**Digital Commerce Empowerment Ecosystem in rural areas (DCEE)**

*Project Report Submitted by*

**Josna Mary Thomas**

**Reg. No.: AJC20MCA-I041**

*In Partial fulfillment for the Award of the Degree Of*

**MASTER OF COMPUTER APPLICATIONS**

**(INTEGRATED MCA)**

**APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY**



**AMAL JYOTHI COLLEGE OF ENGINEERING**

**KANJIRAPPALLY**

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

# 2024-2025

## DEPARTMENT OF COMPUTER APPLICATIONS

### AMAL JYOTHI COLLEGE OF ENGINEERING

**KANJIRAPPALLY**



**CERTIFICATE**

This is to certify that the Project report, “**Digital Commerce Empowerment Ecosystem in rural areas (DCEE)”** is the bona fide work of **JOSNA MARY THOMAS (Regno: AJC20MCA-I041)** in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2024-25.

**Ms. Sona Maria Sebastian Mr. Binumon Joseph**

**Internal Guide Coordinator**

**Rev. Fr. Dr. Rubin Thottupurathu Jose**

**Head of the Department**

**DECLARATION**

I hereby declare that the project report **“Digital Commerce Empowerment Ecosystem in rural areas (DCEE)”** 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 2024-2025.

**Date: JOSNA MARY THOMAS**

**KANJIRAPPALLY Reg: AJC20MCA-I041**

# 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. Roy Abraham Pazhayaparampil** 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 **Mr. Binumon Joseph** **(Assistant Professor)** 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 **Ms. Sona Maria Sebastian (Assistant Professor)** 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.

JOSNA MARY THOMAS

# ABSTRACT

The Digital Commerce Empowerment Ecosystem (DCEE) project is a transformative initiative aimed at digitally uplifting small and local Indian businesses, empowering them to thrive in an increasingly competitive online market. The platform offers a comprehensive suite of tools and features designed to enhance digital presence, streamline operations, and foster customer relationships. By providing a customizable digital storefront, secure payment gateways, and an intuitive user interface, DCEE enables businesses to cultivate trust and effectively engage with their customers.

At the core of DCEE's functionality is its advanced analytics module, powered by the RoBERTa model, which facilitates predictive analytics for inventory management. This feature allows businesses to accurately forecast market demands, optimize stock levels, and reduce waste, ultimately leading to improved revenue and profitability. The platform is structured to cater to various user roles—storefront owners, customers, and administrators—each with tailored functionalities such as profile management, transaction oversight, and real-time data insights.

Moreover, DCEE incorporates a robust chatbot feature that provides users with immediate assistance, guiding them to appropriate resources and helping to resolve common issues. This promotes a seamless user experience, encouraging greater engagement and retention. DCEE also addresses critical challenges such as regulatory compliance, payment facilitation, and digital literacy gaps, ensuring that small businesses are equipped to navigate the complexities of the digital landscape.

In addition to these features, DCEE fosters community resilience by enabling small businesses to leverage technology for growth. By prioritizing accessibility and user-friendliness, the platform aims to create a supportive ecosystem where local entrepreneurs can compete effectively and contribute to economic development in their communities. Ultimately, DCEE not only serves as a catalyst for individual business success but also enriches the entrepreneurial spirit that is vital to India's economic landscape. Through this holistic approach, DCEE is poised to redefine how small businesses operate and succeed in the digital age, making a significant impact on the broader economy.

**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** | | **DATA BASE 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

# CHAPTER 1

# INTRODUCTION

### PROJECT OVERVIEW

The Digital Commerce Empowerment Ecosystem (DCEE) is an innovative platform designed to empower small and local Indian businesses by offering essential tools to navigate the digital economy. Focusing on bridging the digital divide, DCEE provides solutions for establishing a digital presence, fostering customer trust, managing rural payments, optimizing inventory, and supporting digital literacy. Key features include a predictive analytics module powered by the RoBERTa model for inventory insights, a chatbot for user guidance, and a payment module for streamlined transactions. Built with a scalable tech stack, DCEE equips business owners, customers, and admins with robust tools to manage profiles, storefronts, and transactions, creating a dynamic and accessible environment for local business growth.

### PROJECT SPECIFICATION

The Digital Commerce Empowerment Ecosystem (DCEE) supports small Indian businesses by providing digital tools to enhance visibility, optimize operations, and build customer trust. The platform features three user roles:

* Digital Storefront Owner: Manages business profiles, adds products, views customer interactions, and processes payments.
* Customer: Registers, shops, manages cart, and completes secure transactions, interacting smoothly with local businesses.
* Admin: Oversees user profiles, manages transactions, and ensures platform security.

DCEE aims to drive digital transformation, support small businesses, and promote economic growth in India’s local markets.

# CHAPTER 2

# SYSTEM STUDY

# 

### INTRODUCTION

The Digital Commerce Empowerment Ecosystem (DCEE) is designed to revolutionize small and local Indian businesses by addressing the specific challenges they face in the digital landscape. Many of these enterprises struggle with limited access to technology and digital tools, which inhibits their ability to engage customers, streamline operations, and enhance their market presence. DCEE offers a holistic solution that empowers storefront owners with essential tools for online management, including customer profiles, product listings, and secure payment processing. This integration not only simplifies their operations but also enhances their ability to reach a wider audience.

DCEE features an analytics module that harnesses predictive insights, helping businesses manage inventory effectively and avoid the risks associated with overstocking and stockouts. The use of advanced machine learning tools, such as the RoBERTa model, enables accurate demand forecasting, enhancing decision-making capabilities. Additionally, the platform incorporates a user-friendly chatbot that assists users with inquiries and navigational support, significantly reducing the digital literacy barrier that many small businesses owners face.

By facilitating rural payment options and ensuring compliance with regulatory requirements, DCEE fosters customer trust and accessibility. This ecosystem not only streamlines operations but also nurtures entrepreneurial spirit by providing training and resources that promote digital literacy. Ultimately, DCEE aspires to bridge the digital divide, empowering small businesses to thrive, contribute to local economies, and sustain their growth in a rapidly evolving digital marketplace.

### EXISTING SYSTEM

### The existing systems for small and local Indian businesses are often fragmented, relying on manual processes for inventory management, customer interactions, and sales tracking. Many businesses use basic spreadsheets or paper records, leading to inefficiencies and difficulties in real-time stock monitoring, resulting in overstocking and stockouts. Additionally, these businesses struggle with establishing an online presence due to the complexity and cost of current e-commerce platforms. Limited payment processing options and lack of digital literacy further hinder their operations. Overall, the existing systems do not provide the comprehensive support needed for small businesses to thrive in the digital economy, highlighting the necessity for the Digital Commerce Empowerment Ecosystem (DCEE).

**2.2.1 NATURAL SYSTEM STUDIED**

The natural system for small and local Indian businesses operates within a cultural and economic context that emphasizes community relationships and local networks. Traditionally reliant on in-person transactions, these businesses foster strong customer ties, often offering unique products based on local resources and traditional knowledge. However, limited access to digital tools and infrastructure hinders their competitiveness against larger enterprises. As digital transformation accelerates, the increasing adoption of smartphones and the internet presents opportunities for enhancing visibility and streamlining operations. DCEE aims to integrate these traditional practices with modern technology, promoting growth while preserving the spirit of local entrepreneurship.

**2.2.2 DESIGNED SYSTEM STUDIED**

The Digital Commerce Empowerment Ecosystem (DCEE) is a comprehensive platform designed to empower small and local Indian businesses by enhancing their digital presence and operational efficiency. It features three primary user roles: Digital Storefront Owners, who can manage their profiles and products; Customers, who can browse, purchase, and manage their orders; and Admins, who oversee user management and transactions. The system includes an Authentication Module for secure access, a Platform Module for storefront management, a Payment Module for smooth financial transactions, an Analytics Module for insights into user behavior, and a Chatbot Module for user assistance. Developed using Flask API for the backend and MongoDB for data storage, with a responsive frontend built on Bootstrap, DCEE aims to bridge the digital divide and foster entrepreneurship in local communities.

### DRAWBACKS OF EXISTING SYSTEM

* Limited Digital Presence: Many businesses struggle to establish a strong online presence, reducing their visibility to potential customers.
* Inefficient Inventory Management: Existing systems often lack effective tools for managing stock levels, leading to issues like overstocking or understocking.
* Complex Payment Processes: Payment methods can be cumbersome, discouraging customers from completing transactions.
* Regulatory Compliance Challenges: Navigating regulatory requirements can be complicated, making it difficult for businesses to remain compliant.
* Digital Literacy Gaps: Many business owners and employees lack the necessary digital skills, preventing them from effectively using existing technologies.

### PROPOSED SYSTEM

The proposed system for small and local Indian businesses aims to transform their digital presence and operational efficiency. It will provide a user-friendly platform that allows businesses to establish and manage online storefronts, significantly enhancing visibility and customer reach. Integrated predictive analytics tools will enable accurate forecasting of inventory needs, minimizing the risks associated with overstocking and understocking. Additionally, the system will streamline payment solutions, offering secure and simplified transaction methods for both customers and business owners. To ensure compliance with local regulations, built-in features will guide users through the necessary requirements. Furthermore, the system will include resources and training programs focused on improving digital literacy, empowering users to leverage the platform effectively and enhance their overall business operations.

### ADVANTAGES OF PROPOSED SYSTEM

* Enhanced Digital Presence
* Improved Inventory Management
* Simplified Payment Processes
* Streamlined Regulatory Compliance
* Increased Digital Literacy and Support
* Greater Customer Trust and Engagement
* Scalable and Flexible Solutions

# CHAPTER 3

# REQUIREMENT ANALYSIS

## FEASIBILITY STUDY

The feasibility study evaluates the viability of the Digital Commerce Empowerment Ecosystem (DCEE) project by examining its technical, economic, operational, legal, and market aspects. It assesses whether the chosen technology stack (Flask for backend, Bootstrap for frontend, and MongoDB for database) is suitable and compatible with project requirements. The economic analysis focuses on development costs, potential revenue, and return on investment. Operational feasibility investigates how the system will integrate into current business practices and address user needs. Legal considerations ensure compliance with data protection regulations and payment processing laws. Lastly, market feasibility assesses the demand for the system among small businesses, ensuring it aligns with market trends and customer expectations. This comprehensive analysis aims to determine the project's overall viability and sustainability.

### Economical Feasibility

The economic feasibility study for the Digital Commerce Empowerment Ecosystem (DCEE) assesses the project's financial viability, focusing on costs, revenue potential, and return on investment. Initial costs encompass development, hosting, and salaries, while ongoing expenses include maintenance and support. Revenue sources are expected from subscription fees, transaction fees, and advertising.

Key components include:

* Market Demand: Evaluating the need for digital solutions.
* Cost-Benefit Analysis: Comparing costs with anticipated benefits.
* Scalability: Potential for service expansion.
* Risk Assessment: Identifying and mitigating financial risks.
* Funding Sources: Exploring grants and investments.
* The study indicates that DCEE can deliver substantial financial benefits and support local business growth.

### Technical Feasibility

The technical feasibility study for the Digital Commerce Empowerment Ecosystem (DCEE) evaluates the project's technological requirements and capabilities to ensure successful implementation. It examines the software, hardware, and network infrastructure necessary to develop and operate the platform effectively.

Key components include:

* Technology Stack: Utilization of Bootstrap for frontend development, Flask API for backend services, and MongoDB for database management.
* System Integration: Ensuring seamless integration between various modules, including authentication, payment processing, analytics, and chatbot functionalities.
* Scalability: Assessing the system's ability to handle increased user traffic and data as the platform grows.
* Security Measures: Implementing robust security protocols to protect user data and transactions.
* User Support: Providing necessary training and documentation to users for effective system utilization.

The study concludes that the DCEE project is technically feasible, leveraging existing technologies and best practices to create a reliable and secure digital commerce platform.

### Behavioral Feasibility

The behavioral feasibility study for the Digital Commerce Empowerment Ecosystem (DCEE) examines user acceptance, training needs, and potential behavioral changes among digital storefront owners, customers, and administrators. Key aspects include:

* User Acceptance: Assessing the willingness of small businesses to adopt the platform.
* Training Needs: Identifying necessary support for effective platform navigation.
* User Engagement: Understanding expectations to enhance loyalty and usage.
* Behavioral Change: Evaluating how the platform can shift purchasing and management practices.
* Feedback Mechanisms: Establishing channels for continuous improvement based on user input.

This study indicates that with adequate support and engagement strategies, the DCEE project is poised for high user acceptance and positive behavior changes in digital commerce.

**3.1.4 Questionnaire**

1. What are the biggest challenges you face in running your business online (if applicable)?
   * They do not have a website yet, and managing social media for sales is overwhelming and get a lot of questions through email, and it's hard to keep track of everything.
2. Do you currently have an online store? If yes, what platform do you use?
   * No, they do not have any online platform
3. What are your biggest concerns about adopting a new platform like DCEE? (e.g., cost, learning curve, security)
   * Will DCEE be affordable for a small business like them and they are not very tech-savvy, so they are worried it will be difficult to learn a new platform.
4. What features are most important to you in an e-commerce platform? (e.g., ease of use, secure payment processing, customer relationship management tools)
   * A user-friendly platform is crucial for them. Safe and secure transactions are essential for building customer trust.
5. How comfortable are you with using technology?
   * They are somewhat comfortable with technology, but not an expert
6. Would you be interested in educational resources to help you navigate the DCEE platform and improve your online presence?
   * Educational resources would be a huge help for the business and any resources that can help improve online presence
7. What is your typical budget for marketing and online tools?
   * They have not allocated a specific budget
8. How many employees do you have in your business?
   * There are no employees only the owner and his wife
9. What are your long-term goals for growing your business online?

* The goal is to establish a strong online presence and increase my customer base

1. Would you be interested in a free trial of the DCEE platform to see if it meets your needs?

* Yes, they are interested to free trial in DCEE

## SYSTEM SPECIFICATION

### Hardware Specification

Processor - Intel Core i3

RAM - 8 GB

Hard disk - 256 GB SSD or higher

### Software Specification

Front End - BOOTSTRAP

Backend - FLASK, Mongo

Client on PC - Windows 7 and above.

Technologies used - JS, HTML5, AJAX, CSS

## SOFTWARE DESCRIPTION

### Eg. FLASK

Flask is a lightweight and flexible web framework written in Python, designed for creating web applications quickly and with minimal overhead. It follows a modular approach, offering only essential tools and allowing developers to add extensions as needed for database connections, form handling, authentication, and more. Flask operates on the WSGI (Web Server Gateway Interface) standard and uses Jinja2 for templating, providing robust tools for rendering dynamic HTML with minimal effort.

Being “micro,” Flask is unopinionated, meaning it does not enforce any project structure or dependency requirements, giving developers control over the app’s architecture and behavior. Its simplicity and adaptability make it ideal for small to medium-sized applications, while its scalability supports more complex setups when combined with additional components. Flask is a powerful yet minimalist web framework written in Python, ideal for developers who prefer simplicity and control in web application development. As a "micro-framework," Flask includes only the essential components for handling requests, routing, and templating, making it exceptionally lightweight and flexible. It leverages WSGI (Web Server Gateway Interface) for handling web requests and Jinja2, a fast and expressive templating engine, for creating dynamic HTML responses.

One of Flask’s standout features is its extensibility—developers can choose specific libraries or extensions for functionalities like database integration, form validation, and authentication without carrying any unnecessary dependencies. This "plug-and-play" approach allows developers to build both simple applications and scalable, production-level systems. Flask is also highly compatible with other Python libraries, making it popular for projects that include machine learning, data visualization, or complex business logic. Its straightforward syntax and clear documentation make Flask a top choice for beginners, while its flexibility and powerful extensions appeal to experienced developers who need a tailored, high-performing application stack.

### Eg. MongoDB

### MongoDB is a NoSQL database known for its scalability, flexibility, and document-based structure. Unlike traditional relational databases that store data in tables with rows and columns, MongoDB stores data in BSON (Binary JSON) documents, allowing it to handle unstructured or semi-structured data efficiently. Each document is a self-contained data unit, making MongoDB well-suited for applications that require rapid data integration, real-time analytics, and frequent updates to data structure without disrupting the entire system.

### MongoDB’s schema-less nature allows developers to add or modify fields easily without predefined data schemas. This flexibility makes it ideal for projects that evolve quickly or handle large volumes of diverse data types, like social media applications, e-commerce platforms, and IoT solutions. It also supports a rich query language, indexing, aggregation, and powerful features like horizontal scaling (sharding) and replication for high availability, which enhances its performance and fault tolerance. MongoDB integrates well with modern tech stacks, making it a popular choice for full-stack developers and organizations adopting cloud-native, distributed application architectures.

# CHAPTER 4

# SYSTEM DESIGN

* 1. **INTRODUCTION**

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

## UML DIAGRAM

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

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

## USE CASE DIAGRAM

## A Use Case Diagram is a visual representation of a system's functionality, highlighting the interactions between users (actors) and the system itself. It serves as a high-level overview, showing how users achieve their goals by utilizing the system's various functions, represented as "use cases." These diagrams are commonly used in the early stages of software design to communicate system requirements, focusing on what the system does rather than how it does it.

## In a Use Case Diagram:

## Actors represent the users or external systems that interact with the system, such as admins, customers, or third-party services.

## Use Cases represent specific functionalities or actions the system performs, such as "Login," "Register," "Purchase Product," or "View Orders."

## Associations are the lines connecting actors and use cases, indicating interactions or relationships.

## System Boundary is a rectangle that encapsulates all the use cases, defining the scope of the system.

## Use Case Diagrams are useful for understanding user requirements, identifying main functionalities, and defining roles within a system. They also serve as a basis for further detailed analysis and design, helping ensure the system aligns with user needs.

## 

## SEQUENCE DIAGRAM

## A sequence diagram is a type of interaction diagram in Unified Modeling Language (UML) that shows how objects interact in a particular sequence within a system. It visually represents the flow of messages or events between objects over time, illustrating how and in what order operations are carried out.

## Key components of a sequence diagram include:

## Actors: Represent users or external systems interacting with the system.

## Objects or Classes: Represent entities within the system that perform actions.

## Lifelines: Vertical lines showing the life span of an object during the sequence.

## Activation Bars: Rectangles on the lifelines indicating the period when an object is active.

## Messages: Arrows between objects indicating communication; they can be synchronous or asynchronous.

## Return Messages: Dashed lines that show the return or response after a message is processed.

## Use and Advantages:

## Sequence diagrams are integral to modelling the dynamic aspects of systems and are especially useful during the design phase for:

## Clear Communication: They provide a visual, step-by-step breakdown of processes, improving communication among team members.

## Requirement Validation: Diagrams help validate requirements by demonstrating how processes should flow and helping stakeholders confirm intended functionality.

## System Design and Debugging: Developers use sequence diagrams to refine system architecture, identify bottlenecks, and troubleshoot issues by understanding exact message flow.

## Sequence diagrams support collaboration by showing the workflow in a way that is easy to understand and highly detailed. They help developers, stakeholders, and testers confirm requirements and understand how different components should interact, assisting in debugging, performance optimization, and overall design clarity.

## 

## 4.2.2 State Chart Diagram

A state chart diagram, or state diagram, illustrates the various states an object or system component can undergo throughout its lifecycle, along with the transitions between those states triggered by events. Each state is represented as a rounded rectangle, while transitions are shown as arrows connecting the states, labeled with the events that cause the transitions.

**Key Elements:**

**States**: Conditions an object can be in (e.g., Idle, Processing).

**Transitions**: Arrows indicating changes between states, triggered by events.

**Events and Actions**: Triggers for state changes and activities that occur during transitions.

**Entry and Exit Actions**: Actions taken when entering or leaving a state.

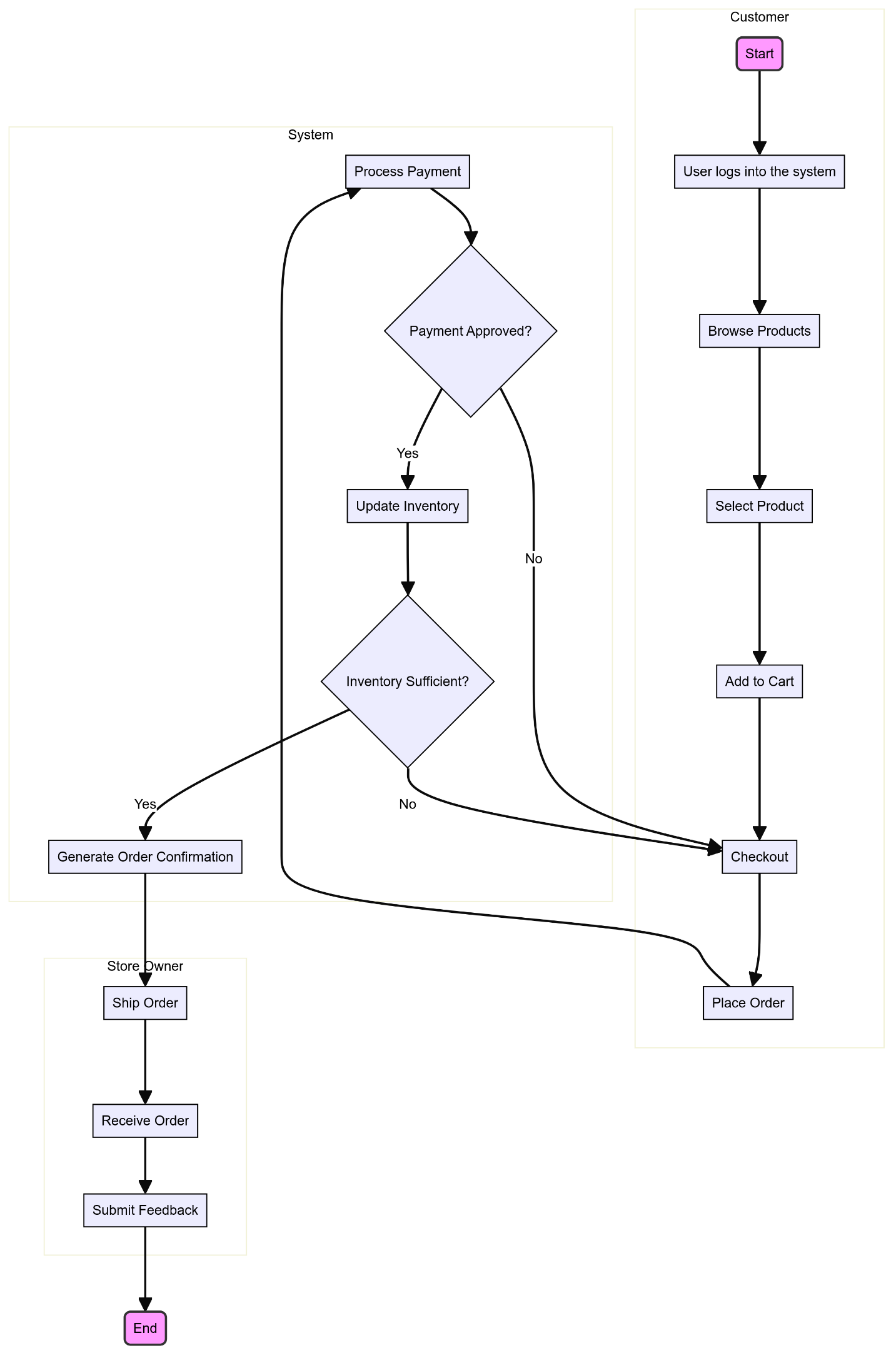
**Composite States**: Hierarchical states that contain sub-states for complex behaviors.

**Use and Benefits:**

State chart diagrams model dynamic behavior, simplify complex systems, and validate that all expected states and transitions are covered. They are commonly used in applications with state-dependent behavior, such as user interfaces and workflow management systems, providing a clear view of how an object responds to various events.

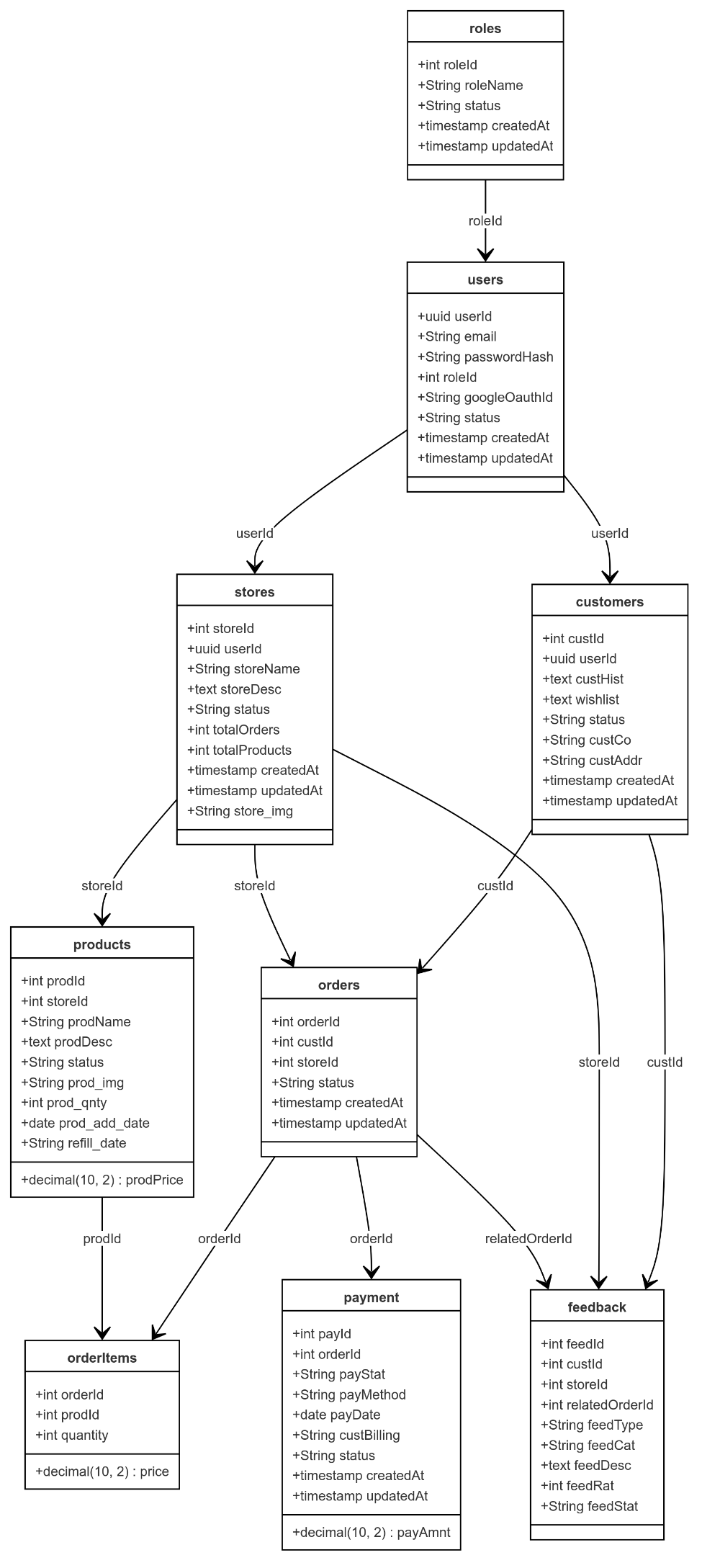
## Activity Diagram

An activity diagram is a behavioral diagram in Unified Modeling Language (UML) that illustrates the flow of activities or actions within a system, highlighting the dynamic aspects of processes. It features key elements such as activities, represented by rounded rectangles, which indicate tasks performed within the system. The diagram begins with a start node, depicted as a filled circle, and concludes with an end node, a circle encased within another circle, marking the process's completion. Transitions, represented by arrows, connect activities to demonstrate the flow from one task to another. Decision nodes, illustrated as diamonds, indicate points where the flow can diverge based on specific conditions, while forks and joins (depicted as bars) represent parallel activities. Activity diagrams are particularly valuable for modeling complex workflows, use case scenarios, and business processes, as they clarify the sequence of operations, identify potential bottlenecks, and facilitate communication among stakeholders by visualizing activities and their interdependencies.



## Class Diagram

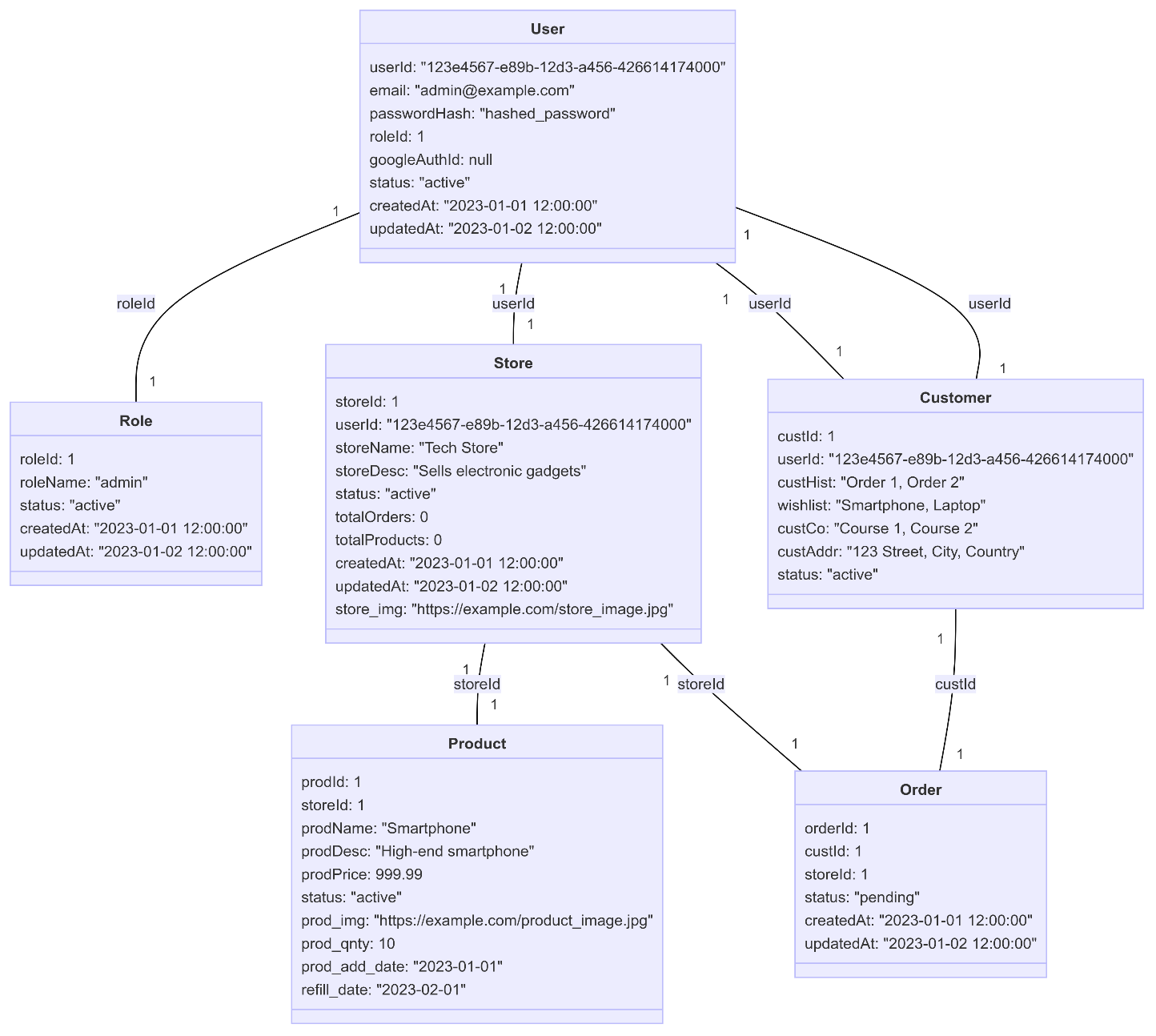
A class diagram is a static structure diagram in Unified Modeling Language (UML) that represents the system's classes, their attributes, methods, and the relationships among them. It provides a visual blueprint of the system's architecture, highlighting how different classes interact and how data is organized. Each class is depicted as a rectangle divided into three sections: the top section contains the class name, the middle section lists its attributes, and the bottom section shows its methods (or operations). Relationships between classes are illustrated through lines, with various notations to indicate the type of relationship, such as inheritance (generalization), associations, aggregations, and compositions. Class diagrams are essential for object-oriented design, as they help in understanding the system's structure, guiding the implementation of classes, and ensuring that all components interact correctly. They also serve as a communication tool among developers, designers, and stakeholders by providing a clear representation of the system's data model and its relationships.



## Object Diagram

An object diagram is a type of static structure diagram in Unified Modeling Language (UML) that provides a snapshot of the instances of classes (objects) and their relationships at a specific moment in time. Unlike class diagrams, which focus on the blueprint of classes and their relationships, object diagrams emphasize the concrete examples of those classes in a particular state.

Each object is represented by a rectangle, similar to a class in a class diagram, but it includes the object's name and its current state or attribute values. Lines connecting the objects illustrate their relationships, such as associations, aggregations, or compositions. Object diagrams are particularly useful for visualizing complex relationships and interactions among objects in a system, demonstrating how they collaborate to perform tasks or represent data.



## Component Diagram

A component diagram is a type of structural diagram in Unified Modeling Language (UML) that represents the physical components in a system, such as software applications, libraries, and packages. It illustrates how these components interact with one another and their dependencies, providing a high-level view of the system's architecture.

Components are depicted as rectangles with two smaller rectangles on the left side, indicating that they can be independently developed and deployed. They may also include interfaces that define the operations available for other components to use. Component diagrams are particularly useful for modeling complex systems, as they help to visualize the relationships between various software components, their functionalities, and how they fit into the overall system architecture. This aids in understanding the system’s modular structure, facilitates better planning for system integration, and enhances the communication between developers.

**4.2.8 Deployment Diagram**

A deployment diagram is another type of UML diagram that shows the physical deployment of artifacts on nodes. It illustrates how software is distributed across hardware and how different components communicate within the system infrastructure. Deployment diagrams are particularly useful in visualizing the physical arrangement of software components in a distributed system, such as client-server architectures or cloud-based applications.

In a deployment diagram, nodes represent physical devices (like servers, computers, or mobile devices), and artifacts (such as executables, libraries, or database schemas) represent the software deployed on those nodes. Connections between nodes depict communication paths, showing how information flows within the system. This diagram is essential for understanding the system’s architecture and can help in assessing performance, scalability, and reliability.

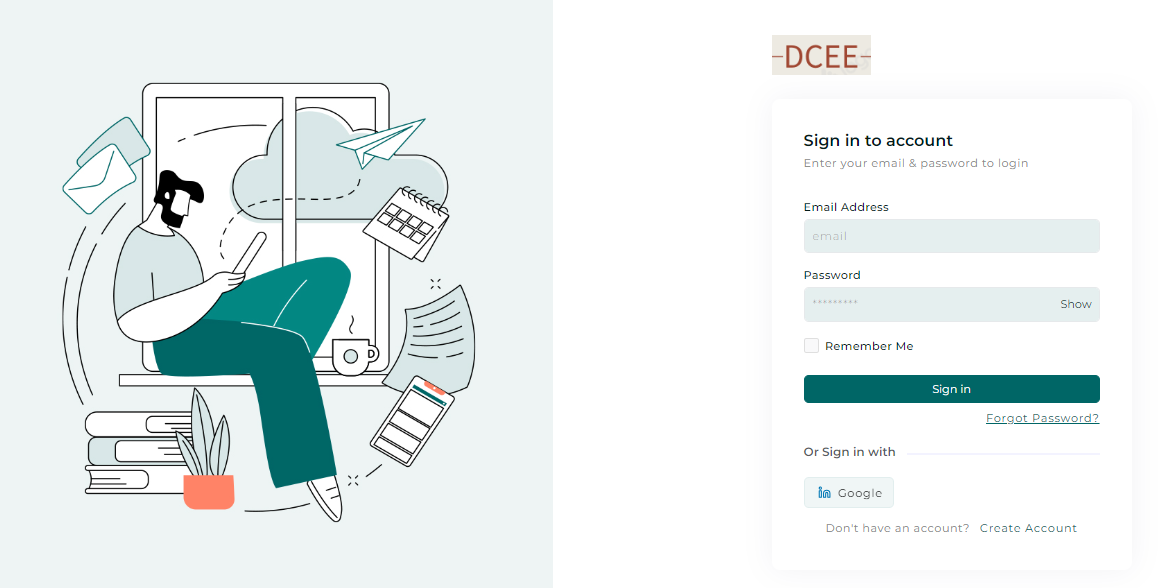
**4.2.9 Collaboration Diagram**

A collaboration diagram, or communication diagram, is a type of UML diagram that illustrates the interactions between objects in a system. It emphasizes the relationships and associations among objects while depicting the messages exchanged between them. Each object is represented as a rectangle, and lines connect these objects to indicate their relationships.

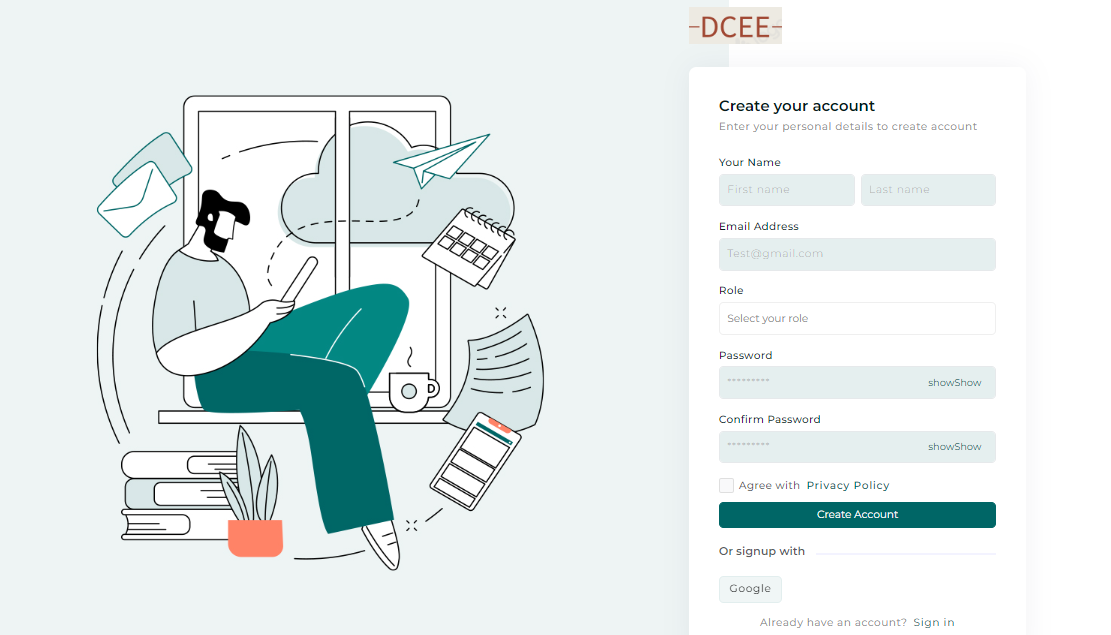
Messages are numbered to show the sequence of interactions, providing a clear view of how different components work together to achieve a specific task. Collaboration diagrams help visualize dynamic behavior, clarify roles, and enhance communication among team members during development, making them a valuable tool in system design.

## 4.3 USER INTERFACE DESIGN USING FIGMA

**Form Name: Login**



**Form Name: Register**

****

**Form Name: Home Page**

## 4.4 DATABASE DESIGN

### 4.4.1 Relational Database Management System (RDBMS)

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

**Relationships**

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

### 4.4.2 Normalization

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

* Normalize the data
* Choose proper names for the tables and columns.
* Choose the proper name for the data.

**First Normal Form**

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

**Second Normal Form**

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

**Third Normal Form**

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

**Fourth Normal Form**

The fourth normal form (4NF) is a database normalization rule that further refines data modeling by addressing multi-valued dependencies. When a table contains multiple independent sets of repeating data, we can break it down into smaller tables, with each table containing only one set of related data.

This reduces data redundancy and improves data consistency by ensuring that each table represents a single, well-defined concept or entity. To achieve 4NF, we need to ensure that all multi-valued dependencies are removed from the table, and that each table contains only attributes that are functionally dependent on the primary key.

**Fifth Normal Form**

5NF is indeed the highest level of normalization in relational database design, and it deals with complex data models that involve multiple overlapping multi-valued dependencies. In 5NF, tables are decomposed into smaller tables in order to eliminate any possible redundancy caused by overlapping dependencies, while ensuring that there is no loss of data.

The goal of 5NF is to ensure that each table represents a single entity or relationship, and that the data is organized in a way that minimizes redundancy, eliminates anomalies, and improves data integrity. By breaking down tables into smaller, more specialized tables, 5NF helps to eliminate potential issues with update anomalies, insertion anomalies, and deletion anomalies, which can occur when data is not properly normalized.

### 4.4.3 Sanitization

### An automated procedure called "sanitization" is used to get a value ready for use in a SQL query. This process typically involves checking the value for characters that have a special significance for the target database. To prevent a SQL injection attack, you must sanitize(filter) the input string while processing a SQL query based on user input. For instance, the user and password input is a typical scenario. In that scenario, the server response would provide access to the 'target user' account without requiring a password check.

**4.4.4 Indexing**

By reducing the number of disk accesses needed when a query is completed, indexing helps a database perform better. It is a data structure method used to locate and access data in a database rapidly. Several database columns are used to generate indexes. The primary key or candidate key of the table is duplicated in the first column, which is the Search key. To make it easier to find the related data, these values are kept in sorted order. Recall that the information may or may not be kept in sorted order.

### 4.5 TABLE DESIGN

**1.Tbl\_roles**

Eg.Primary key: roleId

Eg.Foreign key: roleId references table Tbl\_user

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No: | Fieldname | Datatype (Size) | Key Constraints | Description of the Field |
| 1. | roleId | int | PRIMARY KEY | Unique identifier for the role |
| 2. | roleName | varchar | NOT NULL | Name of the role (e.g., admin, customer) |
| 3. | status | varchar | NOT NULL | Status of the role (e.g., active, inactive) |

**2.Tbl\_ users**

Eg.Primary key: userId

Eg.Foreign key: userId references table Tbl\_roles

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No: | Fieldname | Datatype (Size) | Key Constraints | Description of the Field |
| 1. | userId | int | PRIMARY KEY | Unique identifier for the user |
| 2. | email | varchar | NOT NULL | User's email address |
| 3. | passwordHash | varchar | NOT NULL | Hashed password for the user |
| 4. | roleId | int | FOREIGN KEY | User's assigned role ID (references roles) |
| 5. | googleAuthId | int | NOT NULL | Google authentication ID (if applicable) |
| 6. | status | varchar | NOT NULL | Status of the role (e.g., active, inactive) |

**3.Tbl\_ stores**

Eg.Primary key: **stores**

Eg.Foreign key: **stores** references table Tbl\_products

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No: | Fieldname | Datatype (Size) | Key Constraints | Description of the Field |
| 1. | storeId | int | PRIMARY KEY | Unique identifier for the store |
| 2. | userId | int | FOREIGN KEY | User ID who owns the store (references users) |
| 3. | storeName | varchar | NOT NULL | Name of the store |
| 4. | storeDesc | varchar | NOT NULL | Description of the store |
| 5. | status | varchar | NOT NULL | Status of the store (e.g., active, inactive) |
| 6. | totalOrders | Int | NOT NULL | Total number of orders placed for the store |
| 7. | totalProducts | int | NOT NULL | Total number of products in the store |
| 8. | store\_img | varchar | NOT NULL | URL or path to the store image |

**4.Tbl\_ products**

Eg.Primary key: **products**

Eg.Foreign key: **products** references table Tbl\_ **orderItems**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No: | Fieldname | Datatype (Size) | Key Constraints | Description of the Field |
| 1. | prodId | int | PRIMARY KEY | Unique identifier for the product |
| 2. | storeId | varchar | FOREIGN KEY | Store ID where the product belongs (references stores) |
| 3. | prodName | varchar | NOT NULL | Name of the product |
| 4. | prodDesc | varchar | NOT NULL | Description of the product |
| 5. | prodPrice | int | NOT NULL | Price of the product |
| 6. | status | varchar | NOT NULL | Status of the product (e.g., active, inactive) |
| 7. | prod\_img | varchar | NOT NULL | URL or path to the product image |
| 8. | prod\_qnty | int | NOT NULL | Quantity of the product in stock |
| 9. | prod\_add\_date | Date | NOT NULL | Date the product was added to the store |
| 10. | refill\_date | date | NOT NULL | Date the product needs to be restocked (optional) |

**5.Tbl\_ orderItems**

Eg.Primary key: orderId

Eg.Foreign key: orderId references table Tbl\_ **orders**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No: | Fieldname | Datatype (Size) | Key Constraints | Description of the Field |
| 1. | orderId | int | FOREIGN KEY | Unique identifier for the order |
| 2. | prodId | int | FOREIGN KEY | Product ID of the ordered item (references products) |
| 3. | quantity | int | NOT NULL | Quantity of the product ordered |
| 4. | price | int | NOT NULL | Price of the ordered product |

**6.Tbl\_ orders**

Eg.Primary key: orderId

Eg.Foreign key: orderId references table Tbl\_products

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No: | Fieldname | Datatype (Size) | Key Constraints | Description of the Field |
| 1. | orderId | int | PRIMARY KEY | Unique identifier for the order |
| 2. | custId | varchar | FOREIGN KEY | Customer ID who placed the order (references customers) |
| 3. | storeId | varchar | NOT NULL | Store ID from where the order was placed (references stores) |
| 4. | status | varchar | NOT NULL | Status of the order (e.g., pending, processing, shipped) |

**7.Tbl\_ customers**

Eg.Primary key: custId

Eg.Foreign key: custId references table Tbl\_products

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No: | Fieldname | Datatype (Size) | Key Constraints | Description of the Field |
| 1. | custId | int | PRIMARY KEY | Unique identifier for the customer |
| 2. | userId | varchar | FOREIGN KEY | User ID associated with the customer (references users) |
| 3. | custHist | varchar | NOT NULL | Customer's order history |
| 4. | wishlist | varchar | NOT NULL | Customer's wishlist of products |
| 5. | custCo | varchar | NOT NULL | Courses enrolled in by the customer |
| 6. | custAddr | varchar | NOT NULL | Customer's shipping address |
| 7. | status | varchar | NOT NULL | Status of the customer |

**8.Tbl\_ payment**

Eg.Primary key: payId

Eg.Foreign key: payId references table Tbl\_order

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No: | Fieldname | Datatype (Size) | Key Constraints | Description of the Field |
| 1. | payId | int | PRIMARY KEY | Unique identifier for the payment |
| 2. | orderId | int | FOREIGN KEY | Order ID associated with the payment (references orders) |
| 3. | payAmnt | int | NOT NULL | Amount paid for the order |
| 4. | payStat | varchar | NOT NULL | Payment status (e.g., pending, completed, failed) |
| 5. | payMethod | varchar | NOT NULL | Payment method used (e.g., credit card, debit card) |
| 6. | payDate | date | NOT NULL | Date the payment was made |
| 7. | custBilling | varchar | NOT NULL | Customer's billing address |
| 8. | status | varchar | NOT NULL | Status of the payment (e.g., active, refunded) |

# CHAPTER 5

# SYSTEM TESTING

* 1. **INTRODUCTION**

Software Testing is the process of executing software in a controlled manner, in order to answer the question - Does the software behave as specified? Software testing is often used in association with the term’s verification and validation. Validation is the checking or testing of items, includes software, for conformance and consistency with an associated specification. Software testing is just one kind of verification, which also uses techniques such as reviews, analysis, inspections, and walkthroughs. Validation is the process of checking that what has been specified is what the user wanted.

Other activities which are often associated with software testing are static analysis and dynamic analysis. Static analysis investigates the source code of software, looking for problems and gathering metrics without executing the code. Dynamic analysis looks at the behavior of software while it is executing, to provide information such as execution traces, timing profiles, and test coverage information.

Testing is a set of activity that can be planned in advanced and conducted systematically. Testing begins at the module level and work towards the integration of entire computers-based system. Nothing is complete without testing, as it vital success of the system testing objectives, there are several rules that can serve as testing objectives. They are:

Testing is a process of executing a program with the intent of finding an error.

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

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

* For correctness
* For implementation efficiency
* For computational complexity

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

## TEST PLAN

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

The levels of testing include:

* Unit testing
* Integration Testing
* Data validation Testing
* Output Testing

### Unit Testing

Unit testing focuses verification effort on the smallest unit of software design – the software component or module. Using the component level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and uncovered scope established for unit testing. The unit testing is Whitebox oriented, and step can be conducted in parallel for multiple components. The modular interface is tested to ensure that information properly flows into and out of the program unit under test. The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm’s execution. Boundary conditions are tested to ensure that all statements in a module have been executed at least once. Finally, all error handling paths are tested.

Tests of data flow across a module interface are required before any other test is initiated. If data do not enter and exit properly, all other tests are moot. Selective testing of execution paths is an essential task during the unit test. Good design dictates that error conditions be anticipated and error handling paths set up to reroute or cleanly terminate processing when an error does occur. Boundary testing is the last task of unit testing step. Software often fails at its boundaries.

Unit testing was done in Sell-Soft System by treating each module as separate entity and testing each one of them with a wide spectrum of test inputs. Some flaws in the internal logic of the modules were found and were rectified. After coding each module is tested and run individually. All unnecessary code were removed and ensured that all modules are working, and gives the expected result.

### Integration Testing

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

### Validation Testing or System Testing

This is the final step in testing. In this the entire system was tested with all forms, code, modules, and class modules. This form of testing is popularly known as Black Box testing or System tests.

Black Box testing method focuses on the functional requirements of the software. That is, Black Box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program.

Black Box testing attempts to find errors in the following categories; incorrect or missing functions, interface errors, errors in data structures or external data access, performance errors and initialization errors and termination errors.

### Output Testing or User Acceptance Testing

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

* Input Screen Designs.
* Output Screen Designs

The above testing is done taking various kinds of test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested using that test data. While testing the system by which test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

* + 1. **Automation Testing**

A test case suite is executed using specialized automated testing software tools as part of the software testing technique known as automation testing. The test stages are meticulously carried out by a human performing manual testing while seated in front of a computer. Additionally, the automation testing software may generate thorough test reports, compare expected and actual findings, and enter test data into the System Under Test. Software test automation necessitates significant financial and material inputs. Repeated execution of the same test suite will be necessary during subsequent development cycles. This test suite can be recorded and replayed as needed using a test automation tool. No further human involvement is needed once the test suite has been automated.

* + 1. **Selenium Testing**

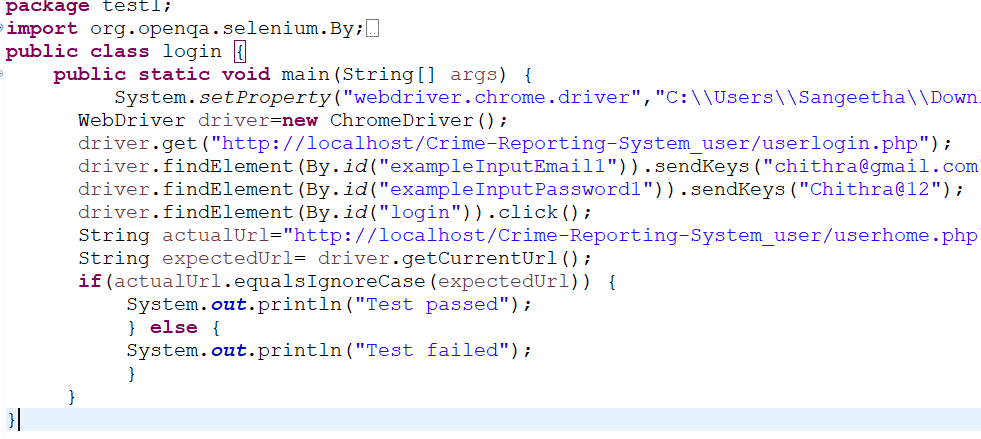
Selenium is a free and open-source tool for testing web applications across multiple browsers and operating systems. Selenium Test Scripts can be written in different programming languages, including Java, C#, JavaScript, Python, etc. Automation performed using the Selenium framework is referred to as Selenium Automation testing.

.

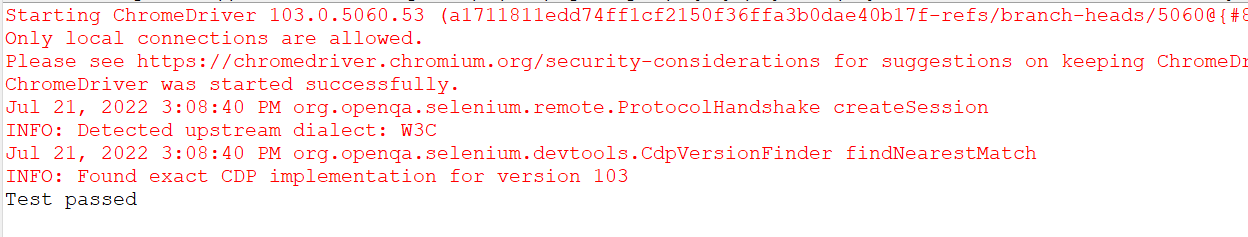
**Example:**

**Test Case 1**

**Code**



**Eg.Screenshot**



**Eg.Test Report**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case 1** | | | | | |
| **Project Name: Digital Commerce Empowerment Ecosystem (DCEE)** | | | | | |
| **Login Test Case** | | | | | |
| **Test Case ID: Test\_1** | | | **Test Designed By: Josna Mary Thomas** | | |
| **Test Priority (Low/Medium/High):** | | | **Test Designed Date: 20/10/2024** | | |
| **Module Name**: Registration Page | | | **Test Executed By: Sona Maria Sebastian** | | |
| **Test Title: Registration of user account** | | | **Test Execution Date: 22/10/2024** | | |
| **Description: Registration user account** | | |  | | |
| **Pre-Condition: User** have unique email id | | | | | |
| **Step** | **Test Step** | **Test Data** | **Expected Result** | **Actual Result** | **Status (Pass/**  **Fai l)** |
| 1 | Navigate to the Registration page | http://127.0.0.1:5000/auth/register | Registration page should be displayed | Registration page was loaded | Pass |
| 2 | Enter first name | “Josna” | The first name field should be filled with "Josna" | First name entered | Pass |
| 3 | Enter last name | “Thomas” | The last name field should be filled with "Thomas" | Last name entered | Pass |
| 4 | Enter role | “Customer” | The role field is selected as customer | Role selected | Pass |
| 5 | Enter email | “josnamarythomas25@gmail.com” | The email field should be filled with “josnamarythomas25@gmail.com” | Email entered | Pass |
| 6 | Enter password | “Josna@123” | The password field should be filled with “Josna@123” | Password inserted | pass |
| 7 | Enter confirm password | “Josna@123” | The confirm password field should be filled with “Josna@123” | Confirm password inserted | Pass |
| 8 | Click on the register button | N/A | If registration is successful, user should be redirected to login page | User was redirected to login page | Pass |
|  |  |  |  |  |
| **Post-Condition: First name, last name, email, password, confirm password can be registered** | | | | | |

**Test Case 2:**

**Code**

**Screenshot**

**Test report**

**Minimum 4 test cases (1 login 3 functionalities)**

# CHAPTER 6

# IMPLEMENTATION

## INTRODUCTION

Implementation is the stage of the project where the theoretical design is turned into a working system. It can be the most crucial stage in achieving a successful new system gaining the users confidence that the new system will work and will be effective and accurate. It is primarily concerned with user training and documentation. Conversion usually takes place about the same time the user is being trained or later. Implementation simply means convening a new system design into operation, which is the process of converting a new revised system design into an operational one.

At this stage the main work load, the greatest upheaval and the major impact on the existing system shifts to the user department. If the implementation is not carefully planned or controlled, it can create chaos and confusion.

Implementation includes all those activities that take place to convert from the existing system to the new system. The new system may be a totally new, replacing an existing manual or automated system or it may be a modification to an existing system. Proper implementation is essential to provide a reliable system to meet organization requirements. The process of putting the developed system in actual use is called system implementation. This includes all those activities that take place to convert from the old system to the new system. The system can be implemented only after through testing is done and if it is found to be working according to the specifications. The system personnel check the feasibility of the system. The more complex the system being implemented, the more involved will be the system analysis and design effort required to implement the three main aspects: education and training, system testing and changeover. The implementation state involves the following tasks:

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

## IMPLEMENTATION PROCEDURES

Implementation of software refers to the final installation of the package in its real environment, to the satisfaction of the intended uses and the operation of the system. In many organizations someone who will not be operating it, will commission the software development project. In the initial stage people doubt about the software but we must ensure that the resistance does not build up, as one must make sure that:

* The active user must be aware of the benefits of using the new system. Their confidence in the software is built up.
* Proper guidance is imparted to the user so that he is comfortable in using the application.

Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running in the server. If the server object is not up running on the server, the actual process will not take place

### User Training

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

### Training on the Application Software

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

### System Maintenance

System maintenance is an effective component such that System maintenance refers to the ongoing activities required to ensure that a system or application operates effectively and efficiently after it has been implemented. It involves regular updates, bug fixes, and performance optimizations to keep the system running smoothly and securely.

System maintenance is essential to ensure that a system remains operational and effective after implementation. By establishing maintenance procedures and following them consistently, project teams can ensure that the system operates smoothly, remains secure, and continues to meet the needs of the end-users.

# CHAPTER 7

# CONCLUSION AND FUTURE SCOPE

## CONCLUSION

## 

The DCEE project is a comprehensive digital empowerment platform designed to bridge critical gaps faced by small and local Indian businesses. Through its intuitive interface, DCEE streamlines essential tasks, from inventory management and secure payment transactions to customer engagement and compliance support. The project integrates a predictive analytics tool powered by the RoBERTa model, providing insights that help businesses forecast demand, optimize stock, and navigate market trends. DCEE's commitment to digital inclusivity and user-friendly tools equips entrepreneurs with the resources to strengthen customer trust, improve operational efficiency, and foster growth. By encouraging digital literacy and facilitating rural payment solutions, DCEE is an all-encompassing tool aiming to sustain local commerce, empower business owners, and nurture an interconnected economic ecosystem.

* 1. **FUTURE SCOPE**

The future scope of the DCEE project is promising, with opportunities to integrate advanced machine learning models for even more precise demand forecasting and customer behavior insights. Additional features could include multilingual support, catering to a diverse user base across India, and more sophisticated digital literacy modules to help users become proficient in the platform’s tools. As digital payments continue to grow, DCEE could expand its financial offerings with options like microfinancing and lending solutions tailored for small businesses. Furthermore, the platform could incorporate blockchain technology for enhanced transaction transparency and security, building trust with customers and streamlining compliance processes. DCEE's future evolution will continue to empower small businesses to adapt, compete, and grow sustainably in an increasingly digital marketplace.

.

# CHAPTER 8

# BIBLIOGRAPHY

### REFERENCES:

* + - ..
    - ..
    - ..
    - ..
    - ...

### WEBSITES:

* + - [..](http://www.w3schools.com/)
    - [..](http://www.jquery.com/)
    - [..](http://homepages.dcc.ufmg.br/%7Erodolfo/es-1-03/IEEE-Std-830-1998.pdf)
    - [..](http://www.agilemodeling.com/artifacts/useCaseDiagram.html)

# CHAPTER 9

# APPENDIX

## Sample Code

Main functionalities

## Screen Shots