Software Requirements Specifications

Final version

Gift economy/Gift culture app

Seif Saber

Omar Ahmed  
Omar Amir  
Karim Mansour  
David Dawood   
Yousef El-Bedewy

**1 Introduction**

***1.1. Purpose***

The purpose of this document is to present a detailed description of the Web Publishing System. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the stakeholders and the developers of the system and will be proposed to the Regional Historical Society for its approval.

***1.2. Scope of Project***

This software system will be a web-based gifting platform designed to allow individuals within a community to give away unused or unwanted items to others, completely free of charge. The system is intended to maximize the efficiency and ease with which users can share items by automating the listing, browsing, communication, and exchange process, which would otherwise be conducted manually through informal means such as social media or word-of-mouth.

By simplifying the gifting process and promoting a culture of reuse, this system will meet community needs for sustainability, convenience, and social connection—while remaining easy to understand and use for a broad audience.

More specifically, this platform enables users to:

* Register and manage personal accounts
* Post items they wish to give away, including images, categories, and descriptions
* Browse available items in different categories
* Contact item donors through internal messaging
* Mark items as gifted or remove them once taken

The system also supports administrative functionality for overseeing user activity and managing listings. Communication between users is facilitated via internal request features and automated email notifications.

To manage data effectively, the system integrates with a relational database, which maintains structured records of users, item listings, categories, and request transactions. The database supports secure storage and efficient retrieval of data essential to the platform's operation.

***1.3 Glossary***

|  |  |
| --- | --- |
| Term | Definition |
| Author | Person submitting an article to be reviewed |
| Database | Collection of all the information monitored by this system. |
| Reader | Anyone visiting the site to read articles. |
| Software Requirements Specification (SRS) | A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document. |
| Stakeholder | Any person with an interest in the project who is not a developer. |
| End user | The person that uses the system |
| MoSCoW scheme | The **MoSCoW scheme** is a prioritization technique used in software requirements gathering to categorize features based on their importance. It classifies requirements into four groups: **Must-Have** – Essential features required for the system to function. **Should-Have** – Important but not critical; workarounds may exist. **Could-Have** – Desirable features that can enhance user experience but are not necessary. **Won’t-Have** Features that are agreed to be excluded in the current release but may be considered later. |

***1.4 Stakeholders***

Below is a comprehensive list of stakeholders involved in the development, use, and maintenance of the EchoGifts platform, along with a detailed description of their roles and interests in the system:

|  |  |
| --- | --- |
| Stakeholder | Description |
| End Users (Community Members) | These are individuals who register on the platform to give away or receive items. They rely on the system to browse listings, post giveaways, and communicate with other users. Their primary needs are usability, accessibility, and trust in the platform |
| System Administrator | Responsible for maintaining the overall health and functionality of the system. Their duties include moderating listings, managing user accounts, monitoring content, and ensuring the platform operates smoothly and securely. |
| Developers | The technical team tasked with implementing the features, fixing bugs, and maintaining the codebase. They work closely with designers and testers to ensure the system meets both functional and non-functional requirements |
| Testers | Responsible for validating the functionality, performance, usability, and security of the platform. They create and execute test cases to ensure the system behaves as expected under various scenarios. |
| UI designers | Ensure the user interface is intuitive, visually appealing, and accessible. Their focus is on creating a seamless user experience across devices while aligning with user expectations. |

***1.5 references***

IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications.

MoSCoW Prioritization Method:

Employed for categorizing requirement priorities into Must-Have, Should-Have, Could-Have, and Won’t-Have.

https://www.cse.msu.edu/~cse435/Handouts/SRSExample-webapp.doc

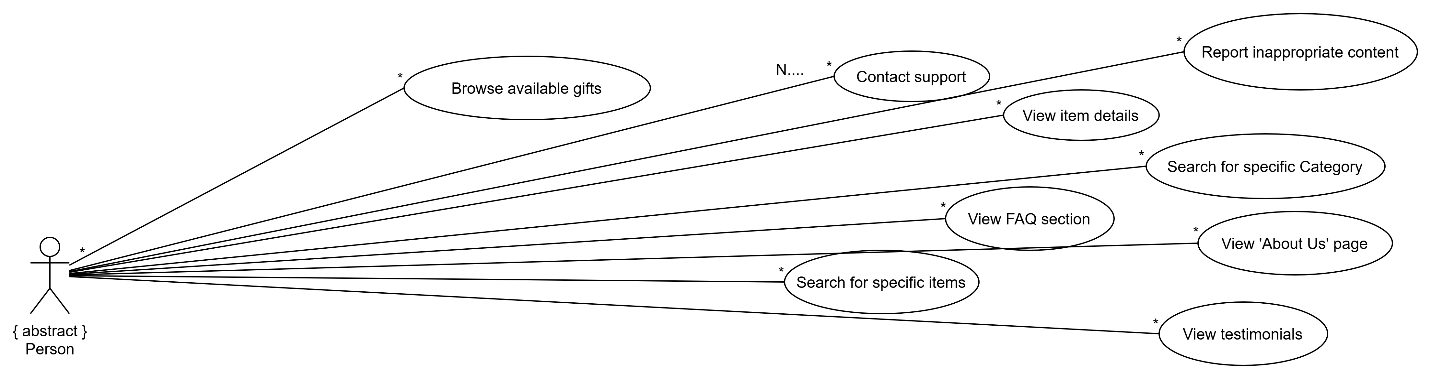
**2 Functional Requirements**

***2.1 User requirements specification***

This section outlines the use cases for each type of user individually. registered user is the primary actor within the system.

2.1.1 Abstract class use cases

This abstract class is the base for all types of users which all of them inherent from



**Important Use Case**

Browse Available Gift

Author

Browse available gifts

|  |  |
| --- | --- |
| Trigger | The user presses the products page |
| Preconditions | None |
| Basic path | 1) User navigates to the homepage of the Echo Gifts website.  2) User selects the products section.  3) The system displays a list of available products.  4) User uses filters (category) or the search bar to narrow results.  5) The system updates the gift listings based on the search/filter criteria.  6) User clicks on a specific product to view more details.  7) The system displays the full gift description, including contact info or request option.  8) User may choose to contact the gift owner or save the gift for later. |
| Alternative paths | If user searches for somethings that is not there:  System shows “not found” |
| Postcondition | The user has saved the product or has messaged the giver |

2.1.2 Visitor Use Cases



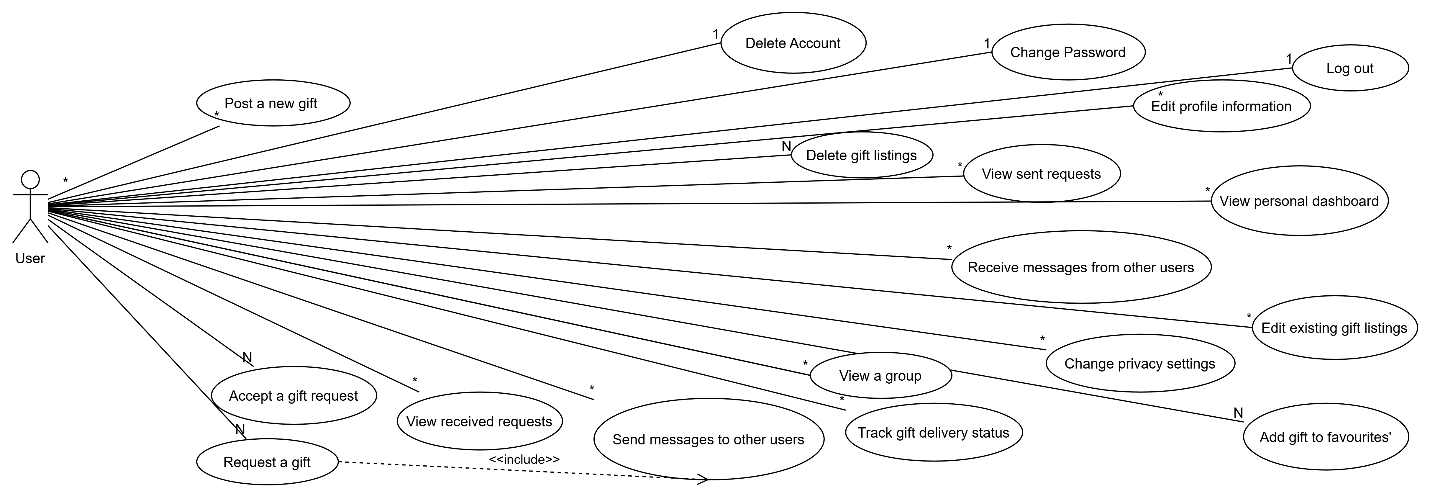
Use case: Register account

The user registers for the first time which will redirect them to the login page afterwards. The visitor is promoted to a user

Use Case: Login

The user inputs his/her credentials and the system validate his/her credentials before continuing to the home page.

2.1.3 User Use Cases



## Use Case: Post a New gift

|  |  |
| --- | --- |
| Trigger | User clicks on 'add a post' button |
| Preconditions | User must be logged in |
| Basic path | 1) Navigate to 'add a post' page 2) Enter product details (title, description, category, image) 3) Submit form 4) System saves and displays listing |
| Alternative paths | If fields are empty or invalid, system prompts user to correct them |
| Postcondition | Product is added and visible to other users |

## Use Case: Request a gift

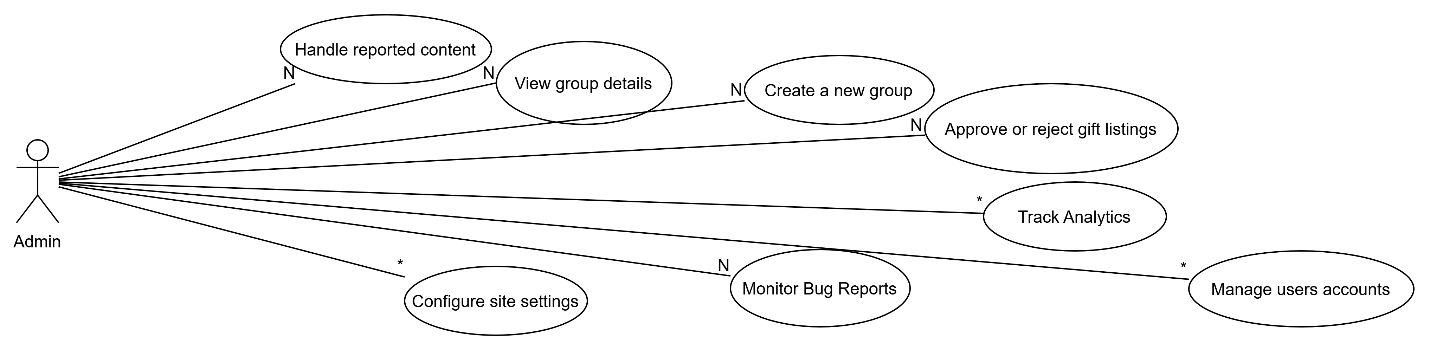
|  |  |
| --- | --- |
| Trigger | User clicks 'Request' on a gift listing |
| Preconditions | User must be logged in |
| Basic path | 1)Open item detail page 2) Click 'message' 3) System notifies the gift owner |
| Alternative paths | If gift is already marked as taken, system shows error message |
| Postcondition | Request is recorded and sent to the owner |

## Use Case: Add gift to Favorites

|  |  |
| --- | --- |
| Trigger | User clicks 'Add to Favorites’ on a listing |
| Preconditions | User must be logged in |
| Basic path | 1) User views item 2) Clicks 'Add to Favorites’ 3) System adds gift to user’s favorites list |
| Alternative paths | If item is already in favorites, system may show 'Already added' |
| Postcondition | Item is added to user's favorites |

2.1.4 Admin Use Cases

Admin inherits from The user and abstract class thus having their use cases in addition to its own use cases



**Use Case: Approve or Reject gift Listings**

|  |  |
| --- | --- |
| Trigger | Admin views pending listings |
| Preconditions | Admin must be logged in with proper permissions |
| Basic path | 1) Navigate to 'Pending Listings' 2) Review each listing 3) Click 'Approve' or 'Reject' |
| Alternative paths | If listing is missing required fields, admin can reject with reason |
| Postcondition | Gift status is updated accordingly |
|  | |

## Use Case: Manage User Accounts

|  |  |
| --- | --- |
| Trigger | Admin opens user management dashboard |
| Preconditions | Admin must be authenticated and authorized |
| Basic path | 1) Open user management panel 2) Select user profile 3) Suspend, warn, or delete user |
| Alternative paths | If admin lacks permission, system shows access denied |
| Postcondition | User account status is updated |

***2.2 system requirements***

| **Requirement ID** | **SR-01** |
| --- | --- |
| **Title** | System Support for Browsing Gifts |
| **Description** | The system shall allow users (both guests and logged-in users) to view a catalog of available gifts. The catalog shall be dynamically fetched from the database and displayed with images, names, and brief descriptions. Users can filter or sort items based on categories, price, or popularity. |
| **Inputs** | None (initiated by visiting homepage or catalog section) |
| **Outputs** | List of gifts retrieved from the database, rendered on the webpage |
| **Process/Logic** | The system queries the gift database, applies any selected filters/sorting, and returns a list of gift objects to the frontend. |
| **Dependencies** | Gift database, frontend rendering components, gift filtering logic |
| **Constraints** | Response time for gift loading must not exceed 3 seconds under normal conditions. |

| **Requirement ID** | **SR-02** |
| --- | --- |
| **Title** | System Support for User Registration |
| **Description** | The system shall provide a registration form that allows new users to sign up by entering their full name, email, username, and password. Upon submission, the system validates the input, checks for existing email or username conflicts, and stores the new user data in the database. |
| **Inputs** | User's full name, email address, username, password |
| **Outputs** | Confirmation message upon successful registration, or error messages in case of input validation failures |
| **Process/Logic** | The system checks for input completeness and uniqueness of email/username. If valid, it stores the new user object and redirects to the login page. |
| **Dependencies** | Registration form UI, user database, input validation logic |
| **Constraints** | Email and username must be unique. Password must meet minimum security requirements. |

| **Requirement ID** | **SR-03** |
| --- | --- |
| **Title** | System Support for User Login |
| **Description** | The system shall provide a login form where users input their username and password. Upon submission, the system authenticates the credentials and grants access if they are valid. |
| **Inputs** | Username, Password |
| **Outputs** | User redirected to homepage or dashboard upon success, or shown an error message on failure |
| **Process/Logic** | The system checks the credentials against the stored user records. If valid, it starts a session for the user. If invalid, an error message is displayed. |
| **Dependencies** | Login form, user database, authentication logic, session management |
| **Constraints** | The system must prevent brute-force attacks through limited login attempts or verification after multiple failures. |

| **Requirement ID** | **SR-04** |
| --- | --- |
| **Title** | System Support for user to Add a Product |
| **Description** | The system shall allow a user to add a new product by entering the product name, description, image, category, and optional tags. The system saves this data to the product catalog in the database. |
| **Inputs** | Product name, description, image, category, tags |
| **Outputs** | Confirmation message upon successful addition, or error message if inputs are invalid |
| **Process/Logic** | The system validates all required fields and image is a valid format. Once validated, it creates a new product record and updates the catalog view. |
| **Dependencies** | Product form UI, product database, admin panel access, image storage logic |
| **Constraints** | All required fields must be completed. Image file must not exceed specified size limit (e.g., 5MB). |

| **Requirement ID** | **SR-05** |
| --- | --- |
| **Title** | System Support for User to Edit a Custom Product |
| **Description** | The system shall allow a logged-in user to edit the details of a previously created or saved gift item. Editable fields include name, description, price estimate, tags, and image. Upon submission, the system validates the changes and updates the corresponding entry in the user's profile. |
| **Inputs** | Updated custom product fields: name, description, price, image, tags |
| **Outputs** | Confirmation message upon successful update, or validation error messages |
| **Process/Logic** | The system retrieves the selected gift entry, populates the edit form, and upon submission, validates and saves changes to the user’s profile or associated database entry. |
| **Dependencies** | User authentication, custom product database, form UI components |
| **Constraints** | Only the user who created the custom gift can edit it. All input validations must pass. Image uploads must adhere to size and format rules. |

| **Requirement ID** | **SR-07** |
| --- | --- |
| **Title** | System Support for Admin to Approve or Reject Product Listings |
| **Description** | The system shall allow an admin to review newly submitted product listings by users (e.g., custom gifts or suggestions) and either approve them for public display or reject them with an optional comment. |
| **Inputs** | Submitted product information, admin decision (Approve / Reject), optional feedback |
| **Outputs** | Confirmation message, updated product status (Approved or Rejected), user notification |
| **Process/Logic** | Admin reviews submitted listings from the pending queue. Upon choosing an action, the system updates the product’s approval status and optionally sends a rejection reason to the submitting user. Approved listings become publicly visible. |
| **Dependencies** | Admin dashboard, pending listings queue, product database, notification system |
| **Constraints** | Only users with admin privileges may access this function. Each listing must have a clear approval status stored. Users must be notified of the outcome. |

| **Requirement ID** | **SR-07** |
| --- | --- |
| **Title** | System Support for Admin to Approve or Reject Product Listings |
| **Description** | The system shall allow an admin to review newly submitted product listings by users (e.g., custom gifts or suggestions) and either approve them for public display or reject them with an optional comment. |
| **Inputs** | Submitted product information, admin decision (Approve / Reject), optional feedback |
| **Outputs** | Confirmation message, updated product status (Approved or Rejected), user notification |
| **Process/Logic** | Admin reviews submitted listings from the pending queue. Upon choosing an action, the system updates the product’s approval status and optionally sends a rejection reason to the submitting user. Approved listings become publicly visible. |
| **Dependencies** | Admin dashboard, pending listings queue, product database, notification system |
| **Constraints** | Only users with admin privileges may access this function. Each listing must have a clear approval status stored. Users must be notified of the outcome. |

| **Requirement ID** | **SR-08** |
| --- | --- |
| **Title** | System Support for Admin to Manage User Accounts |
| **Description** | The system shall allow an admin to view, edit, deactivate, or delete user accounts. Admins can search for users, view their registration details, and take appropriate actions to ensure platform integrity. |
| **Inputs** | User account selection, admin actions (edit/deactivate/delete), updated user details (if applicable) |
| **Outputs** | Confirmation messages for each action taken, updated user status or data |
| **Process/Logic** | The system displays a list or search view of all registered users. Admins can select an account, view details, and perform actions. Changes are validated and then saved to the user database. |
| **Dependencies** | Admin dashboard, user database, search/filter UI, authentication system |
| **Constraints** | Only authorized admins can manage accounts. Critical actions (e.g., delete) may require confirmation. Edits must comply with validation rules. |

***2.3 Requirements’ priorities (MoSCoW scheme)***

|  |  |
| --- | --- |
| Requirement / Use Case | MoSCoW Priority |
| Browse Available Gifts | Must-Have | core to platform’s function; all users must access item listings. |
| Register Account | Must-Have | Enables identity creation. |
| Login | Must-Have | Required for user-specific actions |
| Post a New Gift | Must-Have | Essential to maintain item inventory and purpose of platform. |
| Request a Gift | Must-Have | Enables exchange important core platform goal. |
| Approve or Reject Gift Listings (Admin) | Must-Have | Maintains content moderation and platform trust. |
| Manage User Accounts (Admin) | Must-Have | Essential for platform security and control. |
| Search & Filter Gifts | Must-Have | Ensures usability when many items exist. |
| Internal Messaging Between Users | Must-Have | Required for gift exchange coordination. |
| Mark Item as Gifted | Must-Have | Keeps platform clean and listings accurate. |
| Guest Browsing Without Login | Must-Have | To allow engagement before requiring sign-up. |
| Add Gift to Favorites | Should-Have | Enhances user experience but not essential functionality. |
| Edit Gift After Posting | Should-Have | Improves usability; not strictly required if deletion is possible. |
| Delete Own Gift Listing | Should-Have | Users should manage their listings; complements 'Edit Gift'. |
| Profile Management (User) | Should-Have | Enhances UX; includes changing password, profile picture, etc. |
| Mobile Responsiveness | Should-Have | Important for accessibility but can be improved iteratively. |
| Email Notifications (Request / Approval) | Could-Have | Enhances communication but platform can function without it. |
| Multi-language Support | Could-Have | Depends on user base; can be added after English version. |
| Social Media Sharing | Won’t-Have | Not critical for MVP; can be explored in future versions. |
| Advanced Analytics for Admin | Won’t-Have | Useful later, but not needed for first release. |

**3 Non-functional requirements**

***3.1 Requirements Categories***

The requirements are going to be grouped into 8 categories according to Roberston & Robertson:

*1) Look-and-Feel REQs: The spirit of the product’s appearance.*

*2) Usability & Humanity REQs: The product’s ease of use, and any*

*special usability considerations.*

*3) Performance requirements: How fast, how safe, how accurate, how*

*reliable, and how available the functionality must be.*

*4) Operational & Environmental REQs: The environment on which the*

*product will have to work (e.g., under water), & what considerations*

*must be made for this environment.*

*5) Maintainability & Support REQs: Expected changes, and the time*

*allowed to make them.*

*6) Cultural REQs: Special requirements that come about because of*

*the people involved in the product’s development and operation.*

*7) Legal REQ: The laws and standards that apply to the product.*

*8) Security REQ: The security and confidentiality of the product.*

***3.2 Non-functional requirements specification***

### *3.2.1* ***Look-and-Feel Requirements***

* The platform should have a clean, modern, and visually appealing UI that reflects the spirit of gifting and celebration.
* Consistent color themes (green and white) and intuitive navigation should be maintained across all pages.
* The design should be responsive and mobile-friendly, ensuring the same aesthetic on smartphones and tablets.

### *3.2.2* ***Usability & Humanity Requirements***

* New users should be able to sign up and place a gift order within 5 minutes.
* The system should provide helpful tooltips, validations, and error messages for better guidance.
* Accessibility should be ensured for users with visual impairments via contrast settings and screen reader support.
* User actions such as searching for gifts, adding items to the cart, and making payments should be achievable without complexity.

### *3.2.3* ***Performance Requirements***

* The website should load within 3 seconds on standard networks.
* The platform should support at least 100 concurrent users without performance degradation.
* System uptime must be at least 99.5% monthly.

### *3.2.4* ***Operational & Environmental Requirements***

* The web application should function on major browsers (Chrome, Firefox, Edge, Safari).
* It must operate smoothly in environments with moderate internet latency.
* Hosting infrastructure should be cloud-based for scalability and remote accessibility.
* Offline functionality is not required but the application should degrade gracefully in case of internet disconnections.

### *3.2.5* ***Maintainability & Support Requirements***

* Codebase should be modular and well-documented to allow new developers to onboard within 2 weeks.
* Bug fixes should be implementable within 48 hours of identification.
* Feature updates or minor changes (e.g., adding a new gift category) should take no longer than 5 business days.
* A ticketing system should be in place for users to report issues or request support.

### *3.2.6* ***Cultural Requirements***

* The platform should support international holidays and regional gifting customs (e.g., Ramadan, Christmas).
* Support for multilingual interfaces (starting with English) should be considered.
* Avoid culturally insensitive icons, colors, or phrases.

### *3.2.7* ***Legal Requirements***

* User data must be stored and processed in compliance with GDPR and local data protection laws.
* User consent must be obtained before collecting any personal data (via checkboxes during registration).
* Terms of service and privacy policies must be easily accessible and accepted before using the platform.

### *3.2.8* ***Security Requirements***

* The system must implement secure authentication (e.g., email verification, 2FA).
* All data transactions should occur over HTTPS.
* Sensitive user data (addresses, contact info) must be encrypted.
* Regular security scans should be conducted.

***3.3 The fit criteria for every testable Non-Functional Requirement***

|  |  |
| --- | --- |
| Requirement | Fit criterion |
| Look-and-Feel Requirements | |  | | --- | |  |  |  | | --- | | The system must apply the EchoGifts brand color palette and logo across all pages. At least 95% of surveyed users must rate the interface as visually appealing and clear. | |
| Usability & Humanity Requirements | |  | | --- | | New users must be able to complete the registration process in under 2 minutes. 90% of users must navigate and use core features (e.g., search, add to favorites) without assistance. |  |  | | --- | |  | |
| |  | | --- | | Performance Requirements |  |  | | --- | |  | | |  | | --- | | The homepage and product listings must load within 3 seconds under standard network conditions. The system should support 200 concurrent users without slowing down. |  |  | | --- | |  | |
| |  | | --- | | Operational & Environmental Requirements |  |  | | --- | |  | | |  | | --- | | The system must function correctly on major browsers (Chrome, Firefox, Safari, Edge). |  |  | | --- | |  | |
| Maintainability & Support Requirements | |  | | --- | | Content and gift listings should be updatable through an admin panel without code modifications.. |  |  | | --- | |  | |
| |  | | --- | | Legal requirements | | None |
| |  | | --- | | Security Requirements |  |  | | --- | |  | | User passwords must be stored using encryption. Admin-only features must be access-controlled. |

***3.4 How would the above-mentioned Non-Functional Requirements affect the System's overall Architecture?***

The non-functional requirements influence the system’s architecture in key ways. Look-and-feel and usability needs guide the use of a user-friendly, responsive UI framework. Performance and operational requirements affect back-end efficiency, necessitating fast database queries, and scalability. Maintainability encourages modular design for easier updates. Legal and security requirements enforce secure storage, encrypted communication, and role-based access.

**4 Design & Implementation Constraints**.

* The system must be implemented using **HTML, CSS, JavaScript,** and **PHP**.
* The backend must use **SQL** for data storage.
* The application must be **web-based** and accessible via modern browsers (e.g., Chrome, Firefox).
* The system must follow a client-server architecture.
* Admin access must be restricted and protected via role-based authentication.
* Usernames must be unique and cannot exceed 20 characters.

**5 System evolution**

***5.1 anticipated changes***

We are willing to further develop our website to be available as a mobile version for both IOS and Android devices, additionally, there will be more languages available. We will also be enhancing our algorithms to further show the users the desired products more often.

***5.2 How will any of the anticipated changes affect the future***

Mobile application development is a must because as time goes on the mobile users increase and so is technology, so to reach as many users as possible we need to implement a mobile app. Furthermore, for languages, all texts and interfaces should be stored in external resource file for easier translation.

Algorithms will be enhanced using A.I technology to track the users’ requirements as efficiently as possible.

**6 Requirements Discovery approaches**

The analysis we used were documented analysis and use Case analyzing.

Documented Analysis: Upon reviewing similar gifting sites like Buynothingproject and reviewing its SRS for insights.

Use case analyzing: developed use Case diagram including the requirements and prioritizing them using the MoSCoW scheme. Defining clear definitions for important functions like “register”, “login” ,and “browse product” as mentioned in **REF 2.1.1**

**7) requirements validation techniques**

Reviewing was the main way of validation technique that was used were we Systematically reading and checking the SRS and diagrams with team members.

Upon Testing we saw that the login feature was not sending the info properly so we employed a method to fix it.