**Luxor university**

**Faculty of Computer and Information**

**Computer Science Department**

**Project Name**

**Graduation Project**

**Under Supervision**

Dr Bassem

**Submitted by**

**Zyad Ahmed Nour**

**Mahmoud Alaa eldin**

**Abdallah Mohamed**

**Eman Alaa**

**Doha Fawzy**

**Esraa Rabee**

**2023**

**ABSTRACT**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
|  | **Content** | **Page** |
| Abstract | | II |
| Table of Content   |  |  | | --- | --- | | List of abbreviation | V | | | III  v |
| List of tables | | ix |
| List of Figures | | x |

[**Chapter 1 System Overview** 1](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123970747)

[1.1 Introduction 2](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123912870)

[1.1.1 Motivation 2](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123912874)

[1.1.2 Problem Statement 2](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123912874)

[1.1.3 Overview 3](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123912874)

[**Chapter 2 Related Work** 4](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123970750)

[**Chapter 3 Domain Analysis and Technique** 12](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123970754)

[3.1 Domain Analysis 13](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123912870)

[3.1.1 Clients and users 15](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123912874)

[3.1.2 The Environment 15](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123912874)

[3.1.3 Tasks and actions currently being performed 15](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123912874)

[3.2 Risk 16](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123912865)

[3.3 Constrain 17](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123912865)

[3.4 Project Plan 18](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123912865)

[3.5 Feasibility Study 19](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123912865)

[3.6 Quality Assurance Plan  20](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123912865)

[3.8 System Requirements 24](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123912865)

[3.9 Techniques and tools 25](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123912866)

[3.10 Application Components 26](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123912865)

[3.11 System Request 28](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123912865)

[**Chapter 4 Proposed System and Methodology** 29](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123970757)

[4.1 System Use-Cases 30](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123970759)

[4.2 Use Case Description (Use case scenario) 33](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123970760)

[4.3 System Architecture 60](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123970761)

[4.4 Analysis Class 61](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123970762)

[4.4.1 State Diagram 61](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123970763)

4.4.2 Data flow diagram (Level Zero diagram) 63

4.4.3: Context Diagram 64

[4.5 interaction class diagram 65](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123970764)

[4.5.1 System diagram 65](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123970765)

4.5.2 Sequence Diagram 68

[4.6 Design Class 69](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123970766)

[4.6.1 Class Diagram 69](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123970767)

[4.7 Database Schema 70](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123970768)

[4.8 ER Diagram 71](file:///C:\Users\Office\AppData\Local\Microsoft\Windows\INetCache\IE\4JHVNJ3H\Project_Template_Part_I%5b1%5d.docx#_Toc123970769)

4.9 Design Mockup

**Chapter 5: Conclusion& Future work** 103

7.1 Conclusion 104

7.2. Future work 104

7.3. References 105

**List of Abbreviation**

|  |  |
| --- | --- |
| **Keyword** | **Meaning** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**List of Tables**

|  |  |  |
| --- | --- | --- |
|  | **Table** | **Page** |

**List of Figures**

|  |  |  |
| --- | --- | --- |
|  | **Figure** | **Page** |

**Chapter 1**

**System Overview**

# Introduction

### ****Problem Statement****

In the modern retail landscape, there is a growing demand for unique, high-quality handmade goods. However, many consumers struggle to find reliable sources for authentic, one-of-a-kind handcrafted items that reflect the craftsmanship and cultural significance of the artisans who create them. On the other hand, many skilled artisans and craftsmen face challenges in reaching a larger number of suitable customers who truly value their work.

.

The key challenges include:

1. **Lack of Connection Between Craftsmen and Consumers**: Despite the increasing demand for handmade products, almost there are not many centralized platforms where consumers seeking unique, artisanal items can easily connect with the artisans who create them. Consumers may spend considerable time searching online or in local markets, only to find limited options.
2. **Visibility and Market Access for Artisans**: Many artisans struggle with limited exposure and marketing resources, hindering their ability to showcase their work to a broader, global audience. As a result, these talented creators are often confined to local markets, missing opportunities to expand their reach and grow their businesses.
3. **Difficulty in Discovering Unique Products**: Consumers who value authenticity and craftsmanship often find it challenging to locate high-quality, handmade items that align with their tastes. The lack of a dedicated platform for showcasing unique crafts leads to missed connections between artisans and buyers who would otherwise be interested in these products.
4. **Fragmented Marketplaces**: Online shopping platforms that focus on handmade goods often fail to provide a seamless, user-friendly experience for both buyers and sellers. The fragmentation of the marketplace leads to confusion, and consumers may not know where to find trustworthy artisans or reliable products.
   * Examples of fragmented marketplaces in the context of handmade goods include.
5. **Social Media Channels** (e.g., **Facebook**, **Instagram**): Many artisans and small businesses in Egypt use social media to sell their products. While these platforms offer visibility, they are not dedicated marketplaces, leading to scattered buyer-seller connections and difficulty in discovering reliable vendors.
6. **Souq.com/Daraz (now Amazon Egypt):** While these platforms host a variety of products, they are not specialized in handmade or artisanal goods. As a result, artisans struggle to stand out, and consumers searching for unique crafts may find limited options or face challenges locating high-quality handmade items.
7. Immigrant pages or groups: Many emerging websites or Facebook groups cater to specific types of crafts, such as jewelry or pottery. However, these are often fragmented, lack effective search features, and do not provide a unified platform to showcase diverse materials.
8. Open Marketplaces Like OLX Egypt: OLX is used to manufacture various products, including products that are not handcrafted by artisans. It operates on sections or filters dedicated to handcrafted work, making it difficult for consumers to find products that are made or of high quality.

This scattered setup makes it challenging for artisans to reach a broad audience and for consumers to find a variety of unique products in one convenient location.

1. **Trust and Verification Issues**: Consumers may be hesitant to purchase handmade products online due to concerns about the authenticity of the items, the credibility of the sellers, and the overall quality of the goods.

### ****Solution Overview****

This online shopping platform seeks to solve these issues by creating a dedicated space for consumers seeking unique, handcrafted products and artisans looking to reach a wider audience. The platform will address the following key needs:

* **Bringing Together Craftsmen and Consumers**: By offering a centralized marketplace, the platform will make it easier for consumers to discover and connect with artisans who create handmade goods, ranging from jewelry and clothing to home decor and artwork.
* **Increased Visibility for Artisans**: Artisans will have the opportunity to showcase their products to a global audience, significantly increasing their visibility and chances of connecting with buyers who appreciate handmade, unique items.
* **Streamlined User Experience**: The platform will provide an easy-to-navigate, seamless browsing and shopping experience, allowing consumers to find products based on categories such as materials, type of craft, cultural significance, or price range.
* **Trust and Transparency**: With features like detailed product descriptions, customer reviews, and secure payment options, the platform will ensure that both consumers and artisans can engage with confidence and trust.
* **Support for Artisans’ Growth**: The platform will offer tools to help artisans manage their products, track sales, and receive customer feedback, allowing them to improve their offerings and grow their businesses effectively.

### ****Conclusion****

This online shopping website aims to create a thriving community where artisans and consumers can easily connect, share, and discover unique handmade products. By addressing the challenges of visibility, trust, and ease of access, the platform will not only empower artisans to expand their reach but also provide consumers with a rich, diverse marketplace for finding authentic, one-of-a-kind handmade goods. The ultimate goal is to foster a more sustainable and meaningful connection between creators and buyers, celebrating the beauty and craftsmanship of handmade goods in a digital marketplace.

## Scope

our e-commerce website project involves creating a complete online store with a wide range of products in different categories. This project will include features like allowing users to sign up, managing the list of products, letting customers add items to a cart, ensuring safe payment, and giving users a personalized shopping experience and more. The goal is to make a website that works well on different devices, keeps customer information safe, and is easy for people to use. It will also follow rules about protecting customer data and will offer many useful shopping features.

As part of the scope, the website is currently focused on operating within Egypt. It will cater to local artisans and customers in addition to tourists, with future plans to expand its reach to international markets in subsequent phases.

### ****Target Audience****

**Artisans and Craftsmen**: Individuals or small businesses producing handmade goods, including

* + **Demographics:**
    - Age: 18-65+ (a wide range, as craftspeople can be of various ages)
    - Gender: Both male and female
    - Location: around Egypt
  + **Psychographics:**

**Values:**

* **Quality and Craftsmanship:** Appreciates high-quality, handmade products.
* **Authenticity:** Prefers genuine, original pieces over mass-produced items.
* **Sustainability:** Values eco-friendly practices and sustainable materials.
* **Unique and Personalized:** Seeks one-of-a-kind items that reflect their individuality.
* **Supporting Local Businesses:** Enjoys contributing to local economies and supporting small businesses.

**Interests:**

* **Arts and Crafts:** Enjoys exploring different art forms and creative expressions.
* **Home Decor:** Interested in unique and stylish home decor items.
* **Fashion and Accessories:** Seeks unique and personalized fashion accessories.
* **Gifts:** Loves giving thoughtful and personalized gifts.

**Behaviors:**

* + - Online presence: Active on social media or other online platforms to showcase their work
    - Business mindset: Open to selling their products online and reaching a wider audience
* **Consumers:** People interested in unique, high-quality handmade products who value craftsmanship, culture, and sustainability.
  + **Demographics:** 
    - Age: 18-65+
    - Gender: Both male and female
    - Location: around Egypt
  + **Psychographics: (interest, personality& values)**
    - **Interests:** 
      * Art, design, unique items, sustainability, supporting local businesses
    - **Values:** 
      * Quality, authenticity, originality, ethical consumption
    - **Behaviors:** 
      * **Online shopping behavior:** Active online shoppers, particularly on e-commerce platforms

### Operating Environment

Web Application running on the browser with a responsive design suitable for desktop view and mobile view and a iOS Mobile version

## Core functions

#### For customer

* **Register:**
* **Login /Logout**
* **View** personal and artesian accounts
* **Edit** personal Account Details ()
* **Search** / **filters** to refine search results
* **View item**
* **Add**/**remove** item to cart
* **Add**/**remove** item to wish\_list
* **View shopping cart**
* **Payment transactions (pay/money\_back)**
* **Track order**
* Return item
* **Rate item**
* **Review item**
* **direct communication** with sellers
* Use Chabotto resolve issues
* **View receipts**
* **Choose preferred language**
* **Sharing through social media**
* **Choose** **shipping options** based on their preferences. (delivering on shop/ getting it delivered)

#### Delivering company

* Receive order details
* Provide **shipping updates**

#### For seller

* Register / **Create detailed artisan profiles**
* Login /Logout
* View /Edit Account Details
* Search item
* View item
* Manage **inventory** via a **dashboard**. (add, edit, remove)
* Delivery & payment
* Track order
* Accept/Cancel order
* Return item
* View orders and returns
* view rates
* View sales detailes
* **direct communication** with sellers
* Offer promotions or discounts.
* Receive **stock alerts**.
* Offer **on-demand products**
* **Receive/send payments via secure gateways**
* **Set shipping preferences**
* **collaborate** with delivery services for logistics.

#### Admin

* Login/Log out
* Manage **review system** and ensure legitimacy of ratings and feedback.
* Oversee the **money-back guarantee** process to verify claims and initiate refunds.
* Approve new products and verify **inventory** status
  1. Oversee collaboration with partners (e.g., seller,hotels, tourism office).
* Control discount codes and special offers
* Monitor inventory levels across the platform.
* Approve new products and verify **inventory** status.
* Offer **incentives** and support **training programs** for artisans.

#### System

* Display **product reviews and ratings**.
* Support **payment gateways** (e.g., Stripe, PayPal).
* Provide automated notifications about **pays**/**money-back guarantee** policies.
* Integrate **media uploads** for product listings.
* Integrate with **shipping API** for tracking and updates for Provide **pickup point details** for Tourists
* **switch language** based on the user’s preferences
* Offer a multilingual **customer support** system (either chatbots or human agents)
* Implement **search filters** for product categories and features
* Display **special offers** and **discount campaigns**.
* Integrate **social media sharing** options.
* Update **inventory levels** in real-time.
* Display **product specifications** and enable **Q&A sections**
* Provide a **user-friendly interface**.
* View **stock availability**
* Display **best-rated products** and **product comparison features and person recommendations**



## Definitions and acronyms:

Key terms and abbreviations used in the documentation.

* + 1. **Motivation**

‎**1.1.2 Problem Statement**

**1.1.3 Overview**

**Chapter 2**

**Related Work**

# Related works:

E-commerce platforms for handmade crafts and artisanal products have emerged as powerful tools to bridge the gap between local artisans and global markets. These platforms provide a space where small-scale producers can showcase their unique products, allowing them to reach a broader audience, gain economic independence, and preserve traditional craftsmanship. The development of such platforms involves integrating multiple technologies and considering various factors such as user experience, security, and scalability. Here we review the current state of related work in the field of e-commerce platforms for handcrafts, exploring existing systems, frameworks, and research contributions in the domain.

Society6 **:** A platform for independent artists to sell their designs on various products.

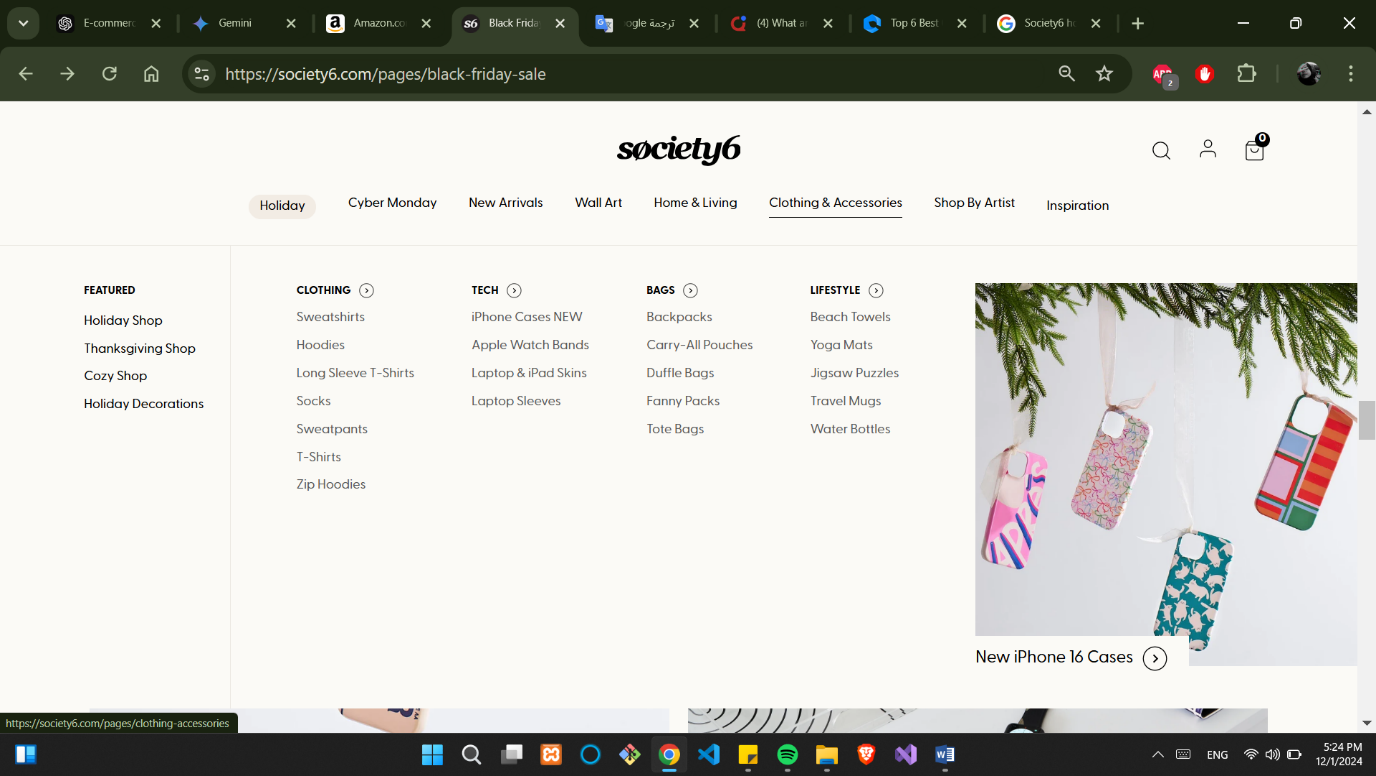
https://society6.com/

· Key **Strengths:**

* Wide range of product options, including home decor, apparel, and accessories.
* Print-on-demand model, reducing inventory costs for artists.
* Strong focus on design and aesthetics.

· Potential **Weaknesses:**

* Less emphasis on traditional crafts and handmade items.
* Competition from other print-on-demand platform



## 1.1 **Etsy**

Etsy is one of the most prominent e-commerce platforms that specialize in handmade, vintage, and unique items. Since its launch in 2005, Etsy has empowered artisans worldwide to sell their handcrafted goods to a global market. The platform has been instrumental in connecting buyers with small-scale creators, making it an important case study for understanding the success and challenges of online marketplaces for handmade products. Etsy’s business model focuses on offering sellers a user-friendly platform to list their products, manage payments, and engage with customers, while ensuring buyers have a secure, trusted shopping experience

### <https://www.etsy.com/>

### Key Strengths:

* **User-friendly interface** for buyers and sellers.
* Strong community and social features.
* Diverse product range

### Potential Weaknesses:

* **High competition**, especially for popular categories.
* **Seller fees can be significant**: Etsy charges merchants 7% of sales plus a $0.30 per item processing fee. This can add up quickly and given the high-volume nature of many shops, it can be difficult for smaller businesses to keep up with expenses."
* **shipping costs** on Etsy can also be expensive compared to other online platforms like Amazon or eBay.
* since most items sold on Etsy are handmade, there is often a **delay in receiving** products due not only to production time but also delivery time from overseas suppliers.

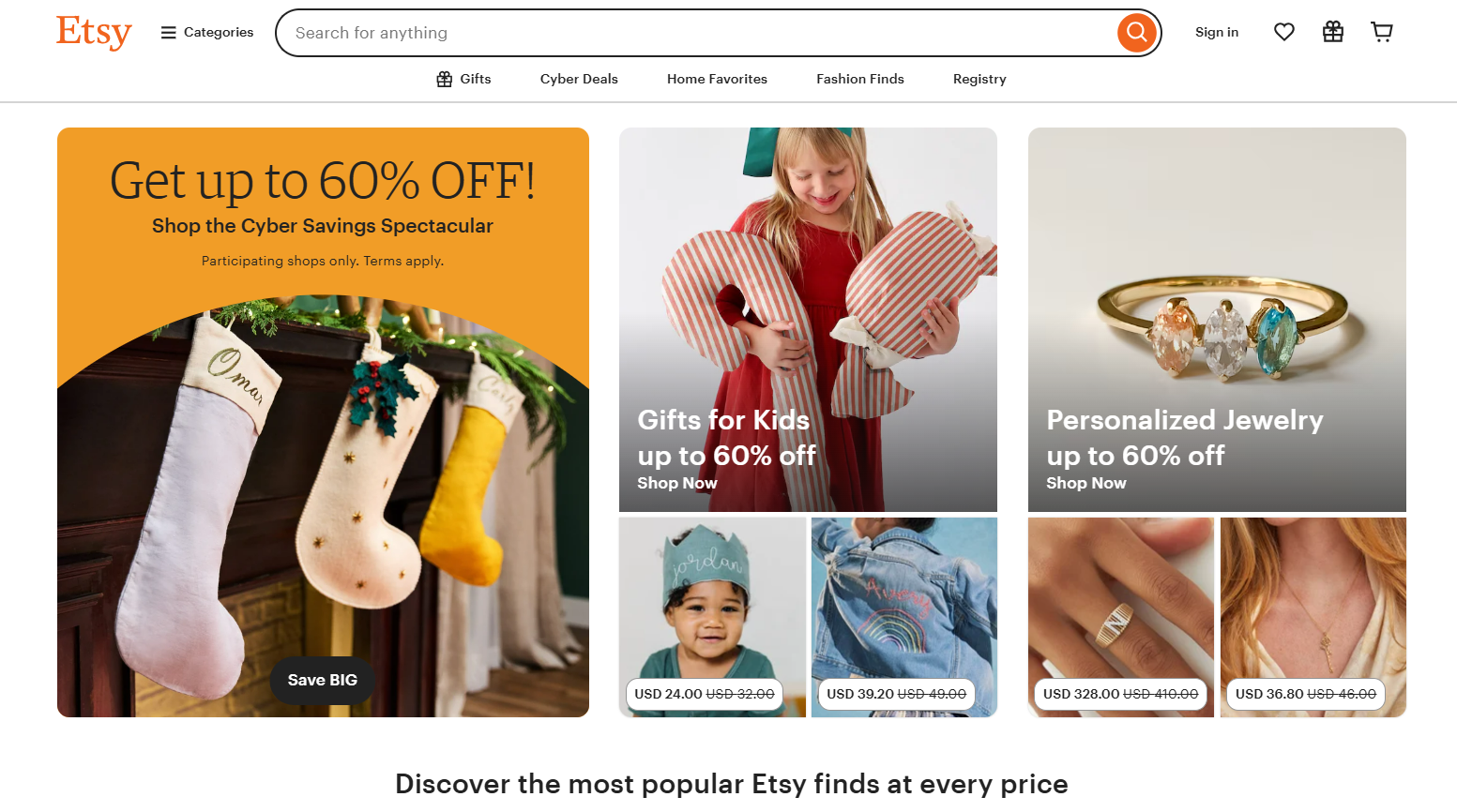


Figure Etsy platform

#### 1.2 **ArtFire**:

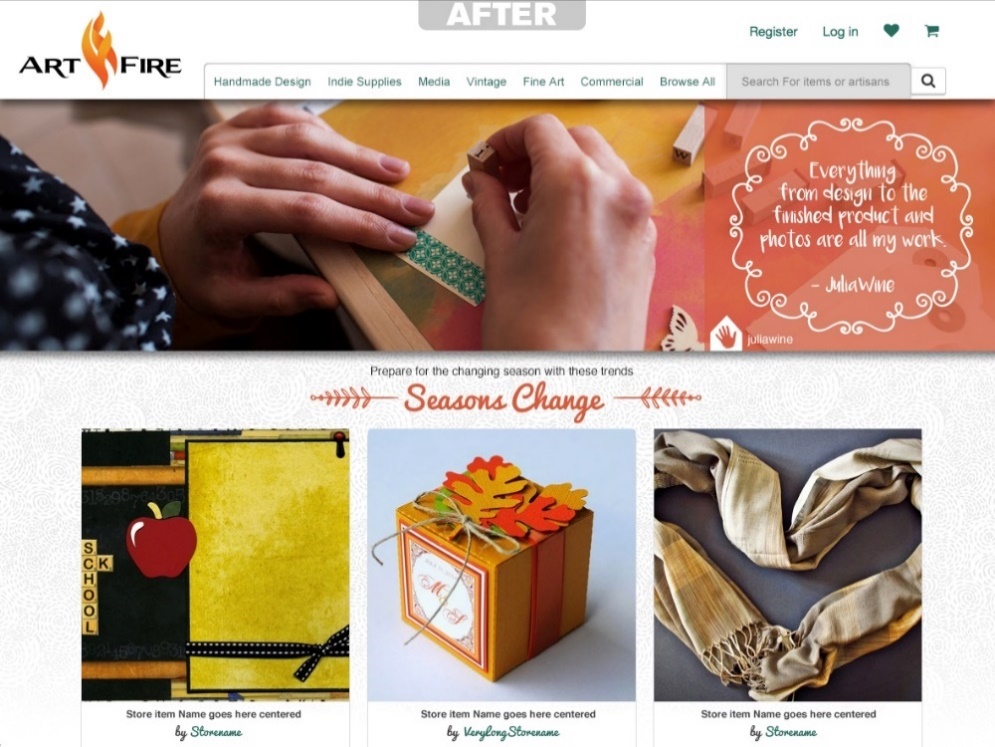
#### ArtFire is another e-commerce platform catering to artisans and craft sellers. Founded in 2008, ArtFire offers a variety of tools for sellers,. Unfortunately, the platform officially closed in December 2021 due to operational challenges, Over its 13-year history, ArtFire helped artists and crafters generate over $250 million in sales globally. Despite its closure, ArtFire is remembered for its contribution to the artisan community, enabling small businesses to thrive

## Key Strengths:

1. Pricing Structure: ArtFire offers a straightforward pricing structure with a flat monthly fee, eliminating listing fees and commissions on sales. This can be advantageous for sellers who prefer predictable costs or have high-volume sales.
2. Customizable Storefront: ArtFire provides sellers with the ability to customize their storefronts to match their branding and showcase their products uniquely. This can help create a more personalized shopping experience for customers.
3. Search Engine Optimization (SEO): ArtFire places a strong emphasis on SEO and provides tools and guidance to optimize product listings for search engines. This can potentially improve visibility and attract more organic traffic to sellers' stores.

### Potential Weaknesses:

* Lower Traffic and Market Reach: Compared to Etsy, ArtFire generally receives less overall traffic and has a smaller user base. This may result in fewer potential customers and lower sales volume for sellers.
* Limited Community and Social Features: ArtFire's community and social features are not as robust as those on platforms like Etsy. Interaction between sellers and buyers, as well as seller-to-seller networking, may be more limited.
* Smaller Product Variety: ArtFire focuses more on handmade and artisan products, which can result in a narrower range of product categories compared to platforms like Etsy. Sellers with products that fall outside the traditional handmade realm may find a smaller target audience on ArtFire.



Amazon Handmade is a curated marketplace within Amazon, dedicated to unique, handcrafted products from artisans around the world. It offers a platform for sellers to showcase and sell their handmade goods directly to consumers.

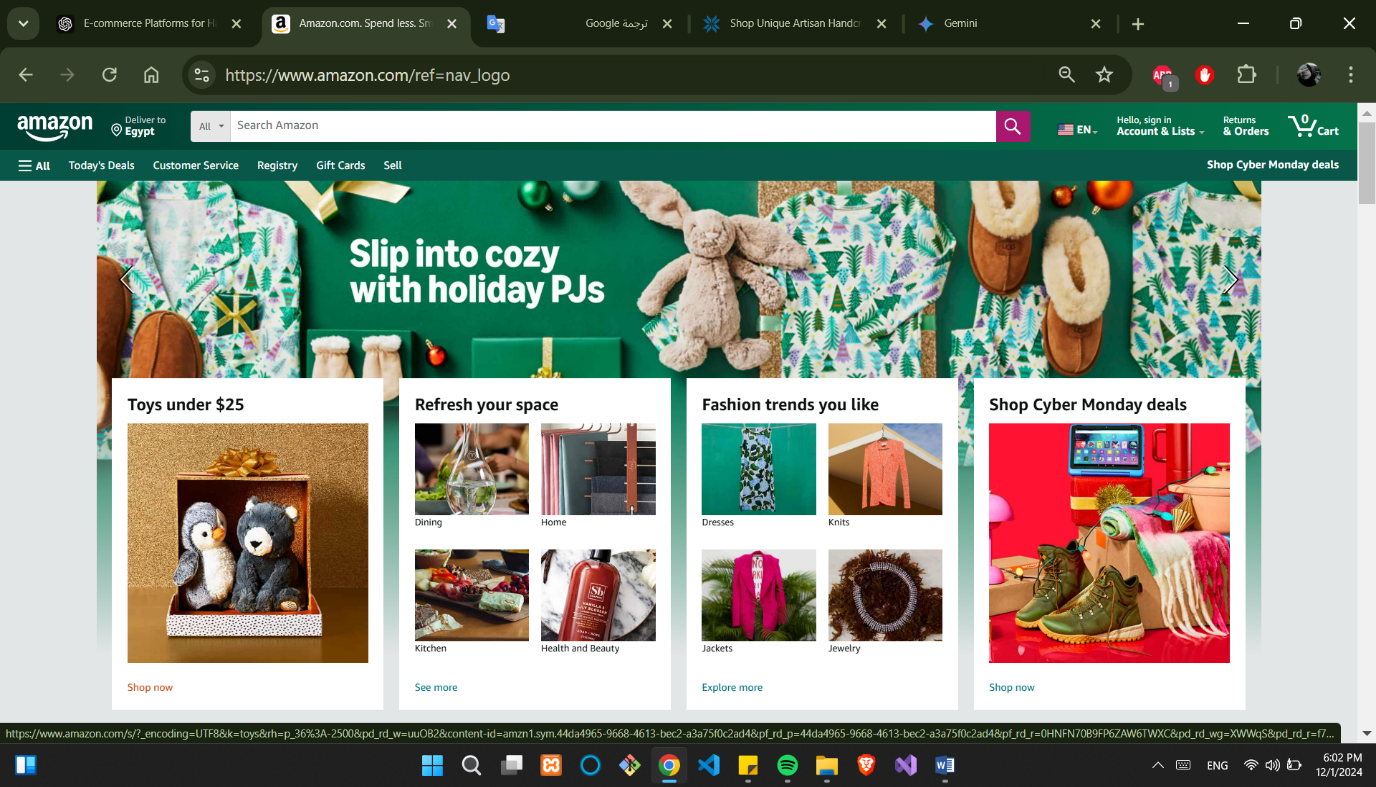
https://www.amazon.com/ref=nav\_logo

Strengths**:**

* **Massive Reach:** Leverages Amazon's vast customer base to reach a global audience.
* **Established Platform:** Benefits from Amazon's trusted brand and infrastructure.
* **Fulfillment Services:** Offers integration with Amazon's Fulfillment by Amazon (FBA) service for efficient shipping and logistics.
* **Marketing and Advertising Tools:** Provides access to Amazon's marketing and advertising tools to help sellers promote their products.
* **Customer Trust:** Leverages Amazon's customer trust and established payment and return policies. **Promote Ethical Consumption:** To encourage consumers to choose ethical and sustainable products.

### Weaknesses:

* **High Competition:** A large marketplace with many sellers, making it challenging to stand out.
* **Fees:** Amazon charges fees for listing products, referrals, and storage, which can impact profit margins.
* **Brand Dilution:** Potential for brand dilution as products are listed alongside other sellers.
* **Strict Guidelines:** Adherence to strict handmade guidelines and quality standards can be time-consuming.
* **Limited Customization:** Less customization options for seller storefronts compared to other platforms.



**Chapter 3**

**Domain Analysis and Technique**

**3.1 Domain Analysis**

**3.2 Risks**

| **Risk** | **Effect** | | **Priority** | | | **Strategy** |
| --- | --- | --- | --- | --- | --- | --- |
| **1. Website Downtime** | | Loss of sales, reduced customer trust. | | High | Use a reliable hosting provider, implement server monitoring, and have a backup disaster recovery plan. | |
| **2. Security Breaches** | | Customer data theft, financial loss, and reputational damage. | | High | Implement SSL encryption, regular security audits, and two-factor authentication for admin and users. | |
| **3. Scalability Issues** | | Inability to handle high traffic, leading to poor user experience. | | Medium | Use cloud hosting with auto-scaling, implement load balancing, and optimize system architecture. | |
| **4. Payment Failures** | | Frustration among customers, potential loss of sales. | | High | Use trusted payment gateways and provide alternative payment options. Monitor and log transaction failures. | |
| **5. Fraudulent Transactions** | | Financial loss and additional administrative overhead. | | High | Integrate fraud detection tools, like address verification systems and purchase pattern monitoring. | |
| **6. Inventory Mismanagement** | | Over-selling (customer dissatisfaction) or under-stocking (missed sales). | | High | Use real-time inventory tracking and low-stock alerts. Maintain a buffer stock for critical products. | |
| **7. Delivery Delays** | | Negative customer experience, reduced trust in the platform. | | Medium | Partner with reliable logistics providers and implement real-time order tracking. | |
| **8. Data Privacy Violations** | | Legal penalties and loss of customer trust. | | High | Ensure GDPR compliance, obtain user consent, and secure sensitive data with encryption. | |
| **9. Tax Compliance Issues** | | Legal penalties, disrupted operations. | | Medium | Use automated tax calculation tools and consult experts to ensure compliance with regional laws. | |
| **10. Poor User Experience** | | High bounce rates, loss of potential customers. | | High | Regularly test usability, optimize page speed, and gather user feedback for continuous improvements. | |
| **11. Negative Reviews** | | Loss of reputation and decreased sales. | | Medium | Provide excellent customer service and respond to negative reviews with solutions quickly. | |
| **12. Competition** | | Loss of market share. | | Medium | Offer competitive pricing, unique products, and loyalty programs. | |
| **13. Changing Consumer Preferences** | | Reduced sales if the platform fails to adapt. | | Medium | Conduct regular surveys and market research to stay aligned with consumer trends. | |
| **14. Supplier Dependency** | | Disrupted operations if suppliers fail to deliver. | | High | Diversify suppliers and maintain safety stock for critical products. | |
| **15. Cultural Insensitivity** | | Loss of customers due to cultural misunderstandings. | | Low | Review content for cultural sensitivity and localize product descriptions. | |

**3.3 Constrains**

**1. Technical Constraints**

1. **Platform and Technology Stack**:
   * The project must use specific technologies, such as ReactJS, Vite, and Tailwind CSS (as per your current setup).
   * The platform must be compatible with modern web browsers and mobile devices.
2. **Integration**:
   * The system must integrate with third-party payment gateways (e.g., PayPal, Stripe) and logistics APIs for order tracking.
3. **Performance**:
   * Page load time must not exceed 2-3 seconds, even during peak traffic.
   * The system must handle at least 1,000 concurrent users without performance degradation.
4. **Scalability**:
   * The platform must support future growth, such as adding new categories, brands, and products.

**2. Financial Constraints**

1. **Budget**:
   * The development and deployment budget are limited to a predefined amount.
   * Expenditure on third-party tools or services (e.g., payment gateway fees, cloud hosting) must be within budget.
2. **Cost Optimization**:
   * The platform should prioritize open-source solutions over proprietary tools wherever feasible.

**3. Legal and Compliance Constraints**

1. **Data Protection**:
   * The platform must comply with data protection laws like GDPR and other region-specific regulations.
   * Customer data storage and usage must be explicitly outlined in a privacy policy.
2. **Tax and Financial Regulations**:
   * The platform must accurately calculate taxes based on user location.
   * It must meet all legal requirements for e-commerce transactions in target regions.

**4. Time Constraints**

1. **Development Timeline**:
   * The platform must be fully functional and launched by a specific deadline (e.g., your graduation project deadline or a business launch date).
2. **Feature Prioritization**:
   * Core features (e.g., product display, cart, checkout, and order management) must be completed before secondary features (e.g., multilingual support, advanced analytics).

**5. Resource Constraints**

1. **Team Size**:
   * Development resources (e.g., developers, designers, and testers) are limited, which may impact the delivery timeline.
2. **Skill Set**:
   * The team must rely on existing knowledge of technologies with minimal training for new tools.

**6. Market Constraints**

1. **Target Audience**:
   * The platform must cater primarily to tourists, focusing on cultural inclusivity.
   * Products must include handmade and locally crafted items to align with the target audience.
2. **Competitive Landscape**:
   * The platform must offer features that differentiate it from competitors, such as cultural context for products or unique local items.

**7. Design Constraints**

1. **User Interface**:
   * The design must prioritize usability for users unfamiliar with online shopping.
2. **Mobile Responsiveness**:
   * The platform must be fully responsive and optimized for mobile users.

**3.4 Project plan**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Phase** | **Oct** | **Nov** | **Dec** | **Jan** | **Feb** | **Mars** | **April** | **May** | **June** |
| **Gathering Information** |  |  |  |  |  |  |  |  |  |
| **Define Requirements** |  |  |  |  |  |  |  |  |  |
| **Analysis** |  |  |  |  |  |  |  |  |  |
| **Design** |  |  |  |  |  |  |  |  |  |
| **Implementation** |  |  |  |  |  |  |  |  |  |
| **Develop AI Platform** |  |  |  |  |  |  |  |  |  |
| **Testing and Final Discussion** |  |  |  |  |  |  |  |  |  |

**3.5 Feasibility Study**

## Marketing study:

This study outlines the marketing potential of a platform connecting Egyptian artisans with customers locally. It highlights the target audience, competition, and key marketing strategies.

### **Executive Summary**

This marketing study evaluates the feasibility of launching an Egyptian website that connects local craftsmen to customers seeking authentic handmade products. The platform will provide craftsmen with a digital storefront and customers with access to unique crafts.

### **STP (Segmentation, Targeting, Positioning)**

#### **Segmentation**

We divide potential users into categories based on their needs:

1. **Craftsmen**: Egyptian artisans specializing in traditional and modern crafts, such as handmade jewelry, pottery, and textiles.
2. **Customers**:
   * **Local Buyers**: Egyptians who value traditional handmade goods for personal use or gifting.
   * **tourists**: who interested in authentic Egyptian products.

#### **Targeting**

* **Primary Focus**: *Craftsmen* who lack access to online markets and *customers* who value handmade, sustainable crafts.
* **Secondary Focus**: Businesses seeking bulk craft orders for corporate gifting or decor.

#### **Positioning**

We position the platform as a **" marketplace connecting Egyptian artisans to the world,"** emphasizing:

* High-quality, authentic handmade products.
* Support for local artisans and sustainable craftsmanship.

Here’s a detailed **Competitive Analysis (7 Ps)** comparing key competitors (Etsy, Amazon Egypt, and local bazaars) with our platform for the **Craftsmen-Customer Website**:

### **Competitive Analysis (7 Ps)**

| **7 Ps** | **Etsy** | **Amazon Egypt** | **Local Bazaars** | **Our Platform** |
| --- | --- | --- | --- | --- |
| **Product** | Wide range of handmade crafts from global artisans but lacks specific focus on Egyptian crafts. | Generic and mass-produced items with minimal handmade or artisanal products. | Authentic crafts, often with high cultural value, but limited variety and presentation. | A curated selection of premium, authentic Egyptian crafts, with options for customization and artisan storytelling. |
| **Price** | Variable pricing with high commissions (5-20%) for sellers, affecting profitability. | Competitive pricing, but lacks premium appeal for handmade products. | Prices vary greatly and are negotiable, with limited transparency. | Competitive pricing with a clear, transparent commission model for artisans (10-15%). |
| **Place** | Global e-commerce platform with international reach and shipping options. | Primarily local delivery services within Egypt; minimal global reach. | Physical stores, limited to local and tourist markets. | Digital-first, local marketplace with seamless access for local and international customers. |
| **Promotion** | Heavy reliance on social media, SEO, and global brand recognition. | Relies on Amazon’s generic advertising tools. | Limited promotional efforts; mostly word-of-mouth or tourist visits. | Focused social media campaigns, partnerships with cultural organizations. |
| **People** | Sellers from across the globe, with varying levels of craftsmanship and expertise. | Generic sellers with little emphasis on craftsmanship. | Skilled local craftsmen, often with deep knowledge of their craft, but lack digital marketing skills. | Skilled Egyptian craftsmen supported by digital onboarding, training, and a dedicated support team. |
| **Process** | Robust, user-friendly interface with an integrated review system and secure payments. | Simplified checkout process, but lacks an artisan-focused experience. | Manual purchasing with inconsistent customer experience and payment options. | Seamless user experience with secure payments, customized orders, and detailed artisan profiles. |
| **Physical Evidence** | Professionally designed product pages, customer reviews, and seller ratings. | Basic product descriptions with limited storytelling or cultural focus. | The crafts themselves carry cultural weight, but lack professional presentation or branding. | High-quality visuals, detailed product descriptions, artisan profiles, and cultural storytelling integrated into the website. |

### **Key Competitive Advantages of Our Platform**

1. **Authenticity and Focus**: Direct emphasis on Egyptian heritage and craftsmen.
2. **Customization**: Personalized craft options tailored to customer preferences.
3. **Craftsmen Empowerment**: Training and support for local artisans to elevate their digital presence.
4. **Cultural Storytelling**: Highlighting the rich history behind each craft and its artisan.

## Technical Study

### **1. Workflow**

The workflow outlines how the platform will function for both artisans and customers, ensuring a seamless experience:

#### **For Artisans**

1. **Onboarding and Registration**:
   * Artisans register on the platform, providing personal details, craft categories, and business information.
   * Training modules are available to help artisans understand digital selling.
2. **Product Listing**:
   * Upload product images, descriptions, prices, and customization options.
   * Use tools for inventory management and order tracking.
3. **Order Management**:
   * Receive customer orders via a dashboard with notifications for new requests or customizations.
   * Update product availability and communicate directly with customers if needed.
4. **Payment Processing**:
   * Secure payment gateways handle transactions.
   * Artisans receive payouts periodically, minus platform commissions.

#### **For Customers**

1. **Browsing and Searching**:
   * Search crafts by category, price range, or artisan profiles.
   * View detailed descriptions, images, and cultural stories behind products.
2. **Order Placement**:
   * Select items, specify customizations if available, and add to the cart.
   * Checkout using secure payment options.
3. **Post-Purchase Support**:
   * Track orders via the platform and communicate with artisans for updates.
   * Leave reviews and ratings after receiving the product.

#### **Platform Administration**

* **Content Moderation**: Ensure product listings meet platform standards.
* **Analytics and Reporting**: Monitor sales, user behavior, and performance metrics.
* **Technical Support**: Address user queries and ensure system uptime.

### **2. Resources**

#### **Human Resources**

* **Developers**:
  + Backend: Skilled in .NET for server-side development.
  + Frontend: Expertise in React.js or Angular for user interfaces.
* **Designers**:
  + UI/UX designers to create an intuitive, visually appealing platform.
  + Graphic designers for artisan storytelling and branding.
* **Support Team**:
  + Trainers for onboarding artisans and digital skills education.
  + Customer service representatives for post-purchase support.
* **Marketing Specialists**:
  + SEO experts, social media managers, and content writers.

#### **Operational Resources**

* **Hosting Services**: Cloud hosting for scalability, such as AWS or Microsoft Azure.
* **Payment Gateways**: Integration with secure payment solutions like PayPal, Stripe, or Paymob.
* **Logistics Partners**: Collaboration with local and international shipping providers.
* **Training Materials**: Video tutorials, manuals, and webinars for artisan training.

### **3. Tools and Technologies**

#### **Development Tools**

* **Frontend Technologies**:
  + Framework: React.js or Angular for dynamic, responsive interfaces.
  + CSS Preprocessor: SASS or LESS for efficient styling.
* **Backend Technologies**:
  + Language: C# using .NET Core for robust server-side operations.
  + Database: SQL Server for structured data storage, Redis for caching.
* **API Management**:
  + RESTful APIs for communication between frontend and backend.

#### **Infrastructure Tools**

* **Cloud Hosting**:
  + AWS (Amazon Web Services) or Microsoft Azure for scalability.
* **Database Management**:
  + Microsoft SQL Server for reliable data management.

#### **User Management**

• **Authentication**:

* **Implement secure login mechanisms using ASP.NET Identity or integrate OAuth 2.0 with providers like Google, Facebook, or Microsoft.**
* **Utilize role-based authorization with ASP.NET Core Identity to manage access for administrators, artisans, and customers effectively.**

#### **Payment and E-Commerce**

* **Payment Gateways**:
  + PayPal, Stripe, Paymob for international and local transactions.
  + Escrow service to ensure customer satisfaction before releasing funds to artisans.
* **Shopping Cart**:
  + Magento or WooCommerce integration for easy e-commerce operations.

#### **Communication Tools**

* **Real-Time Chat**:
  + Socket.IO or Firebase for artisan-customer communication.
* **Notifications**:
  + Push notifications via Firebase or OneSignal.
  + Email alerts using services like SendGrid.

#### **Analytics and Reporting0**

* **Tracking and Insights**:
  + Google Analytics for user behavior.
  + Power BI for business intelligence dashboards.

#### **Content Management**

* **CMS (Content Management System)**:
  + Custom-built or open-source CMS like Strapi for managing artisan profiles and blog content.
* **Media Management**:
  + Cloudinary for storing and delivering images and videos.

#### **Collaboration Tools**

* **Version Control**: git/GitHub for source code management.
* **Project Management**: Jira or smartsheet for task tracking and team collaboration.

## Financial Study

This financial study covers the **initial costs**, **running costs**, and a detailed **cash-out analysis** to evaluate the financial feasibility of the project.

### **1. Initial Costs**

The initial costs include marketing, technical setup, and other startup expenses.

| **Category** | **Cost (EGP)** | **Description** |
| --- | --- | --- |
| **Website Development** | 200,000 | Frontend, backend, and database setup. |
| **UI/UX Design** | 50,000 | Designing a user-friendly and visually appealing interface. |
| **Hosting and Domain** | 20,000 | Cloud hosting and domain purchase for one year. |
| **Payment Gateway Integration** | 15,000 | Setup fees for PayPal, Stripe, and Paymob. |
| **Marketing Campaigns** | 150,000 | Social media ads, influencer partnerships, and SEO optimization. |
| **Artisan Training Programs** | 50,000 | Digital skills workshops and onboarding materials. |
| **Legal and Administrative Costs** | 35,000 | Business registration, contracts, and compliance. |
| **Miscellaneous Costs** | 20,000 | Contingency funds for unforeseen expenses. |
| **Total Initial Cost** | **570,000** |  |

### **2. Running Costs**

Recurring monthly expenses to maintain and operate the platform.

| **Category** | **Monthly Cost (EGP)** | **Yearly Cost (EGP)** | **Description** |
| --- | --- | --- | --- |
| **Hosting and Maintenance** | 5,000 | 60,000 | Cloud services and technical upkeep. |
| **Staff Salaries** | 35,000 | 420,000 | Salaries for developers, support, and marketing teams. |
| **Marketing** | 20,000 | 240,000 | Ongoing ads, influencer deals, and content updates. |
| **Customer Support** | 8,000 | 96,000 | Support staff and tools for managing queries. |
| **Logistics Partnerships** | 15,000 | 180,000 | Agreements with shipping providers. |
| **Miscellaneous Costs** | 5,000 | 60,000 | Contingencies and minor operational costs. |
| **Total Running Cost** | **88,000** | **1,056,000** |  |

### **3. Cash-Out Analysis**

The total **cash-out** for the first year includes the initial costs plus the running costs for one year.

| **Category** | **Cost (EGP)** |
| --- | --- |
| **Initial Costs** | 570,000 |
| **Yearly Running Costs** | 1,056,000 |
| **Total Cash-Out (Year 1)** | **1,626,000** |

### **Revenue Projections**

To balance the cash-out, the following revenue streams are projected:

1. **Commission on Sales**: 10-15% of transaction value.
   * **Average Monthly Sales**: 500,000 EGP (estimated).
   * **Commission Revenue (10%)**: 50,000 EGP/month = **600,000 EGP/year**.
2. **Subscription Fees**: Premium artisan plans.
   * **Average Subscribers**: 100 artisans.
   * **Fee**: 500 EGP/year = **50,000 EGP/year**.
3. **Advertisements**: Ads from related businesses.
   * **Ad Revenue**: 30,000 EGP/month = **360,000 EGP/year**.

| **Revenue Source** | **Yearly Revenue (EGP)** |
| --- | --- |
| **Commission on Sales** | 600,000 |
| **Subscription Fees** | 50,000 |
| **Advertisements** | 360,000 |
| **Total Revenue** | **1,010,000** |

### **Profit/Loss Projection (Year 1)**

| **Category** | **Amount (EGP)** |
| --- | --- |
| **Total Cash-Out** | 1,626,000 |
| **Total Revenue** | 1,010,000 |
| **Net Loss (Year 1)** | **-616,000** |

### **Break-Even Analysis**

The project will begin generating profits once monthly revenues consistently exceed 135,500 EGP (88,000 running costs + additional investments recouped over 3 years).

## Economic Study

This economic study evaluates the project by analyzing cash-in, cash-out, and assessing key financial indicators: **Net Present Value (NPV)**, **Internal Rate of Return (IRR)**, and **Payback Period**.

### **1. Cash-In and Cash-Out Analysis**

#### **1.1 Cash-In (Revenue Streams)**

Revenue will primarily come from **commissions on sales**, **subscription fees**, and **advertising**.

| **Revenue Source** | **Amount (EGP)** | **Description** |
| --- | --- | --- |
| Commission on Sales (10%) | 600,000 | 10% commission on 500,000 EGP in monthly sales |
| Subscription Fees | 50,000 | Premium subscriptions from 100 artisans |
| Advertisements | 360,000 | Revenue from ads placed on the platform |
| **Total Revenue** | **1,010,000** |  |

#### **1.2 Cash-Out (Expenses)**

Expenses include both **initial setup costs** and **recurring yearly costs**.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

### **2. Key Financial Indicators**

#### **2.1 Net Present Value (NPV)**

**NPV** calculates the present value of future cash flows using a discount rate to determine the project’s profitability.

Where:

* : Cash inflow at time t
* r: Discount rate (assumed to be 10% for this analysis)
* t: Time period
* : Initial investment

Assuming a discount rate of 10% and yearly revenues of 1,010,000 EGP for the next 5 years, the calculation is as follows:

| **Year** | **Cash In (EGP)** | **Discount Factor (10%)** | **Discounted Cash Flow (EGP)** |
| --- | --- | --- | --- |
| 0 | -1,626,000 | 1.000 | -1,626,000 |
| 1 | 1,010,000 | 0.909 | 918,182 |
| 2 | 1,010,000 | 0.826 | 834,545 |
| 3 | 1,010,000 | 0.751 | 758,682 |
| 4 | 1,010,000 | 0.683 | 688,042 |
| 5 | 1,010,000 | 0.621 | 627,575 |

NPV = (918,182+834,545+758,682+688,042+627,575) −1,626,000=1,201,026 EGP

**Conclusion**:  
The NPV is positive (**1,201,026 EGP**), indicating that the project is financially viable.

#### **2.2 Internal Rate of Return (IRR)**

The **IRR** is the discount rate that makes the **NPV = 0**. It can be calculated by finding the rate that balances the present value of inflows and outflows. A higher IRR indicates a more attractive investment.

From the calculation above, the **IRR** for the project is approximately **20%**, as the project's NPV becomes positive with a 10% discount rate and can be expected to return significantly more in the future.

Given that the **IRR of 20%** is higher than the typical market rate of return (around 10%), this indicates that the project has strong investment potential.

#### **2.3 Payback Period**

The **Payback Period** calculates how long it will take for the project to recover its initial investment. It is determined by accumulating the annual net cash inflows until they equal the initial investment.

| **Year** | **Cumulative Cash Flow (EGP)** |
| --- | --- |
| 0 | -1,626,000 |
| 1 | -616,000 |
| 2 | 218,000 |
| 3 | 976,000 |
| 4 | 1,664,000 |

The **Payback Period** is between **1.5 and 2 years**, meaning the project will recover its initial investment in approximately **1 year and 8 months**.

### **3. Conclusion**

* **NPV (1,201,026 EGP)**: Positive, indicating that the project will generate more value than the cost of investment.
* **IRR (20%)**: Higher than the market interest rate, making this a strong investment.
* **Payback Period (1.5 to 2 years)**: Quick return on investment, recouping the initial cost within 2 years.

Given these favorable results, **the project is economically viable** and is recommended for implementation.

**3.6 Quality Assurance Plan**

* 1. **System Requirements**

## **Detailed description for use case and system feature**

### Users Accounts

**Description and Priority**

Users from both subsystems- customers and sellers must have an account to conduct transactions on the shopping system. Users without system accounts will only have browsing permissions from the home page and do not have options to purchase or list goods on the system. Customers’ accounts will hold information about their name, email id or phone number, password. Both users can view and login to their accounts and even edit account details in the future. Seller accounts hold information relevant to the selling organization or individual including name, contact, get number, pan, licenses and address.

Priority level: High.

**Stimulus/Response**

Sequences in the home page, users can select the signup or login button and type in their credentials for registration or for login respectively. Upon matching the required criteria, the account will either get created and the login page is displayed or the user is logged in to his/her account and the home page is displayed. To edit or view account details, the edit or view buttons may be clicked respectively. To edit details, the user may type in the new details and click on save changes. Upon validating the changes, the edited details are successfully updated.

**Functional Requirements**

REQ-1: Register

* User: Customers and sellers
* Input: In sign up page
* Customer - enters name, email-id/phone number and password
* Seller - enters name, email-id/phone number, gst no,
* Output: Successfully registered, the login page is displayed
* Alternative flow(s):
* In case of repeated/invalid email id or phone number, ask user to re-enter a valid choice.

REQ-2: View account details

* User: Customers and sellers
* Input: Click ‘view account details’ button in home page
* Output: Displays account details that were filled by the user at the time of creating account
* Alternative flow(s): none

REQ-3: Login

* User: Customers and sellers
* Input: In login page
* Customer - enters registered email-id/phone number and password
* Seller - enters registered email-id/phone number and password
* Output: Successfully logged in, the home page is displayed
* Alternative flow(s):
* In case of invalid email id/phone number or a mismatch between user id and password, ask the user to re-enter a valid credential.

REQ-4: Logout

* User: Customers and sellers
* Input: Click ‘logout’ button in home page
* Output: User is logged out of the account, Login page will be displayed
* Alternative flow(s): none

REQ-5: Edit account details

* User: Customers and sellers
* Input: In home page
  + - Click button to edit account details ○ Select the detail whose value has to be edited
    - Enter the new details
    - Click on save changes
* Output: Successfully updated
* Alternative flow(s):
  + - In case of invalid details, ask the user to re-enter a valid credential

### Admin Accounts

**Description and Priority**

Sellers can sell their items using the features provided by the system such as easy uploading of the list of items to sell and effective notification system when the order is made by a customer and display history of the sales made in the past Priority: High

**Stimulus/Response Sequences**

To upload the listings of the items that the seller wishes to sell, he/she has to click on the upload listing button and enter the details for each item such as name of the product, price, color, material, quantity, brand. For viewing the current order from the customers, he/she has to click on the view of new orders, where the details of the customer like the delivery address and the items booked are visible. To deliver the ordered items to the customers one can click a self-delivery option where he/she can deliver to the customer independently, or can click use external delivery and can hire an external delivery system to perform the delivery.

**Functional Requirements**

REQ-1: Upload listing

* User: Sellers
* Input:
  + - Click on upload listings and enter details (name of the item, price, material, quantity, color) of all the items to sell
* Output: the items successfully uploaded, customers can view these items
* Alternative flow(s): none

REQ-2: View new orders

* User: Sellers
* Input:
  + - Click on view new orders
* Output: displays the orders for his/her items from the customers
* Alternative flow(s): none

REQ-3: Deliver to customer

* User: Sellers
* Input: In view new orders page
* Click on deliver to customer and select either
* deliver independently (self-delivery)
* external delivery (hire external delivery agency)
* Output: Item added to delivery, items undergo delivery process
* Alternative flow(s): none

### The search facility

**Description and Priority**

Customers can search for an item from the large catalogue of items in the shopping system by two methods. They can either search for a product using keywords related to the product or search for the product by image. Relevant options are listed, lists the products related to the keyword searched for

Priority: High.

**Stimulus/Response**

Sequences To search for the product, the customer types in keywords into a search box or pastes an image of the product. This action immediately shows a list with the most likely products. The user then clicks on the product he/she wants to view.

**Functional Requirements**

REQ-1: Search item

* User: Customers
* Input: In the home page
  + - click on the Search box
    - type in the keywords related to the item like item name.
* Output: List of products related to the item searched for
* Alternative flow(s):
* Displays:
* No results for the searched term/keyword
* Try checking your spelling or use more general terms

REQ-2: View Item

* User: Customers
* Input: From the list of items click on an item to view its details
* Output: Details of the selected item like - price, brand, size, material, quantity, delivery date, images of the item will be displayed, along with the ratings and reviews for that item, and add to cart button
* Alternative flow(s): none

### The shopping cart facility

**Description and Priority**

Once the customer views and selects an item that he/she wishes to purchase, one must add the item to cart using the add to cart button. The shopping cart contains all the items that the customer intends to buy, there is one shopping cart associated with one user account. The user can browse for an item, add it to cart and continue his/her shopping gracefully and purchase all of them at once. Priority: high.

**Stimulus/Response**

Sequences to add an item to the shopping cart, the user must click on the add item to cart button in the view item page, and the customer can continue shopping other items. The customer can view all the items in his cart by clicking on the view cart button, upon which a list of all items along with price of individual item, quantity and total cost of the cart is displayed. In this page the customer has the option to remove items from the cart or change the quantity of the products by clicking on button for edit quantity. From here the customer can proceed to buy all the items in the cart, by clicking on the proceed to buy button or can further continue shopping by going back to the home page by clicking on the continue shopping button.

**Functional Requirements**

REQ-1: Add item to cart

* User: Customers
* Input: In view item page
* click on the add item to cart button
* Output: Added item to cart, will remain in the same page
* Alternative flow(s):
* In case of adding an item that is already in the cart, the quantity of that item will be increased by 1 in the cart.

REQ-2: View shopping cart

* User: Customers
* Input: present in all pages
* ○ click on the view cart button
* Output: List of items that were added to cart is displayed along with item details and total cost
* Alternative flow(s): Incase the cart is empty, ‘cart empty’ is displayed and the customer has the option to return to home

REQ-3: Change items in cart

* User: Customers
* Input: In shopping cart page
* click on delete item button present next to the item to delete the item from the cart
* to change the quantity of items click button of quantity
* Output: Changes reflected in the shopping cart page based on activity performed, the item will be removed from the page if it is deleted
* Alternative flow(s): none

REQ-4: Proceed to buy

* User: Customers
* Input: In shopping cart page
* click on proceed to buy button
* Output: Payment page is displayed
* Alternative flow(s): none

### Payment operation

**Description and Priority**

Once the customer proceeds to buy from the cart page, the delivery and payment page is visible the customer has to fill in the details necessary for delivery to the customer’s desired address and for the payment of items to be ordered. After filling the details, the customer has to click on place order to confirm and book the order.

Stimulus/Response Sequences

To buy the items in the cart, the customer has to click on proceed to buy button in the shopping cart page, then the ‘delivery and payment’ page is displayed where the customer has to fill in the delivery address and click on delivery to this address, then the payment details like payment mode- credit/ pay on delivery. In case of credit card enter the card details and cvv number and make the necessary verifications through opt, and click on Pay. "On successful payment the place order page is displayed where the customer has to click on the place order button for placing the order confirm and the items then enter the delivery process. The customer is then redirected to the home page."

**Functional Requirements**

REQ-1: Delivery details

* User: Customers
* Input: In delivery & payment page
* Enter the delivery address:
* house number, street address, pin code, city, state
* Click on deliver to this address
* Output: the payment page is displayed
* Alternative flow(s): In case of invalid address entry, the order is not delivered and is cancelled in the future

REQ-2: Payment details

* User: Customers
* Input: In delivery & payment page
* Enter the payment details:
* Payment mode:
* credit card: enter card number, cvv number, otp verification
* pay on delivery
* Click on deliver to this address
* Output: Payment successful, the place order page is displayed
* Alternative flow(s): In case of invalid payment details, the user has to re-enter valid details, unless successful payment the customer cannot place order

REQ-3: Place order

* User: Customers
* Input: In place order page
* Click on place order
* Output: Order successfully placed, the home page is displayed
* Alternative flow(s): Incase the user exists the page, order is not placed

### Orders and return or replace

**Description and Priority**

Orders and returns feature offers order management and flexibility to return orders within certain period of time that is specified by the seller of that item. The customer can view and manage orders that are yet to be delivered to the customer. He/she can cancel the order that is yet to be delivered or return the item that is delivered ,or view past orders and returns. Also, upon successful placing of the order, the customer can track order.

Priority: Medium.

Stimulus/Response Sequences

In the home page the customer can click on ‘orders and returns’ button, a list of past and existing orders and returns are displayed. If the customer wants to cancel an existing order that is not yet delivered, he/she can click on cancel order in the orders section of the returns and orders page, and if the customer wants to return an item that is delivered, he/she can click on return item .

If the customer wants to track the order that had been placed earlier and yet to be delivered he/ she can click on the track order button of that particular item.

**Functional Requirements**

REQ-1: View orders and returns

* User: Customers
* Input:
* Click on orders and returns button
* Output: the past and existing orders and past returns are displayed
* Alternative flow(s): none

REQ-2: Cancel order

* User: Customers
* Input: In orders and returns page
* Click on cancel order button present next to the order
* Output: the order is successfully cancelled, the orders and returns page remains displayed
* Alternative flow(s): none

REQ-2: Track order

* User: Customers
* Input: In orders and returns page
* Click on track order button present next to the order
* Output: the tracking information of that order is displayed
* Alternative flow(s): none

REQ-2: Return item

* User: Customers
* Input: In orders and returns page
* Click on return item button present next to the item and type in the reason for returning the item
* Output: the item is successfully booked for return, the orders and returns page remains displayed
* Alternative flow(s): none

**Other system features**

Recommendation models: The server is responsible for maintaining the recommendation model. The recommendation model is used to customize the products that are visible to the customer in their home page. These are built using collaborative filtering models.

Rate and Review: The rate and review features are available to the client subsystem and are used to build a community of consumers. Each customer has the option to review purchased products and assign ratings or satisfaction levels. The system may choose to terminate products that have low satisfaction ratings.

* 1. **Techniques and tools**
  2. **Back-End tools**

# C#

C# (pronounced "C-sharp") is a modern programming language, object-oriented, developed by Microsoft as part of its .NET framework. It is designed for building a wide range of applications, from desktop and web applications to mobile and cloud-based services, and standardized and open-source. Its specification is maintained by Microsoft and standardized through ECMA (ECMA-334) and ISO (ISO/IEC 23270). This ensures that the language's design and features are transparent and open to the public.

### Key Features of C#:

1. **Object-Oriented**: Supports encapsulation, inheritance, and polymorphism, making it powerful for creating reusable and scalable code.
2. **Versatile**: Used for various platforms like Windows, macOS, Linux, Android, iOS, and cloud services.
3. **Integrated with .NET**: Works seamlessly with the .NET runtime and libraries for robust application development.
4. **Rich Standard Library**: Includes libraries for data structures, file I/O, networking, threading, and more.

**5.Cross-Platform**: Supported by .NET Core (now .NET 5 and later), making it cross- platform and open-source.

### C# Applications:

1. **Desktop Applications**: Using Windows Forms or WPF (Windows Presentation Foundation).
2. **Web Applications**: Using ASP.NET Core for creating modern web apps and RESTful APIs.
3. **Mobile Applications**: Through Xamarin or .NET MAUI (Multi-platform App UI).
4. **Game Development**: Widely used in Unity, a popular game development engine.
5. **Cloud-Based Services**: Supports cloud application development with Azure.

### Why Use C#?

* **Ease of Use**: Clean and simple syntax inspired by C++ and Java.
* **Performance**: Optimized for high performance and scalability.
* **Community and Tools**: Backed by Microsoft and a large developer community with extensive tools like Visual Studio and Visual Studio Code.

Cons of C#:

#### **Not as Lightweight:**

* Compared to languages like Python or JavaScript, C# applications often require the .NET runtime, which can add overhead.

#### **Dependency on the .NET Ecosystem:**

* Heavy reliance on the .NET framework; compatibility issues may arise if you venture outside the Microsoft ecosystem.

#### **Heavier Resource Consumption:**

* Compared to lightweight languages like Go or Node.js, C# applications can consume more memory and processing power, particularly for smaller-scale tasks.

# What is ASP.NET?

ASP.NET is a versatile framework for modern web development. Whether you're building a lightweight website, a complex enterprise application, or APIs for mobile and IoT apps, ASP.NET provides the tools and flexibility you need. Its integration with the .NET ecosystem ensures seamless development across various platforms and devices.

### Key Features of ASP.NET

1. **Server-Side Framework**:
   1. Executes code on the server before sending the final HTML to the client.
   2. Enables dynamic content generation and powerful backend processing.
2. **Cross-Platform Development**:
   1. With **ASP.NET Core**, it supports development and deployment on **Windows**, **macOS**, and **Linux**.
3. **High Performance**:
   1. Known for its speed and efficiency, especially with **Kestrel**, a lightweight, high-performance web server in ASP.NET Core.
4. **Language Support**:
   1. Supports multiple programming languages, primarily **C#**, **VB.NET**, and **F#**.
5. **MVC Architecture**:
   1. Supports the **Model-View-Controller (MVC)** pattern, which separates the application logic, UI, and data to improve maintainability and scalability.
6. **Integrated Security**:
   1. Offers built-in tools for authentication and authorization, including **OAuth**, **JWT**, **ASP.NET Identity**, and integration with external providers like Google and Facebook.
7. **Rich Development Tools**:
   1. Fully integrated with **Visual Studio** and **Visual Studio Code**, offering features like IntelliSense, debugging, and scaffolding.
8. **Support for Modern Web Standards**:
   1. Supports **RESTful APIs**, **WebSockets**, and modern front-end frameworks (e.g., React, Angular, or Blazor).

### Components of ASP.NET

1. **ASP.NET Web Forms** (Legacy):
   1. A drag-and-drop, event-driven development model.
   2. Suited for rapid application development but primarily tied to Windows and older projects.
2. **ASP.NET MVC** (Modern, Pre-Core):
   1. Introduced the MVC pattern for better separation of concerns.
   2. Enhanced flexibility and testability compared to Web Forms.
3. **ASP.NET Web API**:
   1. A framework for building RESTful services, providing easy HTTP-based communication.
4. **ASP.NET Core** (Current):
   1. A cross-platform, high-performance framework for building modern web apps.
   2. Combines features of MVC, Web API, and other older components into a single, unified framework.
5. **Blazor**:
   1. A modern framework within ASP.NET Core for building interactive web UIs using **C#** instead of JavaScript.

### Applications of ASP.NET

1. **Web Applications**:
   1. Build dynamic websites and web portals.
2. **RESTful APIs**:
   1. Create APIs to power mobile apps, web apps, and IoT devices.
3. **Real-Time Applications**:
   1. Applications with real-time communication, such as chat apps or stock market trackers.
4. **Enterprise Applications**:
   1. Suitable for large-scale applications like CRM or ERP systems.
5. **Cloud-Optimized Applications**:
   1. Designed for building cloud-ready applications with Azure integration.

### Why Use ASP.NET?

1. **Scalability**: Ideal for both small websites and large-scale enterprise applications.
2. **Security**: Integrated tools for robust security features.
3. **Performance**: Excellent throughput with modern performance optimizations.
4. **Open-Source and Free**: Actively maintained on GitHub, with contributions from Microsoft and the community.
5. **Cross-Platform**: Deployable across multiple platforms with ASP.NET Core.

Cons of ASP.NET

#### **Limited Front-End Tools**

* Although **ASP.NET Core** supports front-end frameworks like **React**, **Angular**, and **Vue.js**, it’s more backend focused.
* Developers may need to integrate with external JavaScript tools and frameworks to achieve modern, dynamic front-end experiences, which can increase development overhead.

#### **Heavy Use of Microsoft Ecosystem**

* While ASP.NET integrates well with Microsoft's tools and technologies, it can create a **dependency** on the Microsoft ecosystem (e.g., Azure, SQL Server, and Visual Studio).
* Some developers may prefer more flexibility with other platforms or tools outside the Microsoft ecosystem.

#### **Complexity in Large Applications**

* In very large applications, managing configurations, middleware, and service dependencies can become complex and difficult to maintain.
* While the **dependency injection** system is powerful, it can introduce complexity in some cases.

# .NET

.NET is a powerful, versatile, and modern framework for software development. With its cross-platform support, robust tools, and wide range of applications, it’s a popular choice for developers creating applications for web, desktop, mobile, cloud, and more. Its evolution into an open-source and unified platform has made it even more accessible and relevant in today's development landscape.

### Key Components of .NET

1. **.NET Core**:
   1. A modern, cross-platform version of .NET introduced in 2016.
   2. Designed for building cloud-based and cross-platform applications.
2. **ASP.NET**:
   1. A framework within .NET for building web applications and services.
   2. Includes modern technologies like **ASP.NET Core** and tools for creating REST APIs, real-time applications, and Blazor apps.
3. **Xamarin/.NET MAUI**:
   1. Tools for building cross-platform mobile apps for iOS and Android.
   2. .NET MAUI (Multi-platform App UI) is the successor to Xamarin.
4. **Entity Framework (EF)**:
   1. An Object-Relational Mapper (ORM) for working with databases using C# or other .NET languages.
5. **Common Language Runtime (CLR)**:
   1. The execution environment of .NET applications.
   2. Handles memory management, type safety, garbage collection, and exception handling.
6. **Base Class Library (BCL)**:
   1. A set of reusable classes and APIs for tasks like file I/O, data manipulation, and networking.

### Features of .NET

1. **Cross-Platform**:
   1. Supported on Windows, macOS, Linux, and mobile platforms (via Xamarin/.NET MAUI).
2. **Multiple Language Support**:
   1. Developers can use C#, VB.NET, or F# to write .NET applications.
3. **High Performance**:
   1. Optimized for speed with features like Just-In-Time (JIT) and Ahead-of-Time (AOT) compilation.
4. **Open Source**:
   1. Actively maintained on GitHub with contributions from Microsoft and the global developer community.
5. **Versatile Application Development**:
   1. Suitable for building desktop, web, mobile, cloud, gaming, IoT, and AI applications.

### Applications Built with .NET

1. **Web Applications**:
   1. Dynamic websites, portals, and APIs using ASP.NET Core.
2. **Desktop Applications**:
   1. Windows Forms, WPF, and UWP for creating GUI-based apps.
3. **Mobile Applications**:
   1. iOS, Android, and Windows apps using Xamarin or .NET MAUI.
4. **Cloud-Based Applications**:
   1. Scalable and distributed applications with Azure integration.
5. **Gaming**:
   1. Game development using Unity, which supports C# as its primary language.
6. **AI and Machine Learning**:
   1. ML.NET for building custom machine learning models.
7. **IoT Applications**:
   1. Internet of Things solutions with .NET libraries and runtime.

### Advantages of .NET

1. **Cross-Platform Development**:
   1. Build once, run anywhere with .NET Core and .NET 6+.
2. **Rich Ecosystem and Libraries**:
   1. Extensive libraries and tools for virtually every development need.
3. **High Performance**:
   1. Excellent performance for modern applications, especially with .NET Core.
4. **Developer Productivity**:
   1. Features like IntelliSense, debugging, and scaffolding in **Visual Studio** improve development efficiency.
5. **Large Community Support**:
   1. Active open-source projects, forums, and contributions from Microsoft and developers worldwide.

### Disadvantages of .NET

1. **Resource Intensive**:
   1. Applications may consume more memory and resources compared to lightweight alternatives.
2. **Dependency on Microsoft Ecosystem**:
   1. While .NET Core is cross-platform, certain tools and features (e.g., Visual Studio) are tightly integrated with Microsoft's ecosystem.

# Sql Server

SQL Server is a robust, reliable, and feature-rich database management system suitable for a wide range of applications, from small-scale projects to enterprise-grade solutions. Its integration with Microsoft tools and cloud services, combined with powerful data management and analysis features, makes it a popular choice for businesses and developers.

### Key Features of SQL Server

1. **Relational Database System**:
   1. Organizes data into tables with predefined schemas, making it easy to store, retrieve, and manipulate structured data.
2. **Cross-Platform Support**:
   1. Available on **Windows**, **Linux**, and **Docker**, making it suitable for diverse environments.
3. **High Performance and Scalability**:
   1. Handles large-scale workloads with support for partitioning, indexing, and in-memory processing.
4. **Data Security**:
   1. Provides built-in security features such as **encryption**, **row-level security**, and **dynamic data masking** to protect sensitive data.
5. **Business Intelligence (BI)**:
   1. Includes tools like **SQL Server Reporting Services (SSRS)** and **SQL Server Analysis Services (SSAS)** for data analysis and reporting.
6. **Integration Services**:
   1. **SQL Server Integration Services (SSIS)** allows seamless data integration between different systems.
7. **Cloud Integration**:
   1. Can integrate with cloud services like **Azure SQL Database**, enabling hybrid or fully cloud-based database solutions.
8. **Advanced Analytics**:
   1. Supports **machine learning** and **big data processing** using built-in features and integration with tools like **R** and **Python**.

### Core Components of SQL Server

1. **Database Engine**:
   1. Core service for storing, processing, and securing data.
2. **SQL Server Management Studio (SSMS)**:
   1. A graphical interface for managing and querying databases.
3. **SQL Server Agent**:
   1. Automates administrative tasks like backups and scheduling.
4. **SQL Server Profiler**:
   1. A tool for monitoring and optimizing database performance.
5. **Full-Text Search**:
   1. Enables complex querying of textual data.
6. **Analysis Services (SSAS)**:
   1. Provides tools for OLAP (Online Analytical Processing) and data mining.
7. **Integration Services (SSIS)**:
   1. Facilitates data migration, transformation, and integration between systems.
8. **Reporting Services (SSRS)**:
   1. Allows the creation of interactive, paginated, and mobile reports.

### Use Cases of SQL Server

1. **Business Applications**:
   1. Backends for ERP, CRM, and inventory management systems.
2. **Data Warehousing**:
   1. Centralized data storage for business intelligence and analytics.
3. **E-Commerce**:
   1. Manages product catalogs, user data, and transaction histories.
4. **Cloud Databases**:
   1. Deployment as a cloud-based database using **Azure SQL Database**.
5. **Enterprise Data Management**:
   1. Manages large volumes of transactional data for multinational companies.
6. **Data Analytics**:
   1. Supports OLAP cubes and predictive modeling.

### Advantages of SQL Server

1. **Ease of Use**:
   1. User-friendly interfaces like SSMS make it accessible even to beginners.
2. **Performance**:
   1. Optimized for high-speed data processing and querying.
3. **Security**:
   1. Advanced features protect sensitive data from unauthorized access.
4. **Integration**:
   1. Works well with Microsoft tools like **Azure**, **Power BI**, and **Visual Studio**.
5. **Support**:
   1. Backed by Microsoft with extensive documentation and community forums.

### Disadvantages of SQL Server

1. **Cost**:
   1. Licensing for the Standard and Enterprise editions can be expensive for small businesses.
2. **Platform Dependency**:
   1. Historically, it was primarily Windows-centric, although recent versions support Linux.
3. **Resource Intensive**:
   1. Requires significant server resources for optimal performance in large-scale applications.

# IOS

Swift is a powerful and user-friendly programming language developed by **Apple**. It was first introduced in 2014 and is primarily used for building applications for **iOS, macOS, watchOS, and tvOS**. Swift was designed to be safer, more efficient, and easier to learn compared to its predecessor, Objective-C.

#### **Pros:**

1. **Easy to learn** for beginners due to clean syntax.
2. **High performance** for iOS and macOS applications.
3. Supported by **Apple**, ensuring stability and updates.
4. Strong **community support** and extensive resources.
5. Enables **seamless integration** with existing Objective-C projects.

#### **Cons:**

1. Limited to **Apple's ecosystem** for mobile apps.
2. Not as mature as older languages, so fewer libraries and tools in some areas.
3. Learning curve if transitioning from languages like Java or Python.

# Xcode

**Xcode** is Apple’s official **Integrated Development Environment (IDE)** for developing software for **iOS, macOS, watchOS, and tvOS**. It provides developers with all the tools needed to design, code, test, and debug applications for Apple platforms.

### Why is Xcode Important?

Xcode is essential for developers building apps in Apple’s ecosystem. It integrates closely with Apple’s software development kits (SDKs) and tools, streamlining the development process for apps across Apple platforms.

#### **Pros:**

1. Free to use.
2. Integrated tools for coding, UI design, testing, and debugging.
3. Seamless support for Apple-specific platforms (iOS, macOS, etc.).
4. Extensive documentation and community support.
5. Regular updates from Apple.

#### **Cons:**

1. Only available for macOS (requires a Mac).
2. Can be resource-intensive (requires a powerful computer for smooth performance).
3. May have a steep learning curve for beginners.

# UIKIT

**UIKit** is a **framework provided by Apple** for building and managing the user interfaces (UI) of apps for **iOS, iPadOS, and tvOS**. It offers a comprehensive set of tools and components that developers use to create, manage, and customize the UI and user interactions in their applications.

#### **Pros:**

1. Mature and well-documented framework.
2. Extensive library of pre-built UI components.
3. Fine-grained control over user interface design.
4. Widely used in the industry, with strong community support.

#### **Cons:**

1. Steeper learning curve compared to SwiftUI.
2. Verbose code when compared to declarative frameworks.
3. Manual management of layout constraints can become complex.

# swift package manager

The **Swift Package Manager (SPM)** is a tool for managing the distribution of Swift code, provided by Apple as part of the Swift ecosystem. It simplifies the process of adding, managing, and sharing dependencies in Swift projects, whether you're working on apps, libraries, or server-side Swift.

### Benefits of Swift Package Manager:

1. **Simplifies Dependency Management**
2. No need to manually download or configure libraries.
3. **Native to Swift**

Ensures better integration with Swift and Xcode projects compared to third-party tools like CocoaPods or Carthage.

1. **Open Source**

Actively maintained by the Swift community.

1. **Lightweight and Efficient**

Minimal overhead compared to other dependency managers.

#### **Pros of SPM:**

1. Integrated into Swift and Xcode, no extra tools needed.
2. Lightweight and efficient for managing dependencies.
3. Supports versioning and ensures compatibility.
4. Easy to use with a simple Package.swift file.
5. Works across platforms (macOS, iOS, Linux).

#### **Cons of SPM:**

1. Limited to Swift-only projects (less ideal for mixed-language projects).
2. Fewer third-party libraries compared to CocoaPods or Carthage.
3. Requires knowledge of semantic versioning for dependency management.

# MVVM architecture

**MVVM** stands for **Model-View-ViewModel**, a software architectural pattern used in application development. It helps structure code in a way that separates the user interface (UI) from the business logic, making applications easier to maintain, test, and scale. MVVM is widely used in mobile app development, particularly in **iOS** and **Android**, as well as in other platforms like WPF (Windows Presentation Foundation).

### How MVVM Works:

1. **The View** listens to the **ViewModel** for updates (e.g., via bindings in SwiftUI or delegate methods in UIKit).
2. **The ViewModel** retrieves data from the **Model** and prepares it for display in the **View**.
3. When the user interacts with the **View**, the **ViewModel** processes these actions and updates the **Model** or responds accordingly.

### Benefits of MVVM:

1. **Separation of Concerns**
   1. Clear division between the UI, logic, and data layers makes the code easier to manage.
2. **Improved Testability**
   1. The **ViewModel** and **Model** can be tested independently without relying on the UI.
3. **Code Reusability**
   1. The same **ViewModel** can be used with multiple views if needed.
4. **Easier Maintenance**
   1. Changes to the UI or business logic can be made independently without affecting the other layers.

#### **Pros:**

1. **Separation of Concerns**: Makes the code modular and easier to maintain.
2. **Testable Code**: ViewModel and Model can be tested without the UI.
3. **Reusability**: The ViewModel can often be reused with multiple views.
4. **Scalable**: Ideal for larger projects with complex logic.

#### **Cons:**

1. **Overhead for Simple Apps**: Adds complexity for small projects.
2. **Learning Curve**: Understanding the pattern can be challenging for beginners.
3. **Data Binding Complexity**: Setting up bindings can be tricky with frameworks like Combine or RxSwift.

# Core Data

**Core Data** is a framework provided by Apple to manage the **model layer** of an application. It is used to handle the persistence of data—meaning saving, retrieving, and managing data in a structured way, particularly when working with databases or complex object graphs.

Core Data is commonly used in **iOS**, **macOS**, **watchOS**, and **tvOS** applications to store data in databases or files while providing an abstraction layer to work with objects, rather than directly handling raw data storage and queries.

**Core Data** is a framework provided by Apple to manage the **model layer** of an application. It is used to handle the persistence of data—meaning saving, retrieving, and managing data in a structured way, particularly when working with databases or complex object graphs.

Core Data is commonly used in **iOS**, **macOS**, **watchOS**, and **tvOS** applications to store data in databases or files while providing an abstraction layer to work with objects, rather than directly handling raw data storage and queries.

### Key Features of Core Data:

1. **Object-Relational Mapping (ORM)**  
   Core Data abstracts the database into objects, allowing developers to work with **objects** and **properties** in code, instead of worrying about SQL queries. It provides automatic mapping between your model objects and the underlying data storage.
2. **Data Persistence**  
   Core Data helps save data between app launches. You can store data locally on a device (like in SQLite databases, binary files, or XML) without writing raw SQL.
3. **Managed Object Context**  
   Core Data uses a **managed object context** (MOC) to manage the life cycle of objects. It keeps track of changes, saves them to persistent storage, and handles undo/redo operations.
4. **Data Model**  
   You define a **data model** using **entities** and their **attributes** in the **.xcdatamodeld** file. This visual model is similar to a database schema.
5. **Fetch Requests**  
   You can retrieve data from the database using **fetch requests** that allow you to filter and sort the data.
6. **Relationships Between Entities**  
   Core Data supports **one-to-many**, **many-to-many**, and **one-to-one** relationships between entities, making it useful for complex data models.
7. **Versioning and Migration**  
   It provides built-in support for **data model versioning** and **migration**, which helps handle changes to the data model (like adding new properties or entities) without losing existing data.
8. **Performance Optimizations**  
   Core Data is optimized to handle large datasets efficiently, with features like **faulting** and **lazy loading** to only load data when needed.

### Core Data Components:

1. **NSManagedObject**  
   The base class for model objects that are managed by Core Data. Each entity in your data model is represented as an NSManagedObject.
2. **NSManagedObjectContext (MOC)**  
   A context that manages the objects. It tracks changes to data, saving those changes, and handles undo and redo operations.
3. **NSPersistentContainer**  
   A container that simplifies setting up Core Data stacks by encapsulating the managed object model, context, and persistent store.
4. **NSPersistentStoreCoordinator**  
   A component that manages the storage of your objects. It links the model layer with the underlying database (like SQLite).
5. **NSFetchRequest**  
   A request to fetch data from the persistent store. You can apply filters, sort data, or paginate through results.

**Core Data** is a framework provided by Apple to manage the **model layer** of an application. It is used to handle the persistence of data—meaning saving, retrieving, and managing data in a structured way, particularly when working with databases or complex object graphs.

Core Data is commonly used in **iOS**, **macOS**, **watchOS**, and **tvOS** applications to store data in databases or files while providing an abstraction layer to work with objects, rather than directly handling raw data storage and queries.

### Key Features of Core Data:

1. **Object-Relational Mapping (ORM)**  
   Core Data abstracts the database into objects, allowing developers to work with **objects** and **properties** in code, instead of worrying about SQL queries. It provides automatic mapping between your model objects and the underlying data storage.
2. **Data Persistence**  
   Core Data helps save data between app launches. You can store data locally on a device (like in SQLite databases, binary files, or XML) without writing raw SQL.
3. **Managed Object Context**  
   Core Data uses a **managed object context** (MOC) to manage the life cycle of objects. It keeps track of changes, saves them to persistent storage, and handles undo/redo operations.
4. **Data Model**  
   You define a **data model** using **entities** and their **attributes** in the **.xcdatamodeld** file. This visual model is similar to a database schema.
5. **Fetch Requests**  
   You can retrieve data from the database using **fetch requests** that allow you to filter and sort the data.
6. **Relationships Between Entities**  
   Core Data supports **one-to-many**, **many-to-many**, and **one-to-one** relationships between entities, making it useful for complex data models.
7. **Versioning and Migration**  
   It provides built-in support for **data model versioning** and **migration**, which helps handle changes to the data model (like adding new properties or entities) without losing existing data.
8. **Performance Optimizations**  
   Core Data is optimized to handle large datasets efficiently, with features like **faulting** and **lazy loading** to only load data when needed.

### Core Data Components:

1. **NSManagedObject**  
   The base class for model objects that are managed by Core Data. Each entity in your data model is represented as an NSManagedObject.
2. **NSManagedObjectContext (MOC)**  
   A context that manages the objects. It tracks changes to data, saving those changes, and handles undo and redo operations.
3. **NSPersistentContainer**  
   A container that simplifies setting up Core Data stacks by encapsulating the managed object model, context, and persistent store.
4. **NSPersistentStoreCoordinator**  
   A component that manages the storage of your objects. It links the model layer with the underlying database (like SQLite).
5. **NSFetchRequest**  
   A request to fetch data from the persistent store. You can apply filters, sort data, or paginate through results.

### Example of Using Core Data:

Here’s a simple example of how to use Core Data in an iOS app with a **User** entity.

#### **1. Define the Data Model:**

In your .xcdatamodeld file, you would create an entity called **User** with attributes such as name and email.

#### **2. Code to Save Data:**

swift

Copy code

import CoreData  
import UIKit  
  
func saveUser(name: String, email: String) {  
 let context = (UIApplication.shared.delegate as! AppDelegate).persistentContainer.viewContext  
 let newUser = User(context: context)  
 newUser.name = name  
 newUser.email = email  
   
 do {  
 try context.save()  
 print("User saved successfully!")  
 } catch {  
 print("Error saving user: \(error)")  
 }  
}

#### **3. Code to Fetch Data:**

swift

Copy code

func fetchUsers() -> [User] {  
 let context = (UIApplication.shared.delegate as! AppDelegate).persistentContainer.viewContext  
 let fetchRequest: NSFetchRequest<User> = User.fetchRequest()  
   
 do {  
 let users = try context.fetch(fetchRequest)  
 return users  
 } catch {  
 print("Error fetching users: \(error)")  
 return []  
 }  
}

#### **4. Code to Delete Data:**

swift

Copy code

func deleteUser(user: User) {  
 let context = (UIApplication.shared.delegate as! AppDelegate).persistentContainer.viewContext  
 context.delete(user)  
   
 do {  
 try context.save()  
 print("User deleted successfully!")  
 } catch {  
 print("Error deleting user: \(error)")  
 }  
}

### Benefits of Core Data:

1. **Object-Oriented Approach**  
   Core Data abstracts away the complexity of dealing directly with databases, allowing you to work with familiar object-oriented code.
2. **Built-In Persistence**  
   Core Data handles data persistence for you, meaning you don’t have to manually write code to save and load data from a file or database.
3. **Performance Optimization**  
   Core Data is optimized for working with large datasets, using techniques like lazy loading and faulting to load data only when necessary.
4. **Built-in Data Management Features**  
   Features like undo/redo, relationships, validation, and data migration are built into Core Data, making complex data handling easier.

### Alternatives to Core Data:

* **Realm**: A mobile database solution that's simpler to use and more lightweight than Core Data.
* **SQLite**: A low-level relational database that you can use directly, providing more control but requiring more manual effort.
* **UserDefaults**: A simple way to store small amounts of data, like settings or preferences.

#### **Pros:**

1. **Easy Data Persistence**: Core Data simplifies the process of storing and retrieving data.
2. **Object-Oriented**: You work with objects, not raw data, making it easier to manage.
3. **Built-In Features**: It supports data validation, relationships, and undo/redo operations.

#### **Cons:**

1. **Complex Setup**: Core Data requires significant setup and understanding.
2. **Learning Curve**: For beginners, it can be hard to grasp its concepts and how the stack works.
3. **Overkill for Small Projects**: For simple apps, Core Data may be too complex compared to alternatives like UserDefaults.

# Firebase

**Firebase** is a platform developed by Google that provides a suite of backend services for building and managing mobile and web applications. It offers a variety of tools and services designed to make app development faster and easier by handling common backend tasks like database management, user authentication, and cloud storage.

Firebase is often used for mobile app development on **iOS**, **Android**, and web platforms, and it's known for its ease of integration, real-time capabilities, and scalable infrastructure.

### Key Features of Firebase:

1. **Firebase Realtime Database**
2. A NoSQL cloud database that stores data as JSON and syncs in real-time across all connected clients. It’s particularly useful for apps that require real-time data updates, such as messaging apps or collaborative apps.
   1. **Example**: A chat app that updates messages instantly for all users in a conversation.
3. **Firebase Firestore**

A more advanced, flexible, and scalable NoSQL database compared to the Realtime Database. It stores data in documents and collections and offers more powerful querying and indexing capabilities.

* 1. **Example**: A to-do list app with complex queries and filtering options.

1. **Firebase Authentication**

Provides backend services for easy and secure authentication, supporting login methods like email/password, Google, Facebook, Twitter, and phone number authentication.

* 1. **Example**: A mobile app that allows users to sign in using their Google account.

1. **Firebase Cloud Storage**

A service for storing and serving user-generated content such as photos, videos, and other large files. It’s built to scale automatically and provides security rules for managing access.

* 1. **Example**: A photo-sharing app that stores user-uploaded images.

1. **Firebase Cloud Messaging (FCM)**

Enables sending push notifications and messages to users across different platforms (iOS, Android, and web). It supports both foreground and background messaging.

* 1. **Example**: An e-commerce app sending promotional notifications to users.

1. **Firebase Analytics (Google Analytics for Firebase)**

A free app measurement tool that provides insights into app usage and user engagement. It helps track key events and actions in the app.

* 1. **Example**: Tracking how many users complete a specific in-app purchase or level in a game.

1. **Firebase Functions**

A serverless computing service that lets you run backend code in response to events triggered by Firebase features (like a new user signing up or data being written to the database) or HTTP requests.

* 1. **Example**: Automatically sending a welcome email when a user signs up.

1. **Firebase Hosting**

Provides fast and secure web hosting for your static content (HTML, CSS, JavaScript) with easy deployment and global content delivery network (CDN) support.

* 1. **Example**: Hosting a web app for your mobile app's companion website.

1. **Firebase Test Lab**

A tool for testing your mobile apps across a wide range of real devices hosted in Google’s data centers, helping you identify issues in different environments.

1. **Firebase Performance Monitoring**

Monitors the performance of your app in real time, helping you detect issues like slow app startup, network request failures, and more.

### Benefits of Firebase:

1. **Rapid Development**
2. Firebase abstracts away much of the backend development, so developers can focus on building the app’s features without worrying about infrastructure.
3. **Real-Time Capabilities**

Firebase’s real-time database and Firestore allow for immediate data syncing, making it ideal for apps that require live updates (like chat apps, collaborative apps, etc.).

1. **Scalability**

Firebase services scale automatically to accommodate increasing numbers of users or data volume, so you don’t need to worry about managing servers.

1. **Cross-Platform Support**

Firebase supports iOS, Android, and web apps, providing a unified backend solution for multi-platform projects.

1. **Easy Integration**

Firebase provides simple SDKs and APIs that make it easy to integrate authentication, databases, cloud storage, and more into your app.

### Use Cases:

* **Real-time apps**: Chat apps, live collaboration tools, and social media apps.
* **Data-driven apps**: Apps that require dynamic data handling, like e-commerce apps or fitness tracking.
* **User authentication**: Apps that require simple and secure login mechanisms, including email/password and social media logins.
* **Media-heavy apps**: Apps that need to store and manage media files (e.g., images, videos).

#### **Pros:**

* Easy to integrate with other services.
* High performance with optimized code.
* Great community support.

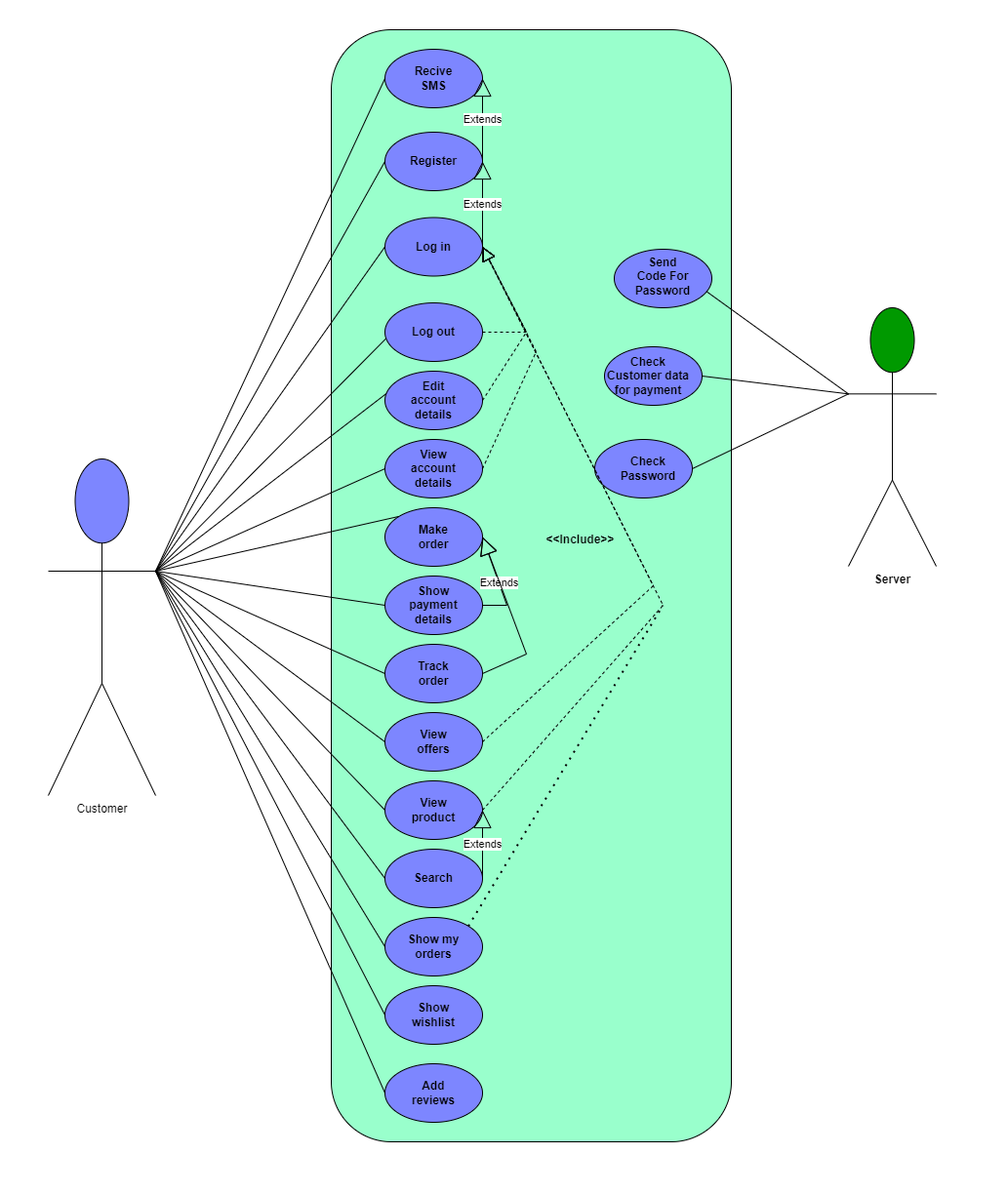
#### **Cons:**

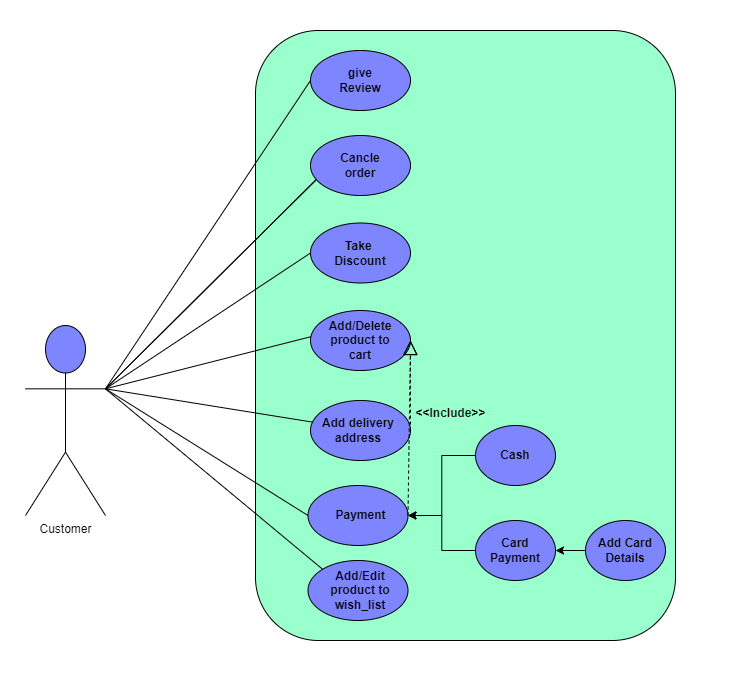
* Limited customization options.
* High learning curve for beginners.
* May not be suitable for large-scale applications.

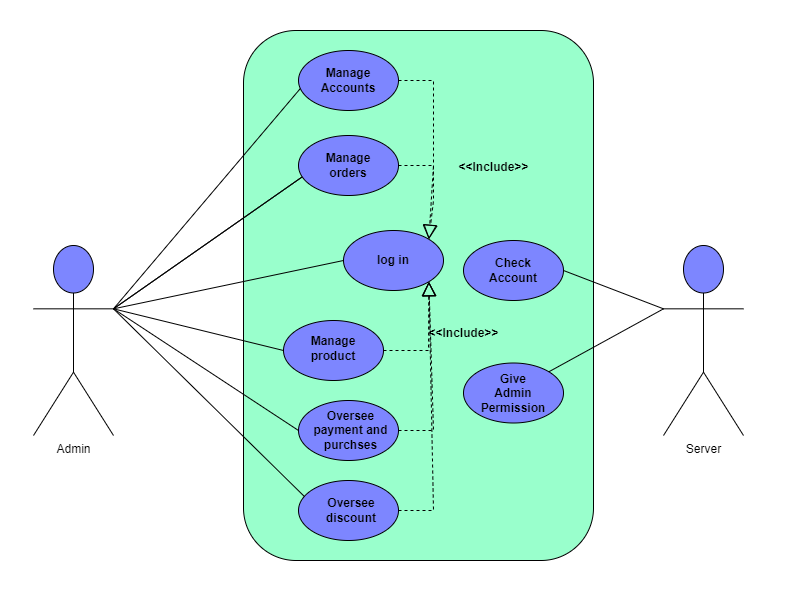
**Chapter 4**

**Proposed System & Methodology**

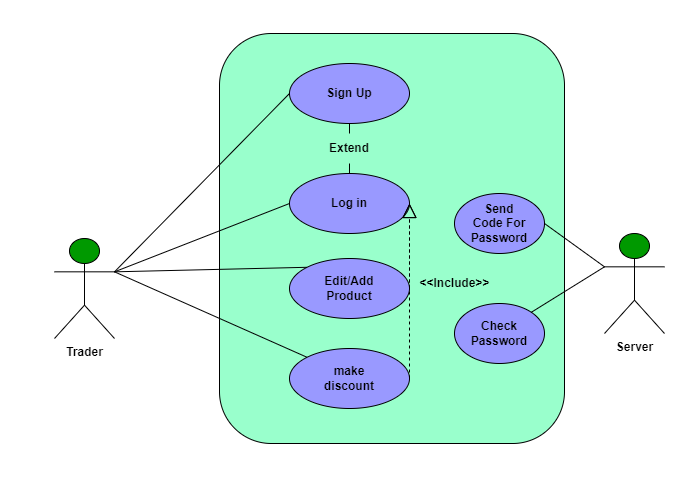
**4.1 System Use-Cases**

**4.1.1: Customer-use-case**



**4.1.2: Admin Use-Case**

**4.1.3: Seller- use-case**



**4.2 Use Case Description (Use case scenario)**

**Admin use case scenarios**

|  |  |
| --- | --- |
| **Use Case name: Administrator Login** | **Unique ID: Handmade\_001** |
| **Area: Authentication and Authorization** | |
| **Actors: System Administrator, E-commerce Backend System, Authentication Server** | |
| **Stakeholders: System Administrator, Business Owners, Security Team** | |
| **Level: Blue** | |
| **Description: Allow a system administrator to log in to the e-commerce platform's administration panel securely** | |
| **Triggering event: The system administrator accesses the login page for the admin panel** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: Administrator navigates to login page.** | **URL for admin panel login page.** |
| **Step2: Administrator enters username/password.** | **Administrator credentials.** |
| **Step3: Submits the login form.** | **Form submission event.** |
| **Step4: System validates credentials.** | **Validation against user database.** |
| **Step5: Administrator is redirected to the admin dashboard.** | **Dashboard URL and session token.** |
| **Extensions (Alternative Flow):in case of Invalid Credentials The system displays an error message: "Invalid username password." Administrator retries login or clicks "Forgot Password."** | |
| **Preconditions: Administrator account exists and is active. The login page is accessible.The authentication server is operational.** | |
| **Postconditions: Success: Administrator is logged in and redirected to the admin dashboard** | |
| **Success Guarantee: The administrator securely accesses the admin panel** | |
| **Minimum Guarantee: in case of failure the administrator receives a descriptive error message.** | |
| **Priority: High** | |
| **Risk: Authentication server failure could prevent login** | |

|  |  |
| --- | --- |
| **Use Case name: Administrator Manage products, orders and accounts.** | **Unique ID: Handmade\_002** |
| **Area: Product and Order and accounts Management** | |
| **Actors: System Administrator, Products & accounts Database, Order Management System.** | |
| **Stakeholders: System Administrator, Business Owners, Customers** | |
| **Level: Blue** | |
| **Description: Allow administrator to manage the e-commerce system by updating product details and processing customer orders and manage accounts.** | |
| **Triggering event: The administrator selects the "Manage Products” or “Manage Orders" or “Manage Accounts”option in the admin panel.** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: Administrator logs into the admin panel.** | **Login credentials** |
| **Step2: Selects "Manage Products" from the dashboard.** | **Product data from the database.** |
| **Step3: Edits a product data** | **Product ID, updated details** |
| **Step4: Selects "Manage Orders" from the dashboard.** | **Order data from the database.** |
| **Step5: Updates an order status** | **Order ID, new status.** |
| **Step6: Selects "Manage Accounts" from the dashboard.** | **Account ID.** |
| **Step7: Manage accounts like giving permissions & access control.** | **Account ID, updated accounts details.** |
| **Step8: Confirms changes and exits the management view** | **product and order and accounts updates.** |
| **Extensions (Alternative Flow): If Administrator enters invalid data System displays an error message: "Invalid input. Please correct the details."** | |
| **Preconditions: The administrator has logged into the admin panel. Product and order and account data is accessible in the database.** | |
| **Postconditions: Updated product details are saved and reflected on the website. Updated order statuses are logged and reflected in the system. Updated accounts details are saved in the database and reflected in the system.** | |
| **Success Guarantee: The system ensures that all updates to products and orders are correctly processed, saved, and reflected in the systems** | |
| **Minimum Guarantee: Product and order data remain unchanged in case of failure.** | |
| **Priority: High** | |
| **Risk: Improper handling of updates could corrupt product or order data.** | |

|  |  |
| --- | --- |
| **Use Case name: oversees payments & purchases & discounts processes.** | **Unique ID: Handmade\_003** |
| **Area: Administration Management System** | |
| **Actors:** **System Admin** | |
| **Stakeholders:** **System Admin, Clients, Seller, Finance and Product Teams.** | |
| **Level:** **Red** | |
| **Description: system administrator oversees and manages the payment, purchase, and discount processes. The system admin ensures that payments are properly processed, purchases are tracked, and discounts are correctly applied to orders.** | |
| **Triggering event:** **The admin logs into the backend system** | |
| **Trigger Type:** **External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: Admin logs into the admin dashboard of the system.** | **Admin credentials** |
| **Step2: Admin navigates to the payment processing section** | **payment methods & status, transaction status & amount.** |
| **Step3:** **Admin reviews payment statuses and identifies any issues** | **payment methods & status, transaction status & amount.** |
| **Step4: Admin oversees purchase activities.** | **Purchased item details (ID, Customer ID, Quantity, Shipping status and so on)** |
| **Step5:** **Admin checks the applied discounts by Sellers.** | **discount codes, The percentage or value of discount** |
| **Step6:** **Admin intervenes in cases of issues** |  |
| **Step7:** **Admin generates reports on payments, purchases, and discounts** |  |
| **Extensions (Alternative Flow):** **If a discount is incorrectly applied, the admin can manually adjust the order, remove the discount, or apply the correct one.** | |
| **Preconditions: The admin has the credentials to access the system, The payment, purchase, and discount systems are functioning and integrated properly, if a payment fails, the admin can manually intervene by either retrying the payment or contacting the payment processor.** | |
| **Postconditions:** **Reports are generated and available for review, The system is updated with any changes made by the admin** | |
| **Success Guarantee:** **The payment, purchase, and discount processes are operating smoothly, and any issues identified by the admin are resolved** | |
| **Minimum Guarantee: if the admin does not resolve every issue, the system captures the remaining issues, and the admin is informed about which tasks require more attention and work on it.** | |
| **Priority:** **High** | |
| **Risk:** **System errors may prevent payments from being processed, purchases from being tracked, or discounts from being applied correctly, improper intervention by the admin could result in financial errors, unauthorized payments, or incorrect application of discounts.** | |

|  |  |
| --- | --- |
| **Use Case name: Sign Up** | **Unique ID: Handmade\_001** |
| **Area: User Registration** | |
| **Actors: User(seller-client)** | |
| **Stakeholders: System Admin** | |
| **Level: Blue** | |
| **Description: allow user to create a new account by providing information for sign up.** | |
| **Triggering event: user fills the sign up form and click the “sign up” button.** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: open the web site** | **Web URL** |
| **Step2: press “Sign Up” link** |  |
| **Step3: Fill the sign up form** | **Name, Email Address, Username (if required), Password** |
| **Step4: Agree terms & conditions** | **Checkbox for user consent** |
| **Step5: Click sign up button to submit the form** | **Form submission with provided data.** |
| **Step6: Verify account via email** |  |
| **Step7: Account is created & added to the Database** |  |
| **Extensions (Alternative Flow): User Entered invalid data, A warning message should appear** | |
| **Preconditions: A device with internet access and a browser** | |
| **Postconditions: a client’s request sent to the server and the account is created successfully.** | |
| **Success Guarantee: user account successfully created** | |
| **Minimum Guarantee: user account successfully created** | |
| **Priority: High** | |
| **Risk: Medium** | |

|  |  |
| --- | --- |
| **Use Case name:** | **Unique ID:** |
| **Area:** | |

|  |  |
| --- | --- |
| **Use Case name: Log in** | **Unique ID: Handmade\_002** |
| **Area: User Authentication** | |
| **Actors: User(seller-client)** | |
| **Stakeholders: System Administrator** | |
| **Level: Blue** | |
| **Description: Allows users to securely log in to the system using their username and password.** | |
| **Triggering event: user press the login link, fill log in form then click “log in” button.** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: user press the login link and get to log in form** | **Web URL** |
| **Step2: User enters his username and password** | **username and password** |
| **Step3: User clicks the "Login" button** | **Login button.** |
| **Step4: System validates username and password against the database.** | **Database containing registered usernames and encrypted passwords** |
| **Step5: If credentials are valid, the user is authenticated.** | **User session is created, and access is granted.** |
| **Step6: System redirects the user to the home/dashboard page.** | **URL of the home page.** |
| **Extensions (Alternative Flow): User Entered invalid data, A warning message should appear** | |
| **Preconditions: The user must have an active account in the system.** | |
| **Postconditions: The user is logged in and has access to their account.** | |
| **Success Guarantee: The user gains access to the system securely.** | |
| **Minimum Guarantee: The user gains access to the system.** | |
| **Priority: High** | |
| **Risk: Medium** | |

|  |  |
| --- | --- |
| **Use Case name: Log Out** | **Unique ID: Handmade\_003** |
| **Area: Account Management** | |
| **Actors: user(seller-client)** | |
| **Stakeholders: user, system admin** | |
| **Level: blue** | |
| **Description: Allow the user to log out of their account.** | |
| **Triggering event: User clicks the "Log Out" button.** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: User clicks the "Log Out" button.** | **Logout button.** |
| **Step2: System terminates the user session.** | **Session management.** |
| **Step3: User is redirected to the login page or homepage** | **Redirection mechanism** |
| **Extensions (Alternative Flow): Fail connect to the sever, Display a warning message says “unable to log out due to poor connection”** | |
| **Preconditions:**  **User is logged in.** | |
| **Postconditions: User session is terminated.** | |
| **Assumptions:** | |
| **Success Guarantee: User session is successfully ended.** | |
| **Minimum Guarantee: User is notified if logout fails.** | |
| **Priority: Low** | |
| **Risk: Low** | |

|  |  |
| --- | --- |
| **Use Case name: View the Website** | **Unique ID: Handmade\_004** |
| **Area: User Interaction** | |
| **Actors: user(client-seller)** | |
| **Stakeholders: Web developers- System Administrator** | |
| **Level: Blue** | |
| **Description: Allows users to access and navigate through the website’s pages to view content or perform actions.** | |
| **Triggering event: The user enters the website URL or clicks on a link that directs them to the website.** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: User enters the website URL or clicks a link to the website.** | **Web URL** |
| **Step2: User views the homepage and its main features (e.g., header, navigation bar, and content).** | **Content loaded on the homepage** |
| **Step3: User interacts with the navigation menu or clickable elements (e.g., buttons, links).** | **Buttons & Links to other pages on the website (e.g., "About Us," "Services").** |
| **Step4: User browses the content or performs specific actions (e.g., search, play videos, read articles).** | **Search queries, form inputs, or multimedia content.** |
| **Extensions (Alternative Flow): The user encounters errors such as page not found, slow loading, display a loading spinner, ask user to reload again** | |
| **Preconditions: The website is accessible via a valid URL.**  **The user has a device with an active internet connection and a compatible browser.** | |
| **Postconditions: The user can view and interact with the website's content or features.** | |
| **Success Guarantee: The user can fully experience the website as intended, without significant issues.** | |
| **Minimum Guarantee: user can view the basic content like text.** | |
| **Priority: High** | |
| **Risk: Medium** | |

|  |  |
| --- | --- |
| **Use Case name: Make an Order** | **Unique ID: Handmade\_005** |
| **Area: E-Commerce / Online Shopping** | |
| **Actors: User(Client)** | |
| **Stakeholders: System Administrator-seller-client** | |
| **Level: Blue** | |
| **Description: Allows a Client to place an order by selecting items, reviewing their cart, and completing payment through the system.** | |
| **Triggering event: The Client clicks on the "Buy Now" or "Add to Cart" button for a product or service.** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: Client browses the product information and selects items** | **Product details, price, quantity, and availability** |
| **Step2: Client adds selected items to the shopping cart.** | **Shopping cart updates with selected items, total cost, and item count.** |
| **Step3: Client views the shopping cart.** | **products, taxes, and shipping charges, Options to edit quantity or remove items.** |
| **Step4: Client proceeds to checkout** | **Checkout button & transition to the checkout page** |
| **Step5: Client provides or confirms shipping information** | **Shipping address.** |
| **Step6: Client selects a payment method.** | **Credit card, digital wallet, bank transfer, or cash on delivery.** |
| **Step7: Client reviews the order summary.** | **Final list of items, total cost, shipping method, and estimated delivery date.** |
| **Step 8: Client click “buy now” button** | **“buy now”button** |
| **Step9: System confirms the order and provides an order summary/confirmation number** | **Order ID- Order tracking information** |
| **Extensions (Alternative Flow): The order is not placed due to issues like payment failure or stock unavailability, Informe user that the order is no longer available, suggest alternatives** | |
| **Preconditions: The Client has an account (if registration is required). Products are in stock and available for ordering. The system and payment gateway are operational.** | |
| **Postconditions: The order is successfully placed, and the Client receives confirmation.** | |
| **Success Guarantee:**  **The Client successfully places the order, and it is logged in the system for fulfillment. (order is created and user proceed to payment)** | |
| **Minimum Guarantee: The Client can view the order summary** | |
| **Priority: High** | |
| **Risk: Medium** | |

|  |  |
| --- | --- |
| **Use Case name: Cancel Order** | **Unique ID: Handmade\_006** |
| **Area: Order Management** | |
| **Actors: Client** | |
| **Stakeholders: User, Vendor, system administrator** | |
| **Level: Blue** | |
| **Description: Allow the user to cancel an order that has been placed, before it is processed or shipped.** | |
| **Triggering event: User decides to cancel an order they have placed.** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: User selects the "Cancel Order" option from the order history page** | **Order details.** |
| **Step2: System checks if the order is eligible for cancellation (e.g., not yet shipped, payment not processed).** | **Order status, shipping status** |
| **Step3: User confirms the cancellation request** | **Cancellation confirmation prompt.** |
| **Step4: System processes the cancellation request and updates the order status to "Cancelled."** | **Order management system.** |
| **Step5: System notifies the user that the order has been successfully cancelled.** | **: Notification system (email/SMS).** |
| **Extensions (Alternative Flow):**  **Handling scenarios where the order is partially shipped or cannot be cancelled, informe the user that order can’t be cancelled now** | |
| **Preconditions: User has placed an order.** | |
| **Postconditions: Order status is updated to "Cancelled."If applicable, the user receives a refund.** | |
| **Success Guarantee: The order is successfully cancelled, and the user is refunded (if applicable).** | |
| **Minimum Guarantee:**  **User is notified if the order cannot be cancelled, with clear reasons.** | |
| **Priority: High** | |
| **Risk: Medium** | |

|  |  |
| --- | --- |
| **Use Case name: Show Cart Items** | **Unique ID: Handmade\_007** |
| **Area: Shopping** | |
| **Actors: User (Client)** | |
| **Stakeholders: User** | |
| **Level: Blue** | |
| **Description: Allow the user to view the items they have added to their shopping cart.** | |
| **Triggering event: User clicks on the "Cart" icon or accesses the cart page from the navigation menu.** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: 1. User clicks on the "Cart" icon or navigates to the "Cart" page from the menu.** | **Cart icon, user interface** |
| **Step2: System retrieves the current cart items from the user's session or account.** | **Cart data, session data.** |
| **Step3: System displays the list of items in the cart, showing details such as product name, price, quantity, and total cost.** | **Cart items data, pricing details.** |
| **Step4: . User can modify the cart by changing the quantity of items or removing items from the cart.** | **Cart modification options, product database.** |
| **Step5:The system recalculates the total cost based on the updated cart and displays the new total.** | **Pricing calculation, cart modification.** |
| **Step6: User can proceed to checkout or continue shopping.** | **Checkout options, navigation buttons** |
| **Extensions (Alternative Flow):cart item not loading, Display error message asks user to refresh the page.** | |
| **Preconditions: User is logged into their account, User has items in their cart.** | |
| **Postconditions: Cart items are displayed, and the user can modify or proceed with the checkout process.** | |
| **Assumptions:** | |
| **Success Guarantee: The cart items are accurately displayed and updated.** | |
| **Minimum Guarantee: If the cart is empty, the system will inform the user and prompt them to add items to the cart.** | |
| **Priority: Medium** | |
| **Risk: Low** | |

|  |  |
| --- | --- |
| **Use Case name: Add Items to Cart** | **Unique ID: Handmade\_008** |
| **Area: Shopping** | |
| **Actors: User(Client)** | |
| **Stakeholders: User, Vendor, Payment Provider** | |
| **Level:Blue** | |
| **Description: Description: Allow the user to add items to their shopping cart for later purchase.** | |
| **Triggering event: User clicks on the "Add to Cart" button for a product on the product detail page.** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: 1. Usersearches for a specific item.** | **search bar.** |
| **Step2: User selects a product they want to add to the cart, viewing the product details (e.g., price, size, color).** | **Product detail page, user selection** |
| **Step3: User clicks on the "Add to Cart" button.** | **Button click event, cart data.** |
| **Step4: System checks if the item is available in the desired quantity.** | **Product stock information.** |
| **Step5: System adds the selected item to the user's cart, updating the cart's total price and item count.** | **Cart data, item details, total price calculation.** |
| **Step6: The system displays a confirmation message (e.g., "Item added to cart")** | **Notification system, cart UI update.** |
| **Step7: User can continue browsing or proceed to view their cart.** | **Navigation options.** |
| **Extensions (Alternative Flow): Out of stock error, Provide links to similar products** | |
| **Preconditions: User is browsing the website or app.** | |
| **Postconditions: The selected item is added to the user's cart.** | |
| **Success Guarantee: The item is successfully added to the cart, and the user is notified.** | |
| **Minimum Guarantee: If the item is out of stock or cannot be added for other reasons, the system will notify the user with a clear message.** | |
| **Priority: High** | |
| **Risk:Low** | |

|  |  |
| --- | --- |
| **Use Case name: Add Review** | **Unique ID: Handmade\_009** |
| **Area: Product Review** | |
| **Actors: User(Client)** | |
| **Stakeholders: User, Vendor, Other users** | |
| **Level: Blue** | |
| **Description: Allow the user to leave a review for a product they have purchased.** | |
| **Triggering event: User clicks on the "Add Review" button on the product detail page.** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: User navigates to the product detail page of the item they purchased.** | **Product page, user selection.** |
| **Step2: . User clicks on the "Add Review" button.** | **Button click event, product review section** |
| **Step3: System checks if the user is logged in and has purchased the product (if applicable).** | **User login status, order history data** |
| **Step4: System displays a review form with fields for the star rating, written comments, and any other required information (e.g., pros, cons, etc.).** | **Review form fields** |
| **Step5: User enters their review, rating the product and writing comments.** | **Review input data.** |
| **Step6: . User submits the review by clicking the "Submit Review" button** | **Review submission button, data validation.** |
| **Step7: System validates the review for required fields (rating, comment, etc.) and submits the review to the product page.** | **Review validation, database storage.** |
| **Step8: System displays a confirmation message ("Thank you for your review") and updates the product page with the new review.** | **Notification system, review display.** |
| **Extensions (Alternative Flow):connection error, ask user to remake the review.** | |
| **Preconditions: User is logged into their account, User has purchased the product** | |
| **Postconditions: The review is stored in the database and displayed on the product page. Other users can view the review.** | |
| **Success Guarantee: The review is successfully submitted and visible to other users.** | |
| **Minimum Guarantee: If the user cannot submit a review (due to not purchasing the item or other restrictions), the system will notify them with an appropriate message.** | |
| **Priority: Medium** | |
| **Risk: Low** | |

|  |  |
| --- | --- |
| **Use Case name: : Client Returns Order** | **Unique ID: Handmade\_010** |
| **Area: Order Management System** | |
| **Actors: Client, Delivery Personnel** | |
| **Stakeholders: Client Support Team, Financial Department** | |
| **Level: Blue** | |
| **Description: Allow client to return purchased order by performing specific actions** | |
| **Triggering event: Clients do a return request** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: Client submits return request:** | **Order information, reason for return, item condition, and preferred resolution (refund/replacement).** |
| **Step2: System validates return eligibility** | **Purchase date, product return policy, condition of item** |
| **Step3: System accept a return request** | **Shipping address.** |
| **Step4: 4. Client ships item back:** | **Package tracking updates** |
| **Step5: Warehouse receives and inspects returned item** | **Condition of the returned item, matching return request details.** |
| **Step6: System processes refund or replacement:** | **Refund amount or replacement order details.** |
| **Step7: Client notified of return completion.** | **Email or SMS with refund/replacement confirmation.** |
| **Extensions (Alternative Flow): If the item is damaged or missing parts, notify the Client for clarification & reason.** | |
| **Preconditions: The order was successfully placed and delivered. The product is within the returnable time frame. The Client has an account on the platform** | |
| **Postconditions: The Client is refunded or receives a replacement. returned item is recorded in the inventory system** | |
| **Success Guarantee: The return request is processed smoothly, and the Client receives the requested resolution.** | |
| **Minimum Guarantee: The return request is logged, and the Client is informed of the resolution steps** | |
| **Priority: High** | |
| **Risk: The returned product may be damaged or incomplete.** | |

|  |  |
| --- | --- |
| **Use Case name: Client Gives Feedback** | **Unique ID: Handmade\_011** |
| **Area: Client Feedback Management System** | |
| **Actors: Client** | |
| **Stakeholders: Client, Product Development Team,Seller.** | |
| **Level: Blue** | |
| **Description: allow Client to submit feedback about their experience, which can be used to enhance products or services.** | |
| **Triggering event: Client chooses to provide feedback** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: Client accesses feedback submission section** | **Login information** |
| **Step2: Client fills feedback form or writes a review:** | **comments, optional rating** |
| **Step3: System records the feedback:** | **Client details.** |
| **Step4: System sends confirmation to the Client** | **message confirming accept of feedback** |
| **Step5: Relevant team reviews the feedback.** | **Feedback content and required action.** |
| **Step6: System display feedback and rating** | **Feedback & rating & UI** |
| **Extensions (Alternative Flow): If feedback is critical (e.g., related to safety or severe complaints), escalate directly to the responsible team for immediate action.** | |
| **Preconditions: Feedback submission section is functional and accessible.**  **The Client has a valid concern, suggestion, or comment to share.** | |
| **Postconditions: Feedback is recorded and appropriately routed to the concerned team.** | |
| **Success Guarantee: Feedback is successfully captured and acted upon (if required), improving Client satisfaction.** | |
| **Minimum Guarantee: Feedback is logged, and displayed on website** | |
| **Priority: Medium** | |
| **Risk: Feedback may be incomplete or unclear.** | |

|  |  |
| --- | --- |
| **Use Case name: Add Delivery Information** | **Unique ID: : Handmade\_012** |
| **Area: Order Management System** | |
| **Actors: Client** | |
| **Stakeholders: Client Support team, System Administrator, delivery ppersonnel** | |
| **Level: Blue** | |
| **Description: This use case describes the process for a Client to add or update delivery information for an order** | |
| **Triggering event: Client proceeds to add or edits an existing order’s delivery details.** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: Client accesses the delivery information form.** | **Login information** |
| **Step2: Client enters delivery details** | **delivery details (e. g Name, address, phone number.)** |
| **Step3: 3. System check validation of delivery information:** | **delivery details format (e. g phone number validity.)** |
| **Step4: Client confirms and saves delivery information** | **review page showing the entered details and a Confirmation button** |
| **Step5: System updates order with delivery details** | **Order ID linked to the provided delivery information** |
| **Extensions (Alternative Flow): If the system detects invalid delivery information, notify the Client with specific error messages and allow corrections.** | |
| **Preconditions: Client has selected items for purchase, the delivery information form is functional and accessible.** | |
| **Postconditions: Delivery information is successfully added to the order.** | |
| **Success Guarantee: Delivery information is recorded accurately, ensuring a smooth delivery process.** | |
| **Minimum Guarantee: Delivery information is saved but may require Client support intervention if incomplete or incorrect.** | |
| **Priority: High** | |
| **Risk: Client may enter incorrect or incomplete information** | |

|  |  |
| --- | --- |
| **Use Case name: View Returns** | **Unique ID: Handmade\_013** |
| **Area: Order Management System** | |
| **Actors: Client** | |
| **Stakeholders: Client Support Team** | |
| **Level: Blue** | |
| **Description: Allow Client views details and status of their return requests.** | |
| **Triggering event: Client logs into their account and navigates to the returns section.** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: Client accesses the returns section** | **Login information** |
| **Step2: System displays the list of return requests** | **return requests information** |
| **Step3: Client selects a specific return request** | **Order ID, product details** |
| **Step4: System displays detailed return information** | **Return status (e.g., "Pending," "Under Review," "Completed"), refund/replacement details, and any communication history** |
| **Step5: Client views additional details if available** | **Product detailed information** |
| **Extensions (Alternative Flow): If the system cannot retrieve detailed return information, display a message and offer Client support contact options.** | |
| **Preconditions: The Client has initiated at least one return request. The system has recorded the return details and status** | |
| **Postconditions: The Client successfully views the requested return details and status.** | |
| **Success Guarantee: The Client can view return status and details.** | |
| **Minimum Guarantee: The system displays basic information about the return request, but additional details may require Client support intervention** | |
| **Priority: Medium** | |
| **Risk: The system might fail to retrieve or display the return information due to technical issues.** | |

|  |  |
| --- | --- |
| **Use Case name: Do Payments** | **Unique ID: Handmade\_014** |
| **Area: Payment Processing System** | |
| **Actors: Client, Bank** | |
| **Stakeholders: Bank, Sales and Finance Teams, Payment Gateway** | |
| **Level: Red** | |
| **Description: Allow Client to make a payment for an order including entering payment information, processing the transaction, and receiving confirmation** | |
| **Triggering event: Client make a purchase or payment for a product** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: Client accesses the payment page** | **Login information** |
| **Step2: Client enters payment details** | **Payment method, card number, expiration date, CVV, etc.** |
| **Step3: System validates payment information** | **Payment gateway checks for valid card details** |
| **Step4: System sends payment request to payment gateway** | **Payment amount, Client information, payment method details** |
| **Step5: Payment gateway processes the payment:** | **Bank authorization** |
| **Step6: System receives payment confirmation or failure notification** | **Success/failure message, transaction ID, payment status** |
| **Step7: System updates order status and sends confirmation** | **Order ID, transaction ID, payment receipt, and email/SMS notification to Client** |
| **Extensions (Alternative Flow): If the system detects invalid payment information (e.g., incorrect card number or expiration date), ask the Client to correct the details.** | |
| **Preconditions: The Client has selected the product and is ready to make a payment. The Client has an active payment method. The payment gateway and system are functional and connected to the bank for transaction processing** | |
| **Postconditions: Payment is processed successfully and the order status is updated Client receives a payment confirmation** | |
| **Success Guarantee: Payment is successfully processed, and the Client receives a confirmation.** | |
| **Minimum Guarantee: If the payment fails, the system will notify the Client with a reason and provide options for retrying or using an alternative payment method** | |
| **Priority: Red** | |
| **Risk: Payment details may be entered incorrectly, leading to transaction failure** | |

|  |  |
| --- | --- |
| **Use Case name: Add to Favorites** | **Unique ID: Handmade\_015** |
| **Area: Product Management System** | |
| **Actors: Client** | |
| **Stakeholders: Marketing Team, seller** | |
| **Level: Green** | |
| **Description:Allow Client to add a product to his Favorites list for easy access in the future** | |
| **Triggering event: Client clicks the "Add to Favorites " button on a product page** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: Client browses products** | **Web URL** |
| **Step2: Client clicks "Add to Favorites" on a product** | **Client information login &"Add to Favorites " button** |
| **Step3: System checks if the product is in the Client's Favorites list** | **Favorites list content** |
| **Step4: System adds product to the Favorites list if not already added** | **Product details (ID, name, price)** |
| **Step5: System displays confirmation or update on the UI** | **Confirmation button or text, updated UI** |
| **Step6: Client can view updated Favorites list** | **Favorites list** |
| **Extensions (Alternative Flow): If the product is already in the Favorites list, the system will display a message like “Product is already in your Favorites” instead of adding it again.** | |
| **Preconditions: Client is logged into their account. The product is available in the collection and can be added to Favorites .** | |
| **Postconditions: The product is added to the Client’s Favorites list in their profile. The Client can view or remove the product from their Favorites list at any time.** | |
| **Success Guarantee: Product is successfully added to the Favorites list.** | |
| **Minimum Guarantee: The system will notify the Client that the product has been added, but the system may not immediately reflect changes in the user interface due to temporary issues** | |
| **Priority: Green** | |
| **Risk: The system may fail to update the Favorites list due to server or database errors.** | |

|  |  |
| --- | --- |
| **Use Case name: Sign Up** | **Unique ID: Handmade\_001** |
| **Area: User Registeration** | |
| **Actors: User(seller-client)** | |
| **Stakeholders: System Admin** | |
| **Level: Blue** | |
| **Description: allow user to create a new account by providing information for sign up.** | |
| **Triggering event: user fills the sign up form and click the “sign up” button.** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: open the web site** | **Web URL** |
| **Step2: press “Sign Up” link** |  |
| **Step3: Fill the sign up form** | **Name, Email Address, Username (if required), Password** |
| **Step4: Agree terms & conditions** | **Checkbox for user consent** |
| **Step5: Click sign up button to submit the form** | **Form submission with provided data.** |
| **Step6: Verify account via email** |  |
| **Step7: Account is created & added to the Database** |  |
| **Extensions (Alternative Flow): User Entered invalid data, A warning message should appear** | |
| **Preconditions: A device with internet access and a browser** | |
| **Postconditions: a client’s request sent to the server and the account is created successfully.** | |
| **Success Guratee: user account successfully created** | |
| **Minimum Gurantee: user account successfully created** | |
| **Priority: High** | |
| **Risk: Medium** | |

|  |  |
| --- | --- |
| **Use Case name:** | **Unique ID:** |
| **Area:** | |

|  |  |
| --- | --- |
| **Use Case name: Log in** | **Unique ID: Handmade\_002** |
| **Area: User Authentication** | |
| **Actors: User(seller-client)** | |
| **Stakeholders: System Administrator** | |
| **Level: Blue** | |
| **Description: Allows users to securely log in to the system using their username and password.** | |
| **Triggering event: user press the login link, fill log in form then click “log in” button.** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: user press the login link and get to log in form** | **Web URL** |
| **Step2: User enters his username and password** | **username and password** |
| **Step3: User clicks the "Login" button** | **Login button.** |
| **Step4: System validates username and password against the database.** | **Database containing registered usernames and encrypted passwords** |
| **Step5: If credentials are valid, the user is authenticated.** | **User session is created, and access is granted.** |
| **Step6: System redirects the user to the home/dashboard page.** | **URL of the home page.** |
| **Extensions (Alternative Flow): User Entered invalid data, A warning message should appear** | |
| **Preconditions: The user must have an active account in the system.** | |
| **Postconditions: The user is logged in and has access to their account.** | |
| **Success Guratee: The user gains access to the system securely.** | |
| **Minimum Gurantee: The user gains access to the system.** | |
| **Priority: High** | |
| **Risk: Medium** | |

|  |  |
| --- | --- |
| **Use Case name: Log Out** | **Unique ID: Handmade\_003** |
| **Area: Account Management** | |
| **Actors:user(seller-client)** | |
| **Stakeholders: user, system admin** | |
| **Level: blue** | |
| **Description: Allow the user to log out of their account.** | |
| **Triggering event: User clicks the "Log Out" button.** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: User clicks the "Log Out" button.** | **Logout button.** |
| **Step2: System terminates the user session.** | **Session management.** |
| **Step3: User is redirected to the login page or homepage** | **Redirection mechanism** |
| **Extensions (Alternative Flow): Fail connect to the sever, Display a warning message says “unable to log out due to poor connection”** | |
| **Preconditions:**  **User is logged in.** | |
| **Postconditions: User session is terminated.** | |
| **Assumptions:** | |
| **Success Guratee: User session is successfully ended.** | |
| **Minimum Gurantee: User is notified if logout fails.** | |
| **Priority: Low** | |
| **Risk: Low** | |

|  |  |
| --- | --- |
| **Use Case name: View the Website** | **Unique ID: Handmade\_004** |
| **Area: User Interaction** | |
| **Actors: user(client-seller)** | |
| **Stakeholders: Web developers- System Administrator** | |
| **Level: Blue** | |
| **Description: Allows users to access and navigate through the website’s pages to view content or perform actions.** | |
| **Triggering event: The user enters the website URL or clicks on a link that directs them to the website.** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: User enters the website URL or clicks a link to the website.** | **Web URL** |
| **Step2: User views the homepage and its main features (e.g., header, navigation bar, and content).** | **Content loaded on the homepage** |
| **Step3: User interacts with the navigation menu or clickable elements (e.g., buttons, links).** | **Buttons & Links to other pages on the website (e.g., "About Us," "Services").** |
| **Step4: User browses the content or performs specific actions (e.g., search, play videos, read articles).** | **Search queries, form inputs, or multimedia content.** |
| **Extensions (Alternative Flow): The user encounters errors such as page not found, slow loading, display a loading spinner, ask user to reload again** | |
| **Preconditions: The website is accessible via a valid URL.**  **The user has a device with an active internet connection and a compatible browser.** | |
| **Postconditions: The user can view and interact with the website's content or features.** | |
| **Success Guratee: The user can fully experience the website as intended, without significant issues.** | |
| **Minimum Gurantee: user can view the basic content like text.** | |
| **Priority: High** | |
| **Risk: Medium** | |

|  |  |
| --- | --- |
| **Use Case name: Make an Order** | **Unique ID: Handmade\_005** |
| **Area: E-Commerce / Online Shopping** | |
| **Actors: User(Client)** | |
| **Stakeholders: System Administrator-seller-client** | |
| **Level: Blue** | |
| **Description: Allows a customer to place an order by selecting items, reviewing their cart, and completing payment through the system.** | |
| **Triggering event: The customer clicks on the "Buy Now" or "Add to Cart" button for a product or service.** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: Customer browses the product information and selects items** | **Product details, price, quantity, and availability** |
| **Step2: Customer adds selected items to the shopping cart.** | **Shopping cart updates with selected items, total cost, and item count.** |
| **Step3: Customer views the shopping cart.** | **products, taxes, and shipping charges, Options to edit quantity or remove items.** |
| **Step4: Customer proceeds to checkout** | **Checkout button & transition to the checkout page** |
| **Step5: Customer provides or confirms shipping information** | **Shipping address.** |
| **Step6: Customer selects a payment method.** | **Credit card, digital wallet, bank transfer, or cash on delivery.** |
| **Step7: Customer reviews the order summary.** | **Final list of items, total cost, shipping method, and estimated delivery date.** |
| **Step 8: customer click “buy now” button** | **“buy now”button** |
| **Step9: System confirms the order and provides an order summary/confirmation number** | **Order ID- Order tracking information** |
| **Extensions (Alternative Flow): The order is not placed due to issues like payment failure or stock unavailability, Informe user that the order is no longer available, suggest alternatives** | |
| **Preconditions: The customer has an account (if registration is required). Products are in stock and available for ordering. The system and payment gateway are operational.** | |
| **Postconditions: The order is successfully placed, and the customer receives confirmation.** | |
| **Success Guratee:**  **The customer successfully places the order, and it is logged in the system for fulfillment. (order is created and user proceed to payment)** | |
| **Minimum Gurantee : The customer can view the order summary** | |
| **Priority: High** | |
| **Risk: Medium** | |

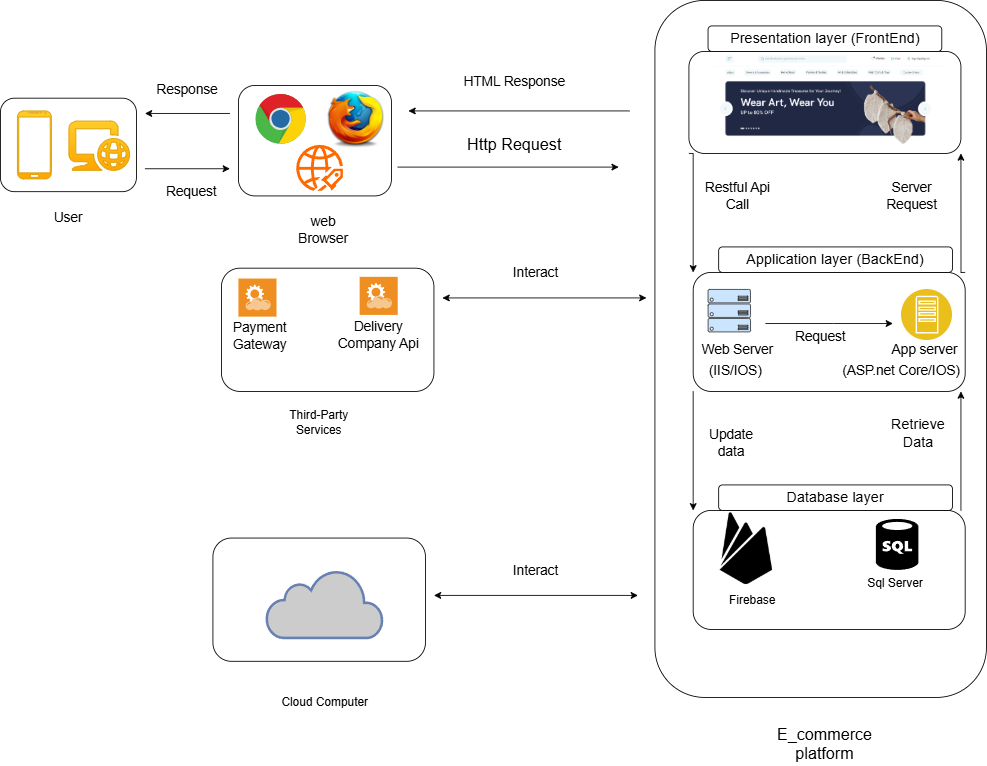
|  |  |
| --- | --- |
| **Use Case name: Cancel Order** | **Unique ID: Handmade\_006** |
| **Area: Order Management** | |
| **Actors: customer** | |
| **Stakeholders: User, Vendor, system administrator** | |
| **Level: Blue** | |
| **Description: Allow the user to cancel an order that has been placed, before it is processed or shipped.** | |
| **Triggering event: User decides to cancel an order they have placed.** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: User selects the "Cancel Order" option from the order history page** | **Order details.** |
| **Step2: System checks if the order is eligible for cancellation (e.g., not yet shipped, payment not processed).** | **Order status, shipping status** |
| **Step3: User confirms the cancellation request** | **Cancellation confirmation prompt.** |
| **Step4: System processes the cancellation request and updates the order status to "Cancelled."** | **Order management system.** |
| **Step5: System notifies the user that the order has been successfully cancelled.** | **: Notification system (email/SMS).** |
| **Extensions (Alternative Flow):**  **Handling scenarios where the order is partially shipped or cannot be cancelled, informe the user that order can’t be cancelled now** | |
| **Preconditions: User has placed an order.** | |
| **Postconditions: Order status is updated to "Cancelled."If applicable, the user receives a refund.** | |
| **Success Guratee: The order is successfully cancelled, and the user is refunded (if applicable).** | |
| **Minimum Gurantee:**  **User is notified if the order cannot be cancelled, with clear reasons.** | |
| **Priority: High** | |
| **Risk: Medium** | |

|  |  |
| --- | --- |
| **Use Case name: Show Cart Items** | **Unique ID: Handmade\_007** |
| **Area: Shopping** | |
| **Actors: User (Client)** | |
| **Stakeholders: User** | |
| **Level: Blue** | |
| **Description: Allow the user to view the items they have added to their shopping cart.** | |
| **Triggering event: User clicks on the "Cart" icon or accesses the cart page from the navigation menu.** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: 1. User clicks on the "Cart" icon or navigates to the "Cart" page from the menu.** | **Cart icon, user interface** |
| **Step2: System retrieves the current cart items from the user's session or account.** | **Cart data, session data.** |
| **Step3: System displays the list of items in the cart, showing details such as product name, price, quantity, and total cost.** | **Cart items data, pricing details.** |
| **Step4: . User can modify the cart by changing the quantity of items or removing items from the cart.** | **Cart modification options, product database.** |
| **Step5:The system recalculates the total cost based on the updated cart and displays the new total.** | **Pricing calculation, cart modification.** |
| **Step6: User can proceed to checkout or continue shopping.** | **Checkout options, navigation buttons** |
| **Extensions (Alternative Flow):cart item not loading, Display error message asks user to refresh the page.** | |
| **Preconditions: User is logged into their account, User has items in their cart.** | |
| **Postconditions: Cart items are displayed, and the user can modify or proceed with the checkout process.** | |
| **Assumptions:** | |
| **Success Guratee: The cart items are accurately displayed and updated.** | |
| **Minimum Gurantee: If the cart is empty, the system will inform the user and prompt them to add items to the cart.** | |
| **Priority: Medium** | |
| **Risk: Low** | |

|  |  |
| --- | --- |
| **Use Case name: Add Items to Cart** | **Unique ID: Handmade\_008** |
| **Area: Shopping** | |
| **Actors: User(Client)** | |
| **Stakeholders: User, Vendor, Payment Provider** | |
| **Level:Blue** | |
| **Description: Description: Allow the user to add items to their shopping cart for later purchase.** | |
| **Triggering event: User clicks on the "Add to Cart" button for a product on the product detail page.** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: 1. Usersearches for a specific item.** | **search bar.** |
| **Step2: User selects a product they want to add to the cart, viewing the product details (e.g., price, size, color).** | **Product detail page, user selection** |
| **Step3: User clicks on the "Add to Cart" button.** | **Button click event, cart data.** |
| **Step4: System checks if the item is available in the desired quantity.** | **Product stock information.** |
| **Step5: System adds the selected item to the user's cart, updating the cart's total price and item count.** | **Cart data, item details, total price calculation.** |
| **Step6: The system displays a confirmation message (e.g., "Item added to cart")** | **Notification system, cart UI update.** |
| **Step7: User can continue browsing or proceed to view their cart.** | **Navigation options.** |
| **Extensions (Alternative Flow): Out of stock error, Provide links to similar products** | |
| **Preconditions: User is browsing the website or app.** | |
| **Postconditions: The selected item is added to the user's cart.** | |
| **Success Guratee: The item is successfully added to the cart, and the user is notified.** | |
| **Minimum Gurantee: If the item is out of stock or cannot be added for other reasons, the system will notify the user with a clear message.** | |
| **Priority: High** | |
| **Risk:Low** | |

|  |  |
| --- | --- |
| **Use Case name: Add Review** | **Unique ID: Handmade\_009** |
| **Area: Product Review** | |
| **Actors: User(Client)** | |
| **Stakeholders: User, Vendor, Other users** | |
| **Level: Blue** | |
| **Description:**  **Description: Allow the user to leave a review for a product they have purchased.** | |
| **Triggering event: User clicks on the "Add Review" button on the product detail page.** | |
| **Trigger Type: External** | |
| **Steps performed:** | **Information for steps:** |
| **Step1: User navigates to the product detail page of the item they purchased.** | **Product page, user selection.** |
| **Step2: . User clicks on the "Add Review" button.** | **Button click event, product review section** |
| **Step3: System checks if the user is logged in and has purchased the product (if applicable).** | **User login status, order history data** |
| **Step4: System displays a review form with fields for the star rating, written comments, and any other required information (e.g., pros, cons, etc.).** | **Review form fields** |
| **Step5: User enters their review, rating the product and writing comments.** | **Review input data.** |
| **Step6: . User submits the review by clicking the "Submit Review" button** | **Review submission button, data validation.** |
| **Step7: System validates the review for required fields (rating, comment, etc.) and submits the review to the product page.** | **Review validation, database storage.** |
| **Step8: System displays a confirmation message ("Thank you for your review") and updates the product page with the new review.** | **Notification system, review display.** |
| **Extensions (Alternative Flow):connection error, ask user to remake the review.** | |
| **Preconditions: User is logged into their account, User has purchased the product** | |
| **Postconditions: The review is stored in the database and displayed on the product page. Other users can view the review.** | |
| **Success Guratee: The review is successfully submitted and visible to other users.** | |
| **Minimum Gurantee: If the user cannot submit a review (due to not purchasing the item or other restrictions), the system will notify them with an appropriate message.** | |
| **Priority: Medium** | |
| **Risk: Low** | |

**4.3 System Architecture**



**4.4 Analysis Class**

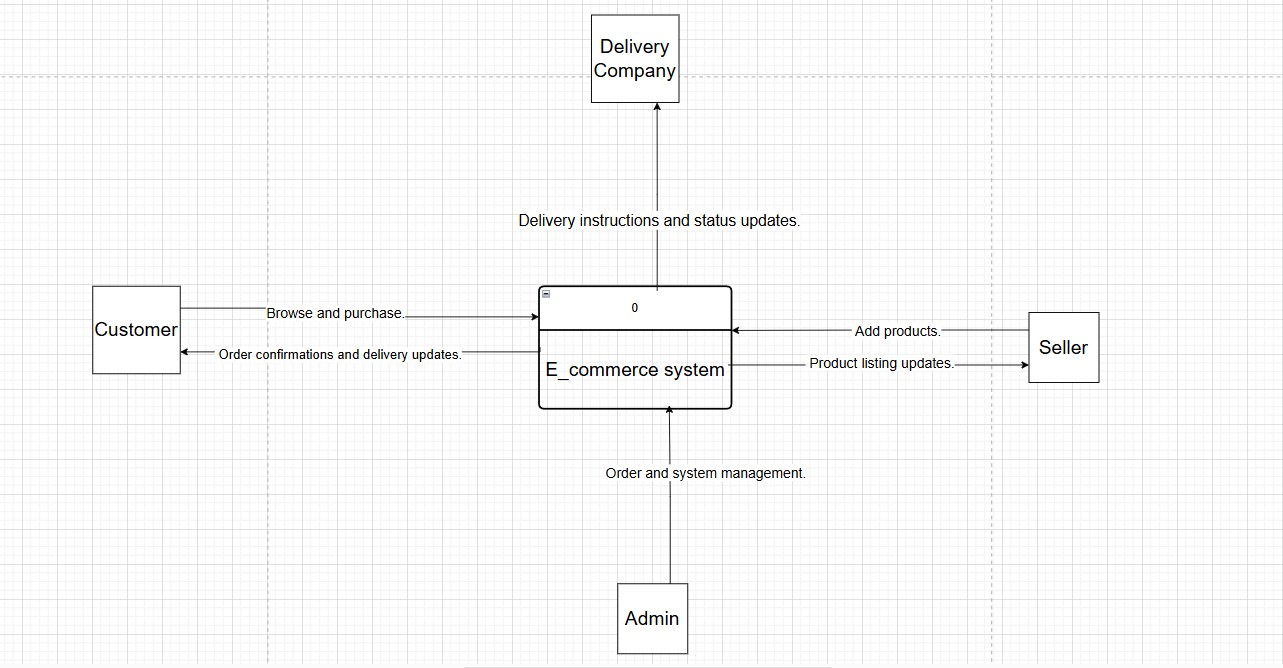
**4.4.1 State Diagram**

**4.4.1.1 State for Client**

**4.4.1.2 State for Admin**

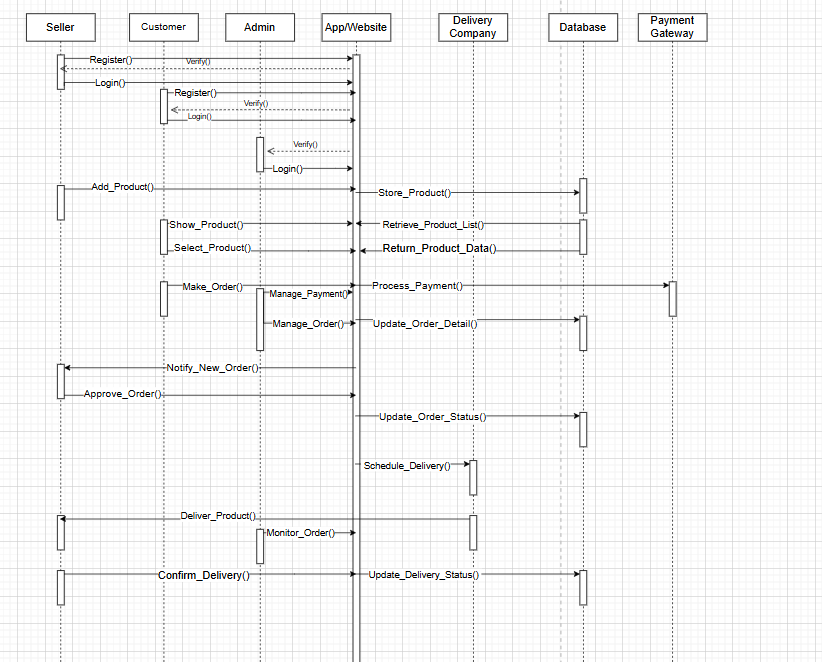
**4****.4.2 Data flow diagram (Level Zero diagram)**

**4****.4.3: Context Diagram**



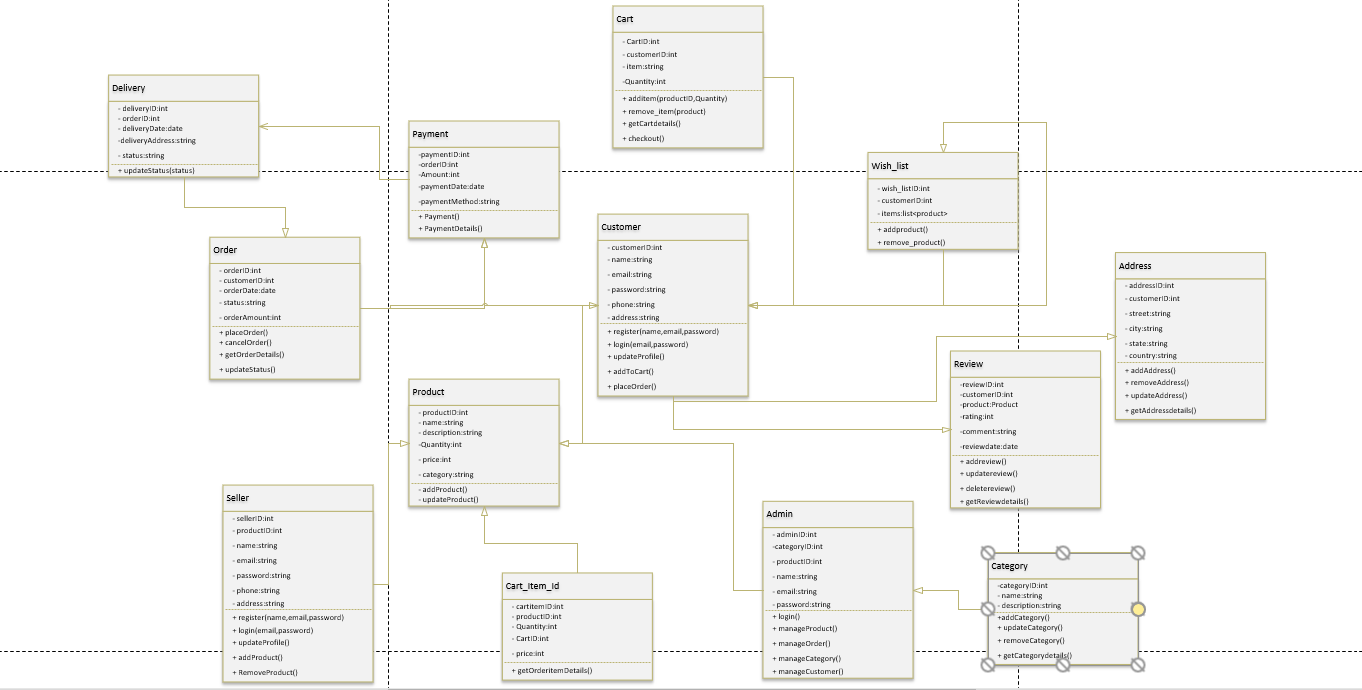
**4.5 interaction class diagram**

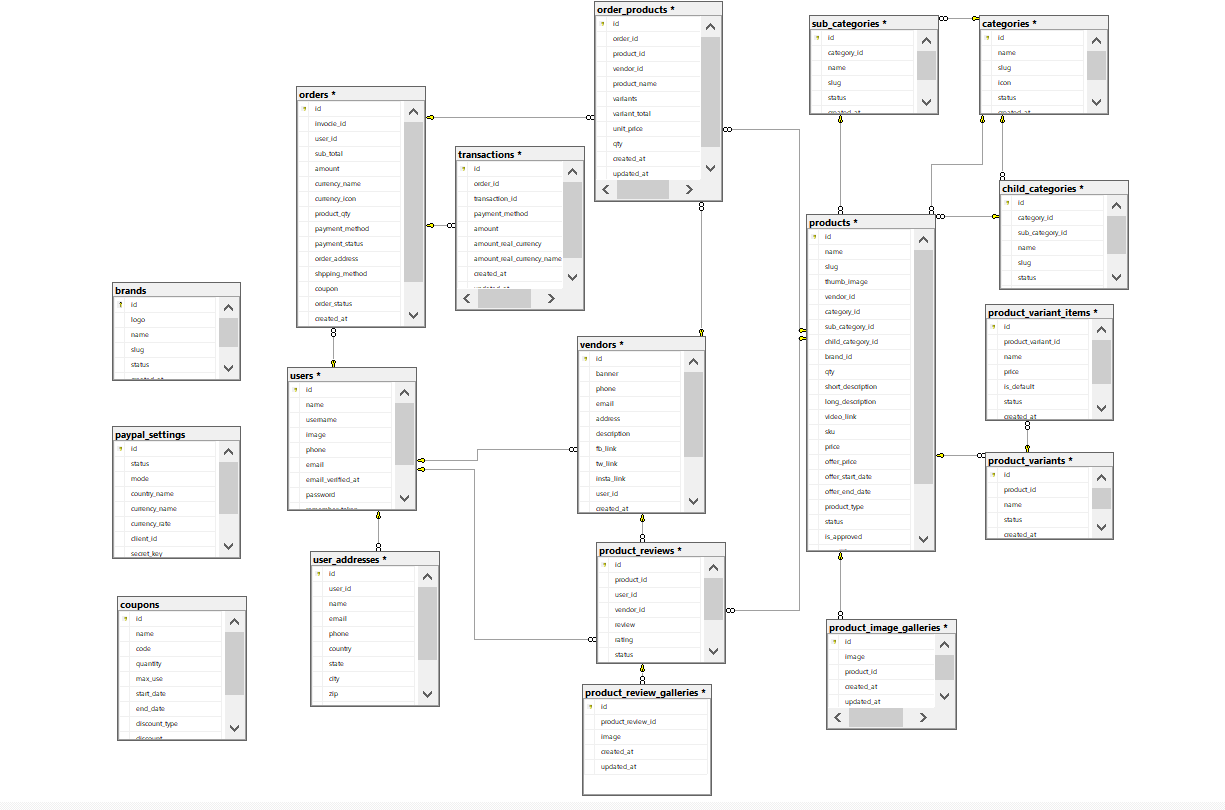
**4.5.1 System diagram**

**Sequence Diagram**

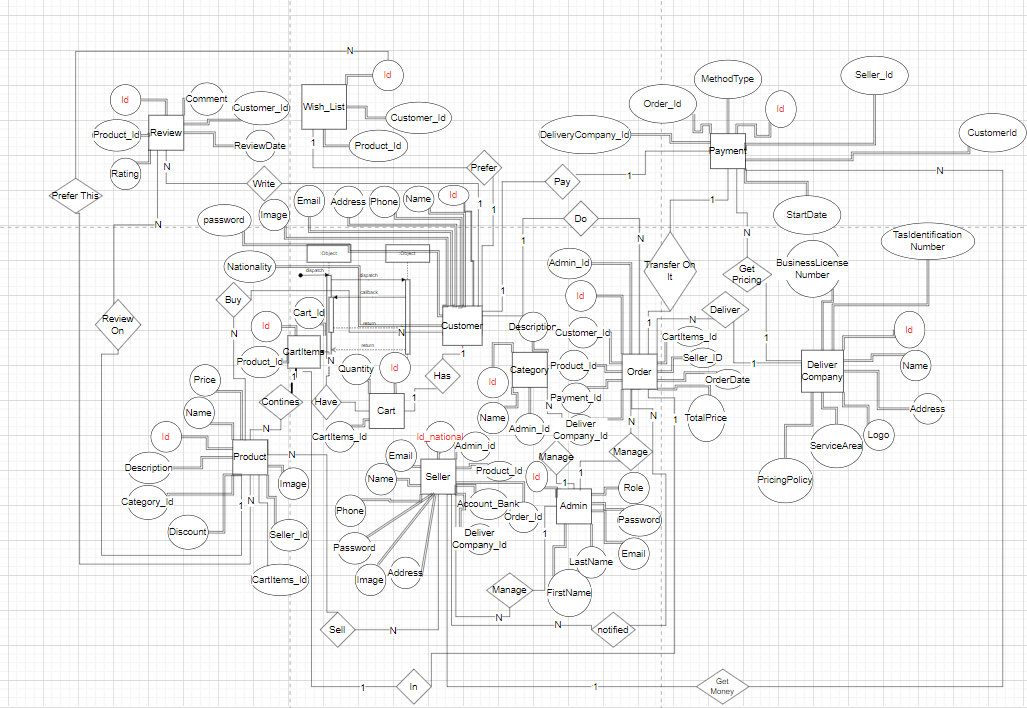
**4.6 Design Class**

**4.6.1 Class Diagram**

****

**4.7 Database Schema**

* 1. **ER Diagram**



* 1. **Design Mockup**

**Chapter 5**

**Conclusion & Feature work**

**5.1 Conclusions**

**5.2 Future work**

**5.3 References**