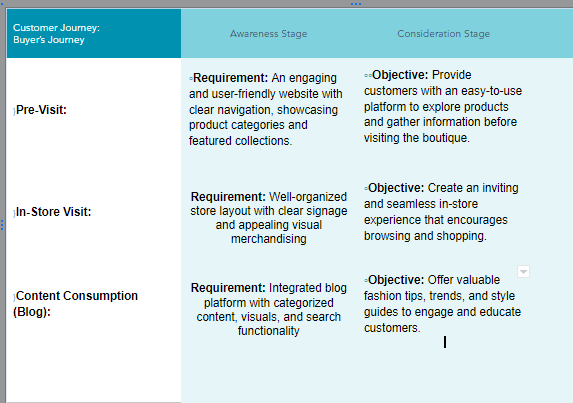
|  |  |  |
| --- | --- | --- |
| Project Title | : | Content Marketing Plan Template Based on HubSpot |
| NM Id | : | DDA9CBF432CA23B932244481FEA55216 |
| Industry Mentor(s) Name | : | Abdul Malic |

**Project Design Phase - Part 2**

**Determine The Requirements (Customer Journey Maps):**

* **Pre-Visit:**
  + **Requirement:** An engaging and user-friendly website with clear navigation, showcasing product categories and featured collections.
  + **Objective:** Provide customers with an easy-to-use platform to explore products and gather information before visiting the boutique.
* **In-Store Visit:**
  + **Requirement:** Well-organized store layout with clear signage and appealing visual merchandising.
  + **Objective:** Create an inviting and seamless in-store experience that encourages browsing and shopping.
* **Content Consumption (Blog):**
  + **Requirement:** Integrated blog platform with categorized content, visuals, and search functionality.
  + **Objective:** Offer valuable fashion tips, trends, and style guides to engage and educate customers.
* 

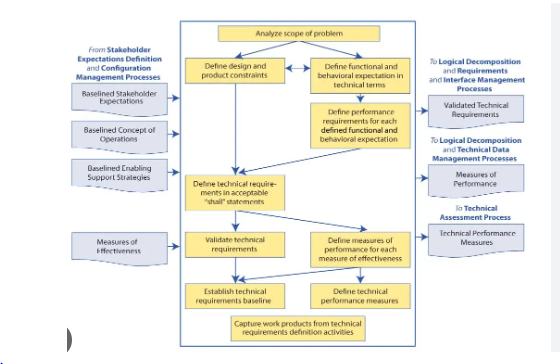
**Requirement Analysis (Functional, Operational, Technical) / Flow Charts**

**Functional Requirements:**

* **Online Shopping:** Customers should be able to browse products, view details, add to cart, and make purchases online.
* **Appointment Scheduling:** If offering personal styling consultations, there should be a feature for customers to schedule appointments.
* **Content Management:** The boutique should be able to create, edit, and publish blog content related to fashion trends, tips, and style guides.

**Operational Requirements:**

* **Store Operations:** In-store staff should be able to manage inventory, process sales, and provide customer assistance.
* **Order Fulfillment: Operations** staff should be able to efficiently process and fulfill online orders.
* **Customer Support: There** should be channels for customers to seek assistance, whether through email, phone, or in-store.

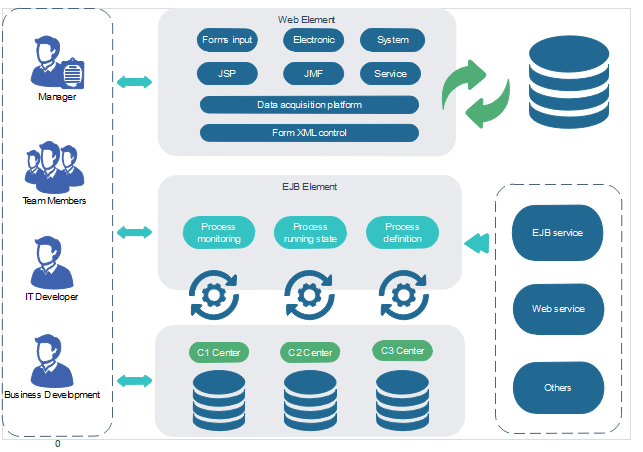


**Technical Architecture**

**Front-End:**

* + **Web Application:** The front-end of the boutique's platform is a web application accessible to customers through their browsers. This includes the website's user interface for browsing products, reading content, and interacting with features like appointment scheduling and loyalty program enrollment.

**Back-End:**

* + **Server:** The server hosts the application's logic, processes requests from the front-end, interacts with the database, and manages various functionalities.
  + **Database:** This component stores essential data, including customer profiles, product information, inventory levels, purchase history, and loyalty program details. It's essential for efficient data retrieval and storage.
* **E-commerce Platform:**
  + **Shopping Cart System:** This component manages the selection of products, allows customers to review their choices, and facilitates the checkout process.
  + **Payment Gateway Integration:** A secure payment gateway is integrated to handle online transactions, ensuring the safety of customer payment information
  + .

**Open Source Frameworks:**

**Web Development:**

* + **Django:** A high-level Python web framework that encourages rapid development and clean, pragmatic design.
  + **Ruby on Rails:** A web application framework written in Ruby that follows the Model-View-Controller (MVC) pattern.
  + **Express.js:** A minimal and flexible Node.js web application framework used for building web and mobile applications.
* **Front-End Development:**
  + **React:** A JavaScript library for building user interfaces, maintained by Facebook.
  + **Angular:** A TypeScript-based open-source framework for building dynamic web applications.
  + **Vue.js:** A progressive JavaScript framework for building modern web applications.
* **Mobile App Development:**
  + **Flutter:** An open-source UI software development toolkit by Google for building natively compiled applications for mobile, web, and desktop from a single codebase.
  + **React Native:** A framework for building mobile applications using React and JavaScript.

**Open Source Frameworks:**

**Web Development:**

* + **Django:** A high-level Python web framework that encourages rapid development and clean, pragmatic design.
  + **Ruby on Rails:** A web application framework written in Ruby that follows the Model-View-Controller (MVC) pattern.
  + **Express.js:** A minimal and flexible Node.js web application framework used for building web and mobile applications.
* **Front-End Development:**
  + **React:** A JavaScript library for building user interfaces, maintained by Facebook.
  + **Angular:** A TypeScript-based open-source framework for building dynamic web applications.
  + **Vue.js:** A progressive JavaScript framework for building modern web applications.
* **Mobile App Development:**
  + **Flutter:** An open-source UI software development toolkit by Google for building natively compiled applications for mobile, web, and desktop from a single codebase.
  + **React Native:** A framework for building mobile applications using React and JavaScript.

**Third-Party API’s**

**Social Media APIs:**

* + **Face book Graph API:** Allows access to Face book's social graph, enabling developers to interact with user profiles, posts, and more.
  + **Twitter API:** Provides access to Twitter's functionalities, allowing developers to post tweets, retrieve user information, and interact with the Twitter platform.
  + **Instagram Graph API:** Allows developers to interact with Instagram's platform, including posting content, retrieving user data, and managing interactions.
* **Mapping and Location APIs:**
  + **Google Maps API:** Enables integration of maps, location-based services, and geospatial data into applications.
  + **Map box API:** Provides customizable and interactive maps for web and mobile applications.
* **Payment Gateway APIs:**
  + **Stripe API:** Allows businesses to securely accept payments online, handling transactions, subscriptions, and more.
  + **PayPal API:** Provides a range of payment-related services, including processing payments, managing invoices, and handling subscriptions.

**Cloud Deployment:**

* **Backup and Disaster Recovery:**
  + Establish regular backup and disaster recovery processes to protect against data loss or system failures.
* **Implement Security Measures:**
  + Apply security best practices, including access controls, encryption, and regular security audits.
* **Continuous Integration and Continuous Deployment (CI/CD):**
  + Set up CI/CD pipelines for automated testing and deployment to streamline the development and release process.
* **Scaling and Auto-scaling (Optional):**
  + Configure auto-scaling policies to dynamically adjust resources based on traffic demands.
* **Monitoring and Maintenance:**
  + Regularly monitor the application's performance, security, and costs. Perform routine maintenance tasks and updates.