies a significant leap towards a more sustainable and efficient waste management system, embodying innovation and user-centric design.

2.5 ADVANTAGES OF PROPOSED SYSTEM

The proposed system for EcoRecover brings a host of advantages over the existing waste management systems, enhancing efficiency, sustainability, and user experience. Some of the key advantages include:

- Enhanced Data Security: Robust user authentication ensures that only authorized personnel have access to sensitive data, significantly improving data security and confidentiality.
- Efficient Waste Collection: The Waste Collection Module streamlines waste collection processes, making it more organized and user-friendly, thus reducing delays and missed collections.
- Convenient Online Payments: Secure online payment and order management make it easy for users to purchase recycled products and services, improving convenience and user satisfaction.
- Improved Data Management: Database design and management enhance the organization and accessibility of user information and waste collection records, leading to more efficient data management.
- Advanced Waste Categorization: The system's waste type recognition and disposal method using image classification is a groundbreaking feature that ensures more accurate and efficient waste categorization and disposal methods.
- User Engagement: The platform actively engages users in sustainability efforts, enabling them to participate in recycling drives, workshops, seminars, and eco-friendly events, fostering a sense of community and responsibility.
- Promotion of Sustainability: EcoRecover promotes sustainability by facilitating the purchase of eco-friendly products, thereby contributing to a circular economy and reducing environmental impact.
- Transparency: The system provides transparency by allowing users to track their waste collection orders and actively engage in recycling campaigns and events.

CHAPTER 3
REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

A feasibility study is a crucial step in assessing the viability and potential success of a project before proceeding with its development. In the case of the Waste Management and Recycled Product Selling Site, conducting a feasibility study will help evaluate various aspects of the project to determine if it is financially, technically, and operationally feasible.

3.1.1 Economical Feasibility

Economic feasibility refers to determining whether the Waste Management and Recycled Product Selling Site is financially viable and can generate sufficient returns on investment. In simpler terms, it involves assessing if the project makes good financial sense. This analysis considers the initial costs of developing the platform, ongoing expenses for maintenance and operations, and potential revenue streams. Additionally, it evaluates the market demand for such a platform and the potential to attract users and sustainable businesses. By comparing projected revenues to expenses, we can gauge if the project is economically feasible and can lead to long-term profitability. A positive economic feasibility study suggests that the site can create value, contribute to waste reduction efforts, and be financially sustainable over time. On the other hand, if the study indicates significant financial challenges or uncertainties, stakeholders may need to reevaluate their approach or consider adjustments to the project scope to ensure economic viability.

3.1.2 Technical Feasibility

In the realm of our Waste Management and Recycled Product Selling Site, technical feasibility serves as a critical evaluation process to determine the project's achievability from a technical perspective. This thorough assessment encompasses a range of crucial factors, including the availability of image recognition algorithms, hosting capabilities, and the essential programming languages and frameworks required for successful platform development and maintenance. Furthermore, technical feasibility considers the platform's capacity to efficiently manage user traffic, data storage, and meet stringent security requirements. The overarching objective is to ensure that the project can be technically realized within the existing confines of technology and infrastructure. What's particularly noteworthy is that stakeholders need not possess specialized expertise in the technologies used, as our platform is designed with user-friendliness in mind. Moreover, the required technology and infrastructure are readily available and easily accessible, ensuring a seamless path toward project implementation. In essence, our Waste Management and Recycled Product Selling Site stands as a testament to the

attainability of our goals, making it an exciting and accessible endeavor for all involved.

3.1.3 Behavioral Feasibility

Behavioral feasibility assesses whether the proposed Waste Management and Recycled Product Selling Site is acceptable and likely to be adopted by its target users. It focuses on understanding the users' attitudes, behaviors, and willingness to use the platform. In simpler terms, it evaluates whether people would actually use and engage with the site. By conducting surveys and user interviews, the project team can gather valuable insights about users' preferences and needs. Behavioral feasibility considers factors like user habits, their level of comfort with technology, and their motivation to participate in waste management efforts. It also examines how the platform aligns with users' existing waste disposal practices and if it encourages positive behavior change. To ensure behavioral feasibility, the platform needs to be user-friendly, easy to navigate, and provide clear benefits to users. It should offer incentives, rewards, and engaging features that motivate users to actively participate in waste reduction campaigns and purchase recycled products. By addressing user concerns and preferences, the platform can foster a positive user experience and gain user acceptance. Furthermore, behavioral feasibility analyses potential resistance to change and addresses any barriers that may hinder user adoption. It emphasizes the importance of promoting awareness and educating users about the benefits of proper waste management and recycling. By designing the platform with user behavior in mind, it enhances the likelihood of successful user engagement and sustainable usage over time.

3.1.4 Feasibility Study Questionnaire

Question 1: What are the current Waste Management Practices?

In Eco Friendly Solutions, waste collection primarily relies on traditional bin-based systems, but the region faces significant challenges. One pressing issue is the prevalence of illegal dumping, particularly in certain areas, which not only blights the environment but also strains waste management resources. Additionally, there is a critical need for improving recycling awareness among residents to promote sustainable waste disposal practices and reduce the overall environmental impact..

Question 2: What are the expected outcomes of your project?

Anticipated outcomes include a 30% reduction in landfill waste, a 20% increase in recycling rates, and a heightened sense of environmental responsibility among the community.

Question 3: What are the key challenges faced in waste management?

Eco Friendly Solutions grapples with several key challenges, including improper waste disposal

practices that contribute to environmental degradation. Furthermore, limited recycling awareness exacerbates the issue, while illegal dumping not only harms the environment but also poses logistical and financial burdens on waste management efforts. Additionally, managing hazardous waste presents a critical concern due to potential health and environmental risks.

Question 4: How do you engage the public and businesses to participate in waste management programs?

Engaging the public through educational workshops, reward-based initiatives, and partnership with businesses fosters active participation in waste management efforts.

Question 5 :How do you address the challenges of hazardous waste disposal?

Proper hazardous waste classification, specialized containers, and collaborating with licensed disposal facilities ensure safe hazardous waste management.

Question 6: How do you measure the success of a waste management program?

Success can be measured through key performance indicators such as waste diversion rates, recycling percentages, and community satisfaction surveys.

Question 7: What steps can be taken to minimize food waste in residential and commercial settings?

Educating individuals about food waste prevention, implementing food donation programs, and encouraging composting can help reduce food waste.

Question 8: What are the key challenges faced in waste management?

Eco Friendly Solutions faces a multifaceted set of challenges in its waste management efforts. These challenges encompass improper waste disposal, which can harm the environment, limited recycling awareness, which hinders sustainable practices, illegal dumping, leading to both environmental damage and operational complications, and the management of hazardous waste, which demands strict protocols to mitigate potential health and environmental risks

Question 9: How do you ensure the efficient collection and disposal of e-waste?

E-waste is efficiently managed through designated e-waste collection centers, periodic electronic waste disposal drives, and partnerships with certified e-waste recycling facilities to safely and responsibly handle electronic waste.

Question 10: What strategies are in place to promote sustainable waste management behavior among the community?

Sustainable waste management behavior is promoted through community outreach programs, awareness campaigns, and the implementation of easy-to-use, segregated waste disposal bins, alongside incentive programs to reward eco-friendly practices.

3.1 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor - Ryzen 7
RAM - 8 G B
Hard disk - 5 1 2 G B

3.2.2 Software Specification

Front End -HTML,CSS
Back End- Django
Database - SQLite
Client on PC -Windows 7 and above.
Technologies used - JS, HTML5, AJAX, J Query, CSS

3.3 SOFTWARE DESCRIPTION

3.3.1 SQLlite

SQLite is a lightweight, embedded, open-source relational database management system (RDBMS) that is widely used for local data storage and management in a variety of applications. It stands out for its simplicity, portability, and efficiency, making it an excellent choice for mobile applications, desktop software, and small to medium-sized projects. SQLite stores data in a single, self-contained file, eliminating the need for a separate server or complex database administration. It supports SQL (Structured Query Language), providing a familiar query language for data manipulation and retrieval. Despite its minimal footprint, SQLite is capable of handling complex tasks, offering features like ACID compliance, data typing, indexing, and support for transactions. Its self-contained nature, cross-platform compatibility, and ease of integration have made it a popular choice for developers seeking a local data storage solution in their applications.

3.3.2 Django

Django is a high-level, open-source web framework written in Python that facilitates the development of robust and maintainable web applications. It follows the Model-View-Controller (MVC) architectural pattern, but in Django, it's referred to as Model-View-Template (MVT). Django offers a wealth of features and tools, making it a popular choice for web developers. Some key components and features of Django include an Object-Relational Mapping (ORM) system for database interactions, a powerful templating engine, a user authentication system, and automatic admin interfaces for managing data. It also provides built-in support for URL routing, session management, security features, and internationalization.

3.3.3 HTML

HTML is the standard markup language used to create the structure and content of web pages. It consists of a set of elements or tags that define various components on a webpage, such as headings, paragraphs, images, links, forms, and more. These elements are used to organize and structure the content, making it understandable for both browsers and users. HTML provides the foundation for creating the skeleton of a webpage, and it's essential for creating the basic structure and layout.

3.3.4 CSS

CSS is a stylesheet language used for defining the visual presentation and layout of web pages written in HTML. It allows web developers to control how HTML elements are displayed, including aspects like fonts, colors, spacing, and positioning. CSS separates the content (defined by HTML) from its presentation, enabling consistent and attractive designs across a website. It uses selectors to target HTML elements and declarations to specify the desired styling properties.

In practice, HTML and CSS work together: HTML defines the structure and content of a webpage, while CSS is responsible for the presentation, including layout and design. Together, they form the building blocks of web development and are essential for creating user-friendly and visually appealing websites.

CHAPTER 4
SYSTEM DESIGN

4.1 INTRODUCTION

Any designed system or product's development starts with the design phase. An efficient system depends on well-executed design, which is a creative process. It entails utilizing a variety of approaches and concepts to define a process or system in enough depth to allow for its actual execution. Regardless of the development model chosen, the design phase is critical in software engineering. It strives to produce the architectural detail needed to build a system or product and serves as the technical backbone of the software engineering process.

This program has through a thorough design phase that optimizes every aspect of effectiveness, performance, and accuracy. A user-oriented document is converted into a document for programmers or database employees during the design process.

4.2 UML DIAGRAM

A standardized dialect called Unified Modelling Language (UML) is utilized to conceptualize, characterize, plan, and depict program frameworks. The Question Administration Gather (OMG) was dependable for creating UML, and the primary draft of the UML 1.0 definition was discharged in January 1997. Programming dialects like Java, C++, and COBOL are not the same as UML. It could be a nonexclusive visual demonstrating dialect utilized for computer program frameworks and a pictorial dialect utilized for program outlines. UML may be utilized for non-software frameworks, such as fabricating forms, indeed though it is generally utilized to speak to program frameworks.

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

4.2.1 USE CASE DIAGRAM

A use case diagram may be a graphical delineation that appears how clients and other outside on-screen characters associated with a system's inside components. A utilize case diagram's essential work is to perceive, layout, and orchestrate a system's utilitarian needs as seen through the eyes of its clients. The Unified Modelling Language (UML), a standard language for modelling actual things and systems, is frequently used to construct use case diagrams.

Actors:

Waste Generators (Customer) play a pivotal role in the platform. They can seamlessly register on EcoRecover, creating their accounts to access an array of system features. Once registered, users can log in to their accounts, granting them access to various functionalities within the system. Waste Generators have the freedom to manage their profiles, ensuring their personal information is up to date. They can take advantage of the system's capabilities to book waste bins for collection, and if necessary, reschedule collection orders. EcoRecover offers a secure online payment system that allows users to make payments for services and products, including eco-friendly items. Feedback from Waste Generators is encouraged, allowing them to provide valuable insights to enhance the system and its services. Additionally, users can track the status of their waste collection orders and actively engage in sustainability efforts by booking participation in recycling campaigns and events.

Admins hold a significant role as system administrators. They have specific privileges, requiring them to log in to their accounts to manage various aspects of the system. Admins are responsible for the management of user accounts, which includes creating new accounts and making updates as needed. They also oversee and manage the waste bin order process, ensuring efficient and organized waste collection. Additionally, Admins have the authority to authorize waste disposal processes, a crucial responsibility for maintaining system integrity.

Event Managers are instrumental in managing recycling events. They have the capacity to register on EcoRecover, gaining access to their accounts. Once logged in, Event Managers can enrich the system by adding comprehensive details about recycling drives, workshops, seminars, and other events. They can also update event information as necessary to keep listings current and relevant. In cases where events become outdated or irrelevant, Event Managers can delete them from the system.

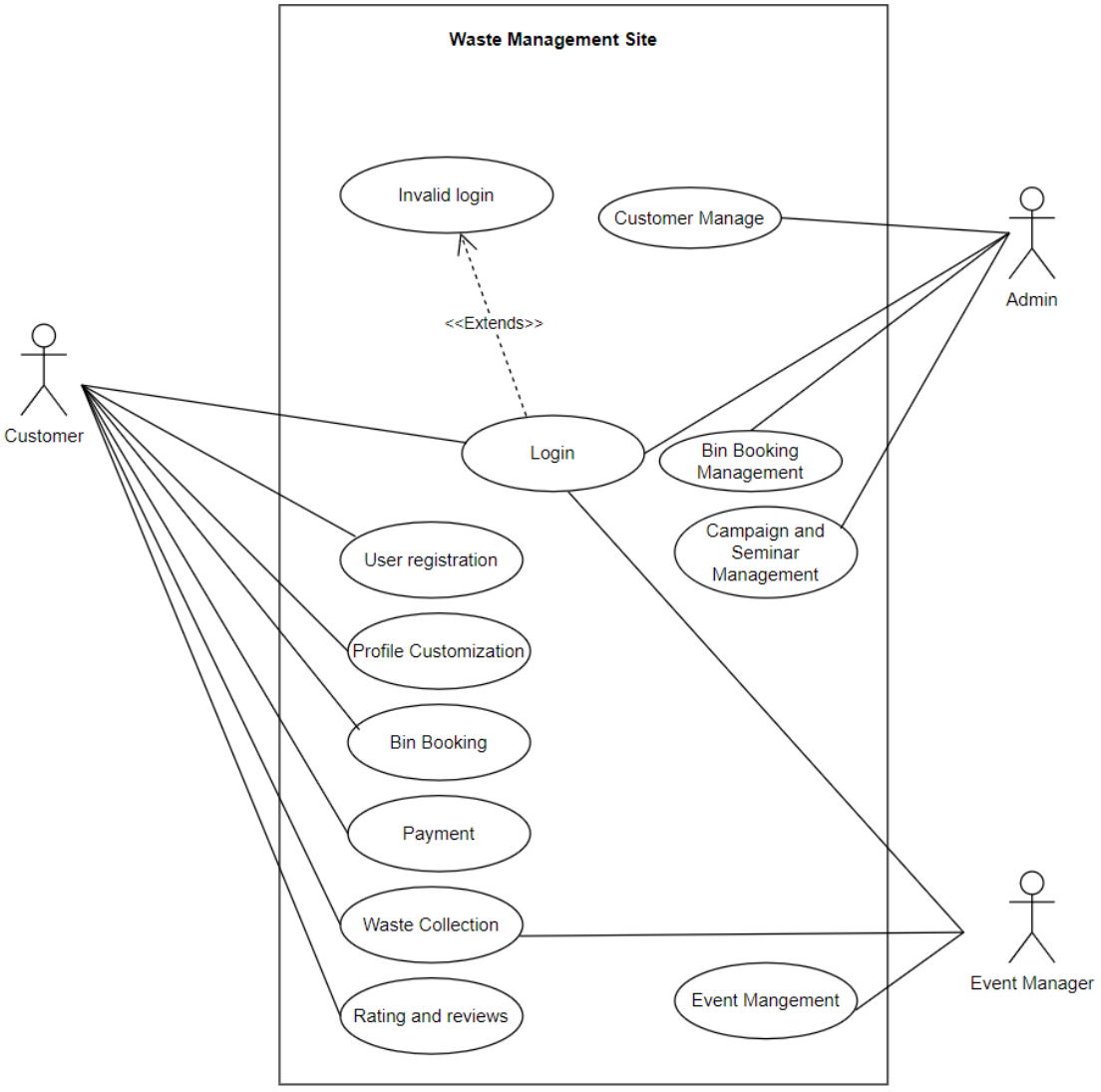


Fig 1: Use case diagram for EcoRecover.

4.2.1 SEQUENCE DIAGRAM

A sequence diagram is a type of interaction diagram in Unified Modeling Language (UML) used to visualize the interactions and communication sequences between various objects or components in a system over time. It illustrates the order in which messages or method calls are exchanged among these objects, providing a dynamic view of system behavior.

- Customer initiates the sequence. This could represent a user or actor interacting with the system.
- Client UI is the user interface through which the customer interacts with the system.
- Backend represents the server-side or backend components that process the requests and interactions from the user.
- User registration is the first action, and the diagram shows the steps involved in the registration process. The system registers the new user's information in the database. The system sends login credentials to the user.
- Login: The user proceeds to log in, and the system checks their credentials. The "check credential" step involves the system's validation of user login credentials, determining their accuracy. If successful, the process proceeds to "user found," indicating a successful login, but if the system can't verify the credentials, it leads to "user not found," signifying an unsuccessful login attempt.

- Register: If it's a new user, they can choose to register.

- process image: The system processes the image. In the "check condition" step, the system assesses if the uploaded image fulfills specified criteria. If it does meet the conditions, the expected outcome is recycling; however, if it doesn't meet the criteria, the result is a failure marked as "Not Recycle."

- Additionally, users who are not new have the option to proceed to "Bin Booking" in the sequence.

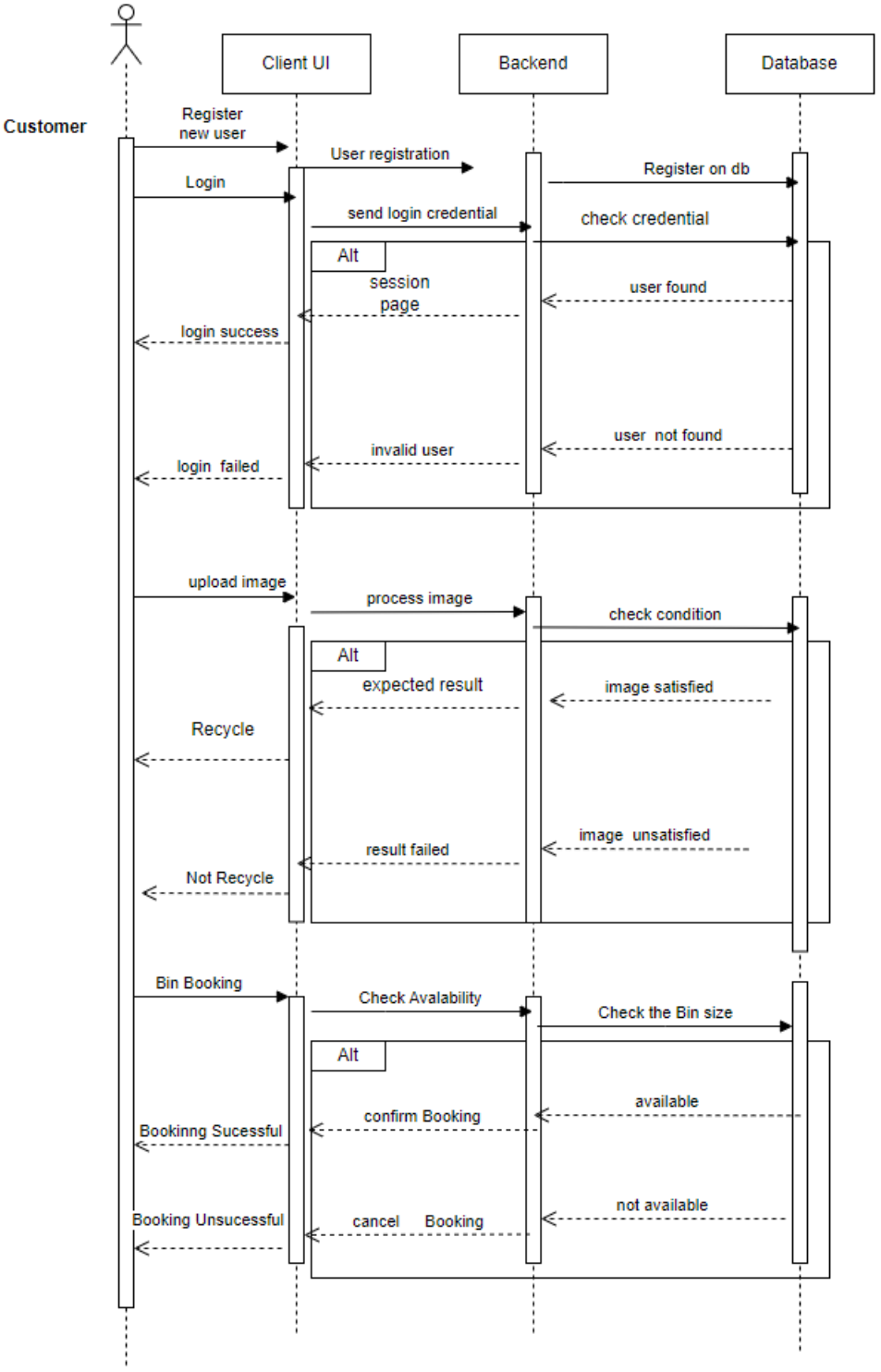


Fig 2: Sequence diagram for EcoRecover

4.2.2 State Chart Diagram

A state diagram is a visual representation of a finite-state machine or a state-transition diagram. It is a modeling technique used in various fields, including computer science, software engineering, and systems design, to describe the different states that an object, system, or process can exist in and how it can transition between those states.

In the context of the EcoRecover project, a State Chart Diagram can be used to depict the different states that users (Customers, Event Managers, and Administrators) and the system itself can go through during specific processes or interactions.

- New User: This represents the initial state where a person starts as a new user before interacting with the system.
- Login: This step indicates the process of logging in, where the user provides their credentials to access their account or specific features.
- Bin Booking: This state suggests that users can make bookings related to bins or waste management services. They may request a bin for a specific purpose, such as waste collection.
- Payment: This phase involves the financial transaction, where users might make payments for the services they've booked, which can include bin rentals, waste disposal fees, or related charges.
- Confirm: This step indicates that after the payment is made, users may need to confirm their booking or transaction. Confirmation might include verifying booking details and payment information.
- Complete: This final state suggests that the process, whether it's booking a bin or any other transaction, has been successfully completed. The user's interaction with the specific service or system is finished.

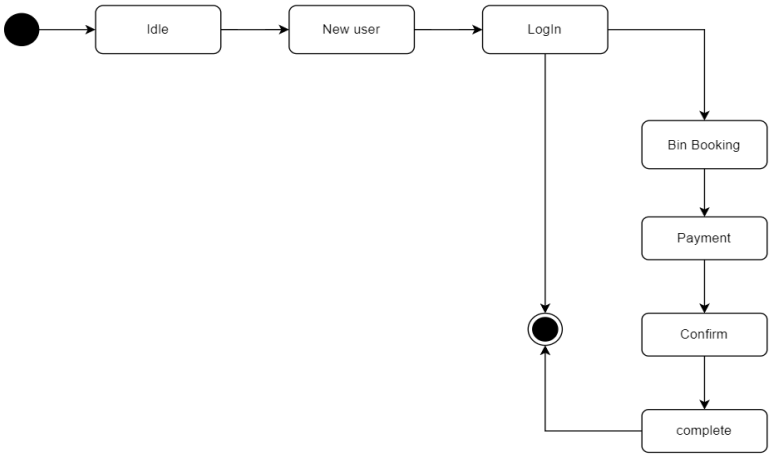


Fig 3: State Chart diagram for EcoRecover

4.2.2 Activity Diagram

An activity diagram is a powerful visual tool that represents the workflow of a system or process, illustrating how various activities or tasks interact and lead from one to another. In an activity diagram, each activity is considered a system operation, and the connections between them depict the order and flow of these operations. Activity diagrams offer a dynamic view of a system's behavior, showcasing how control flows from the initiation point through different actions, including decision points and parallel flows.

Key components of an activity diagram include:

- **Initial Node:** This serves as the starting point of the activity diagram and is marked by a solid black circle. It designates where the workflow begins.
- **Activity**: Each activity represents a specific task or action performed by the system or entity.
- **Control Flow**: Control flow, denoted by arrows, symbolizes the sequence in which activities or actions are executed.
- **Decision Node**: A decision node, often shown as a diamond shape, serves as a branching point in the activity flow.
- **Merge Node**: This node is used to consolidate multiple branches of activity flow back into a single path.
- **Fork Node**: A fork node splits the activity flow into multiple parallel branches, making it possible to depict concurrent processes.

- **Join Node**: A join node brings together multiple parallel flows into a single flow.
- **Final Node**: This node marks the conclusion of the activity diagram and is indicated by a solid black circle with a dot inside.
- **Object Flow**: Object flow, represented by dashed arrows, illustrates the movement of objects or data between activities, helping to clarify how information or data is transferred within the process.

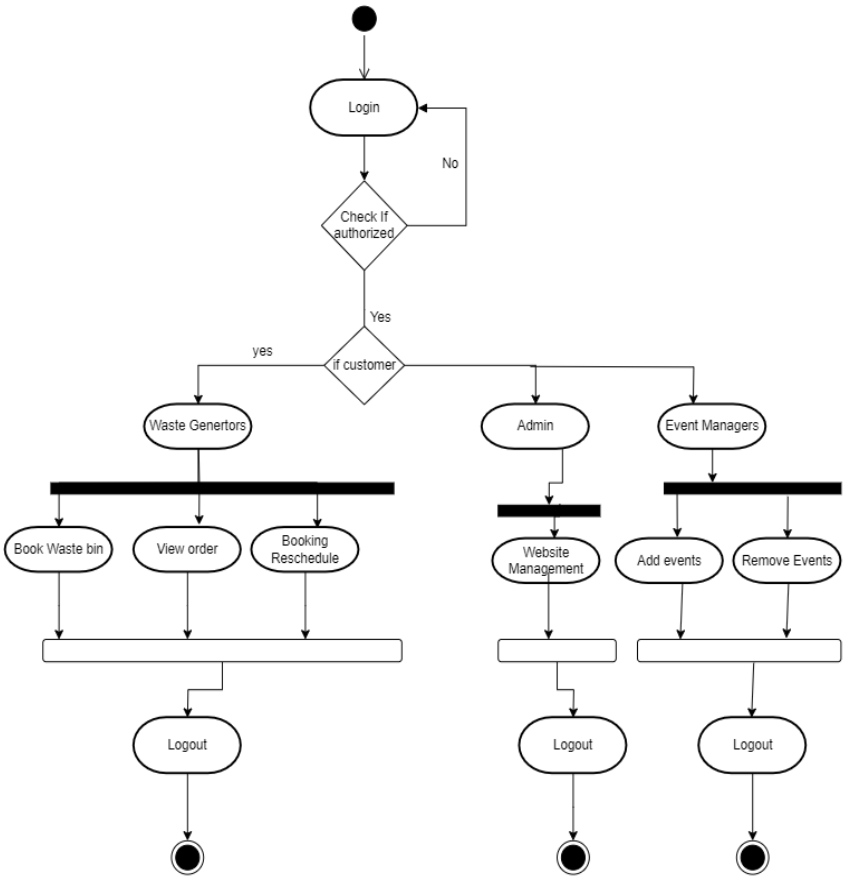


Fig 4: Activity diagram for EcoRecover

4.2.3 Class Diagram

The class diagram is a fundamental component of object-oriented modeling and serves as the primary means of conceptual modeling for the structure of an application. Additionally, class diagrams can be used for detailed modeling that can be translated into programming code. They can also be employed for data modeling purposes. Class diagrams are a crucial component of UML used to represent classes, objects, interfaces, and their relationships and attributes in a system. Some important components of a class diagram are:

- Class: It is a blueprint or template for creating objects and is represented as a rectangle with the class name, attributes, and methods.
- Interface: It is a collection of abstract methods that specify a contract between a class and the outside world. It is represented as a circle with the interface name inside.
- Object: It is an instance of a class with state and behavior. It is represented as a rectangle with the object name inside.
- Association: It is a relationship between two classes that represents a connection or link and is represented as a line with optional directionality, multiplicity, and role names.
- Aggregation: It is a part-whole relationship where the whole (aggregator) is composed of parts (aggregates) and is represented as a diamond shape on the aggregator side.
- Composition: It is a stronger form of aggregation where the parts cannot exist without the whole and is represented as a filled diamond shape on the aggregator side.
- Inheritance: It is a relationship between a superclass and its subclasses that represents an "is-a" relationship and is represented as a line with an open arrowhead pointing from the subclass to the superclass.
- Dependency: It is a relationship where a change in one class may affect the other class and is represented as a dashed line with an arrowhead pointing from the dependent class to the independent class.
- Multiplicity: It represents the number of instances of a class that can be associated with another class and is represented as a range of values near the association or aggregation line.

Class diagrams are essential in designing and modeling object-oriented software systems as they provide a visual representation of the system's structure, its functionality, and the relationships between its objects. They facilitate software development, maintenance, and improve communication among team members.

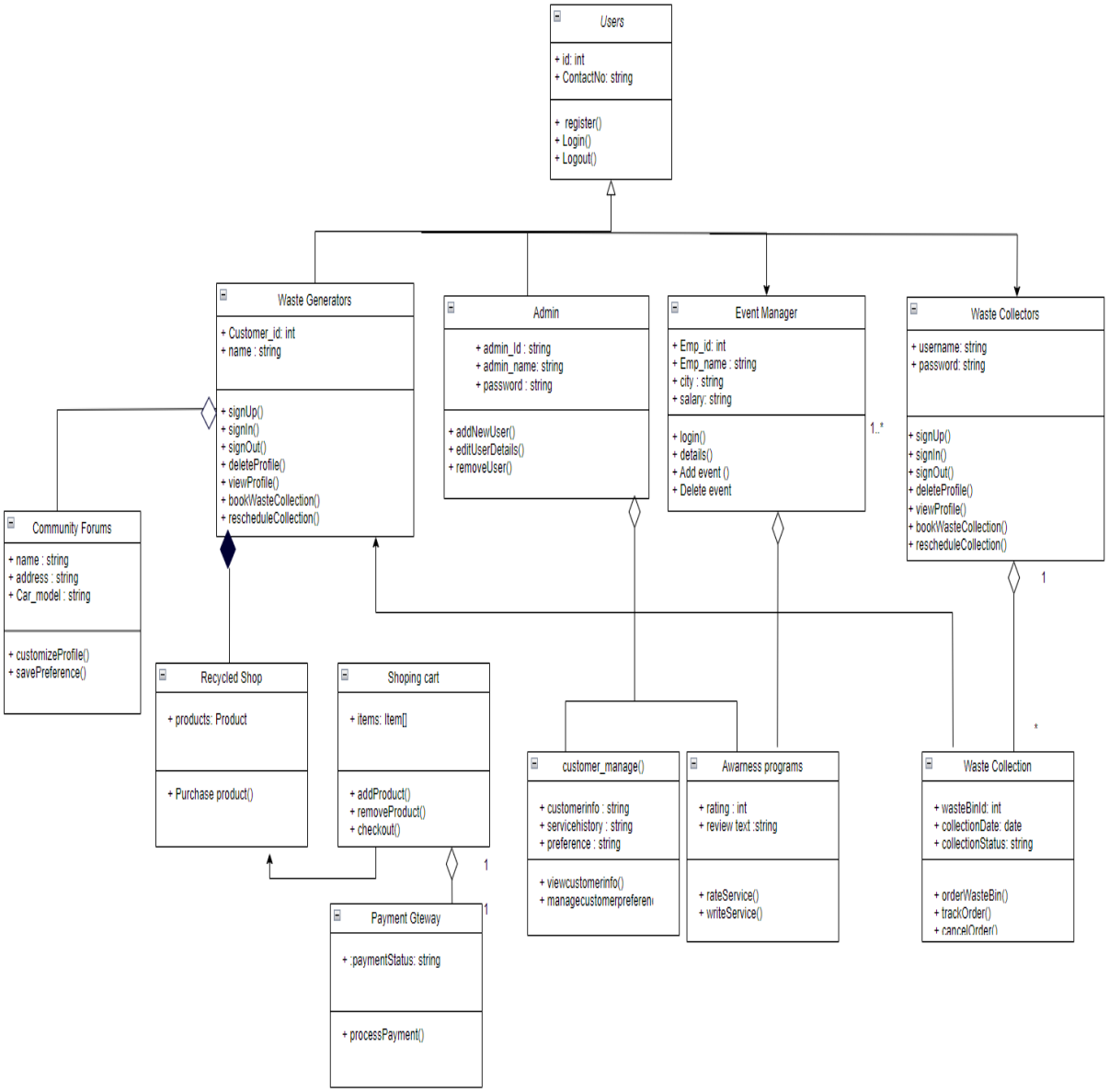


Fig 5: Class diagram for EcoRecover

4.2.4 Object Diagram

Class diagrams and object diagrams are closely related in object-oriented modeling. Object diagrams are instances of class diagrams, which represent a snapshot of the system at a given moment in time. Both types of diagrams use the same concepts and notation to represent the structure of a system. While class diagrams are used to model the structure of the system, including its classes, attributes, and methods, object diagrams represent a group of objects and their connections at a specific point in time.

An object diagram is a type of structural diagram in UML that shows instances of classes and their relationships. The main components of an object diagram include:

- Object: An object is an instance of a class that represents a specific entity in the system. It is represented as a rectangle with the object name inside.
- Class: A class is a blueprint or template for creating objects that defines its attributes and methods. It is represented as a rectangle with three compartments for the class name, attributes, and methods.
- Link: A link is a relationship between two objects that represents a connection or association. It is represented as a line connecting two objects with optional labels.
- Attribute: An attribute is a property or characteristic of an object that describes its state. It is represented as a name-value pair inside the object rectangle.
- Value: A value is a specific instance or setting of an attribute. It is represented as a value inside the attribute name-value pair.
- Operation: An operation is a behavior or action that an object can perform. It is represented as a method name inside the class rectangle.
- Multiplicity: Multiplicity represents the number of instances of a class that can be associated with another class. It is represented as a range of values (e.g. 0..1, 1..*, etc.) near the link between objects.

Object diagrams help to visualize the relationships between objects and their attributes in a system. They are useful for understanding the behavior of a system at a specific point in time and for identifying potential issues or inefficiencies in the system

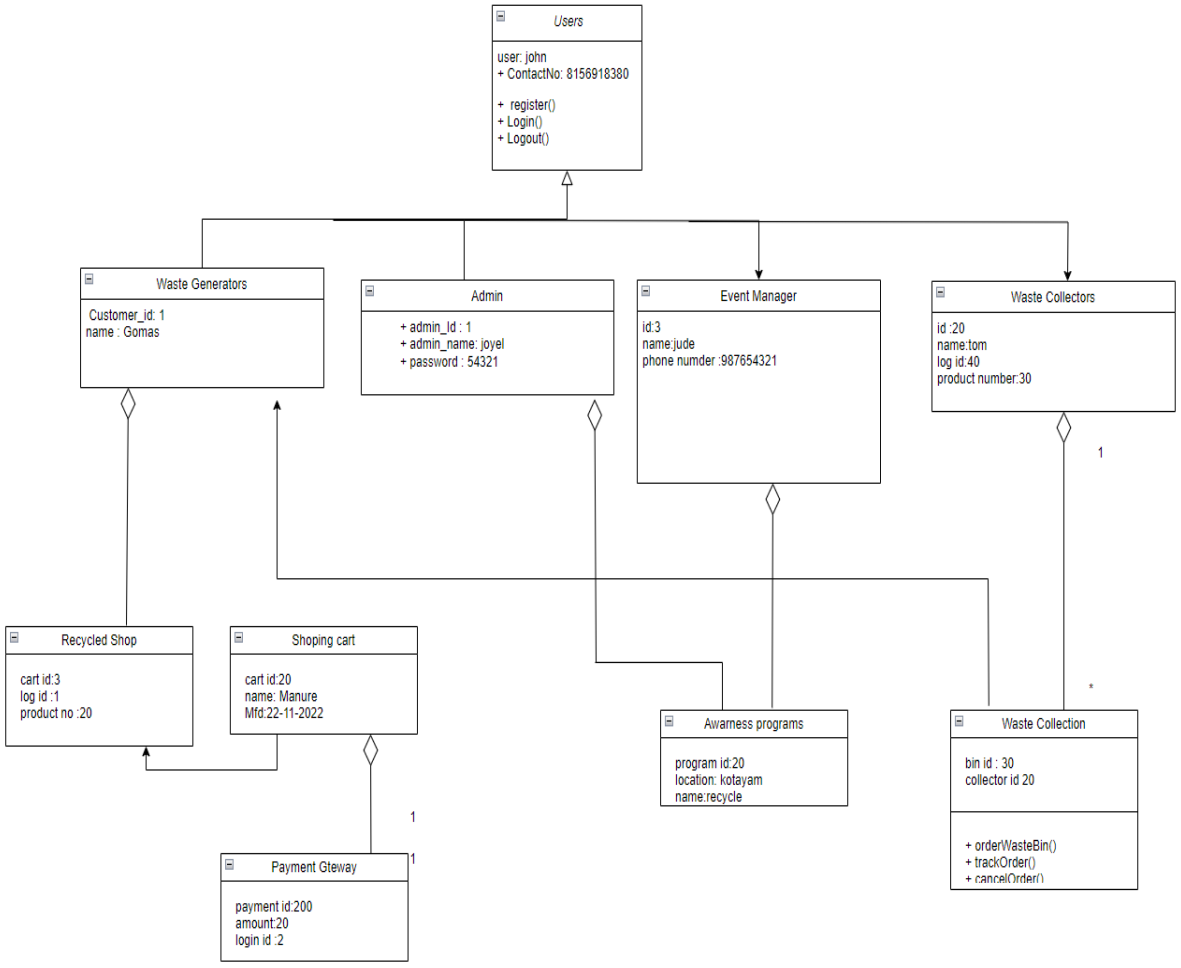


Fig 6: Object diagram for EcoRecover

4.2.5 Component Diagram

A component diagram in UML illustrates how various components are interconnected to create larger components or software systems. It is an effective tool for representing the structure of complex systems with multiple components. By using component diagrams, developers can easily

visualize the internal structure of a software system and understand how different components work together to accomplish a specific task.

Its key components include:

- Component: A modular and encapsulated unit of functionality in a system that offers interfaces to interact with other components. It is represented as a rectangle with the component name inside.
- Interface: A contract between a component and its environment or other components, specifying a set of methods that can be used by other components. It is represented as a circle with the interface name inside.
- Port: A point of interaction between a component and its environment or other components. It is represented as a small square on the boundary of a component.
- Connector: A link between two components that enables communication or data exchange. It is represented as a line with optional adornments and labels.
- Dependency: A relationship between two components where one component depends on another for its implementation or functionality. It is represented as a dashed line with an arrowhead pointing from the dependent component to the independent component.
- Association: A relationship between two components that represents a connection or link. It is represented as a line connecting two components with optional directionality, multiplicity, and role names.
- Provided/Required Interface: A provided interface is an interface that a component offers to other components, while a required interface is an interface that a component needs from other components to function properly. These are represented by lollipops and half-circles respectively.

Component diagrams are useful for modeling the architecture of a software system, and can help identify potential issues and improvements in the design. They can also be used to communicate the structure and behavior of a system to stakeholders, such as developers and project managers.

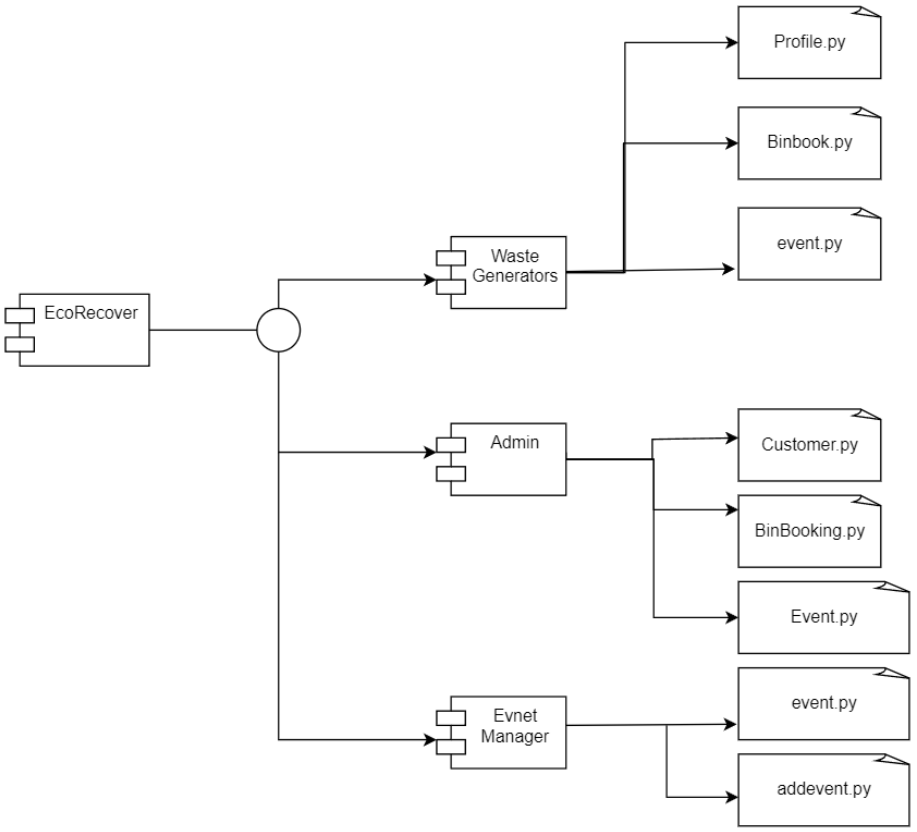


Fig 7: Component diagram for EcoRecover

4.2.8 Deployment Diagram

A deployment diagram in the Unified Modeling Language (UML) provides a visual representation of how software components and artifacts are mapped onto the physical hardware or nodes within a system's architecture. It serves as a powerful tool to illustrate how software applications or services are distributed across servers, machines, or devices. The diagram not only highlights the relationships between nodes and components but also specifies deployment details such as version, location, and configuration parameters through deployment specifications.

Key components of a deployment diagram include:

- Node: Represented as a named box, nodes can be physical or virtual entities where software components are deployed, showcasing the hardware aspects of a system.

- Component: Components are software entities responsible for specific functions,

displayed within rectangles along with their names, emphasizing the software architecture.

- Artifact: Depicted in rectangles with their names, artifacts represent data or files associated with a component, elucidating the data flow.

- Deployment Specification: These specifications detail how components or artifacts are deployed on nodes, including specific configuration and version information.

- Association: Associations establish relationships between nodes and components or artifacts, depicting deployment dependencies, which can be unidirectional or bidirectional with multiplicity and role names.

- Communication Path: Represented by lines with optional labels and adornments, communication paths illustrate connections between nodes, signifying network links or communication channels.

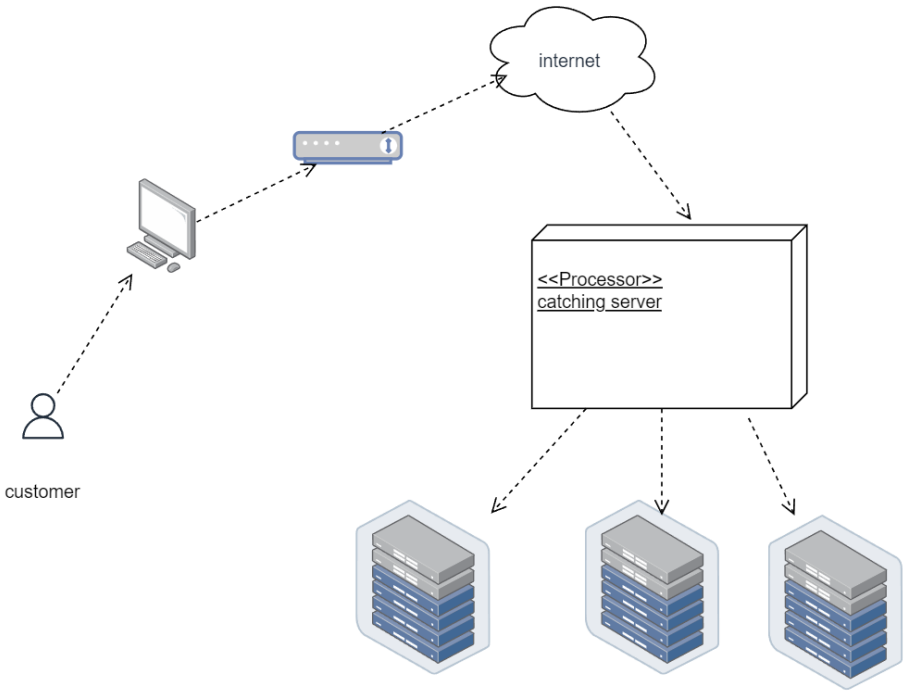


Fig 8: Deployment diagram for EcoRecover

4.2.9 Collaboration Diagram

Collaboration diagrams provide a visual representation of how objects within a system are interconnected, focusing on the structural relationships between objects. They are valuable tools for understanding the architecture and interactions in complex software systems, aiding in system design and documentation. These diagrams enhance communication among development teams and facilitate a shared understanding of object-oriented system structures, streamlining the development process.

Collaboration diagrams encompass key components that facilitate the visual representation of interactions within a system. These components include objects, actors, links, and messages, each with a distinct role in depicting the relationships and communications among elements:

- Objects: Objects are symbolized by names paired with their corresponding classes, differentiating class instances within the system. While not every class requires an object representation, naming objects is essential for clear identification.
- Actors: Actors initiate interactions and are identified by names and designated roles. One actor typically commences the use case in the diagram, playing a pivotal role in system interactions.
- Links: Links, often instances of associations, connect objects and actors, defining the relationships between them. They serve as pathways for message transmission and are depicted as solid lines, aiding objects in navigating through the system.
- Messages: Messages are the means of communication between objects, carrying information and actions. These messages are indicated by sequence numbers and labeled arrows, denoting the sender and receiver of the message, with a clear direction and content comprehension.

Collaboration diagrams offer a clear and structured visual representation of system interactions, making them a valuable tool for understanding complex relationships and communication patterns within an object-oriented system.

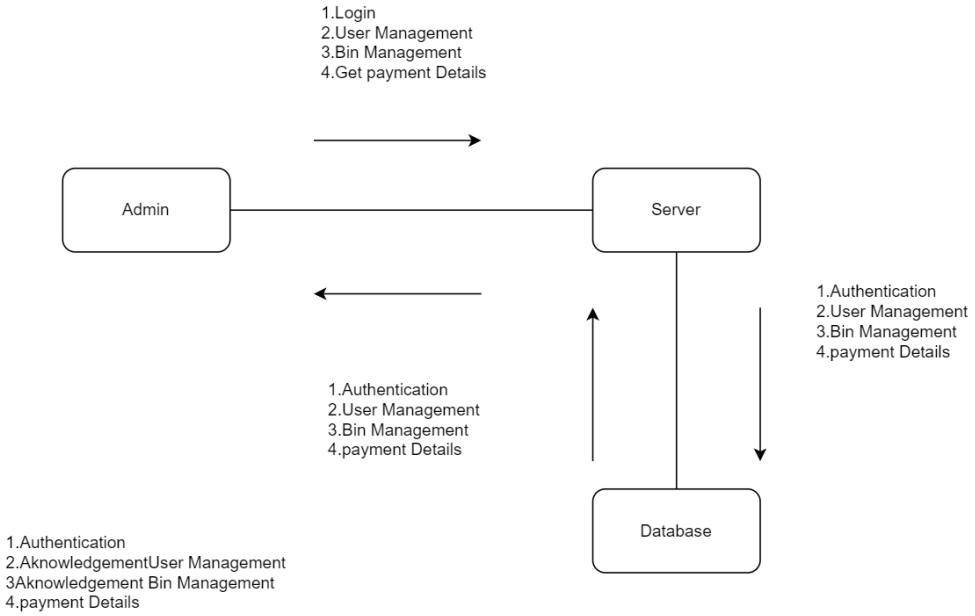
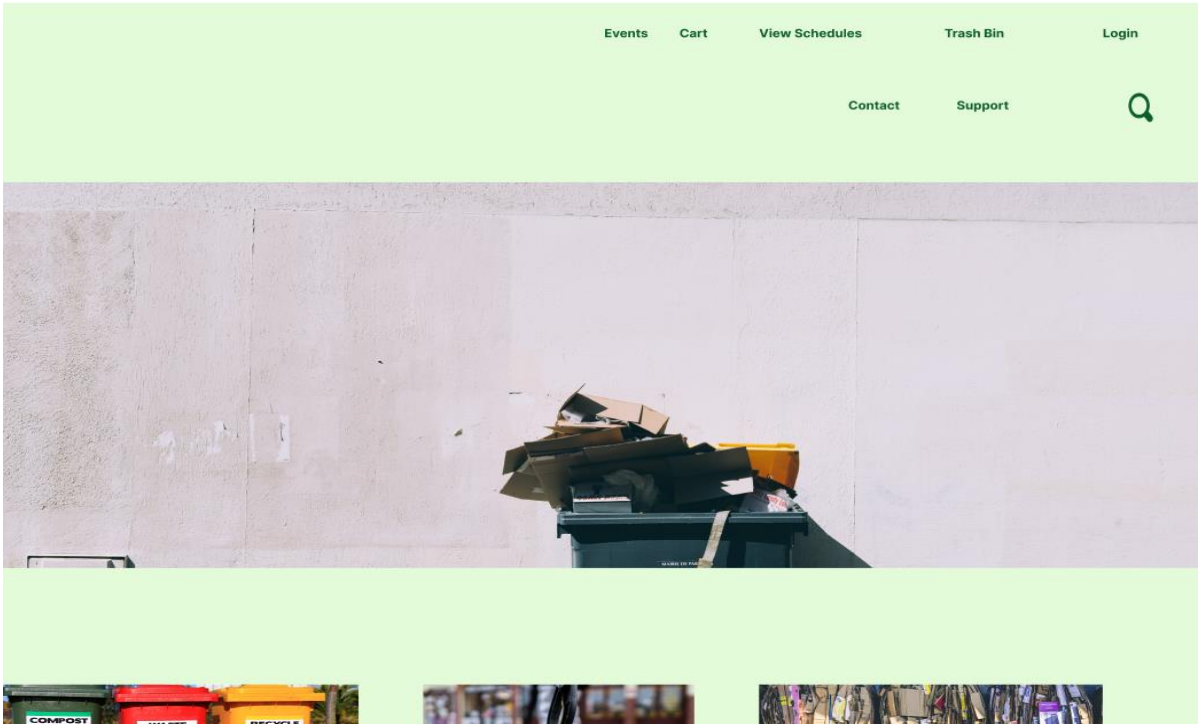


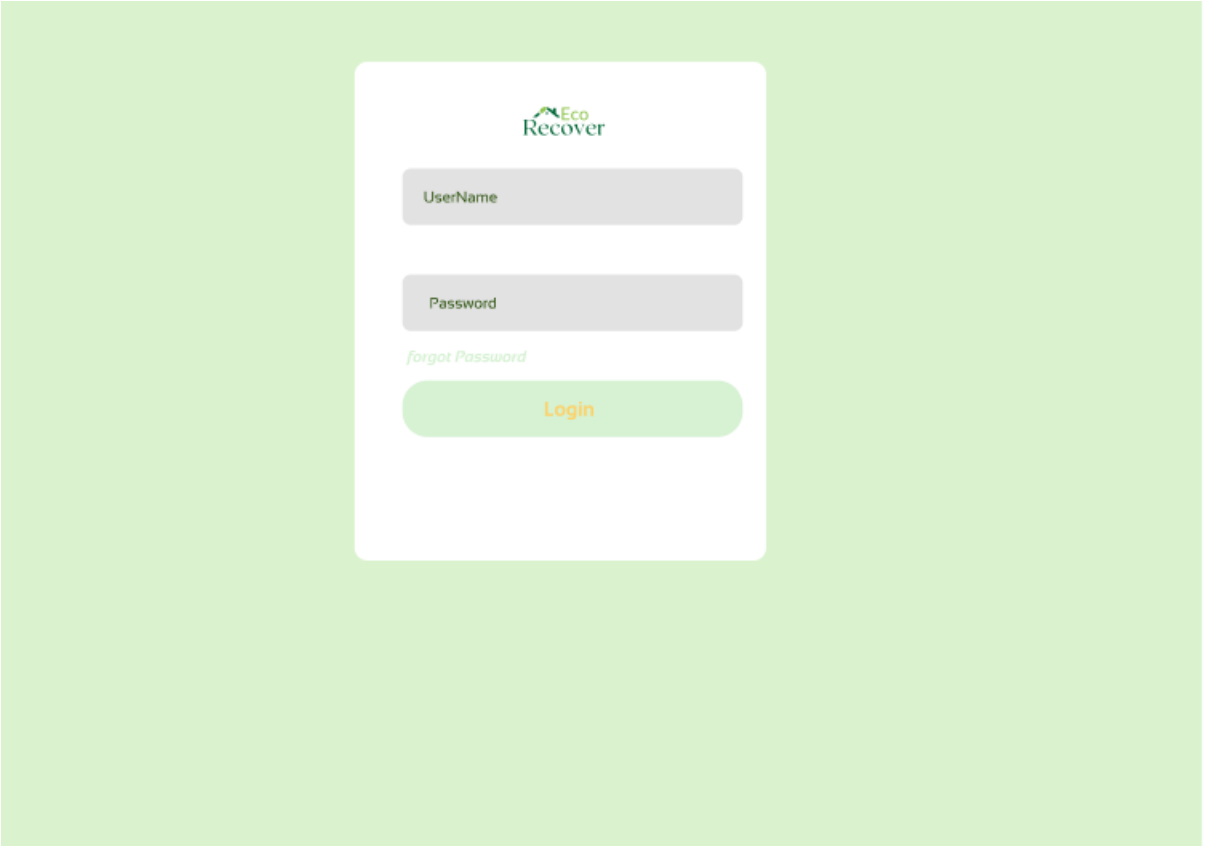
Fig 9: Collaboration diagram for EcoRecover

4.3 USER INTERFACE DESIGN USING FIGMA

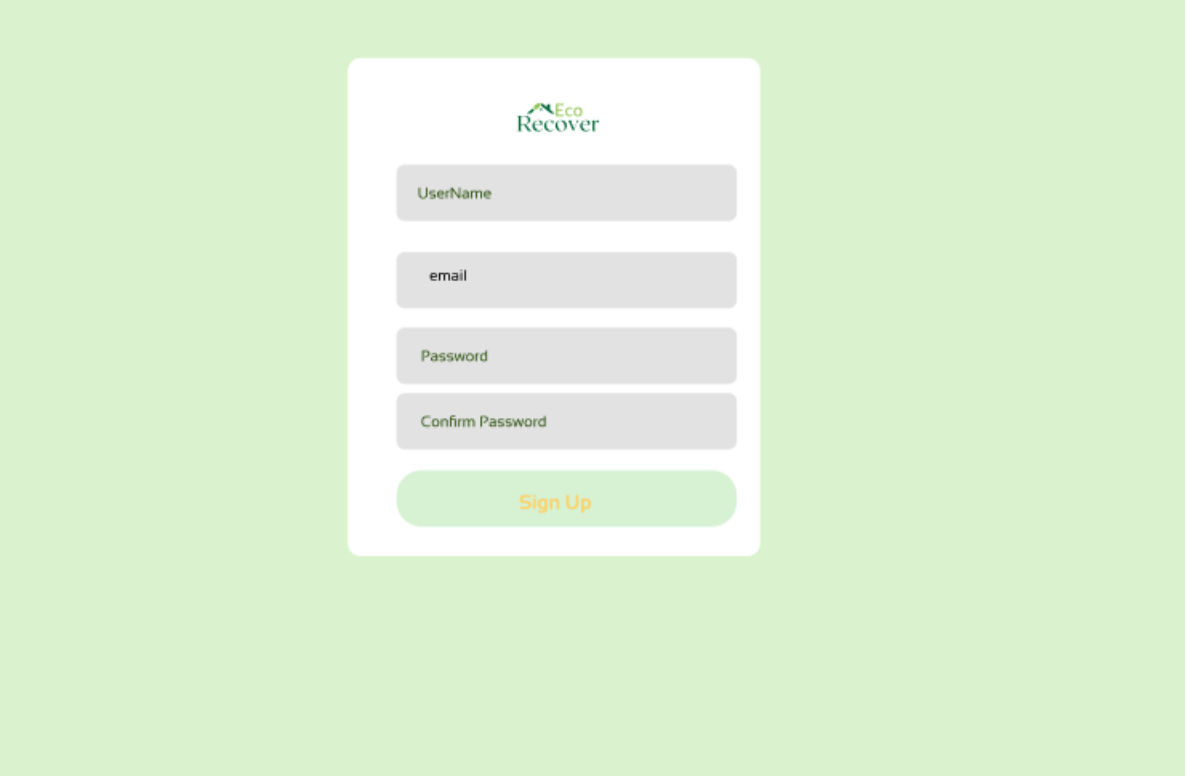
1. Form Name: Home Page



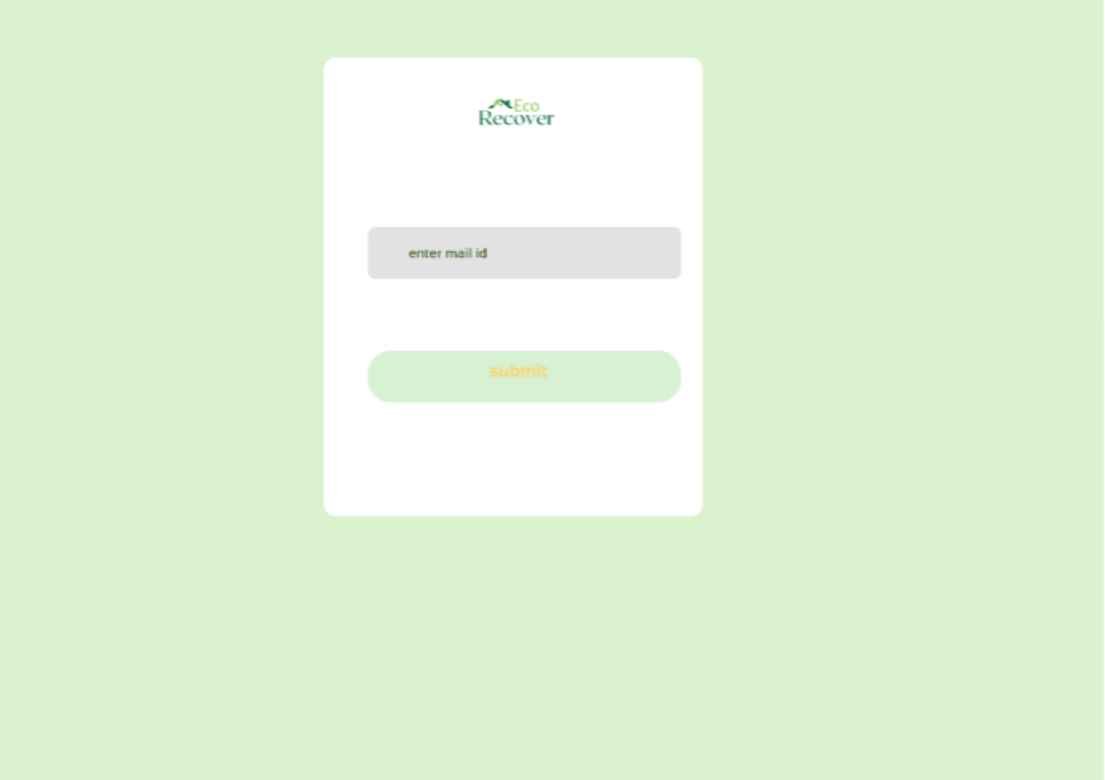
2. Form Name: Login Page



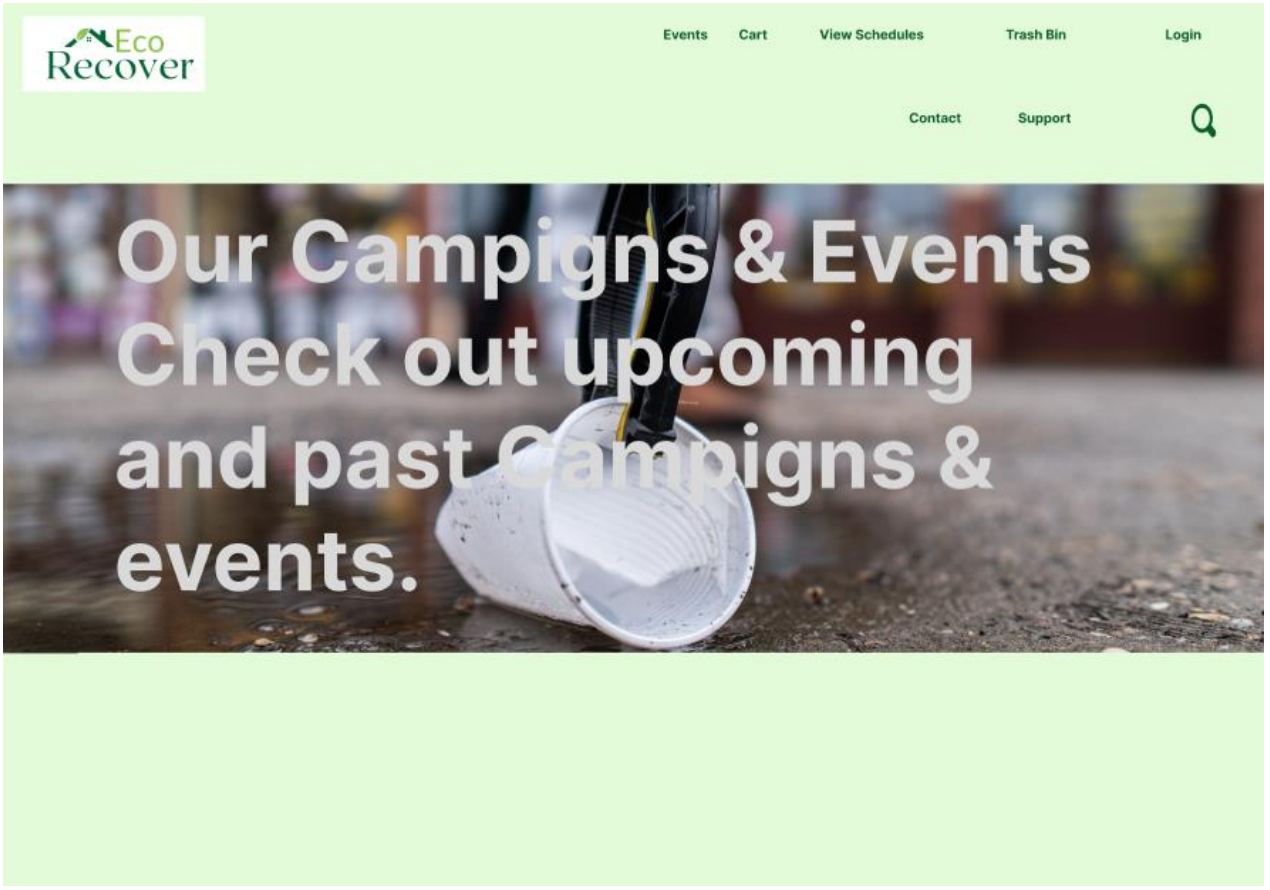
3. Form Name: Registration Page



4. Form Name: Reset password Page



5. **Form Name: Event page**



4.4 DATABASE DESIGN

A database plays a crucial role as an organized repository of information, meticulously structured to facilitate easy access, efficient management, and seamless updates. One of the paramount goals in any database system is to ensure the security of the stored data. The process of designing a database typically comprises two distinct phases. In the initial phase, the focus is on gathering and analyzing user requirements to create a database that aligns with their needs as seamlessly as possible. This phase, often referred to as the conceptual design phase, is conducted independently of any specific Database Management System (DBMS). Subsequently, the design evolves from a conceptual level into a concrete DBMS design that is tailored to effectively implement the system. This later phase, known as physical design, takes into consideration the specific features and requirements of the chosen DBMS. Together with system design, database design plays a pivotal role in striving to achieve two primary objectives: maintaining data integrity and ensuring data independence.

4.4.1 Relational Database Management System (RDBMS)

A Relational Database Management System (RDBMS) stands as a prevalent database system that organizes data into tables, streamlining the connections between various datasets. These tables have the capacity to store vast amounts of data, spanning from hundreds to millions of rows, which are formally known as records. In the language of the relational model, a row is referred to as a tuple, a column header is designated as an attribute, and the table itself is termed a relation. A relational database consists of multiple tables, each having a unique name. Within each table, every row represents a collection of interconnected values. In the realm of a relational database, relationships between tables are predetermined to ensure both referential and entity relationship integrity. A domain, marked as D, encompasses a set of fundamental, indivisible values, typically defined by selecting a specific data type that characterizes its data values. Assigning a name to a domain simplifies comprehension of the nature of its values. Each value within a relation is considered atomic and cannot be further divided.

Within a relational database, table relationships are established through the use of keys, with primary keys and foreign keys holding paramount importance. These keys serve as the linchpins for ensuring entity integrity and referential integrity within the database. Entity integrity guarantees that primary keys cannot contain null values, while referential integrity dictates that every unique foreign key value must correspond to a matching primary key value within the same domain. Furthermore, other key types, including super keys and candidate keys, also play pivotal roles in shaping the interconnections within the database.

4.4.2 Normalization

Data organization is a fundamental aspect of database design, and one key method to achieve this is through normalization. Normalization is a formal process aimed at optimizing data structures by reducing redundancy and enhancing data integrity. By employing normalization techniques, redundant or unnecessary fields are eliminated, and large tables are broken down into smaller, more manageable ones. Furthermore, normalization safeguards against anomalies during data insertion, deletion, and updates. In this process, two critical concepts come into play: keys and relationships. A key uniquely identifies a row within a table, and within this context, primary keys and foreign keys play distinct roles. A primary key is an attribute, or a combination of attributes, in a table that acts to distinguish records within the same table. Conversely, a foreign key is a column within a table used to uniquely identify records in other related tables. Normalization, often applied up to the third normal form, ensures that data is organized efficiently and without unnecessary redundancy. Normalization is a process in database design that aims to organize data into proper tables and columns, making it easily correlated to the data by the user. This process eliminates data redundancy that can be a burden on computer resources.

4.4.3 Sanitization

Data sanitization is a crucial process aimed at eliminating any unauthorized or malicious characters or values from data. In the context of web applications, safeguarding against security vulnerabilities requires thorough user input sanitization. PHP, a widely used server-side scripting language, offers an integrated filter extension specifically designed for the sanitization and validation of diverse types of external input, such as email addresses, URLs, IP addresses, and more. These filters simplify and speed up data sanitization, ensuring that only legitimate and safe information is accepted. As an example, PHP's filter extension includes a function capable of stripping away all characters except letters, digits, and specific special characters, as defined by a designated flag. Web applications continuously receive external input from various sources, including user-submitted data from forms, cookies, information from web services, server variables, and results from database queries. Therefore, it's paramount to systematically sanitize all external input to maintain a secure environment and mitigate the risk of potentially harmful code or values infiltrating the system.

4.4.4 Indexing

An index serves as a fundamental database structure designed to significantly enhance the efficiency of table operations. These indices can be created on one or more columns within a table, enabling swift data retrieval and streamlined record ordering. The strategic creation of indexes plays a pivotal role in optimizing database performance, especially in scenarios where specific columns are frequently involved in SQL queries. In practice, indexes can be thought of as a separate, hidden table that houses a primary key or an indexed field alongside pointers to actual records within the main table. Importantly, these index structures remain hidden from end-users and are exclusively utilized by the database's search engine, aiding it in the rapid location of records. The process of creating indexes is executed using the "CREATE INDEX" statement in SQL. It's essential to bear in mind that while indexes significantly improve SELECT statement performance, making data retrieval faster, they can introduce a trade-off. When tables have indexes, the execution of INSERT and UPDATE statements may take slightly longer.

4.5 TABLE DESIGN

Table 1: User

Primary key: **user_id**

Field Name	Data Type	Constraints
user_id	Primary Key (PK)	
username	CharField	Max length: 150
password	CharField	Max length: 128
role	CharField	Max length: 50

Table 2: UserProfile

Primary key: **userprofile_id**

Foreign key: **user_id** that references the **User.**

Field Name	Data Type	Constraints
user_id	Primary Key (PK)	
address	CharField	Max length: 255
mobile_number	CharField	Max length: 20, Default: "None"

Table 3: Bin_For_Home

Primary key: **Bin_id**

Field Name	Data Type	Constraints
bin_id	Primary Key (PK)	
title	CharField	Max length: 100
size	CharField	Max length: 50
Capacity	CharField	Max length: 50
description	TextField	
image	ImageField	Blank: True

Table 4: Bin_For_Events

Primary key: **bin_id**

Field Name	Data Type	Constraints
bin_id	Primary Key (PK)	
title	CharField	Max length: 100
size	CharField	Max length: 50
capacity	CharField	Max length: 50
description	TextField	
image	ImageField	Upload to 'bin_images/', Null: True, Blank: True

Table 5: Bin_Booking_for_home

Primary key: **booking_id**

Foreign key: **user_id** references **User**, **bin_id** references **Bin**

Field Name	Data Type	Constraints
booking_id	Primary Key (PK)	Unique
user_id	Foreign Key (FK)	References User (user_id), on_delete: CASCADE
bin_id	Foreign Key (FK)	References Bin (bin_id), on_delete: CASCADE

house_number	CharField	Max length: 50
landmark	CharField	Max length: 100, Default: None
pin_code	CharField	Max length: 10
bin_size	CharField	Max length: 20
bin_capacity	CharField	Max length: 20
collection_period	CharField	Max length: 20, Default: None

Table 6: BinBooking_For_Event

Primary key: **booking_id**

Foreign key: bin references binEvent.

Field Name	**Data Type**	**Constraints**
booking_id	AutoField (Primary Key)	Unique
bin	Foreign Key to BinEvent	References BinEvent (bin_id), on_delete: CASCADE
event_date_time	DateTimeField	Max Length: 255
event_location	CharField ()	Max Length: 255
delivery_time	DateTimeField	Auto_now_add: True
pickup_time	DateTimeField	Auto_now_add: True
number_of_bins_needed	PositiveIntegerField	

Table 7: Booked Bin Status

Primary key: **Status id**

Foreign key: **booking_id** references **Booked Bin**

Field Name	**Data Type**	**Constraints**
status_id	Primary Key (PK)	
booking_id	Foreign Key (FK)	References Bin Booking (booking_id), on_delete: CASCADE
fill_level	PositiveIntegerField	Choices: 20, 40, 50, 70, 90
created_at	DateTimeField	Auto_now_add: True
updated_at	DateTimeField	Auto_now: True

Table 8: Event

Primary key: **event_id**

Foreign key: **User_id** references **User table**

Field Name	Data Type	Constraints
event_id	Primary Key (PK)	
user_id	Foreign Key (FK)	References User (user_id), on_delete: CASCADE
name	CharField	Max length: 200
description	TextField	
date	DateField	
time	TimeField	
location	CharField	Max length: 200
max_participants	IntegerField	Default: 0
image	ImageField	Upload to 'event_images/', Null: True, Blank: True

Table 9: Event_Booking

Primary key: **booking_id**

Foreign key: **user_id** references **User**

Field Name	**Data Type**	**Constraints**
booking_id	Primary Key (PK)	
user_id	Foreign Key (FK)	References User (user_id), on_delete: CASCADE
event_id	Foreign Key (FK)	References Event (event_id), on_delete: CASCADE
attendees	PositiveIntegerField	

Table 10: Event_Category

Primary key: category_id

Foreign key: event_id references **Event**

Field Name	**Data Type**	**Constraints**

category_id	Primary Key (PK)	
event_id	Foreign Key (FK)	References EventDetails (event_id), on_delete: CASCADE
category	CharField	Max length: 50

Table 11: Payment

Primary key: **Payment_id**

Foreign key: **user_id** references **User**

Field Name	Data Type	Constraints
payment_id	Primary Key (PK)	
user_id	Foreign Key (FK)	References User (PK)
order_id	CharField	
signature	CharField	
amount	DecimalField	
timestamp	DateTimeField	auto-generated on creation

Table 12: feedback

Primary key: **feedback_id**

Foreign key: **user_id** references **User**

Field Name	Data Type	Constraints
feedback_id	Primary Key (PK)	Unique identifier for feedback
user_id	Foreign Key (FK)	References User (PK)
star_rating	IntegerField	
message	TextField	Maxlength:200

CHAPTER 5
SYSTEM TESTING

5.1 INTRODUCTION

Software testing involves executing a software program in a controlled manner to determine if it behaves as intended, often using verification and validation methods. Validation involves evaluating a product to ensure it complies with specifications, while verification can involve reviews, analyses, inspections, and walkthroughs. Static analysis examines the software's source code to identify issues, while dynamic analysis examines its behavior during runtime to gather information like execution traces, timing profiles, and test coverage details.

Testing involves a series of planned and systematic activities that start with individual modules and progress to the integration of the entire computer-based system. The objectives of testing include identifying errors and bugs in the software, ensuring that the software functions according to its specifications, and verifying that it meets performance requirements. Testing can be performed to assess correctness, implementation efficiency, and computational complexity.

A successful test is one that detects an undiscovered error, and a good test case has a high probability of uncovering such errors. Testing is crucial to achieving system testing objectives and can involve various techniques such as functional testing, performance testing, and security testing.

5.2 TEST PLAN

A test plan is a document that outlines the required steps to complete various testing methodologies. It provides guidance on the activities that need to be performed during testing. Software developers create computer programs, documentation, and associated data structures. They are responsible for testing each component of the program to ensure it meets the intended purpose. To address issues with self-evaluation, an independent test group (ITG) is often established.

Testing objectives should be stated in quantifiable language, such as mean time to failure, cost to find and fix defects, remaining defect density or frequency of occurrence, and test work-hours per regression test.

The different levels of testing include:

- Unit testing
- Integration testing
- Data validation testing

- Output testing

5.2.1 Unit Testing

Unit testing is a software testing technique that focuses on verifying individual components or modules of the software design. The purpose of unit testing is to test the smallest unit of software design and ensure that it performs as intended. Unit testing is typically white-box focused, and multiple components can be tested simultaneously. The component-level design description is used as a guide during testing to identify critical control paths and potential faults within the module's perimeter.

During unit testing, the modular interface is tested to ensure that data enters and exits the software unit under test properly. The local data structure is inspected to ensure that data temporarily stored retains its integrity during each step of an algorithm's execution. Boundary conditions are tested to ensure that all statements in a module have been executed at least once, and all error handling paths are tested to ensure that the software can handle errors correctly.

Before any other testing can take place, it is essential to test data flow over a module interface. If data cannot enter and exit the system properly, all other tests are irrelevant. Another crucial duty during unit testing is the selective examination of execution pathways to anticipate potential errors and ensure that error handling paths are set up to reroute or halt work when an error occurs. Finally, boundary testing is conducted to ensure that the software operates correctly at its limits.

In the Sell-Soft System, unit testing was carried out by treating each module as a distinct entity and subjecting them to a variety of test inputs. Any issues with the internal logic of the modules were fixed, and each module was tested and run separately after coding. Unused code was eliminated, and it was confirmed that every module was functional and produced the desired outcome.

5.2.2 Integration Testing

Integration testing is a systematic approach that involves creating the program structure while simultaneously conducting tests to identify interface issues. The objective is to construct a program structure based on the design that uses unit-tested components. The entire program is then tested. Correcting errors in integration testing can be challenging due to the size of the overall program, which makes it difficult to isolate the causes of the errors. As soon as one set of errors is fixed, new ones may arise, and the process may continue in an apparently endless cycle.

Once unit testing is complete for all modules in the system, they are integrated to check for any interface inconsistencies. Any discrepancies in program structures are resolved, and a unique program structure is developed.

5.2.3 Validation Testing or System Testing

The final stage of the testing process involves testing the entire software system, including all forms, code, modules, and class modules. This is commonly referred to as system testing or black box testing. The focus of black box testing is on testing the functional requirements of the software. A software engineer can use this approach to create input conditions that will fully test each program requirement. The main types of errors targeted by black box testing include incorrect or missing functions, interface errors, errors in data structure or external data access, performance errors, initialization errors, and termination errors.

5.2.4 Output Testing or User Acceptance Testing

User acceptance testing is performed to ensure that the system meets the business requirements and user needs. It is important to involve the end users during the development process to ensure that the software aligns with their needs and expectations. During user acceptance testing, the input and output screen designs are tested with different types of test data. The preparation of test data is critical to ensure comprehensive testing of the system. Any errors identified during testing are addressed and corrected, and the corrections are noted for future reference.

5.2.5 Automation Testing

Automation testing is a software testing approach that employs specialized automated testing software tools to execute a suite of test cases. Its primary purpose is to verify that the software or equipment operates precisely as intended. Automation testing identifies defects, bugs, and other issues that may arise during product development.

While some types of testing, such as functional or regression testing, can be performed manually, there are numerous benefits to automating the process. Automation testing can be executed at any time of day and uses scripted sequences to evaluate the software. The results are reported, and this information can be compared to previous test runs. Automation developers typically write code in programming languages such as C#, JavaScript, and Ruby.

5.2.6 Selenium Testing

Selenium is an open-source automated testing framework used to verify web applications across different browsers and platforms. Selenium allows for the creation of test scripts in various programming languages such as Java, C#, and Python. Jason Huggins, an engineer at Thought Works, developed Selenium in 2004 while working on a web application that required frequent testing. He created a JavaScript program called "JavaScriptTestRunner" to automate browser actions and improve testing efficiency. Selenium has since evolved and continues to be developed by a team of contributors.

Python Selenium is a popular implementation of Selenium that leverages Python as the scripting language for test automation. Python is known for its simplicity and readability, making it a preferred choice among testers and developers for creating Selenium test scripts. Python Selenium offers a user-friendly and powerful way to interact with web browsers. Testers can use Python Selenium to automate various web testing tasks, including form submissions, navigation, clicking links, and validating web elements. Python Selenium scripts can be written quickly, and the Python ecosystem provides a wide array of libraries and tools that can be integrated seamlessly with Selenium. This combination of Selenium's capabilities and Python's flexibility makes Python Selenium a strong choice for web application testing, especially when combined with the extensive support and resources available within the Python community.

Test Case 1

Code:

```
from django.test import TestCase
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
class Hosttest(TestCase):

    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.implicitly_wait(10)
        self.live_server_url = 'http://127.0.0.1:8000/'

    def tearDown(self):
        self.driver.quit()



    def test_01_login_page(self):
        driver = self.driver
        driver.get(self.live_server_url)
        driver.maximize_window()
        time.sleep(2)
        theme=driver.find_element(By.CSS_SELECTOR,"a[href='/loginn']")
        theme.click()
        time.sleep(4)
        elem = driver.find_element(By.NAME, "username")
        elem.send_keys("Awin")
        elem = driver.find_element(By.NAME, "password")
        elem.send_keys("12045#CEah")
        submit_button = driver.find_element(By.CSS_SELECTOR, "#btn_login")
        submit_button.click()
        time.sleep(5)
        driver.execute_script("window.scrollBy(0, 2900);")
        time.sleep(5)
        browse=driver.find_element(By.CSS_SELECTOR,"a[href='/show_events/']")
        browse.click()
        time.sleep(5)
        driver.execute_script("window.scrollBy(0, 2900);")
        time.sleep(5)
        view=driver.find_element(By.CSS_SELECTOR,"a[href^='/book_event/']")
        view.click()
        time.sleep(1)
        driver.execute_script("window.scrollBy(0, 2900);")
```

```
        time.sleep(4)
        elem = driver.find_element(By.NAME, "attendees")
        elem.send_keys("4")
        chap = driver.find_element(By.CSS_SELECTOR, "button.btn.btn-primary.btn-
register")
        chap.click()
        time.sleep(1)
        options = driver.find_element(By.CSS_SELECTOR, "a.btn.btn-secondary.btn-
back")
        options.click()
        time.sleep(1)
        select=driver.find_element(By.CSS_SELECTOR, "a.logo")
        select.click()
        time.sleep(3)
        options=driver.find_element(By.CSS_SELECTOR,
"a[href='/user_profile_view']")
        options.click()
        time.sleep(2)
        driver.execute_script("window.scrollBy(0, 600);")
        time.sleep(2)


if __name__ == '__main__':
    import unittest
    unittest.main()
```

Screenshot

```
(jo) C:\Users\joyel\OneDrive\Desktop\django\microproject>python manage.py test
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).

DevTools listening on ws://127.0.0.1:63877/devtools/browser/71e3c35b-6ad5-4267-aeba-a93db14e5ed7
.
----------------------------------------------------------------------
Ran 1 test in 66.036s

OK
Destroying test database for alias 'default'...

(jo) C:\Users\joyel\OneDrive\Desktop\django\microproject>
```

Test Report

Test Case 1

Project Name: EcoRecover					
Customer Test Case					
Test Case ID: 1			**Test Designed By:** Joyel Joy		
Test Priority (Low/Medium/High): High			**Test Designed Date: 25/09/2023**		
Module Name: Login Screen			**Test Executed By:** Mr. Jinson Devis		
Test Title: Customer Login			**Test Execution Date: 27/09/2023**		
Description: Verify login with valid email and Booking of Events Related to Waste Management					
Pre-Condition: Customer has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigation to Login Page		Dashboard should be displayed	Login page displayed	Pass
2	Provide Valid username	Username: Awin	Customer should be able to login	Customer successfully login to the home page	Pass
3	Provide Valid Password	Password: 12045#CEah			
4	Click on Login button				
5	Click on Campaigns and Seminar		Campaigns and Seminar page will be displayed	Customer navigated to the page and Book Sucessfully	Pass
6	Click to logout		Customer want to be logout	Customer successfully logout	Pass
Post-Condition: Customer Successfully Login and Navigate to Campaigns and Seminar Page then Book a Seminar and Logout					

Test Case 2:

Code

```
from django.test import TestCase
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
class Hosttest(TestCase):

    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.implicitly_wait(10)
        self.live_server_url = 'http://127.0.0.1:8000/'

    def tearDown(self):
        self.driver.quit()



    def test_02_login_page(self):
        driver = self.driver
        driver.get(self.live_server_url)
        driver.maximize_window()
        time.sleep(2)
        theme=driver.find_element(By.CSS_SELECTOR,"a[href='/loginn']")
        theme.click()
        time.sleep(4)
        elem = driver.find_element(By.NAME, "username")
        elem.send_keys("admin1")
        elem = driver.find_element(By.NAME, "password")
        elem.send_keys("12045#CEah")
        submit_button = driver.find_element(By.CSS_SELECTOR, "#btn_login")
        submit_button.click()
        time.sleep(5)
        driver.execute_script("window.scrollBy(0, 2900);")
        time.sleep(5)
        browse=driver.find_element(By.CSS_SELECTOR,"span.hide-menu")
        browse.click()
        time.sleep(5)
        driver.execute_script("window.scrollBy(0, 2900);")
        time.sleep(5)
```

```
        view=driver.find_element(By.CSS_SELECTOR,"select#role-filter.filter-
select")
        view.click()
        time.sleep(1)
        driver.execute_script("window.scrollBy(0, 2900);")
        time.sleep(4)
        view=driver.find_element(By.CSS_SELECTOR,"select#role-filter.filter-select
option[value='admin']")
        view.click()
        chap = driver.find_element(By.CSS_SELECTOR, "span.hide-menu")
        chap.click()
        time.sleep(1)
        options = driver.find_element(By.CSS_SELECTOR,
"img[src='/static/images/profile/flat-business-man-user-profile-avatar-in-suit-
vector-4333496.jpg']")
        options.click()
        time.sleep(3)
        options = driver.find_element(By.CSS_SELECTOR, "a[href='/loggout']")
        options.click()
        time.sleep(1)



if __name__ == '__main__':
    import unittest
    unittest.main()
```

Screenshot

```
(jo) C:\Users\joyel\OneDrive\Desktop\django\microproject>python manage.py test
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).

DevTools listening on ws://127.0.0.1:63877/devtools/browser/71e3c35b-6ad5-4267-aeba-a93db14e5ed7
.
----------------------------------------------------------------------
Ran 1 test in 66.036s

OK
Destroying test database for alias 'default'...

(jo) C:\Users\joyel\OneDrive\Desktop\django\microproject>
```

Test report

Test Case 2

Project Name: EcoRecover					
Admininstrator Test Case					
Test Case ID: 2			**Test Designed By:** Joyel Joy		
Test Priority (Low/Medium/High): High			**Test Designed Date: 25/09/2023**		
Module Name: Admin			**Test Executed By:** Mr. Jinson Devis		
Test Title: Admin Dashbord			**Test Execution Date: 27/09/2023**		
Description: Verify login, view the Dashbord, navigate to the user details ,filter the admin user , navigate to the feedback page , logout					
Pre-Condition:Admin has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigation to Login Page		Dashboard should be displayed	Login page displayed	Pass
2	Provide Valid username	Username: admin1	Admin should be able to login	Admin successfully login to the home page	Pass
3	Provide Valid Password	Password: 12045#CEah			
4	Click on Login button				
5	Click on User Details		List of all Users Displayed	Tuor navigated to UserDetails	Pass
6	Click to Filter				
7	Click to Admin				
8	Click Feedback		List of Feedback will display	Feedback of all users Displayed	Pass
8	Click to logout		Tutor wants to be logout	Tutor successfully logout	Pass
Post-Condition: Admin logined successfully and navigated to user details then filter the admin users and then navigate to the feedback session and logout					

Test Case 3:

Code

```
from django.test import TestCase
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
class Hosttest(TestCase):

    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.implicitly_wait(10)
        self.live_server_url = 'http://127.0.0.1:8000/'

    def tearDown(self):
        self.driver.quit()



    def test_02_login_page(self):
        driver = self.driver
        driver.get(self.live_server_url)
        driver.maximize_window()
        time.sleep(2)
        theme=driver.find_element(By.CSS_SELECTOR,"a[href='/loginn']")
        theme.click()
        time.sleep(4)
        elem = driver.find_element(By.NAME, "username")
        elem.send_keys("Staff1")
        elem = driver.find_element(By.NAME, "password")
        elem.send_keys("12045#CEah")
        submit_button = driver.find_element(By.CSS_SELECTOR, "#btn_login")
        submit_button.click()
        time.sleep(2)
        driver.execute_script("window.scrollBy(0, 2900);")
        time.sleep(2)
        browse=driver.find_element(By.CSS_SELECTOR,"a.sidebar-
link[href='/event_details_view/'] span.hide-menu")
        browse.click()
        time.sleep(2)
        driver.execute_script("window.scrollBy(0, 2900);")
        time.sleep(2)
        view=driver.find_element(By.CSS_SELECTOR,"a[href='/delete_event/h1'].btn-
delete")
        view.click()
```

```
        time.sleep(1)
        options = driver.find_element(By.CSS_SELECTOR,
"img[src='/static/images/profile/flat-business-man-user-profile-avatar-in-suit-
vector-4333496.jpg']")
        options.click()
        time.sleep(3)
        options = driver.find_element(By.CSS_SELECTOR, "a[href='/loggout']")
        options.click()
        time.sleep(1)



if __name__ == '__main__':
    import unittest
    unittest.main()
```

Screenshot

```
(jo) C:\Users\joyel\OneDrive\Desktop\django\microproject>python manage.py test
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).

DevTools listening on ws://127.0.0.1:63877/devtools/browser/71e3c35b-6ad5-4267-aeba-a93db14e5ed7
.
----------------------------------------------------------------------
Ran 1 test in 66.036s

OK
Destroying test database for alias 'default'...

(jo) C:\Users\joyel\OneDrive\Desktop\django\microproject>
```

Test report

Test Case 3

Project Name: EcoRecover	
Event Manger Test case	
Test Case ID: 3	**Test Designed By:** Joyel Joy
Test Priority (Low/Medium/High): High	**Test Designed Date: 25/09/2023**
Module Name: Event Manager Dashboard	**Test Executed By:** Mr. Jinson Devis

Test Title: Event Manager Dashbord			**Test Execution Date: 27/09/2023**		
Description: Event Manger Login , naviaget to event details , delete a event ,logout					
Pre-Condition: Event Manger has valid username and password, after the login want to edit the profile					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigation to Login Page		Dashboard should be displayed	Login page displayed	Pass
2	Provide Valid username	Username: Staff1	Event Manger should be able to login	Event Manger successfully login to the dashboard page	Pass
3	Provide Valid Password	Password: 12045#CEah			
4	Click on Login button				
5	Click on Event Details		Admin can edit his profile	Admin edited the profile and save	Pass
6	Delete a Event		Event Deletion	Delete a event	Pass
7	Click to logout		Event Manger want to be logout	Event Manger successfully logout	Pass
Post-Condition: Event Manger successfully login and navigate to the Event Details page and view all the Events then Delete one Event and successfully logout					

CHAPTER 6

IMPLEMENTATION

6.1 INTRODUCTION

The implementation phase of a project is where the design is transformed into a functional system. It is a crucial stage in ensuring the success of the new system, as it requires gaining user confidence that the system will work effectively and accurately. User training and documentation are key concerns during this phase. Conversion may occur concurrently with user training or at a later stage. Implementation involves the conversion of a newly revised system design into an operational system.

During this stage, the user department bears the primary workload, experiences the most significant upheaval, and has the most substantial impact on the existing system. Poorly planned or controlled implementation can cause confusion and chaos. Whether the new system is entirely new, replaces an existing manual or automated system, or modifies an existing system, proper implementation is essential to meet the organization's needs. System implementation involves all activities required to convert from the old to the new system. The system can only be implemented after thorough testing is done and found to be working according to specifications. System personnel evaluate the feasibility of the system. Implementation requires extensive effort in three main areas: education and training, system testing, and changeover. The implementation phase involves careful planning, investigating system and constraints, and designing methods to achieve changeover.

6.2 IMPLEMENTATION PROCEDURES

Software implementation is the process of installing the software in its actual environment and ensuring that it satisfies the intended use and operates as expected. In some organizations, the software development project may be commissioned by someone who will not be using the software themselves. During the initial stages, there may be doubts about the software, but it's important to ensure that resistance does not build up.

This can be achieved by:

- Ensuring that active users are aware of the benefits of the new system, building their confidence in the software.
- Providing proper guidance to the users so that they are comfortable using the application. Before viewing the system, users should know that the server program must be running on the server. Without the server object up and running, the intended process will not take place.

6.2.1 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.

6.2.2 Training on the Application Software

After providing the necessary basic training on computer awareness, it is essential to provide training on the new application software to the user. This training should include the underlying philosophy of using the new system, such as the flow of screens, screen design, the type of help available on the screen, the types of errors that may occur while entering data, and the corresponding validation checks for each entry, and ways to correct the data entered. Additionally, the training should cover information specific to the user or group, which is necessary to use the system or part of the system effectively. It is important to note that this training may differ across different user groups and levels of hierarchy.

6.2.3 System Maintenance

The maintenance phase is a crucial aspect of the software development cycle, as it is the time when the software is actually use and performs its intended functions. Proper maintenance is essential to ensure that the system remains functional, reliable, and adaptable to changes in the system environment. Maintenance activities go beyond simply identifying and fixing errors or bugs in the system. It may involve updates to the software, modifications to its functionalities, and enhancements to its performance, among other things. In essence, software maintenance is an ongoing process that requires continuous monitoring, evaluation, and improvement of the system to meet changing user needs and requirements.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

EcoRecover is an innovative and forward-looking platform dedicated to addressing the critical challenges of waste management while promoting sustainability through the sale of recycled products. The project envisions a circular economy, focusing on efficient waste disposal and the adoption of eco-friendly products. At its core, EcoRecover leverages a stack of technologies to create a user-friendly, responsive, and data-efficient platform. This integration seamlessly brings together waste generators, collectors, and eco-conscious consumers. For waste generators, the platform offers user-friendly functionalities. Users can efficiently create profiles, book waste bins, reschedule collections, make secure payments, track their orders, and even participate in eco-friendly campaigns. EcoRecover aims to make waste management an engaging and accessible process for all.

Waste collectors benefit from comprehensive course management. They can efficiently manage waste collection orders, ensuring timely and effective waste disposal. The system's robust database management ensures the seamless storage and retrieval of critical information, contributing to an efficient operation. Administrators play a pivotal role in ensuring the smooth operation of the platform. They can manage user profiles, authorize waste disposal processes, and oversee event management, further contributing to a well-organized and sustainable waste management ecosystem.

In conclusion, EcoRecover is not just another waste management platform. It represents a dynamic ecosystem driven by cutting-edge technologies, intelligent database design, and user-friendly interfaces. It empowers waste generators to responsibly manage their waste, waste collectors to efficiently carry out their tasks, and administrators to maintain a well-organized waste management environment. EcoRecover's holistic approach promotes responsible waste disposal, recycling, and the adoption of eco-friendly products, ultimately contributing to a cleaner and more sustainable environment.

7.2 FUTURE SCOPE

The future scope of EcoRecover holds substantial promise, offering numerous avenues for growth and influence in the field of environmental sustainability. As the platform demonstrates its effectiveness in waste management and the promotion of recycled products, there are several key areas to consider for future development and expansion.

One significant aspect of the platform's future lies in its scalability. EcoRecover can broaden its reach by expanding to cover a wider geographical area, serving more waste generators, collectors, and environmentally conscious consumers. This expansion may involve establishing partnerships with local waste management organizations, recycling facilities, and manufacturers of eco-friendly products, creating a more extensive network of sustainable practices.

Furthermore, the development of dedicated mobile applications for EcoRecover presents an opportunity to enhance accessibility and user-friendliness. Mobile apps can offer features like real-time order tracking, push notifications, and convenient payment options, ultimately improving the overall user experience and engagement. In the realm of data, advanced analytics and machine learning can play a pivotal role. Implementing these technologies allows EcoRecover to gain valuable insights into waste management trends and user behaviors. These insights can be used to optimize collection schedules, predict waste generation patterns, and provide personalized recommendations to users, making the platform more responsive to individual needs.

IoT (Internet of Things) technology opens the door to the development of smart waste bins. These bins, equipped with sensors, can automatically detect waste levels, reducing the need for manual scheduling and enhancing collection efficiency. Such innovation aligns with EcoRecover 's mission to stay at the forefront of sustainability practices.

Collaborations and partnerships are another essential aspect of Eco Recover’s future scope. By working closely with local businesses, municipalities, and recycling centers, the platform can create a network of sustainability partners. These partnerships can lead to improved waste disposal infrastructure, increased product offerings, and potential incentives for users, further motivating them to engage in eco-friendly practices.

In conclusion, the future of EcoRecover is bright, with ample opportunities for growth and impact. By embracing technology, fostering partnerships, and continually engaging users in sustainable practices, EcoRecover is well-positioned to contribute to a cleaner, greener, and more sustainable future on a local and potentially global scale.

CHAPTER 8
BIBLIOGRAPHY

REFERENCES:

- https://www.geeksforgeeks.org/handling-ajax-request-in-django/
- https://www.pluralsight.com/guides/work-with-ajax-django
- https://www.tangowithdjango.com/book17/chapters/ajax.html
- https://realpython.com/django-and-ajax-form-submissions/

WEBSITES:

- https://www.wm.com/
- www.w3schools.com
- https://www.recycling.com/
- https://chat.openai.com/chat
- www.jquery.com

CHAPTER 9
APPENDIX

9.1 Sample Code

Login

```
{% load static %}
{% load socialaccount %}
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login Page</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      display: flex;
      justify-content: center;
      align-items: center;
      min-height: 100vh;
      background: linear-gradient(to bottom right, #ABCDEF, #E5F2FF);
    }

    .login-container {
      background-color: rgba(255, 255, 255, 0.8);
      border-radius: 10px;
      padding: 20px;
      box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);
      width: 350px;
      text-align: center;
    }

    .login-container h1 {
      margin-bottom: 20px;
      color: #333;
    }

    .form-group {
      margin-bottom: 20px;
      text-align: left;
    }

    label {
      display: block;
      margin-bottom: 5px;
      color: #666;
    }
```

```
    input[type="text"],
    input[type="password"] {
      width: 100%;
      padding: 9px;
      border: 1px solid #ccc;
      border-radius: 4px;
      font-size: 13px;
      background-color: #f7f7f7;
    }

    button {
      background-color: #fb641b;
      color: #fff;
      padding: 10px 20px;
      border: none;
      border-radius: 4px;
      font-size: 14px;
      cursor: pointer;
      width: 100%;
      transition: background-color 0.3s ease;
    }

    button:hover {
      background-color: #f78239;
    }

    a {
      color: #fb641b;
      text-decoration: none;
    }
    .forgot-password {
      font-size: 12px;
      margin-top: 5px;
  }
  .logo {
    font-family: Arial, sans-serif;
    font-size: 2rem;
    text-decoration: none;
    color: black;
    display: inline-block;
    transition: color 0.3s, transform 0.3s;
}

.logo em {
    color: grey;
}
```

```
.logo:hover {
    color: #2ecc71;
    transform: scale(1.1);
}
body {
  font-family: Assistant, sans-serif;
  margin: 0;
  padding: 0;
  background: linear-gradient(to bottom right, #ABCDEF, #E5F2FF);
  background-image: url('{% static "images/orginal/13.jpg" %}'); /* Replace with
your image URL */
  background-size: cover;
}
.login-container {
  display: flex;
  flex-direction: column;
  align-items: center;
  padding: 30px;
  border-radius: 10px;
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);
  width: 350px;
  margin: auto;
  text-align: center;
  background-color: rgba(255, 255, 255, 0.8);
}
.form-group {
  margin-bottom: 15px;
}
input[type="text"],
input[type="password"] {
  width: 100%;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 3px;
}
button {
  background-color: #007bff;
  color: #fff;
  border: none;
  padding: 10px 20px;
  border-radius: 3px;
  cursor: pointer;
}
a {
  color: #27350F;
  text-decoration: none;
}
  </style>
```

```
</head>
<body>
  <div class="login-container">
    <a href="{% url 'index' %}" class="logo">Eco<em>Recover</em></a>
    <form method="POST">
      {% csrf_token %}
      <div class="form-group">
        <input type="text" id="username" name="username" placeholder="Username">
      </div>
      <div class="form-group">

        <input type="password" id="password" name="password"
placeholder="Password">

      <div class="form-group">
        <a href="{% url 'password_reset' %}" style="color: #27350F; text-
decoration: none">Forgot password?</a>
      </div>

      </div>
      <button id="btn_login" onclick="validateLoginForm()">Sign In</button>
      <p>Don't have an account? <a href="{% url 'register' %}">Register</a></p>
      <a href="{% provider_login_url 'google'%}?next=/">Login with Google</a>
      {% for message in messages %}

                <p> </P> {{ message }}
              {% endfor %}
    </form>

  </div>
</body>
<script>
  function validateLoginForm() {
      var username = document.getElementById("username").value;
      var password = document.getElementById("password").value;

      var usernameError = document.getElementById("email-error");
      var passwordError = document.getElementById("password-error");

      var isValid = true;

      if (username.trim() === "") {
          usernameError.textContent = "Please enter a username.";
          isValid = false;
      } else {
          usernameError.textContent = "";
      }
```

```
        if (password.trim() === "") {
            passwordError.textContent = "Please enter a password.";
            isValid = false;
        } else {
            passwordError.textContent = "";
        }

        if (isValid) {
            // Call the function to attempt login
            attemptLogin(username, password);
        }
    }

    function attemptLogin(username, password) {
        // Include the CSRF token in the headers of the AJAX request
        var csrftoken = document.querySelector('[name=csrfmiddlewaretoken]').value;

        fetch('/login/', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
                'X-CSRFToken': csrftoken, // Include the CSRF token here
            },
            body: JSON.stringify({ username: username, password: password }),
        })
        .then(response => response.json())
        .then(data => {
            if (data.success) {
                window.location.href = '/'; // Redirect to the main page on
successful login
            } else {
                alert("Invalid Login"); // Show an alert on unsuccessful login
            }
        })
        .catch(error => {
            console.error('Error:', error);
        });
    }
    </script>
{% if error_message %}
    <script>
        alert("{{ error_message }}");
    </script>
{% endif %}
</html>

from django.contrib.auth import authenticate, login
```

```
from django.shortcuts import render, redirect
from django.contrib import messages

def loginn(request):
    if request.user.is_authenticated:
        if request.user.is_staff or request.user.is_superuser:
            return redirect('hello_admin')
        else:
            return redirect('index')

    if request.method == "POST":
        email = request.POST['username']
        password = request.POST['password']
        user = authenticate(username=email, password=password)

        if user is not None:
            login(request, user)
            if user.is_staff or user.is_superuser:
                return redirect('booking_chart')
            else:
                return redirect('index')
        else:
            messages.error(request, "Invalid Login")
            return render(request, 'Login.html')
    else:
        return render(request, 'login.html')
```

User Profile

```
{% load static %}
<html lang="en">
    <div class="messages">
        {% if messages %}
            <ul class="messages">
                {% for message in messages %}
                    {% if message.tags == 'success edit-profile' %}
                        <li class="{{ message.tags }}">{{ message }}</li>
                    {% endif %}
                {% endfor %}
            </ul>
        {% endif %}
    </div
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>User Profile</title>
```

```
    <link rel="stylesheet" href="{% static 'path-to-your-css-file.css' %}">
    <div class="user-options">
        <ul class="user-menu">
            <li><a href="{% url 'edit_profile' %}">Edit Profile</a></li>
            <li><a href="{% url 'reset_password' %}">Reset Password</a></li>
            <li><a href="{% url 'loggout' %}">Logout</a></li>
        </ul>
    </div>



</head>

<body>
    <div class="container">
        <div class="profile">
            <a href="{% url 'index' %}" class="logo">Eco<em>Recover</em></a>
            <div class="user-image">
                <img src="{% static 'images/profile/flat-business-man-user-profile-
avatar-in-suit-vector-4333496.jpg' %}" alt="User Profile Picture">
            </div>
            <h1>{{ profile.user.username }}</h1>
        </div>



        <div class="section address">
            <h2 class="section-title">Address</h2>
            <p class="user-info">{{ profile.address }}</p>
        </div>

        <div class="section contact-info">
            <h2 class="section-title">Contact Information</h2>
            <p class="user-info">Email: {{ profile.user.email }}</p>
            <p class="user-info">Phone: {{ user.userprofile.mobile_number }}</p>

        </div>

        <div class="section subscription">
            <h2 class="section-title">Subscription Details</h2>
            <p class="user-info">Subscribed: {% if profile.subscribed %} Yes {%
else %} No {% endif %}</p>
            {% if profile.subscribed %}
            <p class="user-info">Subscription Expiration: {{
profile.subscription_expiration }}</p>
            <p class="user-info">Subscription Duration: {{
profile.subscription_duration }} months</p>
            {% endif %}
```

```
            <p class="user-info">Date Joined: {{ user.date_joined }}</p>
            <p class="user-info">Last Login: {{ user.last_login }}</p>
        </div>
    </div>
    <div class="messages">



    </div>
</div></div></div>
<style>
    from django.contrib.auth.decorators import login_required
from django.shortcuts import render
from .models import UserProfile

@login_required(login_url='loginn')
def user_profile_view(request):
    profile = UserProfile.objects.get(user=request.user)

    context = {
        'profile': profile,
    }
    return render(request, 'profile/profile.html', context)
```

Campaigns and Seminar Booking page

```
<div class="pre-header">
        <div class="container">
            <div class="row">
                <div class="col-lg-6 col-sm-6">
                    <div class="text-button">

                    </div>
                </div>
            </div>
        </div>
    </div>

    <!-- ***** Header Area Start ***** -->
    <header class="header-area header-sticky">
        <div class="container">
            <div class="row">
                <div class="col-12">
                    <nav class="main-nav">
                        <!-- ***** Logo Start ***** -->
```

```
                        <a href="{% url 'index' %}"
class="logo">Eco<em>Recover</em></a>


                        <!-- ***** Logo End ***** -->
                        <!-- ***** Menu Start ***** -->
                        <ul class="nav">
                            <li><a href="{% url 'index' %}">Home</a></li>
                            <li><a href="about.html">About Us</a></li>
                            <li><a href="rent-venue.html">Contact Us</a></li>
                            <li><a href="{% url 'show_events' %}"
class="active">Campigns and Events</a></li>

                        </ul>
                        <a class='menu-trigger'>
                            <span>Menu</span>
                        </a>
                        <!-- ***** Menu End ***** -->
                    </nav>
                </div>
            </div>
        </div>
    </header>
    <!-- ***** Header Area End ***** -->

    <!-- ***** About Us Page ***** -->
    <div class="page-heading-shows-events">
        <div class="container">
            <div class="row">
                <div class="col-lg-12">
                    <h2>Our Campigns & Events</h2>
                    <span>Check out upcoming  Campigns & events.</span>
                </div>
            </div>
        </div>
    </div>

    <div class="shows-events-tabs">
        <div class="container">
            <div class="row">
                <div class="col-lg-12">
                    <div class="row" id="tabs">
                        <div class="col-lg-12">
                            <div class="heading-tabs">
                                <div class="row">
                                    <div class="col-lg-8">
                                        <ul>
                                            <li>
```

```
                                                <a class="tab-link" href="#tabs-1"
style="color: #FF5733; font-weight: bold;">
                                                    <span class="tab-icon"><i
class="fa fa-calendar"></i></span>
                                                    Upcoming Events
                                                </a>
                                            </li>
                                        </ul>
                                    </div>
                                </div>
                            </div>
                        </div>

                        <!-- Add links in the same row -->


                        <style>
                             /* .category-links {
                                display: flex;
                                justify-content: center;
                                align-items: center;
                                margin-top: 20px;
                            }  */

                            .category-link {
                                text-decoration: none;
                                color: #000; /* Set the text color to black */
                                font-weight: bold; /* Make the text bold */
                                margin: 0 10px; /* Adjust the margin here for spacing
*/
                            }
                            .para1
                            {
                                text-align:justify;
                            }
                        </style>


                        <div class="row mt-3 category-links">
                            <div class="col-lg-8">
                                <!-- Add links to filter events by category -->
                                <ul class="list-inline">
                                    <li class="list-inline-item">
                                        <a href="{% url 'show_events' %}"
class="category-link">All Events</a>
                                    </li>
                                    <li class="list-inline-item">
```

```
                                    <a href="{% url 'show_events_by_category'
'category5' %}" class="category-link">Hackathon</a>
                                </li>
                                <li class="list-inline-item">
                                    <a href="{% url 'show_events_by_category'
'category1' %}" class="category-link">Campaign</a>
                                </li>
                                <li class="list-inline-item">
                                    <a href="{% url 'show_events_by_category'
'category2' %}" class="category-link">Seminar</a>
                                </li>
                                <li class="list-inline-item">
                                    <a href="{% url 'show_events_by_category'
'category3' %}" class="category-link">Expo</a>
                                </li>
                                <li class="list-inline-item">
                                    <a href="{% url 'show_events_by_category'
'category4' %}" class="category-link">Workshop</a>
                                </li>

                                <!-- Add more category links as needed -->
                            </ul>
                        </div>
                    </div>









                    </div>
                </div>
            </div>
        </div>
    </div>



    <div class="row">
        {% if no_events_message %}
        <div class="col-lg-12">
            <p>{{ no_events_message }}</p>
        </div>
        {% endif %}

        <div class="col-lg-12">
            <section class="tabs-content">
```

```
                <article id="tabs-1">
                    <div class="row">
                        <div class="col-lg-9">
                            <div class="row">
                                {% for event in events %}
                                <div class="col-lg-12">
                                    <div class="event-item">
                                        <div class="row">
                                            <div class="col-lg-4">
                                                <div class="left-content">
                                                    <h4>{{ event.name }}</h4>
                                                    <p class="para1">{{
event.description }}</p>
                                                </div>
                                            </div>
                                            <div class="col-lg-4">
                                                <div class="thumb">
                                                    <img src="{{ event.image.url
}}" alt="{{ event.name }}">
                                                </div>
                                            </div>
                                            <div class="col-lg-4">
                                                <div class="right-content">
                                                    <ul class="list-unstyled event-
details">
                                                        <li>
                                                            <i class="fa "></i>
                                                            <h6>Event Date: {{
event.date|date:"M d D" }}</h6>
                                                            <h6>Event Time:  {{
event.time|time:"g:i A" }}</h6>
                                                            <h6>Event Location:  {{
event.location }}</h6>
                                                        </li>
                                                        <li>
                                                            <i class="fa "></i>
                                                            <br> <!-- Add a line
break here -->
                                                        </li>
                                                        <li>
                                                            <br>
                                                            <br>
                                                            <br>
                                                            <br> <!-- Add a line
break here -->
                                                            {% if event.category ==
'category2' or event.category == 'category4' %}
```

```
                                                        <div class="main-dark-
button">
                                                            <a href="{% url
'event_booking' event.event_id %}">BOOK NOW</a>
                                                        </div>
                                                    {% endif %}
                                                    </li>
                                                </ul>
                                            </div>
                                        </div>
from django.shortcuts import render
from .models import Event  # Import your Event model

from django.shortcuts import render
from .models import Event

def show_events(request, category=None):
    if category:
        # Filter events by category if a category is provided
        events = Event.objects.filter(category=category)
    else:
        # If no category is provided, get all events
        events = Event.objects.all()

    context = {
        'events': events,
    }

    if not events:
        context['no_events_message'] = "No upcoming events at the moment."

    return render(request, 'event/showsevents.html', context)
```

Bin Booking for Home

```
<div class="tickets-page">
    <div class="container">
        <div class="row">
            <div class="col-lg-12">
                <div class="section-heading">
                    <h2>For Home</h2>
                </div>
            </div>

            {% for bin in bins %}
            <div class="col-lg-4">
```

```
                <div class="ticket-item">
                    <div class="thumb">
                        <img src="{{ bin.image.url }}" alt="{{ bin.title }}"
class="bin-image">
                    </div>
                    <div class="down-content">
                        <h4>{{ bin.title }}</h4>
                        <p class="availability">Available: {{ bin.size }} size</p>
                        <ul class="bin-details">
                            <li><i class="fa "></i> Capacity: {{ bin.capacity
}}</li>
                            <li><i class="fa "></i> {{ bin.description }}</li>
                        </ul>
                        <div class="order-button">
                            <a href="{% url 'orderforhome' %}">Order Bin</a>
                        </div>
                    </div>
                </div>
            </div>
            {% endfor %}

        </div>
    </div>
</div>
@login_required(login_url='loginn')
def bin_order(request):
    bins = Bin.objects.all()
    context = {'bins': bins}
    return render(request, 'bin/binorder.html', context)
```

Bin Booking for Events

```
  <div class="tickets-page">
        <div class="container">
            <div class="row">
                <div class="col-lg-12">
                    <div class="heading">
                        <h2>For Event</h2>
                    </div>
                </div>
                <style>
                    .ticket-item {
                        transition: transform 0.2s ease-in-out;
                    }

                    .ticket-item:hover {
```

```
                        transform: scale(1.05); /* Increase the size of the item on
hover */
                        box-shadow: 0 0 10px rgba(0, 0, 0, 0.2); /* Add a subtle
shadow on hover */
                    }
                </style>
                  {# Use your base template or change this as needed #}
{% block content %}
    {% for bin in binss %}
        <div class="col-lg-4">
            <div class="ticket-item">
                <div class="thumb">
                    <img src="{{ bin.image.url }}" alt="{{ bin.title }}">
                    <div class="price">
                        <span>1 bin<br>{{ bin.capacity }} capacity</span>
                    </div>
                </div>
                <div class="down-content">
                    <span>Available: {{ bin.size }} size</span>
                    <h4>{{ bin.title }}</h4>
                    <ul>
                        <li><i class="fa fa-clock-o"></i> Bin ID: {{ bin.bin_id
}}</li>
                        <li><i class="fa fa-map-marker"></i>{{ bin.description
}}</li>
                    </ul>
                    <div class="main-dark-button">
                        <a href="{% url 'save_bin_booking_event' %}">Order Bin</a>
                    </div>
                </div>
            </div>
        </div>
    {% endfor %}
{% endblock %}


            </div>
        </div>

    </div>


            </div>
        </div>
    </div>
@login_required(login_url='loginn')
def bin_order_event(request):
    binss = BinEvent.objects.all()
```

```
    context = {'binss': binss}
    return render(request, 'bin/binorderevent.html', context)
```

Bin Order For Event form

```
<body>
    <h1>Bin Booking Event Form</h1>

    <form method="post" action="{% url 'save_bin_booking_event' %}"
onsubmit="return submitForm(event)">
        {% csrf_token %}

        <label for="binDetails">Select Bin Size and Capacity:</label>
        <select name="bin" id="binSelect" class="custom-select"
onchange="updateAmount()">
            <option value="" disabled selected>Select Bin Size and
Capacity</option>
            {% for bin in bins %}
            <option value="{{ bin.bin_id }}" data-bin-price="{{ bin.price }}">{{
bin.size }}</option>
            {% endfor %}
        </select>

        <!-- Error message placeholder -->
        <p id="binError" style="color: red;"></p>

        <label for="event_date_time" style="font-weight: bold; font-size: 18px;
color: #333;">Event Date:</label>
        <input type="date" name="event_date_time" id="eventDateTime"
onblur="validateEventDateTime()" style="padding: 8px; border: 1px solid #ccc;
border-radius: 5px; font-size: 16px;">
        <p id="eventDateTimeError" style="color: red;"></p>

        <label for="event_location">Event Location:</label>
        <input type="text" name="event_location" id="eventLocation"
onblur="validateEventLocation()">
        <p id="eventLocationError" style="color: red;"></p>

        <label for="delivery_time">Delivery Time:</label>
        <input type="datetime-local" name="delivery_time" id="deliveryTime"
onblur="validateDeliveryTime()">
        <p id="deliveryTimeError" style="color: red;"></p>

        <label for="pickup_time">Pickup Time:</label>
        <input type="datetime-local" name="pickup_time" id="pickupTime"
onblur="validatePickupTime()">
        <p id="pickupTimeError" style="color: red;"></p>

        <label for="number_of_bins_needed">Number of Bins Needed:</label>
```

```
        <input type="number" name="number_of_bins_needed" id="numBins" value="0">
        <p id="numBinsError" style="color: red;"></p>

        <div id="amount-to-pay-section">
            <p id="amount-to-pay" style="font-weight: bold; font-size:
18px;">Amount to Pay: ₹0</p>
        </div>

        <input type="hidden" id="binPrice" name="bin_price" value="0"> <!-- Hidden
field to store the bin price -->

        <button type="submit" id="saveButton">Save</button>
    </form>

    <script>
        // Function to calculate the amount based on the selected bin size and
number of bins
        function updateAmount() {
            var binSelect = document.getElementById("binSelect");
            var numBinsInput = document.getElementById("numBins");
            var amountToPay = document.getElementById("amount-to-pay");
            var binPrice = document.getElementById("binPrice");

            // Get the selected option and its data attributes
            var selectedOption = binSelect.options[binSelect.selectedIndex];
            var binPriceValue = parseFloat(selectedOption.getAttribute("data-bin-
price")); // Parse as a float

            // Update the hidden input field with the bin price
            binPrice.value = binPriceValue;

            // Calculate the amount based on the selected bin size and number of
bins
            var numBins = parseInt(numBinsInput.value);
            var totalAmount = numBins * binPriceValue;

            // Update the amount to pay with proper formatting
            amountToPay.textContent = "Amount to Pay: ₹" + totalAmount.toFixed(2);
// Format to two decimal places
        }

        // Add event listeners to the numBins input field to update the amount
immediately
        var numBinsInput = document.getElementById("numBins");
        numBinsInput.addEventListener("input", updateAmount);

        // Initialize the amount calculation when the page loads
        updateAmount();
```

```
    </script>
</body>
</html>
```

Payment page

```
<form method="POST">
  {% csrf_token %}
<div class='wrapper'>
    <div class='package'>
      <div class='name'>Monthly</div>
      <div class='price'>200 RS </div>
      <div class='trial'>Available for a month</div>
      <hr>
      <ul>
        <li>
          <strong></strong>
          Waste Collection Throughout a Month
        </li>
        <li>
          <strong>One-Time </strong>
          Bulk Waste Collection
        </li>
      </ul>

      {% if user.is_authenticated %}
      {% if user.userprofile.subscribed %}
        <a href="{% url 'index' %}" class="subscribe-button">Subscribed</a>
      {% else %}
        <a href="{% url 'paymentform' %}?amount=200" class="subscribe-
button">Subscribe Now</a>
      {% endif %}
      {% else %}
        <a href="{% url 'loginn' %}" class="subscribe-button">Subscribe Now</a>
      {% endif %}       </ul>
    </div>
    <div class='package brilliant'>
      <div class='name'>6 Months</div>
      <div class='price'>400 RS</div>
      <div class='trial'>Free 30 day trial</div>
      <hr>
      <ul>
        <li>
          <strong></strong>
          Waste Collection Throughout 6 Months
        </li>
```

```
        <li>
          <strong>Three-Times </strong>
          Bulk Waste Collection
        </li>
      </ul>
      {% if user.is_authenticated %}
  {% if user.userprofile.subscribed %}
    <a href="{% url 'index' %}" class="subscribe-button">Subscribed</a>
  {% else %}
    <a href="{% url 'paymentform' %}?amount=400" class="subscribe-button">Subscribe
Now</a>
  {% endif %}
  {% else %}
    <a href="{% url 'loginn' %}" class="subscribe-button">Subscribe Now</a>
  {% endif %} </div>
    <div class='package'>
      <div class='name'>1 year</div>
      <div class='price'>600 RS</div>
      <div class='trial'>Totally free</div>
      <hr>
      <ul>
        <li>
          <strong></strong>
          Waste Collection Throughout a Year
        </li>
        <li>
          <strong>Ten-Times </strong>
          Bulk Waste Collection
        </li>
      </ul>
      {% if user.is_authenticated %}
  {% if user.userprofile.subscribed %}
    <a href="{% url 'index' %}" class="subscribe-button">Subscribed</a>
  {% else %}
    <a href="{% url 'paymentform' %}?amount=600" class="subscribe-button">Subscribe
Now</a>
  {% endif %}
  {% else %}
    <a href="{% url 'loginn' %}" class="subscribe-button">Subscribe Now</a>
  {% endif %}     </div>
  </div>
</form>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Payment Details</title>
</head>
```

```
<body>
    <h1>Payment Details</h1>
    <p>Amount to Pay: ₹<span id="amount-to-pay">{{ amount_to_pay }}</span></p>
    <!-- Add other payment-related details as needed -->
</body>
<script>
    // Find the "Save" button by its ID
    var saveButton = document.getElementById("saveButton");

    // Add a click event listener to the "Save" button
    saveButton.addEventListener("click", function (event) {
        event.preventDefault(); // Prevent the default form submission behavior

        // Get the form element
        var form = event.target.form;

        // Create a FormData object from the form
        var formData = new FormData(form);

        // Fetch API to send the form data to the server
        fetch(form.action, {
            method: 'POST',
            body: formData,
            headers: {
                'X-CSRFToken': getCookie('csrftoken') // Include the CSRF token in
the request header
            }
        })
        .then(response => response.json())
        .then(data => {
            if (data.success) {
                // Redirect to the payment_details.html page with the amount
                window.location.href = `/payment_details/${data.amount_to_pay}/`;
            } else {
                // Display the error message
                var successMessage = document.getElementById("success-message");
                successMessage.style.color = 'red';
                successMessage.textContent = data.error_message;
            }
        })
        .catch(error => {
            console.error('Error:', error);
        });
    });

    // Function to get the CSRF token from cookies
    function getCookie(name) {
        var cookieValue = null;
```

```
        if (document.cookie && document.cookie !== '') {
            var cookies = document.cookie.split(';');
            for (var i = 0; i < cookies.length; i++) {
                var cookie = cookies[i].trim();
                if (cookie.substring(0, name.length + 1) === (name + '=')) {
                    cookieValue = decodeURIComponent(cookie.substring(name.length +
1));
                    break;
                }
            }
        }
        return cookieValue;
    }
</script>
</html>

from django.conf import settings
from django.views.decorators.csrf import csrf_exempt
from django.shortcuts import render
from django.http import HttpRequest, HttpResponse, HttpResponseBadRequest
import razorpay

razorpay_client = razorpay.Client(auth=(settings.RAZOR_KEY_ID,
settings.RAZOR_KEY_SECRET))

def paymentform(request: HttpRequest):
    currency = 'INR'
    amount = int(request.GET.get("amount")) * 100  # Rs. 200

    razorpay_order = razorpay_client.order.create(dict(amount=amount,
currency=currency, payment_capture='0'))
    razorpay_order_id = razorpay_order['id']
    callback_url = '/paymenthandler/'
    context = {}
    context['razorpay_order_id'] = razorpay_order_id
    context['razorpay_merchant_key'] = settings.RAZOR_KEY_ID
    context['razorpay_amount'] = amount / 100
    context['currency'] = currency
    context['callback_url'] = callback_url

    return render(request, 'payment/payment_form.html', context=context)

from django.conf import settings
@csrf_exempt
@login_required
def paymenthandler(request):
    if request.method == "POST":
        try:
```

```
            payment_id = request.POST.get('razorpay_payment_id', '')
            razorpay_order_id = request.POST.get('razorpay_order_id', '')
            signature = request.POST.get('razorpay_signature', '')
            params_dict = {
                'razorpay_order_id': razorpay_order_id,
                'razorpay_payment_id': payment_id,
                'razorpay_signature': signature
            }
            client = razorpay.Client(auth=('rzp_test_Ul5agYuKOPazq3',
'lhozVEM1VKK2RmUmLIJg0i9C'))
            payment = client.payment.fetch(payment_id)
            payment_amount = payment['amount']
            result = razorpay_client.utility.verify_payment_signature(params_dict)

            if result is not None:
                authenticated_user = request.user
                user_profile = UserProfile.objects.get(user=authenticated_user)


                amount = 200 if payment_amount == 20000 else (400 if payment_amount
== 40000 else 60000)

                # Set the subscription duration based on the plan
                if amount == 200:
                    user_profile.subscription_duration = 1  # 1 month
                elif amount == 400:
                    user_profile.subscription_duration = 6  # 6 months
                elif amount == 60000:
                    user_profile.subscription_duration = 12  # 12 months



                # Calculate the subscription expiration date
                current_date = datetime.now().date()
                expiration_date = current_date + timedelta(days=30 *
user_profile.subscription_duration)  # Assuming 30 days per month
                user_profile.subscription_expiration = expiration_date

                # Set the user as subscribed
                user_profile.subscribed = True
                user_profile.save()

                username = authenticated_user.username

                # Your code for sending a successful subscription email


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

```
                return render(request, 'home.html')
        except Exception as e:
            return render(request, 'home.html', {'error_message': str(e)})
    else:
        return render(request, 'paymentsuccess.html')
```

Event Details

```
<body>
    <div class="container">
        <div class="event-section">
            <h2>Event Details</h2>
            <br>
            <ul class="event-list">
                {% for event in events %}
                    <li class="event-item">
                        <span class="event-number">{{ forloop.counter }}</span>
                        <img src="{{ event.image.url }}" alt="{{ event.name }}
Image">
                        <div class="event-details">
                            <h2>{{ event.name }}</h2>
                            <br>
                            <p class ="t1">{{ event.description }}</p>
                            <p><strong>Date:</strong> {{ event.date }}</p>
                            <p><strong>Time:</strong> {{ event.time }}</p>
                            <p><strong>Location:</strong> {{ event.location }}</p>
                        </div>
                          
                        <div class="event-actions">
                            <a href="{% url 'delete_event' event.event_id %}"
class="btn btn-delete" style="margin-left: 10px;">Delete</a>


                            <!-- Add margin to the "Edit" button to create space --
>
                            <a href="{% url 'edit_event' event.event_id %}"
class="btn btn-edit" style="margin-left: 10px;">Edit</a>
                        </div>

                    </li>
                {% endfor %}
            </ul>

            <div class="nav-links">
                <a class="nav-link" href="{% url 'addevent' %}">Add Events</a>
```

```
            </div>
        </div>
    </div>

    <script>
        function confirmDelete(eventName) {
            const confirmDelete = confirm(`Are you sure you want to delete the
event "${eventName}"?`);
            if (confirmDelete) {
                document.querySelector('.delete-form').submit();
            }
        }
    </script>
```

Admin Dashboard

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css">

<script>
    function toggleMenu() {
        var submenu = document.getElementById('binDetailsSubMenu');
        if (submenu.style.display === 'block') {
            submenu.style.display = 'none';
        } else {
            submenu.style.display = 'block';
        }
    }

</script>




                    <!-- Bin Booking Details - Visible for staff and superuser -->
                    <!-- Bin Booking Details - Visible for staff and superuser -->
                    {% if user.is_staff or user.is_superuser %}
    <li class="sidebar-item">
        <a class="sidebar-link waves-effect waves-dark sidebar-link" href="#"
onclick="toggleSubMenu('binBookingSubMenu')">
            <i class="mdi mdi-alert-outline"></i>
            <span class="hide-menu">Bin Booking Details</span>
        </a>
    </li>
    <div id="binBookingSubMenu" class="sidebar-submenu">
```

```
        <ul class="submenu-list">
            <li><a href="{% url "bin_booking_list" %}"><i class="fas fa-home"></i>
Bin Booking for Home</a></li>
            <li><a href="{% url "display_bin_booking_events" %}"><i class="fas fa-
calendar-alt"></i> Bin Booking for Events</a></li>
        </ul>
    </div>

    <script>
        function toggleSubMenu(submenuId) {
            var submenu = document.getElementById(submenuId);
            if (submenu.style.display === 'block') {
                submenu.style.display = 'none';
            } else {
                submenu.style.display = 'block';
            }
        }
    </script>
{% else %}
    <!-- Your other content for non-staff/superuser users -->


                    <!-- Booked Bin - Visible for customers -->
                    <li class="sidebar-item">
                        <a class="sidebar-link waves-effect waves-dark sidebar-
link" href="{% url 'bin_details' user.id %}" aria-expanded="false">
                            <i class="mdi mdi-delete"></i>
                            <span class="hide-menu">Booked Bin</span>
                        </a>
                    </li>
                    <!-- Event Booking Details - Visible for customers -->
                    <li class="sidebar-item">
                        <a class="sidebar-link waves-effect waves-dark sidebar-
link" href="{% url 'user_booked_events' %}" aria-expanded="false">
                            <i class="mdi mdi-calendar-text"></i>
                            <span class="hide-menu">Event Booking Details</span>
                        </a>
                    </li>
                    <li class="sidebar-item">
                        <a class="sidebar-link waves-effect waves-dark sidebar-
link" href="{% url 'collection_detail' user_id=request.user.id %}" aria-
expanded="false">
                            <i class="mdi mdi-calendar-text"></i>
                            <span class="hide-menu">Collection Details</span>
                        </a>
                    </li>
                    <li class="sidebar-item">
                        <a class="sidebar-link waves-effect waves-dark sidebar-
link" href="{% url "bins_low_fill_level" %}" aria-expanded="false">
```

```
                            <i class="mdi mdi-calendar-text"></i>
                            <span class="hide-menu">Bins Near you For
exchange</span>
                        </a>
                    </li>
                    {% endif %}
            </ul>

                </nav>
                <!-- End Sidebar navigation -->
            </div>
            <!-- End Sidebar scroll-->
        </aside>
        <!-- ============================================================== -->
        <!-- End Left Sidebar - style you can find in sidebar.scss  -->
        <!-- ============================================================== -->
        <!-- ============================================================== -->
        <!-- Page wrapper  -->
        <!-- ============================================================== -->
        <div class="page-wrapper">
            <!-- ============================================================== -->
            <!-- Bread crumb and right sidebar toggle -->
            <!-- ============================================================== -->
            <div class="page-breadcrumb">
                <div class="row align-items-center">
                    <div class="col-5">
                        <h4 class="page-title">Dashboard</h4>
                        <div class="d-flex align-items-center">
                            <nav aria-label="breadcrumb">
                                <ol class="breadcrumb">
                                    <li class="breadcrumb-item"><a
href="#">Home</a></li>
                                    <li class="breadcrumb-item active" aria-
current="page">Dashboard</li>
                                </ol>
                            </nav>
                        </div>
                    </div>


                </div>
            </div>
            <div class="row">
              <div class="col-md-8">
                  <div class="card">
                     <div class="card-body">
<!DOCTYPE html>
<html>
```

```
<head>
    <!-- Include necessary chart.js script -->
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
    <h1>Booking Chart</h1>
    <canvas id="bookingChart" width="1200" height="400"></canvas>

    <h1>User Chart</h1>
    <canvas id="userChart" width="1000" height="400"></canvas>

    <h1>Event Booking Chart</h1>
<canvas id="eventBookingChart" width="400" height="300"></canvas>


    <script>
        // Get the data passed from the view for the booking chart
        var bookingLabels = {{ booking_labels|safe }};
        var bookingData = {{ booking_data|safe }};

        // Create a booking chart using Chart.js
        var bookingCtx = document.getElementById('bookingChart').getContext('2d');
        var bookingChart = new Chart(bookingCtx, {
            type: 'bar',
            data: {
                labels: bookingLabels,
                datasets: [{
                    label: 'Booking Count',
                    data: bookingData,
                    backgroundColor: 'rgba(75, 192, 192, 0.2)',
                    borderColor: 'rgba(75, 192, 192, 1)',
                    borderWidth: 1
                }]
            },
            options: {
                scales: {
                    y: {
                        beginAtZero: true
                    }
                }
            }
        });

        // Get the data passed from the view for the user chart
        var userLabels = {{ user_labels|safe }};
        var userData = {{ user_data|safe }};

        // Create a user pie chart using Chart.js
        var userCtx = document.getElementById('userChart').getContext('2d');
```

```
        var userChart = new Chart(userCtx, {
            type: 'bar',
            data: {
                labels: userLabels,
                datasets: [{
                    data: userData,
                    backgroundColor: [
                        'rgba(255, 99, 132, 0.2)',
                        'rgba(54, 162, 235, 0.2)'
                    ],
                    borderColor: [
                        'rgba(255, 99, 132, 1)',
                        'rgba(54, 162, 235, 1)'
                    ],
                    borderWidth: 1
                }]
            },
            options: {
                responsive: false,  // Disable responsive mode to maintain fixed
dimensions
                plugins: {
                    legend: {
                        position: 'top',
                    },
                },
            }
        });

        var eventCategories = {{ event_categories|safe }};
        var eventCounts = {{ event_counts|safe }};

        // Create an event booking pie chart using Chart.js
        var eventBookingCtx =
document.getElementById('eventBookingChart').getContext('2d');
        var eventBookingChart = new Chart(eventBookingCtx, {
            type: 'pie',
            data: {
                labels: eventCategories,
                datasets: [{
                    label: 'Event Booking Count',
                    data: eventCounts,
                    backgroundColor: 'rgba(75, 192, 192, 0.2)',
                    borderColor: 'rgba(75, 192, 192, 1)',
                    borderWidth: 1
                }]
            },
            options: {
```

```
                responsive: false, // Disable responsive mode to maintain fixed
dimensions
                plugins: {
                    legend: {
                        position: 'top',
                    },
                },
            }
        });


    </script>
</body>
from django.shortcuts import render
from .models import BinBooking, User

def booking_chart(request):
    booking_data = BinBooking.objects.all()
    event_bookings = EventBooking.objects.all()

    booking_labels = []
    booking_data_points = []
    event_booking_counts = {}
    for booking in booking_data:
        booking_labels.append(booking.booking_id)
        booking_data_points.append(1)


    user_labels = []
    user_data_points = []

    for user in users:
        user_labels.append(user.username)  # You can use any user attribute as the
label
        user_data_points.append(1 if user.is_active else 0)  # 1 for active, 0 for
inactive
    for event_booking in event_bookings:
        category = event_booking.event.category
        if category in event_booking_counts:
            event_booking_counts[category] += 1
        else:
            event_booking_counts[category] = 1

    # Extract category labels and counts
    event_categories = list(event_booking_counts.keys())
    event_counts = list(event_booking_counts.values())
    context = {
        'booking_labels': booking_labels,
```

```
        'booking_data': booking_data_points,
        'user_labels': user_labels,
        'user_data': user_data_points,
        'event_categories': event_categories,
        'event_counts': event_counts,
    }

    return render(request, 'admin/chart.html', context)
```

Feedback

```
{% load static %}
<html>
<head>
    <!-- Add your CSS and other head content here -->
    <style>
        body {
            background-image: url('{% static "images\orginal\10.jpg" %}'); /*
Replace with the URL of your background image */
            background-size: cover;
            background-repeat: no-repeat;
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
        }

        /* Add CSS styles for the entire form */
        h1 {
            font-size: 36px;
            text-align: center;
            margin-bottom: 20px;
            color: #fff; /* Text color for the heading */
        }

        form {
            max-width: 500px;
            margin: 0 auto;
            text-align: center;
            background-color: rgba(255, 255, 255, 0.8); /* Background color with
transparency */
            padding: 20px;
            border-radius: 10px;
        }

        /* Add CSS styles for the star rating */
        .star-rating {
```

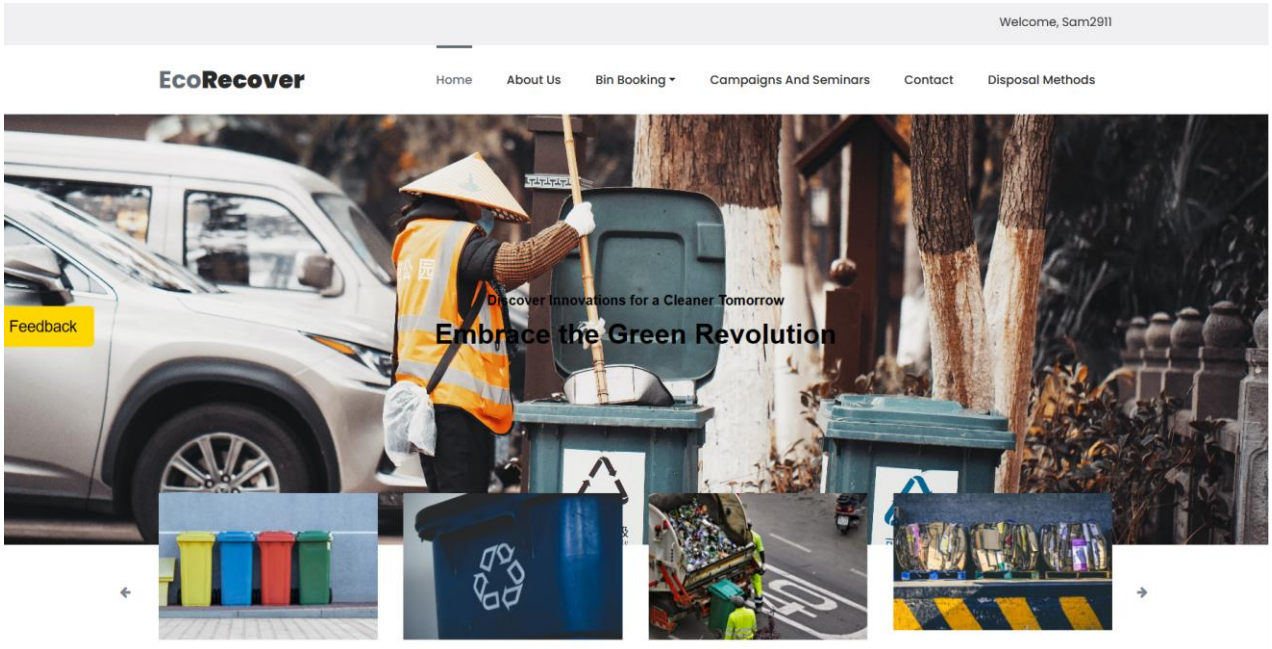
```
            font-size: 0;
            display: inline-block;
            margin-bottom: 20px;
        }

        .star-rating input[type="radio"] {
            display: none;
        }

        .star-rating label {
            font-size: 30px;
            cursor: pointer;
            padding: 5px;
            color: #ccc; /* Default star color */
        }

        .star-rating label::before {
            content: "\2605"; /* Unicode character for a star (you can use an image
instead) */
            display: inline-block;
        }

        .star-rating input[type="radio"]:checked ~ label,
        .star-rating input[type="radio"]:hover ~ label {
            color: #FFD700; /* Yellow color for selected stars and stars on hover
*/
        }

        /* Add CSS styles for the feedback textarea and button */
        p {
            font-size: 24px;
            margin-bottom: 10px;
            color: #333; /* Text color for paragraphs */
        }

        #message {
            width: 100%;
            padding: 10px;
            border: 1px solid #ccc;
            border-radius: 5px;
        }

        button {
            background-color: #FFD700;
            color: #000;
            font-size: 20px;
            padding: 10px 20px;
            border: none;
```

```
            border-radius: 5px;
            cursor: pointer;
            transition: background-color 0.3s;
        }

        button:hover {
            background-color: #FFA500; /* Orange color on hover */
        }

    </style>
</head>
<body>
    <h1>Feedback Form</h1>
    <form method="post">
        {% csrf_token %}
        <p>Rate Us:</p>
        <div class="star-rating">
            <input type="radio" id="star1" name="star_rating" value="1" /><label
for="star1"></label>
            <input type="radio" id="star2" name="star_rating" value="2" /><label
for="star2"></label>
            <input type="radio" id="star3" name="star_rating" value="3" /><label
for="star3"></label>
            <input type="radio" id="star4" name="star_rating" value="4" /><label
for="star4"></label>
            <input type="radio" id="star5" name="star_rating" value="5" /><label
for="star5"></label>
        </div>

        <p>Your Feedback:</p>
        <textarea name="message" id="message" rows="4" required></textarea>

        <br>
        <br>
        <button type="submit">Submit Feedback</button>
    </form>

    <!-- JavaScript to handle star rating interactions -->
    <script>
        const starRatingInputs = document.querySelectorAll('.star-rating
input[type="radio"]');
        starRatingInputs.forEach(input => {
            input.addEventListener('change', () => {
                // Get the index of the selected star
                const selectedIndex = Array.from(starRatingInputs).indexOf(input);

                // Update the color of stars up to the selected star
                for (let i = 0; i <= selectedIndex; i++) {
```

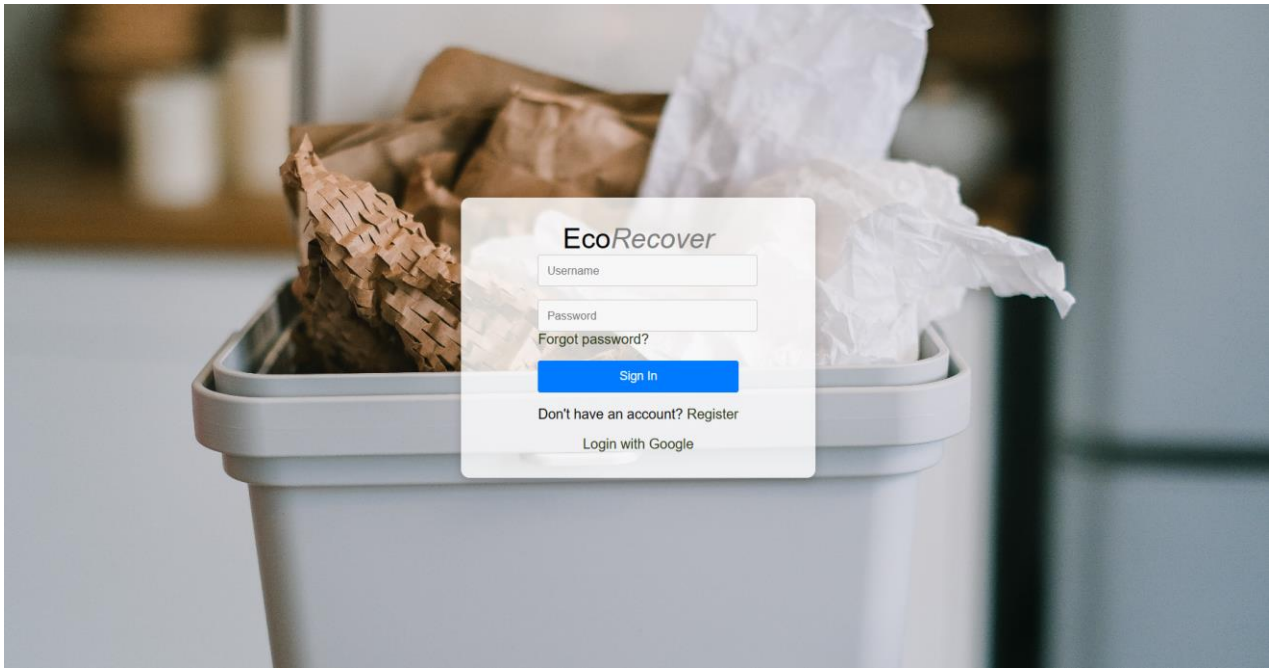
```
                    starRatingInputs[i].nextElementSibling.style.color = '#FFD700';
                }

                // Reset the color of stars after the selected star
                for (let i = selectedIndex + 1; i < starRatingInputs.length; i++) {
                    starRatingInputs[i].nextElementSibling.style.color = '#ccc';
                }
            });
        });
    </script>
</html>
```

9.1 Screen Shots

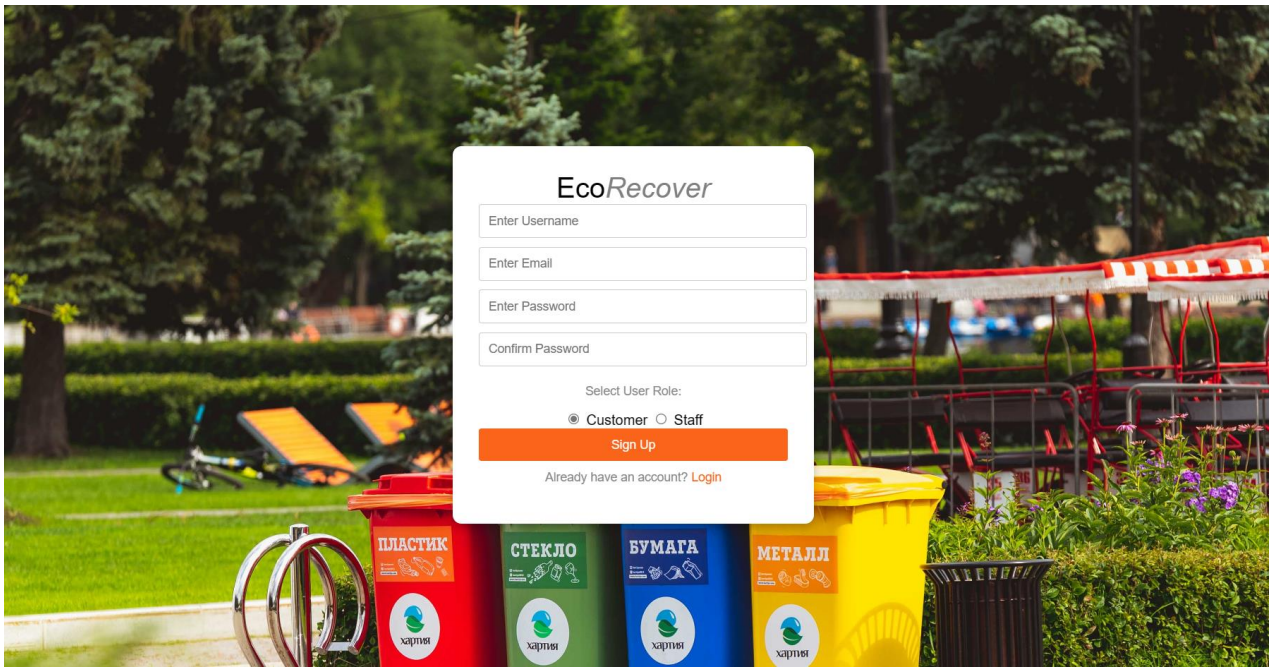
1. Index page



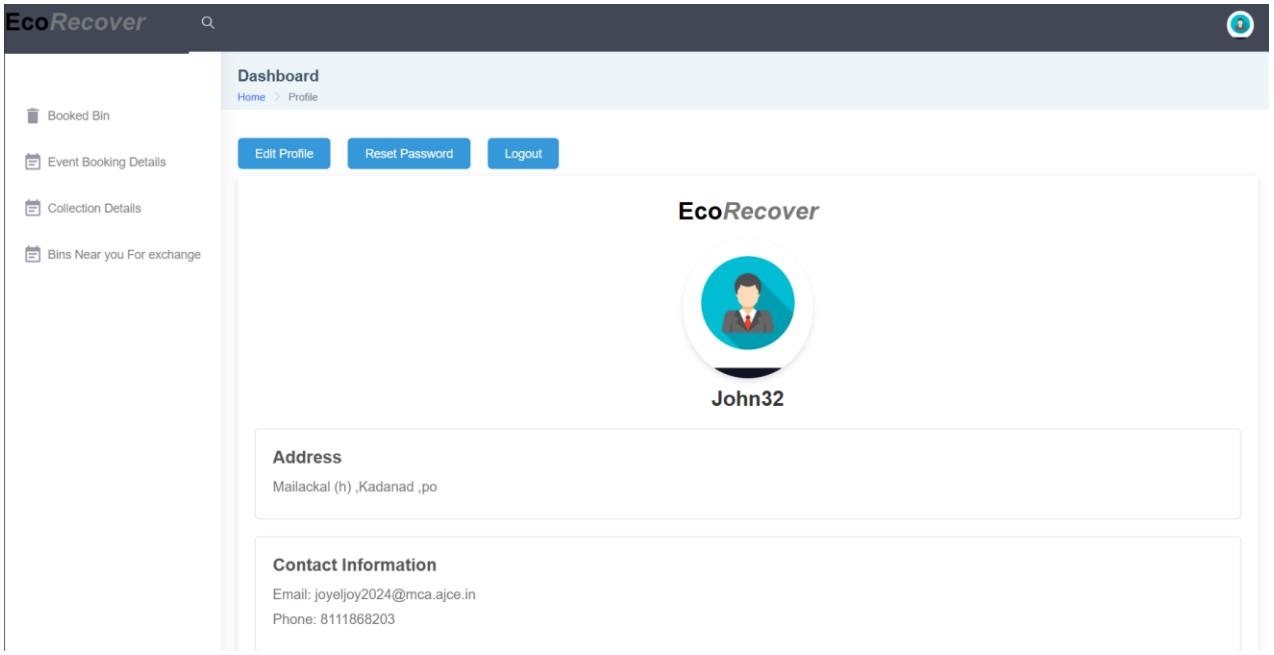
2 . Login page



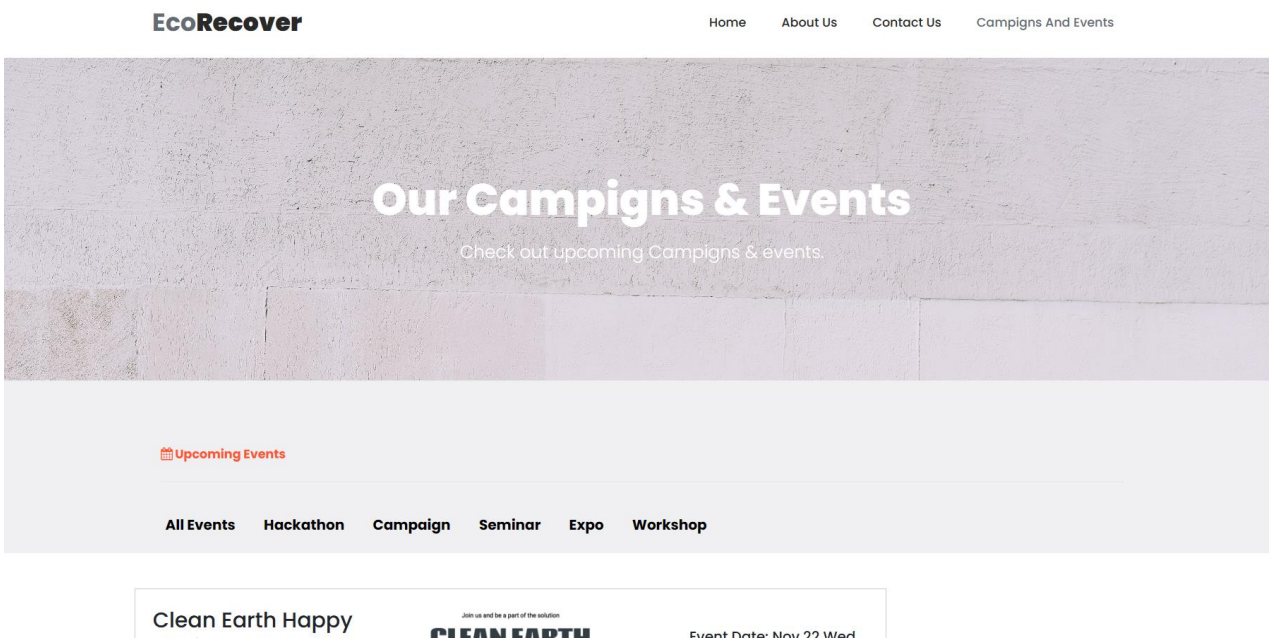
3.Registration page



1. User Profile

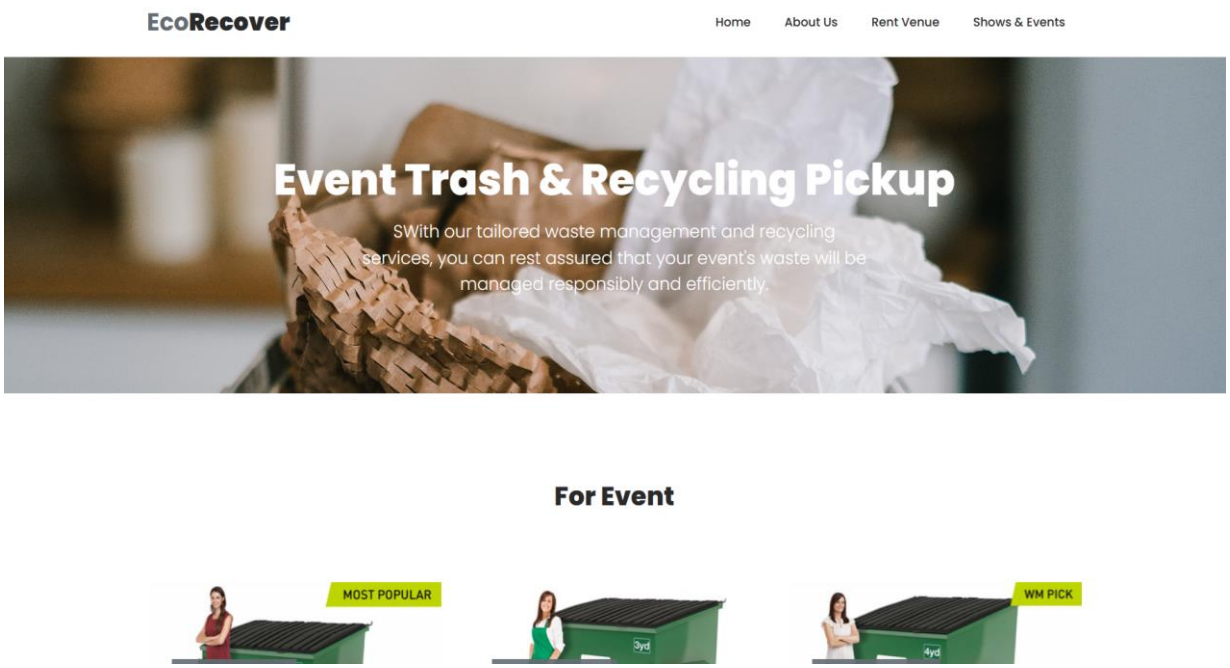


2. Campaigns and Seminar Booking page

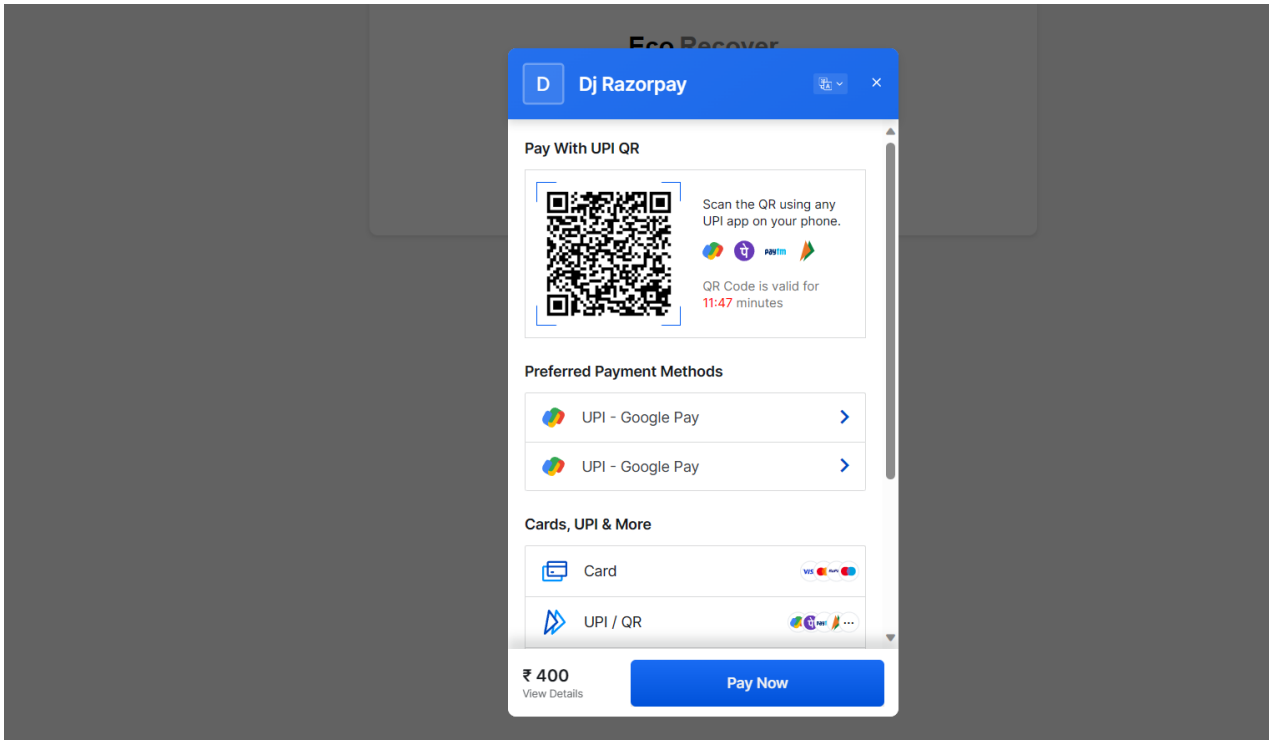


3. Bin Booking for Home

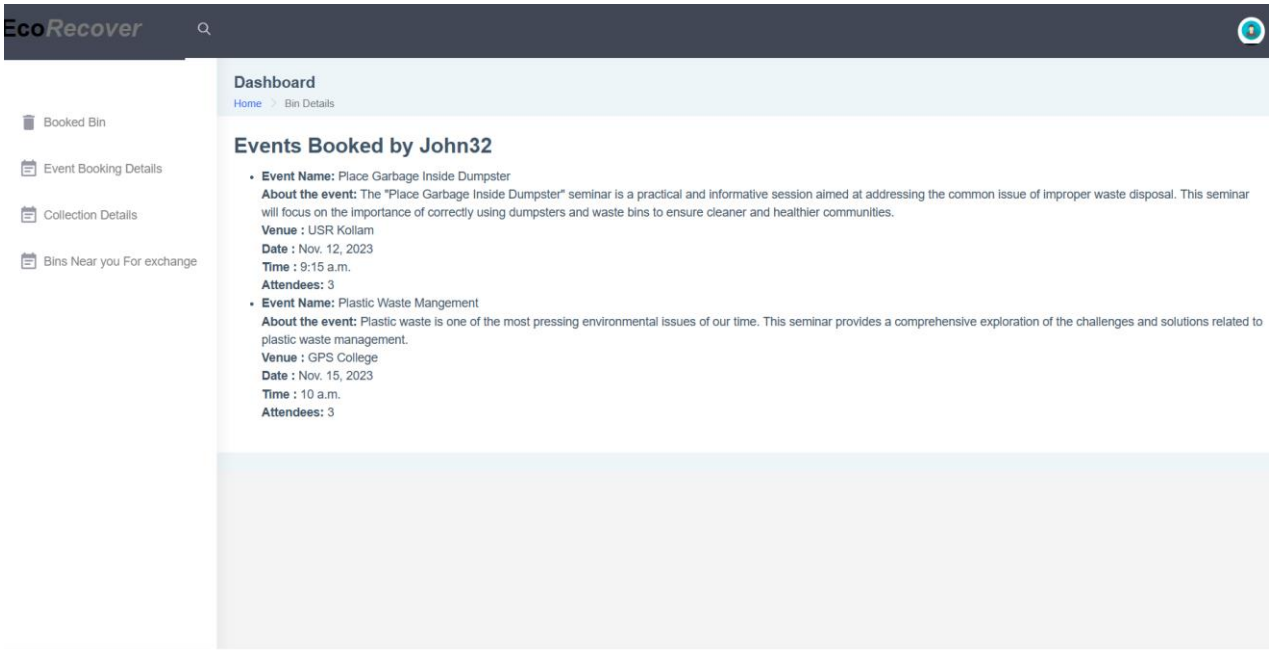
4. Bin Booking for Events



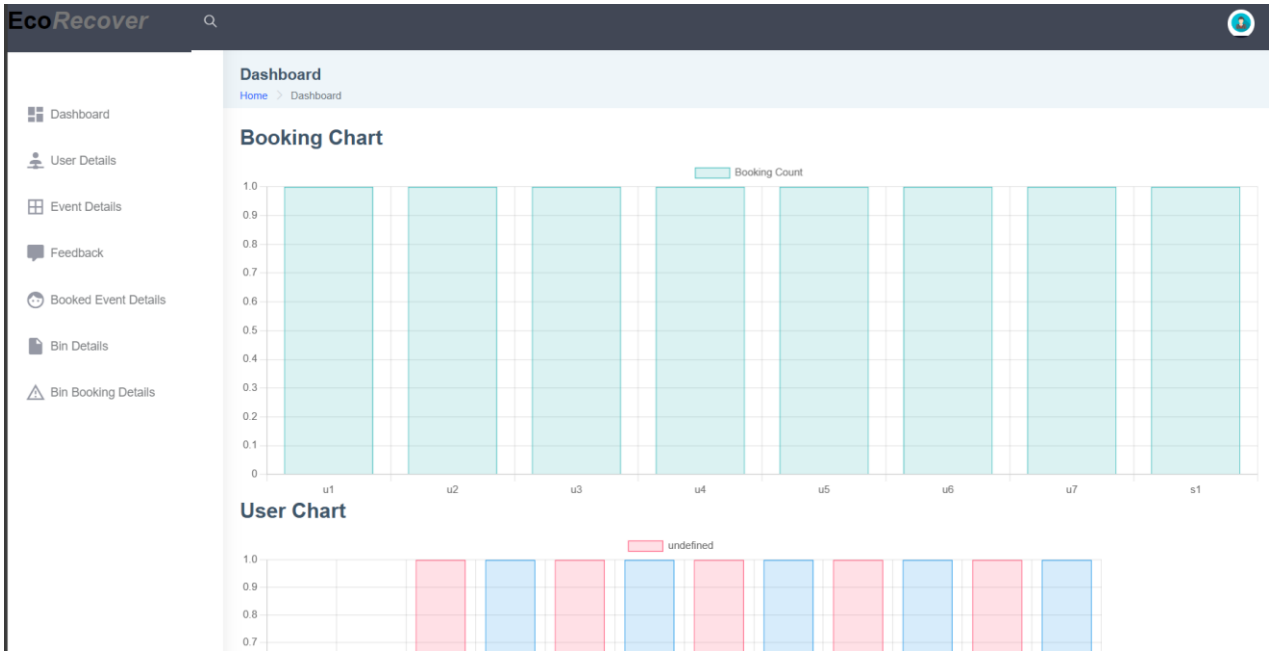
8 . Payment page



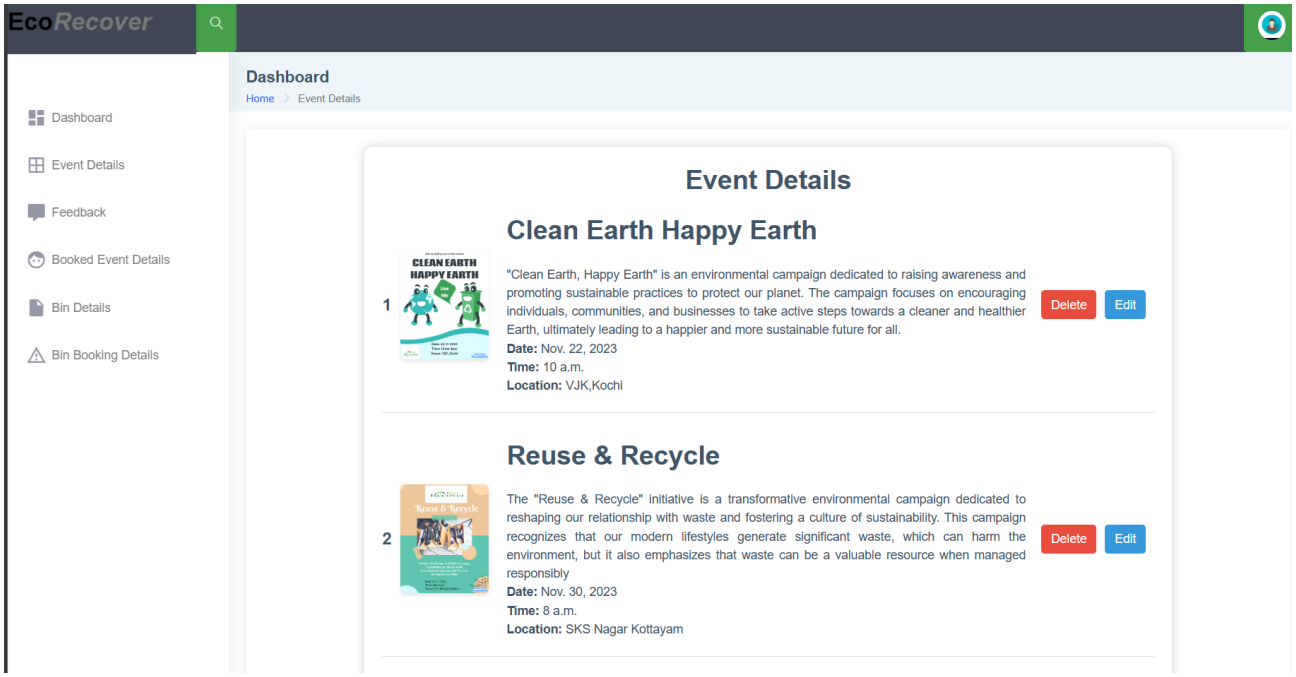
9. Booked Event Details



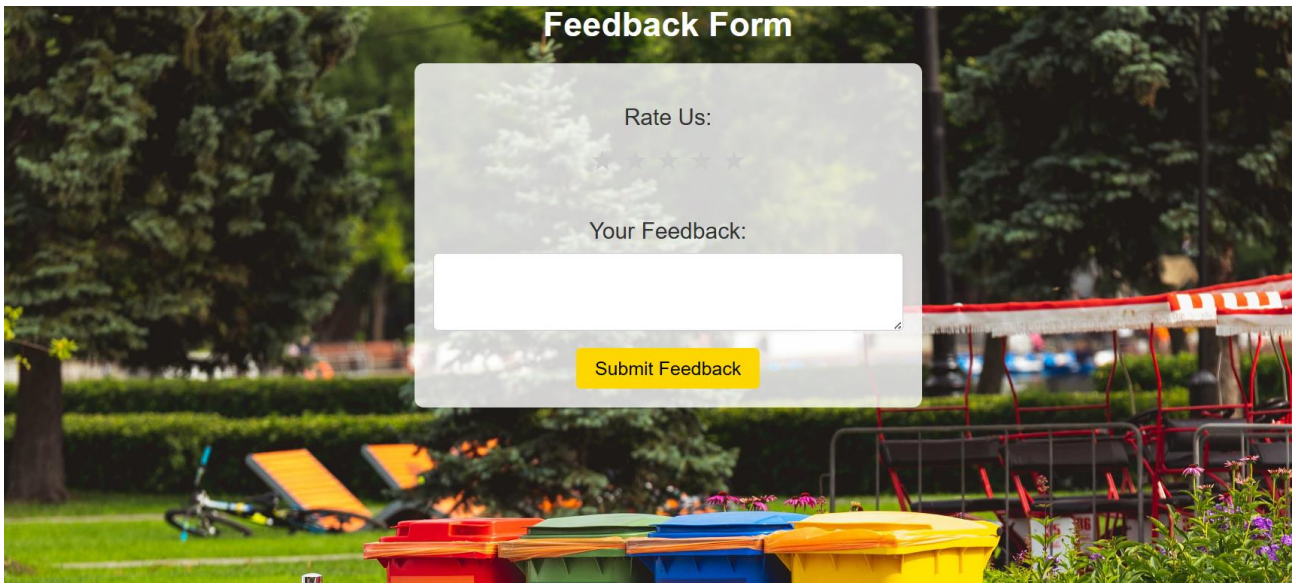
10.Admin Dashboard



11. Event Manager Dashboard



12. Feedback Form



13. Waste Classification and Disposal Method