

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

Thangam Jewellery strives to redefine the online jewellery shopping experience by providing an extensive array of meticulously crafted pieces, ranging from timeless classics to contemporary designs. The platform boasts an intuitive and visually appealing interface, ensuring customers can effortlessly navigate through diverse jewellery categories. Detailed product descriptions accompanied by high-quality images empower customers to make informed purchase decisions. Committed to customer satisfaction, Thangam Jewellery features a robust user account system, allowing individuals to personalize their preferences, track orders, and receive tailored recommendations. The implementation of secure payment gateways ensures that transactions are protected, instilling confidence in users to make purchases without concerns about data security.

1.2 PROJECT SPECIFICATION

There are three modules.

User:

- Display a variety of jewellery items including rings, chains etc.
- Provide high-quality images of each product.
- Include detailed descriptions, materials used, and pricing information for each item.
- Implement search functionality to allow users to quickly find specific products.
- Provide filtering options based on categories, price ranges, materials, etc
- Offer detailed product pages with zoomable images, product specifications, and sizing information.
- Allow users to create a Wishlist of their favourite jewellery items.
- Provide live video of the product in below the product and options try at home
- Provide user registration and login functionality.
- Allow users to track their order history, manage addresses, and update their personal information.
- Implement a secure and user-friendly checkout process
- Offer multiple payment options, such as google pay and other online payment methods.
- Implement a virtual try-on feature that allows users to see how jewellery looks on them using augmented reality technology
- Provide contact information for customer support
- Allow users to purchase and send gift cards to others.

Admin:

- Add, edit, and delete jewellery products with details such as images, descriptions, prices, and availability.
- Create and manage product categories and subcategories to organize jewellery items.
- View and manage customer orders, including order processing, and fulfilment.
- Mark orders as shipped or completed, and update order statuses.
- Manage customer accounts, including registration, login, and account details
- Assist customers with account-related issues and inquiries
- Manage static content on the website, including text, images, banners, and announcements
- Admin can check the purity of the jewellery and provide the status based on BIS.
- Process returns and refunds, and manage communication with customers regarding these processes.
- Regularly update the website platform, plugins, and security patches to ensure optimal performance and security.

Vendor:

- Provide tools for sellers to view and manage incoming orders, mark orders as shipped, and update order statuses
- Allow sellers to define their own return, refund, and exchange policies.

Languages Using

HTML, CSS, Django, Sqlite3, JavaScript

Features:

- **User-Friendly Interface:** The platform features an intuitive and visually appealing interface for easy navigation and exploration of jewellery items.
- **High-Quality Product Presentation:** High-resolution images and detailed descriptions showcase each jewellery product, highlighting materials used and pricing information.
- **Search and Filtering Options:** Users can efficiently find specific products through a robust search functionality and filtering options based on categories, price ranges, and materials.
- **Wishlist Functionality:** Users can create and manage a Wishlist of their fav jewellery items.
- **User Account System:** Seamless user registration and login functionality, allowing users to track order history, manage addresses, and update personal information.
- **Secure Checkout Process:** Implementation of a secure and user-friendly checkout process with multiple payment options, including Google Pay and other online methods.

- Virtual Try-On Feature: Integration of augmented reality technology for a virtual try-on experience, allowing users to visualize how jewellery looks on them.
- Customer Support: Contact information for customer support is provided, and users can purchase and send gift cards to others.
- Admin Control Panel: Admins can easily add, edit, and delete jewellery products, manage categories, and view and process customer orders.
- Purity Check: Admins can check the purity of jewellery items based on BIS standards and provide status information.
- Returns and Refunds: Admins can process returns and refunds, managing communication with customers throughout these processes.
- Website Maintenance: Regular updates to the website platform, plugins, and security patches ensure optimal performance and security.
- Vendor Tools: Vendors have tools to view and manage incoming orders, mark orders as shipped, and define their own return, refund, and exchange policies.

CHAPTER 2

SYSTEM STUDY

2. INTRODUCTION:

Thangam Jewellery envisions a paradigm shift in online jewellery shopping, introducing a sophisticated e-commerce platform that seamlessly blends aesthetics, user engagement, and operational efficiency. The proposed system aims to address the limitations of the existing model by incorporating advanced features to elevate the overall shopping experience.

2.1 EXISTING SYSTEM:

The Thangam jewellery shopping system lacks a cohesive and immersive user experience. Navigation is often cumbersome, and the absence of features such as a virtual try-on limits customers' ability to assess products accurately. Incomplete user accounts and insufficient vendor tools contribute to a less-than-optimal shopping journey for both buyers and sellers.

2.2 NATURAL SYSTEM STUDIED:

The study of natural systems emphasized the need for a dynamic and adaptable platform. Thangam Jewellery incorporates these principles by creating a user-friendly interface that seamlessly adapts to the evolving needs of customers. The design mimics the fluidity and interconnectedness found in nature, ensuring a harmonious and engaging user experience.

2.3 DESIGNED SYSTEM STUDIED:

Insights from studied designed systems underscored the importance of robust administrative controls and vendor tools. Thangam Jewellery integrates these features to streamline operations, providing administrators with efficient tools for product management and order processing. This aligns with the best practices observed in successful e-commerce platform...

2.4 DRAWBACKS OF EXISTING SYSTEM:

The drawbacks of the existing system include a lack of a virtual try-on feature, hindering customers from fully experiencing the products. Incomplete user accounts limit personalization options, impacting the sense of individualized service. Additionally, insufficient vendor tools restrict sellers' capabilities, hampering their efficiency in managing orders and defining policies.

2.5 PROPOSED SYSTEM:

The proposed system addresses the limitations of the existing system by introducing a user-friendly interface, detailed product information, virtual try-on features, and secure payment options. The platform aims to redefine the way users explore and purchase jewellery online.

ADVANTAGES OF PROPOSED SYSTEM:

Multi-Layered Security: Thangam Jewellery incorporates multi-layered security protocols to ensure the utmost protection of user data and financial transaction.

Dynamic Pricing: The platform employs dynamic pricing algorithms to adjust product prices based on market trends, ensuring competitiveness and value for customers.

Social Media Integration: Users can seamlessly share their favourite jewellery items on social media platforms, promoting user engagement and expanding the brand's reach.

Real-Time Inventory Management: The system includes real-time inventory tracking, allowing administrators to monitor stock levels and update product availability promptly.

Educational Content: Thangam Jewellery provides informative content about jewellery materials, craftsmanship, and care tips to educate and engage users, fostering a deeper connection with the brand.

Interactive Customer Reviews: Users can leave detailed reviews and ratings for products, contributing to an interactive community and assisting potential buyers in their decision-making process.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

A feasibility study is an analysis of the viability of an idea, project, or proposal that determines if it is practical, realistic, and capable of being successfully completed. It typically includes an examination of the technical, financial, operational, legal, and market aspects of a project to determine its potential for success. The purpose of a feasibility study is to provide decision-makers with the information they need to determine whether to proceed with a project, and if so, how best to move forward. The purpose of a feasibility study is to evaluate the potential of a proposed project or idea, determine its viability, and provide decision-makers with the information they need to make informed decisions.

- Various feasibility studies are:
- Economic Feasibility
- Technical Feasibility
- Behavioural Feasibility

3.1.1 Economic Feasibility

This section evaluates the economic aspects of The Thangam Jewellery. The initial cost estimation includes expenses for software development, hardware, payment gateway fees, and AR technology integration. A detailed breakdown is required. Revenue projections should consider sales volumes, pricing strategies, and transaction fees from payment gateways. Determine the payback period for the project. Assess the overall economic feasibility by comparing the project's benefits to its costs. Priority Calculation: Assign priorities based on the significance and urgency of each economic factor. Consider factors such as cost components, revenue streams, and payback period when assigning priorities. This will help in focusing efforts on critical aspects of economic feasibility.

3.1.2 Technical Feasibility

This section evaluates the technical aspects of implementing The Thangam Jewellery. Front-end: HTML/CSS is a widely accepted choice for building user-friendly interfaces. It provides the necessary tools for creating a visually appealing and responsive website. Back-end: Python Django is a robust and scalable framework for handling the complex functionalities required for e-commerce, including user accounts, product management, and payment processing. Implementing virtual try-on functionality using augmented reality (AR) technology may require specialized development skills and resources. The feasibility depends on the availability of AR development

expertise and tools .Integration with multiple payment gateways, including Google Pay and other online methods, is technically feasible but may involve transaction costs and security considerations .Priority Calculation: Evaluate the technical aspects based on their impact on project success and implementation timelines. Prioritize tasks and technologies by considering their dependencies and potential bottlenecks

.3.2.3 Behavioural Feasibility

The Thangam Jewellery project aligns with user behaviours, offering a seamless online shopping experience. The familiarity of HTML/CSS for front-end and Python Django for back-end ensures user-friendly interfaces and robust functionalities. The integration of virtual try-on through AR technology caters to evolving user expectations, enhancing engagement. Standard operational features like user registration and order processing align with common online shopping behaviours. Ongoing attention to product updates and customer communication supports sustained user satisfaction. Priority Calculation: Assess the behavioural aspects based on their impact on user experience and satisfaction. Prioritize features and updates that directly influence user engagement and contribute to long-term customer satisfaction

.3.1.4 Feasibility Study Questionnaire.

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

Users

These are the end-users of your e-commerce platform, primarily interested in purchasing jewellery. Users interact with the platform to explore, browse, and purchase jewellery items. They also use various features, such as searching for products, creating wishlist, trying on jewellery virtually, and making payments.

Admins

Admins are responsible for managing and overseeing the entire platform. Admins have a wide range of responsibilities, including adding, editing, and deleting products, managing product categories, processing and managing customer orders, managing customer accounts, handling customer support inquiries, updating website content, checking jewellery purity, processing returns and refunds, and ensuring the platform's security and performance.

Vendors

Vendors are individuals or businesses who sell their jewellery products on your platform. Vendors use the platform to manage their incoming orders, mark orders as shipped, update order statuses, define their return, refund, and exchange policies, and potentially add or edit their product listings.

List the Modules included in your System?

Customer: Modules, registration, product browsing, cart management, wish-list, checkout payment and feedback.

Admin module: Dashboard, product management, order management, customer management Sales analytics.

Vendor modules: Vendor login, product management, feedback viewing Chatbots and Customer support

Identify the users in your project?

Customers: Register, browse, shop, provide feedback. Sellers: Manage products, view feedback. Admin: Oversee all aspects of the system, manage products, orders, customers, and analytics

Who owns the system?

A Jewellery Retailer: If Thangam Jewellery is the online platform of a physical jewellery store or brand, the retailer would own and operate the system. An E-commerce Business: If Thangam Jewellery is an independent online jewellery retailer, it may be owned and operated by an individual or a team running the e-commerce business. A Technology Company: In some cases, a technology company or development agency may develop and maintain the e-commerce platform on behalf of a jewellery retailer or business owner. A Partnership: Thangam Jewellery could also be a joint venture or partnership between multiple entities, where ownership and operational responsibilities are shared.

System is related to which firm/industry/organization?

This system is related to the e-commerce industry, specifically for selling jewellery

Details of person that you have contacted for data collection?

It seems like you're asking for the details of a person contacted for data collection, but there is

no specific information provided in the project description about a person or entity that was contacted for data collection. It's important to note that data collection for a project like this typically involves gathering product information from various sources such as suppliers, manufacturers, or through data entry by project team members. If you have a specific person or entity in mind for data collection, you would typically need to provide their contact information, role, and responsibilities in the project. However, based on the information provided in your project description, it doesn't specify a particular person or entity for data collection. If you have any specific questions or need guidance on how to approach data collection for your project, please feel free to ask, and I'll be happy to provide assistance.

1.How the user check the purity of the gold?

On each product page, prominently display the purity information for the gold used in the jewellery. This information should include the carat (e.g., 18K, 22K) and any other relevant quality attributes.

2.How the admin check the purity of the gold?

Provided Information: When adding a new jewellery item to the website's database, the admin may rely on information provided by the supplier or manufacturer. This information could include the gold's carat (e.g., 18K, 22K) and any associated certifications. BIS Standards: The admin would need to be familiar with the purity standards set by the Bureau of Indian Standards (BIS) for gold jewellery. In India, jewellery is typically marked with the BIS hallmark, which indicates the gold's purity. For example, 22K gold has 91.6% purity.

3.How do you determine the pricing of your jewellery products?

The rate of the products is depend upon the current rate of gold, diamond, platinum and silver

4.What payment methods do you accept on your website?

Google Pay: Many websites support digital wallets like Google Pay, enabling users to make secure payments using their mobile devices.

5.Describe the overall design and layout of your website?

Homepage: The homepage welcomes users with an elegant and clean design .A visually striking hero banner displays featured jewellery pieces or promotions, accompanied by high-quality images .Product Listing: Product listings are presented in a visually appealing grid format, displaying

high-quality images of jewellery pieces .Each product listing includes essential details like the product name, price, and a brief description. Product Pages: Individual product pages provide comprehensive information about each jewellery piece. High-resolution images from various angles and zoom functionality allow users to examine the product in detail .User Account: The user account section allows customers to register, log in, and manage their profiles .Users can track their order history, manage shipping addresses, and update personal information. Navigation: The navigation menu is clear and intuitive, allowing users to explore different jewellery categories such as rings, necklaces, earrings, and more .A search bar at the top of the page enables users to quickly find specific products.

6.What customer support options do you provide?

Contact Information: Provide clear and easily accessible contact information, including phone numbers and email addresses, for customers to reach out for assistance Email Support: Offer email support for customers to send inquiries or report issues. Ensure timely responses and clear communication via email Virtual Assistant or Chatbot: Use AI-powered chatbots to handle routine queries and provide automated assistance to customers. Chatbots can help with order tracking, basic product information, and more.

7.What is your return policy for jewellery purchased on your website?

Return Eligibility: Jewellery items are eligible for return within numbers days from the date of delivery. To be eligible for a return, the jewellery item must be in its original condition, including the original packaging, tags, and any accompanying certificates of authenticity Damaged or Defective Items: If a jewellery item arrives damaged or with a manufacturing defect, customers should contact our customer support team immediately. We may request clear photos or additional information to assess the issue. Depending on the situation, we will arrange for a replacement, repair, or refund. Refund exceptions in certain cases, refunds may be subject to deductions for restocking fees or refurbishing costs, especially for items that require extensive restoration.

8.Can you provide a brief description of this website's mission and purpose?

E-commerce Excellence: Thangam Jewellery aims to offer a user-friendly interface where customers can easily navigate and explore various categories of jewellery. The website's purpose is to provide a secure and efficient platform for customers to browse and make jewellery purchases online Product Showcase: The website serves as a showcase for a diverse range of jewellery items, including rings, necklaces, earrings, and more. Its purpose is to present detailed product

descriptions and high-quality images to help customers make informed purchasing decisions.

Customer Engagement: Thangam Jewellery strives to engage with customers by providing features such as user accounts, Wishlist and a virtual try-on feature using augmented reality. The purpose is to enhance the overall shopping experience and build a loyal customer base

Inventory Management: The website's purpose includes efficient inventory management to ensure that products are accurately represented in terms of availability, materials used, and pricing. It also

aims to facilitate order processing and fulfilment for administrators

Customer Support: Thangam Jewellery is committed to providing customer support through various channels, including a chatbot and contact information for assistance. The purpose is to address customer inquiries and

issues promptly and effectively

Security: Ensuring secure payment gateways and the protection of customer data is a fundamental purpose of the website. It aims to provide a safe and trustworthy environment for online transactions.

9. Do you provide certificates of authenticity for certain products?

Precious Metals: Certificates can confirm the purity and authenticity of precious metals like gold, platinum, or silver

Designer or Branded Products: Some luxury and designer brands provide certificates of authenticity with their products to assure customers of their authenticity.

10. How do you ensure the quality and authenticity of the jewellery you sell?

Product Sourcing: Jewellery sellers typically establish relationships with reputable suppliers, manufacturers, or artisans with a history of producing high-quality jewellery

Hallmarks and Stamps: Precious metal jewellery, like gold or silver, may bear hallmarks or stamps indicating the purity and authenticity of the metal.

Transparency in Product Descriptions: Sellers should provide clear and accurate descriptions of their jewellery, including details about the materials used, gemstone grades, and any treatments or enhancements applied.

3.1 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor - Intel Core i3 or equivalent

RAM - 4 GB or higher

Hard disk - 256 GB or higher

3.2.2 Software Specification

Front End - HTML, CSS, JAVA SCRIPT

Back End - DJANGO

Database	- SQLite3
Client on PC	- Windows 7 and above.
Technologies used	- JS, HTML, Sqlite3, CSS,

3.3 SOFTWARE DESCRIPTION

3.3.1 DJANGO

Django is a high-level Python web framework that facilitates rapid development and clean, pragmatic design. It follows the Model-View-Controller (MVC) architectural pattern, often referred to as Model-View-Template (MVT) in Django's context. Django provides a set of tools and libraries for building web applications, including an Object-Relational Mapping (ORM) system for database management, URL routing, template rendering, and more. Known for its "Don't Repeat Yourself" (DRY) philosophy, Django encourages efficient and reusable code. It also emphasizes security measures by default and supports the development of scalable and maintainable web applications. Popular for its simplicity and versatility, Django is widely used for developing a variety of web projects, from small applications to large-scale, complex systems.

3.3.2 SQLite3

SQLite3 is a lightweight, serverless, and self-contained relational database management system (RDBMS). It is a C library that provides a relational database engine accessible through a simple programming interface. Unlike traditional client-server databases, SQLite3 is embedded directly into the application, making it a popular choice for small to medium-sized applications, mobile apps, and scenarios where a standalone, file-based database is sufficient.

CHAPTER 4
SYSTEM DESIGN

4.1 INTRODUCTION

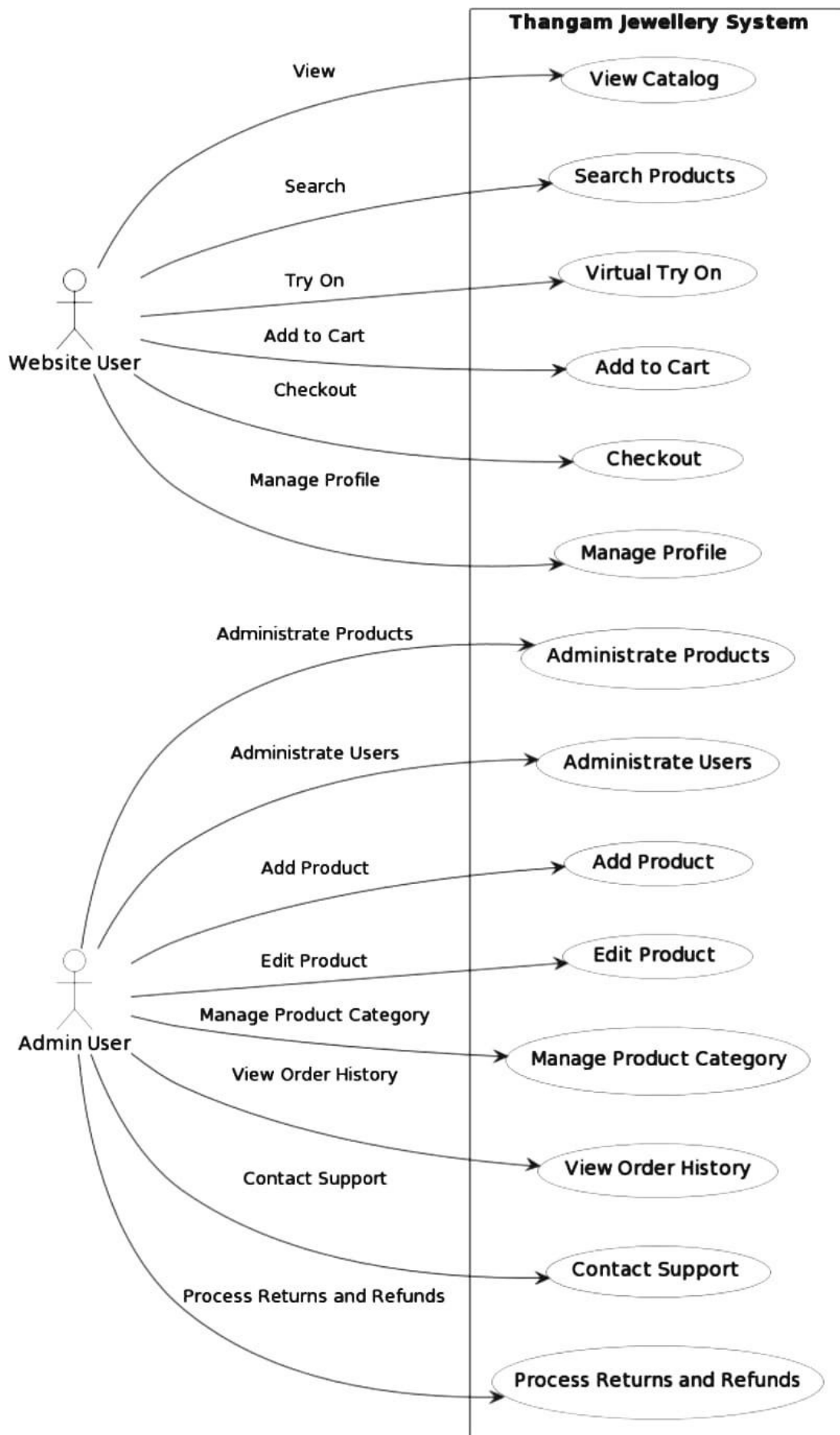
System design is an important phase in the project development process, which involves creating a high-level plan for how the system will be built and how it will function. It is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy the specified requirements. During system design, the project team defines the system's overall structure, and how its various components will interact and integrate. They consider the software and hardware components that will be needed, as well as the user interface and data flow. System design also involves defining the system's security, scalability, reliability, and performance characteristics. The system design process often involves the creation of diagrams, flowcharts, and other visual aids to help illustrate how the system will work. The resulting system design document serves as a blueprint for the development team, helping them to create the system more efficiently and accurately. Overall, system design is to create a detailed plan for how the system will be built and how it will function, which will help ensure that the system meets the needs of its users and stakeholders.

4.2 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 Modelling 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 modelling 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.

4.2.1 USE CASE DIAGRAM:

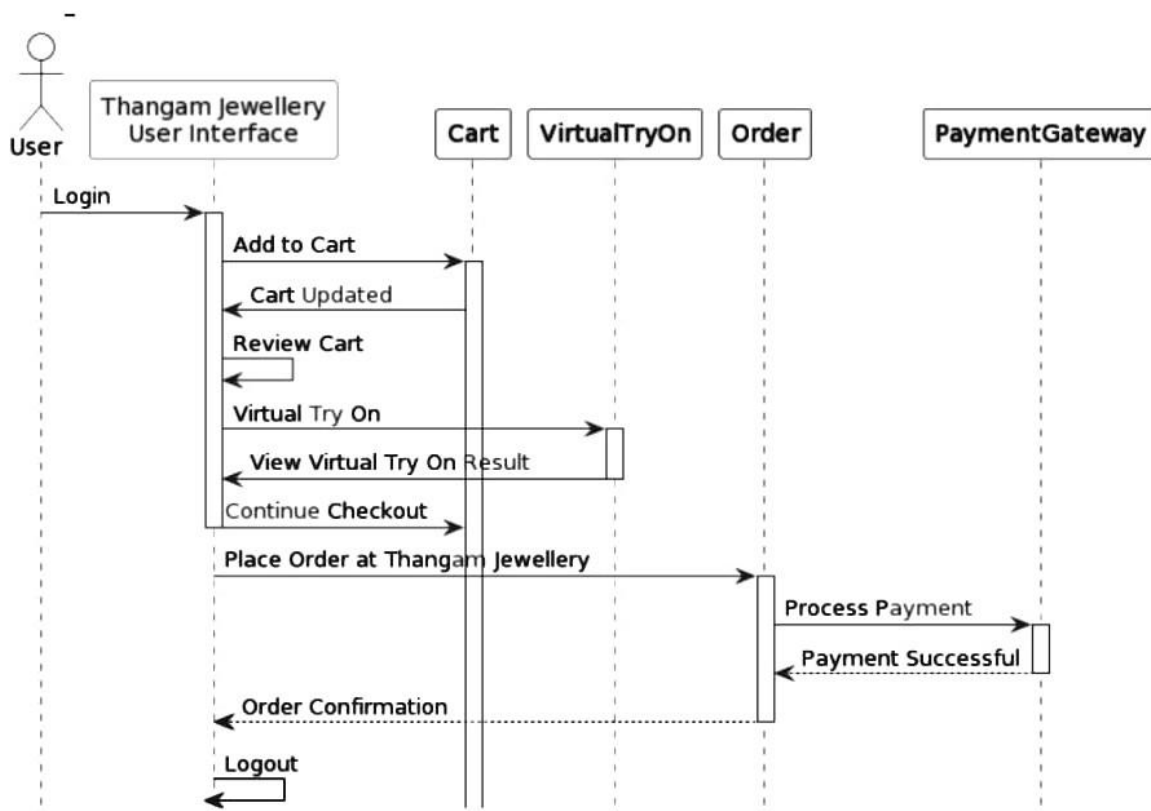
A use case diagram is a type of visual representation that illustrates the interactions between actors (users or external systems) and a system in terms of use cases (actions or activities). It's a powerful tool used during the requirements analysis phase of software development to understand and communicate the system's functionality and how it will be used by various actors. The use case diagram helps stakeholders understand the requirements of the system, and provides a visual guide to help identify potential gaps or issues in the system's design. It also serves as a blueprint for the development team, helping them to understand what functionality the system needs to provide and how it will be used by various actors. Use case diagrams are typically created by identifying the different actors who will interact with the system, such as users or other systems, and then identifying the various use cases or actions that those actors will perform. Each use case describes a specific task or functionality that the system should be able to perform. The actors and use cases are represented as graphical elements in the use case diagram, with arrows indicating the interactions between them.



4.2.2 SEQUENCE DIAGRAM:

A sequence diagram is a type of UML diagram that shows the interactions between objects or components in a system over time. It uses vertical lines to represent objects and horizontal arrows to show the order of messages or events between them. Sequence diagrams are commonly used in software development to visualize the behaviour of systems and identify potential issues.

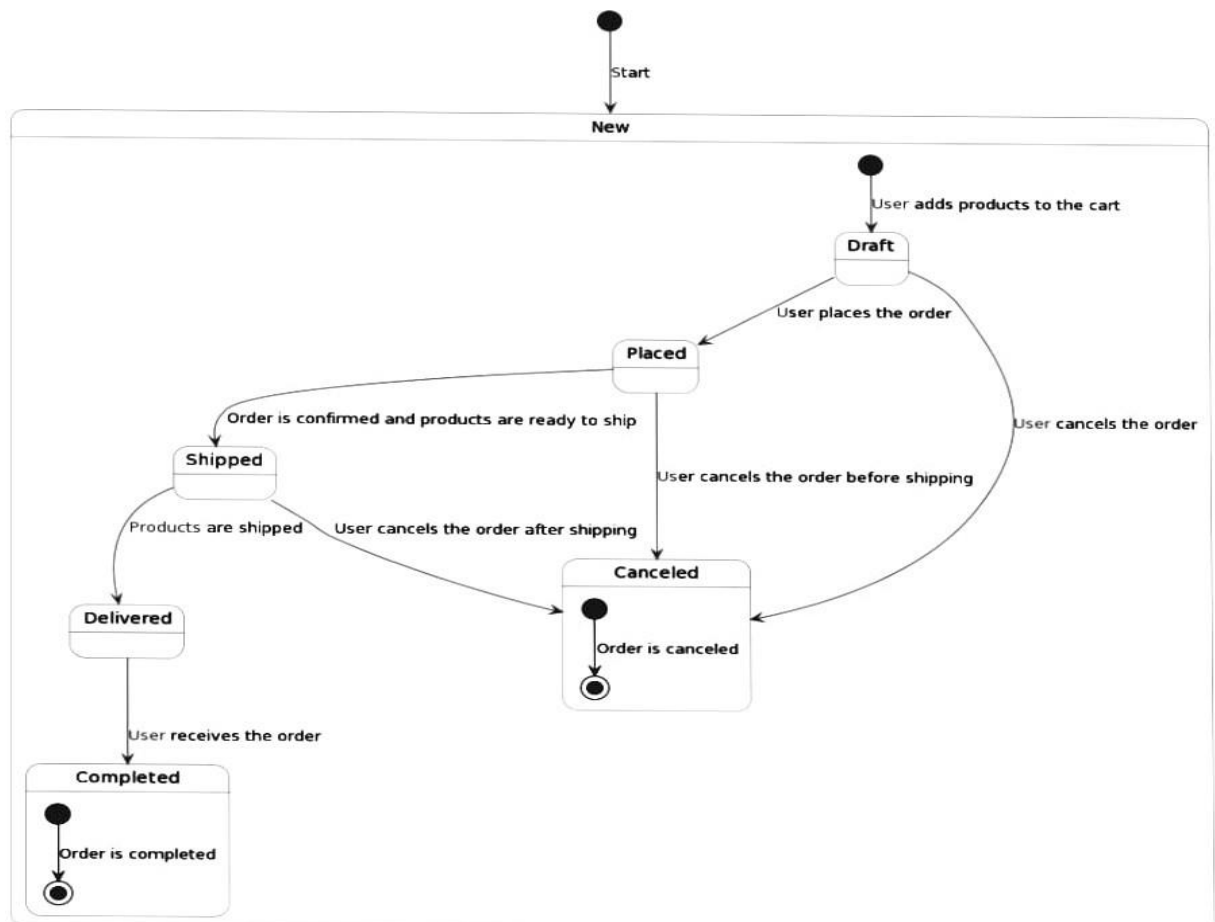
Sequence diagrams consist of the following elements: **Actors:** Actors are the objects or systems that interact with each other. They are represented by vertical lines called lifelines. **Messages:** Messages represent the interactions between actors or objects. Messages are represented by arrows that connect the lifelines of the actors. **Activation:** Activation represents the time interval during which an actor is executing an operation. Activation is represented by a thin rectangle on the lifeline of the actor. **Return:** Return represents the return of a value or result to the calling actor. Return is represented by a dashed arrow.



4.2.3 STATE CLASS DIAGRAM

A state chart diagram is a type of UML diagram that depicts the behaviour of a system or an object over time. It represents the various states that an object or a system can go through during its lifetime, and the transitions between those states. State chart diagrams are often used to model the behaviour of complex systems, such as software applications, communication protocols, or business

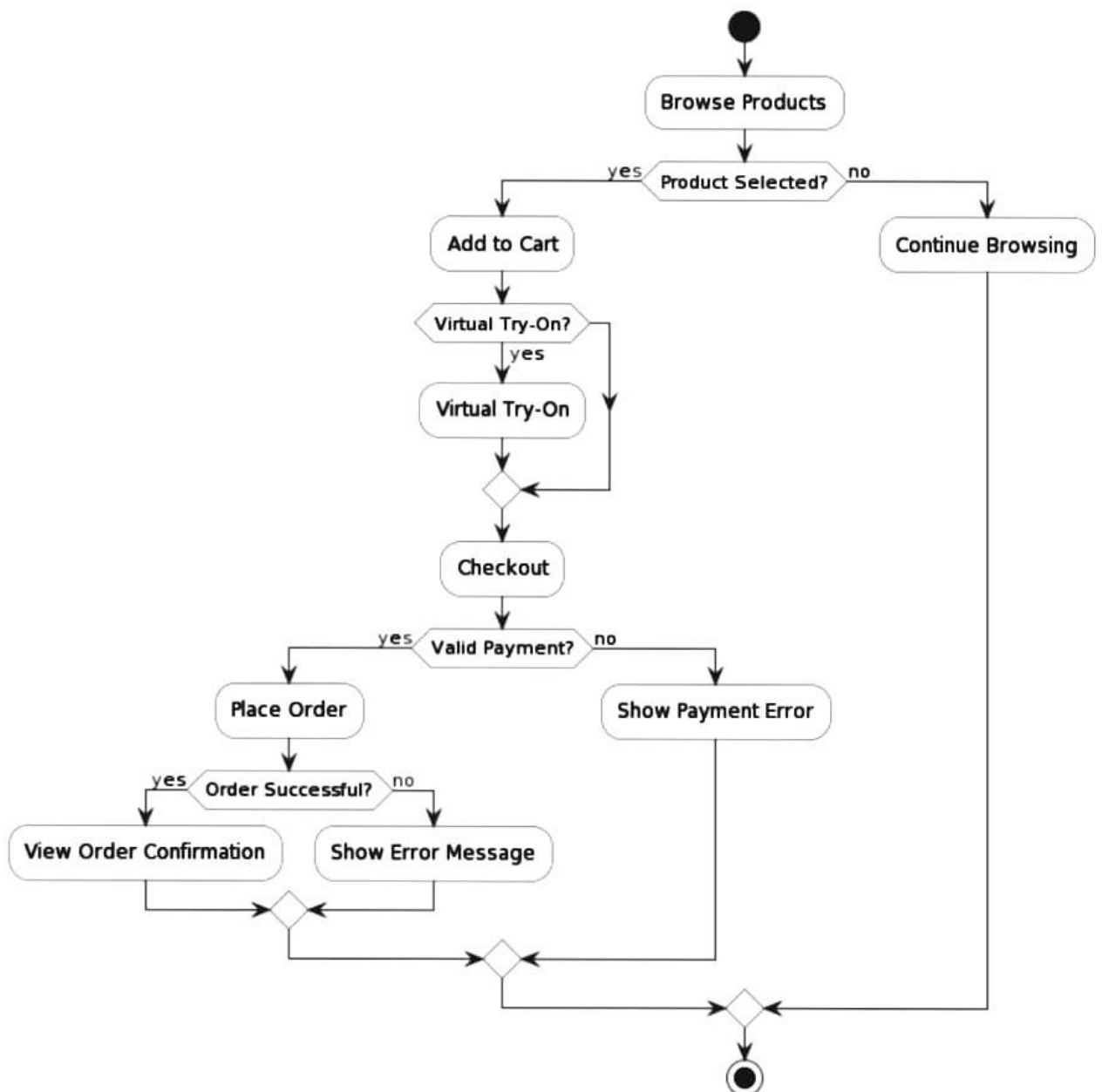
workflows. They can be used to visualize the different states that a system can be in, and the events or actions that trigger transitions between those states. In a state chart diagram, states are represented as rounded rectangles, and transitions are represented as arrows between the states. Each state can have entry and exit actions, which are performed when the object or system enters or leaves the state. Additionally, states can have internal actions, which are performed while the object or system is in that state. State chart diagrams can be a useful tool for understanding the behaviour of complex systems, and for communicating that behaviour to other stakeholders in a clear and concise manner.



4.2.4 ACTIVITY DIAGRAM:

An activity diagram is a graphical representation of the steps, actions, and decisions involved in a process or workflow. It is a type of UML (Unified Modelling Language) diagram used in software engineering and other fields to describe and document the steps of a system or process.

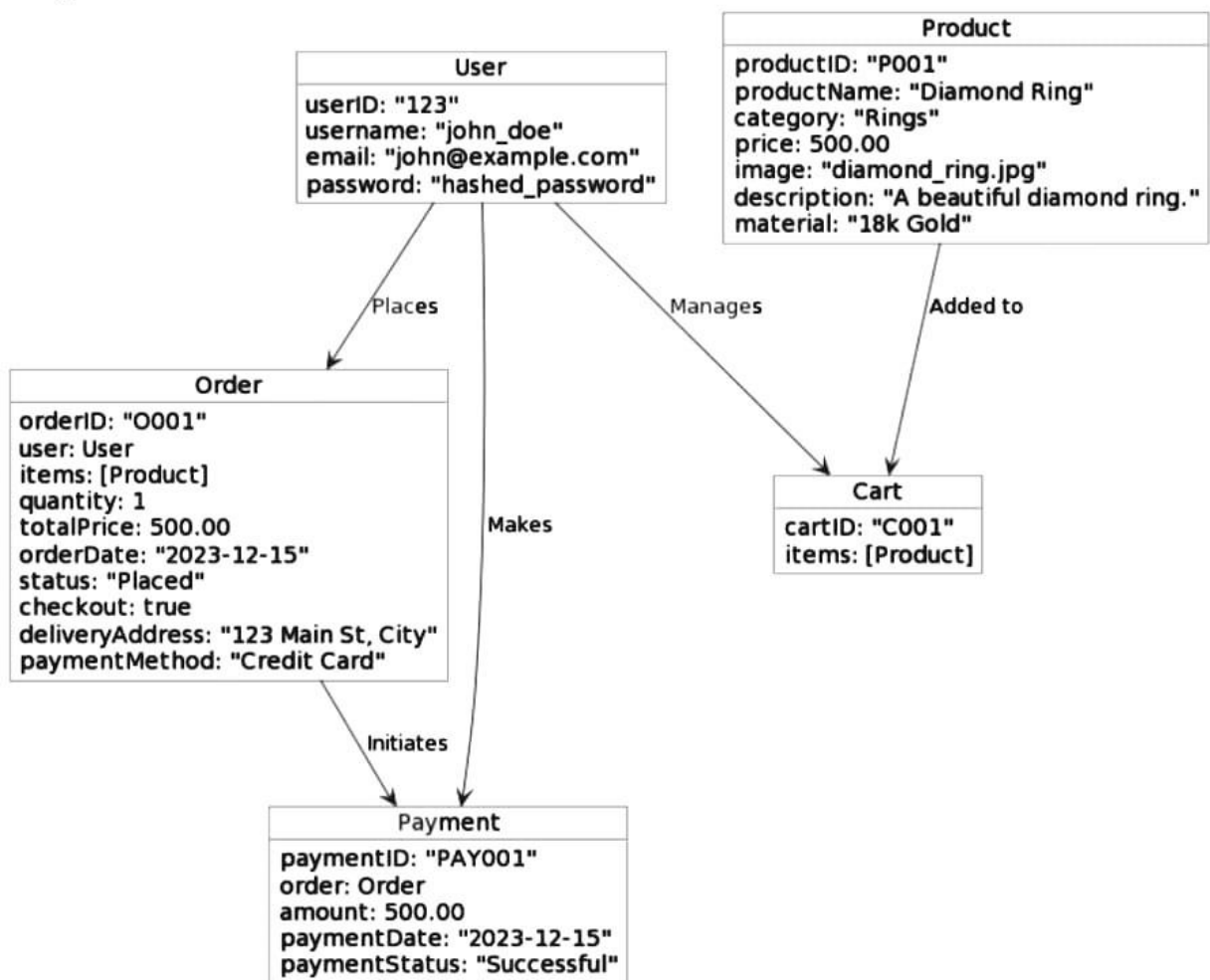
Activity diagrams consist of nodes and edges. Nodes represent actions, decisions, or other types of activity, while edges represent the flow of control between nodes. Arrows indicate the direction of flow, and symbols and labels can be used to indicate branching, synchronization, and other aspects of the process. Activity diagrams are useful for modelling complex workflows and for communicating the steps and logic of a process to stakeholders.



4.2.5 CLASS DIAGRAM:

A class diagram is a type of UML diagram that shows the structure of a system or software application by modelling its classes, attributes, operations, and relationships. It's used to represent the static structure of a system, including its objects, classes, and their relationships. Class diagrams consist of the following elements:

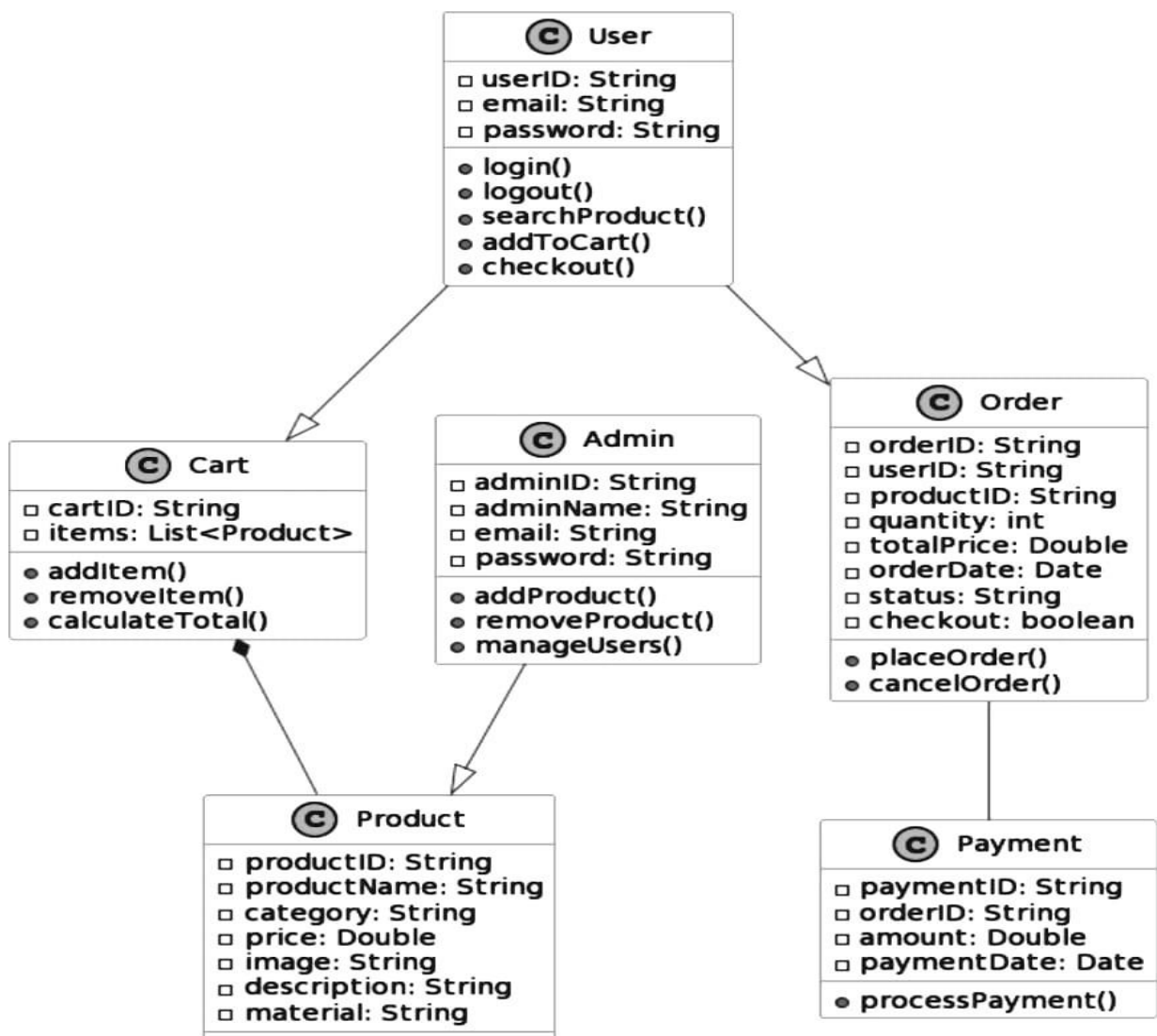
- Classes:** Classes represent a collection of objects that share similar attributes and behaviours. They are represented by rectangles with the class name at the top.
- Attributes:** Attributes represent the characteristics or properties of a class, such as its name, type, and value. They are represented by smaller rectangles inside the class rectangle.
- Operations:** Operations represent the actions or behaviours that can be performed on a class, such as methods or functions. They are represented by rectangles inside the class rectangle with the operation name and parameter list.
- Relationships:** Relationships represent the connections or associations between classes, such as inheritance, composition, aggregation, and association. They are represented by lines connecting the classes, with arrows indicating the direction of the relationship.



4.5.6 OBJECT DIAGARM:

An object diagram in UML (Unified Modelling Language) is a type of structural diagram that shows a snapshot of the instances of classes in a system at a particular moment in time. It provides a visual representation of objects, their attributes, and the relationships between them. Object diagrams are used to illustrate specific scenarios or use cases in a system, and they can help to verify the correctness of class diagrams and other UML models.

They are particularly useful for understanding how objects interact with each other and for testing and validating system designs. In an object diagram, objects are represented as rectangles with the name of the object inside. The attributes of the object are listed below the name, and their values are shown in the rectangle. The relationships between objects are shown as lines between the objects, with arrows indicating the direction of the relationship.

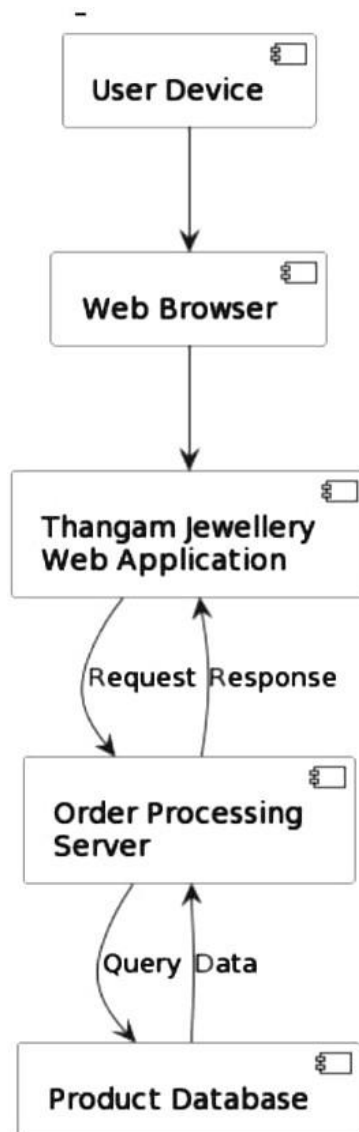


4.2.7 Component Diagram

A component diagram in UML is a type of structural diagram that shows the dependencies, relationships, and interactions among the components of a system. Components are self-contained, modular parts of a system that provide specific functionality, and can be built and deployed independently. In a component diagram, the components are represented as rectangles with their name inside. The dependencies between the components are shown as lines connecting the components, with an arrow indicating the direction of the dependency. The interface of a component is represented by a small rectangle or circle on the border of the component. The interface specifies the operations or services that a component provides or requires. Component diagrams can help to visualize the overall architecture of a system, and to identify areas where the system can be divided into separate components.

4.2.8 Deployment Diagram

A deployment diagram in UML is a type of structural diagram that shows how software and hardware components of a system are deployed and configured in a physical environment. Deployment diagrams are typically used to model the deployment of software systems, including client-server and distributed systems. In a deployment diagram, the nodes are used to represent the hardware and software components that make up the system. Nodes are represented as rectangles with the name of the node inside. Artifacts, which are the software components that are deployed to the nodes, are represented as rectangles with the name of the artifact inside. Deployment relationships, which indicate how artifacts are deployed to nodes, are represented as dashed lines with an arrow indicating the direction of the deployment.

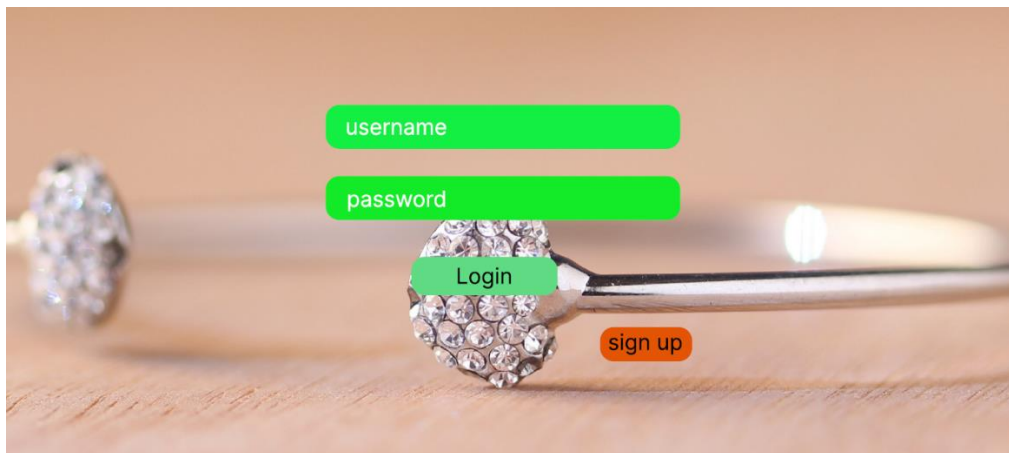


4.2.9 Collaboration Diagram

A collaboration diagram in UML is a type of behavioural diagram that shows the interactions and communications between objects or roles in a system. Collaboration diagrams are used to illustrate the dynamic behaviour of a system, and to help visualize and understand how objects or roles work together to achieve a specific goal. In a collaboration diagram, the objects or roles involved in the interaction are represented as rectangular boxes with the name of the object or role inside.

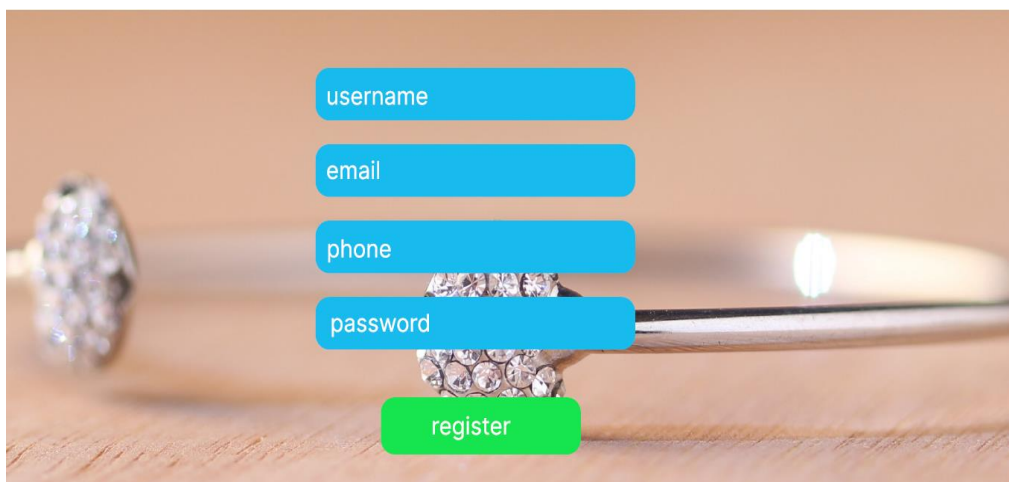
4.3 USER INTERFACE DESIGN USING FIGMA

Form Name: Login Form



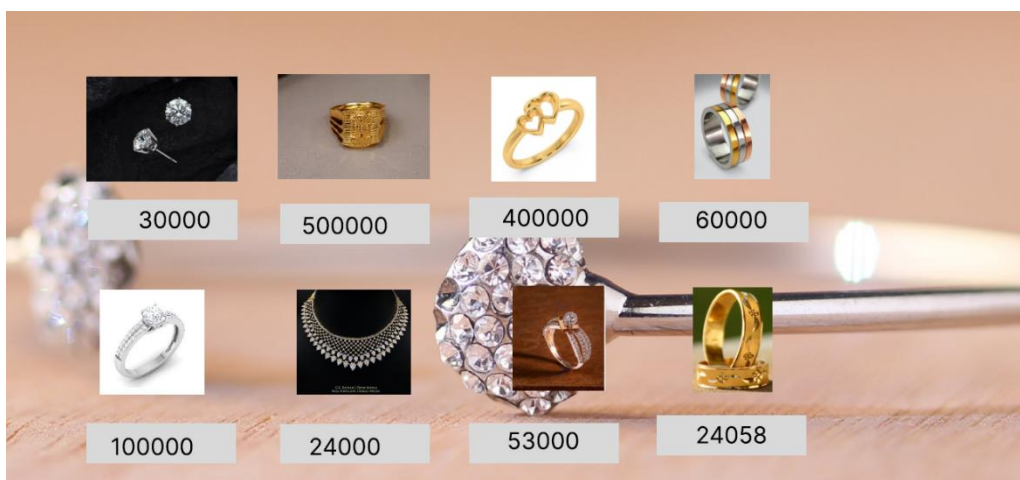
A login form UI design overlaid on a background image of a diamond ring. The form consists of two input fields: 'username' and 'password', both with green borders. Below these fields are two buttons: a green 'Login' button and an orange 'sign up' button.

Form Name: Register Form











A register form UI design overlaid on a background image of a diamond ring. The form consists of four input fields: 'username', 'email', 'phone', and 'password', all with blue borders. Below these fields is a green 'register' button.

Form Name: Product Form



A product form UI design overlaid on a background image of a diamond ring. The form displays eight product images arranged in two rows of four. Each image is accompanied by a price tag below it. The products and their prices are:

Product Image	Price
	30000
	500000
	400000
	60000
	100000
	24000
	53000
	24058

4.3 DATABASE DESIGN

4.4.1 Relational Database Management System (RDBMS)

Relational Database Management System (RDBMS) is a software system that manages relational databases. A relational database is a database that organizes data into one or more tables, where each table has a unique identifier known as a primary key, and each column in the table represents a different attribute or piece of information about the data. RDBMSs are used to manage and manipulate data stored in relational databases. They provide users with a way to query, update, and manage the data in the database using SQL (Structured Query Language). SQL is a standardized programming language used to interact with RDBMSs and is used to create and modify database schemas, manipulate data stored in the database, and retrieve data from the database. Some popular RDBMSs include MySQL, Oracle Database, Microsoft SQL Server, PostgreSQL, and SQLite. These systems vary in terms of their capabilities, features, and cost, but all provide a reliable and efficient way to manage relational databases.

4.4.2 NORMALIZATION:

Normalization is the process of organizing and structuring data in a relational database to reduce data redundancy and dependency. The goal of normalization is to eliminate data anomalies and inconsistencies, ensure data integrity, and improve database performance. Normalization involves dividing a database into smaller, more manageable tables and defining relationships between them. There are several levels of normalization, referred to as Normal Forms, with each level building upon the previous one. First Normal Form (1NF): This requires that each table has a primary key and that every attribute in the table is atomic, meaning it cannot be further divided into smaller values. Second Normal Form (2NF): This requires that all non-key attributes in a table are functionally dependent on the entire primary key, rather than just a portion of it. Third Normal Form (3NF): This requires that all non-key attributes in a table are dependent only on the primary key and not on any other non-key attributes.

Other normal forms, such as Fourth Normal Form (4NF), Fifth Normal Form (5NF), and Boyce-Codd Normal Form (BCNF), exist beyond 3NF, but they are less commonly used. By normalizing a database, you can reduce data redundancy and improve data consistency and accuracy, which can lead to better overall database performance and more efficient querying.

4.4.3 Sanitization

Sanitization in RDBMS refers to the process of removing or masking sensitive or confidential data from a database to protect the privacy of individuals or organizations and comply with regulations such as GDPR or HIPAA. This involves methods such as data masking, data scrambling, data removal, or data substitution. The objective is to ensure that sensitive data is not accessible or identifiable in the database, thus reducing the risk of unauthorized access or data breaches.

4.4.4 Indexing

Indexing is a technique used in databases to improve the speed of queries by creating a data structure that allows for quick access to specific rows of data. An index is created on one or more columns of a table, and contains a sorted list of values and pointers to the actual data in the table. When a query is executed on a table with an index, the database can use the index to quickly locate the rows that match the search criteria, without having to scan the entire table. This can significantly improve the performance of queries that involve large amounts of data. There are several types of indexing techniques, including B-Tree indexing, hash indexing, and bitmap indexing. B-Tree indexing is the most common and is used for range queries and partial matches. Hash indexing is faster for exact matches but less efficient for range queries or partial matches. Bitmap indexing is used for columns with a small number of distinct values.

4.5 TABLE DESIGN

USER:

Primary key: user _id

No	Fieldname	Datatype (size)	Key constraints	Description of the field
1	User id	Int	Primary key	This is a unique identifier for each user in the system.
2	Username	Varchar (30)	Not null	This field stores the name of the user.
3	Email	Varchar (30)	Not null	This field stores the email address of the user.
4	Password	Varchar (30)	Not null	This field stores the password of the user.
5	Address	Varchar (255)	Not null	User's address.
6	Pin code	Varchar (10)	Not null	Pin code associated with the user's location.
7	Gender	Varchar (10)	Not null	Gender of the user.
8	Phone Number	Varchar (15)	Not null	Phone number of the user.

WISHLIST:

Primary key: Wishlist id

Foreign key: user id references table user

product id references table customer

No	Fieldname	Datatype (size)	Key constraints	Description of the field
----	-----------	-----------------	-----------------	--------------------------

1	Wishlist id	Int	Primary key	This is a unique identifier for each wish list.
2	User id	Int	Foreign key	This field links to the user who created the wish list.
3	Product id	Int	Foreign key	This field links to the product added to the wish list.

PRODUCT:

Primary key: product id

Foreign key: category id references table category

No	Fieldname	Datatype (size)	Key constraints	Description of the field
1	Product id	Int	Primary key	This is a unique identifier for each product in the system.
2	Category id	int	Foreign key	This field links to the category to which the product belongs. it's like Platinum, Gold, diamond. reference table is category
3	Product name	Varchar (30)	Not null	This field stores the name of the product.

4	Price	Decimal (10,2)	Not null	This field stores the price of the product.
5	Description	Text	Not null	This field stores detailed information about the product, including GST and BIS purity for gold.

ORDER:

Primary key: order _id

Foreign key: user id references user, product id references category

No	Fieldname	Datatype (size)	Key constraints	Description of the field
1	Order id	Int	Primary key	This is a unique identifier for each order in the system.
2	User id	Int	Foreign key	This field links to the user who placed the order.
3	Product id	Int	Foreign key	This field links to the product that was ordered.
4	Order date	Date	Not null	This field stores the date when the order was placed.
5	Quantity	int	Not null	This field stores the quantity of

				the product ordered.
6	Total cost	Decimal (10,2)	Not null	This field stores the total cost of the order.
7	GST	Decimal (5,2)	Not null	This field stores the Goods and Services Tax (GST) applied to the order.

CHATBOT MESSAGE:

Primary key: chatbot message id, message

Foreign key: user id references user

No	Fieldname	Datatype (size)	Key constraints	Description of the field
1	Chatbot Message id	Int	Primary key	This is a unique identifier for each chatbot message.
2	User id	Int	Foreign key	This field links to the user who sent the chatbot message.
3	Message id	int	Primary key	This field stores a unique identifier for each chatbot message.
4	Message txt	Text	Not null	This field stores the text content of the chatbot message.

Feedback:

Primary key: feedback id

Foreign key: user id reference user, productid references customer

No	Fieldname	Datatype (size)	Key constraints	Description of the field
1	Feedback id	Int	Primary key	This is a unique identifier for each feedback entry.
2	User id	Int	Foreign key	This field links to the user who provided the feedback.
3	Productid	Int	Foreign key	This field links to the product for which the feedback is provided.
4	Rating	Int	Not null	This field stores the rating provided by the user for the product.
5	Command	text	Not null	This field stores the feedback or comments provided by the user.

GIFT CARD:

Primary key: Gift id.

Foreign key: user id references user

No	Fieldname	Datatype (size)	Key constraints	Description of the field
1	Gift id	int	Primary key	This is a unique identifier for each gift card.
2	User id	Int	Foreign key	This field links to the user who owns the gift card.
3	Amount	Decimal (10,2)	Not null	This field stores the monetary amount associated with the gift card.

CHAPTER 5

SYSTEM TESTING

5.1 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 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 actually 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 actually executing the code. Dynamic analysis looks at the behaviour 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.

5.2 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

5.2.1 Unit Testing

Unit testing is a software testing technique where individual units or components of a software application are tested in isolation from the rest of the application. The objective of unit testing is to identify and isolate defects in individual units or modules of the software before the software is integrated and tested as a whole. The process of unit testing involves writing and executing automated test cases for each individual unit or module. The test cases are designed to verify that the unit performs as intended, meets its functional and non-functional requirements, and produces the expected output for a given input. Unit testing is typically performed by developers during the development phase of software. The benefits of unit testing include the ability to catch defects early in the development cycle, reduce the cost of defect fixing, improve the quality of the software, and reduce the risk of defects in the overall system. Unit testing is often integrated into continuous integration and continuous delivery (CI/CD) pipelines, where automated tests are run every time a code change is made. This ensures that any defects introduced by the code changes are caught early in the development cycle, reducing the risk of defects in the overall system.

5.2.2 Integration Testing

Integration testing is a type of software testing where different modules or components of a software system are tested together to verify that they function correctly when integrated. The objective of integration testing is to identify defects in the interaction between different components of the system, including data transfer, control flow, and communication protocols. Integration testing can be performed at various levels, including component integration testing, where individual components are tested together, and system integration testing, where the complete system is tested. The process of integration testing involves creating test cases that cover all possible scenarios of integration between components, including positive and negative test cases, boundary conditions, and error conditions.

Integration testing is typically performed after unit testing is completed, and individual components of the system are verified to work correctly. The benefits of integration testing include the ability to identify defects that arise from the interaction between different components, reduce the risk of

defects in the overall system, and ensure that the system meets its functional and non-functional requirements.

5.2.3 Validation Testing or System Testing

This is the final step in testing. In this the entire system was tested as a whole 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

5.2.4 Output Testing or User Acceptance Testing

Output testing is a type of software testing that verifies the accuracy and completeness of the output generated by a system. This can include reports, documents, or other artifacts that the system produces. The objective of output testing is to ensure that the output meets the specified requirements and is consistent with the expected output. User acceptance testing (UAT), on the other hand, is a type of software testing that involves end-users or stakeholders testing the system to ensure that it meets their requirements and expectations. The objective of UAT is to verify that the system is fit for its intended use and satisfies the needs of the end-users or stakeholders. Both output testing and UAT are important in ensuring the quality and usability of a software system. While output testing verifies that the system produces accurate and complete output, UAT ensures that the system meets the business objectives and satisfies the needs of the end-users or stakeholders. Ultimately, both types of testing play a crucial role in ensuring that the software system meets its intended purpose and is of high quality.

5.2.5 Automation Testing

Automation testing is a software testing approach that uses tools and software to automate the execution of test cases and scenarios. The objective of automation testing is to reduce the time, effort, and cost of testing by automating repetitive and time-consuming tasks. Automation testing can be used for different types of testing, including functional testing, regression testing, performance testing, and load testing. It involves creating test scripts using automation testing tools and executing them using automation frameworks. The benefits of automation testing include

improved test coverage, increased efficiency and accuracy, faster test execution, and reduced testing costs. Automation testing can also help identify defects early in the development cycle, enabling developers to fix them before they become more significant and costly to fix. However, automation testing requires significant upfront investment in terms of time and resources to set up and maintain the automation framework and test scripts. It is also not suitable for all types of testing, and manual testing may still be required for certain scenarios.

5.2.6 Selenium Testing

Selenium is an open-source testing framework that is widely used for automating web browsers. It enables developers and testers to automate the testing of web applications across different browsers and platforms, by simulating user interactions such as clicking buttons, filling forms, and navigating through pages. Selenium supports various programming languages like Java, Python, and C#, making it a versatile and flexible tool for automated testing. It also has a robust set of features, including the ability to create test scripts, execute tests in parallel, and integrate with other testing tools. Overall, Selenium has become an essential tool for ensuring the quality and functionality of web applications.

Test Case 1:

Login page Test Code

```
def test_login(self):
    # Replace the following lines with the actual login logic using Selenium
    self.selenium.get(self.live_server_url) # Replace with your login page URL
    username_input = self.selenium.find_element_by_id('benz')
    password_input = self.selenium.find_element_by_id('Benz@123')

    # Fill in the login form
    username_input.send_keys('benz')
    password_input.send_keys('Benz@123')

    # Submit the form (replace with your submit button or form submission logic)
    password_input.submit()

    # Replace this with assertions that validate the successful login
    self.assertIn('Welcome', self.selenium.page_source)
```


Screenshot

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

(env) D:\new product db update\thangamproject>python manage.py test
Found 0 test(s).
System check identified no issues (0 silenced).

-----
Ran 0 tests in 0.000s

OK

```

Test Report 1

Test Case 1					
Project Name:					
Login Test Case					
Test Case ID: Test_1			Test Designed By: Edvin Sarju		
Test Priority(Low/Medium/High): High			Test Designed Date: 01/12/2023		
Module Name: login page			Test Executed By: Ms. Meera rose Mathew		
Test Title: login page			Test Execution Date:02/12/2023		
Description: To test login page					
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 Username	Username: benz	Admin should be able to Login	Admin Logged in and navigated to Admin Dashboard with records	Pass
3	Provide Valid Password	Password: Benz@123			

4	Click on Login button				
Post-Condition: Admin is validated with database and successfully login into account. The Account session details are logged in database					

Test Case 2:

Add to cart:

```

from django.test import TestCase
from django.contrib.auth import get_user_model
from .models import ShoppingCart, Product

class ShoppingCartTestCase(TestCase):
    def setUp(self):
        # Create a custom user for testing
        User = get_user_model()
        self.user = User.objects.create_user(username='Benz', password='Benz@123')

        # Create a product for testing with price and discount set
        self.product = Product.objects.create(
            product_name='Leaf Motif Stone Encrusted Gold Ring',
            price=100.0,
            discount=10.0,
            # Add other necessary fields for your Product model
        )

        # Log in the user
        self.client.login(username='benz', password='Benz@123')

    def test_total_price_calculation(self):
        # Create a ShoppingCart item for the user
        shopping_cart_item=ShoppingCart.objects.create(user=self.user,
            product=self.product, quantity=2)

        # Handle None values gracefully in the test
        expected_total_price=(shopping_cart_item.quantity
            (self.product.calculate_sale_price() if self.product else 0)
        )

        # Check if the total price is calculated correctly
        self.assertEqual(shopping_cart_item.calculate_total_price(), expected_total_price

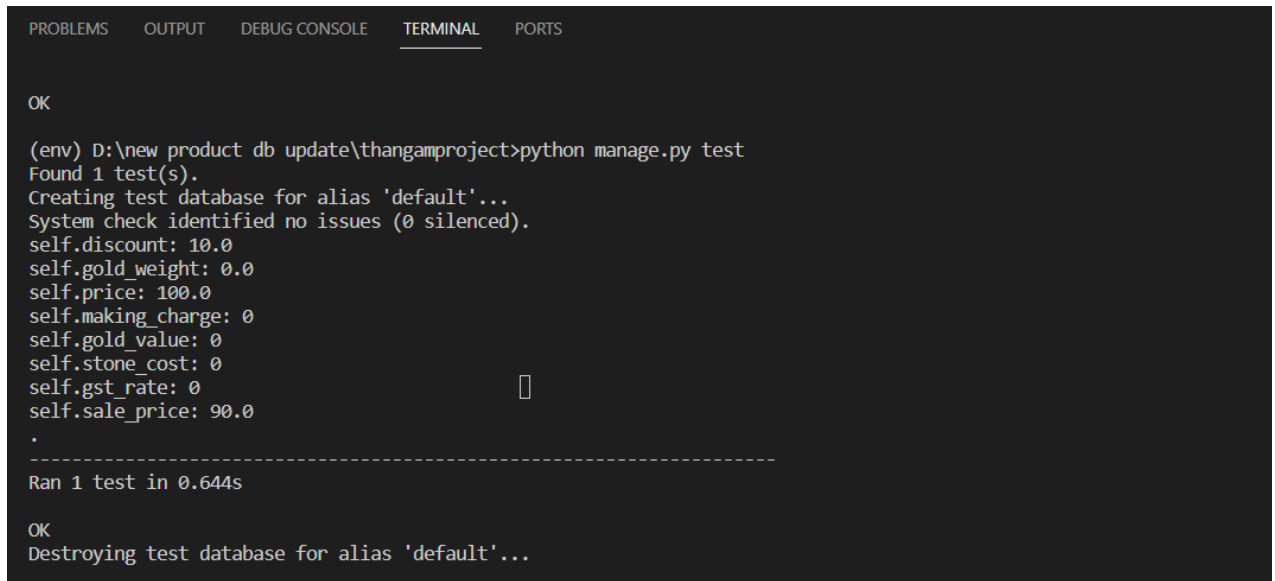
```

Test Report 2

Test Case 2					
Project Name: Thangam Jewellery					
Add to cart Test case					
Test Case ID: Test_2			Test Designed By: Edvin Sarju		
Test Priority(Low/Medium/High): High			Test Designed Date: 01/12/2023		
Module Name: Add to cart			Test Executed By: Ms. Meera rose Mathew		
Test Title: Add to Cart Functionality			Test Execution Date: 02/12/2023		
Description: To test the functionality of adding a product to the shopping cart					
Pre-Condition: User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Login to the Application	Username: benz, Password: Benz@123	User should be logged in successfully	User logged in successfully	Pass
2	Navigate to Product Details Page		Product details page should be displayed	Product details page displayed	Pass
3	Click on "Add to Cart" button		Product should be added to the shopping cart	Product added to the shopping cart successfully	Pass
4	View Shopping Cart		Shopping cart page should be displayed	Shopping cart page displayed	Pass
5	Verify Added Product in Shopping Cart		Added product should be displayed in the cart	Added product displayed in the shopping cart	Pass

Post-Condition:

User is logged in and there is a product available

Screenshot:

The screenshot shows a terminal window with the following content:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

OK

(env) D:\new product db update\thangamproject>python manage.py test
Found 1 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
self.discount: 10.0
self.gold_weight: 0.0
self.price: 100.0
self.making_charge: 0
self.gold_value: 0
self.stone_cost: 0
self.gst_rate: 0
self.sale_price: 90.0
.
-----
Ran 1 test in 0.644s

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

Test case 3:**Add product:**

```
from django.test import TestCase, Client
from django.urls import reverse
from .models import Product
```

```
class AddProductViewTest(TestCase):
```

```
    def setUp(self):
```

```
        # Initialize the client
```

```
        self.client = Client()
```

```
    def test_add_product_view(self):
```

```
        # Your test data
```

```
        form_data = {
```

```
            'product-name': 'Leaf Motif Stone Encrusted diamond',
```

```
            'category-name': 'diamond',
```

```
            'subcategory-name': 'rings',
```

```
            'quantity': '1',
```

```
            'description': 'Diamonds are highly coveted gemstones known for their  
exceptional brilliance and hardness, symbolizing enduring love and luxury.',
```

```
            'price': '40987.00',
```

```
            'discount': '1.00',
```

```
            'status': 'Active',
```

```
            'making-charge': '50.00',
```

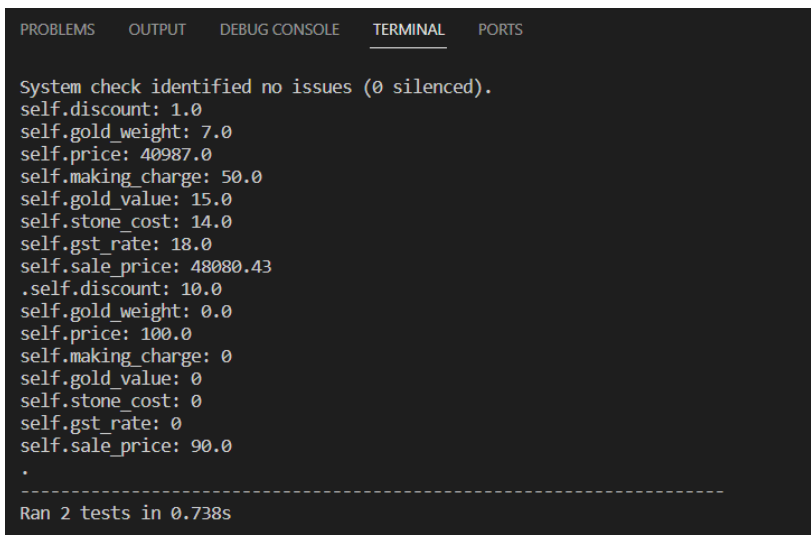
```
            'gold-value': '15.00',
```

```
'stone-cost': '14.00',
'gst-rate': '18.00',
'gold-weight': '7.00',
'purity-of-gold': '18K',
# Include any other form fields as needed
}
url = reverse('add_product')
# Make a POST request with the form data
response = self.client.post(url, form_data, format='multipart')
self.assertEqual(response.status_code, 302)
self.assertEqual(Product.objects.count(), 1)

# Retrieve the created product
created_product = Product.objects.first()
self.assertEqual(created_product.product_name, 'Leaf Motif Stone Encrusted
diamond')
self.assertEqual(created_product.category, 'diamond')
self.assertEqual(created_product.subcategory, 'rings')
# Add more attribute checks as needed

# Redirect to a success page or any other desired action
self.assertRedirects(response, reverse('adminpage'))
```

Screenshot



The screenshot shows a terminal window with a dark background and light text. At the top, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected), and 'PORTS'. The terminal output displays the results of a system check and a series of assertions for a product object. The system check reports no issues. The assertions verify attributes like discount, gold_weight, price, making_charge, gold_value, stone_cost, gst_rate, and sale_price for two different states of the product. The final line indicates that 2 tests were run successfully in 0.738 seconds.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

System check identified no issues (0 silenced).
self.discount: 1.0
self.gold_weight: 7.0
self.price: 40987.0
self.making_charge: 50.0
self.gold_value: 15.0
self.stone_cost: 14.0
self.gst_rate: 18.0
self.sale_price: 48080.43
self.discount: 10.0
self.gold_weight: 0.0
self.price: 100.0
self.making_charge: 0
self.gold_value: 0
self.stone_cost: 0
self.gst_rate: 0
self.sale_price: 90.0
.
-----
Ran 2 tests in 0.738s
```

Test Report

Test Case 3					
Project Name: Thangam Jewellery					
Add product Test Case					
Test Case ID: Test_3			Test Designed By: Edvin Sarju		
TestPriority(Low/Medium/High: High			Test Designed Date:01/12/2023		
Module Name:			Test Executed By: Ms. Meera rose Mathew		
Test Title: Add Product Functionality			Test Execution Date: 02/12/2023		
Description: To test the functionality of adding a new product to the system					
Pre-Condition: User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Login as Admin to the Application	Username: admin, Password: admin	Admin should be logged in successfully	Admin logged in successfully	Pass
2	Navigate to Add Product Page		Add product page should be displayed	Add product page displayed	Pass
4	Fill in Product Information	Product data including name, category, quantity, etc.	Product information should be filled in	Product information filled in as expected	Pass
5	Submit the Product Form		Product should be added to the system	Product added to the system successfully	Pass
6	Verify Product in Product List		Added product should be visible in the product list	Added product visible in the product list	Pass
Post-Condition Admin is logged in and the necessary fields are filled correctly					

Test case:**Customer list:**

```
from django.test import TestCase
from django.urls import reverse
from django.contrib.auth import get_user_model

class CustomerListViewTest(TestCase):
    def setUp(self):
        # Create a test user
        self.user = get_user_model().objects.create_user(
            username='admin',
            password='admin'
        )

        # ... rest of your test cases ...

    def test_customer_list_view(self):
        # Login the test user
        self.client.login(username='admin', password='admin')

        # Access the customer list view
        response = self.client.get(reverse('customer_list'))

        # Check if the view returns a 200 status code
        self.assertEqual(response.status_code, 200)

        # Check if the rendered template is correct
        self.assertTemplateUsed(response, 'loginview.html')

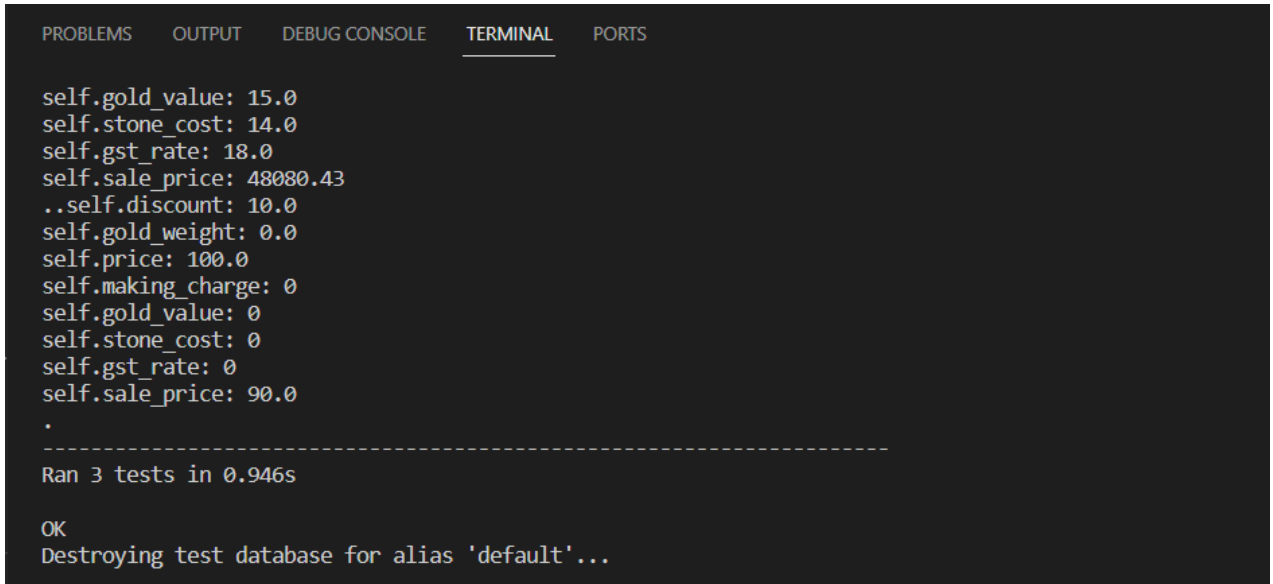
        # Check if the user's information is present in the rendered HTML
        self.assertContains(response, 'admin')
        # Update the email to match the one you expect
        self.assertContains(response, 'edvin@gmail.com')

    def test_customer_list_view_unauthenticated(self):
        # Logout the test user
        self.client.logout()

        # Access the customer list view without authentication
        response = self.client.get(reverse('customer_list'))

        # Check if the view redirects to the login page
```

```
self.assertEqual(response.status_code, 302) # Ensure it's a redirect
# Update the URL to match your project structure
expected_redirect_url = reverse('login') + f'?next={reverse("customer_list")}'
self.assertRedirects(response, expected_redirect_url)
```

Screenshot:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

self.gold_value: 15.0
self.stone_cost: 14.0
self.gst_rate: 18.0
self.sale_price: 48080.43
..self.discount: 10.0
self.gold_weight: 0.0
self.price: 100.0
self.making_charge: 0
self.gold_value: 0
self.stone_cost: 0
self.gst_rate: 0
self.sale_price: 90.0
.
-----
Ran 3 tests in 0.946s

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


Test report

Test Case 4					
Project Name: Thangam Jewellery					
Admin Customer Management Test Case					
Test Case ID: Test_4			Test Designed By: Edvin Sarju		
Test Priority(Low/Medium/High): High			Test Designed Date: 01/12/2023		
Module Name: Admin Customer Management			Test Executed By: Ms. Meera rose Mathew		
Test Title: Admin Customer Management Functionality			Test Execution Date: 02/12/2023		
Description: Admin can view the customer list					
Pre-Condition: User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Login as Admin to the Application	Username: admin, Password: admin	Admin should be logged in successfully	Admin logged in successfully	Pass
2	Navigate to Admin Customer List Page		Admin customer list page should be displayed	Admin customer list page displayed	Pass
3	Verify Admin Customer Manageme nt Options		Options for activating and deactivating customers should be visible	Activation and deactivation options visible for each customer	Pass
4	Activate a Customer Account		Customer account should be activated successfully	Customer account activated successfully	Pass
	Deactivate a Customer Account		Customer account should be deactivated successfully	Customer account deactivated successfully	Pass

Post-Condition: Admin is logged in and there are existing customer records

CHAPTER 6

IMPLEMENTATION

6.1 INTRODUCTION

Implementing a system is essential for ensuring its successful long-term use. The implementation stage involves creating the necessary infrastructure and processes to ensure that the system can be used to its fullest potential. This includes establishing the necessary hardware, software, and other resources that are needed to make the system run efficiently. It also involves training users on how to use the system, troubleshooting issues that arise, and monitoring performance over time.

During the implementation stage, the system is built according to the requirements and specifications outlined in the design phase. The development team writes the code, creates the database schema, and develops any necessary interfaces or integration points. The team also creates and executes test cases to ensure that the system works as intended. Once the system has been developed and tested, it is deployed into production. This involves installing the software on the appropriate hardware and configuring it to work with any necessary systems or interfaces. The implementation team also performs user acceptance testing to ensure that the system meets the needs of the end-users. After the system has been deployed, the implementation team provides training and support to end-users to ensure that they can use the system effectively. The team may also conduct post-implementation reviews to evaluate the success of the implementation and identify any areas for improvement.

6.2 IMPLEMENTATION PROCEDURES

Implementing a system can be a complicated process, but with the right steps and procedures in place it can become much easier. Having an understanding of the implementation procedure is essential in order to ensure that the system is installed correctly and that it operates efficiently. It is important to understand the different stages of the implementation procedure and how they interact with each other in order to achieve success in system implementation. This includes understanding the need for user training, testing, maintenance and support services. However, some general steps that can be followed for implementing a web-based project are: Planning: Create a project plan that outlines the scope, objectives, deliverables, and timeline for the project. Identify the resources and team members needed to complete the project successfully. Design: Based on the requirements, design the user interface, database schema, and system architecture. Develop a detailed design document that includes wireframes, flowcharts, and system diagrams. Development: Implement the design by writing the code, configuring the server, and integrating third-party libraries and APIs. Conduct unit testing and system testing at each stage of development. Deployment: Deploy the application to a production environment, such as a web server, and ensure that it is accessible to users. Test the application thoroughly in a real-world environment and address any issues that arise. Maintenance: Continuously monitor and maintain the application to ensure that it is running efficiently and meeting user needs. Address any bugs or issues that arise, and update the application as necessary to incorporate new features or functionality.

6.2.1 User Training

User training is designed to prepare the user for testing and converting the system. To achieve the objective and benefits expected from computer-based system, it is essential for the people who will be involved to be confident of their role in the new system. As system becomes more complex, the need for training is more important. By user training the user comes to know how to

enter data, respond to error messages, interrogate the database and call up routine that will produce reports and perform other necessary functions

6.2.2 Training on the Application Software

After providing the necessary basic training on computer awareness 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 data 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

6.2.3 System Maintenance

System maintenance is the stage in the software development life cycle (SDLC) where the system is updated and improved after it has been deployed into production. This involves fixing any issues that arise, adding new features, and adapting the system to changes in the environment or user needs. The objective of system maintenance is to ensure that the system continues to function effectively and meet the needs of its users over its lifetime.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The Thangam Jewellery Project is a comprehensive e-commerce platform designed to enhance the online shopping experience for users interested in purchasing exquisite jewellery. The user-centric features, such as detailed product information, high-quality images, filtering options, and a virtual try-on feature, contribute to a seamless and enjoyable browsing and purchasing process. The project not only caters to the needs and preferences of customers but also provides efficient management tools for administrators and vendors, ensuring smooth inventory and order management. The implementation of secure payment gateways, personalized user accounts, and a responsive customer support system adds an extra layer of reliability and trust to the platform. The inclusion of a virtual try-on feature using augmented reality technology is a notable innovation, allowing users to visualize how jewellery will look on them before making a purchase.

7.2 FUTURE SCOPE

Enhanced Virtual Try-On Features: Expand the virtual try-on feature by incorporating advanced augmented reality and machine learning technologies. This could include real-time customization options, suggesting complementary items, and refining the accuracy of how jewellery looks on users. **Integration of Blockchain for Purity Verification:** Consider integrating blockchain technology to provide an immutable record of the jewellery's purity, based on the Bureau of Indian Standards (BIS). This can enhance transparency and trust in the authenticity of the jewellery. **AI-Driven Recommendations:** Implement artificial intelligence algorithms to user preferences and behaviours, providing personalized recommendations for jewellery items. This can enhance the user experience and drive more sales. **Global Expansion:** Explore opportunities for expanding the platform globally, collaborating with international jewellery designers and vendors. This can attract a diverse customer base and offer a wider range of unique and culturally diverse jewellery. **Mobile Application Development:** Develop a mobile application for both Android and iOS platforms to reach a broader audience. Mobile apps provide a more convenient and accessible way for users to browse and make purchases on the go. **Enhanced Vendor Management:** Provide vendors with analytics tools to sales trends, customer preferences, and inventory management. This can help vendors optimize their offerings and improve their overall performance on the platform. **Social Media Integration:** Integrate social media functionalities to allow users to share their favourite jewellery items, reviews, and purchases on their social profiles. This can increase brand visibility and attract new customers through social networks. **Environmental Sustainability Initiatives:** Consider incorporating information about the sustainability practices of jewellery vendors, such as the use of ethically sourced materials or environmentally friendly manufacturing processes. This aligns with the growing trend of eco-conscious consumerism. **Continuous Platform Optimization:** Regularly update the platform with the latest technological advancements, security patches, and user interface improvements to stay competitive in the rapidly evolving e-commerce landscape.

CHAPTER 8

BIBLIOGRAPHY

REFERENCES:

E-commerce User Experience and Design: Book: "Don't Make Me Think" by Steve Krug. This book provides insights into web usability and user experience design. E-commerce Platform Management: Book: "E-commerce Operations and Management" by David Walters. This book covers various aspects of e-commerce operations, including platform management. Customer Relationship Management in E-commerce: Book: Customer Relationship Management: Concepts and Technologies" by Francis Buttle. This book explores CRM strategies that can be applied in an e-commerce context E-commerce Security: Book: "E-commerce 2019" by Kenneth C. Laudon and Carol Traver. This book may cover aspects of e-commerce security, including payment gateways.

WEBSITES:

- <https://www.kalyanjewellers.net/>
- <https://bhima.com/>
- <https://www.joyalukkas.in/>

CHAPTER 9

APPENDIX

9.1 Sample Code

Login page:

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>THANGAM</title>
  <meta content="width=device-width, initial-scale=1.0" name="viewport">
  <meta content="eCommerce HTML Template Free Download" name="keywords">
  <meta content="eCommerce HTML Template Free Download" name="description">

  <!-- Favicon -->
  <link href="{% static 'img/favicon.ico' %}" rel="icon">

  <!-- Google Fonts -->
  <link href="https://fonts.googleapis.com/css?family=Open+Sans:300,400|Source+Code+Pro:700,900&display=swap" rel="stylesheet">

  <!-- CSS Libraries -->
  <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" rel="stylesheet">
  <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.10.0/css/all.min.css" rel="stylesheet">
  <link href="{% static 'lib/slick/slick.css' %}" rel="stylesheet">
  <link href="{% static 'lib/slick/slick-theme.css' %}" rel="stylesheet">

  <!-- Template Stylesheet -->
  <link href="{% static 'css/style.css' %}" rel="stylesheet">
</head>
<!-- Nav Bar End -->
<style>
  /* Your existing CSS styles here */

  /* Additional styles for the header */
  .header {
    text-align: center;
    background-color: #fff; /* Change the background color as needed */
    padding: 20px 0;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
  }

  /* Additional styles for the footer */
  .footer {
    text-align: center;
    background-color: #fff; /* Change the background color as needed */
    padding: 10px 0;
    position: absolute;
    bottom: 0;
    width: 100%;
    box-shadow: 0 -2px 5px rgba(0, 0, 0, 0.1);
  }
  #login-section {
    background-color: white;
  }
```

```

padding: 20px;
border-radius: 10px;
box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
#login-section .row img {
max-width: 80%; /* Adjust the maximum width of the image */
height: 80%; /* Maintain the aspect ratio of the image */
border-radius: 10px; /* Add border-radius for a rounded appearance */
box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); /* Add a subtle box shadow */
}

#login-section .row .col-md-6 {
padding: 20px;
margin-top: 10px;
margin-bottom: 10px; /* Adjust the margin space below the form */
background-color: white;
}

#login-section table {
width: 100%; /* Make the table fill the container width */
border-collapse: collapse; /* Collapse border spacing */
margin-top: 10px; /* Adjust margin space above the table */
}

#login-section th, #login-section td {
border: 1px solid #ddd; /* Add border for table cells */
padding: 8px; /* Add padding for spacing inside cells */
text-align: left; /* Align text to the left */
}

#login-section th {
background-color: #f2f2f2; /* Add a background color for table header cells */
}

</style>
<body>
<!-- Nav Bar Start -->
<div class="nav">
<div class="container-fluid">
<nav class="navbar navbar-expand-md bg-dark navbar-dark">
<a href="#" class="navbar-brand"></a>
<button type="button" class="navbar-toggler" data-toggle="collapse" data-
target="#navbarCollapse">
<span class="navbar-toggler-icon"></span>
</button>

<div class="collapse navbar-collapse justify-content-between" id="navbarCollapse">
<div class="navbar-nav ml-auto"> <!-- Add ml-auto class to move other buttons to the right
-->
<a href="#" class="nav-item nav-link active"></a>
<div class="nav-item dropdown">

</div>

```

```

        </div>
    </div>
</div>
</nav>
</div>
</div>
<div class="header">
    <h1>Thangam Jewellery</h1>
</div>

<div id="login-section">
    <div class="row">
        <div class="col-md-6">
            
        </div>

        <div class="card-body">
            <form id="login" action="" method="POST">
                { % csrf_token % }
                <h3 class="text-center">Login</h3>
                <div class="form-group">
                    <label for="username">Username</label>
                    <input type="text" id="username" name="username" class="form-control" required>
                    <div class="error-message" id="username-error"></div>
                </div>
                <div class="form-group">
                    <label for="password">Password</label>
                    <input type="password" id="password" name="password" class="form-control" required>
                    <div class="error-message" id="password-error"></div>
                </div>
                &nbsp;
                <p>Forgot your password? <a href="{ % url 'password_reset' % }">Reset it here.</a></p>

                <div class="text-center">
                    <button type="submit" class="btn btn-primary">Login</button>
                </div>
            </form>
            <!-- Sign Up Link -->
            <div class="text-center mt-3">
                <p>Don't have an account <a href="{ % url 'register' % }">| Sign Up</a></p>
            </div>
        </div>
    </div>

</div>

<div class="footer">
    <p>&copy; 2023 Thangam Jewellery - All rights reserved</p>
    <p><a href="#">Privacy Policy</a> | <a href="#">Terms of Service</a></p>
</div>
<script>
    document.addEventListener("DOMContentLoaded", function () {
        const form = document.getElementById('login');
        const usernameInput = document.getElementById('username');
        const passwordInput = document.getElementById('password');
        const usernameError = document.getElementById('username-error');
        const passwordError = document.getElementById('password-error');

```

```
function validateUsername() {
  if (usernameInput.value.trim() === "") {
    usernameInput.classList.add('is-invalid');
    usernameError.textContent = 'Username is required';
    return false;
  } else {
    usernameInput.classList.remove('is-invalid');
    usernameError.textContent = "";
    return true;
  }
}

function validatePassword() {
  if (passwordInput.value.trim() === "") {
    passwordInput.classList.add('is-invalid');
    passwordError.textContent = 'Password is required';
    return false;
  } else {
    passwordInput.classList.remove('is-invalid');
    passwordError.textContent = "";
    return true;
  }
}

// Function to validate username in real-time
usernameInput.addEventListener('input', validateUsername);

// Function to validate password in real-time
passwordInput.addEventListener('input', validatePassword);

// Form submission validation
form.addEventListener('submit', function (e) {
  let valid = true;

  // Additional validation logic can be added here if needed
  if (!validateUsername()) {
    valid = false;
  }

  if (!validatePassword()) {
    valid = false;
  }

  if (!valid) {
    e.preventDefault();
  }
});
});
</script>

<script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.bundle.min.js"></script>
<script src="{% static 'lib/easing/easing.min.js' %}"></script>
<script src="{% static 'lib/slick/slick.min.js' %}"></script>
```

```

<!-- Template Javascript -->
<script src="{% static 'js/main.js' %}"></script>
</body>
</html>

```

View product page:

```

{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>WELCOME TO THANGAM JEWELS </title>
  <meta content="width=device-width, initial-scale=1.0" name="viewport">
  <meta content="eCommerce HTML Template Free Download" name="keywords">
  <meta content="eCommerce HTML Template Free Download" name="description">

  <!-- Favicon -->
  <link href="{% static 'img/favicon.ico' %}" rel="icon">

  <!-- Google Fonts -->
  <link href="{% static 'https://fonts.googleapis.com/css?family=Open+Sans:300,400|Source+Code+Pro:700,900&display=swap' %}" rel="stylesheet">

  <!-- CSS Libraries -->
  <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" rel="stylesheet">
  <link href="https://cdn.jsdelivr.cloudflare.com/ajax/libs/font-awesome/5.10.0/css/all.min.css" rel="stylesheet">
  <link href="{% static 'lib/slick/slick.css' %}" rel="stylesheet">
  <link href="{% static 'lib/slick/slick-theme.css' %}" rel="stylesheet">

  <!-- Template Stylesheet -->
  <link href="{% static 'css/style.css' %}" rel="stylesheet">
  <style>
    .left-corner-heading {
      position: absolute;
      left: 0; /* Align it to the left side of its container */
      top: 0; /* Align it to the top of its container */
    }

    .delete-button {
      background-color: #ff0000; /* Red color */
      color: #fff; /* Text color */
      border: none;
      padding: 5px 10px;
      border-radius: 5px;
      cursor: pointer;
    }

    .delete-button:hover {
      background-color: #cc0000; /* Darker red color on hover */
    }

    /* Add this style for the Logout button container */
    .logout-button-container {

```

```
position: absolute;
top: 0; /* Align it to the top of the header */
right: 20px; /* Align it to the right side of the header with 20px spacing */
color: #ccc;
}

/* Style for the Logout button */
.logout-button-container a {
  color: #666; /* Button text color */
  text-decoration: none;
  padding: 8px 16px; /* Adjust the padding for button size */
  border-radius: 4px;
}

.save-changes-button {
  background-color: #28a745; /* Green color */
  color: #fff; /* Text color */
  border: none;
  padding: 5px 10px;
  border-radius: 5px;
  cursor: pointer;
  margin-right: 5px; /* Adjust the margin for spacing */
}

.save-changes-button:hover {
  background-color: #218838; /* Darker green color on hover */
}

/* Style for the "Edit" link */
.edit-link {
  color: #007bff; /* Blue color */
  text-decoration: none;
  margin-right: 5px; /* Adjust the margin for spacing */
}

.edit-link:hover {
  text-decoration: underline; /* Underline the link on hover */
}

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

th {
  padding: 10px;
  text-align: left;
}

/* Apply styles to the table cells */
td {
  padding: 10px;
  text-align: left;
}
```



```
th:nth-child(1),
td:nth-child(1) {
    width: 5%;
}

th:nth-child(2),
td:nth-child(2) {
    width: 15%;
}

th:nth-child(3),
td:nth-child(3) {
    width: 10%;
}

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

th {
    background-color: #f2f2f2;
}

img {
    max-width: 100px;
    max-height: 100px;
}

.delete-button, .edit-button {
    padding: 5px 10px;
    background-color: #ff0000;
    color: #fff;
    text-decoration: none;
    border: none;
    cursor: pointer;
    font-size: 14px;
}

.delete-button:hover, .edit-button:hover {
    background-color: #cc0000;
}

/* Add this CSS for the sidebar */
.sidebar {
    width: 20%;
    background-color: #f0f0f0;
    padding: 20px;
    position: fixed;
    height: 100%;
    overflow: auto;
}
```

```
.sidebar-item {
    margin-bottom: 20px;
    padding: 10px;
    background-color: #ccc;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

.sidebar-item:hover {
    background-color: #ddd;
}

.account-settings {
    background-color: #f9f9f9;
    border: 1px solid #ddd;
    padding: 20px;
}

.account-box {
    background-color: #fff;
    border: 1px solid #ddd;
    padding: 10px;
    margin-top: 10px;
}

.content {
    margin: 20px 20px 20px 0; /* Adjust the margin as needed */
    padding: 20px;
    /* ... (other styles) ... */
}

/* Add this style to clear the float after the content and sidebar */
.clearfix:after {
    content: "";
    display: table;
    clear: both;
}

/* Adjust the width of the table container */
.table-container {
    width: 80%; /* Set the width based on your design */
    float: right;
}

.nav {
    width: 100%; /* Set your desired width here */
    /* ... (other styles) ... */
}

h4.view-product-heading {
    color: white;
}

</style>
</head>
```

```

<div class="nav">
  <div class="container-fluid">
    <nav class="navbar navbar-expand-md bg-dark navbar-dark">
      <a href="#" class="navbar-brand"></a>
      <button type="button" class="navbar-toggler" data-toggle="collapse" data-
target="#navbarCollapse">
        <span class="navbar-toggler-icon"></span>
      </button>
      <h4 class="view-product-heading">View Product</h4>

      <div class="collapse navbar-collapse justify-content-between" id="navbarCollapse">
        <div class="navbar-nav ml-auto">
          <a href="" class="nav-item nav-link" style="color: white;"></a>
          { % comment % } <div class="hero_area">
            <header class="header_section">
              <!-- Your header content goes here -->
              <div class="logout-button-container"> { % endcomment % }
                <a href="{ % url 'logout' % }" class="btn btn-primary">Logout</a>
              </div>
            </header>
          </div>
        </div>
      </div>
    </nav>
  </div>
</div>

<div class="clearfix">
  <!-- Sidebar -->
  <div class="sidebar">

    <div class="sidebar-item">
      ORDERS
      <div class="account-box">
        <div class="content-item" href="{ % url 'viewproduct' % }">STAFF</div>
      </div>
    </div>
    <div class="sidebar-item">
      PRODUCT
      <div class="account-box">
        <div class="content-item" onclick="loadAddProductForm()">ADD PRODUCT</div>
      </div>

      <div class="account-box">
        <a class="content-item" href="{ % url 'viewproduct' % }">VIEW PRODUCT</a>
      </div>
    </div>
    <div class="sidebar-item">USERS LISTING</div>
    <div class="account-box">
      <a class="content-item" href="{ % url 'staff_list' % }">STAFF</a>
    </div>

    <div class="account-box">
      <a class="content-item" href="{ % url 'customer_list' % }">CUSTOMERS</a>
    </div>
  </div>

```

```

</div>

</div>

<!-- Content -->
<div class="content">
  <!-- Table Container -->
  <div class="table-container">
    <table class="table">
      <thead>
        <tr>
          <th>No.</th>
          <th>Product Name</th>
          <th>Category</th>
          <th>Subcategory</th>
          <th>Purity</th>
          <th>Quantity</th>
          <th>Description</th>
          <th>Price</th>
          <th>Discount (%)</th>
          <th>Making Charge</th>
          <th>Gold Value</th>
          <th>Gold Weight</th>
          <th>Stone Cost</th>
          <th>GST</th>
          <th>Total Price</th>
          <th>Status</th>
          <th>Product Image</th>
          <th>Edit</th>
        </tr>
      </thead>
      <tbody>
        { % for product in products % }
        <tr>
          <td>{{ forloop.counter }}</td>
          <td>{{ product.product_name }}</td>
          <td>{{ product.category }}</td>
          <td>{{ product.subcategory }}</td>
          <td>{{ product.purity_of_gold }}</td>
          <td>{{ product.quantity }}</td>
          <td>{{ product.description }}</td>
          <td>₹ {{ product.price }}</td>
          <td>{{ product.discount }}%</td>
          <td>₹ {{ product.making_charge }}</td>
          <td>₹ {{ product.gold_value }}</td>
          <td>{{ product.gold_weight }} grams</td>
          <td>₹ {{ product.stone_cost }}</td>
          <td>₹ {{ product.gst_rate }}</td>
          <td>₹ {{ product.sale_price }}</td>
          <td>{{ product.status }}</td>
          <td>
            { % if product.product_image % }
            
            { % else % }

```

```

                <p>No image available</p>
            {% endif %}
        </td>
        <td>
            <a href="{% url 'edit_product' product_id=product.product_id %}" class="edit-
button">Edit</a>
        </td>
        <td>

        </td>
    </tr>
{% endfor %}
</tbody>
</table>
</div>
</div>
</div>

<!-- ... (remaining HTML code) ... -->

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

```

Product Details page:

```

{% load static %}
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>WELCOME TO THANGAM JEWELS </title>
        <meta content="width=device-width, initial-scale=1.0" name="viewport">
        <meta content="eCommerce HTML Template Free Download" name="keywords">
        <meta content="eCommerce HTML Template Free Download" name="description">

        <!-- Favicon -->
        <link href="{% static 'img/favicon.ico' %}" rel="icon">

        <!-- Google Fonts -->
        <link href="{% static 'https://fonts.googleapis.com/css?family=Open+Sans:300,400|Source+Code+Pro:700,900&display=swap' %}" rel="stylesheet">

        <!-- CSS Libraries -->
        <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
rel="stylesheet">
        <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.10.0/css/all.min.css"
rel="stylesheet">
        <link href="{% static 'lib/slick/slick.css' %}" rel="stylesheet">
        <link href="{% static 'lib/slick/slick-theme.css' %}" rel="stylesheet">

        <!-- Template Stylesheet -->
        <link href="{% static 'css/style.css' %}" rel="stylesheet">
    </head>
    <style>

```

```
.product-details {
  display: flex;
  justify-content: space-between;
  align-items: flex-start;
  text-align: left;
  margin: 20px 0; /* Adjust the top and bottom margin */
}

.product-image {
  flex-basis: 50%;
  margin-right: 20px; /* Add margin between image and other fields */
}

.product-image img {
  max-width: 60%;
}

.product-details-container,
.additional-fields {
  flex-basis: 50%;
  margin-top: 0; /* Remove top margin */
  padding-top: 0; /* Remove top padding */
}

.product-table-actions {
  display: flex;
  flex-direction: row; /* Change to row to align buttons horizontally */
  justify-content: flex-start; /* Align items to the start of the container */
  margin-top: 10px; /* Add margin as needed */
}

.buy-now-btn {
  display: inline-block;
  padding: 10px 20px;
  background-color: #FF6F61;
  color: #ffffff;
  text-decoration: none;
  border: 2px solid #FF6F61;
  border-radius: 5px;
  transition: background-color 0.3s ease, color 0.3s ease;
  margin-right: 10px; /* Add margin between buttons */
}

.buy-now-btn:hover {
  background-color: #ffffff;
  color: #FF6F61;
}

.additional-fields {
  background-color: white;
  padding: 20px; /* Adjust padding as needed */
  border-radius: 5px; /* Add border radius for a rounded appearance */
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1); /* Add a subtle shadow for depth */
}
```

```

</style>
<body>
  <div class="nav">
    <div class="container-fluid">
      <nav class="navbar navbar-expand-md bg-dark navbar-dark">
        <a href="#" class="navbar-brand"></a>
        <button type="button" class="navbar-toggler" data-toggle="collapse" data-
target="#navbarCollapse">
          <span class="navbar-toggler-icon"></span>
        </button>

        <div class="collapse navbar-collapse justify-content-between" id="navbarCollapse">
          <div class="navbar-nav ml-auto">
            <div class="collapse navbar-collapse justify-content-between" id="navbarCollapse">
              <div class="navbar-nav ml-auto">
                <!-- Add this block to display the username if the user is authenticated -->
                {% if user.is_authenticated %}
                <a href="#" class="nav-item nav-link" style="color: white;">{{ user.username }}</a>
                <a href="{% url 'logout' %}" class="nav-item nav-link" style="color: white;">Logout</a>
                {% else %}
                <a href="{% url 'login' %}" class="nav-item nav-link" style="color: white;">Login</a>
                {% endif %}
                <a href="{% url 'contact' %}" class="nav-item nav-link active">Contact Us</a>
                <a href="{% url 'view_cart' product_id=2 %}" class="nav-item nav-link
active">Cart</a>
                <div class="nav-item dropdown">
                  <a href="#" class="nav-link dropdown-toggle" data-toggle="dropdown">User
account</a>

                  <div class="dropdown-menu">
                    <a href="{% url 'register' %}" class="nav-item nav-link">REGISTER</a>
                  </div>
                </div>
              </div>
            </div>
          </div>
        </div>
      </nav>
    </div>
  </div>
  <br><br>
  <br><br>
  <div class="product-details" style="background-color: white; padding: 20px; border-radius: 5px; box-
shadow: 0 0 10px rgba(0, 0, 0, 0.1);">
    <div class="product-image">
      <a href="{% url 'product_details' product_id=product.product_id %}">
        
      </a>
    </div>
    <div class="product-details-container">
      <h2>{{ product.product_name }}</h2>

      <div class="product-details-container">

```

```

        <p><b>Purity of Gold:</b> {{ product.purity_of_gold }}</p>
        <p><b>Description:</b> {{ product.description }}</p>
        <p><b>Discount:</b> {{ product.discount }}%</p>
        <p><b>Original Price:</b>₹{{ product.price }}</p>
    </div>

    <div class="additional-fields">

        <p><b>Making Charge:</b>₹{{ product.making_charge }}</p>
        <p><b>Gold Value:</b>₹{{ product.gold_value }}</p>
        <p><b>Gold weight:</b>{{ product.gold_weight |floatformat:"0" }} Grams</p>
        <p><b>Stone Cost:</b>₹{{ product.stone_cost }}</p>
        <p><b>GST:</b>{{ product.gst_rate|floatformat:"0" }}%</p>
        <p><b>Total Price:</b>₹{{ product.sale_price }}</p>
    </div>

    <div class="product-table-actions">
        <a href="#" class="buy-now-btn">Buy Now</a>
        <form method="post" action="{% url 'add_to_cart' product_id=product.product_id %}">
            {% csrf_token %}
            <button type="submit" class="buy-now-btn">Add to Cart</button>
        </form>
    </div>
</div>
</div>
</div>
</div>
<br>
<br>
<br>
<div class="footer">
    <div class="container-fluid">
        <div class="row">
            <div class="col-lg-3 col-md-6">
                <div class="footer-widget">
                    <h2>Get in Touch</h2>
                    <div class="contact-info">
                        <p><i class="fa fa-map-marker"></i>vellore,Tamil nadu,india</p>
                        <p><i class="fa fa-envelope"></i>edvinsarju@gmail.com</p>
                        <p><i class="fa fa-phone"></i>8923567845</p>
                    </div>
                </div>
            </div>
            <div class="col-lg-3 col-md-6">
                <div class="footer-widget">
                    <h2>Follow Us</h2>
                    <div class="contact-info">
                        <div class="social">
                            <a href=""><i class="fab fa-facebook-f"></i></a>
                            <a href=""><i class="fab fa-linkedin-in"></i></a>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>

```



```

<div class="col-lg-3 col-md-6">
  <div class="footer-widget">
    <h2>Company Info</h2>
    <ul>
      <li><a href="#">About Us</a></li>
      <li><a href="#">Privacy Policy</a></li>
      <li><a href="#">Terms & Condition</a></li>
    </ul>
  </div>
</div>

<div class="col-lg-3 col-md-6">
  <div class="footer-widget">
    <h2>Purchase Info</h2>
    <ul>
      <li><a href="#">Pyament Policy</a></li>
      <li><a href="#">Shipping Policy</a></li>
    </ul>
  </div>
</div>
</div>

<div class="row payment align-items-center">
  <div class="col-md-6">
    <div class="payment-method">
      <h2>We Accept:</h2>
      
    </div>
  </div>
</div>
</div>
</div>
</div>
<!-- Footer End -->

<!-- Footer Bottom Start -->
<div class="footer-bottom">
  <div class="container">
    <div class="row">
      <div class="col-md-6 copyright">
        <p>Copyright &copy; <a href="#">THANGAM JEWELLERY</a>. All Rights
Reserved</p>
      </div>


      <div class="col-md-6 template-by">
      </div>
    </div>
  </div>
</div>

</body>
</html>

```

9.1 Screen Shots

Login page:



Thangam Jewellery

Login

Username

Password

Forgot your password? [Reset it here.](#)





[Login](#)

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

© 2023 Thangam Jewellery - All rights reserved

[Privacy Policy](#) | [Terms of Service](#)


View product:

ORDERS	escription	Price	Discount (%)	Making Charge	Gold Value	Gold Weight	Stone Cost	GST	Total Price	Status	Product Image	Edit
STAFF	old is a precious metal prized r its lustrous beauty, rarity, and during value, serving as a 'mbol of wealth and a store of onomic value throughout story.	₹55903.00	1.00%	₹100.00	₹15.00	5.00 grams	₹100.00	₹18.00	₹65630.38	active		Edit
PRODUCT	iamonds are highly coveted mstones known for their ceptional brilliance and rrdness, symbolizing enduring ve and luxury.	₹40987.00	1.00%	₹50.00	₹15.00	7.00 grams	₹14.00	₹18.00	₹48080.43	active		Edit
ADD PRODUCT	old is a precious metal prized r its lustrous beauty, rarity, and during value, serving as a 'mbol of wealth and a store of onomic value throughout story.	₹56758.00	1.00%	₹15.00	₹25.00	0.00 grams	₹14.00	₹18.00	₹66368.42	active		Edit
VIEW PRODUCT	xdfcgvhbjnkmjiuytrcextcfygvhbu	₹50000.00	0.00%	₹0.00	₹1.00	5.00 grams	₹0.00	₹18.00	₹59005.90	active		Edit
USERS LISTING												
STAFF												
CUSTOMERS												

Product Details page:

[BENZ](#)
[LOGOUT](#)
[CONTACT US](#)
[CART](#)
[USER ACCOUNT](#)

Thangam Jewellery



Leaf Motif Stone Encrusted Gold Ring

Purity of Gold: 22k

Description: Gold is a precious metal prized for its lustrous beauty, rarity, and enduring value, serving as a symbol of wealth and a store of economic value throughout history.

Discount: 1.00%

Original Price: ₹55903.00

Making Charge: ₹100.00

Gold Value: ₹15.00

Gold weight: 5Grams

Stone Cost: ₹100.00

GST: 18%

Total Price: ₹65630.38

[Buy Now](#)
[Add To Cart](#)

Contact page:

[EDVINSARJU](#)
[WISHLIST](#)
[CART](#)

Our Office

vellore,Tamil nadu,india
 edvinsarju@gmail.com
 +8923567845

Our Store

Manali,Chennai
 edvin778940@gmail.com
 8935784569

[Send Message](#)

