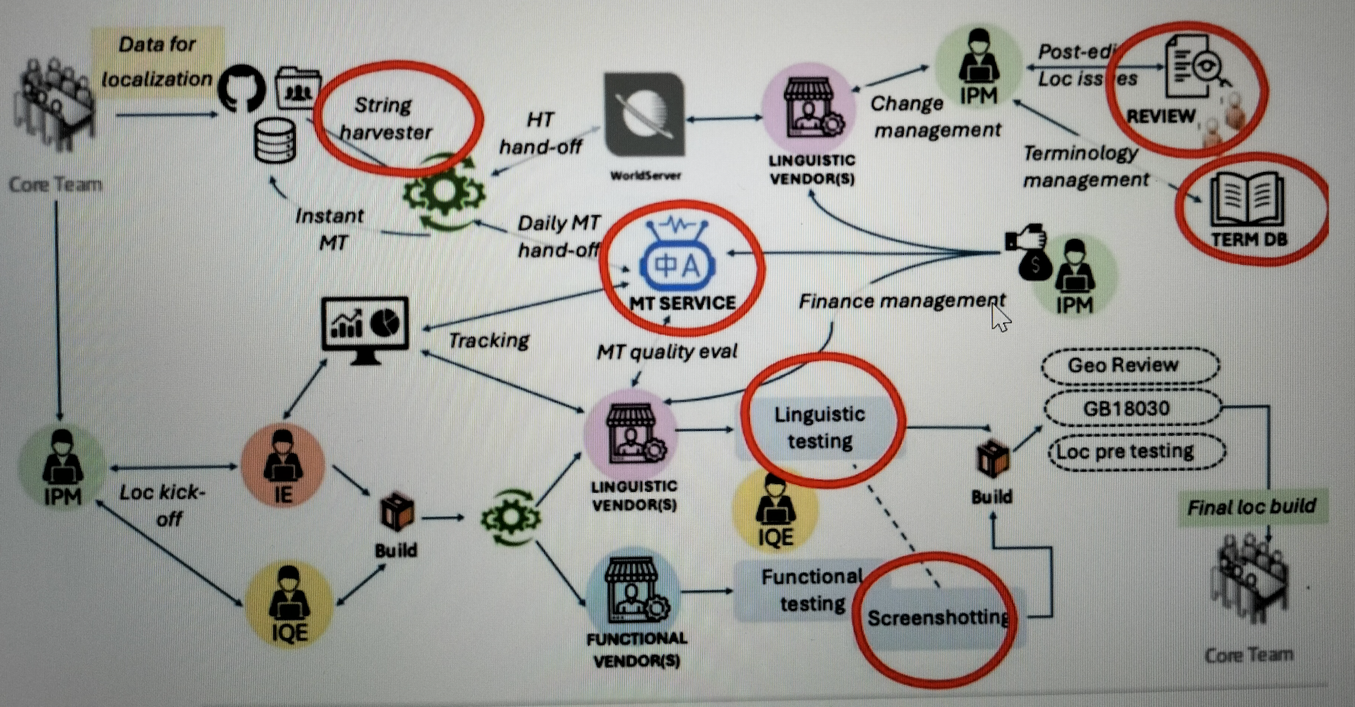
AutoLocalize Project: Automatic Website Localization and Translation Software

The AutoLocalize Project is aimed at automatically localizing and translating live websites with machine translation technologies through Application Programming Interface (API) calls. The types of websites targeted in this project include, but are not limited to: web stores, blogs, content management systems (CMS), and enterprise systems. The project expects to automate the localization and translation process using Large Language Models (LLM) and human translators for proofreading, post-editing, and review where appropriate.

The websites coming onboard as clients will be translated from and to Japanese, English, Simplified Chinese, Traditional Chinese, and Korean. The software is expected to concurrently handle multiple automatic localization requests from multiple websites at the same time. The deliverables of the project are the following components built as part of a big localization and translation ecosystem to process and automate:

* Automatic String Detection and Extraction
* Automatic Localization and Translation Workflow Tool for Live Websites
* Automatic Localization and Translation Quality Evaluation and Review Process
* Automatic Localization and Translation Plugin for CMS (all the major CMSs in the software industry)
* Continuous Localization Workflow: Automatic Multilingual Build Generation and Screenshot Collection for Quality Assurance (QA)/Linguistic Review Process

The above points are the deliverables of the project which are expected to be developed in the Python Django web application framework. The processes are expected to be executed remotely through a well-defined and established RESTful framework using the Django Rest Framework as a backend. Below is an image of the general ecosystem on which the AutoLocalize application should be deployed. The red-circled components are the expected deliverables of the AutoLocalize project where they are all automated within the ecosystem.



**Figure:** Localization Process: Potential Areas for Automation/AI Usage.

# Detailed Description of the Deliverables: The Main Software Components

* **COMP-1: Onboarding and Initialization Setup for Live Websites’ Localization:**  
  The workflow of the onboarding and Initialization Setup of Live Websites’ Localization is defined as follows:
  + The source tree and structure of the live website to be localized are registered to enable access to all files in its source tree.
  + This component builds the tree structure of the registered live website.
  + This component builds and maintains 2 options to scrap the registered live website’s HTML content.
    - Option 1: copy all the HTML files in the tree in a dedicated directory in the AutoLocalize filesystem.
    - Option 2: directly extract all localizable content from the registered live website’s HTML files.
  + The registration allows the user to select the source and target languages to be considered in the localization process.
  + Once the registration is submitted, the “Automatic String Detection and Extraction” component starts work.
* **COMP-2: Automatic String Detection and Extraction:**  
  The workflow of the String Detection and Extraction component is defined as follows:
  + This component is responsible for scraping the registered websites’ source tree and pulling all the copies of their original HTML localizable hard-coded content along with their JSON translation resource files in a dedicated folder in the AutoLocalize application.
  + This component extracts the text of each HTML element of the web pages and replaces it with localization resource string placeholders (keys) while sending and storing the original strings in a JSON translation resource file with their respective keys.
  + This component must create copies of the original HTML files and apply its procedures on the copies and not on the originals. The original HTML files must never be affected at this stage.
  + All the HTML files copied along with their corresponding JSON translation resource files must be stored in a dedicated folder in the AutoLocalize web application file system. The HTML file and its JSON resource file must have the same unique names to easily identify which JSON file belongs to which HTML file in the dedicated folder.
  + The component must rebuild to localize the HTML files to accept JSON translation resource strings in their corresponding key placeholders’ positions.
* **COMP-3: Automatic Localization and Translation Workflow Tool for Live Websites:**  
  The workflow of this component is defined as follows:
  + To trigger an automatic rebuilding or adjustment process once a change or update occurs in a JSON translation resource file.
  + To connect to a Translation Management Server (TMS) via API and send all texts extracted from the JSON files for translation.
  + To receive translated texts from the TMS via API and replace the texts in the JSON files with their respective translated texts.
  + To update the live websites’ HTML files with the fully translated HTML files once the “Automatic Localization and Translation Quality Evaluation and Review Process” is done.
* **COMP-4: Automatic Localization and Translation Quality Evaluation and Review Process:**
  + To build the workflow responsible for the handling of the version control of the texts being processed for translation.
  + To assign version control categorization to the texts being processed for translation such as “raw”, “translated”, “reviewed”, “approved”, and “rejected”.
  + To allow multiple users to review the translations based on a well-structured permission and role management mechanism in the Django permissions framework.
  + To allow multiple users to make changes to translations and change the category of the text being reviewed with the defined categories.
  + To enable collaboration among users on the Quality Evaluation and Review Process.
* **COMP-5: Continuous Localization Workflow: Automatic Multilingual Build Generation and Screenshot Collection for Quality Assurance (QA)/Linguistic Review Process:**
  + To build an automatic pipeline through which the application is listening to changes on the live websites’ HTML files such as textual changes or image changes to trigger a new build of localization files.
  + To trigger and automate the update of JSON translation resource files that are affected by the updates on the live websites.
  + To capture screenshots for QA and Linguistic Review (LR).
  + To trigger and enable the QA and LR process which also involves user collaborations and feedback mechanisms.
  + To implement the automation of the change/update listeners on the live websites and execute the necessary actions once change/update is detected.
  + To implement the constant running of the listeners and triggers on the server where the AutoLocalize application is deployed.

AutoLocalize Requirements Specification

# 1. Introduction

## 1.1 Purpose

The AutoLocalize platform is designed to automate the process of localizing and translating websites in real-time, utilizing advanced machine translation technologies integrated through Application Programming Interfaces (APIs). This document outlines the technical specifications and requirements necessary for the development, deployment, and maintenance of the platform, ensuring that all functional and non-functional aspects are clearly defined and understood by the project team and stakeholders.

## 1.2 Document Conventions

This document adheres to the standard specification format which includes comprehensive definitions of terms, acronyms, and abbreviations used throughout. Each requirement is uniquely identified to aid in traceability and future references.

## 1.3 Intended Audience and Reading Suggestions

The primary audience for this document includes the project's software developers, project managers, and technical leads. Additionally, it serves as a reference for stakeholders who are involved in overseeing the project's progress. Readers are advised to begin with the introduction for a project overview and then proceed to specific sections as relevant to their needs.

## 1.4 Project Scope

The scope of the AutoLocalize platform encompasses the development of a comprehensive system capable of localizing and translating online services in multiple languages (Japanese, English, Simplified Chinese, Traditional Chinese, and Korean) and is designed to integrate with various types of websites including e-commerce platforms, blogs, content management systems, and enterprise systems. The platform aims to streamline the localization process, reduce manual intervention, and maintain high-quality translation standards.

# 2. Overall Description

## 2.1 Product Perspective

The AutoLocalize platform will be a part of a larger ecosystem of localization tools and services, interfacing with existing content management systems (CMS) and leveraging the Django REST Framework for backend operations. The system is designed to function independently while also being capable of seamless integration with external systems via well-defined APIs.

## 2.2 Product Features

* **Automatic String Detection and Extraction:** Identifies and extracts translatable content from websites automatically.
* **Real-time Localization and Translation Workflow:** Manages the translation process dynamically as content changes.
* **Quality Evaluation and Review Processes:** Ensures the accuracy and appropriateness of translations through rigorous review mechanisms.
* **Continuous Localization and Multilingual Build Generation:** Updates and generates localized content continuously as part of an ongoing process.

## 2.3 User Classes and Characteristics

The platform will cater to various user roles including administrators who manage website registrations for onboarding, content creators who monitor content changes, and translators who engage in the post-editing, proofreading, and review processes of translated content. The system will provide tailored interfaces for each user type, emphasizing ease of use and minimal technical complexity.

## 2.4 Operating Environment

AutoLocalize will be developed using the Python programming language with Django as the web application framework. The platform is intended to be deployed in a cloud-based environment to ensure scalability and reliability, with a responsive user interface that is compatible with various device screens.

## 3. System Features

## 3.1 Onboarding and Initialization Setup

**Description:**

This feature facilitates the initial setup and registration of websites for localization, including the automation of content scraping and initial analysis of the website structure to ensure comprehensive content detection and localization.

**Functional Requirements:**

**FR1:** The system must provide a user-friendly interface for website registration, allowing users to define and store the source tree and structure of the website.

**FR2:** Two options must be available for scraping content:

* Direct copying of HTML files from a provided URL of a live website.
* Direct copying of HTML files from a base directory path of a locally available website files.

**FR3:** The initiation of the localization process must be automated upon completion of the registration, triggering the string detection component.

## 3.2 Automatic String Detection and Extraction

**Description:**

This component is responsible for the intelligent extraction of localizable text from registered websites, replacing original content with placeholders for translated text, and organizing the data into manageable units for further processing.

**Functional Requirements:**

**FR4:** Extract text content from HTML elements and replace them with localization placeholders, while storing original strings in a structured format.

**FR5:** Ensure that all extracted data is systematically stored in a dedicated directory within the platform's filesystem, maintaining a clear association between original files and their translations.

## 3.3 Translation Workflow Tool

**Description:**

This tool manages the end-to-end translation workflow, interfacing with a Translation Management Server (TMS) to handle the submission of texts for translation and the integration of returned translations back into the website content.

**Functional Requirements:**

**FR6:** The system must dynamically respond to changes in translation files automatically updating content as new translations are available.

**FR7:** Upon completion of the translation process, the system should ensure that updated content is accurately reflected on the live website following a final quality assurance process.

## 3.4 Quality Evaluation and Review

**Description:**

This component establishes a comprehensive process for the evaluation and revision of translated content, ensuring it meets quality standards and is contextually appropriate before final approval and publication.

**Functional Requirements:**

**FR8:** Implement robust version control mechanisms for all translations to track changes and maintain historical data for audits.

**FR9:** Support a multi-tier review system that allows designated users to approve, reject, or suggest modifications to translations based on predefined roles and permissions.

## 3.5 Continuous Localization Workflow

**Description:**

This component automates the continuous integration of changes in the website content ensuring that all updates are promptly localized and translated. It also ensures that the translations are up-to-date and are reflected in the most current version of the website.

**Functional Requirements:**

**FR10:** Detect and automatically process textual or structural changes on the website triggering localization updates as needed.

**FR11:** Implement mechanisms to automatically capture and utilize screenshots for quality assurance and linguistic reviews facilitating accuracy and feedback.

# 4. External Interface Requirements

## 4.1 User Interfaces

* The system will provide a comprehensive web-based dashboard that allows users to manage projects, view progress, and interact with the system tools.
* The interface will be designed to be intuitive and accessible for users with varying levels of technical expertise. This ensures that all functionalities are easily navigable.

## 4.2 Hardware Interfaces

* The platform is designed to be compatible with standard web server hardware configurations to ease deployment and maintenance.

## 4.3 Software Interfaces

* Utilize Django REST Framework for seamless API integrations, facilitating communication between the platform and external systems such as websites and the TMS.
* Ensure compatibility with major web application platforms to broaden the application’s usability across different types of websites.

# 5. Other Nonfunctional Requirements

## 5.1 Performance Requirements

* The platform is designed to handle multiple concurrent localization requests without degradation in performance, aiming for response times under two seconds for user interactions and system processes.
* System scalability to handle increased load and user base without significant changes to the infrastructure.

## 5.2 Security Requirements

* The system will implement industry-standard security measures, including data encryption, secure API interactions, and comprehensive access controls to protect sensitive information.
* Regular security audits and updates will be part of the maintenance schedule to address new vulnerabilities.

## 5.3 Quality Attributes

* Scalability is a core attribute allowing the system to accommodate growth in the number of users and volume of data without performance compromise.
* Maintainability and modular design will be emphasized to facilitate updates and enhancements without disrupting existing functionality.

AutoLocalize Architectural Diagram

A diagram of a software application

Description automatically generated

# Description of Components

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Component** | **Description** | **Objects** | **Requirement** | **Input** | **Output** | **Target** | **SLA** |
| **Dashboard** | The Dashboard serves as the central hub for users to manage and oversee various localization projects. It provides tools for tracking project progress, accessing detailed reports, and managing project settings, all through an interactive, web-based interface designed for ease of use and efficient navigation. | **Web Framework**: Django  **Front-end**: HTML, CSS, and JavaScript for dynamic content rendering  **API**: Django Rest Framework (DRF) services for backend communication | FR1, FR2, FR3 | User commands through graphical interface | Visual feedback on project status, reports, and project settings | To provide stakeholders with a comprehensive tool for monitoring and managing localization projects, thereby enhancing decision-making and project oversight. | 99.9% system uptime, average response time of less than 2 seconds for loading data and updates. |
| **User Management** | This component is crucial for maintaining the security and integrity of the platform. It manages user authentication, role assignments, and permission configurations, ensuring that each user accesses only the appropriate level of data and functionality according to their role within the organization. | **Library**: Django Authentication for secure login processes  **Database**: PostgreSQL to store user credentials and role data  **Security**: Implementation of authentication and permission mechanisms for robust access control | FR4, FR5 | Login credentials and role modification requests | Confirmation of login status and role updates; access control adjustments as needed | To secure the platform by enforcing role-based access controls, thus protecting sensitive data and ensuring operational integrity. | 99.99% uptime with a focus on immediate recovery from failures, ensuring continuous access control. |
| **Onboarding and Initialization** | Handles the initial setup processes for new websites seeking localization services. This includes registering website details, scraping the initial content, and preparing the system for ongoing localization activities. The module is designed to facilitate a smooth and automated setup process, reducing manual effort and potential errors. | **Framework**: Django for server-side operations  **Scraper**: Scrapy or Beautiful Soup library for effective HTML content scraping  **API**: DRF APIs for managing project setup and data ingestion | FR6, FR7 | Website URLs, specific configurations for scraping | Confirmation messages indicating successful setup, error logs for troubleshooting | To streamline the onboarding of new localization projects, ensuring a quick start and reducing initial barriers for users. | 98% uptime with a recovery time objective (RTO) of less than 5 minutes for setup processes. |
| **String Detection and Extraction** | This component is designed to automate the detection and extraction of translatable text from websites. It utilizes sophisticated pattern recognition algorithms to identify text elements that require translation and replaces them with localization placeholders. This process is essential for preparing the text for the translation workflow. | **Library**: Scrapy/BeautifulSoup  **Database**: Postgres Database for storing and retrieving string keys and text  **Data Handling**: JSON for organizing and transferring data between processes | FR4, FR5 | HTML content of registered websites | Localizable text replaced with placeholders and associated keys stored in a JSON files and Database | To automate the preparation of website content for translation, enhancing efficiency and accuracy in the localization process. | 99% uptime, with a focus on maintaining over 95% accuracy in text detection and extraction. |
| **Workflow Management** | Manages the core translation workflow, including the synchronization of translated content updates and interaction with the Translation Management System (TMS). This module ensures that changes in content are accurately reflected in the localized versions and that all translations are up to date. | **API**: Custom DRF API for communication with TMS  **Scheduler**: Cron jobs for monitoring content updates  **Data Sync**: DRF API for synchronizing content across systems | FR6, FR7 | Updated content files, requests for translation | Synchronized content files, updated translations | To manage and streamline the translation process, ensuring timely updates and maintaining consistency across localized versions. | 99% uptime with transaction response times not exceeding 10 seconds for synchronization operations. |
| **Quality Control** | Dedicated to maintaining the high standards of translation quality, this module involves various stakeholders in the review and approval process of translated content. It employs a systematic approach to version control and audit trails, ensuring that all modifications are tracked and reversible. | **Tool**: Django Object Relational Mapper (ORM) to map and store updates and decisions on translation versions  **Database**: PostgreSQL for storing version histories and review records  **Permissions**: Django permissions framework for managing access to review functions | FR8, FR9 | Translations awaiting approval, review inputs | Approved translations, records of review sessions | To ensure that all translations meet the set quality standards and are appropriate for their respective markets before being published. | 99.8% uptime with less than 0.5% error rates in the handling and storage of review data. |
| **DRF API Gateway** | Serves as the intermediary for all external API communications, particularly between the AutoLocalize platform and external TMS. It ensures secure data exchange by implementing robust security protocols and managing the flow of information to prevent unauthorized access or data breaches. | **API**: Custom DRF API for communication with TMS  **Scheduler**: Cron jobs for monitoring content updates  **Data Sync**: DRF API for synchronizing content across systems | All | API calls to and from onboard websites and the TMS, | Secure and validated API responses to and from the onboard websites and the TMS | To facilitate secure, efficient, and reliable API communications between internal and external systems, enhancing integration capabilities. | 99.99% uptime, with API response times kept below 5 seconds for all transactions. |
| **TMS Integration** | Manages the direct interactions with the Translation Management System, which includes sending texts for translation and receiving the translated outputs. This component is crucial for the seamless integration of external translation services into the localization workflow. | **API**: External TMS API for seamless integration  **Data Format**: JSON for efficient data handling  **Networking**: Secure network configurations for reliable data transmission | FR6, FR7, FR10, FR11 | Source texts needing translation | Translated texts ready for integration into websites | To efficiently manage the translation lifecycle by interfacing directly with external translation services, ensuring timely and accurate translations. | 99% uptime with average response times of less than 5 seconds for sending and receiving data. |
| **Database Management** | Responsible for all data storage and management aspects of the platform, this component ensures that user data, project metadata, and translation memories are securely stored, easily accessible, and reliably managed within the system. | **Database System**: PostgreSQL for robust data handling  **Schema**: Django ORM for seamless data integration and manipulation  **Backup**: Scheduled backups with crontab systems for data availability | All | Data queries, data storage requests | Query results, confirmation of data storage | To provide a comprehensive and reliable data management solution that supports all aspects of the localization platform. | 99.95% uptime, with a focus on maintaining data integrity and providing continuous data access. |
| **File Storage** | Manages the storage and retrieval of critical files related to website content and translations. This component is designed to handle large volumes of data efficiently, ensuring that files are available and up to date for the localization process. | **Storage System**: Django File System framework  **File Management**: Django for handling file operations  **Security**: Encryption and secure access controls for data protection | FR4 – FR11 | Files to be stored, retrieval requests | Stored files, retrieval confirmations | To efficiently manage the storage and retrieval of website and translation files, ensuring data is secure and readily available for localization tasks. | 99.9% uptime, with a focus on minimizing data loss and ensuring quick retrieval times. |

### Explanation of Table Columns:

* **Component**: Names the specific parts or modules of the system as designed in the architecture indicating a specific function or group of functions.
* **Description**: Provides a detailed explanation of what the component does, its role in the system, and how it interacts with other parts of the system.
* **Objects**: Lists the specific technologies, frameworks, libraries, APIs, and data structures used within each component. It also highlights the technical environment and tools employed to fulfill the component's functions.
* **Requirement**: Identifies the specific functional requirements that are related to the component as designated in the requirements specification.
* **Input**: Describes what data or user actions are received by the component to initiate its processes, detailing the type of information or triggers needed for the component to operate.
* **Output**: Specifies what the component produces, whether it be data, a physical product, or a change in the system's state, providing clarity on the results of the component's activities.
* **Objective**: Explains the purpose or goal of the component within the overall system, detailing why the component is necessary and what it aims to achieve in the context of system functionality.
* **SLA**: Defines the Service Level Agreement terms for the component, which are the agreed standards for performance and reliability that the component must meet. It ensures the operation of the component within the necessary parameters to satisfy user expectations and system requirements.