Cyber Whale

SaaS Data Enrichment Application

Internship Project Report

Internship Duration:

September 1 - September 27, 2024

Supervisor:
Vladimir Stajilov, CTO, CyberWhale

Interns:
Dmitrii Belîh, Software Engineer
Mihaela Catan, Software Engineer
Sabina Popescu, Software Engineer

Chișinău, 2024

Abstract

This report details the development and implementation of a SaaS Data Enrichment Application designed to enhance organizational efficiency by automating the management and enrichment of datasets related to individuals and organizations. The application addresses prevalent challenges in data handling, such as the lack of critical information, labor-intensive manual processing, and integration issues. By utilizing a robust technological stack, including Next.js, Strapi, and various APIs, the platform allows users to upload datasets, match columns, and enrich data dynamically, all while providing upfront cost estimations to facilitate informed decision-making. The project underscores the importance of seamless user authentication, effective data management, and the potential for future scalability and feature enhancement, making it a valuable tool for data analysts, marketing professionals, HR managers, and business executives.

Keywords: API, Application, Data Enrichment, Data Management, Efficiency

Contents

Introduction

0.1 Project Overview

The proposed SaaS application represents a transformative approach to the management and enhancement of datasets relating to organizations and individuals. This platform is strategically designed to automate and streamline the process of data upload, column matching, and enrichment, addressing significant inefficiencies currently faced by various business sectors in data handling. By integrating technology with user-centric design, the application aims to deliver substantial improvements in data quality, operational efficiency, and decision-making accuracy across multiple organizational functions.

The primary aim of this SaaS application is to enhance organizational efficiency by providing an advanced platform for managing and enriching datasets related to individuals and businesses. This platform simplifies the complex process of data management by automating the enrichment of raw data, thereby ensuring that organizations can make more informed decisions based on accurate and comprehensive datasets.

Organizations today face several challenges regarding data management:

- Many datasets lack critical information, which can lead to suboptimal decision-making[1]. The application automates the enrichment process, filling in missing data and correcting inaccuracies to improve the overall quality and usability of the data.

- Manual data processing is often labor-intensive and error-prone[2]. This application provides tools for automating data uploads, matching columns to predefined fields, and enriching data, which streamlines operations and minimizes human errors.

- Businesses often struggle to integrate enriched data seamlessly into their existing systems[3]. This platform outputs data in a standardized JSON format, easing the integration process and ensuring compatibility with various business intelligence tools.

- Data enrichment processes can be expensive and often come with hidden costs[4]. This platform includes a transparent cost estimation feature that allows users to see the potential costs upfront, enabling better budget management.

The application is designed for a broad range of users who rely heavily on accurate and extensive data, including:

- **Data Analysts and Scientists:** These professionals can utilize enriched data for deeper analytical insights and more accurate forecasting.

- **Marketing Professionals:** Enhanced customer data enables targeted marketing strategies and personalized customer interactions.
- **HR Managers:** Comprehensive employee data aids in efficient workforce management and organizational planning.
- **Business Executives:** Executives and strategists can leverage enriched data for more informed decision-making and strategic planning.

The application provides a robust suite of data management functionalities:

- **Dataset Upload and Preprocessing:** Users can upload data in commonly used formats such as CSV and Excel. The platform prepares these datasets for further processing.
- **Column Matching and Data Mapping:** Users can intuitively match dataset columns to predefined fields via the user interface, setting the stage for precise data enrichment.
- **Data Enrichment:** Leveraging API calls to different services, the platform automatically enriches data, adding valuable context and filling in gaps.
- **Cost Estimation and Management:** The platform offers upfront cost estimations for data enrichment, enhancing budget management by providing clear financial projections before the process begins.
- **Data Storage and Retrieval:** Once enriched, data is securely stored and readily accessible, allowing users to download or integrate it with other systems as needed.

0.2 Objectives

This project aims to create a feature-rich SaaS platform that enables users to manage and enhance datasets related to organizations or individuals efficiently. The core functionality focuses on dataset uploads, column matching, data enrichment via external API calls, and transparent cost estimation. Below is a detailed breakdown of the objectives.

User Authentication:

- Implement secure user authentication using NextAuth to allow users to sign up, log in, and manage their profiles.
- Ensure that only authorized users can access critical features of the platform, such as dataset upload and enrichment functionalities.

Dataset Upload:

- Provide functionality for users to upload datasets in commonly used formats, such as CSV and Excel.
- Support datasets containing information like organization names, individual names, email addresses, roles, and phone numbers.
- Ensure the datasets are parsed upon upload for a smooth and efficient user experience.

Data Preview and Column Matching:

- After a successful upload, allow users to preview the uploaded dataset and match columns with predefined fields (e.g., Name, Email, Role, Company) to ensure data integrity and compatibility.
- Implement a user-friendly interface that allows column matching, which is critical for the subsequent data enrichment process.
- Provide error handling and feedback to assist users with potential mismatches or errors in uploaded datasets.

Data Enrichment Using API Calls:

- Enable users to select specific fields (e.g., Email, Role, Company, Phone Number) for data enrichment.
- Leverage external APIs (such as LinkedIn, Veriphone, and other data providers) to automatically enrich the selected fields. The enriched data may include details like phone number validation, company information, and role verification.
- Ensure that the application can handle various external API responses and integrate enriched data dynamically without refreshing the page.
- Generate a standardized JSON schema for the data enrichment process to ensure consistency in the output, making the enriched data easily integratable into other systems or workflows.

Cost Estimation:

- Implement a feature that calculates and displays an estimated cost for the data enrichment process. The cost is calculated based on the number of fields selected for enrichment, the volume of the data, and API usage.
- Provide a user-friendly cost summary, allowing users to review and confirm the estimated costs before initiating the enrichment process. This will enable better decision-making and financial transparency.
- Allow users to adjust their enrichment requests based on the cost, helping them manage their data processing budgets efficiently.

Final Data Storage:

- Upon confirmation, securely store the enriched dataset in the backend using Strapi, ensuring data integrity and security.
- Provide users with the option to download the enriched dataset in CSV or Excel format for future use or further processing.
- Ensure that users can also view the enriched data within the platform, allowing them to analyze or retrieve specific information directly from the user interface.

Dynamic Data Handling:

- Enable dynamic data handling by adding new columns (enriched through API calls) to the dataset without requiring a page refresh or disrupting the user flow.
- Ensure that the platform can handle large datasets and multiple API responses, dynamically integrating the new data as it becomes available from external sources.

0.3 Technological Stack

The technologies used in this project ensure efficient development, seamless functionality, and smooth deployment across both frontend and backend services.

Frontend: Next.js, NextAuth, Ant Design, TypeScript

- **Next.js (v14.2.7):** Next.js is used to build the dynamic frontend, providing server-side rendering for faster data-heavy pages like dataset upload, column matching, and dataset enrichment. It also manages API routes for seamless communication between the frontend and backend, enabling real-time updates during the data enrichment process[5].
- **NextAuth (v4.13.0):** NextAuth is implemented for secure user authentication, ensuring only authorized users can access key features like uploading datasets and performing data enrichment. It handles login, sign-up, and session management[6].
- **Ant Design (v5.9.0):** Ant Design provides ready-made UI components (tables, forms, modals) to build a professional, user-friendly interface for tasks like column matching, dataset uploads, and data enrichment actions[7].
- **TypeScript (v5.2.2):** TypeScript is used across the project to ensure type safety, especially when handling complex data structures (datasets, API responses). It improves code reliability and reduces potential errors in data management processes[8].

Backend: Strapi

Strapi (v4.25.7): Strapi serves as the backend CMS, managing the storage and retrieval of datasets and enrichment metadata. It provides API endpoints that the frontend uses to fetch and display datasets, as well as store results after data enrichment[9].

Containerization: Docker

- **Docker (v24.0.6):** Docker is used to containerize the entire application, ensuring consistent development and deployment across environments. It isolates the frontend, backend, and database services, making the setup and deployment process more streamlined[10].
- **Docker Compose (v2.22.0):** Docker Compose orchestrates multiple containers for the frontend, backend, and other services, ensuring all components run together seamlessly during development and deployment[11].

System Architecture

0.4 Main Processes

The main processes are integral to delivering the primary functionalities of the application, directly affecting the user experience and service delivery.

User Authentication

- **Process:** Secure login and session management via Google authentication.
- **Purpose:** Facilitates secure access to user accounts, ensuring that user data is securely handled and privacy is maintained.

Dataset Upload

- **Process:** Uploading and temporary storage of datasets in CSV or Excel formats.
- **Purpose:** Allows users to input initial data, which is essential for generating insights and value from the application.

Data Preview and Column Matching

- **Process:** Parsing uploaded datasets and enabling automatically to align dataset columns with predefined fields.
- **Purpose:** Ensures accurate setup of data for subsequent processing, crucial for maintaining data integrity and relevance.

Data Enrichment

- **Process:** Selection of fields for AI-driven enrichment and application of customized prompts.
- **Purpose:** Augments the dataset with additional valuable information or insights, significantly enhancing the dataset's utility.

Cost Estimation and User Confirmation

- **Process:** Automated estimation of enrichment costs presented to the user for approval.
- **Purpose:** Provides transparency in the enrichment process, allowing users to make informed decisions regarding service usage.

Final Data Storage

- **Process:** Storage of the enriched dataset within Strapi, with options for users to download or view.
- **Purpose:** Delivers the final enriched product to the user, ensuring availability and accessibility of processed data.

0.5 Secondary Processes

Secondary processes support the main operations or enhance the application's functionality and user experience but are not directly involved in the core data processing tasks.

Error Handling and Logging

- **Process:** Systematic handling and logging of errors.
- **Purpose:** Improves system reliability and maintainability, offering insights into operational performance and issues.

User History Management

- **Process:** Management of user history and past uploads.
- **Purpose:** Enhances user satisfaction and engagement by allowing personalization and customization.

0.6 Functional Requirements

The functional requirements of the SaaS application define the core functionalities that the system must provide to users. These requirements are essential for the system to perform its intended tasks effectively and are detailed below based on the project's implementation and codebase structure.

User Authentication

The system utilizes `authService.ts` and `user.service.ts` to manage user authentication via Google. This approach enhances security and streamlines the login process:

- **User Registration and Login:** Users can register and log in using their Google accounts, allowing for a secure and swift authentication process without the need for traditional username and password management.
- **Session Management:** The application maintains user sessions to ensure continuity and security during its usage, leveraging the robust security standards of Google authentication.

Dataset Upload

Implemented through `FileUploader.tsx` and `uploadService.ts`, this feature allows users to upload datasets in various formats:

- **File Upload:** Support for uploading CSV and Excel files.
- **Initial Data Handling:** Temporary storage and basic processing of uploaded files to prepare for data preview.

Data Preview and Column Matching

- **Data Display:** Datasets uploaded are parsed and displayed in a tabular format.
- **Column Matching:** Server matches dataset columns with predefined schema fields like Name, Phone, and Linkedin.

Data Enrichment

The application enriches data through external APIs, managed by `EnrichmentButton.tsx` `DataTable.tsx`, `DisplayData.tsx` , `linkedInEnrichmentService.ts`, and `phoneEnrichmentSer`

- **Selection of Enrichment Fields:** Users select which data fields to enrich.
- **Enrichment Process:** Integration with APIs to enrich data based on user selection.

Cost Estimation

Before data enrichment, the system estimates costs:

- **Cost Calculation:** The application calculates the estimated cost based on data volume and fields selected for enrichment.
- **User Confirmation:** Cost estimates are displayed to users for confirmation before proceeding with the enrichment.

0.7 Non-Functional Requirements

Non-functional requirements of the SaaS application focus on the system standards and qualities that enhance its functionality, such as performance, security, and usability. These are critical for ensuring the system's reliability, efficiency, and user satisfaction.

Usability

The system is designed with a focus on user experience:

- **User Interface:** Clean, intuitive interfaces ensure easy navigation and usability.
- **Accessibility:** Responsive design accommodates various devices and screen sizes.

Maintainability

Maintainability is key to the system's long-term effectiveness:

- **Code Modularity:** Code is organized into modules for easier maintenance and updates.
- **Documentation:** Comprehensive documentation supports future development and maintenance efforts.

Containerization

Docker is used to ensure consistent environments across different deployment stages.

0.8 System Modelling

This section outlines the core functionalities and workflows of the SaaS application, designed to handle dataset uploads, column matching, and data enrichment. Below are described the key workflows in the system: User Authentication, File Upload, Column Matching, Column Enrichment, and Data Storage Download.

User Authentication

The application implements secure user authentication using Google OAuth. This ensures that only authorized users can access the system. The authentication flow is as follows:

- **User initiates login via OAuth:** The process begins when a user selects to log in using their Google account.
- **Redirect user to Google OAuth login page:** The system redirects the user to Google's login page for credential input.
- **User completes login:** The user enters their credentials on Google's login page.
- **Google returns authentication token:** Upon successful login, Google returns an authentication token to the application.
- **Store token and log user in:** The token is stored securely, and the user is logged into the system, allowing access to further features.

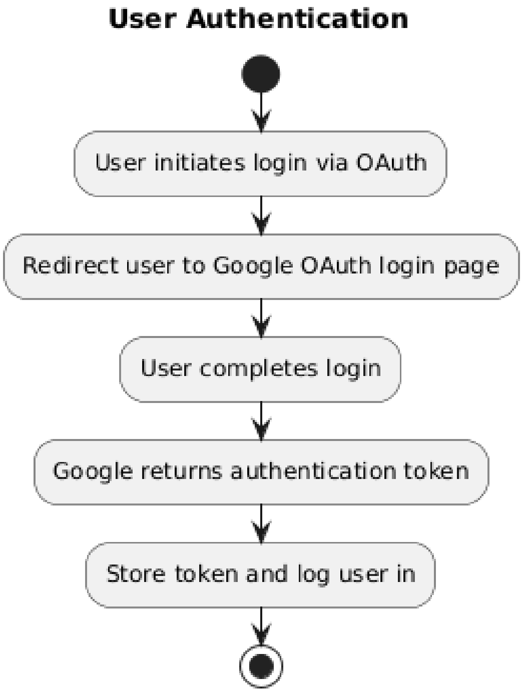
User Authentication
User initiates login via OAuth
Redirect user to Google OAuth login page
User completes login
Google returns authentication token
Store token and log user in

Figure 1: User Authentication Activity Diagram

File Upload

The file upload functionality allows users to upload datasets (in CSV or Excel format), which is essential for the column matching and data enrichment processes. The flow for file upload includes the following steps:

- **User navigates to the file upload page:** The user selects the option to upload a file within the application.
- **User selects a file for upload:** The user chooses a dataset from their local storage.
- **System receives uploaded file:** The system accepts the uploaded file and prepares it for validation.
- **System validates file format:** The system checks if the uploaded file is in the correct format (CSV or Excel). If the file is valid, the process continues. If not, the system returns an error message to the user.
- **Proceed to store file:** Upon successful validation, the file is temporarily stored in preparation for the column matching process.

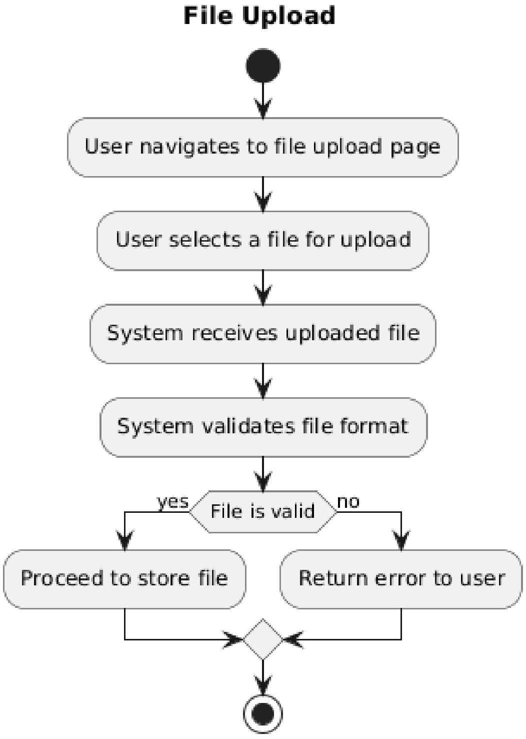
File Upload
User navigates to file upload page
User selects a file for upload
System receives uploaded file
System validates file format
File is valid
yes
no
Proceed to store file
Return error to user

Figure 2: File Upload Activity Diagram

Column Matching

The column matching functionality provides users with the ability to map their uploaded dataset columns to predefined fields such as Name, Email, or Role. AI assistance suggests possible column mappings for a better user experience. The workflow is as follows:

- **System analyzes file content:** The system parses the uploaded file and reads the column headers and data.
- **AI suggests possible column mappings:** Using AI, the system suggests the most appropriate mappings between the file's columns and the predefined fields.
- **User reviews and proceeds with mappings:** The user reviews the suggested mappings and confirms the column matching.

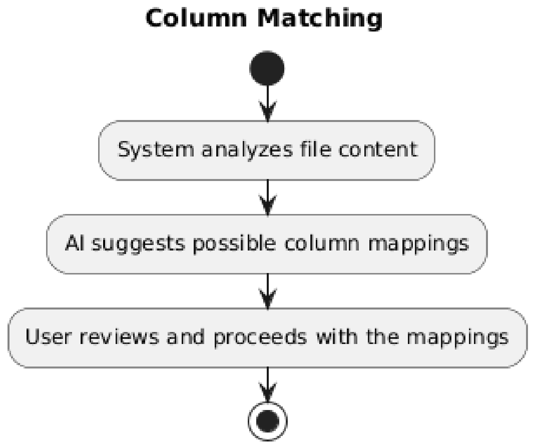
Column Matching
System analyzes file content
AI suggests possible column mappings
User reviews and proceeds with the mappings

Figure 3: Column Matching Activity Diagram

Column Enrichment

This feature allows users to enrich the data in their uploaded dataset by retrieving additional information (e.g., LinkedIn profiles, phone verification) from external APIs. The process is as follows:

- **User selects columns for enrichment:** The user selects which columns in the dataset need enrichment (e.g., phone number, LinkedIn ID).
- **System calculates total cost:** The system estimates the total cost for data enrichment based on the selected fields and available API usage limits.
- **System prepares data for enrichment:** The selected columns are packaged and prepared for the API calls.
- **System calls external enrichment API:** The system sends the relevant data to external APIs to retrieve enriched information (e.g., Veriphone for phone verification, LinkedIn API for user details).
- **Receive enriched data from API:** The system receives the enriched data and processes it for integration with the original dataset.
- **Compile enriched data:** The enriched data is merged with the existing dataset and prepared for storage and further processing.

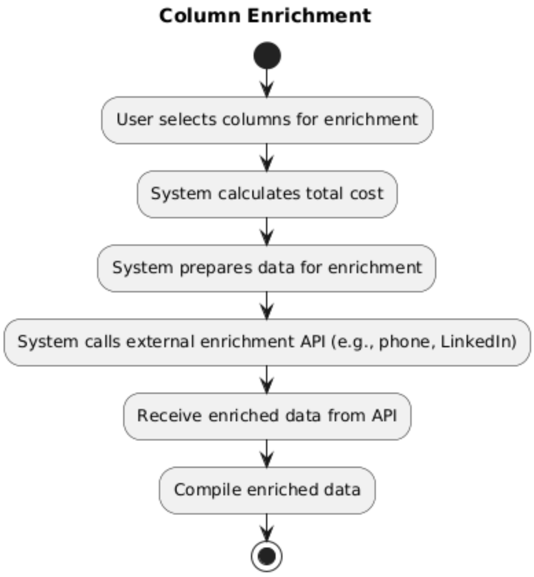
Column Enrichment
User selects columns for enrichment
System calculates total cost
System prepares data for enrichment
System calls external enrichment API (e.g., phone, LinkedIn)
Receive enriched data from API
Compile enriched data

Figure 4: Column Enrichment Activity Diagram

Data Storage and Download

Once the data enrichment process is complete, the enriched dataset is stored, and users are given the option to download the updated dataset for their own use.

- **System stores enriched data in the database:** The final enriched dataset is securely stored in the system's database.
- **System prepares the dataset for download:** The system prepares the dataset in the desired format (e.g., CSV or Excel).
- **System prompts user to download the enriched dataset:** The user is notified that the dataset is ready for download.
- **User downloads the dataset:** The user can download the enriched dataset, which contains the original data along with any additional information retrieved during the enrichment process.

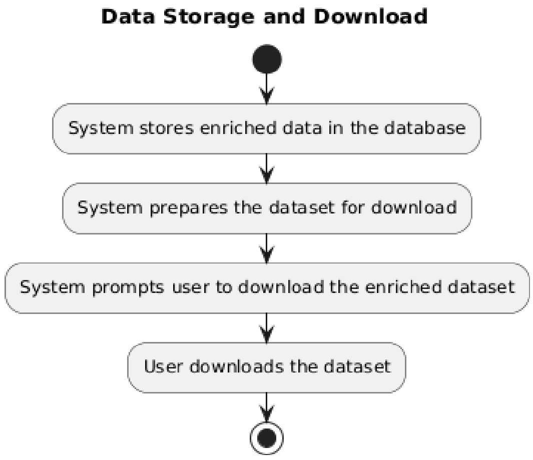
Data Storage and Download
System stores enriched data in the database
System prepares the dataset for download
System prompts user to download the enriched dataset
User downloads the dataset

Figure 5: Data Storage and Download Activity Diagram

The next diagram outlines the primary use cases for two types of actors: User and Admin. It captures the interactions between these actors and the system, detailing how each actor engages with various features of the SaaS application.

Actors:

- **User:** A regular user who interacts with the system primarily to upload datasets, enrich data, and download results. The user's role is crucial to driving the main functionality of the application.
- **Admin:** The admin has additional privileges compared to regular users, with the ability to manage users within the system. This role focuses on ensuring system maintenance and managing access to the platform.

Use Cases:

- **Login (User):** The user initiates the session by logging into the system through a secure authentication mechanism (e.g., Google OAuth). This is the first step to access the system's core functionality.
- **Upload Dataset (User):** The user uploads a CSV or Excel file containing data related to organizations or people. The system processes the file, preparing it for preview, column matching, and enrichment.
- **Preview and Match Columns (User):** After uploading the dataset, the system displays a preview where the user can review and match the dataset's columns to predefined fields (e.g., Name, Email, Phone). This step is essential before proceeding to enrichment.
- **Select Fields for Enrichment (User):** The user selects specific columns within the dataset that need enrichment. This could involve retrieving additional data such as phone verification or LinkedIn profiles.

- **Receive Cost Estimation (User):** Once fields are selected for enrichment, the system calculates the total cost based on the type and quantity of data to be enriched. The cost is presented to the user before proceeding.
- **Confirm Enrichment (User):** The user reviews the cost estimation and confirms whether they want to proceed with the enrichment process. The system then retrieves enriched data from external APIs.
- **Download/View Enriched Data (User):** Once the enrichment process is complete, the user can download or view the enriched dataset. The enriched data is stored temporarily for download in the desired format (CSV or Excel).
- **Manage Users (Admin):** The admin has special privileges to manage users within the system. This includes adding, removing, or modifying user access, ensuring the system operates smoothly for authorized individuals.

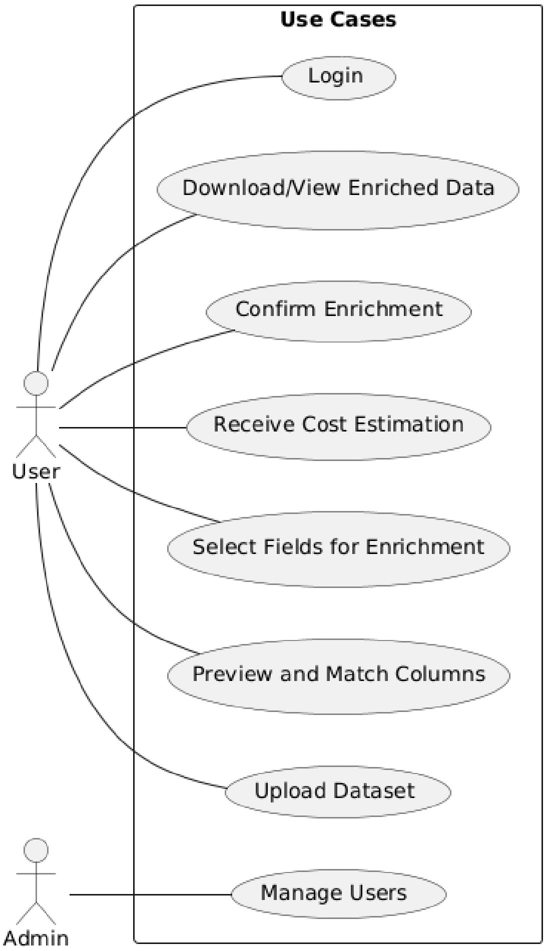

Figure 6: User and Admin Usecase Diagram

The following sequence diagrams illustrate the interaction between the system components, actors (users), and services. These diagrams focus on user authentication, dataset upload, column matching, and the data enrichment process, demonstrating how data flows between the system's frontend, backend services, and external APIs.

User Authentication and Dataset Upload Sequence

This diagram shows the interaction flow for user authentication and dataset upload.

Authenticate User:

- The user starts the authentication process by communicating with the Auth Service.
- The Auth Service validates the user's credentials, typically using an external provider (e.g., Google OAuth).
- Upon successful authentication, the service returns a success message, allowing the user to proceed.

Upload Dataset:

- Once authenticated, the user uploads a dataset (CSV/Excel) through the Upload Service.
- The Upload Service processes the dataset and validates the file.
- Upon successful validation, the dataset is saved in the Dataset Storage.
- This process is the foundation for further steps like column matching and data enrichment.

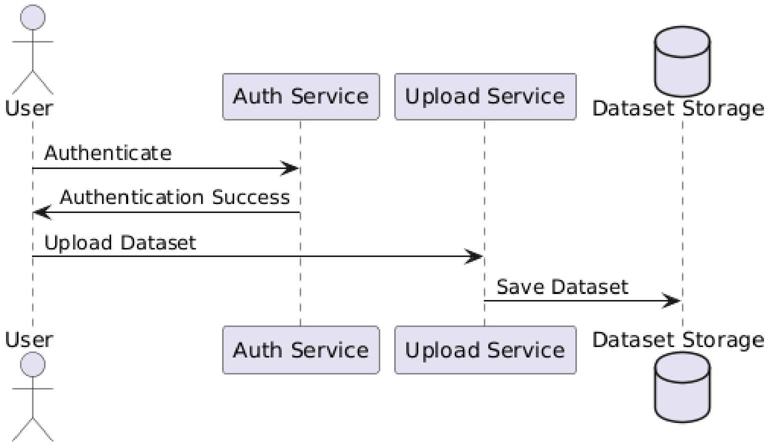

Figure 7: User Authentication and Dataset Upload Sequence Diagram

Column Matching Sequence

This diagram captures the interaction during the column matching phase.

Auto-Match Columns:

- After a dataset is uploaded, the system automatically matches the columns from the dataset with predefined fields (e.g., Name, Email, Role).
- The matched column data is loaded from Dataset Storage and sent to the Data Preview UI for the user's review.

Review Matched Columns:

- The user reviews the suggested column mappings.
- Once satisfied, the user confirms the column matching.

Confirm Column Matching:

- After confirmation, the Together Service stores the finalized column mapping in Dataset Storage.
- This step prepares the dataset for the next stage: data enrichment.

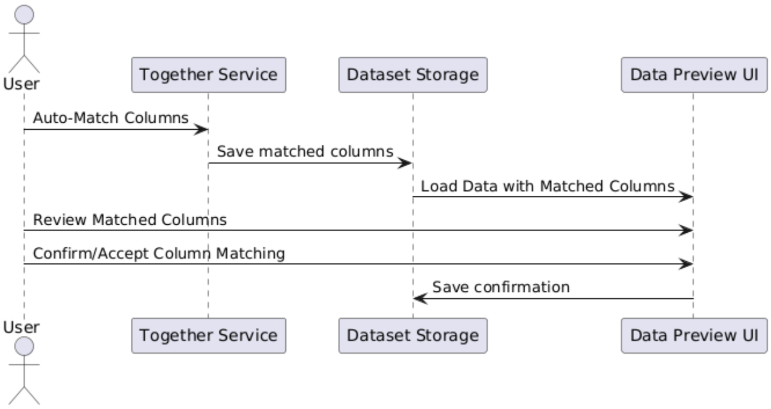

Figure 8: Column Matching Sequence Diagram

Data Enrichment Sequence

This diagram illustrates the interaction during the data enrichment process.

Request Enrichment:

- After column matching is confirmed, the user requests data enrichment via the Data Preview UI.

- The Enrichment Service retrieves the relevant data for enrichment from the Dataset Storage.

Cost Estimation:

- The Enrichment Service calculates the total cost of enrichment based on the selected fields and external API usage.
- The estimated cost is displayed to the user for review in the Data Preview UI.

Proceed with Enrichment:

- If the user agrees to the cost, the system proceeds with enrichment.
- The Enrichment Service connects to external APIs to retrieve enriched data (e.g., phone verification, LinkedIn profile data).

Store Enriched Data:

- The enriched data is stored in the Enriched Data Storage, while the original dataset remains in Dataset Storage.

Download/View Enriched Data:

- The user can download or view the enriched dataset once the enrichment process is complete.
- The final dataset, which includes the enriched fields, is prepared for download via the Data Preview UI.

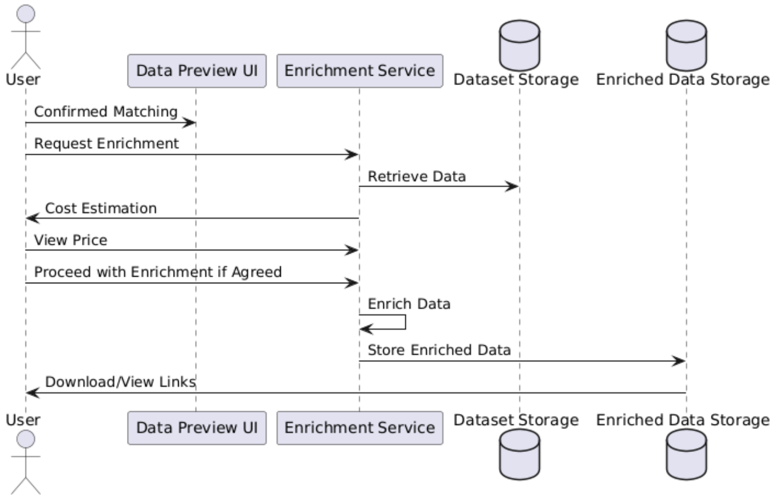

Figure 9: Data Enrichment Sequence Diagram

Technical Documentation

0.9 API Documentation

This section provides a comprehensive overview of the various API endpoints available in our SaaS platform designed for dataset upload and enrichment. Each endpoint is specifically tailored to facilitate user authentication, file management, and data enrichment, thereby ensuring a seamless and efficient user experience.

Endpoint	**Method**	**Request Parameters**	**Response Format**
`/api/auth/google`	GET	(Initiates OAuth flow)	Redirects to Google OAuth login page
`/api/auth/google/callback`	GET	(Handles OAuth callback)	{ "success", "token", "error" }
`/api/upload`	POST	{ "file": File }	{ "success", "fileId", "error" }
`/preview`	GET	None (uses internal AI to analyze dataset)	{ "success", "extractedJson": JSON, "error" }
`/api/enrich`	POST	{ "fileId", "columns": Array }	{ "success", "enrichedData", "error" }
`/api/enrich/phone`	GET	{ "phoneNumbers": Array }	{ "success", "data": Array, "error" }
`/api/enrich/linkedin`	GET	{ "linkedInIds": Array }	{ "success", "profiles": Array, "error" }

0.10 Data Flow

This data flow diagram (DFD) illustrates the movement of data between the main components of the SaaS platform. It highlights how user interactions and system processes interact with various services and storage solutions to achieve the platform's functionality, particularly focusing on user authentication, dataset uploads, column suggestions, and data enrichment.

Components:

User (C): The user is the primary actor, interacting with the system to initiate the processes of authentication, file uploads, and data enrichment. The user starts the authentication process. The user uploads a dataset (CSV/Excel) to the platform. After matching columns, the user requests data enrichment for selected fields.

Google OAuth (C): This component handles user authentication via OAuth. Once the user is authenticated, Google OAuth returns a secure authentication token to the system backend for user session management.

System Backend (C): The system backend is the core component that coordinates the flow of data and manages the overall functionality of the platform. The system backend communicates with various services, processing requests and returning results. It manages user file uploads and stores them temporarily. The backend sends the dataset to an AI analysis service for column suggestions. The backend sends data to enrichment services and processes the responses.

File Storage (C): This component temporarily stores the dataset that users upload. The dataset is saved here temporarily for further processing. The file is sent to the AI Analysis service for suggesting column mappings.

AI Analysis (C): This service analyzes the uploaded dataset to suggest possible column mappings (e.g., Name, Email, Role). Based on the file content, this service suggests appropriate column mappings to the user via the backend.

Enrichment Services (C): These services are external APIs, such as phone verification or LinkedIn data, that provide enriched information for the selected columns. After receiving a request from the backend, these services enrich the provided data and return the results.

Database (C): The Strapi DB is used to store data that has been processed or enriched by the system. It holds the enriched dataset for future use or download by the user.

Data Flow Description:

The process begins when a user initiates the login process. The system sends a request to the Google OAuth service for authentication. Upon successful login, Google returns an authentication token to the System Backend, confirming the user's identity and granting them access to the platform.

Once authenticated, the user can upload a dataset (CSV or Excel file) to the platform. This file is sent to the System Backend, which processes and temporarily stores it in the File Storage. The system then prepares the uploaded dataset for further analysis and column matching.

The uploaded dataset is sent to the AI Analysis service for automated column matching suggestions. The AI engine analyzes the file content and provides suggestions for mapping columns (e.g., Name, Email, Role). These suggestions are sent back to the System Backend, which displays them to the user for review and confirmation.

After the user confirms the column mappings, they can request data enrichment for selected fields (e.g., phone numbers, LinkedIn profiles). The System Backend sends the relevant data to external Enrichment Services, such as APIs for phone verification or LinkedIn data retrieval. These services return enriched data, which the System Backend processes and stores in the Database.

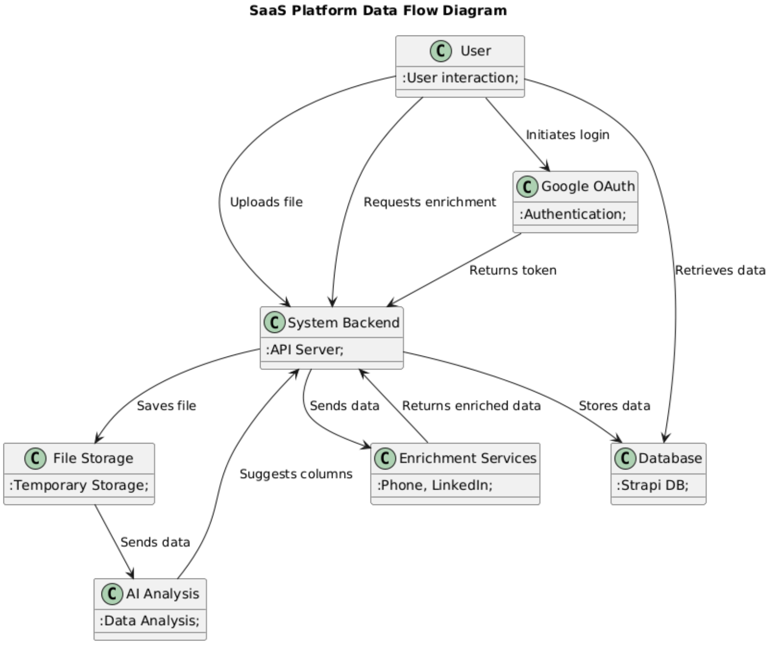

Figure 10: Data Flow Diagram

The enriched dataset is securely stored in the Database (e.g., Strapi DB) for future use. The user can later access, retrieve, and download the enriched dataset in a suitable format (CSV or Excel), allowing for easy integration into other systems or workflows.

0.11 Error Handling

Error handling ensures smooth user experience and robust system operations by properly catching, logging, and handling various potential failures, such as failed file uploads, invalid user input, and network issues.

Error Handling for `fileService.ts`:

- Uses `try-catch` blocks for catching errors during file uploads.
- Logs the error message (`error.response?.data` || `error.message`) if a file upload fails.
- Throws a new `Error` with a user-friendly message, e.g., `"File upload failed"`.
- Ensures that error messages are descriptive and useful for debugging.

Error Handling for `httpService.ts`:

- Includes `try-catch` blocks for managing failed HTTP requests.
- In case of errors like network issues or invalid input, it logs the error and throws a new `Error` with an appropriate message such as `"Network request failed"`.
- Provides specific error feedback based on the error response.

Error Handling for `linkedInEnrichmentService.ts`:

- Catches errors in API calls using `try-catch`.
- Logs and throws meaningful error messages when enrichment fails, such as `"Failed to enrich LinkedIn data"`.
- Error responses from LinkedIn API are logged for debugging.

Error Handling for `phoneEnrichmentService.ts`:

- Implements error catching using try-catch blocks during API interactions.
- If an API call fails, it logs the error and throws a descriptive message like `"Phone number enrichment failed"`.
- Handles rate-limiting errors or invalid phone numbers by providing feedback to the user.

Error Handling for `uploadService.ts`:

- Catches errors during file uploads and metadata saving using `try-catch` blocks.
- Logs error messages, such as `"File upload error"` or `"Metadata save error"`, along with the server's response if available.
- Throws new `Error` objects with user-friendly descriptions to ensure the front-end can display helpful messages to users.

Error Handling for Frontend Components `EnrichmentButton.tsx, FileUploader.tsx, DisplayData.tsx, etc.:`

- File upload components (e.g., `FileUploader.tsx`) validate file size and format before submission. Invalid inputs trigger error messages like `"File is too large"` or `"Unsupported file format"`.
- Error messages from the back-end (such as those from `uploadService.ts`) are caught and displayed in the UI for the user to take corrective actions.
- Components like `DisplayData.tsx` handle potential issues like no data available for display, showing user-friendly fallback messages such as `"No data available to display."`
- Errors during enrichment or API calls trigger loading states and display retry prompts when necessary.

0.12 Project Running

To run your application using docker-compose up, follow these steps:

1. Ensure you have the required files: `Dockerfile` for both the frontend (Next.js) and backend (Strapi) and `docker-compose.yml` file to define how the services (frontend, backend, and database) will run together.
2. Build the containers `docker-compose build`.
3. Start the application `docker-compose up`.
   This will start all services (frontend, backend, and PostgreSQL database) in Docker containers.

   Frontend (Next.js) will be running on http://localhost:3000
   Backend (Strapi) will be running on http://localhost:1337
4. To stop the running containers, use `docker-compose down`.

Future Development and Scalability

0.13 New Features

To keep the system competitive and provide more value to users, several new features can be introduced.

One of the core functionalities of the current system revolves around data enrichment. Currently, the system integrates APIs for phone number verification and LinkedIn enrichment, offering basic information about people and organizations. However, there is significant potential for expanding this feature by incorporating additional APIs that can offer even more detailed insights. For example, integrating social media enrichment APIs, such as those from Twitter or Facebook, can provide users with a fuller picture of public profiles based on social media activity. Additionally, integrating geolocation APIs from services like Google Maps or IP-based geolocation services would allow for location-based insights, which can be particularly useful for organizations interested in regional trends or customer demographics. Similarly, APIs that pull market data and industry insights could be introduced to enrich datasets related to organizations. This would provide users with more specific information, such as revenue data, industry classifications, or even competitor analysis, thus making the enriched data far more comprehensive and useful for decision-making.

As part of the planned feature expansion, the system could also introduce enhanced data management features. One potential feature is the implementation of dataset versioning and history tracking. This feature would allow users to view and restore previous versions of their datasets, providing greater flexibility in managing changes to the data over time.

Another important addition would be more robust export options for users, allowing them to export their enriched data in a variety of formats such as JSON or XML. This would increase the system's compatibility with other platforms or tools that users may need to integrate with, making the system more versatile and adaptable to different use cases.

0.14 Scalability Considerations

As the system grows and more users begin to upload datasets and use data enrichment services, scalability becomes a key concern. Ensuring that the system can handle increased traffic, larger datasets, and more concurrent users without compromising on performance is essential for long-term success. Several strategies can be employed to make sure that the system scales effectively.

The first step in ensuring scalability is through horizontal scaling, which involves adding more servers or instances of the application as the demand grows. A load balancing mechanism can be introduced to distribute incoming requests evenly across multiple application servers. This will ensure that no single server is overwhelmed, which could result in slower response times or crashes. Load balancers such as NGINX or cloud-based solutions like AWS Elastic Load Balancer can be used to manage traffic efficiently. As the system continues to grow, introducing container orchestration tools like Kubernetes will become essential. Kubernetes allows the system to scale dynamically by adding new containers as needed, based on real-time traffic and load. This flexibility ensures that the system can automatically adjust to fluctuations in demand, scaling up during periods of heavy usage and scaling down during lighter usage, all while maintaining consistent performance.

Caching strategies also play a crucial role in enhancing system scalability. Introducing caching mechanisms such as Redis or Memcached can significantly improve performance by reducing the load on the database and backend services. Frequently accessed data, such as user sessions or cached API responses, can be stored in-memory for faster retrieval. This reduces the need to repeatedly query the database or make external API calls for the same data, leading to faster response times and a better user experience. Additionally, for external APIs used in the enrichment process, implementing API request throttling ensures that the system doesn't overwhelm third-party services with excessive requests, particularly during periods of high traffic.

0.15 Security Enhancements

As the application grows and handles more sensitive user data, improving security measures becomes increasingly important. Robust security practices will help safeguard user information and maintain compliance with regulations like the General Data Protection Regulation (GDPR). Several enhancements can be made to ensure the security of both the system and the data it manages.

As the application deals with personal data, ensuring GDPR compliance will be essential, particularly if the system is used by European users. GDPR mandates certain rights for users, such as the right to be forgotten, which allows them to request the deletion of their personal data. Implementing functionality that allows users to delete their data or export it in a format that complies with GDPR regulations is necessary to meet these requirements.

Further security measures could include strengthening the user authentication process. Currently, the system uses Google OAuth, which is a secure authentication method. However, this could be enhanced further by introducing multi-factor authentication (MFA), requiring users to provide an additional layer of verification before accessing sensitive parts of the system. This could include sending a code to the user's phone or email, adding an extra layer of security to protect against account takeovers.

Finally, improving the security of API access is another critical consideration. Implementing API rate limiting will help protect the system from abuse, preventing malicious actors from overloading the API with requests. Additionally, managing API tokens by regularly rotating them and limiting their permissions ensures that if a token is compromised, the impact is minimized. Regular audits of the API endpoints and penetration testing can help identify any potential vulnerabilities in the system before they are exploited by attackers.

Conclusion

The SaaS Data Enrichment Application developed during this internship represents a significant step forward in automating and streamlining data management processes for organizations.

By addressing common inefficiencies associated with data handling, the application not only enhances the quality of data available for decision-making but also improves operational efficiency across various organizational functions. The incorporation of user-friendly features such as dataset upload, column matching, and dynamic data enrichment positions the application as a powerful tool in the data analytics landscape. Furthermore, the considerations for future development, including the integration of additional APIs and enhanced scalability measures, ensure that the application remains adaptable to evolving user needs.

Ultimately, this project not only demonstrates the practical application of modern software engineering principles but also lays a strong foundation for further exploration in the field of data enrichment and management.

References

[1] Why Data Analytics Can't Tell You Everything Accessed September 25, 2024. https://www.informationweek.com/data-management/why-data-analytics-can-t-tell-you-everything

[2] Top 6 Manual Data Entry Challenges and Ways to Overcome Them in 2024-2025 Accessed September 25, 2024. https://www.invensis.net/blog/manual-data-entry-challenges

[3] Integration of Big Data in Data Management Accessed September 25, 2024. https://www.lonti.com/blog/integration-of-big-data-in-data-management

[4] Data Enrichment Pricing: Factors Affecting the Cost Accessed September 25, 2024. https://blog.exactbuyer.com/post/data-enrichment-pricing-factors

[5] Next.js (v14.2.7) Accessed September 25, 2024. https://nextjs.org/docs

[6] NextAuth (v4.13.0) Accessed September 25, 2024. https://next-auth.js.org/getting-started/introduction

[7] Ant Design (v5.9.0) Accessed September 25, 2024. https://ant.design/docs/react/introduce

[8] TypeScript (v5.2.2) Accessed September 25, 2024. https://www.typescriptlang.org/docs/

[9] Strapi (v4.25.7) Accessed September 25, 2024. https://docs.strapi.io/

[10] Docker (v24.0.6) Accessed September 25, 2024. https://docs.docker.com/

[11] Docker Compose (v2.22.0) Accessed September 25, 2024. https://docs.docker.com/compose/