Software Requirements Specification (SRS)

For

Articraft– Online Platform for Selling Handmade Products

Course: MCA

Team Members: Gaurav

Pankaj

Shivani

Faculty Guide: Mr. Prashant Kr Sharma

Mr. Sachin Jindal

Institute: Hindustan Institute of Management & Computer Studies (HIMCS)

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) defines the requirements of **Articraft**, a web-based e-commerce system dedicated to selling handmade and customized products such as keychains, decorated gifts, resin plates, and photo frames.

The document is intended for:

- Developers responsible for implementation using Django and SQLite.
- Project guide and evaluators overseeing the academic project.
- Testers who will verify that the final system meets the stated requirements.
- Future maintainers who may extend the system with additional features such as online payment or recommendation modules.

1.2 Scope

Articraft will act as an **online store** where:

- Customers can browse, search, filter, and purchase handmade products.
- Admin users can manage products, categories, basic orders, and user accounts.
- The application provides a simple and responsive interface suitable for desktop and mobile devices.

1.3 Definitions, Acronyms, and Abbreviations

- **SRS** – Software Requirements Specification
- **UI** – User Interface
- **DB** – Database
- **CRUD** – Create, Read, Update, Delete
- **ORM** – Object Relational Mapper (Django ORM)
- **Cart** – Temporary collection of products selected by a user for purchase • **SKU** – Stock Keeping Unit (unique product identifier)

1.4 References

1. Articraft project brief and previous SRS draft.
2. IEEE Standard 830-1998 – Recommended Practice for Software Requirements Specifications.
3. Django Documentation – official documentation for Django web framework.
4. SQLite Documentation – official documentation for SQLite database.
5. MDN web docs – references for HTML5, CSS3, and JavaScript.

1.5 Overview

The remaining sections of this SRS describe:

- Overall system description and feasibility (Section 2)
- Detailed functional and non-functional requirements (Section 3)
- External interfaces (Section 3.3)
- System models such as use-case and data-flow descriptions (Section 4)

- Other technical constraints and appendices (Sections 5–7)

2. Overall Description

2.1 Feasibility Study

Feasibility Dimension	Assessment and Rationale
Technical Feasibility	High. The system uses well-known technologies: Python (Django), HTML, CSS, JavaScript, and SQLite. All tools are opensource and compatible with the development environment typically used in MCA labs. Django's ORM provides direct support for SQLite, simplifying DB operations.
Operational Feasibility	High. Articraft replaces informal or social-media-based selling with a structured, easy-to-use web interface for both customers and administrators. Admins can manage products and orders through a browser, and customers can shop at any time.
Economic/Schedule Feasibility	High. Since this is an academic project, there is no licensing cost for the stack (Django, SQLite, VS Code, etc.). The main investment is student time, which fits within a semester/miniproject schedule.
Legal/Ethical Feasibility	High. The system stores only basic personal data (name, email, address) for academic purposes. Passwords are encrypted using Django's security mechanisms, and basic privacy practices are followed. No real payment information is stored in the current version.

2.2 Product Perspective

Articraft is a **standalone web application**. It does not depend on existing institutional portals or third-party systems in its initial version. • The application follows the **Model-View-Template (MVT)** pattern of Django.

- SQLite acts as the backend database for storing user details, products, categories, carts, and orders.

- The system can later be integrated with external payment gateways (Razorpay, Stripe, UPI) and delivery APIs, but those are not mandatory for the academic submission.

2.3 Product Functions

Key functions of the Articraft system include:

- **User Module**
  - User registration and login ○ View and edit profile ○ View previous orders
- **Product Module**
  - View list of all products
  - View product details with images, description, price, and stock ○ Filters: by category (keychains, frames, decorated gifts, etc.), price range, and tags such as “resin”, “customized”, “photo frame”
  - Text search on product name and short description
- **Cart & Order Module** ○ Add multiple products to cart ○ Increase/decrease quantity, remove items ○ View total price and estimated delivery info (static) ○ Place order (simulated, with order stored in DB but payment assumed as “Cash on Delivery” or “Paid Offline”)
- **Admin Module**
  - Add, edit, and delete products and categories
  - Upload/replace product images ○ View basic orders and their status ○ View registered users

2.4 User Classes and Characteristics

1. **Customer / Registered User**
   a. Can browse products, manage cart, and place orders.
   b. Requires only basic knowledge of web browsing.
2. **Guest User**
   a. Can browse and filter products, but must register/login to place an order or see order history.
3. **Administrator**
   a. Has elevated privileges to manage catalog, categories, and basic orders.
   b. May be a project team member or faculty representative.

c. Expected to have moderate technical skills and familiarity with Django admin.

2.5 Operating Environment

- **Client Side** ○ Devices: Desktop, laptop, smartphone, tablet ○ OS: Windows / Linux / Android / iOS ○ Browsers: Chrome, Firefox, Edge (latest versions)
- **Server Side** ○ OS: Windows or Linux (for local deployment) ○ Backend: Python 3.x, Django Framework ○ Database: SQLite (default Django DB) ○ Web Server: Django development server for project demonstration

2.6 Assumptions and Dependencies

- Users have a stable internet connection and a modern web browser.
- Product images and descriptions are prepared by the content/marketing team.
- As of the current version, payments are assumed to be handled offline; integration with online payment services is a future dependency.

3. Specific Requirements

3.1 Functional Requirements

User Module:

- Users can register, log in, and view products.

- Users can add products to their cart and place orders.

- Users can view order history.

A**dmin Module:**
- Admin can add, update, and delete products.

- Admin can view orders and manage users.

Product Module:

- Each product has a name, image, price, description, and stock quantity.

- Products can be filtered by category or price range.

Cart & Order Module:

- Users can add multiple products to their cart.

- Orders are confirmed and tracked.

- Payment gateway integration can be added later

3.2 Non-Functional Requirements

Performance: The website should load within 3 seconds for broadband users.

Scalability: The system should handle increasing product and user data.

Security: Passwords are encrypted; admin routes are protected.

Usability: The UI should be clean, responsive, and intuitive.

Availability: Accessible 24/7 unless under maintenance.

Maintainability: Django ensures modular and maintainable backend code

3.3 External Interface Requirements

3.3.1 User Interfaces

- **Login/Signup Pages** ○ Simple forms with validation messages for incorrect data.
- **Home Page**
  - Banner introducing the Articraft store.
  - Featured product section.
- **Shop/Product Listing Page**
  - Grid view of products with image, title, and price.
  - Sidebar or top section for filters (categories, price range).
- **Product Detail Page** ○ Large product image; description; price; stock; “Add to Cart” button.
- **Cart Page**
  - List of selected products with quantity, subtotal, and overall total.
- **Order History Page** ○ List of past orders with date/time, items, and status.

- **Admin Dashboard (Django Admin)** ○ Forms to add/edit/delete products and categories; view users and orders.

3.3.2 Software Interfaces

- Django ORM for all DB operations.
- SQLite as the storage engine (db.sqlite3).
- Optional SMTP configuration for account-related emails (password reset, etc.).

3.3.3 Hardware Interfaces

- Any computer or mobile device capable of running a modern web browser for clients.
- Development machine with at least dual-core CPU, 4 GB RAM, and adequate disk space.

3.3.4 Communication Interfaces • HTTP/HTTPS protocol for

client–server communication.

- Localhost access for development; optional LAN or public hosting for demonstration.

4. System Models

4.1 Use Case Model (Textual Description)

Actors:

- Customer (Registered User)
- Guest User
- Administrator

Main Use Cases:

- **Register** – Guest creates a customer account.
- **Login / Logout** – Customer or admin authenticates and ends session.
- **Browse Products** – Customer or guest views product list and filters.

- **View Product Details** – Customer or guest opens detailed view of a single product.
- **Manage Cart** – Customer adds/removes items and updates quantities.
- **Place Order** – Customer confirms order from cart.
- **View Orders** – Customer views past orders.
- **Manage Products** – Admin performs CRUD operations on products and categories.

4.2 Data Flow Diagram (DFD – Conceptual Description)

Level 0:

- External entities: Customer, Admin • Main processes:
  - ○ *Product Management* ○
  Shopping & Ordering ○
  User Management
- Data stores: Users DB, Products DB, Orders DB, Cart DB (all inside SQLite).

Level 1 (Example: Shopping & Ordering):

1. Customer requests product list → System retrieves products from Products DB → System shows list.
2. Customer selects product and quantity → System updates Cart DB.
3. Customer clicks "Place Order" → System reads Cart DB, creates new order in Orders DB, clears cart.

Level 1 (Example: Product Management):

1. Admin logs in → System validates credentials from Users DB.
2. Admin submits new product form → System validates data, stores in Products DB.
3. Admin updates or deletes product → System modifies Products DB accordingly.

4.3 ER Diagram (Conceptual Description)

Main entities and relationships:

- **User**(UserID, Name, Email, PasswordHash, Address, Phone, Role)

- **Category**(CategoryID, CategoryName)
- **Product**(ProductID, Name, Description, Price, StockQty, ImagePath, CategoryID FK)
- **Cart**(CartID, UserID FK, CreatedAt)
- **CartItem**(CartItemID, CartID FK, ProductID FK, Quantity)
- **Order**(OrderID, UserID FK, OrderDate, TotalAmount, Status)
- **OrderItem**(OrderItemID, OrderID FK, ProductID FK, Quantity, PriceAtOrderTime)

Relationships:

- One **User** can have many **Orders** and one active **Cart**.
- One **Category** can have many **Products**.
- One **Order** can have many **OrderItems**.
- One **Cart** can have many **CartItems**.

5. Other Requirements

- Regular backup of the SQLite database file (db.sqlite3) is recommended during development and before evaluation.
- Sample test data (e.g., preset categories: “Resin Art”, “Keychains”, “Photo Frames”) should be prepared for demonstration.
- The system should follow basic privacy measures: user passwords never logged in plain text; admin access protected by strong credentials.
- For future extension, REST APIs can be added using Django REST Framework to support mobile apps or external integrations.

6. Appendices

6.1 Sample Data (Illustrative)

- **Categories:** Resin Art, Custom Keychains, Decorated Plates, Photo Frames, Gift Hampers
- **Sample Products:**
    - “Galaxy Resin Keychain” – resin, glitter, name customization
    - “Memories in Frame” – custom photo frame with dried flowers
    - “Festive Plate Décor” – decorated plate for festivals

6.2 Tools Used

- Python 3.x
- Django Framework
- SQLite
- HTML5, CSS3, JavaScript
- VS Code

7. Conclusion

This SRS defines in detail the **functional**, **non-functional**, and **environmental** requirements of the Articraft e-commerce platform for handmade products. It demonstrates that the system is technically and operationally feasible using Django and SQLite, and that it satisfies the academic objectives of an MCA project.

The document provides a clear blueprint for designing, implementing, and testing the application, and can serve as a baseline for future enhancements such as online payment integration, recommendation systems, and mobile applications.