

ASSISTIVEGLOBE

Mini Project Report

Submitted by

SRUTHYMOL SUNNY

Reg. No.:AJC19MCA-I056

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, “**ASSISTIVEGLOBE**” is the bona fide work of **SRUTHYMOL SUNNY (Regno: AJC19MCA-I056)** 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.

Ms. Rini Kurian

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 “**ASSISTIVEGLOBE**” 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:

SRUTHYMOL SUNNY

KANJIRAPPALLY

Reg: AJC19MCA-I056

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 **Coordinator Name** 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 **Guide Name** 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.

SRUTHYMOL SUNNY

ABSTRACT

Assistive aids, also known as assistive devices or adaptive equipment, are tools, devices, or technologies designed to assist individuals with disabilities in performing tasks or activities they may find challenging or impossible to do on their own. These aids are crucial in enhancing the independence, mobility, and overall quality of life for people with disabilities. They encompass a wide range of products, from simple items like canes and hearing aids to more complex technologies such as screen readers and mobility scooters. Assistive aids are tailored to address various types of disabilities, including mobility impairments, sensory impairments, cognitive impairments, and more. By providing practical solutions and support, assistive aids play a vital role in promoting inclusivity and enabling individuals with disabilities to actively participate in daily life.

This project is mainly divided into four modules:

- Admin
- User
- Mentor
- Delivery Team

AssistiveGlobe is an inclusive e-commerce platform dedicated to enhancing accessibility for individuals with disabilities.

The User Module facilitates seamless interaction, allowing users to register, log in, and access the platform. They can add or remove products from their shopping cart, write reviews, conduct searches, and view ratings from other customers. A wish list feature enables users to save products for later, and secure payments can be made through a gateway. Order tracking, shipping information.

The Admin Module provides administrative privileges for platform management, including overseeing users, the delivery team, and mentors. Admins can edit product details, monitor orders, and manage stock levels. They also have the capability to assign delivery agents, streamlining order fulfillment.

The Assistive Aid Consultation Module offers expert guidance to users in selecting the most suitable assistive aids, with options for live chat and email consultations, along with the generation of comprehensive reports.

The Delivery Team Module optimizes the order fulfillment process, allowing team members to access detailed order information and update delivery status.

By seamlessly integrating these modules, AssistiveGlobe empowers disabled individuals with the tools they need for independence, contributing to a more inclusive society. Through user-friendly interfaces, personalized recommendations, and expert consultation services, this platform fosters a more inclusive and enriching environment for people with disabilities.

CONTENT

SL. NO	TOPIC	PAGE NO
1	INTRODUCTION	
1.1	PROJECT OVERVIEW	
1.2	PROJECT SPECIFICATION	
2	SYSTEM STUDY	
2.1	INTRODUCTION	
2.2	EXISTING SYSTEM	
2.3	DRAWBACKS OF EXISTING SYSTEM	
2.4	PROPOSED SYSTEM	
2.5	ADVANTAGES OF PROPOSED SYSTEM	
3	REQUIREMENT ANALYSIS	
3.1	FEASIBILITY STUDY	
3.1.1	ECONOMICAL FEASIBILITY	
3.1.2	TECHNICAL FEASIBILITY	
3.1.3	BEHAVIORAL FEASIBILITY	
3.1.4	FEASIBILITY STUDY QUESTIONNAIRE	
3.2	SYSTEM SPECIFICATION	
3.2.1	HARDWARE SPECIFICATION	
3.2.2	SOFTWARE SPECIFICATION	
3.3	SOFTWARE DESCRIPTION	
3.3.1	PHP	
3.3.2	MYSQL	
4	SYSTEM DESIGN	
4.1	INTRODUCTION	
4.2	UML DIAGRAM	
4.2.1	USE CASE DIAGRAM	
4.2.2	SEQUENCE DIAGRAM	
4.2.3	STATE CHART DIAGRAM	
4.2.4	ACTIVITY DIAGRAM	
4.2.5	CLASS DIAGRAM	
4.2.6	OBJECT DIAGRAM	
4.2.7	COMPONENT DIAGRAM	

4.2.8	DEPLOYMENT DIAGRAM	
4.2.9	COLLABORATION DIAGRAM	
4.3	USER INTERFACE DESIGN USING FIGMA	
4.4	DATABASE DESIGN	
5	SYSTEM TESTING	
5.1	INTRODUCTION	
5.2	TEST PLAN	
5.2.1	UNIT TESTING	
5.2.2	INTEGRATION TESTING	
5.2.3	VALIDATION TESTING	
5.2.4	USER ACCEPTANCE TESTING	
5.2.5	AUTOMATION TESTING	
5.2.6	SELENIUM TESTING	
6	IMPLEMENTATION	
6.1	INTRODUCTION	
6.2	IMPLEMENTATION PROCEDURE	
6.2.1	USER TRAINING	
6.2.2	TRAINING ON APPLICATION SOFTWARE	
6.2.3	SYSTEM MAINTENANCE	
7	CONCLUSION & FUTURE SCOPE	
7.1	CONCLUSION	
7.2	FUTURE SCOPE	
8	BIBLIOGRAPHY	
9	APPENDIX	
9.1	SAMPLE CODE	
9.2	SCREEN SHOTS	

List of Abbreviation

IDE – Integrated Development Environment

HTML – Hyper Text Markup Language

CSS – Cascading Style Sheet

JS – JavaScript

AJAX – Asynchronous JavaScript and XML Environment

SQL – Structured Query Language

UML – Unified Modelling Language

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

The AssistiveGlobe project is an ambitious e-commerce platform designed to revolutionize accessibility and availability of assistive aids for individuals with disabilities. This visionary initiative seeks to bridge the gap between disabled users and the essential tools they require for a more independent and enriched lifestyle. Leveraging the power of e-commerce, AssistiveGlobe aims to create an inclusive and user-centric marketplace, offering a diverse range of high-quality assistive aids tailored to the unique needs of disabled individuals. The platform is equipped with four integrated modules: User, Admin, Assistive Aid Consultation, and Delivery Team. The User Module facilitates seamless interaction, allowing users to register, log in, and access the platform. They can manage their shopping cart, write reviews, conduct searches, and benefit from personalized product recommendations driven by machine learning. The Admin Module empowers platform administrators with privileges to oversee users, the delivery team, and mentors, ensuring accurate and up-to-date product information. The Assistive Aid Consultation Module provides expert guidance, enabling users to select the most suitable assistive aids for their unique situations, supported by communication channels like live chat and email consultations. The Delivery Team Module streamlines order fulfillment, allowing team members to efficiently manage and track deliveries. Through the seamless integration of these modules, AssistiveGlobe aspires to empower disabled individuals with the tools and resources they need for a more independent and fulfilling life. This platform, with its user-friendly interfaces, personalized recommendations, and expert consultation services, represents a significant step towards creating a more inclusive society for people with disabilities.

2.1 PROJECT SPECIFICATION

This is a website in which user can buy different kinds of assistive aids based on their needs.

The system consists of 4 modules. They are:

1. User :

- Registration and Login
- User can add, remove product from the cart.
- User can write review for the purchased products.
- Searching the product.
- And, can view the review and ratings of the product.
- Users can add products to their wish list or save them for later.
- Users can add products to their shopping cart and proceed to checkout and make payment through gateway.
- Users can track the status of their orders.
- The platform provides information about shipping options and delivery.

2. Admin:

- Registration and Login.
- Manage users, delivery team, mentor.
- Admin can remove and edit the product details.
- Admin can view the orders that purchased by the customer.
- Keeping track of product availability and stock levels.
- Assign delivery agents to specific orders for delivery fulfilment.

3. Assistive Aid Consultation:

- Register and Login.
- Assisting users in understanding the different types of assistive aids available and helping them select the most suitable options for their unique situations.
- Offer communication channels through which users can request consultation services. These channels may include live chat, email.
- Generating consultation report.

4. Delivery Team:

- Registration and login.
- Displaying order details, including delivery address, contact information, and order items.
- Order status. Delivery team can update whether the order is delivered or not.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

System study is a crucial phase in the field of software engineering that plays a pivotal role in the successful development and implementation of software projects. It is the initial step where comprehensive analysis and evaluation of the existing system, or the proposed system, is conducted. This phase aims to understand the objectives, functions, and requirements of the software under consideration. Through systematic data collection, analysis, and interpretation, system study provides a solid foundation for making informed decisions about the design, development, and deployment of the software solution. It involves studying the current processes, identifying areas for improvement, and defining clear objectives for the upcoming software project. System study is an indispensable aspect of the software development life cycle, serving as a blueprint for subsequent phases and ensuring that the final software product aligns effectively with the needs and expectations of stakeholders. This phase sets the stage for a well-defined and successful software engineering endeavor, ultimately leading to the creation of a robust and reliable software system.

2.2 EXISTING SYSTEM

The existing system for AssistiveGlobe is primarily characterized by a lack of a dedicated platform that comprehensively addresses the needs of disabled individuals for assistive aids. Currently, the accessibility and availability of such aids are limited, leading to significant challenges for users in acquiring the necessary tools for an independent lifestyle. Traditional methods of procurement often involve visiting physical stores or relying on a limited selection of aids available online, which may not cater to the diverse needs of disabled individuals. Additionally, there is typically a lack of expert guidance and consultation services to assist users in making informed decisions about which assistive aids are most suitable for their unique situations.

Furthermore, the existing system lacks the integration of modules like the User, Admin, Assistive Aid Consultation, and Delivery Team, which are crucial for ensuring a seamless and user-centric experience. Without these functionalities, users may face difficulties in navigating and interacting with the platform, and administrators may have limited control and oversight over the system's operations.

Overall, the current state of assistive aid procurement for disabled individuals falls short in terms of accessibility, variety, expert guidance, and user-friendliness. This underscores the critical need for the innovative AssistiveGlobe platform, which aims to bridge these gaps and provide a comprehensive solution for the disabled community. Through the integration of advanced technologies and user-centric features, AssistiveGlobe aspires to revolutionize the way assistive aids are accessed and acquired, ultimately empowering disabled individuals with the tools they need for a more independent and enriched lifestyle.

2.2.1 NATURAL SYSTEM STUDIED

The AssistiveGlobe project can be likened to a natural ecosystem, with distinct modules mirroring key ecological components. The User Module acts as consumers, engaging with and benefiting from the platform's services. Admins function as regulators, overseeing and managing various aspects to maintain stability. The Assistive Aid Consultation Module provides specialized expertise, enhancing user adaptation. The Delivery Team Module ensures efficient resource flow, much like transportation in a natural environment. These interdependent modules collectively contribute to the project's success, mirroring the intricate relationships found within a natural ecosystem.

2.2.2 DESIGNED SYSTEM STUDIED

The designed system for AssistiveGlobe is a comprehensive e-commerce platform tailored to the specific needs of disabled individuals seeking assistive aids. This platform revolutionizes accessibility by offering a user-friendly interface for seamless interaction. Users can easily register, log in, and access the platform. They have the capability to manage their shopping cart, write reviews, conduct searches, and benefit from personalized product recommendations driven by machine learning. Admins possess privileged access to oversee users, the delivery team, and product information, ensuring accuracy and up-to-date details. The Assistive Aid Consultation Module offers expert guidance through live chat and email consultations, along with generating detailed consultation reports. The Delivery Team Module optimizes order fulfillment, enabling efficient delivery management. This integrated system empowers disabled individuals, providing them with the necessary tools for a more independent and enriched lifestyle, ultimately contributing to a more inclusive society.

2.3 DRAWBACKS OF EXISTING SYSTEM

- **Limited Accessibility:** The current system often relies on physical stores or a limited selection of online options. This can be especially challenging for disabled individuals who may face mobility issues or live in areas with limited access to specialized stores.
- **Lack of Variety:** The range of available assistive aids may be limited, potentially leading to difficulties in finding specific or specialized items that cater to individual needs.
- **Insufficient Expert Guidance:** The absence of a dedicated consultation service means users

may struggle to make informed decisions about which assistive aids are best suited to their unique circumstances.

- **Inefficient Procurement Process:** Traditional methods may involve time-consuming and potentially frustrating processes, such as visiting multiple stores or relying on fragmented online resources.

2.4 PROPOSED SYSTEM

The proposed AssistiveGlobe system is a cutting-edge e-commerce platform designed to revolutionize access to assistive aids for individuals with disabilities. It integrates four specialized modules, including User, Admin, Assistive Aid Consultation, and Delivery Team, to create a seamless and user-centric experience. The User Module enables easy registration, personalized product recommendations, and efficient shopping cart management. The Admin Module empowers platform administrators with oversight and product management capabilities. The Consultation Module offers expert guidance, while the Delivery Team Module streamlines order fulfillment. This comprehensive system aims to empower disabled individuals with the tools they need for a more independent and enriched lifestyle, contributing to a more inclusive society.

2.5 ADVANTAGES OF PROPOSED SYSTEM

- **Enhanced Accessibility:** The platform provides a user-friendly interface accessible to individuals with disabilities, ensuring a seamless shopping experience tailored to their specific needs.
- **Diverse Product Range:** With a wide array of high-quality assistive aids available, users have access to a comprehensive selection of products that cater to their unique requirements.
- **Expert Consultation Services:** The inclusion of the Assistive Aid Consultation Module offers specialized guidance, allowing users to make informed decisions about which aids are best suited to their individual situations.
- **Efficient Procurement Process:** Through intuitive features like a user-friendly cart system and secure payment gateways, the platform streamlines the process of selecting and purchasing assistive aids.
- **Accurate Stock Management:** Admins can monitor and manage product availability and stock levels in real-time, ensuring that users have access to the aids they need without encountering inventory shortages.

- **Empowerment of Disabled Individuals:** By providing a dedicated platform that caters to their specific needs, the system empowers disabled individuals with the tools and resources they require for a more independent and enriched lifestyle.
- **Contribution to Inclusivity:** Overall, the AssistiveGlobe system contributes to a more inclusive society by significantly improving the accessibility and availability of assistive aids, thereby enhancing the quality of life for individuals with disabilities.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

The proposed Inclusive E-Commerce Platform for Assistive Aids presents a visionary project with the aim of transforming the accessibility and availability of assistive aids for disabled users. Through the power of e-commerce, the platform seeks to bridge the gap between individuals with disabilities and the essential tools they require for leading independent and enriched lives. The project envisions an inclusive and user-centric marketplace that offers a wide range of high-quality assistive aids to cater to the diverse needs of disabled use

3.1.1 Economical Feasibility

Economic feasibility determines whether the required software can generate financial gains for an organization. It involves the cost incurred by the software development team, the estimated cost of hardware and software, the cost of performing a feasibility study, and so on.

- Cost incurred on software development to produce long-term gains for an organization
- Cost required to conduct full software investigation (such as requirements elicitation and requirements analysis)
- Cost of hardware, software, development team, and training.

Based on the information provided in previous responses, the Inclusive E-Commerce Platform for Assistive Aids indicates economic feasibility. The platform provides several revenue streams, including product sales, consultancy fees, and possible partnerships with assistive aid makers. Users can browse and purchase a wide choice of high-quality assistive aids, generating a constant stream of cash from product sales. Furthermore, the platform provides assistive aid consultation services, where users can seek expert advice, allowing the platform to gain revenue through consultation fees. Furthermore, the potential for collaborations with producers of assistive devices gives a chance for the platform to gain cash through commissions or promotional fees for displaying their products. According to the market analysis, there is a desire for an inclusive e-commerce platform for assistive aids.

3.1.2 Technical Feasibility

Technical feasibility assesses the current resources (such as hardware and software) and technology, which are required to accomplish user requirements in the software within the allocated time and budget. Technical feasibility also performs the following tasks.

- Analyses the technical skills and capabilities of the software development team members
- Determines whether the relevant technology is stable and established
- Ascertains that the technology chosen for software development has many users so that they can be consulted when problems arise or improvements are required.

Based on the information provided in the preceding responses, the Inclusive E-Commerce Platform for Assistive Aids appears to be technically possible. The proposed functionalities, such as user registration, personalized product recommendations, order tracking, and user reviews, are routinely implemented in e-commerce platforms and can be accomplished utilizing current technologies and frameworks. Furthermore, the technical feasibility evaluation noted the availability of competent developers and resources, which is critical for the effective deployment of the platform. The

platform's technical needs can be met effectively with the correct skills and resources. However, it is vital to highlight that the real technological viability would be determined by the exact implementation details, the platform's complexity, and the scalability of the chosen technologies. In addition, incorporating machine learning for customised product.

3.1.3 Behavioral Feasibility

Behavioral feasibility assesses the extent to which the required software performs a series of steps to solve business problems and user requirements. This feasibility is dependent on human resources (software development team) and involves visualizing whether the software will operate after it is developed and be operative once it is installed. Operational feasibility also performs the following tasks.

- Determines whether the problems anticipated in user requirements are of high priority
- Determines whether the solution suggested by the software development team is acceptable
- Analyses whether users will adapt to a new software
- Determines whether the organization is satisfied by the alternative solutions proposed by the software development team.

The operational feasibility study considered a variety of areas of the platform's day-to-day operations, including as order fulfilment, logistics management, customer support, and mentor counselling services. According to the report, collaborations with reputable logistics suppliers and a devoted customer care team may assure smooth order processing and timely product delivery. Furthermore, the platform's capacity to provide consultative services for assistive technology via communication channels such as live chat and email improves its operational practicality. The platform can successfully assist individuals in picking the most appropriate assistive devices for their unique needs by giving professional guidance and customised recommendations to users. Furthermore, the platform's user-centric design, which allows users to leave reviews, track order status, and add things to their wish list or shopping cart, helps to its operational effectiveness.

3.1.4 Feasibility Study Questionnaire

1. Project Overview ?

The Inclusive E-Commerce Platform for Assistive Aids is a revolutionary online marketplace designed specifically for individuals with disabilities. Its primary mission is to empower users by providing a diverse range of assistive aids, enabling them to lead more independent and fulfilling lives. The platform boasts a comprehensive catalogue of mobility devices, communication tools, visual aids, hearing aids, and more, accompanied by detailed descriptions. Leveraging advanced machine learning algorithms, the platform offers personalized product recommendations based on user preferences and interactions, ensuring that users discover the most suitable assistive aids for their specific needs. A unique and invaluable feature is the provision of personalized consultations with expert mentors, allowing users to receive professional guidance in selecting the most appropriate assistive aids. The platform also fosters a supportive community where users can interact, share experiences, and seek advice from others facing similar challenges. Accessibility and inclusivity are paramount, with the platform adhering to stringent accessibility standards and providing customizable options for diverse user needs. By creating a secure

and supportive environment, the platform aims to revolutionize the accessibility of assistive aids, empowering individuals with disabilities and improving their overall well-being.

2. To what extend the system is proposed for ?

This project is intended to serve a large and diverse user base of disabled people. The system's goal is to meet a wide range of assistive aid needs, such as mobility devices, communication tools, visual aids, hearing aids, and other items. It is intended to provide a comprehensive and user-friendly platform that accommodates diverse disabilities, allowing users to quickly access and browse the website in order to locate the most appropriate items for their unique needs. The portal is accessible to both individuals with impairments and their caregivers, as well as healthcare professionals looking to prescribe appropriate assistive aids to their patients. The platform ensures that consumers receive customised service by delivering individualized product suggestions and a consulting service with experienced mentors.

3. Specify the Viewers/Public which is to be involved in the System ?

Users with Disabilities, Caregivers or family members, Healthcare Professionals, General Public.

4. List the Modules included in your System ?

Login and Signup, Cart, Wishlist, Review, Payment.

5. Identify the users in your project ?

User, Admin, Mentor, Delivery Team.

6. Who owns the system ?

Administrator.

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

Healthcare Industry.

8. Details of site that you have visited for data collection.

Enabling Devices - <https://enablingdevices.com/shop/>

Questionnaire to collect details about the project

1. What are the essential features you expect from an inclusive e-commerce platform for assistive aids?

Expert consultations.

2. How the payment is collected from user?

Cash or Credit card facility.

3. How important is it to have access to expert mentors for personalized consultations while choosing assistive aids ?

It is very important, as expert advice can ensure to make the right choice and consider factors that may not have thought of.

4. How do you handle the product delivery ?

For the user's convenience we opt for in-house delivery.

5. How important is it for you to have a seamless and accessible checkout process on the platform ?

It is very important to have an accessible checkout process would make the overall experience more enjoyable.

6. How do you prefer to receive product recommendations and suggestions ?

For the users it is prefer receiving personalized emails with relevant product recommendations.

7. Are users able to track the status of their orders?

Yes, users can track the status of their orders on the platform. They will receive real-time updates about their order's processing, shipping, and delivery status.

8. How does the platform handle order fulfilment and logistics?

The platform partners with reliable logistics providers to ensure efficient order fulfilment and timely delivery of products to customers. This ensures a seamless shopping experience for users.

9. How can users seek expert advice and consultation services on assistive aids?

The platform offers an "Assistive Aid Consultation" module where users can request expert advice on selecting the most suitable assistive aids for their unique situations. They can do this through communication channels such as live chat and email.

10. What are the key functionalities that users can access on the platform?

Users can access a range of functionalities, including user registration and login, personalized product recommendations, the ability to add and remove products from the cart, writing reviews for purchased products, tracking order status.

3.1 SYSTEM SPECIFICATION**3.2.1 Hardware Specification**

Processor - intel core i5

RAM - 8 G B

Hard disk -

3.2.2 Software Specification

Front End - HTML, CSS

Back End - Django

Database - SQL

Client on PC - Windows 7 and above.

Technologies used - JS, HTML5, AJAX, J Query, Django, CSS

3.3 SOFTWARE DESCRIPTION

3.3.1 DJANGO

Django is a powerful and versatile open-source web framework written in Python. It follows the model-template-views (MTV) architectural pattern, which promotes clean, pragmatic design and rapid development. Django simplifies the creation of complex, database-driven websites by emphasizing reusability, pluggability, and the "Don't Repeat Yourself" (DRY) principle. It comes equipped with a wide array of built-in features, including an intuitive admin interface, an ORM for database management, and a robust set of security measures. Django's extensive ecosystem of libraries and packages further accelerates development, making it a popular choice for building dynamic web applications, content management systems, and more. Its emphasis on efficiency, scalability, and maintainability has established Django as a go-to framework for developers looking to build robust and scalable web applications.

3.3.2 SQLite

SQLite is a lightweight, serverless, self-contained, and open-source relational database management system (RDBMS) that operates without the need for a separate server process. It is widely known for its simplicity, efficiency, and portability, making it an ideal choice for embedded systems, mobile applications, and small to medium-sized projects. SQLite stores data in a single, self-contained file, simplifying setup and management. Despite its minimalistic footprint, SQLite supports standard SQL syntax, transactions, and various data types. It is ACID-compliant, ensuring data integrity and reliability. Additionally, SQLite provides a straightforward API, enabling seamless integration into a wide range of programming languages and platforms. Due to its efficiency and ease of use, SQLite is a popular choice for applications where a lightweight, standalone database is preferred.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

System design, often referred to as architectural design, is a critical phase in the software development process. It involves conceptualizing and planning the architecture and structure of a software system to meet specific requirements and objectives. This phase encompasses various aspects, including choosing the appropriate technologies, defining data models, designing algorithms, establishing communication protocols, and determining system components and their interactions. The goal of system design is to create a robust, scalable, and efficient solution that addresses both functional and non-functional requirements. It lays the foundation for the development team to implement and build the software system. Effective system design ensures that the resulting system is not only functional but also maintainable, adaptable, and capable of accommodating future enhancements. It is a crucial step towards achieving a successful and reliable software product.

4.2 UML DIAGRAM

UML, or Unified Modeling Language, is a standardized visual notation used in software engineering to represent systems and their components. It provides a common language for software developers, analysts, and stakeholders to visually communicate and understand the architecture, behavior, and relationships within a system. UML diagrams come in various types, each serving a specific purpose, such as depicting class structures, illustrating user interactions, modeling system behavior, and more. These diagrams play a crucial role in system design, aiding in the planning, development, and documentation of software projects. By offering a clear and concise visual representation, UML diagrams enhance collaboration, facilitate effective communication, and serve as invaluable blueprints for building robust and efficient software systems.

- 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

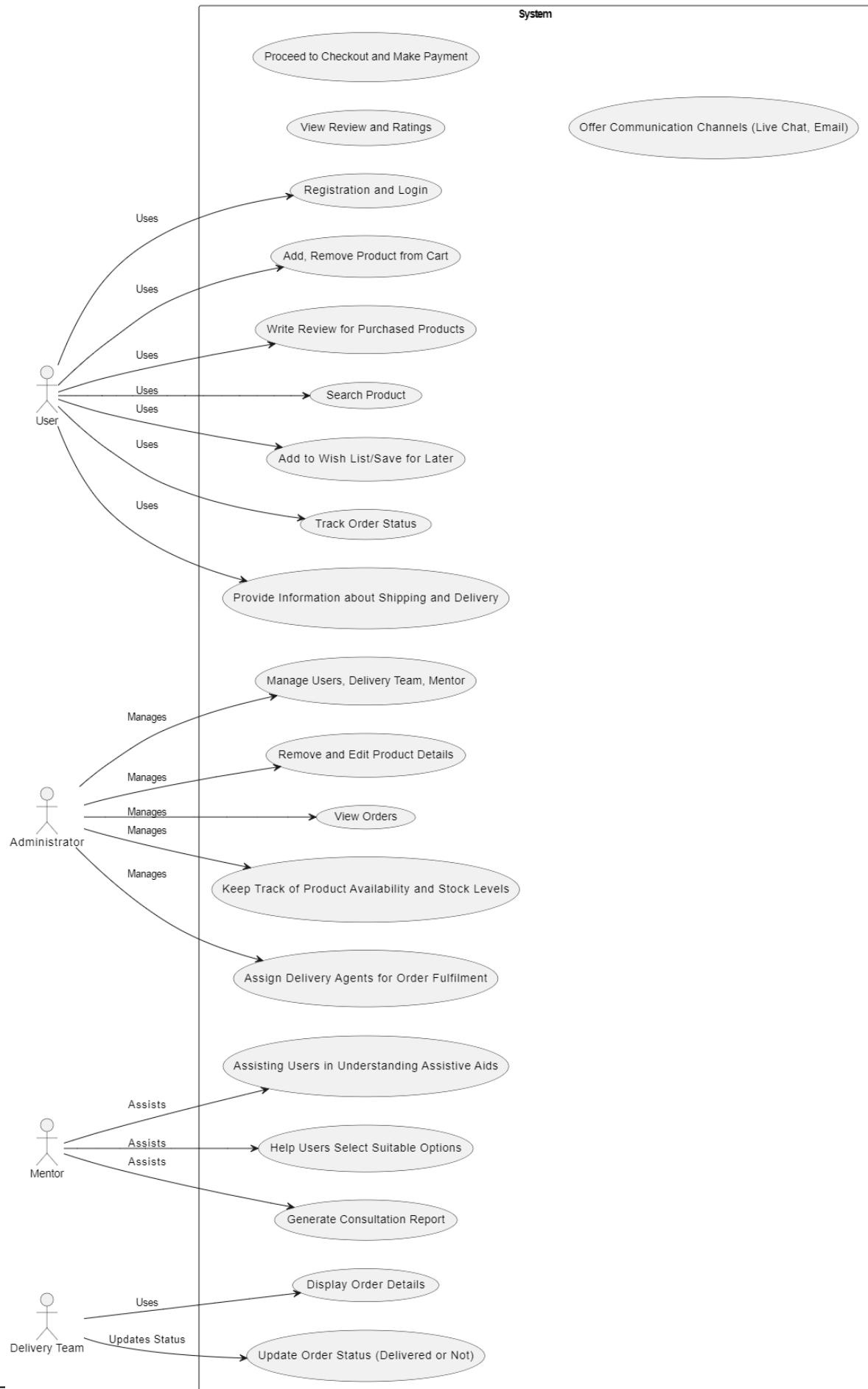
A use case diagram in UML (Unified Modeling Language) is a graphical representation that depicts the interactions between an external user (also known as an actor) and a system. It provides a high-level view of the system's functionality, focusing on what the system does from the perspective of its users.

In a use case diagram, actors are represented as stick figures, and use cases (representing specific functionalities or tasks) are depicted as ovals. Arrows connecting actors to use cases illustrate the interactions or relationships between them.

For example, in an online shopping system, actors could include "Customer" and "Admin," while use cases might encompass actions like "Browse Products," "Add to Cart," and "Manage Inventory." The diagram helps to identify all possible actions a user can take within the system.

Use case diagrams are particularly useful for:

- **Understanding System Functionality:** They provide a visual overview of the system's features and capabilities from an external user's perspective.
- **Defining Requirements:** Use cases serve as a foundation for identifying, organizing, and prioritizing system requirements.
- **Clarifying Communication:** They facilitate communication between stakeholders, developers, and designers, ensuring a common understanding of system functionality.
- **Providing a Basis for Test Cases:** Use cases can be used to derive test scenarios to verify that the system meets its intended functionality.
- **Identifying System Boundaries:** They help define what is inside the system (use cases) and what is outside (actors), clarifying the system's scope.
- **Highlighting System Interactions:** Use case diagrams demonstrate how actors and the system interact, illustrating the flow of tasks and information.
- **Overall,** use case diagrams play a pivotal role in the early stages of system development, providing a clear and structured representation of user-system interactions. They serve as a valuable tool for requirement analysis, system design, and communication among project stakeholders.



4.2.1 SEQUENCE DIAGRAM

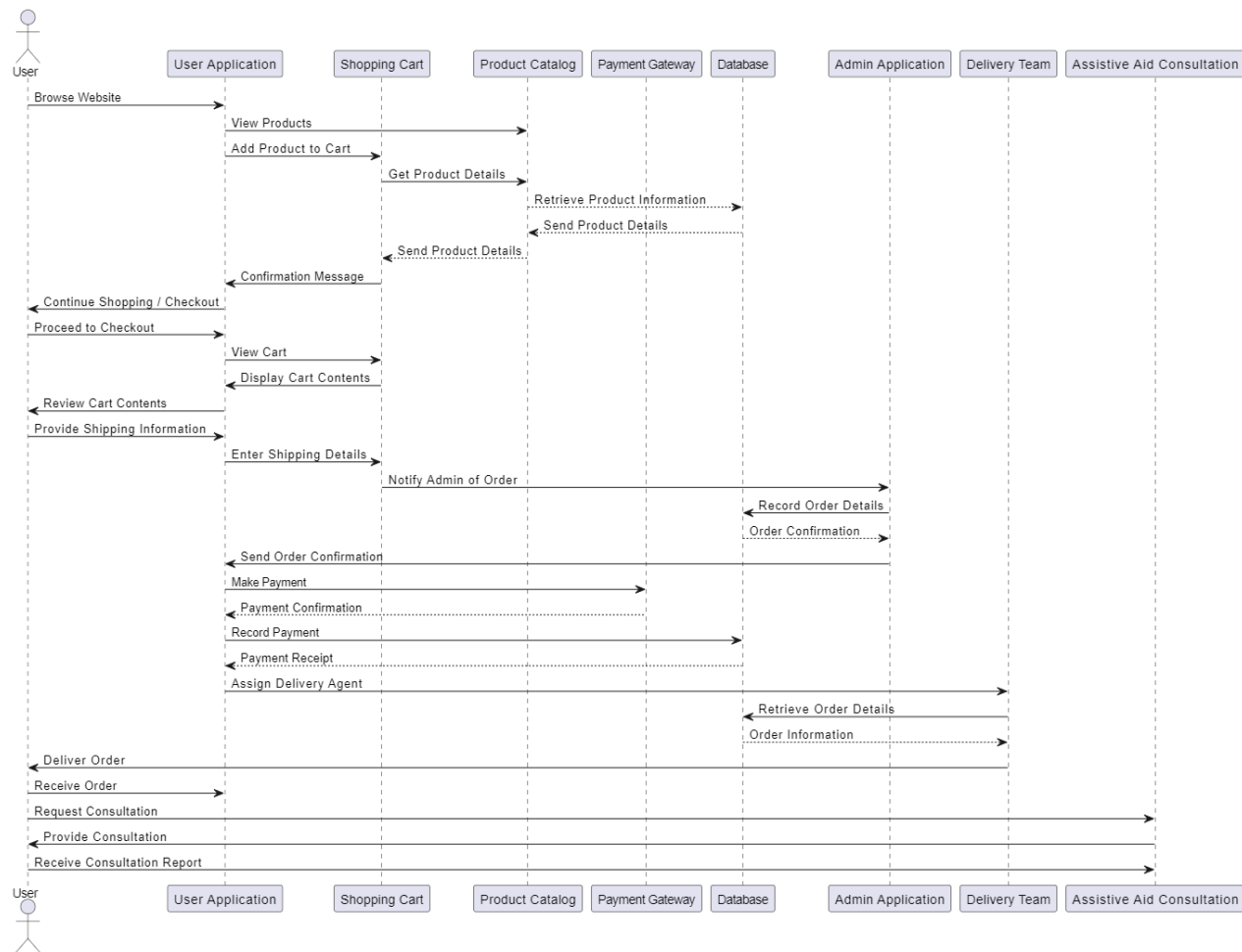
A sequence diagram in UML (Unified Modeling Language) is a visual representation that illustrates the interactions between objects or components within a system over a specific period of time. It emphasizes the chronological order of messages exchanged between these objects, providing a dynamic view of system behavior.

In a sequence diagram, objects are represented as rectangles, with their lifelines depicted vertically. The flow of interactions is shown as arrows between these objects, indicating the messages being sent and received. The timeline progresses from top to bottom, representing the passage of time.

For example, in a simple online ordering system, a sequence diagram might show a "Customer" object interacting with a "Shopping Cart" object by sending messages like "Add Item," "Remove Item," and "Checkout."

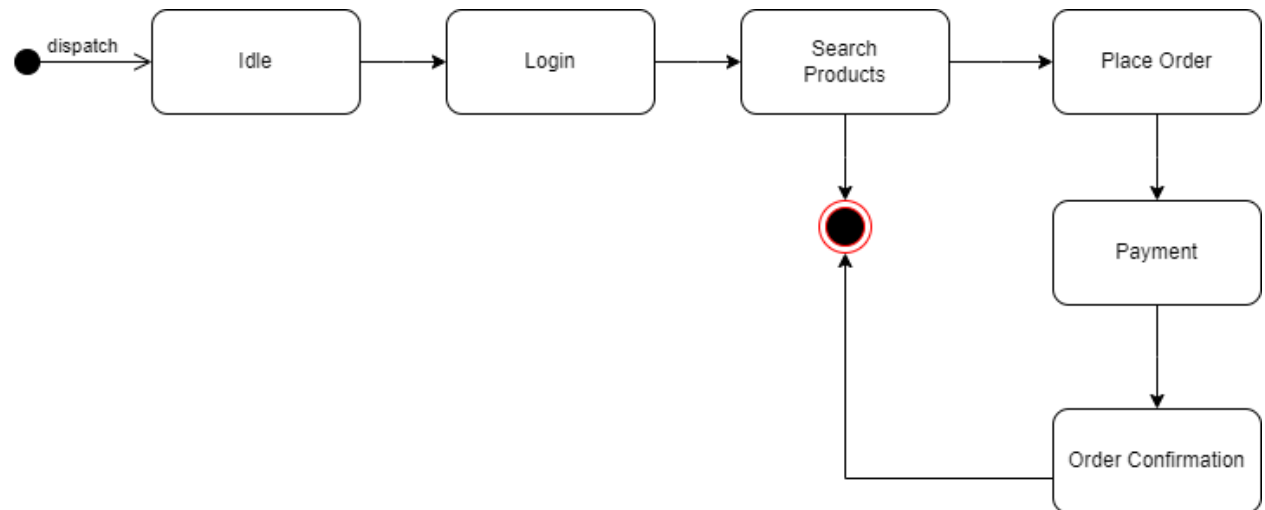
Sequence diagrams are particularly useful for:

- **Understanding System Behavior:** They provide a dynamic view of how objects collaborate and interact to accomplish a specific task or scenario.
- **Modeling Complex Processes:** They break down complex interactions into manageable steps, making it easier to analyze and design systems.
- **Identifying Object Interactions:** They highlight the flow of messages between objects, allowing developers to see how different components collaborate.
- **Testing and Debugging:** They can be used as a basis for creating test cases and scenarios to verify the correctness and functionality of a system.
- **Communication:** Sequence diagrams serve as a valuable communication tool among stakeholders, developers, and designers, ensuring a common understanding of system behavior.
- **Optimizing System Performance:** They can reveal potential areas for optimization by visualizing the sequence of interactions and identifying areas of high or low activity.
- **Overall,** sequence diagrams are crucial for capturing the dynamic aspects of a system, showing how objects collaborate to achieve specific tasks or functionalities. They provide a detailed and concrete representation of system behavior, making them invaluable in the design, analysis, and development of software systems.



4.2.2 State Chart Diagram

A State Chart Diagram is a type of UML behavioral diagram that visualizes the various states an object or system can be in and how it transitions between these states in response to events. It provides a clear depiction of an entity's behavior over time. The diagram consists of states, which represent specific conditions or situations, and transitions, which depict the events or triggers that cause the object to move from one state to another. Initially, the object is in an initial state, often denoted by an arrow. As events occur, the object transitions to different states, reflecting changes in its behavior or attributes. Each state may have entry and exit actions, representing actions performed when entering or leaving that state. This diagram is especially useful for modeling the behavior of complex systems, such as software applications or devices, where understanding the various operational states and their transitions is essential for comprehensive system understanding and development. It aids in identifying potential issues or areas for optimization in a system's behavior by providing a visual representation of its dynamic aspects.



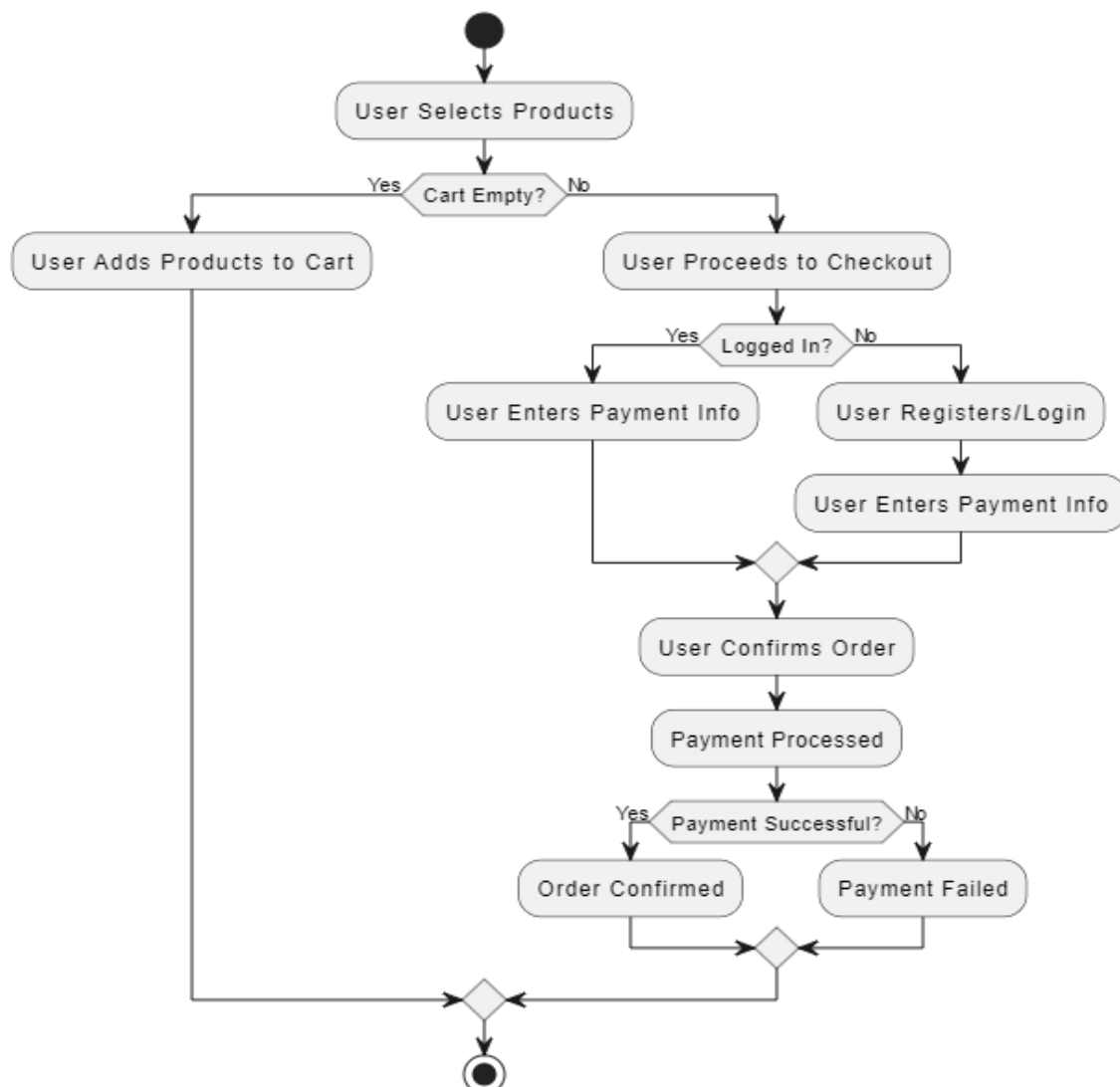
4.2.2 Activity Diagram

An Activity Diagram is a type of UML diagram that visualizes the flow of activities, actions, and decisions within a system or process. It provides a clear and detailed illustration of how tasks and actions are performed, showing the sequence of steps and their dependencies.

Here's a detailed explanation of the components and concepts commonly found in an Activity Diagram:

- **Initial Node:** This is represented by a solid circle and indicates the starting point of the activity. It shows where the process begins.
- **Final Node:** This is represented by a solid circle surrounded by a hollow circle and indicates the end point of the activity. It shows where the process concludes.
- **Action or Activity:** Represented by a rectangle with rounded corners, an action represents a specific task or operation performed within the system. It could be a simple operation like "Calculate Total" or a more complex process.
- **Flow Arrows:** These arrows connect the different elements of the diagram, indicating the flow of control from one activity to another. They show the sequence in which activities are performed.
- **Decision or Branching Point:** Represented by a diamond shape, this indicates a point in the process where a decision is made. Depending on the outcome of the decision, the process may follow different paths.
- **Merge or Join Node:** Also represented by a diamond shape, this indicates a point in the process where multiple paths converge back into a single path.

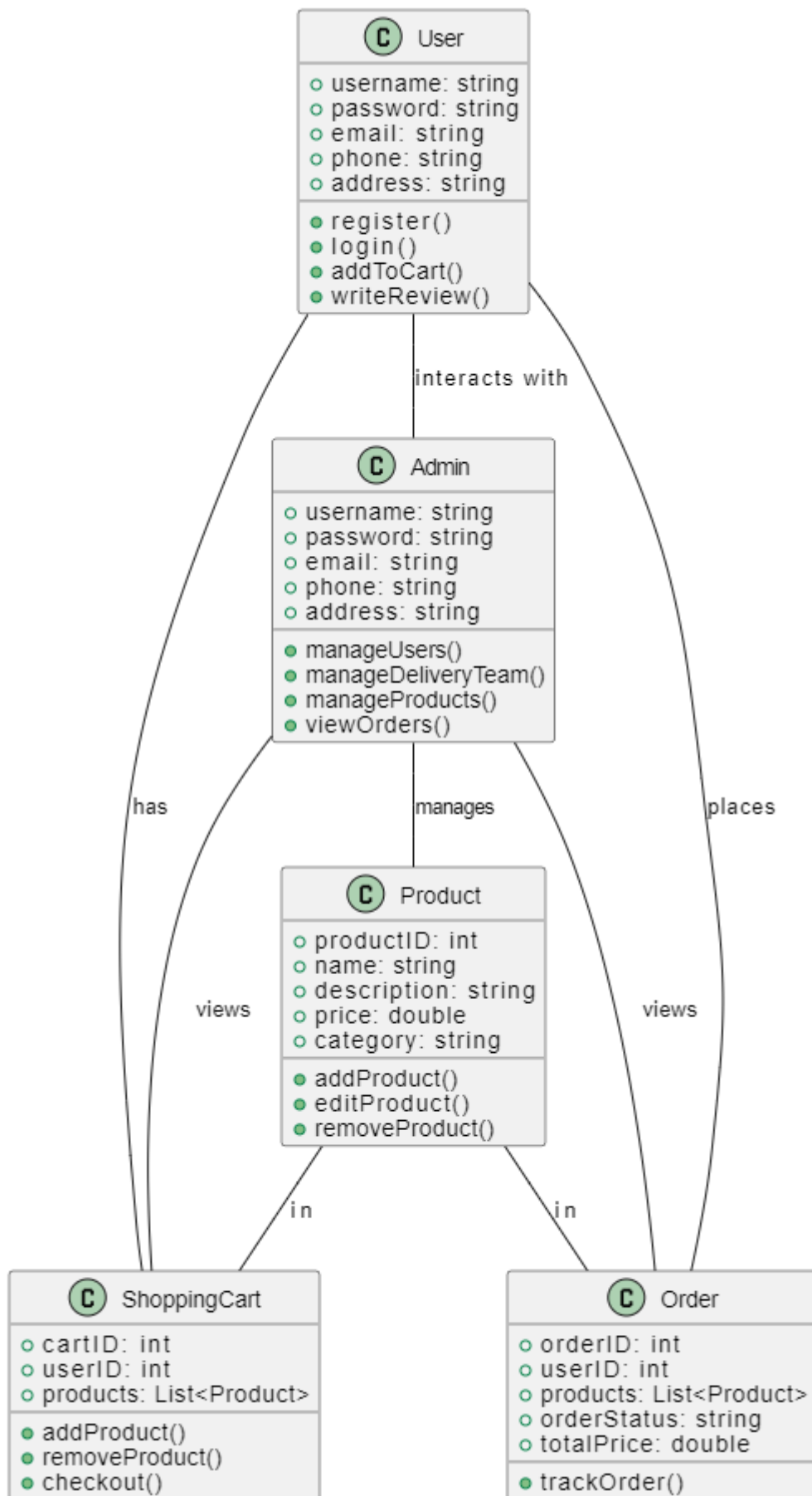
- Fork Node: This indicates a point in the process where a single path splits into multiple concurrent paths. It's represented by a bar.
- Join Node: This indicates a point in the process where multiple concurrent paths converge back into a single path. It's represented by a bar.
- Guard Condition: This is a condition associated with a decision point that determines which path will be taken. It's typically written near the outgoing flow arrow from the decision node.
- Swimlanes: These are used to partition the activities based on the entities or roles responsible for performing them. They help in visualizing which entities are involved in each activity.
- Object or Token: In some Activity Diagrams, you may see objects or tokens moving between activities. These represent the flow of control or data within the system.



4.2.3 Class Diagram

A class diagram is a type of UML (Unified Modeling Language) diagram that provides a visual representation of the structure and relationships within a system. It primarily focuses on the classes, their attributes, methods, and the associations between them. Here's an explanation of the key components of a class diagram:

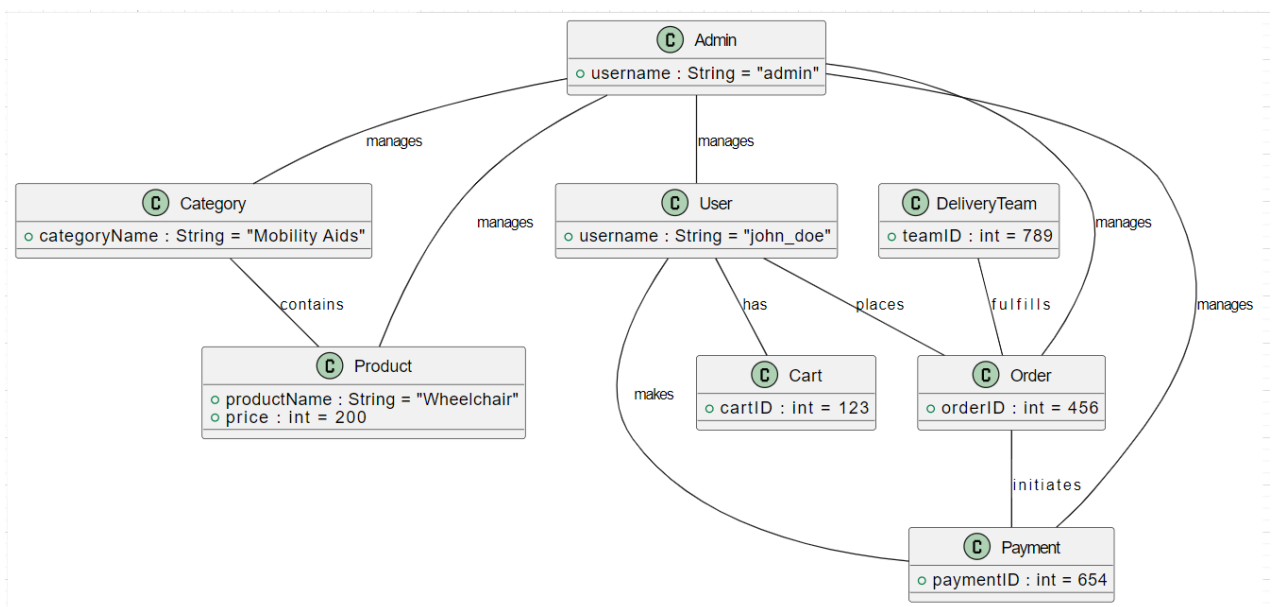
- A class represents a blueprint for creating objects. It defines the attributes (properties) and methods (behaviors) that the objects of the class will have. In a class diagram, a class is represented as a rectangle with three compartments.
- Attributes are the data members or properties of a class. They describe the characteristics or state of the objects that will be created from the class. Attributes are listed in the top compartment of the class rectangle and are typically shown with their data types.
- Methods represent the behaviors or operations that the objects of a class can perform. They define the actions that the objects can take. Methods are listed in the middle compartment of the class rectangle and are usually accompanied by their return types.
- Associations represent the relationships between classes. They show how objects of one class interact or relate to objects of another class. Associations are depicted as lines connecting the classes, often with arrows indicating the direction of the relationship.
- Multiplicity specifies the number of instances of one class that can be associated with a single instance of another class. It is represented using numbers or symbols near the association lines (e.g., "1", "0..*", etc.).
- Inheritance is a mechanism by which one class (subclass) inherits the attributes and methods of another class (superclass). It represents an "is-a" relationship. In a class diagram, inheritance is shown as an arrow pointing from the subclass to the superclass.
- Composition and aggregation represent different forms of "has-a" relationships. Composition implies a strong ownership relationship, where one class (whole) contains another class (part), and the part cannot exist without the whole. Aggregation, on the other hand, represents a weaker form of association.
- Dependency indicates that one class relies on another class, but there is no ownership or strong relationship. It represents a "uses-a" relationship. In a class diagram, dependency is shown as a dashed arrow pointing from the dependent class to the class it uses.



4.2.4 Object Diagram

Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams. Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment. Object diagrams are used to render a set of objects and their relationships as an instance. The object diagram holds the same purpose as that of a class diagram. The class diagram provides an abstract view which comprises of classes and their relationships, whereas the object diagram represents an instance at a particular point of time.

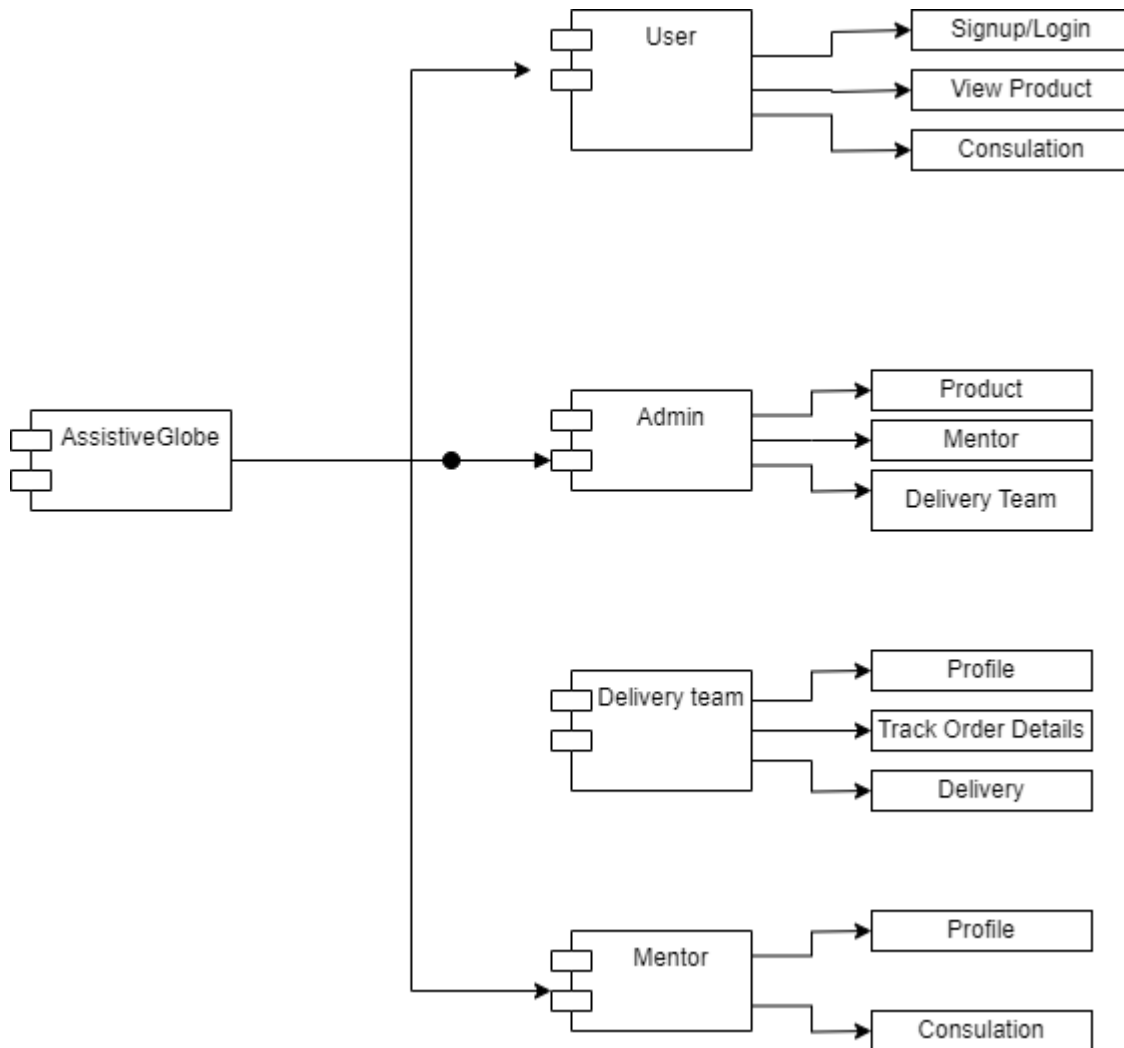
The object diagram is actually similar to the concrete (actual) system behavior. The main purpose is to depict a static view of a system.



4.2.5 Component Diagram

A component diagram is a type of UML (Unified Modeling Language) diagram used in software engineering to visualize the structural organization of a system. It illustrates the various components or modules that make up a software application and the relationships between them. Components represent the physical or logical elements of the system, which can be a class, package, executable file, or any other software entity. These components encapsulate

functionalities and can be independently replaceable or upgradeable. The connections between components indicate how they interact and communicate with one another. Additionally, component diagrams provide a clear overview of the system's architecture, making it easier for developers, stakeholders, and designers to understand the system's composition and dependencies. This diagram type is particularly useful for planning, designing, and managing complex software systems, helping to ensure their modularity, scalability, and maintainability.



4.2.8 Deployment Diagram

A deployment diagram in software engineering is a type of UML (Unified Modeling Language) diagram that provides a visual representation of the physical configuration of a software system. It illustrates how software components and artifacts are distributed across hardware nodes, servers, and devices in a real-world environment.

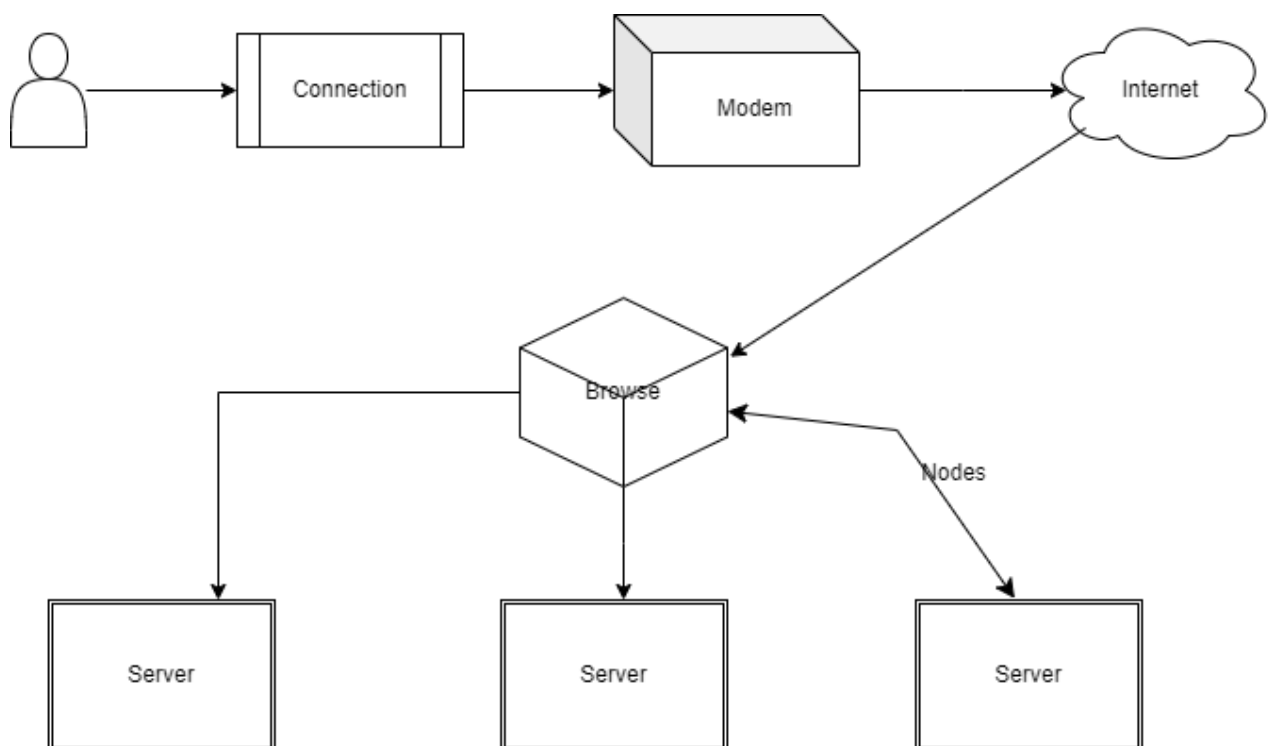
In a deployment diagram, nodes represent the physical entities in the system, which can be actual hardware devices like servers, workstations, or even software containers like databases. Artifacts, on the other hand, represent the software components, such as executable files, libraries, or configuration files.

Lines with arrows, known as associations, connect nodes and artifacts, indicating the relationships and dependencies between them. These associations demonstrate how the software components are deployed on specific hardware nodes.

Deployment diagrams are particularly useful for visualizing and understanding how a system will be physically implemented and operated. They help in planning the distribution of software components to ensure optimal performance, scalability, and reliability.

For example, in a web application deployment diagram, you might have nodes representing web servers, application servers, and database servers. Artifacts could include the web application files, database schemas, and configuration files. The associations would show how these components are deployed and interact with each other.

Overall, a deployment diagram provides a clear overview of the physical infrastructure of a software system, aiding developers, system architects, and stakeholders in making informed decisions about deployment strategies and resource allocation.



4.2.9 Collaboration Diagram

A collaboration diagram, also known as a communication diagram, is a type of UML (Unified Modeling Language) diagram that provides a visual representation of how objects or elements in a system interact with one another. It emphasizes the dynamic behavior and interaction between different elements in a system, focusing on the flow of messages between objects.

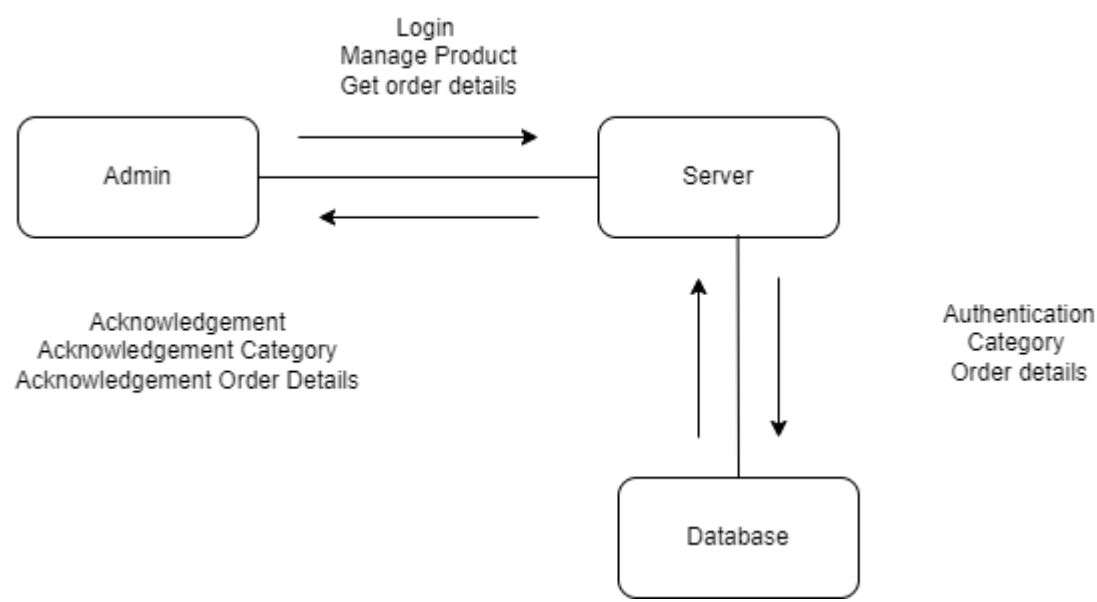
In a collaboration diagram, objects are represented as rectangles or boxes, and the interactions between them are depicted as labeled arrows. These arrows indicate the flow of messages or communication between the objects.

Key elements in a collaboration diagram include:

- **Objects:** Each object is represented as a rectangle or box with the object's name and class/type. These objects can be instances of classes in the system.
- **Links:** Links are the lines or arrows connecting the objects, representing the communication or interaction between them. They are labeled with the message being passed.
- **Messages:** Messages are the communication events between objects. They are represented by arrows connecting the objects, and they indicate the flow of information.
- **Roles:** Roles describe the specific behavior or responsibility of an object in a particular interaction.
- **Multiplicity:** Multiplicity notation can be used to indicate how many instances of an object are involved in a particular interaction.

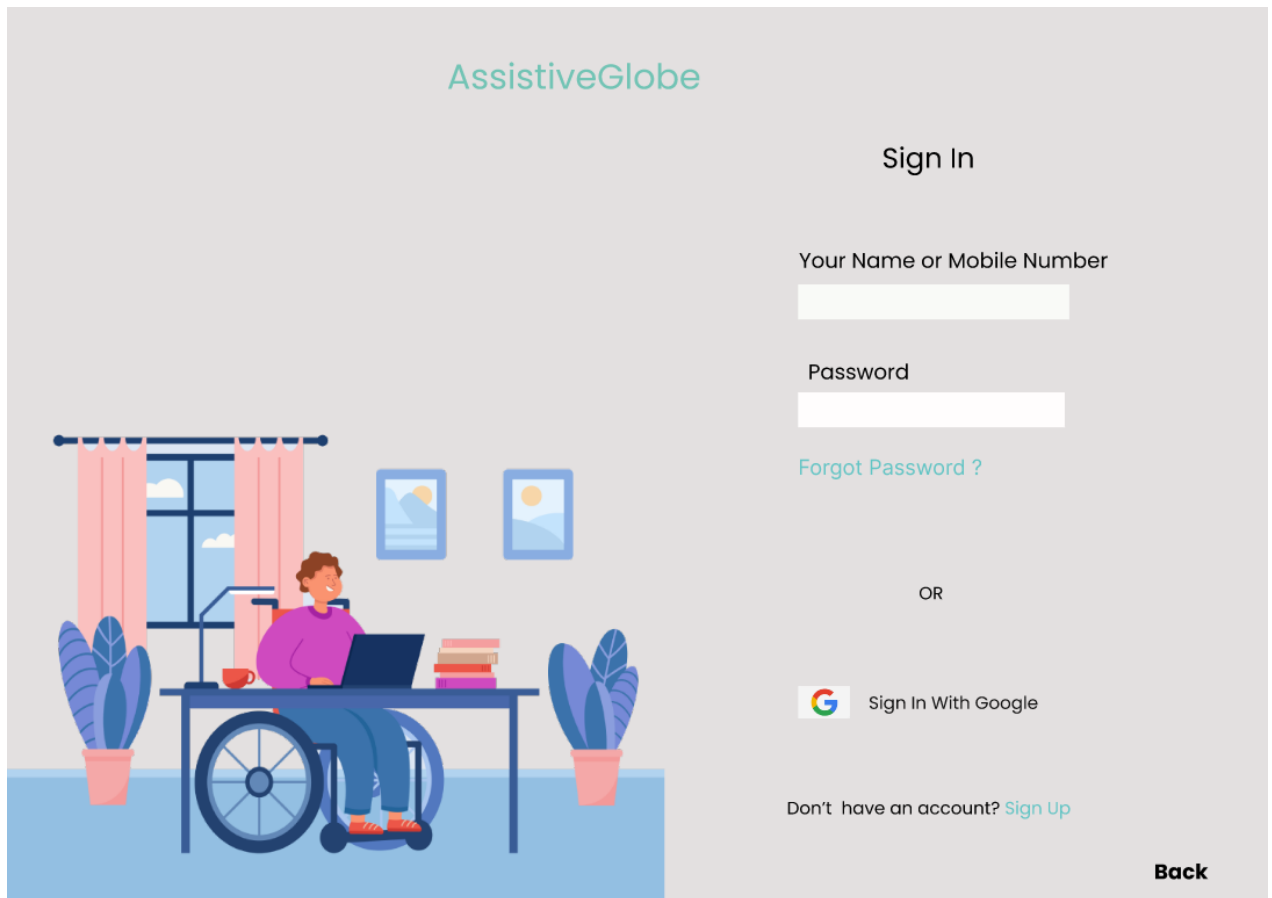
Collaboration diagrams are particularly useful for visualizing complex interactions in a system, especially when there are multiple objects involved and the sequence of interactions is important. They are often used during the design phase of software development to understand and document how different components work together.

Overall, collaboration diagrams provide a dynamic view of a system, showing how objects collaborate to achieve a specific task or functionality. They help in understanding the runtime behavior of a system and can aid in identifying potential design flaws or communication bottlenecks.



4.3 USER INTERFACE DESIGN USING FIGMA

Form Name: Login Form



The login form is titled "AssistiveGlobe" in teal. It features a large illustration on the left showing a person in a wheelchair working at a desk with a laptop, a window with pink curtains, and two potted plants. The form itself is on the right, titled "Sign In". It includes input fields for "Your Name or Mobile Number" and "Password", a "Forgot Password?" link, an "OR" separator, a "Sign In With Google" button with the Google logo, and a "Don't have an account? Sign Up" link. A "Back" button is located at the bottom right.

AssistiveGlobe


Sign In

Your Name or Mobile Number

Password

[Forgot Password ?](#)

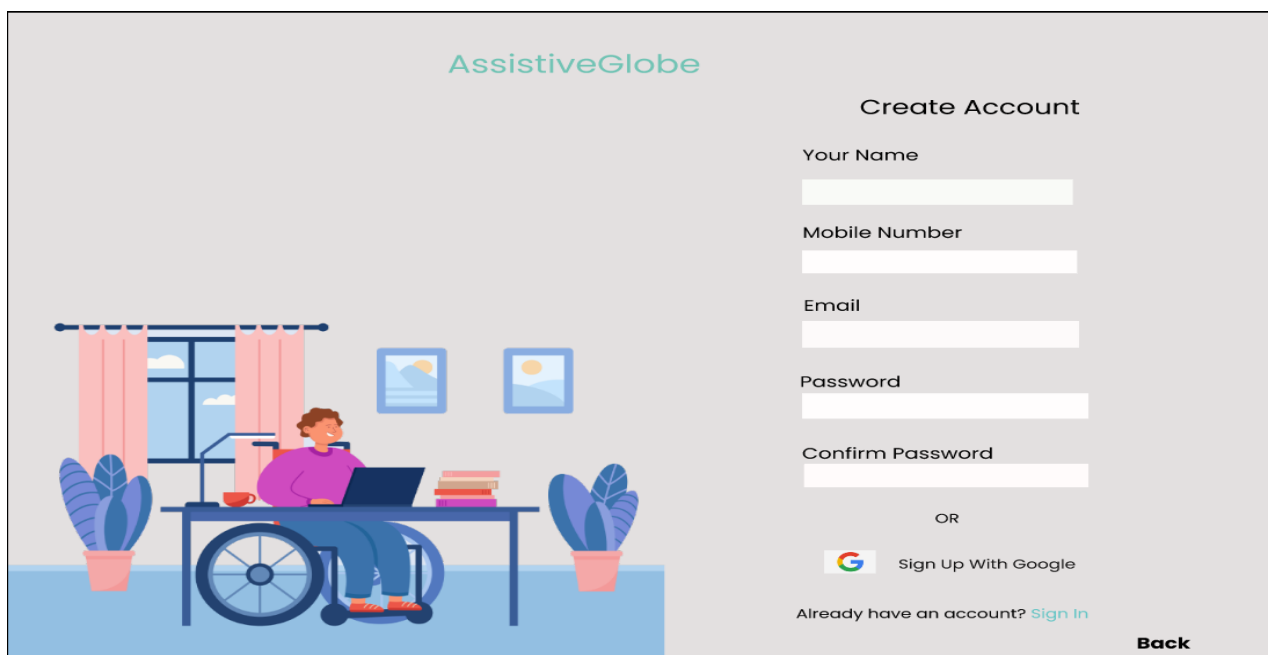
OR

 Sign In With Google

Don't have an account? [Sign Up](#)

Back

Form Name: Sign Up Form



The sign up form is titled "AssistiveGlobe" in teal. It features the same large illustration on the left as the login form. The form is on the right, titled "Create Account". It includes input fields for "Your Name", "Mobile Number", "Email", "Password", and "Confirm Password", an "OR" separator, a "Sign Up With Google" button with the Google logo, and an "Already have an account? Sign In" link. A "Back" button is located at the bottom right.

AssistiveGlobe

Create Account

Your Name


Mobile Number

Email

Password

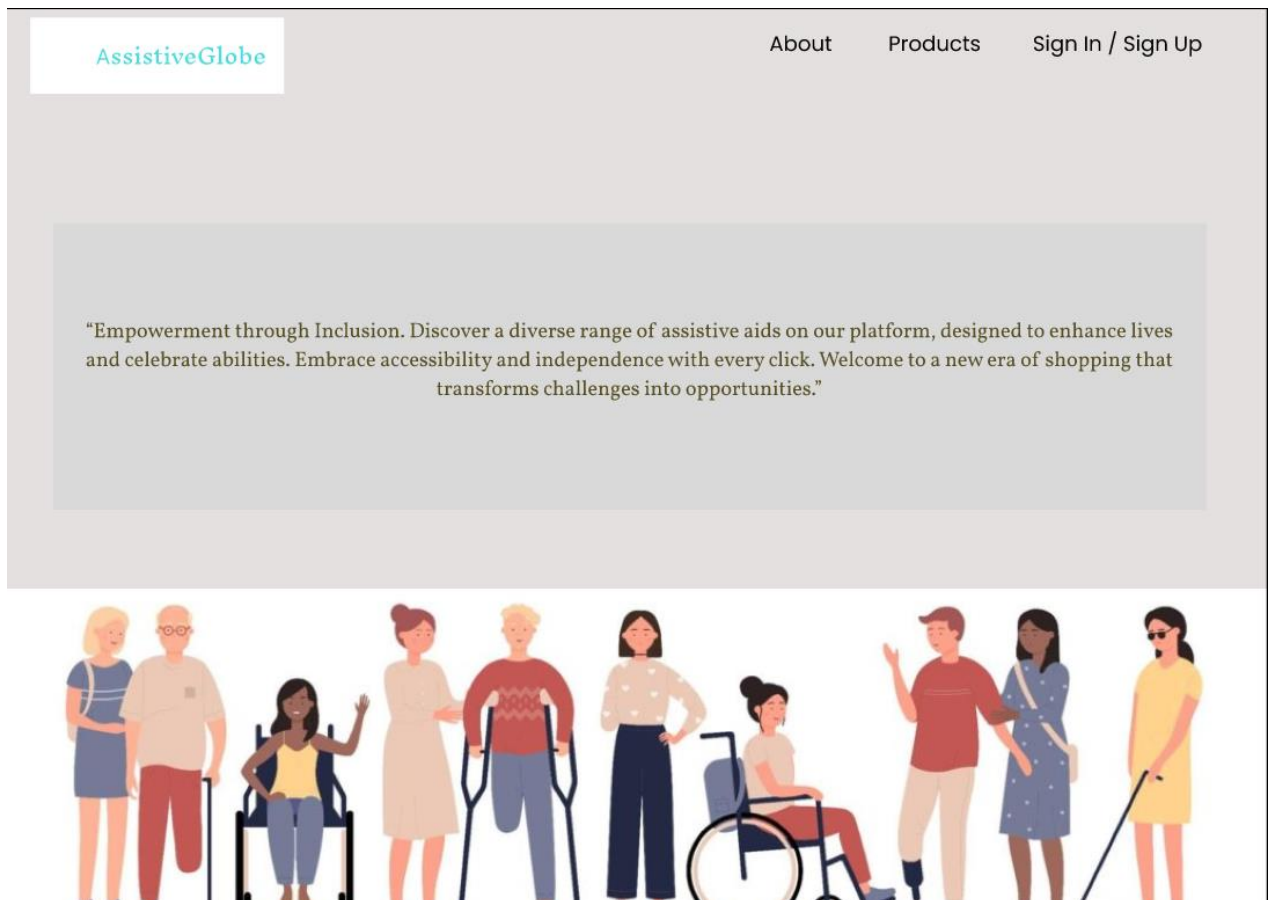
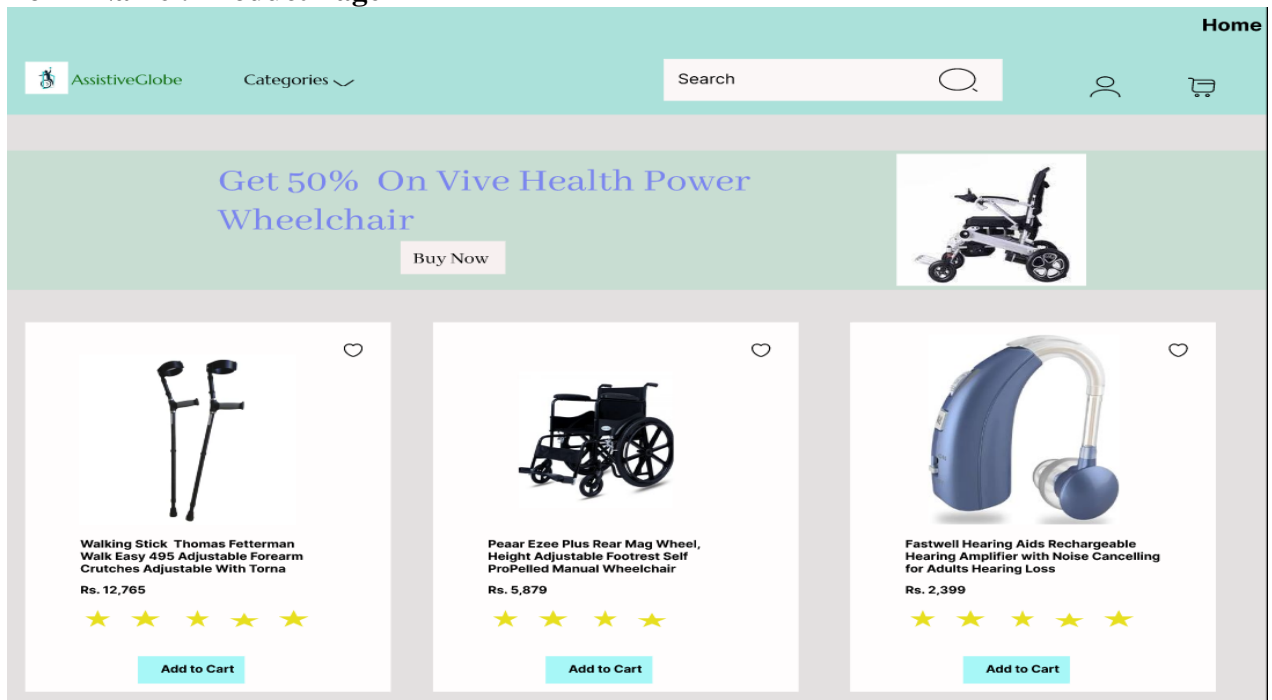
Confirm Password

OR

 Sign Up With Google

Already have an account? [Sign In](#)

Back

Form Name: Home Page**Form Name : Product Page**

4.4 DATABASE DESIGN

In computing, a database is an organized collection of data or a type of data store based on the use of a database management system (DBMS), the software that interacts with end users, applications, and the database itself to capture and analyze the data. The DBMS additionally encompasses the core facilities provided to administer the database. The sum total of the database, the DBMS and the associated applications can be referred to as a database system. Often the term "database" is also used loosely to refer to any of the DBMS, the database system or an application associated with the database. Small databases can be stored on a file system, while large databases are hosted on computer clusters or cloud storage. The design of databases spans formal techniques and practical considerations, including data modeling, efficient data representation and storage, query languages, security and privacy of sensitive data, and distributed computing issues, including supporting concurrent access and fault tolerance.

4.4.1 Relational Database Management System (RDBMS)

A Relational Database Management System (RDBMS) is a software application that facilitates the creation, organization, and management of structured data in a manner that adheres to the principles of the relational model. In an RDBMS, data is organized into tables, each consisting of rows (representing individual records) and columns (representing specific attributes or fields). These tables can establish relationships with one another based on shared attributes, enabling the efficient retrieval and manipulation of information. RDBMS systems provide a standardized query language, commonly known as SQL (Structured Query Language), which allows users to interact with the database by writing queries to retrieve, insert, update, and delete data. They also enforce data integrity through mechanisms like primary keys, foreign keys, and constraints, ensuring the accuracy and consistency of the stored information. ACID (Atomicity, Consistency, Isolation, Durability) compliance is a hallmark of RDBMS, guaranteeing that database transactions occur reliably and securely. This robust and organized approach to data management makes RDBMS systems a cornerstone of modern database technology, extensively employed in applications ranging from web development to large-scale enterprise solutions. Popular examples of RDBMS include MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server, and SQLite.

4.4.2 Normalization

Normalization is a fundamental process in database design aimed at minimizing data redundancy and dependency, ultimately enhancing the efficiency and integrity of a database. It involves organizing the data into multiple related tables, each with a specific purpose and structure. The goal is to reduce duplication of information and ensure that data is logically organized to avoid anomalies like update, insertion, and deletion anomalies. This is achieved by breaking down larger tables into smaller, more manageable ones and establishing relationships between them. Normalization typically follows a series of formal rules, known as normal forms, each building on the previous one. These normal forms help ensure that a database is well-structured, with data distributed appropriately across tables. The process of normalization is crucial for preventing data inconsistencies, improving data integrity, and enabling more efficient data retrieval and manipulation operations in relational database management systems (RDBMS). By adhering to normalization principles, designers can create databases that are not only well-organized but also adaptable to changing business requirements.

Normalization in database design involves organizing data into tables to minimize redundancy and dependency. This process is guided by a series of rules called normal forms (NF). There are several normal forms, each building on the previous one. The most common ones include:

- First Normal Form (1NF): In 1NF, a table is considered normalized if it meets the following criteria:
 - It has a primary key (no duplicate rows).
 - All columns contain atomic values (indivisible values).
 - There are no repeating groups or arrays.
- Second Normal Form (2NF): A table is in 2NF if it meets the criteria of 1NF and:
 - It has no partial dependencies, meaning all non-key attributes are fully functionally dependent on the entire primary key.
- Third Normal Form (3NF): A table is in 3NF if it meets the criteria of 2NF and:
 - It has no transitive dependencies, meaning non-key attributes are not dependent on other non-key attributes.
- Boyce-Codd Normal Form (BCNF): BCNF is an extension of 3NF that addresses certain anomalies in rare cases where a table may have multiple candidate keys. It ensures that there are no non-trivial functional dependencies of non-key attributes on any candidate key.
- Fourth Normal Form (4NF): A table is in 4NF if it meets the criteria of BCNF and:
 - It has no multi-valued dependencies, meaning that a non-key attribute is not dependent on a combination of attributes that form a candidate key.
- Fifth Normal Form (5NF): Also known as Project-Join Normal Form (PJNF), this form addresses situations where there are overlapping candidate keys. It involves decomposing tables to eliminate redundancy while maintaining dependency preservation.

4.4.3 Sanitization

Data sanitization, also known as data cleansing or data scrubbing, is a critical process in data management that involves identifying and rectifying errors, inconsistencies, and inaccuracies within a dataset. It aims to ensure that the information stored in a database or system is accurate, reliable, and conforms to predefined standards. This process encompasses a range of activities, including removing duplicate records, correcting spelling mistakes, standardizing formats, and validating data against predefined rules or constraints. Additionally, data sanitization involves identifying and handling missing or incomplete information. This meticulous attention to data quality is essential for making informed decisions, generating accurate reports, and ensuring the integrity of business operations. Data sanitization is particularly crucial in fields such as finance, healthcare, and any domain where precise, reliable information is paramount. Moreover, it plays a vital role in compliance with regulatory requirements and maintaining the credibility and trustworthiness of an organization's data assets. Through data sanitization, organizations can optimize their data for effective analysis, reporting, and utilization, ultimately enhancing the overall efficiency and reliability of their operations.

4.4.4 Indexing

Indexing is a fundamental technique in database management and information retrieval that enhances the efficiency of data retrieval operations. It involves creating specialized data structures, known as indexes, which serve as pointers or references to the actual data stored in a database table. These indexes are organized in a way that allows for rapid lookup of specific values or ranges. When a query is made against a table, the database engine can consult the index first to quickly locate the relevant records, reducing the need to scan the entire table. This leads to significantly faster query response times, particularly in large datasets. However, it's important to note that while indexing improves retrieval speed, it can slightly increase the time required for data modification operations like inserts, updates, and deletes, as the indexes also need to be maintained. Therefore, the decision to create indexes should be made judiciously, considering the specific use case and the types of queries expected to be performed. Properly designed and managed indexes play a critical role in optimizing database performance and ensuring that applications can retrieve data swiftly and efficiently.

4.5 TABLE DESIGN

1.Tbl_users_login

Primary key: **loginid**

Foreign key: **loginid** references table **Tbl_regusers**, **Tbl_deliveryteam**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	User_Id	Int(10)	Foreign Key	Authenticated User Id
2	Email	Varchar(20)	Not Null	Email
3	Password	Varchar(20)	Not Null	Password
4	Roles	Varchar(20)	Not Null	Identifying different users through user id
5	Status	Int(10)	Not Null	Status of the user

2.Tbl_users_registration

Primary key: user-id

Foreign key: **loginid** references table **Tbl_login**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	User_Id	Int(10)	Primary key	Users registration id

2	User_name	Varchar(20)	Not Null	Name of User
3	Email	Varchar(20)	Not Null	Email
4	Phone	Varchar(20)	Not Null	Phone number of user
5	Password	varchar(20)	Not Null	Password

3.Tbl_Category

Primary key: **cart_id**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Category_id	Int(10)	Primary Key	Category ID
2	Category_name	Varchar(20)	Not Null	Name of Category
3	Status	Int(10)	Not Null	Status of the category

4.Tbl_Product

Primary key: **product_id**

Foreign key: **category_id** references table **Tbl_category**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Product_id	Int(10)	Primary Key	Product ID
2	Category_id	Int(10)	Foreign Key	Category ID
3	Product_name	Varchar(20)	Not Null	Name of Product
4	Product_description	Varchar(50)	Not Null	Description of Product
5	Product_price	Int(10)	Not Null	Price of Product

5.Tbl_Cart

Primary key: **cart_id**

Foreign key: **user_id, product_id** references table **Tbl_login, Tbl_product**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Cart_id	Int(10)	Primary Key	Cart ID
2	User_id	Int(10)	Foreign Key	User ID
3	Product_id	int(10)	Foreign Key	Product ID

6.Tbl_Order

Primary key: **order_id**

Foreign key: **user_id, product_id, cart_id** references table **Tbl_login, Tbl_product, Tbl_cart**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Order_id	Int(10)	Primary Key	Order ID
2	User_id	Int(10)	Foreign Key	User ID
3	Product_id	int(10)	Foreign Key	Product ID
4	Cart_id	Int(10)	Foreign Key	Cart ID
5	Order_date	Date	Not Null	Ordered date
6	Amount	Int(10)	Not Null	Amount of the ordered product
7	Status	String	Not Null	Dispatched/Delivered

7.Tbl_Image

Primary key: **img_id**

Foreign key: **product_id** references table **Tbl_product**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Image_id	Int(10)	Primary Key	Image ID
2	Product_id	int(10)	Foreign Key	Product ID
3	Image	Varchar(10)	Not Null	Image ID

8.Tbl_Address

Primary key: **address_id**

Foreign key: **user_id** references table **Tbl_login**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Address_id	Int(10)	Primary Key	Address ID
2	User_id	int(10)	Foreign Key	User ID
3	Name	Varchar(10)	Not Null	Name of User
4	Address	text	Not Null	Address of the user
5	City	String	Not Null	City
6	Pincode	Varchar(10)	Not Null	Pincode
7	State	String(20)	Not Null	State
8	Phone	Varchar(20)	Foreign Key	Phone Number

8 . Tbl_Payment

Primary key: **payment_id**

Foreign key: **user_id, order_id** references table **Tbl_login, Tbl_order**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Payment_id	Int(10)	Primary Key	Payment ID
2	User_id	int(10)	Foreign Key	User ID
3	Order_id	int(10)	Foreign Key	Order ID
4	Amount	text	Not Null	Address of the user
5	City	String	Not Null	City
6	Pincode	Varchar(10)	Not Null	Pincode
7	State	String(20)	Not Null	State
8	Phone	Varchar(20)	Foreign Key	Phone Number

CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

System testing is a critical phase in the software development process that serves as a comprehensive evaluation of the entire software system. It is conducted in a controlled environment, simulating the actual operational conditions that the software will encounter once deployed. The primary objective of system testing is to validate that the software meets the specified requirements and functions as intended, taking into account all integrated components and modules.

During system testing, a set of well-defined test cases and scenarios are executed, covering a wide range of functionalities and use cases. These tests assess various aspects of the software, including its functionality, performance, reliability, security, and usability. Additionally, system testing examines the software's ability to handle exceptional situations and error conditions, ensuring that it behaves predictably and robustly even under adverse circumstances.

One key aspect of system testing is to verify that all individual components and modules, which have been thoroughly tested during unit and integration testing, work harmoniously together as a cohesive whole. It aims to uncover any issues that may arise due to interactions between different parts of the system.

5.2 TEST PLAN

A test plan is a comprehensive document that outlines the strategy, scope, objectives, resources, and schedule for testing activities within a software development project. It serves as a roadmap that guides the testing process from start to finish. The test plan defines the goals of testing, including what needs to be tested, the features to be covered, and the criteria for success. It also specifies the roles and responsibilities of team members involved in testing, as well as the tools and resources required for effective testing. Additionally, a test plan identifies the specific test cases, scenarios, and data sets that will be used to validate the functionality and performance of the software. It outlines the order in which tests will be executed and provides a framework for reporting and documenting test results. The plan also considers risk assessment and mitigation strategies, highlighting potential challenges and how they will be addressed.

A well-crafted test plan is essential for ensuring thorough and systematic testing, enabling the detection of defects and inconsistencies early in the development process. It provides a clear roadmap for testers, developers, and stakeholders, ensuring that testing efforts are aligned with project objectives and quality standards. The test plan is a crucial component of the overall quality assurance process, contributing to the delivery of a reliable and high-quality software product.

The different levels of testing include:

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

5.2.1 Unit Testing

Unit testing is a foundational practice in software development that involves testing individual units or components of a software application in isolation from the rest of the system. A "unit" in this context typically refers to the smallest testable part of a program, often a function or method. The primary goal of unit testing is to validate that each unit of code functions correctly and independently, according to its design and specifications.

During unit testing, developers write specific test cases that exercise the unit under various scenarios, including normal input, boundary conditions, and exceptional cases. These tests are automated, meaning they can be executed by a testing framework, allowing for quick and frequent validation of code changes. When a unit test is run, it checks whether the actual output of the unit matches the expected output. If they align, the unit test passes; otherwise, it fails, indicating a potential defect in the code.

Unit testing offers several benefits to the development process. It provides rapid feedback to developers, allowing them to catch and address issues early in the development cycle. This leads to higher code quality, reduces the likelihood of bugs persisting into later stages, and improves overall software reliability. Additionally, unit tests serve as a form of documentation, offering insights into the expected behavior of the code. They also facilitate refactoring and code maintenance efforts, enabling developers to make changes with confidence that existing functionality remains intact.

By systematically testing individual units of code, unit testing forms a critical foundation for building robust, maintainable, and high-quality software systems. It complements other testing practices like integration testing and system testing, collectively contributing to the development of reliable and dependable software products.

5.2.2 Integration Testing

Integration testing is a crucial phase in the software development process that focuses on assessing the interactions and interfaces between different modules or components of a system. Unlike unit testing, which tests individual units or pieces of code in isolation, integration testing examines how these units work together when integrated into a complete application. The primary objective of integration testing is to ensure that the integrated components collaborate effectively, exchanging data and triggering functionalities as intended.

During integration testing, various scenarios are executed to validate the flow of data and control between different modules. This includes testing different combinations of integrated components, as well as assessing the handling of boundary conditions and exceptional cases. The testing process may be incremental, with components integrated and tested one by one, or it may follow a big bang approach, where all components are integrated at once.

Integration testing helps identify and address issues related to data flow, interfaces, and communication protocols between different parts of the system. It ensures that the integrated system behaves cohesively and that any potential conflicts or inconsistencies between components are detected and resolved.

Integration testing complements unit testing and precedes system testing in the software testing life cycle. It plays a critical role in verifying that the various pieces of a software application function seamlessly together, contributing to the overall reliability and robustness of the system. By conducting integration testing, development teams can have confidence that the integrated components will work harmoniously when deployed in a real-world environment.

6.2.1 Validation Testing or System Testing

Validation Testing, also known as System Testing, is a critical phase in the software development life cycle that focuses on evaluating the entire integrated system to ensure it meets the specified requirements and functions correctly in its intended environment. This phase comes after the completion of integration testing, which verifies the interactions between individual components. Validation Testing takes a broader perspective, examining the system as a whole to confirm that it aligns with the stakeholders' expectations and fulfills its intended purpose.

During Validation Testing, a comprehensive set of test cases and scenarios are executed, covering a wide range of functionalities and use cases. These tests assess various aspects of the software, including its functionality, performance, security, usability, and compliance with business requirements. The objective is to ensure that the software meets the defined acceptance criteria and delivers the expected value to end-users.

Validation Testing aims to replicate real-world scenarios to simulate how end-users will interact with the system. This includes conducting tests under normal operational conditions, as well as testing for exceptional cases and handling error situations. The goal is to identify any discrepancies, defects, or deviations from the specified requirements.

The results of Validation Testing are crucial in determining whether the software is ready for deployment in a production environment. Any issues discovered during this phase are documented, reported, and addressed by the development team. Once all identified issues are resolved, the software is deemed ready for release to end-users or stakeholders.

Overall, Validation Testing is a pivotal step in the software development process, ensuring that the final product meets the needs and expectations of its intended users. It provides stakeholders with confidence in the quality and reliability of the software system before it is deployed in a live environment.

6.2.2 Output Testing or User Acceptance Testing

Output Testing, also known as User Acceptance Testing (UAT), is a crucial phase in the software development life cycle where the end-users or stakeholders assess the software to determine whether it meets their requirements and expectations. This phase comes after the completion of Validation Testing or System Testing, which ensures that the software functions correctly according to specified criteria.

During Output Testing or UAT, end-users actively engage with the software to evaluate its functionality, usability, and overall user experience. They execute predefined test cases and scenarios that cover various aspects of the system, including core functionalities, user interfaces, and workflows. The focus is on ensuring that the software delivers the intended value and effectively supports the users' tasks and objectives.

Output Testing is conducted in a controlled environment that closely mirrors the production setting. It aims to replicate real-world scenarios and situations, allowing users to interact with the software as they would in their day-to-day operations. This includes testing under normal operational conditions, as well as assessing how the system handles exceptional cases and error situations.

The results of Output Testing play a decisive role in determining whether the software is ready for deployment in a live environment. Any issues, discrepancies, or concerns raised during this phase are carefully documented, reported, and addressed by the development team. Once all identified issues are resolved to the satisfaction of the end-users, the software is deemed ready for official release and deployment.

Output Testing or UAT is a critical step in the software development process, as it provides the final validation that the software meets the needs and expectations of the primary stakeholders. It ensures that the end-users have confidence in the quality and suitability of the software for their operational use. Ultimately, successful Output Testing signifies that the software is prepared for full-scale implementation in the intended environment.

6.2.3 Automation Testing

Automation Testing is a pivotal aspect of software testing that involves the use of automated tools and scripts to perform test cases and validate the functionality of a software application. Unlike manual testing, where testers execute tests manually, Automation Testing employs specialized software to automate the execution of predefined test scenarios. This approach brings several advantages, including increased testing speed, repeatability, and the ability to conduct tests on a large scale.

Automation Testing is particularly beneficial for tasks that are time-consuming, repetitive, or require a high degree of precision. It is well-suited for regression testing, where previously tested functionalities are retested after code changes to ensure that existing features remain unaffected. To implement Automation Testing, testers use testing frameworks and tools that provide scripting capabilities. These tools can simulate user interactions with the software, such as clicking buttons, entering data, and validating outcomes. Popular Automation Testing tools include Selenium, Appium, and TestComplete, among others.

However, Automation Testing is not a one-size-fits-all solution. It is most effective when applied to stable, well-defined functionalities that are unlikely to undergo frequent changes. Additionally, the initial setup of automated tests can be time-consuming, and maintenance efforts are required to update scripts as the application evolves.

Overall, Automation Testing is a valuable addition to the testing process, complementing manual testing efforts. It enhances testing efficiency, accelerates test cycles, and provides a reliable means of validating software functionality, ultimately contributing to the delivery of high-quality software products.

6.2.4 Selenium Testing

Selenium Testing is a widely used open-source tool and framework for automating web browsers. It provides a suite of tools for automating web browsers, allowing testers and developers to perform functional testing, regression testing, and load testing of web applications. Selenium supports various programming languages, including Java, C#, Python, Ruby, and JavaScript, making it versatile and accessible to a wide range of developers and testers.

One of the key advantages of Selenium is its ability to simulate user interactions with web elements. This includes actions like clicking buttons, filling out forms, navigating through pages, and validating the contents of web pages. Selenium allows testers to create automated test scripts that mimic these interactions, enabling the automation of repetitive and time-consuming testing tasks.

Selenium offers several components, with Selenium WebDriver being the most widely used. It provides a programming interface to create and execute test scripts. Selenium Grid is another component that enables the parallel execution of tests on different browsers and platforms, allowing for efficient testing across a diverse range of environments.

Additionally, Selenium can integrate with various testing frameworks and tools, providing additional functionalities and reporting capabilities. This makes it a powerful tool for building comprehensive and scalable test suites.

Selenium has gained immense popularity in the software testing community and is considered a go-to tool for web application testing. Its versatility, support for multiple programming languages, and robust automation capabilities make it an invaluable asset for testing web-based applications across different browsers and platforms.

Test Case 1: Login

Code

```
package assignment;

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

public class loginassign {

    WebDriver driver = null;
```

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

public void browser_is_open() {

    System.setProperty("webdriver.gecko.marionette", "C:\\Users\\ASUS\\eclipse-
workspace\\artifact3\\src\\test\\resources\\Driver\\geckodriver.exe");

    driver = new FirefoxDriver();

    driver.manage().window().maximize();

}

@And("user is on login page")

public void user_is_on_login_page() throws Throwable{

    driver.navigate().to("http://127.0.0.1:8000/login/");

    Thread.sleep(2000);

}

@When("user enters email and password")

public void user_enters_username_and_password() throws Throwable {

    driver.findElement(By.id("email")).sendKeys("alen@gmail.com");

    driver.findElement(By.id("password")).sendKeys("Alen@2013");

    Thread.sleep(3000);

}

@And("User clicks on login")

public void user_clicks_on_login() {

    driver.findElement(By.xpath("//button[@type='submit']")).click();

}

@Then("user is navigated to the home page")

public void user_is_navigated_to_the_home_page() throws Throwable {

    driver.findElement(By.className("nav-link")).isDisplayed();

    Thread.sleep(2000);

    driver.close();

    driver.quit();

}
```

```

}

}

```

Eg.Screenshot

```

Oct 25, 2023 9:55:21 AM cucumber.api.cli.Main run
WARNING: You are using deprecated Main class. Please use io.cucumber.core.cli.Main

Scenario: Check login is successful with valid credentials # src/test/resources/assign3/assignment3.feature:3
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
169820792865 geckodriver INFO Listening on 127.0.0.1:43884
1698207923110 mozzrunner:runner INFO Running command: "C:\\Program Files\\Mozilla Firefox\\firefox.exe" "--marionette" "--remote-debugging-port" "44194" "--remote ... /localh
console.warn: services.settings: Ignoring preference override of remote settings server
console.warn: services.settings: Allow by setting MOZ_REMOTE_SETTINGS_DEVTOOLS=1 in the environment
1698207923441 Marionette INFO Marionette enabled
Dynamically enable window occlusion 0
1698207923538 Marionette INFO Listening on port 7992
Read port: 7992
WebDriver BIDI listening on ws://127.0.0.1:44194
1698207923615 RemoteAgent WARN TLS certificate errors will be ignored for this session
DevTools listening on ws://127.0.0.1:44194/devtools/browser/23510fab-d893-490a-97a9-e122d0b11545
Given browser is open # assignment.loginassign.browser_is_open()
And user is on login page # assignment.loginassign.user_is_on_login_page()
When user enters email and password # assignment.loginassign.user_enters_username_and_password()
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
console.warn: LoginRecipes: "Falling back to a synchronous message for: http://127.0.0.1:8000."
And User clicks on login # assignment.loginassign.user_clicks_on_login()
1698207934942 RemoteAgent INFO Perform WebSocket upgrade for incoming connection from 127.0.0.1:8038
1698207934947 CDP WARN Invalid browser preferences for CDP. Set "fission.webContentIsolationStrategy" to 0 and "fission.bfcacheInParent" to false before Firefox starts.
1698207935021 Marionette INFO Stopped listening on port 7992
Dynamically enable window occlusion 1
Then user is navigated to the home page # assignment.loginassign.user_is_navigated_to_the_home_page()

1 Scenarios (1 failed)
5 Steps (1 failed, 4 passed)
0m14.726s

```

Eg.Test Report

Test Case 1					
Project Name: AssistiveGlobe – Online Ecommerce Site for Assistive Aids					
Login Test Case					
Test Case ID: 1			Test Designed By: Sruthymol Sunny		
Test Priority(Low/Medium/High): High			Test Designed Date: 26-10-2023		
Module Name: Login Screen			Test Executed By : Ms. Rini Kurian		
Test Title : Customer Login			Test Execution Date: 26-10-2023		
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 to login page		Dashboard should be displayed	Login page displayed	Pass
2	Provide valid email	alen@gmail.com	User should be login to home	Home page displayed	Pass

3	Provide valid password	Alen@2013	page		
4	Click Login button		Login page should be displayed	Redirect to login page	Pass
5	Logout and navigate to home page				
Post-Condition: Customer is validated with database and successfully login into account.					

Test Case 2:**Code****Screenshot****Test report****Minimum 4 test cases (1 login 3 functionalities)**

CHAPTER 6

IMPLEMENTATION

6.1 INTRODUCTION

Implementation in software development refers to the process of translating a design or concept into a working and functional system. It involves the actual coding, configuration, testing, and integration of various components to create a software application that meets the specified requirements. During the implementation phase, developers write the source code using programming languages and frameworks suitable for the project. They also set up databases, configure servers, and integrate third-party libraries or APIs if needed. Additionally, rigorous testing is performed to identify and fix any bugs or issues in the code. The implementation phase requires collaboration among developers, designers, and other stakeholders to ensure that the software aligns with the original vision and meets user expectations. Once the implementation is complete, the software undergoes further testing to verify its functionality and stability. Finally, the deployed system is ready for end-users to interact with, providing them with the intended features and functionalities. This phase is a critical step in the software development life cycle, as it brings the project from conceptualization to a tangible, operational product.

6.2 IMPLEMENTATION PROCEDURES

Implementation Procedures refer to the detailed steps and processes involved in transforming a software design into a functional and executable system. This phase of the software development life cycle (SDLC) is crucial for ensuring that the envisioned software solution is effectively translated into a working reality.

The first step in implementation involves setting up the development environment. This includes configuring necessary software tools, setting up version control systems, and establishing coding standards and guidelines to maintain consistency across the codebase.

Next, the actual coding process begins. Skilled developers write the source code using appropriate programming languages and frameworks based on the project's requirements. They follow design specifications and architectural patterns to ensure the code is structured, maintainable, and scalable.

Simultaneously, database design and development take place. This involves creating and configuring databases, defining tables and relationships, and implementing data storage and retrieval mechanisms. Database administrators work closely with developers to ensure seamless integration between the application and the database.

As the codebase grows, developers continuously test their code to identify and rectify bugs or issues. Unit testing verifies the correctness of individual components, while integration testing checks the interactions between different modules. Quality assurance and testing teams play a vital role in this phase, ensuring that the software meets the specified requirements and functions reliably.

Parallel to coding, system configurations are set up, including server configurations, network setups, and deployment strategies. This ensures that the software can run efficiently in the target environment.

Once the initial version of the software is complete, extensive testing is conducted. This includes functional testing to validate that the software behaves as expected, performance testing to assess its responsiveness, and security testing to identify vulnerabilities and risks.

Upon successful testing, the software is deployed to a staging or production environment.

Deployment procedures involve packaging the application, configuring servers, and ensuring proper integration with databases and external systems. Continuous monitoring and logging mechanisms are established to track the software's performance and detect any issues in real-time.

Throughout the implementation phase, collaboration and communication among development

teams, testers, designers, and other stakeholders are crucial. This ensures that the implementation aligns with the initial design and that any deviations or challenges are addressed promptly. In conclusion, the implementation procedures in software development encompass a series of systematic steps that bring the envisioned software solution to life. It involves coding, database development, testing, configuration, and deployment, all conducted with meticulous attention to detail and adherence to established best practices. Effective implementation lays the foundation for a robust, reliable, and functional software system.

6.2.1 User Training

User training is a structured process designed to equip individuals with the knowledge and skills needed to effectively use a particular software or technology. It serves as a vital component in ensuring that end-users can navigate, interact, and leverage the functionalities of the system proficiently. The objective of user training is to enhance user proficiency, productivity, and confidence while minimizing errors or inefficiencies in utilizing the software. This process involves the development of tailored training materials, hands-on practice sessions, and opportunities for users to seek clarifications or provide feedback. Additionally, user training extends beyond initial implementation, with ongoing support and resources available to assist users as they continue to engage with the software. Through user training, organizations aim to optimize the adoption and utilization of technology, ultimately contributing to the overall success of the software implementation.

6.2.2 Training on the Application Software

Training on application software involves a systematic process of educating individuals or users on how to effectively utilize a specific software program or application. This training is essential for enabling users to harness the full potential of the software, ensuring that they can navigate its features, functions, and workflows proficiently. It typically begins with an introduction to the software's interface and basic functionalities, gradually progressing to more advanced features and specialized tasks. Hands-on practice is a critical component, allowing users to interact with the software in a controlled environment. This practical experience helps solidify their understanding and build confidence in using the application. The training may be delivered through various methods, including in-person workshops, online webinars, e-learning platforms, or printed materials. Additionally, it's important to provide ongoing support, access to resources, and avenues for users to seek assistance or clarification even after the initial training is complete. Ultimately, effective training on application software empowers users to maximize their efficiency and productivity, contributing to the overall success of the software implementation within an organization.

6.2.3 System Maintenance

System maintenance, in the context of software development, refers to the ongoing activities and processes undertaken to ensure that a software application continues to operate smoothly and effectively after it has been deployed. It encompasses a range of tasks aimed at preserving the integrity, security, and functionality of the system over its entire lifecycle. This phase is critical for addressing issues that may arise post-implementation, such as software bugs, security vulnerabilities, and performance optimization. Maintenance activities include regular updates to fix identified bugs, applying security patches to protect against emerging threats, and implementing enhancements or new features as per user feedback or evolving requirements. Additionally, system maintenance involves monitoring the software's performance and stability, as well as managing its dependencies on external services or components. It may also encompass activities like data backups, disaster recovery planning, and capacity scaling to accommodate growing user demands. Effective system maintenance not only extends the longevity and reliability of the software but also ensures that it remains aligned with evolving technological standards and user needs. It is an integral phase in the software development life cycle, contributing to the overall sustainability and success of the application in a dynamic and ever-changing environment.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The AssistiveGlobe project, envisioned as an inclusive e-commerce platform for assistive aids, stands as a testament to the potential of technology in enhancing the lives of individuals with disabilities. By revolutionizing the accessibility and availability of assistive aids, this platform addresses a critical need in society. The user-centric approach, encompassing functionalities from seamless registration to personalized product recommendations, reflects a deep understanding of user requirements. The active involvement of administrators, mentors, and delivery teams further underscores the project's commitment to holistic support. The inclusion of an Assistive Aid Consultation module demonstrates a genuine dedication to helping users make informed choices, showcasing a human-centered design ethos. The platform's emphasis on user reviews and ratings fosters a community-driven environment, promoting transparency and trust. The commitment to system maintenance ensures that the platform will remain reliable and effective in the long run. In conclusion, AssistiveGlobe exemplifies the potential of technology to break barriers, empower individuals, and create a more inclusive and accessible world for everyone. This visionary project sets a commendable standard for leveraging e-commerce for the betterment of society, emphasizing that technology's true power lies in its ability to uplift and transform lives.

7.2 FUTURE SCOPE

The future scope of the AssistiveGlobe project is promising and holds potential for further innovation and expansion. One significant avenue lies in the continued enhancement of the personalized product recommendation feature utilizing Machine Learning (ML). As ML algorithms evolve and become more sophisticated, the platform can leverage this technology to provide even more accurate and tailored suggestions to users based on their preferences, purchase history, and user behavior patterns.

Furthermore, the platform could explore partnerships with assistive aid manufacturers and suppliers to offer a wider range of high-quality products. This could involve integrating real-time inventory systems and establishing direct links with manufacturers to ensure a diverse and up-to-date product catalog.

Expanding the platform's reach by introducing mobile applications for both Android and iOS devices could tap into a wider user base. Mobile applications provide greater accessibility, convenience, and flexibility for users, especially those with disabilities.

Introducing additional features such as augmented reality (AR) or virtual reality (VR) tools for users to virtually experience and assess assistive aids before purchase could revolutionize the shopping experience. This could be particularly valuable for products like mobility devices, where fit and comfort are crucial considerations.

Collaborations with healthcare professionals, occupational therapists, and specialists in assistive technology could lead to the development of specialized tools within the platform for assessing individual user needs and making personalized recommendations.

Implementing advanced analytics and data-driven insights could provide valuable information on user preferences, trends, and emerging needs. This data could be used to further refine product offerings and improve the overall user experience.

Lastly, exploring opportunities for international expansion and localization to cater to a global audience could position AssistiveGlobe as a leading player in the global market for assistive aids. In conclusion, the future of AssistiveGlobe is bright, with ample room for growth and innovation. By harnessing emerging technologies and collaborating with industry experts, the platform can continue to make significant strides in revolutionizing the accessibility and availability of assistive aids for individuals with disabilities.

CHAPTER 8

BIBLIOGRAPHY

REFERENCES:

- Pankaj Jalote, “Software engineering: a precise approach”
- Gary B. Shelly, Harry J. Rosenblatt, “System Analysis and Design”, 2009
- Ken Schwaber, Mike Beedle, Agile Software Development with Scrum, Pearson (2008)
- Roger S Pressman, “Software Engineering”
- IEEE Std 1016 Recommended Practice for Software Design Descriptions

WEBSITES:

- <https://www.shysha.in/>
- <https://www.w3schools.com/>
- <https://www.javatpoint.com/>
- <https://getbootstrap>

CHAPTER 9

APPENDIX

9.1 Sample Code

Login:

```
{% load static %}

{% load socialaccount %}
<!doctype html>
<html lang="en">
<head>
    <title>Login</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <link
href="https://fonts.googleapis.com/css?family=Lato:300,400,700&display=swap"
rel="stylesheet">

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">

    <link rel="stylesheet" href="{% static 'css/style.css' %}">
</head>
<body style="background-image: url('/static/images/bg.jpg'); background-size: cover; background-repeat: no-repeat;"></body>
<section class="ftco-section">
    <div class="container">
        <div class="row justify-content-center">
            <div class="col-md-6 text-center mb-5">
                <h1 class="mb-4"><a class="logo" href="#">AssistiveGlobe</a></h1>
                <h2 class="heading-section">Login</h2>
            </div>
        </div>
        <div class="row justify-content-center">
            <div class="col-md-6 col-lg-4">
                <div class="login-wrap py-5">
                    <div class="img d-flex align-items-center justify-content-center" style="background-image: url('/static/images/Logo.png');"></div>
                    <h3 class="text-center mb-0">Welcome</h3>
                    <form method="POST" class="login-form" id="signupForm"
action="">

                        {% csrf_token %}

                        <div class="form-group">
                            <div class="icon d-flex align-items-center justify-content-center"><span class="fa fa-envelope"></span></div>
```

```

        <input type="email" class="form-control" name="email"
id="email" placeholder="Email">
        <span id="email-validation" class="invalid"></span>
    </div>

    <div class="form-group">
        <div class="icon d-flex align-items-center justify-
content-center"><span class="fa fa-lock"></span></div>
        <input type="password" class="form-control"
id="password" name="pass" placeholder="Password">
        <span id="password-validation" class="invalid"></span>
        <div class="toggle-password">
            <i class="fa fa-eye" id="toggle-password-icon"
onclick="togglePasswordVisibility()"></i>
        </div>
    </div>

    <div class="form-group">
        <button type="submit" class="btn form-control btn-
primary rounded submit px-3" id="login-btn">Login</button>
    </div>

</form>
<div class="social-login">
    <a href="{% provider_login_url 'google'%}"next="/"
class="google btn d-flex justify-content-center align-items-center" style="color:
#eb7dbf;">
        
        Login with Google
    </a>
</div>

<div class="w-100 text-center mt-4 text">
    <p class="mb-0">Don't have an account?</p>
    <a href="{% url 'signup' %}">Sign Up</a>
</div>
</div>
</div>
</div>
</section>

```

```
<script src="{% static 'js/jquery.min.js' %}"></script>
<script src="{% static 'js/popper.js' %}"></script>
<script src="{% static 'js/bootstrap.min.js' %}"></script>
<script src="{% static 'js/main.js' %}"></script>

<script>
    // Toggle password visibility
    function togglePasswordVisibility() {
        var passwordInput = document.getElementById("password");
        var toggleIcon = document.getElementById("toggle-password-icon");

        if (passwordInput.type === "password") {
            passwordInput.type = "text";
            toggleIcon.classList.remove("fa-eye");
            toggleIcon.classList.add("fa-eye-slash");
        } else {
            passwordInput.type = "password";
            toggleIcon.classList.remove("fa-eye-slash");
            toggleIcon.classList.add("fa-eye");
        }
    }

    const urlParams = new URLSearchParams(window.location.search);
    const alertType = urlParams.get('alert');
    if (alertType === 'missing_fields') {
        showAlert('Please fill out all required fields', 'danger');
    } else if (alertType === 'invalid_login') {
        showAlert('Invalid Login Credential', 'danger');
    }
    function showAlert(message, alertType) {
        const alertDiv = document.createElement('div');
        alertDiv.classList.add('alert', `alert-${alertType}`, 'alert-top');
        alertDiv.textContent = message;

        const container = document.querySelector('.container');
        container.insertBefore(alertDiv, container.firstChild);

        setTimeout(function() {
            alertDiv.remove();
        }, 5000); // Remove after 5 seconds
    }
</script>
<style>
    /* toggle css */
    .toggle-password {
        position: absolute;
        right: 15px;
        top: 50%;
    }
</style>
```

```

        transform: translateY(-50%);
        cursor: pointer;
    }

    .alert-top {
        position: fixed;
        top: 0;
        left: 50%;
        transform: translateX(-50%);
        width: 100%;
        max-width: 500px;
        z-index: 1000;
    }
</style>

</body>
</html>

```

SignUp:

```

{% load static %}

{% load socialaccount %}
<!doctype html>
<html lang="en">
<head>
    <title>Signup</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <link
href="https://fonts.googleapis.com/css?family=Lato:300,400,700&display=swap"
rel="stylesheet">

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">

    <link rel="stylesheet" href="{% static 'css/style.css' %}">
</head>
<body style="background-image: url('/static/images/bg.jpg'); background-size: cover; background-repeat: no-repeat;"></body>
<section class="ftco-section">
    <div class="container">
        <div class="row justify-content-center">
            <div class="col-md-6 text-center mb-5">
                <h1 class="mb-4"><a class="logo" href="#">AssistiveGlobe</a></h1>
                <h2 class="heading-section">Sign Up</h2>
            </div>

```

```

    </div>
    <div class="row justify-content-center">
        <div class="col-md-6 col-lg-4">
            <div class="login-wrap py-5">
                <div class="img d-flex align-items-center justify-content-center" style="background-image: url('/static/images/Logo.png');"></div>
                <h3 class="text-center mb-0">Welcome</h3>
                <form method="POST" class="login-form" id="signupForm"
action="">

                    {% csrf_token %}
                    <div class="form-group">
                        <div class="icon d-flex align-items-center justify-content-center"><span class="fa fa-user"></span></div>
                        <input type="text" class="form-control" id="name"
name="name" placeholder="Name" onkeyup="validateName()">
                        <span id="name-validation" class="invalid"></span>
                    </div>

                    <div class="form-group">
                        <div class="icon d-flex align-items-center justify-content-center"><span class="fa fa-envelope"></span></div>
                        <input type="email" class="form-control" name="email"
id="email" placeholder="Email" onkeyup="validateEmail()">
                        <span id="email-validation" class="invalid"></span>
                    </div>

                    <div class="form-group">
                        <div class="icon d-flex align-items-center justify-content-center"><span class="fa fa-mobile"></span></div>
                        <input type="tel" class="form-control" id="mobile"
name="phone" placeholder="Mobile Number" onkeyup="validateMobile()">
                        <span id="mobile-validation" class="invalid"></span>
                    </div>

                    <div class="form-group">
                        <div class="icon d-flex align-items-center justify-content-center"><span class="fa fa-lock"></span></div>
                        <input type="password" class="form-control"
id="password" name="pass" placeholder="Password" onkeyup="validatePassword()">
                        <span id="password-validation" class="invalid"></span>
                        <div class="toggle-password">
                            <i class="fa fa-eye" id="toggle-password-icon"
onclick="togglePasswordVisibility()"></i>
                        </div>
                    </div>

                    <div class="form-group">

```



```

        <div class="icon d-flex align-items-center justify-
content-center"><span class="fa fa-lock"></span></div>
        <input type="password" class="form-control"
name="pass" id="confirm_password" placeholder="Confirm Password"
onkeyup="validateConfirmPassword()">
        <span id="confirm-password-validation" class="invalid"
></span>
    </div>

    <div class="form-group">
        <button type="submit" class="btn form-control btn-
primary rounded submit px-3" id="signup-btn">Sign Up</button>
    </div>

</form>

<div class="social-login">
    <a href="{% provider_login_url 'google'%}?next=/"
class="google btn d-flex justify-content-center align-items-center" style="color:
#eb7dbf;">
        
        SignUp with Google
    </a>
</div>

<div class="w-100 text-center mt-4 text">
    <p class="mb-0">Already have an account?</p>
    <a href="{% url 'login' %}">Sign In</a>
</div>
</div>
</div>
</div>
</section>

<script src="{% static 'js/jquery.min.js' %}"></script>
<script src="{% static 'js/popper.js' %}"></script>
<script src="{% static 'js/bootstrap.min.js' %}"></script>
<script src="{% static 'js/main.js' %}"></script>

<script>
    // Validation functions

    //Name valiadtion
    function validateName() {

```

```
var nameInput = document.getElementById("name");
var nameValidation = document.getElementById("name-validation");
var pattern = /^[A-Z][a-zA-Z\s]*{1,25}$/; // Each word starts with a capital
letter, allow only letters and spaces, max 25 characters

if (nameInput.value === "") {
    nameValidation.textContent = "Name cannot be blank!";
    nameValidation.className = "invalid";
} else if (!pattern.test(nameInput.value)) {
    nameValidation.textContent = "Invalid name format. Ensure each word starts
with a capital letter and contains only letters and spaces.";
    nameValidation.className = "invalid";
} else {
    nameValidation.textContent = "Valid name.";
    nameValidation.className = "valid";
}
updateSubmitButton();
}
```

//Email validation

```
function validateEmail() {
    var emailInput = document.getElementById("email");
    var emailValidation = document.getElementById("email-validation");
    var pattern = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/;

    if (emailInput.value === "") {
        emailValidation.textContent = "Email is required!";
        emailValidation.className = "invalid";
    } else if (!pattern.test(emailInput.value)) {
        emailValidation.textContent = "Please enter a valid email address!";
        emailValidation.className = "invalid";
    } else {
        emailValidation.textContent = "Valid email.";
        emailValidation.className = "valid";
    }
    updateSubmitButton();
}
```

//Mobile validation

```
function validateMobile() {
    var mobileInput = document.getElementById("mobile");
    var mobileValidation = document.getElementById("mobile-validation");
    var pattern = /^[6-9]\d{9}$/;
```

```
    if (mobileInput.value === "") {
        mobileValidation.textContent = "Mobile number is required!";
        mobileValidation.className = "invalid";
    } else if (!pattern.test(mobileInput.value)) {
        mobileValidation.textContent = "Please enter a valid 10-digit mobile
number";
        mobileValidation.className = "invalid";
    } else {
        mobileValidation.textContent = "Valid mobile number.";
        mobileValidation.className = "valid";
    }
    updateSubmitButton();
}

//Password validation
function validatePassword() {
    var passwordInput = document.getElementById("password");
    var passwordValidation = document.getElementById("password-validation");
    var pattern = /^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[^a-zA-Z0-9]).{8,}$/;

    if (passwordInput.value === "") {
        passwordValidation.textContent = "Password is required!";
        passwordValidation.className = "invalid";
    } else if (!pattern.test(passwordInput.value)) {
        passwordValidation.textContent = "Password must have at least 8
characters, including uppercase and lowercase letters, numbers, and special
characters.";
        passwordValidation.className = "invalid";
    } else {
        passwordValidation.textContent = "Valid password format.";
        passwordValidation.className = "valid";
    }
    updateSubmitButton();
}

//ConfirmPassword validation
function validateConfirmPassword() {
    var passwordInput = document.getElementById("password");
    var confirmPasswordInput = document.getElementById("confirm_password");
    var confirmPasswordValidation = document.getElementById("confirm-password-
validation");

    if (confirmPasswordInput.value === "") {
        confirmPasswordValidation.textContent = "Confirm Password is required!";
        confirmPasswordValidation.className = "invalid";
    } else if (confirmPasswordInput.value !== passwordInput.value) {
```

```
        confirmPasswordValidation.textContent = "Passwords do not match!";
        confirmPasswordValidation.className = "invalid";
    } else {
        confirmPasswordValidation.textContent = "Passwords match.";
        confirmPasswordValidation.className = "valid";
    }
    updateSubmitButton();
}

// Update submit button state based on field validations
function updateSubmitButton() {
    var invalidSpans = document.querySelectorAll(".invalid");
    var submitButton = document.getElementById("signup-btn");

    if (invalidSpans.length === 0) {
        submitButton.disabled = false;
    } else {
        submitButton.disabled = true;
    }
}

// Form validation
function validateForm() {
    validateName();
    validateEmail();
    validateMobile();
    validatePassword();
    validateConfirmPassword();

    var invalidSpans = document.querySelectorAll(".invalid");
    if (invalidSpans.length > 0) {
        alert("Please correct the highlighted fields.");
        return false;
    }

    return true; // Form is valid
}

// Toggle password visibility
function togglePasswordVisibility() {
    var passwordInput = document.getElementById("password");
    var toggleIcon = document.getElementById("toggle-password-icon");

    if (passwordInput.type === "password") {
        passwordInput.type = "text";
        toggleIcon.classList.remove("fa-eye");
        toggleIcon.classList.add("fa-eye-slash");
    }
}
```

```
    } else {
        passwordInput.type = "password";
        toggleIcon.classList.remove("fa-eye-slash");
        toggleIcon.classList.add("fa-eye");
    }
}

const urlParams = new URLSearchParams(window.location.search);
const alertType = urlParams.get('alert');
if (alertType === 'email_already_existing') {
    showAlert('Email already exists', 'danger');
} else if (alertType === 'registered'){
    showAlert('Registered Successfully', 'success');
} else if (alertType === 'missing_fields'){
    showAlert('Please fill out all required fields', 'danger');
}

function showAlert(message, alertType) {
    const alertDiv = document.createElement('div');
    alertDiv.classList.add('alert', `alert-${alertType}`, 'alert-top');
    alertDiv.textContent = message;

    const container = document.querySelector('.container');
    container.insertBefore(alertDiv, container.firstChild);

    setTimeout(function() {
        alertDiv.remove();
    }, 5000); // Remove after 5 seconds
}

</script>

<style>
    /* toggle css */
    .toggle-password {
        position: absolute;
        right: 15px;
        top: 50%;
        transform: translateY(-50%);
        cursor: pointer;
    }
    .alert-top {
        position: fixed;
        top: 0;
        left: 50%;
        transform: translateX(-50%);
        width: 100%;
        max-width: 500px;
        z-index: 1000;
    }
</style>
```

```
    }  
</style>
```

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

Home:

```
{% 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="">  
    <title>AssistiveGlobe</title>  
    <!-- Favicon -->  
    <link rel="icon" type="image/x-icon" href="{% static 'assets/favicon.ico' %}">  
    <!-- Bootstrap icons -->  
    <link href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.5.0/font/bootstrap-  
icons.css" rel="stylesheet">  
    <!-- Core theme CSS (includes Bootstrap) -->  
    <link href="{% static 'css/styles.css' %}" rel="stylesheet">  
    <style>  
        /* Custom Styles */  
        body {  
            font-family: 'Arial', sans-serif;  
            background-color: #f8f8f8;  
        }  
        .navbar {  
            background-color: #343a40;  
        }  
        .navbar-brand {  
            font-family: 'Garamond', sans-serif;  
            color: #ffffff;  
            font-size: 24px;  
        }  
        .navbar-nav .nav-link {  
            color: #ffffff;  
        }  
        .navbar-toggler-icon {  
            background-color: #ffffff;  
        }  
        .header {
```

```
        background-image: url('/static/images/home_bg.png');
        background-size: contain;
        background-repeat: no-repeat;
        background-position: right top;
        min-height: 500px;
        display: flex;
        align-items: center;
        color: #ffffff;
    }

    .header-content {
        flex: 1;
        padding: 0 20px;
    }

    .header h1 {
        font-size: 60px;
        color: #333333;
    }

    .header p {
        font-size: 18px;
        color: #666666;
    }

    .section-title {
        font-size: 36px;
        text-align: center;
        margin-bottom: 40px;
        color: #333333;
    }

    .card {
        border: none;
        transition: transform 0.2s;
    }

    .card:hover {
        transform: scale(1.05);
    }

    .logo {
        width: 100px;
        height: auto;
    }

    .card-title {
        font-size: 20px;
        color: #333333;
    }

    .card-text {
        font-size: 16px;
        color: #666666;
    }
}
```

```

        .btn-outline-dark {
            color: #343a40;
            border-color: #343a40;
        }
        .btn-outline-dark:hover {
            background-color: #343a40;
            color: #ffffff;
        }
        .footer {
            background-color: #343a40;
            color: #ffffff;
            text-align: center;
            padding: 20px 0;
        }
    </style>
</head>
<body>
    <!-- Navigation -->
    <nav class="navbar navbar-expand-lg">
        <div class="container px-4 px-lg-5">
            <a class="navbar-brand" href="#!">
                
            </a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"
aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarSupportedContent">
                <ul class="navbar-nav ms-auto mb-2 mb-lg-0">
                    <li class="nav-item"><a class="nav-link" href="{% url 'home'
%}">Home</a></li>
                    {% if not user.is_authenticated or user.is_authenticated and
not user.is_superuser %}
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'about' %}">About</a>
                    </li>
                    {% endif %}
                    <!-- <li class="nav-item"><a class="nav-link" href="{% url
'about' %}">About</a></li> -->

                    {% if user.is_superuser %}
                    <li class="nav-item"><a class="nav-link" href="{% url
'dashboard' %}">Dashboard</a></li>

                    <!-- Show this button only to superusers -->

```



```

        <li class="nav-item"><a class="nav-link" href="{% url
'product' %}">Products</a></li>
        <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle" id="navbarDropdown"
href="#" role="button" data-bs-toggle="dropdown" aria-expanded="false">Shop By
Category</a>

            <ul class="dropdown-menu" aria-
labelledby="navbarDropdown">
                <li><a class="dropdown-item"
href="#">Category</a></li>

                <li><hr class="dropdown-divider"></li>
                <li><a class="dropdown-item" href="{% url 'wheelchair'
%}">Wheelchair</a></li>

                <li><a class="dropdown-item" href="{% url 'walker'
%}">Walker</a></li>

                <li><a class="dropdown-item" href="{% url 'crutches'
%}">Crutches</a></li>

            </ul>
        </li>
        <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle" id="navbarDropdown"
href="#" role="button" data-bs-toggle="dropdown" aria-expanded="false">Welcome ,
{{ user.name }} (Admin)</a>
            <ul class="dropdown-menu" aria-
labelledby="navbarDropdown">
                <li><a class="dropdown-item" href="{% url 'logout'
%}">Logout</a></li>

            </ul>
        </li>
    {% elif user.is_authenticated %}

        <li class="nav-item"><a class="nav-link" href="{% url
'product' %}">Products</a></li>
        <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle" id="navbarDropdown"
href="#" role="button" data-bs-toggle="dropdown" aria-expanded="false">Shop By
Category</a>

            <ul class="dropdown-menu" aria-
labelledby="navbarDropdown">
                <li><hr class="dropdown-divider"></li>
                <li><a class="dropdown-item" href="{% url 'wheelchair'
%}">Wheelchair</a></li>

                <li><a class="dropdown-item" href="{% url 'walker'
%}">Walker</a></li>

                <li><a class="dropdown-item" href="{% url 'crutches'
%}">Crutches</a></li>

            </ul>
        </li>
        <li class="nav-item dropdown">

```

```

        <a class="nav-link dropdown-toggle" id="navbarDropdown"
href="#" role="button" data-bs-toggle="dropdown" aria-expanded="false">Welcome ,
{{ user.name }}</a>
        <ul class="dropdown-menu" aria-
labelledby="navbarDropdown">
            <li><a class="dropdown-item" href="{% url 'my_profile'
%}">My Profile</a></li>
            <li><a class="dropdown-item" href="{% url 'logout'
%}">Logout</a></li>

        </ul>
    </li>
    {% else %}
    <li class="nav-item"><a class="nav-link">Welcome
Guest</a></li>
    <li class="nav-item"><a class="nav-link" href="{% url 'login'
%}">Login</a></li>
    <li class="nav-item"><a class="nav-link" href="{% url 'signup'
%}">SignUp</a></li>
    {% endif %}
</ul>
</div>
</div>
</nav>

<!-- Header-->
<header class="bg-white header">
    <div class="container px-4 px-lg-5">
        <div class="header-content">
            <h1 class="display-4 fw-bolder">Empowering Independence</h1>
            <p class="lead fw-normal">Your Source for Assistive Aids and
Accessibility Solutions</p>
        </div>
    </div>
</header>

<!-- Section -->
<section class="py-5">
    <div class="container px-4 px-lg-5 mt-5">
        <h2 class="section-title">Featured Products</h2>
        <div class="row gx-4 gx-lg-5 row-cols-2 row-cols-md-3 row-cols-xl-4
justify-content-center">
            <!-- Product cards go here -->
            <!-- <div class="col mb-5">
                <div class="card h-100">
                    

```

```

        <div class="card-body p-4">
            <div class="text-center">
                <h5 class="card-title fw-bolder" style="font-size:
15px;">Mobility Kart One Key Automatic Folding Ultra-Lightweight Power Electric
Wheelchair with Wireless Remote & Electromagnetic Brakes</h5>
                <p class="card-text" style="font-size:
18px;">₹1,09,998</p>
            </div>
        </div>
        <div class="card-footer p-4 pt-0 border-top-0 bg-
transparent">
            <div class="text-center"><a class="btn btn-outline-
dark mt-auto" href="#">View options</a></div>
        </div>
    </div> -->
    <!-- Add more product cards here -->
    <div class="col mb-5">
        <div class="card h-100">
            
            <div class="card-body p-4">
                <div class="text-center">
                    <h5 class="card-title fw-bolder" style="font-size:
15px;">MCP Jindal 4 Folding Sections Folding Walking Stick (Black)</h5>
                    <p class="card-text" style="font-size:
18px;">₹474</p>
                </div>
            </div>
            <div class="card-footer p-4 pt-0 border-top-0 bg-
transparent">
                <div class="text-center"><a class="btn btn-outline-
dark mt-auto" href="{% url 'walker' %}">View More</a></div>
            </div>
        </div>
    </div>
    <div class="col mb-5">
        <div class="card h-100">
            
            <div class="card-body p-4">
                <div class="text-center">
                    <h5 class="card-title fw-bolder" style="font-size:
15px;">FCS Listing Assistive Device Instant fit BTE F-138 Hearing Aid, Pair of
Both Ear Sound Amplifier for Seniors and Adults with Moderate Hearing Loss</h5>
                    <p class="card-text" style="font-size:
18px;">₹4,950</p>
                </div>
            </div>
        </div>
    </div>

```



```
.navbar-nav .btn {  
    margin-right: 10px;  
}  
  
.cart-total {  
    display: inline-block;  
    margin-left: 5px;  
}  
  
.category-section {  
    margin-bottom: 40px;  
}  
  
.category-section h2 {  
    margin-bottom: 20px;  
}  
  
.product-card {  
    border: 1px solid #ddd;  
    border-radius: 5px;  
    background-color: #fff;  
    box-shadow: 0 4px 8px rgba(0,0,0,0.1);  
    margin-bottom: 20px;  
}  
  
.product-card img {  
    max-width: 100%;  
    height: auto;  
    border-bottom: 1px solid #ddd;  
}  
  
.product-card-body {  
    padding: 20px;  
}  
  
.product-card-title {  
    font-size: 1.25rem;  
    margin-bottom: 10px;  
}  
  
.product-card-price {  
    font-size: 1.25rem;  
}  
  
.product-card-actions {  
    text-align: right;  
}
```

```

.button {
  display: inline-block;
  padding: 10px 20px;
  font-size: 16px;
  background-color: #d9534f;
  color: white;
  text-align: center;
  text-decoration: none;
  border-radius: 4px;
  cursor: pointer;
  border: none;
  transition: background-color 0.3s;
}

.button:hover {
  background-color: #c9302c;
}
</style>

</head>
<body>

  <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <a class="navbar-brand" href="{% url 'product' %}">AssistiveGlobe</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-
expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>

    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav mr-auto">
        <li class="nav-item active">
          <a class="nav-link" href="{% url 'home' %}">Home<span class="sr-
only">(current)</span></a>
        </li>
      </ul>
      <div class="form-inline my-2 my-lg-0" style="margin-right:420px;">
        <form action="{% url 'product_search' %}" method="get">
          <input id="search-input" class="form-control mr-sm-2" type="text"
name="q" placeholder="Search" aria-label="Search" size="15">
          <button type="submit" class="btn btn-success my-2 my-sm-0"
id="search-button" style="background-color: #28a745; border-color:
#28a745;">Search</button>
        </form>
      </div>

      <div class="form-inline my-2 my-lg-0">

```

```

        {% if user.is_authenticated %}
            <a href="{% url 'cart' %}">
                
            </a>

            <!-- Show this button only if the user is logged in -->
            <a href="{% url 'logout' %}" class="btn btn-warning mx-
2">Logout</a>
        {% else %}
            <!-- Show this button only if the user is not logged in -->
            <a href="{% url 'login' %}" class="btn btn-warning mx-2">Login</a>
        {% endif %}
    </div>
</div>
</nav>
<div>
    {% block content %}
        <div class="container">
            {% for category, products in sorted_products.items %}
                <section class="category-section">
                    <h2>{{ category }}</h2>
                    <div class="row">
                        {% for product in products %}
                            <div class="col-lg-4">
                                

                                <div class="box-element product">
                                    <h6><strong>{{ product.name }}</strong></h6>
                                    <hr>
                                    <div style="float: left;">
                                        {% if not request.user.is_superuser %}
                                            <button class="btn btn-outline-
secondary add-btn">
                                                <a href="{% url 'add_to_cart'
product.id %}">Add to Cart</a>
                                            </button>
                                        {% endif %}
                                        <a class="btn btn-outline-success"
href="{% url 'product_detail' product.id %}">View</a>
                                    </div>
                                    <div style="float: right;">
                                        <h4 style="display: inline-
block;"><strong>Rs {{ product.price|floatformat:2 }}</strong></h4>
                                    </div>
                                    <div style="clear: both;"><!-- Clear
the floats -->
                                </div>
                            </div>
                        {% endfor %}
                    </div>
                </section>
            {% endfor %}
        </div>
    {% endblock %}
</div>

```

```

        {% endfor %}
    </div>
</section>
{% endfor %}

</div>
{% endblock content %}
</div>

<script src="https://code.jquery.com/jquery-3.4.1.slim.min.js"
integrity="sha384-
J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n"
crossorigin="anonymous"></script>

<script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
crossorigin="anonymous"></script>

<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"
integrity="sha384-
wfSDF2E50Y2D1uUdj003uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl30g8ifwB6"
crossorigin="anonymous"></script>
</body>
</html>

```

Cart:

```

{% load static %}
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Cart</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.1.0/css/all.min.css">
    <link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.4/dist/umd/popper.min.js"></s
cript>

```



```
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

<style>
/* Add your CSS styles here */
.container {
    font-family: Arial, sans-serif;
    margin: 20px;
}

.card {
    border: 1px solid #ddd;
    border-radius: 5px;
    background-color: #fff;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    margin-bottom: 20px;
    padding: 10px;
    text-align: center;
}

.card img {
    max-width: 100%;
    max-height: 150px;
    height: auto;
    border-bottom: 1px solid #ddd;
}

.cart-table {
    width: 70%;
    margin: 0 auto;
    /* Center the table */
}

table {
    width: 100%;
    border-collapse: collapse;
    margin-bottom: 20px;
}

th,
td {
    padding: 8px;
    text-align: left;
    border-bottom: 1px solid #ddd;
}

input[type="number"] {
```

```
        width: 50px;
    }

    button {
        padding: 5px 10px;
        margin-right: 5px;
    }

    .checkout-btn {
        background-color: #28a745;
        /* Green color */
        color: #fff;
        border: none;
        border-radius: 5px;
        padding: 10px 20px;
        font-size: 16px;
        cursor: pointer;
    }

    .checkout-btn:hover {
        background-color: #218838;
        /* Darker green on hover */
    }

    .remove-btn {
        background-color: #dc3545;
        color: #fff;
        border: none;
        border-radius: 5px;
        padding: 5px 10px;
        font-size: 12px;
        cursor: pointer;
        margin-top: 50px;
    }

    .remove-btn:hover {
        background-color: #bb2d3b;
    }

    .sub-total {
        font-size: 18px;
        font-weight: bold;
    }

    .empty-cart-msg {
        font-size: 24px;
        text-align: center;
        margin-top: 50px;
    }
```

```

.logout-btn {
    position: absolute;
    top: 10px;
    right: 10px;
    background-color: #f0f3f7;
    color: #fff;
    border: none;
    border-radius: 5px;
    padding: 5px 10px;
    font-size: 12px;
    text-decoration: none;
    cursor: pointer;
}
.icon {
    margin-right: 5px;
}
</style>
</head>

<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="#">AssistiveGlobe</a>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNav"
            aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse justify-content-end" id="navbarNav">
            <ul class="navbar-nav">
                <li class="nav-item">
                    <a class="nav-link" href="{% url 'logout' %}">Logout</a>
                </li>
            </ul>
        </div>
    </nav>

    <div class="container mt-4">
        <a href="{% url 'product' %}" class="btn btn-link"><i class="fas fa-arrow-
left icon"></i> Continue Shopping</a>
        {% if cart_items %}
        <div class="cart-table">
            <table>
                <thead>
                    <tr>

```

```

        <th>Product</th>
        <th>Price</th>
        <th>Quantity</th>
        <th>Total</th>
        <th></th>
    </tr>
</thead>
<tbody>
    {% for item in cart_items %}
    <tr>
        <td>
            <div class="card">
                
                <div class="card-details">
                    <h3>{{item.product.name}}</h3>
                </div>
            </div>
        </td>
        <td>Rs.{{ item.product.price|floatformat:2 }}</td>
        <td>
            <div class="quantity">
                <button class="change-quantity" data-cart_id="{{
item.id }}"
                    data-change="decrease">-</button>
                <input type="number" value="{{ item.quantity }}"
min="1" name="quantity_{{ item.id }}"
                    id="quantity_{{ item.id }}">
                <button class="change-quantity" data-cart_id="{{
item.id }}"
                    data-change="increase">+</button>
            </div>
        </td>
        <td class="total" data-cart_id="{{ item.id }}">Rs.{{
item.get_total }}</td>
        <td>
            <a href="{% url 'remove_from_cart' item.id %}"
class="remove-btn">Remove</a>
        </td>
    </tr>
    {% endfor %}
    <tr>
        <td colspan="5" class="sub-total">Subtotal: Rs.{{
total_price|floatformat:2 }}</td>
    </tr>
</tbody>
</table>
<div class="text-center">

```

```

        <button id="checkout" class="btn btn-primary checkout-
btn">Checkout</button>
    </div>
</div>
{% else %}
<div class="empty-cart-msg">Cart is empty !!!</div>
{% endif %}
</div>
<script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
<script>
    var total = 0
    function removeFromCart(cartid) {
        var url = '{% url "remove_from_cart" 0 %}'.replace('0', cartid);

        // Confirm if the user really wants to remove the item
        if (confirm("Are you sure you want to remove this item from the
cart?")) {
            window.location.href = url;
        }
    }

    jQuery(document).ready(function ($) {
        $(document).ready(function () {

            $('.change-quantity').click(function () {
                var cartId = $(this).data('cart_id');
                var change = $(this).data('change');
                var inputField = $(this).parent().find('input');
                var currentValue = parseInt(inputField.val());

                if (change === 'increase') {
                    inputField.val(currentValue + 1);
                } else if (change === 'decrease' && currentValue > 1) {
                    inputField.val(currentValue - 1);
                }

                $.ajax({
                    url: '{% url "update_cart_quantity" 0 %}'.replace('0',
cartId),

                    method: 'POST',
                    data: {
                        'quantity': inputField.val(),
                        'csrfmiddlewaretoken': '{{ csrf_token }}'
                    },
                    success: function (data) {
                        if (data.success) {
                            var newTotal = parseFloat(data.new_total);
                            $('.total[data-cart_id="' + cartId +
'"]')>

```

```

        // Update the subtotal
        var newSubtotal = parseFloat(data.new_subtotal);
        $('.sub-total').text('Subtotal: Rs.' +
newSubtotal.toFixed(2));

        total = newSubtotal.toFixed(2);
    }
}
});
});

// Function to handle input field changes
$('.input[type="number"]').change(function () {
    var cartId = $(this).closest('tr').find('.change-
quantity').data('cart_id');
    var inputField = $(this);
    var newValue = parseInt(inputField.val());

    if (newValue < 1) {
        inputField.val(1);
        newValue = 1;
    }

    $.ajax({
        url: '% url "update_cart_quantity" 0 %}'.replace('0',
cartId),

        method: 'POST',
        data: {
            'quantity': newValue,
            'csrfmiddlewaretoken': '{{ csrf_token }}'
        },
        success: function (data) {
            if (data.success) {
                var newTotal = parseFloat(data.new_total);
                $('.total[data-cart_id="' + cartId +
'"').text('Rs.' + newTotal.toFixed(2));

                // Update the subtotal
                var newSubtotal = parseFloat(data.new_subtotal);
                $('.sub-total').text('Subtotal: Rs.' +
newSubtotal.toFixed(2));

                total = newSubtotal.toFixed(2);
            }
        }
    });
});
$('#checkout').click(function(){
    // Get the subtotal text content
    var subtotalText = document.querySelector('.sub-total').textContent;

```

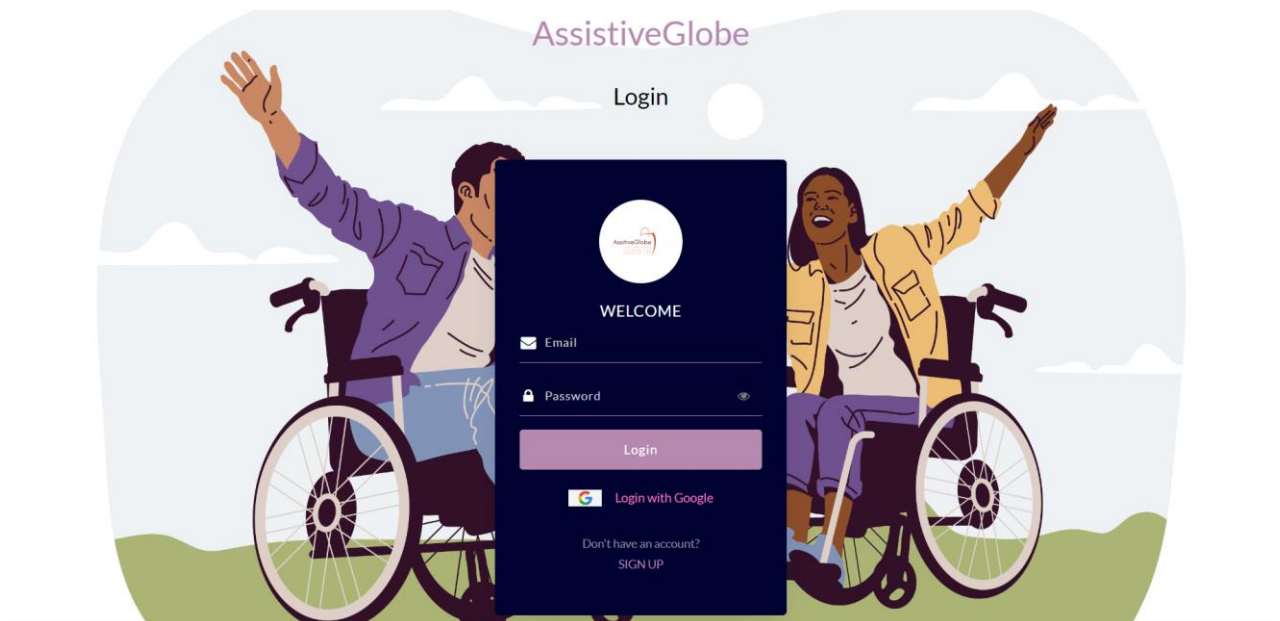
```
// Extract the numeric value from the string
var total = parseFloat(subtotalText.match(/\d+\.\d+/)[0]);

// Now, 'total' contains the total price
window.location.href = "{% url 'checkout' 0 %}".replace("0",total);
})
});
});
</script>

</body>

</html>
```

9.1 Screen Shots





Empowering Independence

Your Source for Assistive Aids and Accessibility Solutions



AssistiveGlobe Home

Search



Logout

Wheelchair



Electric Wheelchair

[Add to Cart](#) [View](#)

Rs 15000.00



Manual Wheelchair

[Add to Cart](#) [View](#)

Rs 4389.00



Powered Wheelchair

[Add to Cart](#) [View](#)

Rs 5230.00



Pediatric Wheelchair

[Add to Cart](#) [View](#)

Rs 3245.00



Pediatric Wheelchair


[Add to Cart](#) [View](#)

Rs 5999.00

AssistiveGlobe

Logout

[← Continue Shopping](#)

Product	Price	Quantity	Total
 Manual Wheelchair	Rs.4389.00	<input type="text" value="1"/>	Rs. Remove
Subtotal: Rs.4389.00			
Checkout			

