

AQUARIUM AND PETSHOP MANAGEMENT SYSTEM E-COMMERCE PLATFORM FOR PETS

Mini Project Report

Submitted by

ROSHAN GEORGE

Reg. No.: AJC22MCA-2075

In Partial Fulfillment for the Award of the Degree of

MASTER OF COMPUTER APPLICATIONS

(MCA TWO YEAR)

[Accredited by NBA]

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, “**Aquarium and Petshop Management System**” is the bona fide work of **ROSHAN GEORGE (Regno: AJC22MCA-2075)** in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2023-24.

Ms. Jetty Benjamin

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 “**Aquarium and Petshop Management System**” 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: 05-12-2023

ROSHAN GEORGE

KANJIRAPPALLY

Reg: AJC22MCA-2075

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 **Ms. Meera Rose Mathew** for her 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 **Ms. Betty Benjamin** 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.

ROSHAN GEORGE

ABSTRACT

Aquarium and Pet Shop Management System The Aquarium and Pet Shop Management System is a web-based application built using the Python Django framework. The project aims to provide an organized and efficient platform for managing a virtual aquarium and pet shop, facilitating the management of diverse aquatic life and pets. This project encompasses various features such as pet inventory management, user authentication, search and filtering and more. This Project has four users Admin, Dealer, Customer and Delivery Man. In mini project there is only three users(except Delivery Man) and In main project there are four users. In this project i am including the technologies like email verification in user login and dealer login and in main project implementing the payment technology and i would like to implement machine learning in Product showcase

CONTENT

SL. NO	TOPIC	PAGE NO
1	INTRODUCTION	1
1.1	PROJECT OVERVIEW	2
1.2	PROJECT SPECIFICATION	2
2	SYSTEM STUDY	4
2.1	INTRODUCTION	5
2.2	EXISTING SYSTEM	5
2.3	DRAWBACKS OF EXISTING SYSTEM	6
2.4	PROPOSED SYSTEM	6
2.5	ADVANTAGES OF PROPOSED SYSTEM	7
3	REQUIREMENT ANALYSIS	8
3.1	FEASIBILITY STUDY	9
3.1.1	ECONOMICAL FEASIBILITY	9
3.1.2	TECHNICAL FEASIBILITY	9
3.1.3	BEHAVIORAL FEASIBILITY	10
3.1.4	FEASIBILITY STUDY QUESTIONNAIRE	10
3.2	SYSTEM SPECIFICATION	15
3.2.1	HARDWARE SPECIFICATION	15
3.2.2	SOFTWARE SPECIFICATION	15
3.3	SOFTWARE DESCRIPTION	16
3.3.1	DJANGO	16
3.3.2	SQLITE	18
4	SYSTEM DESIGN	19
4.1	INTRODUCTION	19
4.2	UML DIAGRAM	19
4.2.1	USE CASE DIAGRAM	20
4.2.2	SEQUENCE DIAGRAM	22
4.2.3	STATE CHART DIAGRAM	24
4.2.4	ACTIVITY DIAGRAM	25
4.2.5	CLASS DIAGRAM	27
4.2.6	OBJECT DIAGRAM	29
4.2.7	COMPONENT DIAGRAM	30

4.2.8	DEPLOYMENT DIAGRAM	30
4.2.9	COLLABORATION DIAGRAM	31
4.3	USER INTERFACE DESIGN USING FIGMA	32
4.4	DATABASE DESIGN	34
5	SYSTEM TESTING	34
5.1	INTRODUCTION	42
5.2	TEST PLAN	43
5.2.1	UNIT TESTING	43
5.2.2	INTEGRATION TESTING	44
5.2.3	VALIDATION TESTING	44
5.2.4	USER ACCEPTANCE TESTING	45
5.2.5	AUTOMATION TESTING	45
5.2.6	SELENIUM TESTING	46
6	IMPLEMENTATION	59
6.1	INTRODUCTION	60
6.2	IMPLEMENTATION PROCEDURE	60
6.2.1	USER TRAINING	60
6.2.2	TRAINING ON APPLICATION SOFTWARE	61
6.2.3	SYSTEM MAINTENANCE	61
7	CONCLUSION & FUTURE SCOPE	62
7.1	CONCLUSION	63
7.2	FUTURE SCOPE	63
8	BIBLIOGRAPHY	64
9	APPENDIX	66
9.1	SAMPLE CODE	67
9.2	SCREEN SHOTS	77

List of Abbreviation

- UML - Unified Modelling Language
- ORM - Object-Relational Mapping
- MVT - Model-View-Template
- MVC - Model-View-Controller
- RDBMS - Relational Database Management System
- 1NF - First Normal Form
- 2NF - Second Normal Form
- 3NF - Third Normal Form
- IDE - Integrated Development Environment
- HTML - HyperText Markup Language
- JS - JavaScript
- CSS - Cascading Style Sheets
- AJAX - Asynchronous JavaScript and XML • JSON - JavaScript Object Notation
- API - Application Programming Interface
- UI - User Interface
- HTTP - Hypertext Transfer Protocol
- URL - Uniform Resource Locator
- PK - Primary Key
- FK - Foreign Key
- SQL - Structured Query Language
- CRUD - Create, Read, Update, Delete

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

The Aquarium and Pet Shop Management System is a comprehensive web-based application designed to streamline the operations of a virtual aquarium and pet shop. Utilizing the Python Django framework, this project incorporates features such as user authentication, pet inventory management, search and filtering functionalities, and more

1.2 PROJECT SPECIFICATION

Customer View:

1. Customer Dashboard:

- . Displays a personalized welcome message upon login
- . Displays relevant categories for quick navigation.
- . Displays a grid of available products with images, names, and prices.

2. Search Bar:

- . Allows customers to search for specific products.
- . Provides autocomplete suggestions for ease of use.

3. Advanced Filters:

Enables customers to filter products based on categories, breeds, prices, and ratings.

4. Product Details:

Clicking on a product reveals detailed information, including images, descriptions, and user reviews.

5. Add to Cart:

Allows customers to add products to their shopping cart.

6. Checkout Process:

- . Guides customers through a secure payment process.
- . Provides order confirmation and receipt upon successful payment.

Dealer View:

1. Dealer Dashboard:

- . Displays a personalized welcome message upon login.

2. Inventory Management:

- . Displays a list of all available products with images, descriptions, and prices.
- . Allows dealers to add new products, update existing ones, or remove items from

inventory.

3. Order and Delivery Management:

- . Provides an overview of pending and completed orders.
- . Real-time updates on delivery status.

4. Account Settings:

- . Allows dealers to update their contact details and business information.
- . Provides an option to change the login password securely.

Admin View:

a. Admin Dashboard:

- . Provides an overview of the entire system, including user statistics, sales analytics, and product trends.

2. User Management:

- . Access to customer and dealer profiles.
- . Ability to add, edit, and delete user accounts.

3. Inventory Control:

- . Ability to add, edit, and delete products and categories.
- . Monitors overall inventory levels.

4. Order and Delivery Oversight:

- . Real-time tracking of all orders.
- . Option to intervene or resolve issues in case of disputes.

5. System Security:

- . Manages user authentication and access control.
- . Monitors login activity for security.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

In a world where technology intertwines seamlessly with everyday tasks, the Aquarium and Pet Shop Management System emerges as a pioneering web-based application, meticulously designed on the robust Python Django framework. This innovative project is envisioned to bring efficiency and organization to the dynamic landscape of virtual aquariums and pet shops. Recognizing the diverse needs of administrators, dealers, customers, and delivery personnel, the system offers a tailored and intuitive platform to streamline operations, manage inventory, and enhance the overall user experience.

2.2 EXISTING SYSTEM

2.2.1 NATURAL SYSTEM STUDIED

In the Aquarium and Pet Shop Management System, a natural system study is integrated to emulate ecological principles and enhance the user experience. Drawing inspiration from natural ecosystems, the project incorporates features such as adaptive inventory management based on biodiversity and ecological balance, mirroring the dynamics of real ecosystems. Natural Language Processing (NLP) is employed to create a more organic user interaction, adapting responses over time. Biological clock concepts inspired by circadian rhythms are implemented for automated scheduling, aligning with the natural behavior of virtual pets. Machine learning in the product showcase mimics natural selection, learning from user preferences for evolving and diverse recommendations. Environmental impact assessment features educate users about the ecological footprint of their choices, fostering responsible virtual pet ownership. This natural system integration contributes to a more realistic and immersive platform that not only entertains but also educates users about ecological considerations in pet management.

2.2.2 DESIGNED SYSTEM STUDIED

In the designed Aquarium and Pet Shop Management System, careful consideration has been given to creating an intuitive and efficient platform using the Python Django framework. The system is structured to cater to four distinct user roles: Admin, Dealer, Customer, and Delivery Man, each with specific responsibilities. The Admin, as the system overseer, has control over dealers and customers, with the ability to view details, manage pet inventory, and add pet and aquarium categories. The Dealer, representing pet shop owners or regular users, manages the inventory through functionalities such as adding, updating, and deleting products. Customers, the end-users purchasing pets and aquariums, experience a user-friendly interface with advanced

search and filtering options. The Delivery Man ensures the distribution of products. The technology stack comprises HTML and CSS for the frontend and Python Django for the backend. Noteworthy features include email verification for user and dealer logins, user authentication with role assignment, and advanced functionalities accessible through a comprehensive Admin Dashboard. The system undergoes evolution from a mini project to a main project, introducing the role of Delivery Man in the latter and incorporating payment technology for secure transactions. Furthermore, there is an ambitious plan to implement machine learning in the product showcase, enhancing the user experience with personalized recommendations. This designed system promises to revolutionize the management of virtual aquariums and pet shops, providing a seamless and innovative solution for diverse aquatic life and pet enthusiasts.

2.3 DRAWBACKS OF EXISTING SYSTEM

- Limited user roles and permissions
- Inefficient inventory management
- Poor user interface and experience
- Inadequate security measures
- Limited search and filtering options
- Lack of integration with modern technologies
- Absence of reporting and analytics features

2.4 PROPOSED SYSTEM

The proposed Aquarium and Pet Shop Management System addresses the limitations of the existing system through a multifaceted approach. It introduces a refined user role structure, ensuring distinct functionalities for Admins, Dealers, Customers, and the newly added Delivery Man. Enhanced inventory management streamlines processes and ensures accurate tracking. The user interface undergoes a redesign for improved aesthetics and user-friendliness. Robust security measures, including advanced authentication and encryption, safeguard user data. Advanced search and filtering options empower users, and the integration of modern technologies such as email verification, secure payment gateways, and machine learning for personalized showcases enhances overall functionality. The system introduces comprehensive reporting and analytics tools, providing administrators with valuable insights into user behaviors and sales trends, facilitating data-driven decision-making. In essence, the proposed system seeks to offer a versatile, secure, and user-centric

solution for efficient management of virtual aquariums and pet shops.

2.5ADVANTAGES OF PROPOSED SYSTEM

- Distinct User Roles
- Efficient Inventory Management
- Optimized User Interface
- Robust Security Measures
- Expanded Search and Filtering
- Integration of Modern Technologies
- Comprehensive Reporting

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

The primary purpose of conducting a feasibility study is to comprehensively assess whether the proposed project can successfully meet the organization's objectives in terms of available resources, labor, and time. This crucial study allows the developers and decision-makers to gain valuable insights into the project's potential viability and prospects. By carefully examining various aspects of the proposed system, such as its impact on the organization, its ability to fulfill user requirements, and the optimal utilization of resources, a feasibility study helps in determining the project's feasibility and potential success.

The assessment of the proposed project's feasibility involves multiple dimensions, each playing a critical role in the decision-making process.

- Technical Feasibility
- Operational Feasibility
- Economic Feasibility

A well-conducted feasibility study provides valuable insights to decision-makers, allowing them to make informed judgments about the project's potential success. It assists in identifying potential risks, challenges, and opportunities associated with the proposed endeavor, enabling stakeholders to devise effective mitigation strategies.

3.1.1 Economical Feasibility

The economic feasibility of the Aquarium and Pet Shop Management System is determined through a comprehensive analysis of costs, benefits, and anticipated financial impacts. This assessment involves a cost-benefit analysis, calculating the return on investment, and evaluating operational cost savings. The system's potential to generate revenue, contribute to market expansion, and enhance competitiveness is crucial. Scalability considerations, time-to-market impact, risk mitigation, regulatory compliance, and the total cost of ownership are key factors influencing economic viability. By meticulously examining these aspects, stakeholders can make informed decisions regarding the project's alignment with financial goals, ensuring it not only addresses operational needs efficiently but also offers a favorable economic outlook for sustainable success.

3.1.2 Technical Feasibility

The technical feasibility of the Aquarium and Pet Shop Management System is contingent on several critical considerations. The chosen technology stack, comprising Python Django for backend development and HTML/CSS for the frontend, must seamlessly integrate to provide a robust and efficient platform. The system architecture is evaluated for scalability and efficiency, ensuring it can handle anticipated user loads and future expansions. Data management capabilities, integration with external systems, and adherence to security protocols, including user authentication and access control, are paramount. Rigorous scalability and performance testing is conducted to identify and address potential bottlenecks. Additionally, the system's adaptability to technology updates, ease of maintenance, and the skill requirements for development and administration teams are integral factors in determining technical feasibility. By meticulously examining these elements, stakeholders can ascertain whether the project aligns with technical requirements and possesses the sustainability needed for successful development and implementation.

3.1.3 Behavioral Feasibility

The behavioral feasibility of the Aquarium and Pet Shop Management System is contingent on several crucial factors. It involves assessing the willingness of users, including administrators, dealers, customers, and delivery personnel, to accept and embrace the proposed system. This evaluation extends to compatibility with the existing organizational culture, necessitating alignment with established values and norms. A robust change management plan is essential, encompassing transparent communication, user involvement in design, and proactive strategies to address resistance. User training and support programs, ongoing monitoring of user satisfaction, and adaptability to evolving workflows are integral components of ensuring behavioral feasibility. By navigating these considerations, stakeholders can effectively gauge the system's alignment with user expectations, its seamless integration into organizational practices, and its potential to foster positive behavioral change within the organization.

3.1.4 Feasibility Study Questionnaire

1. Project Overview?

The Aquarium and Pet Shop Management System is a web-based platform designed to

streamline the management of virtual aquariums and pet shops. It facilitates the organization, sale, and tracking of diverse aquatic life and pet-related products. This system caters to administrators, pet shop owners (dealers), and customers, providing each with specific functionalities.

2. To what extent the system is proposed for?

The proposed "Aquarium and Pet Shop Management System" is designed to provide a comprehensive solution for managing and operating virtual aquariums and pet shops. The extent to which the system is proposed for includes:

Pet Shop Management:

Dealers (pet shop owners) can efficiently manage their inventory of pets and aquariums, including adding, updating, and removing product listings.

Inventory management tools ensure accurate stock tracking and availability management.

Customer Shopping Experience:

Customers can browse a wide range of pets and aquariums with advanced search and filtering options to find the perfect product.

Customers can place orders, receive order status updates, and complete transactions (if payment integration is implemented).

Feedback and Interaction:

Customers can leave reviews and ratings for products, providing valuable feedback to dealers and helping other customers make informed decisions.

Dealers can respond to customer inquiries, reviews, and ratings, fostering communication and customer satisfaction.

Administrator Control:

Administrators have full control over the system, including user account management, system configuration, and reporting.

Admin-specific dashboards and interfaces allow for efficient system management.

Security and Data Privacy:

The system implements robust security measures, including data encryption and secure

user authentication, to protect user data and transactions.

It ensures compliance with data privacy regulations.

Responsive Design and Accessibility:

The system is designed to be responsive, ensuring compatibility across various devices and screen sizes.

It can be deployed on web hosting platforms, making it accessible to users with an internet connection from various locations.

Reporting and Analytics:

Dealers and administrators can generate reports on sales, inventory, user activity, and more.

Analytics tools provide insights into user behavior and system performance.

Continuous Improvement:

The system incorporates a feedback mechanism for both customers and dealers, enabling continuous improvement in product listings and user experience.

Optional Payment Integration:

The system optionally integrates with payment gateways, allowing for secure online transactions.

Customization and Configuration:

Admins can configure system settings, including language, currency, and other behaviors to suit specific requirements.

The system is proposed to streamline the management and shopping experience related to aquatic life and pet products. It caters to the needs of pet shop owners, administrators, and customers, offering a user-friendly and efficient platform for their respective roles. Additionally, it provides valuable reporting and analytics tools for data-driven decision-making. The system can be further customized or extended based on specific project requirements and user feedback.

3. specify the viewers/public which is to be involved in the system?

In the "Aquarium and Pet Shop Management System," various stakeholders and users will interact with the system. Here's a specification of the viewers and the public who are

expected to be involved:

Administrators:

1. Pet shop owners
2. Customers
3. Potential Customers (Public)
4. Developers and IT personnel
5. Regulatory bodies or auditors
6. Pet Shop Owners (Dealers)

4. List the modules included in this project?

1. User Authentication and Access Control:

- User Registration
- User Login
- User Roles (Admin, Pet Shop Owner, Customer)
- User Profile Management
- Password Reset

2. Product Management:

- Add New Pet or Aquarium Listings
- Update Pet or Aquarium Details
- Delete Pet or Aquarium Listings
- View Pet or Aquarium Listings

3. Inventory Management:

- Track Pet and Aquarium Stock Levels
- Set Product Availability Status
- Update Product Prices and Quantities

4. Order Processing:

- Order Placement by Customers
- Order Notification to Pet Shop Owners
- Order Fulfillment and Status Updates

5. Search and Filtering:

- Search Pets and Aquariums by Species, Price, Location, etc.

Apply Filters to Narrow Down Results

6. User Interface and Display:

Display Pet and Aquarium Listings

Dynamic Web Page Generation using Django Templates

User-Friendly Interfaces for Data Entry and Interaction

7. Responsive Design:

Ensure Compatibility Across Various Devices and Screen Sizes

Optimize Layout for Desktop and Mobile Browsing

8. Admin Dashboard and Reporting:

Admin-Specific Interface for Managing Users and Content

Generate Reports on Sales, Stock Levels, etc.

9. Notifications and Alerts:

Notify Users of Product Updates, Promotions, etc.

Send Order Confirmation and Status Updates

10. Payment Integration (Optional):

Allow Customers to Make Payments Online

Integrate Payment Gateways for Secure Transactions

11. Feedback and Reviews:

Allow Customers to Leave Reviews and Ratings

Provide Feedback Mechanism for Continuous Improvement

12. Settings and Configuration:

Admin Panel for Configuring System Settings

Customize Application Behavior (e.g., currency, language)

13. Security and Data Privacy:

Implement Data Encryption and Protection

Ensure Secure User Authentication and Authorization

14.Reporting and Analytics:

Generate Reports on Sales, Inventory, User Activity, etc.

Utilize Analytics to Gain Insights into User Behavior

15.Support and Helpdesk:

Provide User Support and Assistance

Offer Help Documentation or chat bot

5. identify the users in this project?

1.Administrator

2.Dealer

3.Customer

4.Delivery Man

6. Who owns the system?

Admin has Ownership and control over the Aquarium and Petshop Management System

7. System is related to which firm/industry/organization ?

The system is related to Ecommerce. The platform For Selling and buying Aquarium and Pets

8. Details of persons that you have contacted for data collection?

Royal farm and aquarium shop

9. How flexible is the system for future iterations and updates based on evolving technology and user needs?

The Aquarium and Pet Shop Management System is inherently flexible for future iterations and updates, leveraging a modular architecture, open-source technologies, API integration, user feedback mechanisms, and adherence to industry standards to seamlessly adapt to evolving technology and user requirements.

10. Are there any legal or regulatory considerations in the management of pet and aquarium data?

Yes, there are legal and regulatory considerations in the management of pet and aquarium data, including adherence to animal welfare laws, privacy regulations for customer data, and compliance with any specific industry standards governing the sale and distribution of pets and aquariums

3.1 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor - i5
RAM - 16 gb
Hard disk - 1 Tb

3.2.2 Software Specification

Front End - HTML, JS, CSS, jQuery, Ajax, Bootstrap
Back End - Python-Django
Database - Sqlite
Client on PC - Windows 7 and above.
Technologies used - JS, HTML5, AJAX, J Query, PHP, CSS

3.3 SOFTWARE DESCRIPTION

3.3.1 Django

Django is a popular and powerful open-source web framework written in Python, designed to facilitate rapid development and maintainable web applications. It follows the Model-View-Template (MVT) architectural pattern, which is similar to the Model-View-Controller (MVC) pattern. Django provides a structured and efficient way to build web applications, offering several key components and features.

At its core, Django includes a robust Object-Relational Mapping (ORM) system that simplifies database interactions, allowing developers to work with Python objects instead of raw SQL queries. It also includes a URL dispatcher for mapping URLs to view functions, an automatic admin interface for managing application data, and a templating engine for creating dynamic and reusable user interfaces.

Django places a strong emphasis on security, with built-in features to protect against common web vulnerabilities. It offers authentication and authorization systems, middleware support for global request and response processing, and compatibility with various databases.

The framework's scalability, extensibility, and a vibrant community of developers make it a popular choice for building web applications, from simple websites to complex, high-traffic platforms. Django's extensive documentation and ecosystem of third-party packages further

enhance its appeal for web developers.

3.3.2 Sqlite

SQLite is a self-contained and serverless relational database management system (RDBMS) known for its simplicity and efficiency. It stores the entire database in a single file, eliminating the need for a separate server process, which simplifies deployment. This makes SQLite an ideal choice for embedded systems, mobile applications, and small to medium-sized projects. Developers interact with the database directly through function calls, making it an embedded database well-suited for various applications, including mobile apps, desktop software, and web applications.

Despite its lightweight nature, SQLite is ACID-compliant, ensuring data integrity with support for transactions. It offers a wide range of data types, and its compatibility with multiple programming languages, including Python, C/C++, and Java, makes it accessible to a diverse developer community. SQLite is open-source, free, and known for its speed and efficiency, particularly for read-heavy workloads. While not intended for extremely high-concurrency or large-scale applications, SQLite excels in scenarios where simplicity, portability, and low resource consumption are key requirements.

It is commonly used as a local data store, cache, or embedded database within larger applications, contributing to its versatility and widespread adoption in software development.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

The initial stage of developing any engineered product or system is the design phase, which involves a creative approach. A well-crafted design plays a critical role in ensuring the successful functioning of a system. Design is defined as the process of employing various techniques and principles to define a process or system in enough detail to enable its physical realization. This involves using different methods to describe a machine or system, explaining how it operates, in sufficient detail for its creation. In software development, design is a crucial step that is always present, regardless of the development approach. System design involves creating a blueprint for building a machine or product. Careful software design is essential to ensure optimal performance and accuracy. During the design phase, the focus shifts from the user to the programmers or those working with the database. The process of creating a system typically involves two key steps: Logical Design and Physical Design.

4.2 UML DIAGRAM

UML, which stands for Unified Modeling Language, is a standardized language used for specifying, visualizing, constructing, and documenting the elements of software systems. The Object Management Group (OMG) is responsible for the creation of UML, with the initial draft of the UML 1.0 specification presented to OMG in January 1997. Unlike common programming languages such as C++, Java, or COBOL, UML is not a programming language itself. Instead, it is a graphical language that serves as a tool for creating software blueprints.

UML is a versatile and general-purpose visual modeling language that facilitates the visualization, specification, construction, and documentation of software systems. While its primary use is in modeling software systems, UML's applications are not limited to this domain. It can also be employed to represent and understand processes in various contexts, including non-software scenarios like manufacturing unit processes.

It's important to note that UML is not a programming language, but it can be utilized with tools that generate code in different programming languages based on UML diagrams. UML encompasses

nine core diagrams that aid in representing various aspects of a system.

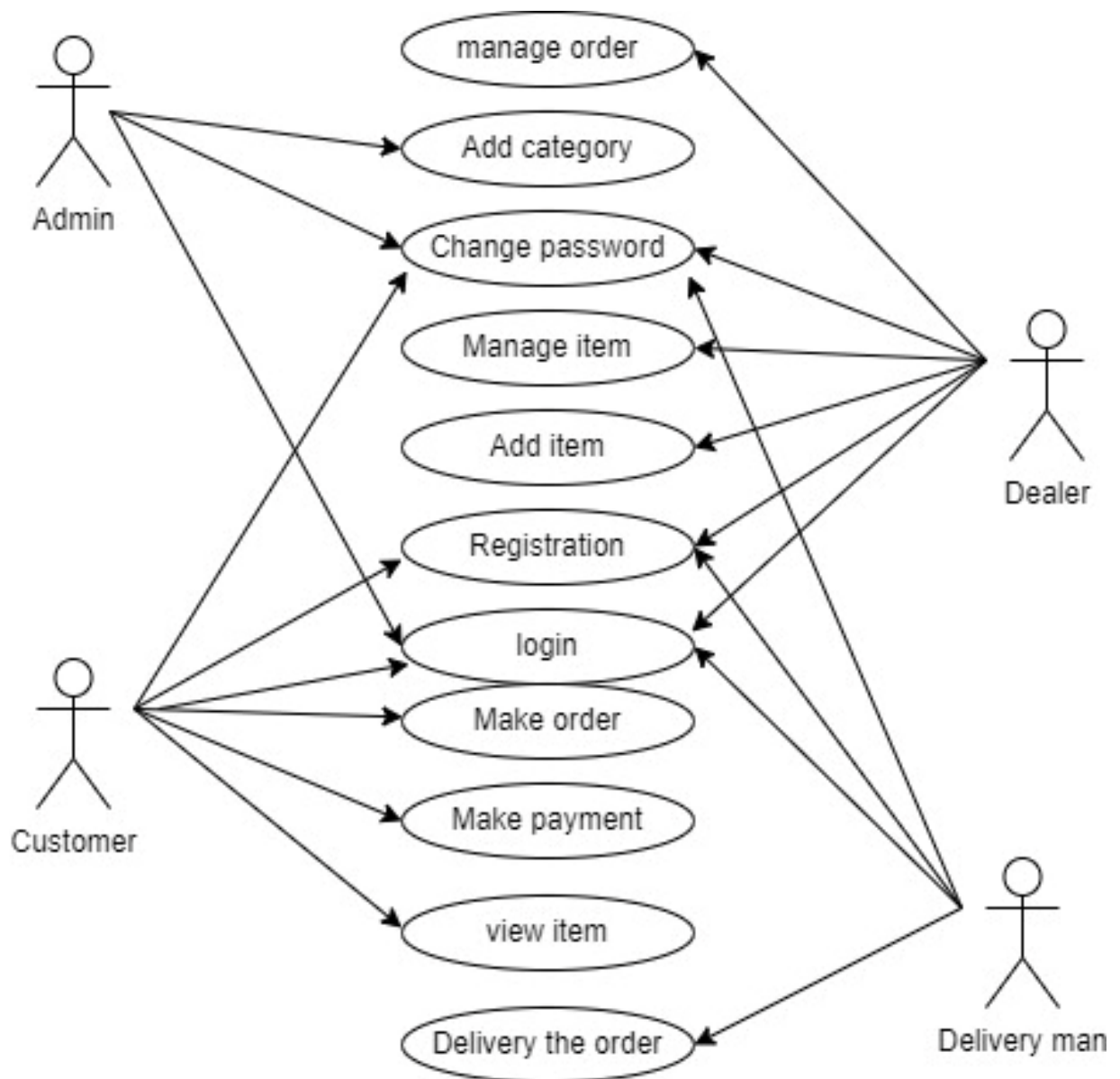
- Class diagram
- Object diagram
- Use case diagram
- Sequence diagram
- Activity diagram
- State chart diagram
- Deployment diagram
- Component diagram

4.2.1 USE CASE DIAGRAM

A use case is a tool for understanding a system's requirements and organizing them, especially in the context of creating or using something like a product delivery website. These tools are represented using "use case" diagrams within the Unified Modeling Language, a standardized way of creating models for real-world things and systems.

A use case diagram consists of these main elements:

- The boundary, which delineates the system and distinguishes it from its surroundings.
- Actors, representing individuals or entities playing specific roles within the system.
- The interactions between different people or elements in specific scenarios or problems.
- The primary purpose of use case diagrams is to document a system's functional specifications. To create an effective use case diagram, certain guidelines must be followed:
 - Providing clear and meaningful names for use cases and actors.
 - Ensuring that the relationships and dependencies are well-defined.
 - Including only necessary relationships for the diagram's clarity.
 - Utilizing explanatory notes when needed to clarify essential details.

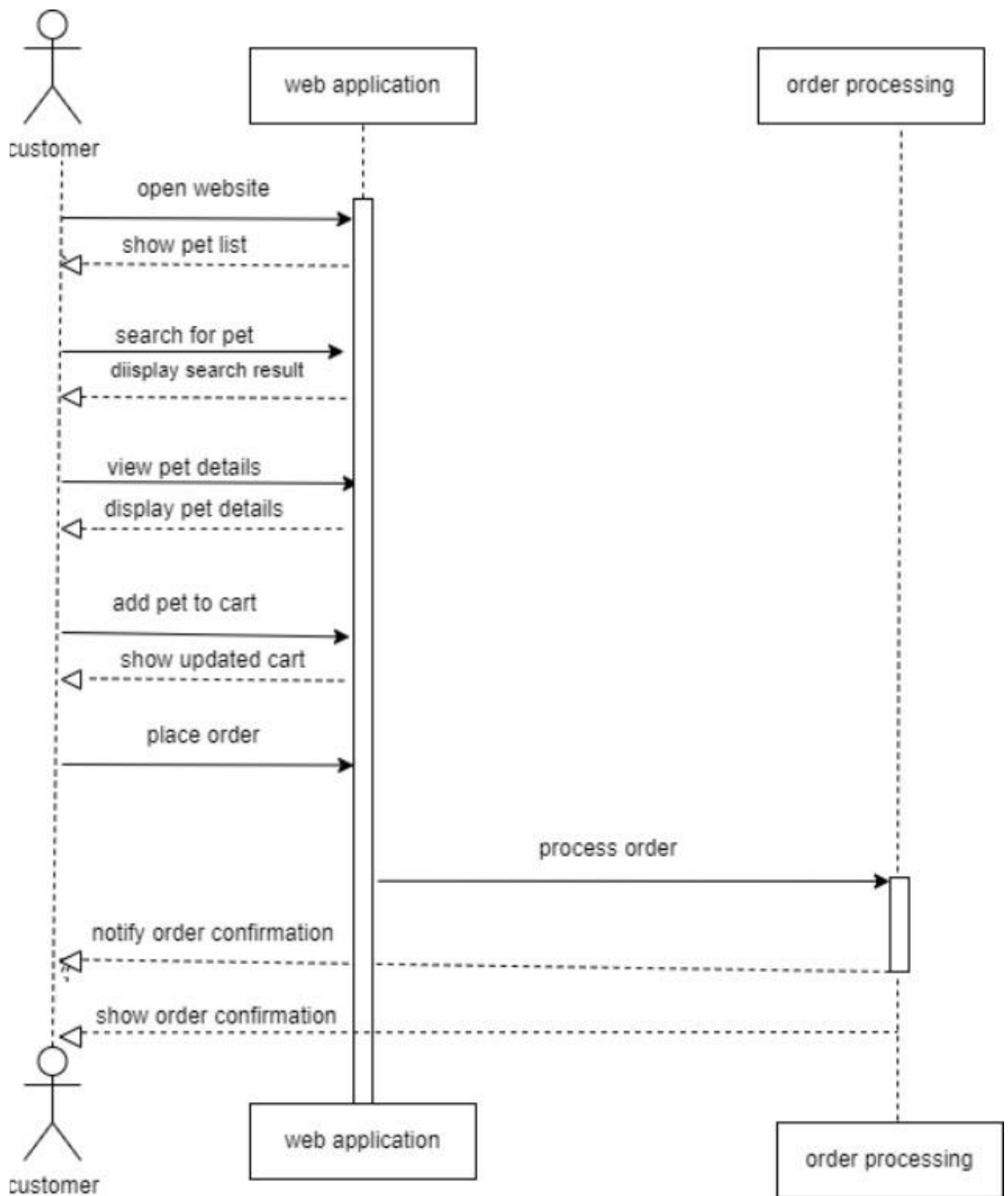


4.2.1 SEQUENCE DIAGRAM

A sequence diagram illustrates the specific order in which objects interact with each other, showcasing the sequential flow of events. This type of diagram is also known as event diagrams or event scenarios. Sequence diagrams serve the purpose of elucidating how various components of a system collaborate and the precise sequence in which these actions occur. These diagrams find frequent application among business professionals and software developers, aiding in the understanding and depiction of requirements for both new and existing systems.

Sequence Diagram Notations:

- i. Actors - Within a UML diagram, actors represent individuals who utilize the system and its components. Actors are not depicted within the UML diagram as they exist outside the system being modeled. They serve as role-players in a story, encompassing people and external entities. In a UML diagram, actors are represented by simple stick figures. It's possible to depict multiple individuals in a diagram that portrays the sequential progression of events.
- ii. Lifelines - In a sequence diagram, each element is presented as a lifeline, with lifeline components positioned at the top of the diagram.
- iii. Messages - Communication between objects is achieved through the exchange of messages, with messages being arranged sequentially on the lifeline. Arrows are employed to represent messages, forming the core structure of a sequence diagram.
- iv. Guards - Within the UML, guards are employed to denote various conditions. They are used to restrict messages in the event that specific conditions are met, providing software developers with insights into the rules governing a system or process.



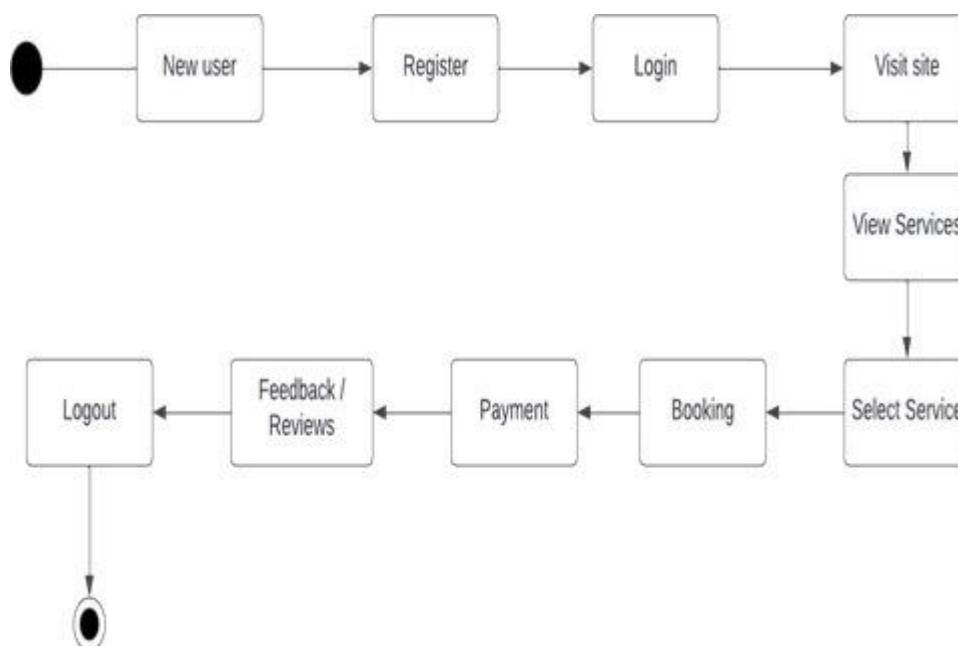
4.2.2 State Chart Diagram

A state machine diagram, also known as a state chart, visually represents the various states an object undergoes within a system and the sequence in which these states are traversed. It serves as a record of the system's behavior, illustrating the collaborative functioning of a group of entities, whether it's a team, a collection of students, a large assembly, or an entire organization. State machine diagrams are a valuable method for depicting the interactions of diverse components within a system, outlining how objects evolve in response to events and elucidating the diverse conditions that each entity or component can inhabit.

arrows

Notations within a state machine diagram encompass:

- Initial state: Symbolized by a black circle, this signifies the commencement of a process.
- Final state: Representing the conclusion of a process, this is denoted by a filled circle within another circle.
- Decision box: Shaped like a diamond, it aids in decision-making by considering the evaluation of a guard.
- Transition: Whenever a change in authority or state occurs due to an event, it is termed a transition. Transitions are depicted as with labels indicating the triggering event.
- State box: It portrays the condition or state of an element within a group at a specific moment. These are typically represented by rectangles with rounded corners.



4.2.2 Activity Diagram

A class diagram serves as a static blueprint for an application, illustrating its components and their relationships when the system is in a dormant state. It provides insights into the system's structure, showcasing the various elements it comprises and how they interact. In essence, a class diagram acts as a visual guide for software development, aiding in the creation of functional applications.

Key Aspects of a Class Diagram:

- **System Overview:** A class diagram offers a high-level representation of a software system, presenting its constituent parts, associations, and collaborative dynamics. It serves as an organizational framework for elements such as names, attributes, and methods, simplifying the software development process.
- **Structural Visualization:** The class diagram is a structural diagram that combines classes, associations, and constraints to define the system's architecture.

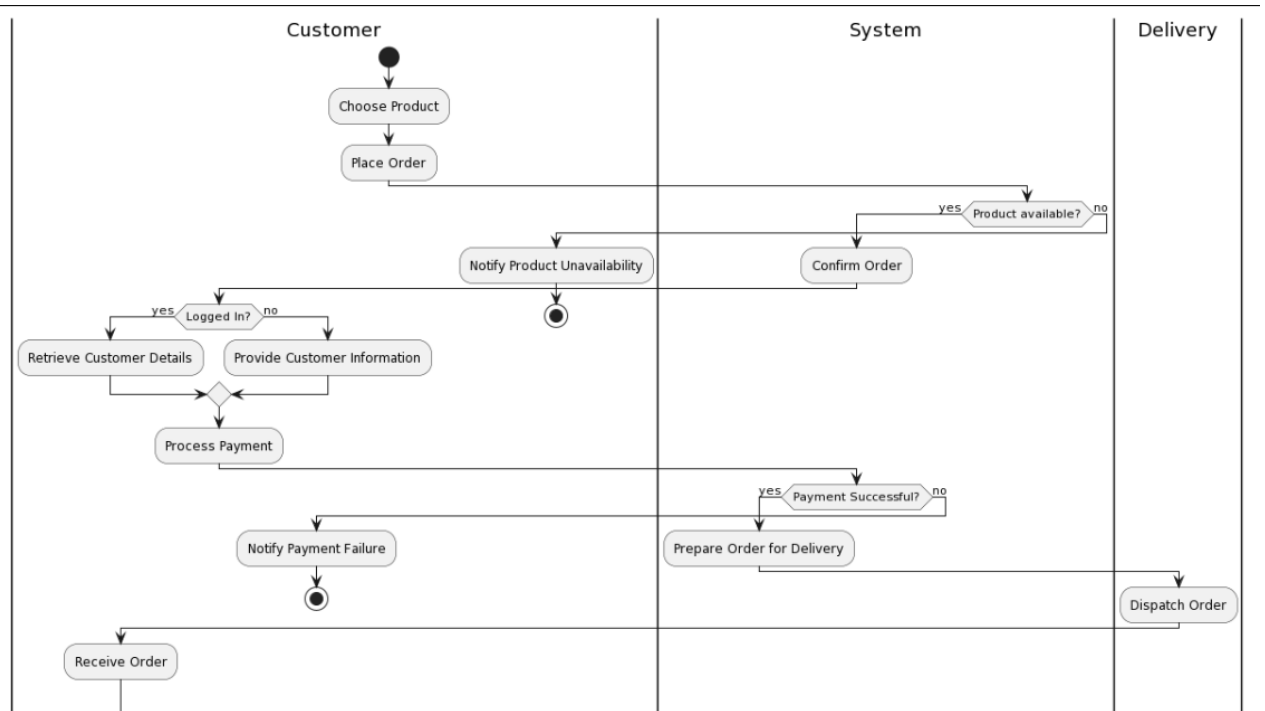
Components of a Class Diagram:

The class diagram comprises three primary sections:

- **Upper Section:** This top segment features the class name, representing a group of objects that share common attributes, behaviors, and roles. Guidelines for displaying groups of objects include capitalizing the initial letter of the class name, positioning it in the center, using bold lettering, and employing slanted writing style for abstract class titles.
- **Middle Section:** In this part, the class's attributes are detailed, including their visibility indicators, denoted as public (+), private (-), protected (#), or package (~).
- **Lower Section:** The lower section elaborates on the class's methods or operations, presented in a list format with each method on a separate line. It outlines how the class interacts with data.

In UML, relationships within a class diagram fall into three categories:

- **Dependency:** Signifying the influence of one element's changes on another.
- **Generalization:** Representing a hierarchical relationship where one class acts as a parent, and another serves as its child.
- **Association:** Indicating connections between elements.
- **Multiplicity:** Defining constraints on the number of instances allowed to possess specific characteristics, with one being the default value when not specified.
- **Aggregation:** An aggregation is a group that is a part of a relationship called association.



4.2.3 Class Diagram

A class diagram serves as a static blueprint for an application, illustrating its components and their relationships when the system is in a dormant state. It provides insights into the system's structure, showcasing the various elements it comprises and how they interact. In essence, a class diagram acts as a visual guide for software development, aiding in the creation of functional applications.

Key Aspects of a Class Diagram:

- **System Overview:** A class diagram offers a high-level representation of a software system, presenting its constituent parts, associations, and collaborative dynamics. It serves as an organizational framework for elements such as names, attributes, and methods, simplifying the software development process.
- **Structural Visualization:** The class diagram is a structural diagram that combines classes, associations, and constraints to define the system's architecture.

Components of a Class Diagram:

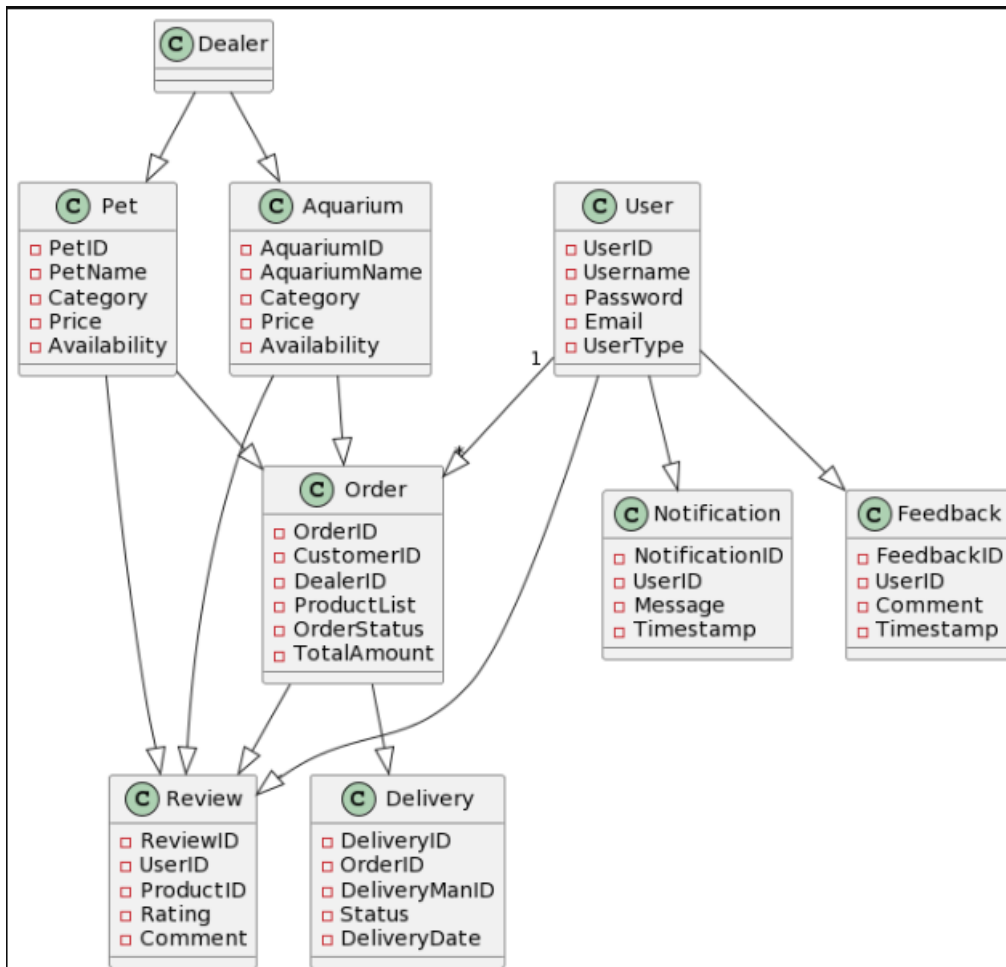
The class diagram comprises three primary sections:

- **Upper Section:** This top segment features the class name, representing a group of objects that share common attributes, behaviors, and roles. Guidelines for displaying groups of objects include capitalizing the initial letter of the class name, positioning it in the center, using bold lettering, and employing slanted writing style for abstract class titles.
- **Middle Section:** In this part, the class's attributes are detailed, including their visibility indicators, denoted as public (+), private (-), protected (#), or package (~).
- **Lower Section:** The lower section elaborates on the class's methods or operations, presented in a list format with each method on a separate line. It outlines how the class interacts with data.

In UML, relationships within a class diagram fall into three categories:

- **Dependency:** Signifying the influence of one element's changes on another.
- **Generalization:** Representing a hierarchical relationship where one class acts as a parent, and another serves as its child.
- **Association:** Indicating connections between elements.
- **Multiplicity:** Defining constraints on the number of instances allowed to possess specific characteristics, with one being the default value when not specified.

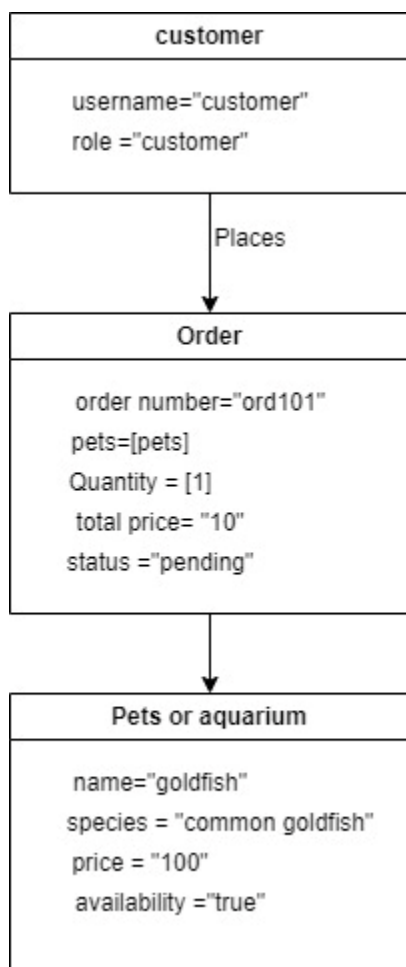
- Aggregation: An aggregation is a group that is a part of a relationship called association.
- Composition: "Composition" is like a smaller part of "aggregation.". This describes how a parent and child need each other, so if one is taken away, the other won't work anymore.



4.2.4 Object Diagram

Object diagrams are derived from class diagrams and rely on them to provide a visual representation. They offer an illustration of a collection of objects related to a particular class. Object diagrams provide a snapshot of objects in an object-oriented system at a specific point in time.

Object diagrams and class diagrams share similarities, but they also have distinctions. Class diagrams are more generalized and do not portray specific objects. This abstraction in class diagrams simplifies the comprehension of a system's functionality and structure.



4.2.5 Component Diagram

A component diagram serves the purpose of breaking down a complex system that utilizes objects into more manageable segments. It offers a visual representation of the system, showcasing its internal components such as programs, documents, and tools within the nodes.

This diagram elucidates the connections and organization of elements within a system, resulting in the creation of a usable system.

In the context of a component diagram, a component refers to a system part that can be modified and operates independently. It retains the secrecy of its internal operations and requires a specific method to execute a task, resembling a concealed box that functions only when operated correctly.

Notation for a Component Diagram includes:

- A component
- A node

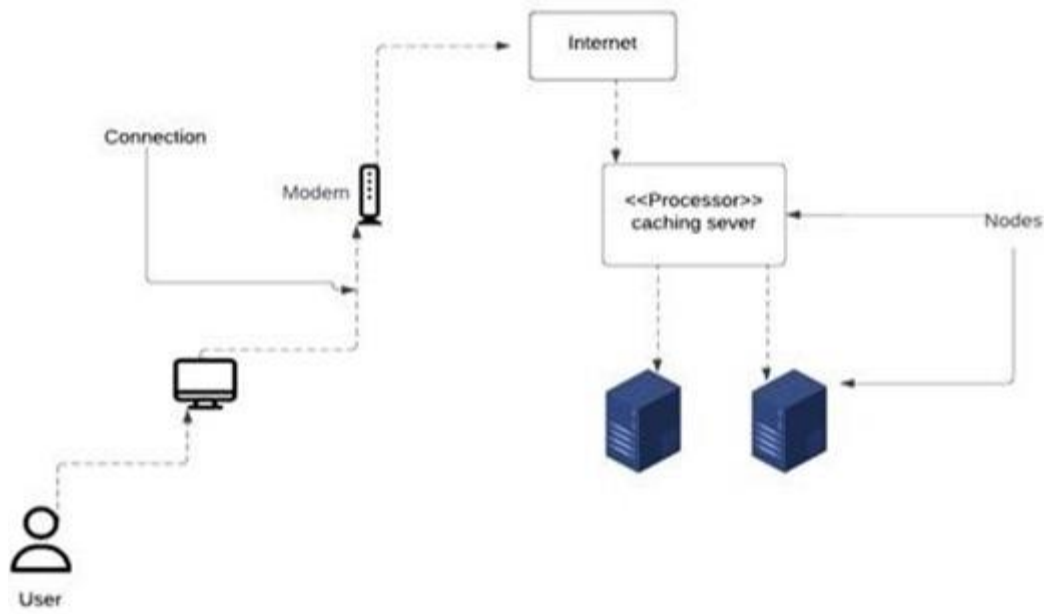
4.2.8 Deployment Diagram

A deployment diagram provides a visual representation of how software is positioned on physical computers or servers. It depicts the static view of the system, emphasizing the arrangement of nodes and their connections.

This type of diagram delves into the process of placing programs on computers, elucidating how software is constructed to align with the physical computer system.

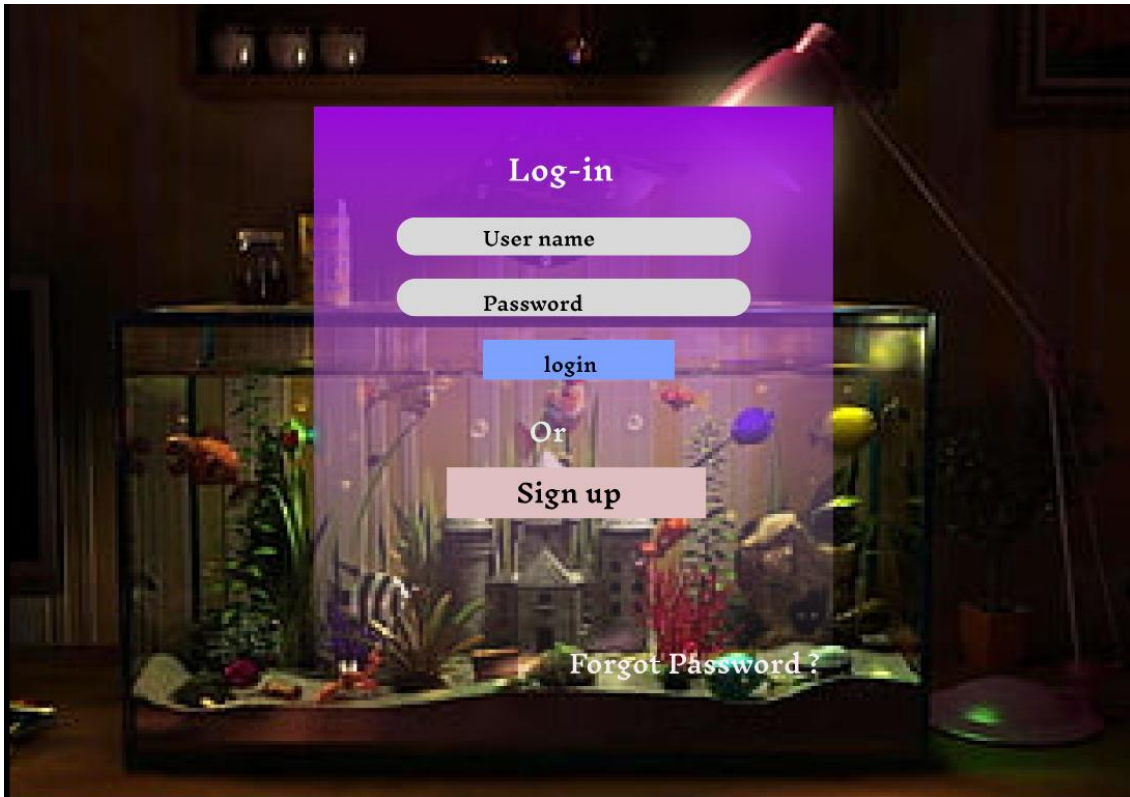
Notations in a Deployment Diagram include:

- A component
- An artifact
- An interface
- A node

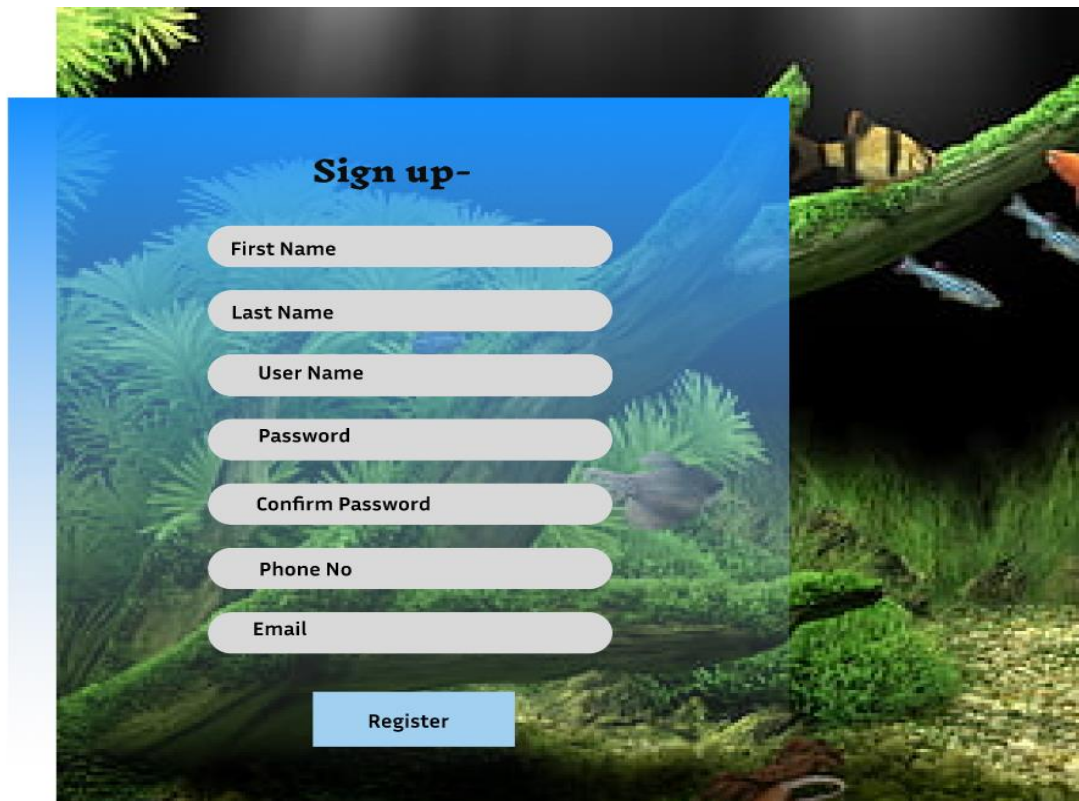


4.3 USER INTERFACE DESIGN USING FIGMA

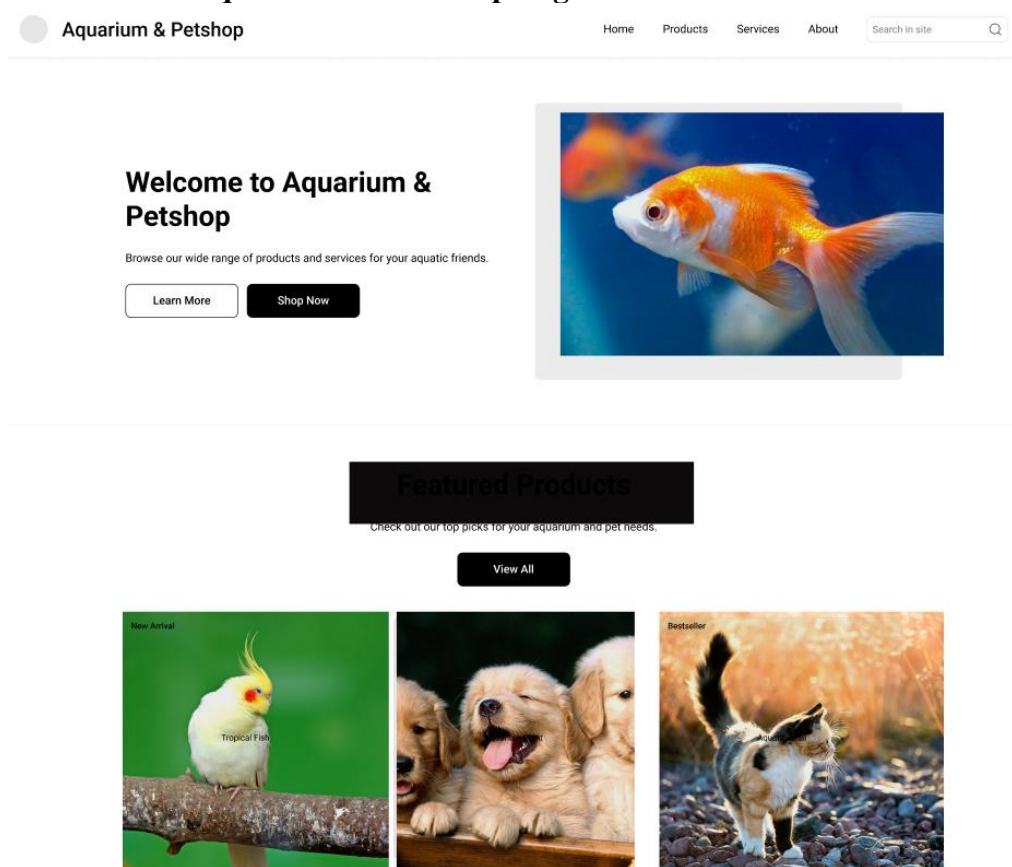
Form Name: Login Page



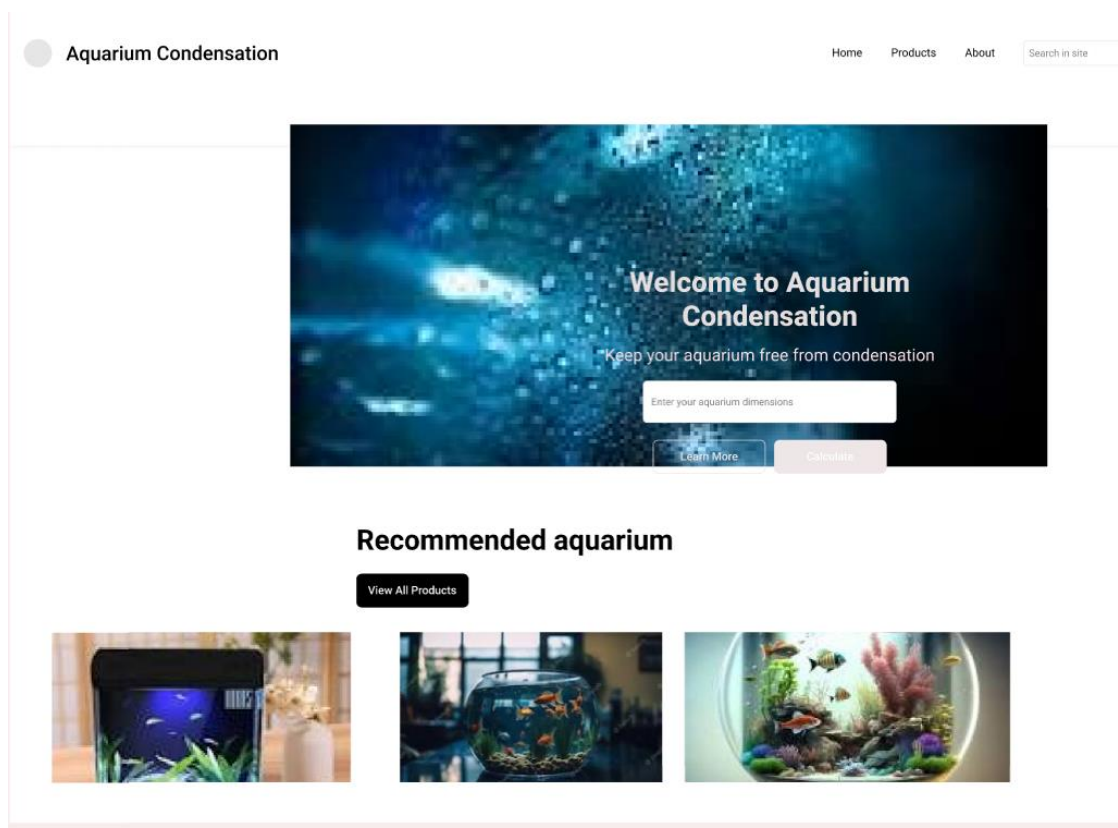
Form Name: Registration Page



Form Name: Aquarium and Petshop Page



Form Name: Aquarium condensation Page



4.4 DATABASE DESIGN

4.4.1 Relational Database Management System (RDBMS)

A database is a system designed to store and facilitate the retrieval and utilization of information.

The integrity and security of the data within a database are of paramount importance.

Creating a database involves two essential steps. Initially, it is crucial to understand the user's requirements and establish a database structure that aligns with their needs. The first step entails devising an organizational plan for the information, which is not reliant on any particular computer program.

Subsequently, this plan is employed to craft a design tailored to the specific computer program that will be employed to construct the database system. This phase is centered on determining how data will be stored within the chosen computer program.

Key aspects of database development include:

- **Data Integrity:** Ensuring the accuracy and consistency of data stored within the database.
- **Data Independence:** The ability to modify the data storage structure without affecting the application's access to the data.

4.4.2 Normalization

A way of showing a database as a group of connections is called a relational model. Every relationship is like a chart of information or a group of data. In the formal way of talking about tables, a row is called a tuple, a column header is called an attribute, and the table is called a relation. A bunch of tables with special names make up a relational database. A row in a story shows a group of things that are connected.

Relations, Domains & Attributes

Tables serve as containers for organized and related information. Within a table, the rows are referred to as tuples, and a tuple is essentially an ordered list containing n items. Columns in a table are synonymous with characteristics, each representing a specific aspect of the data.

In a relational database, tables are interconnected, ensuring coherence and meaningful associations between different sets of data. This relational structure forms the foundation of a well-structured database.

A domain can be thought of as a collection of fundamental values with a shared source or data type.

Naming domains is a helpful practice as it aids in comprehending the significance of the values they

encompass. It's essential to recognize that values within a domain are indivisible.

- Table relationships are established using keys, with the primary key and foreign key being the primary types. Entity Integrity and Referential Integrity are fundamental principles that guide these relationships.
- Entity Integrity mandates that no Primary Key should contain null values, reinforcing the unique and integral nature of primary keys in database relationships.

Normalization is the process of organizing data to ensure its efficient storage and to minimize redundancy while maintaining accuracy and reliability. This approach involves eliminating unnecessary columns and breaking down large tables into smaller, more manageable parts.

Normalization helps prevent errors when adding, removing, or modifying data. In data modeling, two key concepts are integral to its normal form: keys and relationships. Keys serve as unique identifiers to distinguish one row from all others in a table. There are two main types of keys: the primary key, which groups similar records within a table, and the foreign key, which acts as a special code to identify specific records in another table.

1. First Normal Form (1NF): This rule dictates that an attribute can only contain a single value that cannot be further divided. Each value in a tuple must match the attribute's type. In 1NF, tables cannot contain other tables or have tables as part of their data. This form ensures that values are indivisible and represent a single entity. Achieving 1NF often involves organizing data into separate tables, with each table having a designated key based on project requirements. New relationships are established for various data categories or related groups, eliminating redundant information. A relationship adhering to primary key constraints alone is termed the first normal form.

2. Second Normal Form (2NF): In simpler terms, 2NF stipulates that no additional information should be associated with only part of the primary information used for data organization. This involves breaking down the information and creating separate groupings for each part along with its related details. It's essential to maintain a connection between the original primary key and any data dependent on it. This step helps remove data that relies on only a portion of the key. When a set of information has a primary means of identifying each item, and all other

details depend solely on that primary means, it is considered to be in the second normal form.

3. Third Normal Form (3NF): is a critical concept in database normalization, aiming to achieve table independence and control over attributes. In 3NF, a table should not contain columns that depend on other non-key columns, ensuring that each column is functionally dependent only on the primary key. It breaks down relationships involving non-key attributes to eliminate dependencies on attributes not part of the primary key. To meet the criteria for 3NF, data must already be in the Second Normal Form (2NF), and non-key attributes should not rely on other non-key attributes within the same table, avoiding transitive dependencies. This process enhances data integrity, reduces redundancy, and optimizes database structure and organization.

4.4.3 Sanitization

Django incorporates various in-built mechanisms for data sanitization. To begin with, it provides a diverse range of field types that come with automatic validation and sanitization capabilities. For instance, the CharField performs automatic validation and cleaning of text input, while the EmailField ensures that email addresses adhere to the correct format. These field types serve as protective measures to prevent the storage of malicious or invalid data in the database.

Moreover, Django strongly advocates for the utilization of form validation to sanitize user input. Django's forms are equipped with predefined validation methods and validators that can be applied to form fields. These validators execute a range of checks and cleansing operations, such as confirming that numerical values fall within specified ranges or verifying that uploaded files adhere to specific formats. These combined features in Django promote data integrity and security, making it a reliable choice for web application development.

4.4.4 Indexing

The index keeps track of a certain piece of information or group of information, arranged in order of its value. Arranging the index entries helps find things quickly and easily that match exactly or are within a certain range. Indexes make it easy to find information in a database without having to search through every record every time the database is used. An index is like a roadmap for finding information in a database. It helps you look up data quickly and also makes it easy to find records that are in a certain order. It can be based on one or more columns in the table.

4.5 TABLE DESIGN

User Table

Column Name	Data Type	Constraints
UserID	int	Unique, notnull
Username	Varchar (50)	Unique ,notnull
Password	Varchar (50)	notnull
Email	Varchar (50)	Unique

Role Table

Column Name	Data Type	Constraints
RoleID	int	Primary Key, Auto Increment
RoleName	varchar(50)	Unique, notnull

Permission Table

Column Name	Data Type	Constraints
Permissionid	int	Primary Key, Auto Increment
Permissionname	Varchar(20)	Unique,notnull

Aquarium and Pet Table

Column Name	Data Type	Constraints
aqpetid	int	Unique,notnull
categoryid	varchar(50)	Unique,notnull
description	varchar(50)	notnull
price	Decimal(10,2)	Unique

Categories Table

Column Name	Data Type	Constraints
categoryid	int	Primary Key, Auto Increment
categoryname	varchar(50)	notnull

Supplier Table

Column Name	Data Type	Constraints
SupplierID	int	Primary Key, Auto Increment
SupplierName	varchar(100)	Not null
Phno	text	Not null

Inventory Table

Column Name	Data Type	Constraints
Inventoryid	int	Primary Key, Auto Increment
aqpetid	int	Foreign key(aqpetid from aquarium and pet table)
SupplierID	int	Foreign key(supplierid from supplier table)
quantity	int	Not null
purchasedate	date	Not null

Customer Table

Column Name	Data Type	Constraints
customerid	int	Primary Key, Auto Increment
customername	varchar(50)	notnull

Order Table

Column Name	Data Type	Constraints
orderid	int	Primary Key, Auto Increment
customerid	int	Foreignkey, Unique, notnull
orderdate	Not null	notnull
amount	Decimal(10,2)	notnull

Payment Method

Column Name	Data Type	Constraints
methodid	int	Primary Key, Auto Increment
orderid	int	Foreignkey, (ordered in ordertable)Not null
methodname	varchar(100)	Not null

Payment Table

Column Name	Data Type	Constraints
methodid	int	Primary Key, Auto Increment
orderid	int	Foreignkey, (ordered in ordertable)Not null
methodid	int	Foreignkey, (methodid in payment id)Not null

CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

Software testing is an essential procedure employed to verify if a computer program functions as intended. It is conducted to ensure that the software performs its designated tasks accurately and complies with the prescribed requirements and standards. Validation is the process of inspecting and assessing software to confirm its adherence to the specified criteria. Software testing is a method for assessing the performance of a program, often in conjunction with techniques like code inspection and program walkthroughs. Validation ensures that the software aligns with the user's expectations and requirements.

Several principles and objectives guide the process of software testing, including:

1. Testing is the practice of executing a program with the primary aim of identifying errors.
2. An effective test case is one that has a high likelihood of uncovering previously undiscovered errors.
3. A successful test is one that exposes previously undiscovered errors.

When a test case operates effectively and accomplishes its objectives, it can detect flaws within the software. This demonstrates that the computer program is functioning as intended and is performing well. The process of evaluating a computer program encompasses three primary aspects:

1. Correctness assessment
2. Evaluation of implementation efficiency
3. Examination of computational complexity

5.2 TEST PLAN

A test plan serves as a comprehensive set of instructions for conducting various types of tests. It can be likened to a map that outlines the steps to follow when evaluating a computer program. Software developers create instructions for both using and organizing the necessary information for the program's proper functionality. They ensure that each component of the program performs its intended functions. To ensure the thorough testing of the software, a group known as the ITG (Information Technology Group) is responsible for verifying its functionality, instead of relying solely on the software's creators for testing.

The objectives of testing should be clearly defined and measurable. A well-structured test plan should encompass details about the frequency of failures, the associated repair costs, the occurrence

rate of issues, and the time required for complete testing. The levels of testing typically include:

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

5.2.1 Unit Testing

Unit testing checks the smallest part of a software design - the software component or module.

Testing important control paths within a module using the design guide to find errors. This means how difficult the tests are for each small part of a program and what parts of the program haven't been tested yet. Unit testing is a type of testing that looks at how the code works inside and can be done at the same time for different parts of the program.

Before starting any other test, we need to check if the data flows correctly between different parts of the computer program. If the information doesn't move in and out correctly, all other checks are pointless. When designing something, it's important to think about what could go wrong and make a plan for how to deal with those problems. This can mean redirecting the process or stopping it completely.

The Aquarium and Petshop Management System was tested by looking at each part by itself and trying different tests on it.

Some mistakes in the design of the modules were discovered and then fixed. After writing the instructions for different parts, each part is checked and tried out separately. We got rid of extra code and made sure everything works the way it should

5.2.2 Integration Testing

Integration testing is a critical process in software development that involves constructing a program while simultaneously identifying errors in the interaction between different program components. The primary objective is to utilize tested individual parts and assemble them into a program according to the initial plan. This comprehensive testing approach assesses the entire program to ensure its proper and correct functionality.

As issues are identified and resolved during integration testing, it's not uncommon for new problems to surface, leading to an ongoing cycle of testing and refinement. After each individual component

of the system is thoroughly examined, these components are integrated to ensure they function harmoniously. Additionally, efforts are made to standardize all programs to ensure uniformity rather than having disparate versions.

5.2.3 Validation Testing or System Testing

The final phase of testing involves a comprehensive examination of the entire system to ensure the correct interaction of various components, including different types of instructions and building blocks. This testing approach is referred to as Black Box testing or System testing.

Black Box testing is a method employed to determine if the software functions as intended. It assists software engineers in identifying all program issues by employing diverse input types. Black Box testing encompasses the assessment of errors in functions, interfaces, data access, performance, as well as initialization and termination processes. It is a vital technique to verify that the software meets its intended requirements and performs its functions correctly.

5.2.4 Output Testing or User Acceptance Testing

System testing is conducted to assess user satisfaction and alignment with the company's requirements. During the development or update of a computer program, it's essential to maintain a connection with the end-users. This connection is established through the following elements:

- Input Screen Designs.
- Output Screen Designs.

To perform system testing, various types of data are utilized. The preparation of test data plays a crucial role in this phase. Once the test data is gathered, it is used to evaluate the system under investigation. When issues are identified during this testing, they are addressed by following established procedures. Records of these corrections are maintained for future reference and improvement. This process ensures that the system functions effectively and meets the needs of both users and the organization.

5.2.5 Automation Testing

Automated testing is a method employed to verify that software functions correctly and complies with established standards before it is put into official use. This type of testing relies on written instructions that are executed by testing tools. Specifically, UI automation testing involves the use of specialized tools to automate the testing process. Instead of relying on manual interactions where individuals click through the application to ensure its proper functioning, scripts are created to automate these tests for various scenarios. Automating testing is particularly valuable when it is

necessary to conduct the same test across multiple computers simultaneously, streamlining the testing process and ensuring consistency.

5.2.6 Selenium Testing

Selenium is a valuable and free tool designed for automating website testing. It plays a crucial role for web developers as it simplifies the testing process. Selenium automation testing refers to the practice of using Selenium for this purpose. Selenium isn't just a single tool; it's a collection of tools, each serving distinct functions in the realm of automation testing. Manual testing is a necessary aspect of application development, but it can be monotonous and repetitive. To alleviate these challenges, Jason Huggins, an employee at ThoughtWorks, devised a method for automating testing procedures, replacing manual tasks. He initially created a tool named the JavaScriptTestRunner to facilitate automated website testing, and in 2004, it was rebranded as Selenium.

Example:

Test Case 1- Login

Code

```
package Stepdefinition;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.When;
import io.cucumber.java.en.Then;
import static org.junit.Assert.assertEquals;
public class login {

    WebDriver driver = new FirefoxDriver();
    @Given("browser is open")
    public void openBrowser() {
    }

    @Given("user is on login page")
    public void navigateToLoginPage() {
        driver.get("http://127.0.0.1:8000/accounts/login/");
    }

    @When("user enters username and password")
    public void enterCredentials() {
        WebElement usernameInput = driver.findElement(By.id("username"));
```

```

WebElement passwordInput = driver.findElement(By.id("password"));

// Enter valid username and password
usernameInput.sendKeys("Roshangeorge");
passwordInput.sendKeys("Roshan@2k66");
}

```

Test Report

Test Report-

Test Case 1					
Project Name: Bag Emporium					
Login Test Case					
Test Case ID: 1			Test Designed By: Roshan George		
Test Priority(Low/Medium/High):High			Test Designed Date: 04-12-2023		
Module Name: Login Screen			Test Executed By: Mr. Jetty Benjamin		
Test Title: User Login			Test Execution Date: 04-12-2023		
Description: Verify login with valid email and password					
Pre-Condition: User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigation toLogin Page		Dashboard shouldbe displayed	Login page displayed	Pass
2	Provide valid username	Username: Roshangeorge	User should beable to login	User Logged in andnavigated to User Dashboard	Pass
3	Provide Valid Password	Password: Roshan@2k66			
4	Click on Login button				
Post-Condition: User is validated with database and successfully login into account. The Accountsession details are logged in database					

Test Case 2: Registerarion

Code

```
package Stepdefinition;

import io.cucumber.java.en.Given;
import io.cucumber.java.en.When;
import io.cucumber.java.en.Then;
import io.cucumber.java.After;
import io.cucumber.java.en.And;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import static org.junit.Assert.assertEquals;

public class registration {

    WebDriver driver = new FirefoxDriver();

    @Given("browser is open")
    public void openBrowser() {
    }

    @Given("user is on registration page")
    public void navigateToRegistrationPage() {
        driver.get("http://127.0.0.1:8000/register/");
    }

    @When("user enters valid registration information")
    public void enterRegistrationInformation() {
        // Implement logic to fill in the registration form with valid information
        WebElement fullNameInput = driver.findElement(By.id("full-name"));
        WebElement usernameInput = driver.findElement(By.id("username"));
        WebElement roleSelect = driver.findElement(By.id("role"));
        WebElement emailInput = driver.findElement(By.id("email"));
        WebElement passwordInput = driver.findElement(By.id("password"));
        WebElement confirmPasswordInput = driver.findElement(By.id("confirm-
password"));
```

```
        fullNameInput.sendKeys("Edwin Raju");
        usernameInput.sendKeys("edwin");
        roleSelect.sendKeys("Customer");
        emailInput.sendKeys("edwinraju@gmail.com");
        passwordInput.sendKeys("Edwin@2k66");
        confirmPasswordInput.sendKeys("E@2k66");
    }

    @And("user clicks on create account")
    public void clickCreateAccount() {
        WebElement createAccountButton =
driver.findElement(By.cssSelector(".btn.btn-primary.btn-block"));
        createAccountButton.click();
    }

    @Then("user is successfully registered and navigated to the home page")
    public void verifyRegistration() {
        // Wait for the registration to complete (you might need to use WebDriverWait for
a real-world scenario)
        try {
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        // Add verifications for successful registration
        String currentUrl = driver.getCurrentUrl();
        assertEquals("http://127.0.0.1:8000/accounts/login/", currentUrl);
    }

    // After hook to close the browser
    @After
    public void tearDown() {
        if (driver != null) {
```

```
        driver.quit();
    }
}
}
```

Screenshot

```
Scenario: Register a new account with valid information # src/test/resources/Features/register.feature:3
  Given browser is open # Stepdefinition.registration.openBrowser()
  And user is on registration page # Stepdefinition.registration.navigateToRegistrationPage()
  When user enters valid registration information # Stepdefinition.registration.enterRegistrationInfo()
  And user clicks on create account # Stepdefinition.registration.clickCreateAccount()
  Then user is successfully registered and navigated to the home page # Stepdefinition.registration.verifyRegistration()

1 Scenarios (1 passed)
5 Steps (5 passed)
0m9.705s
```

Test report

Test Case 2					
Project Name:Aquarium and Petshop Management System					
Registration Test Case					
Test Case ID: Test_2			Test Designed By: Roshan George		
Test Priority(Low/Medium/High):High			Test Designed Date: 04-12-2023		
Module Name: User Registration			Test Executed By : Ms. Jetty Benjamin		
Test Title : Registration			Test Execution Date: 04-12-2023		
Description: Verify Registration with valid details					
Pre-Condition :User has valid details for Registration					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigation to Registration Page		Dashboard should be displayed	Registration pagedisplayed	Pass
2	Provide	Fullname: Roshan George	User should be able to Register	User Registerd in and navigated to Login Dashboard	Pass
3	Provide valid Username	Username: roshan2k66george			
4	Provide valid Role	Role: customer			
5	Provide valid Email address	Email:	User should beable to Register	User Registerd in and navigated to Login Dashboard	Pass
6	Provide valid Password	Password: George@2k66			
7	Click on Login button				

Post-Condition: User is validated with database and successfully Registered into account.
The Accountsession details are Registered in database

Test Case 3: Dealer Activation

Code

package Stepdefinition;

```
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
```

```
import static org.junit.Assert.assertEquals;
```

```
public class activation {
```

```
    WebDriver driver = new FirefoxDriver();
```

```
    @Given("browser is open")
```

```
    public void openBrowser() {
```

```
        // Initialization of the browser, you may also want to set up the WebDriver in a
```

Before hook

```
    }
```

```
    @Given("user is on the dealer page")
```

```
    public void navigateToDealerPage() {
```

```
        driver.get("http://127.0.0.1:8000/admin_dashboard/dealers.html");
```

```
        // Add logic to navigate to the dealer page
```

```
    }
```

```
    @Then("dealer page should have a sidebar, header, and a table with user accounts")
```

```
    public void verifyDealerPageElements() {
```

```
// Add assertions to verify the presence of sidebar, header, and table elements on  
the dealer page
```

```
}
```

```
@When("user activates a dealer")
```

```
public void activateDealer() {
```

```
    // Add logic to activate a dealer
```

```
}
```

```
@Then("the dealer should be activated")
```

```
public void verifyDealerActivation() {
```

```
    // Add assertions to verify that the dealer is activated
```

```
}
```

```
@When("user deactivates a dealer")
```

```
public void deactivateDealer() {
```

```
    // Add logic to deactivate a dealer
```

```
}
```

```
@Then("the dealer should be deactivated")
```

```
public void verifyDealerDeactivation() {
```

```
    // Add assertions to verify that the dealer is deactivated
```

```
}
```

```
}
```

Screenshot

```
Scenario: Verify Dealer Page Elements # src/test/resources/Features/activation
  Given browser is open # Stepdefinition.activation.openBrowser()
  And user is on the dealer page # Stepdefinition.activation.navigateToDe
  Then dealer page should have a sidebar, header, and a table with user accounts # Stepdefinition.activation.verifyDealer

Scenario: Verify Dealer Activation and Deactivation # src/test/resources/Features/activation.feature:8
  Given browser is open # Stepdefinition.activation.openBrowser()
  And user is on the dealer page # Stepdefinition.activation.navigateToDealerPage()
  When user activates a dealer # Stepdefinition.activation.activateDealer()
  Then the dealer should be activated # Stepdefinition.activation.verifyDealerActivation()
  And user deactivates a dealer # Stepdefinition.activation.deactivateDealer()
  Then the dealer should be deactivated # Stepdefinition.activation.verifyDealerDeactivation()

2 Scenarios (2 passed)
9 Steps (9 passed)
0m0.331s
```

Test report

Test Case 3					
Project Name:Aquarium and Petshop Management System					
Activation Test Case					
Test Case ID: Test_3			Test Designed By: Roshan George		
Test Priority(Low/Medium/High):High			Test Designed Date: 05-12-2023		
Module Name: Dealer Activation and Deactivation			Test Executed By :Ms. Jetty Benjamin		
Test Title : Dealer Activation and Deactivation			Test Execution Date: 04-12-2023		
Description: Disabling or terminating a dealer's access or operational status.			To check and validate the Activation and Deactivation		
Pre-Condition :Dealer should be a valid User					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigation to Admin's Customer view				
2	Provide Username	Username: rijul	User should be able to Activate and Deactivate	User is Activated or Deactivated in and navigated to corresponding Page	Pass
3	Provide Fullname	Fullname: Rijul Rojio			
4	Provide Email	Email: rijulrojio2024b@mca.ajce.in			
5	Provide Action Activation and Deactivation	Action: Activation Or Deactivation	User should be able to Activate and Deactivate	User is Activated or Deactivated in and navigated to corresponding Page	Pass
6	Click on Activation or Deactivation				
Post-Condition: The user successfully logs in, creates a new post, and publishes it on the platform.					

CHAPTER 6

IMPLEMENTATION

6.1 INTRODUCTION

Implementation is the stage where a planned system transforms into a tangible and operational entity. Building trust and instilling confidence in users is of paramount importance for the success of the system. This is a critical phase that places a strong emphasis on user training and the creation of informative materials. The actual transition often occurs during or after user training.

Implementation signifies the act of putting a new system into action after its design phase, turning a conceptual design into a functional one.

Implementation involves the process of transitioning from the old method of operation to the new one, whether it replaces the old system entirely or makes incremental changes. Ensuring that the process is executed accurately is vital to establish a system that aligns well with the organization's requirements. System implementation encompasses the following tasks:

- Meticulous planning.
- Assessment of the existing system and its constraints.
- Designing methods to facilitate the transition.

6.2 IMPLEMENTATION PROCEDURES

Software implementation involves the installation of software in its intended location and ensuring that it functions as intended. In some organizations, an individual who is not directly involved in using the software may oversee and approve the project's development. Initially, there may be some skepticism regarding the software, and it's essential to address these concerns to prevent strong resistance.

To ensure a successful transition, several key steps are important:

- Communicating Benefits: Users need to understand why the new software is an improvement over the old one, instilling trust in its capabilities.
- User Training: Providing training to users is crucial to make them feel confident and comfortable using the application.

To evaluate the outcome, it is essential to verify that the server program is running on the server. If the server is not operational, the expected outcomes will not be achieved.

6.2.1 User Training

User training is a critical component of ensuring that individuals know how to use and adapt to a new system. It plays a vital role in fostering user comfort and confidence with the system. Even when a system becomes more complex, the need for training becomes even more apparent. User training covers various aspects, including inputting information, error handling, querying databases, and utilizing tools for generating reports and performing essential tasks. It aims to empower individuals with the knowledge and skills needed to effectively interact with the system.

6.2.2 Training on the Application Software

Once the fundamental computer skills have been covered, the next step is to instruct individuals on using a new software program. This training should provide a comprehensive understanding of the new system, including navigation through screens, accessing help resources, error management, and resolution procedures. The training aims to equip users or groups with the knowledge and skills necessary to effectively utilize the system or its components. It's important to note that training may be tailored differently for various groups of users and individuals in different roles within the organization to cater to their specific needs and requirements.

6.2.3 System Maintenance

Maintaining a system's functionality is a complex challenge in software development. In the maintenance phase of the software life cycle, the software performs critical functions and operates smoothly. Once a system is operational, it requires ongoing care and maintenance to ensure its continued optimal performance. Software maintenance is a crucial aspect of the development process as it ensures that the system can adapt to changes in its environment. Maintenance involves more than just identifying and correcting errors in the code. It encompasses various tasks that contribute to the system's stability and effectiveness.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The Aquarium and Pet Shop Management System represents an innovative and comprehensive solution for the efficient management of virtual aquariums and pet shops. With a robust technology foundation, including the use of Python Django for backend development and HTML/CSS for frontend design, the system addresses critical aspects of user roles, inventory management, and security. The proposed enhancements, from distinct user roles to advanced search functionalities and integration of modern technologies like machine learning, promise to elevate the user experience and overall system performance. The economic feasibility underscores potential cost savings, revenue generation, and market competitiveness, while the technical feasibility assures compatibility, scalability, and adaptability. Furthermore, behavioral feasibility emphasizes the importance of user acceptance, organizational alignment, and effective change management. As the project advances, the combination of these factors positions the Aquarium and Pet Shop Management System as a holistic and forward-thinking solution, poised to revolutionize the virtual pet management landscape while ensuring sustainable success and user satisfaction.

.

7.2 FUTURE SCOPE

The future scope for the Aquarium and Pet Shop Management System holds immense potential for growth and enrichment. The development of dedicated mobile applications for various platforms could usher in greater accessibility and convenience for users, enabling them to manage virtual aquariums and pet shops seamlessly on mobile devices. Additionally, the integration of augmented reality (AR) could transform the user experience, allowing customers to virtually interact with pets and aquariums before making purchase decisions. Embracing Internet of Things (IoT) technology may lead to the creation of smart aquariums, with real-time monitoring of water quality and temperature parameters. Blockchain technology could be explored for enhanced security and transparency in transactions and data management. Further advancements in machine learning algorithms can personalize product recommendations, while social media integration could foster user interaction and community engagement. Initiatives promoting environmental sustainability and responsible pet ownership could be incorporated, educating users about eco-friendly practices. Advanced analytics, predictive modeling, and the expansion of virtual pet categories, including exotic species, can diversify the offering and attract a broader audience. Finally, the system could evolve into a global collaboration platform, fostering connections among pet enthusiasts and

potentially establishing a virtual marketplace for pet-related products and services. By continuously innovating in these areas, the Aquarium and Pet Shop Management System is poised for sustained growth and relevance in the evolving landscape of pet-related technologies.

.

CHAPTER 8

BIBLIOGRAPHY

REFERENCES:

- chatgpt..
- online platforms like w3schools, MDN web docs
- youtube
- pet industry reports
- online course on python

WEBSITES:

- Django documentation
- Web development resouces like w3schools
- Books like Django for beginners by william s.vincent

CHAPTER 9

APPENDIX

9.1 Sample Code

Login Page

```
{% load static %}

<!DOCTYPE html>

<html lang="en">

    <head>

        <meta charset="utf-8">

        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">

        <meta name="description" content="">

        <meta name="author" content="">

        <link
href="https://fonts.googleapis.com/css?family=Poppins:100,200,300,400,500,600,700,800,900&d
isplay=swap" rel="stylesheet">

        <title>login</title>

        <!-- Additional CSS Files -->

        <link rel="stylesheet" type="text/css" href="assets/css/bootstrap.min.css">

        <link rel="stylesheet" type="text/css" href="assets/css/font-awesome.css">

        <link rel="stylesheet" href="assets/css/templatemo-hexashop.css">

        <link rel="stylesheet" href="assets/css/owl-carousel.css">

        <link rel="stylesheet" href="assets/css/lightbox.css">

    <!--
```

TemplateMo 571 Hexashop

<https://templatemo.com/tm-571-hexashop>

```
-->
</head>

<body>

<!-- ***** Preloader Start ***** -->
<div id="preloader">
  <div class="jumper">
    <div></div>
    <div></div>
    <div></div>
  </div>
</div>
<!-- ***** Preloader End ***** -->


<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login</title>

  <!-- Link to Bootstrap CSS -->
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

  <!-- Include your custom CSS for additional styling -->
  <link rel="stylesheet" href="styles.css">

  <style>
```

```
/* Custom styles for the login form */
body {
    font-family: Arial, sans-serif;
    background-image: url({'% static "/aapmapp/images/dog1.jpg" %}');
    background-size: cover;
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-color: #f0f0f0;
    color: #333;
}

.container {
    max-width: 400px;
    margin: 0 auto;
    padding: 20px;
    text-align: center;
    background-color: #fff; /* Set a white background for the form container */
    border: 1px solid #ddd;
    border-radius: 5px;
    color: #000; /* Set text color to black within the form */
}

h2 {
    font-size: 24px;
    margin-bottom: 20px;
}

.form-group {
    margin-bottom: 15px;
}

label {
    font-weight: bold;
}
```

```
button[type="submit"] {
    background-color: #FF9900;
    color: #fff;
    border: none;
    padding: 10px 20px;
    border-radius: 5px;
    cursor: pointer;
}

button[type="submit"]:hover {
    background-color: #FF8000;
}

a {
    color: #007BFF;
    text-decoration: none;
}

a:hover {
    text-decoration: underline;
}
</style>
</head>
<body><br><br><br>
<div class="container">
    <form class="login-form" action=" " method="post">
        { % csrf_token % }
        <h2>Login</h2>
        <div class="form-group">
            <label for="username">Username</label>
            <input type="text" id="username" name="username" class="form-control"
placeholder="Enter your username" required>
        </div>
```



```
<div class="form-group">
  <label for="password">Password</label>
  <input type="password" id="password" name="password" class="form-control"
placeholder="Enter your password" required>
</div>

<div class="form-group">
  <button type="submit" class="btn btn-primary">Log In</button>
</div>
<div class="form-group">
  <a href="{ % url 'password_reset' % }">Forgot Password?</a>
</div>
<div class="form-group">
  <p>Don't have an account? <a href="{ % url 'register' % }">Sign up</a></p>
</div>
</form>
</div>

<!-- Include Bootstrap JS scripts -->
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.3/dist/umd/popper.min.js"></script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>
</html>

<!-- jQuery -->
<script src="assets/js/jquery-2.1.0.min.js"></script>

<!-- Bootstrap -->
```

```
<script src="assets/js/popper.js"></script>
<script src="assets/js/bootstrap.min.js"></script>

<!-- Plugins -->
<script src="assets/js/owl-carousel.js"></script>
<script src="assets/js/accordions.js"></script>
<script src="assets/js/datepicker.js"></script>
<script src="assets/js/scrollreveal.min.js"></script>
<script src="assets/js/waypoints.min.js"></script>
<script src="assets/js/jquery.counterup.min.js"></script>
<script src="assets/js/imgfix.min.js"></script>
<script src="assets/js/slick.js"></script>
<script src="assets/js/lightbox.js"></script>
<script src="assets/js/isotope.js"></script>

<!-- Global Init -->
<script src="assets/js/custom.js"></script>

<script>

$(function() {
    var selectedClass = "";
    $("p").click(function(){
        selectedClass = $(this).attr("data-rel");
        $("#portfolio").fadeOut(50, 0.1);
        $("#portfolio div").not("." + selectedClass).fadeOut();
        setTimeout(function() {
            $("." + selectedClass).fadeIn();
            $("#portfolio").fadeOut(50, 1);
        }, 500);

    });
});
```

</script>

</body>

</html>

Views function

```
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth import login, authenticate, logout
from .models import dealer
from django.contrib import messages
from django.contrib.auth.decorators import login_required
from django.contrib.auth import authenticate, login
from django.db.models import Q # Import Q for complex queries
from django.http import JsonResponse
from django.http import Http404
from .models import UserProfile, CartItem
from django.shortcuts import render, redirect
from django.contrib.auth.models import User
from django.core.mail import send_mail
from django.template.loader import render_to_string
from django.utils.html import strip_tags
from django.http import HttpResponse
from .urls import *
from django.views.decorators.csrf import csrf_exempt

from django.shortcuts import render
from django.contrib.auth.decorators import login_required
from django.views.decorators.cache import never_cache
def homelogin(request):
    return render(request, 'homelogin.html')
def register(request):
    if request.method == 'POST':
        fullname = request.POST['fullname']
        username = request.POST['username']
```

```
role = request.POST['role']
email = request.POST['email']
password = request.POST['password']
confirm_password = request.POST['confirm_password']

if password != confirm_password:
    return render(request, 'register.html', {'error_message': 'Passwords do not match'})

# Create a user instance but do not save it yet
user = dealer(username=username, email=email,role=role ,fullname=fullname,)

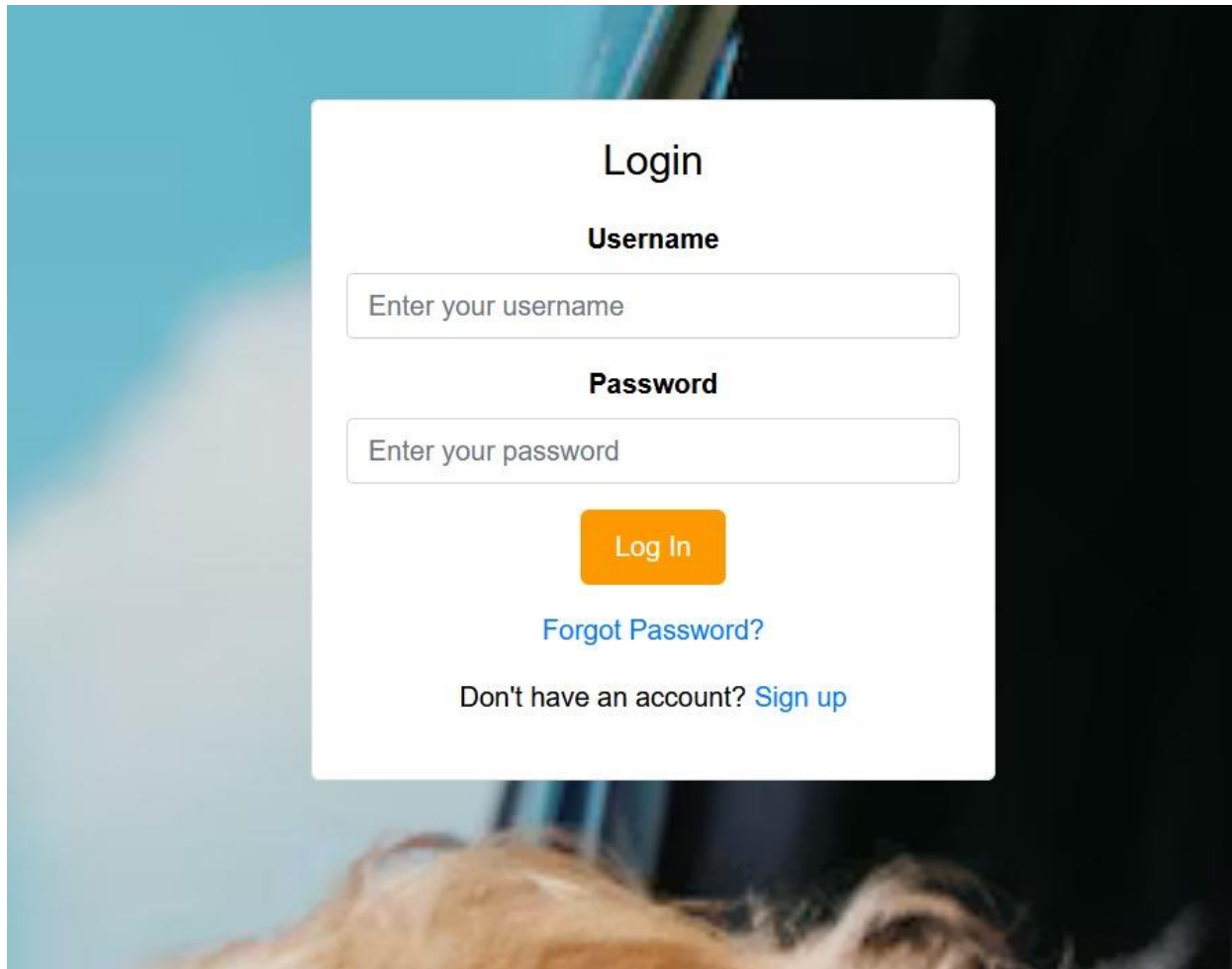
# Set the password for the user
user.set_password(password)

# Save the user to the database
user.save()

# Redirect to the home page or any desired page
return redirect('login')

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

9.1 Screen Shots



Create a New Account

Full Name

Username

Role

Email Address

Password

Confirm Password

Dashboard
User Accounts
Customers
Dealers
Delivery Man
View Aquarium and Pets

Admin Dashboard

Search:

Category	Breed	Pet Age	Price	Location	Image
cat	None	None	Rs1500.00	ernakulam	
fish	None	None	Rs30.00	kottayam	
dog	None	None	Rs5000.00	ernakulam	
fish	None	None	Rs400.00	ernakulam	

Dealer Accounts

Filter by Username

Filter

Username	Fullname	Email	Actions
Jeeva	jeevarag np	jeevargnr01@gmail.com	Deactivate
roshan_2k	Roshan George	roshangeorge2k66@gmail.com	Activate
rijul	Rijul Rojo	rijulrojo2024b@mca.ajce.in	Deactivate
Ashik	Muhamed Ashik	muhammedashik@gmail.com	Deactivate

Pets

"Explore our diverse selection of loving companions, each waiting to bring joy, comfort, and laughter into your life as a cherished member of your family.."



cat
Breed: None
Rs1500.00



fish
Breed: None
Rs30.00







dog
Breed: None
Rs5000.00



fish
Breed: None
Rs400.00



Item Name	Quantity	Price	Description	Location	Image	Buy now
	5	Rs				Buy Now
normal	1	Rs3500.00	None	kozhikode		Buy Now
None	1	Rs5000.00	None	ernakulam		Buy Now
Common Dog	1	Rs250.00	normal dog	ernakulam		Buy Now
None	1	Rs30.00	None	kottayam		Buy Now