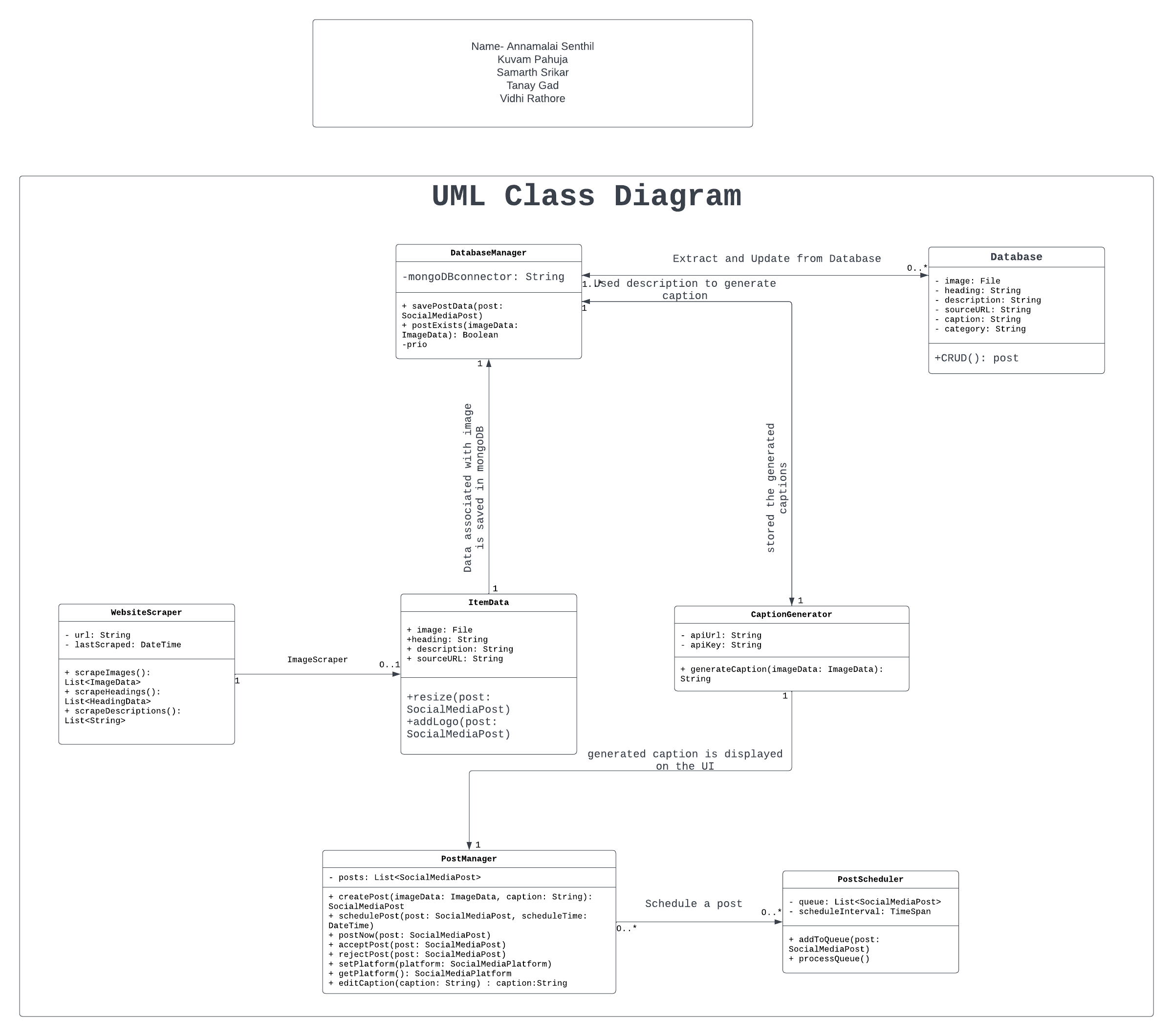
**Product Design**

**Team-43**

**Name- Annamalai Senthil**  
**Kuvam Pahuja**  
**Samarth Srikar**  
**Tanay Gad**  
**Vidhi Rathore**

# Design Model



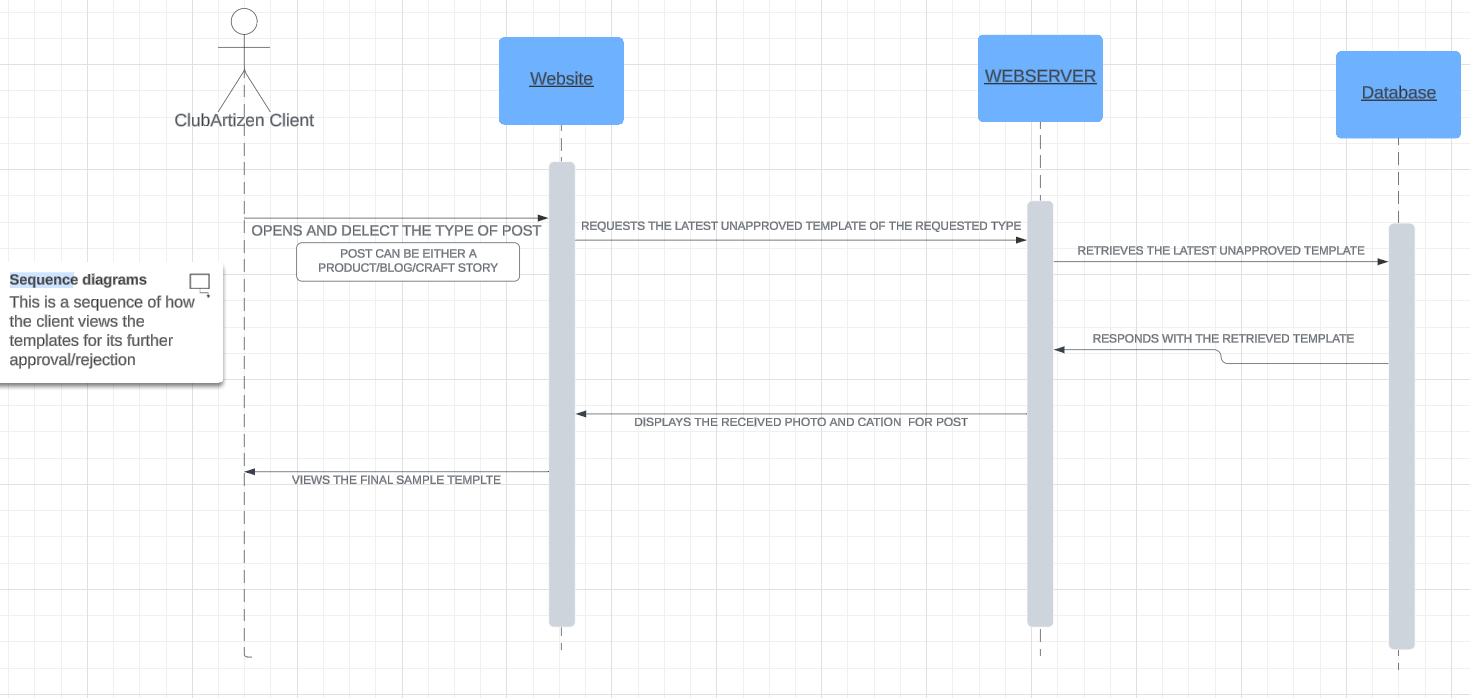
|  |  |
| --- | --- |
| WebsiteScrapper | Class state  Stores the website url to be scraped.  Class behavior  Scrapes all the raw data from the website and sends for further processing |
| ItemData | Class state  Stores raw data extracted  Class behavior  Processes the images and adds a logo to it |
| Database Manager | Class state:  Stores the database connecter string  Class behavior  Acts as a mediator between Database and the other classes |
| Database | Class state:  Stores all the information regarding generated posts.  Class Behavior:  All the CRUD operations are supported to update/ extract info from the database |
| Caption Generator | Class state:  Stores API key, URL to ask queries to a pretrained model  Class Behavior:  Posts queries to generate captions for products to the model and extracts and processes the response. |
| Post Manager | Class state:  Displays Generated posts for approval, scheduling  Class Behavior:  Allows editing of posts, approval for different sites and processing post for scheduling.  Can filter posts based on categories. |
| Post Scheduler | Class state:  Stores the instagram APIs to post  Class Behavior:  Extracts the post from database at the scheduled time and posts it on instagram. |

.

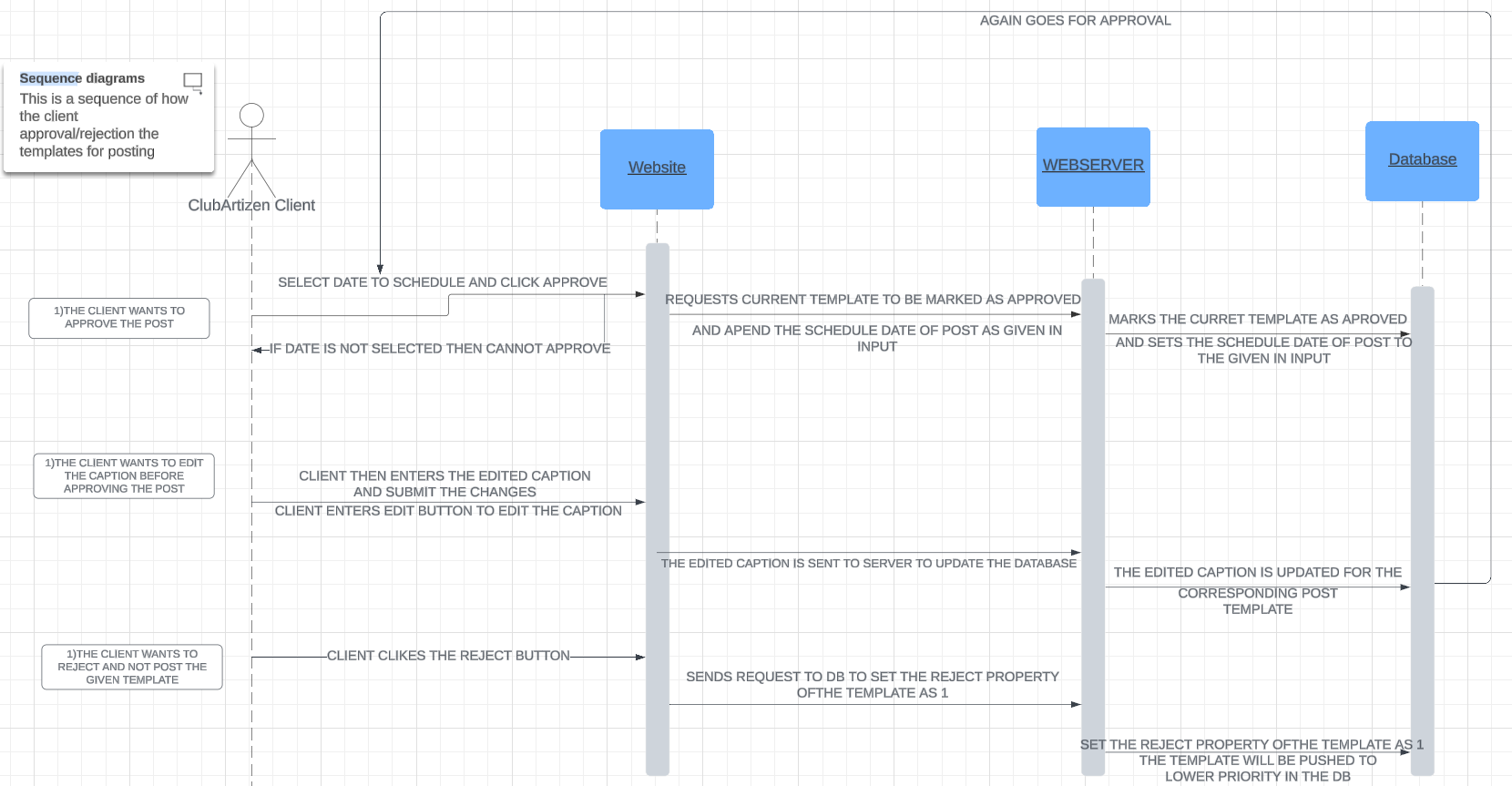
# Sequence Diagram(s)

UML sequence diagram for the agreed significant use cases (about 4 major use cases).

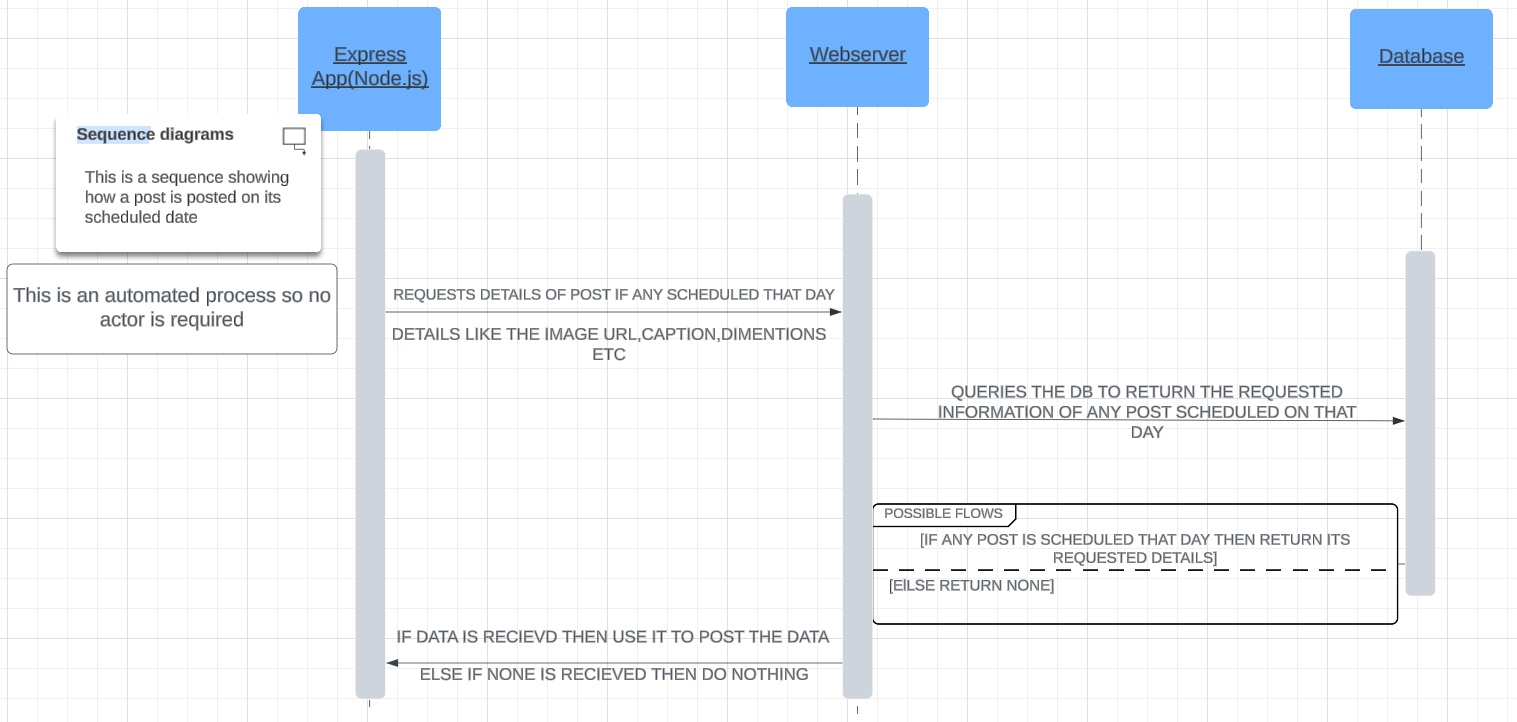
1)Viewing Template on website of the selected Type:

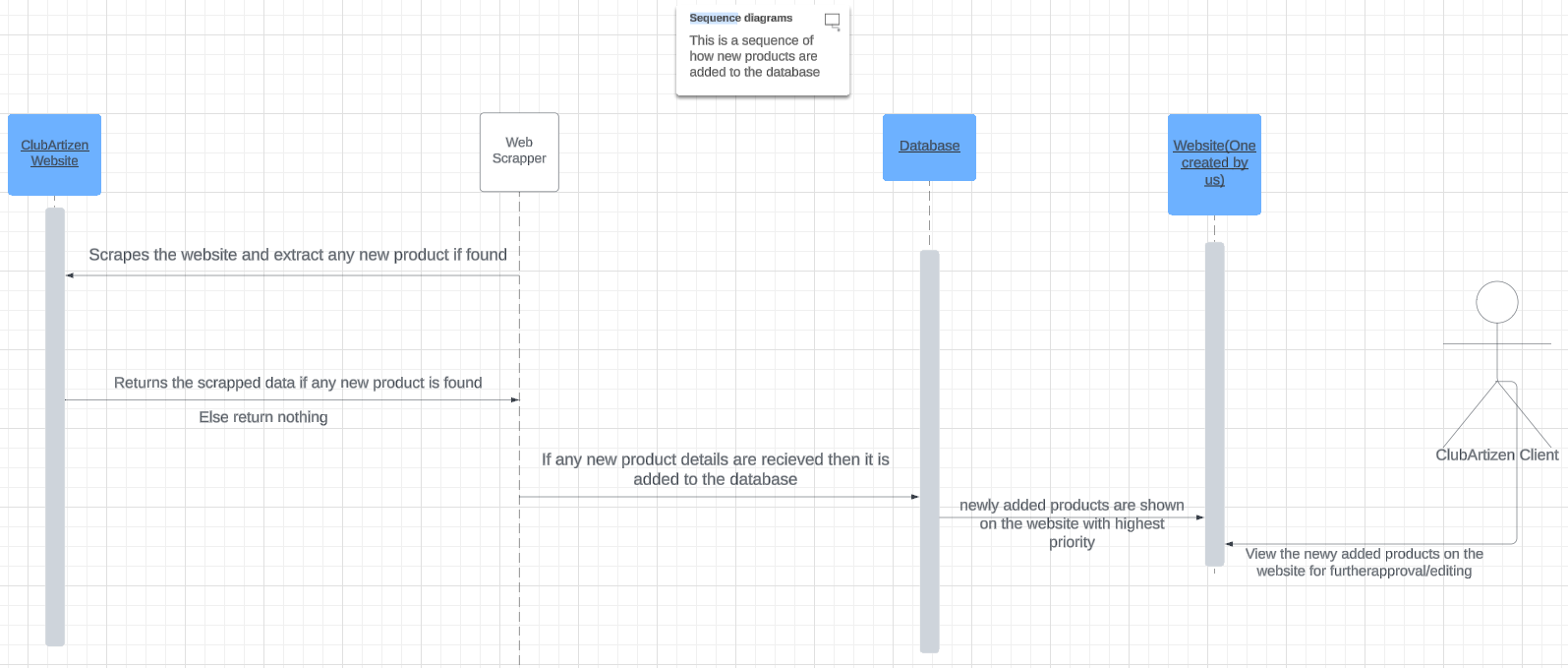


# 2)Approval/Rejection/Editing of the Template



# 3)POSTING PROCESS



4)Adding newly launched products  


**Design Rationale:**

**Evolving Data Storage Format**

* **Initial Approach: One Text File Per Item**
  + *Motivation*: The initial design aimed for simplicity, with each scraped item stored in its own text file. This straightforward approach facilitated quick data storage and retrieval during the development phase.
  + *Advantages*:
    - **Easy Access**: Each item's data was contained within a single file, simplifying the retrieval process.
    - **Straightforward Implementation**: A one-file-per-item structure aligned with the straightforward nature of initial data processing.
* **Evolution to Improved Structure: Separate Files for Headings, URLs, Descriptions**
  + *Challenges Faced*:
    - **Scalability Issues**: As the dataset grew, managing and querying individual text files became inefficient.
    - **Complexity in Updating Data**: Modifying or updating data required handling multiple files, leading to increased complexity.
  + *Solution*: Transitioning to a more structured approach with separate files for headings, image URLs, and descriptions.
  + *Advantages*:
    - **Enhanced Readability**: The separation of data into specific files provides a more organized and readable structure.
    - **Efficient Querying**: Storing related data in dedicated files facilitates more efficient querying and retrieval.
    - **Easier Data Updates**: Modifying or updating specific data components is now more straightforward due to the segregation.

**Transition from CSV to MongoDB for Data Storage**

1. **CSV as Initial Storage Mechanism**
   * *Choice Rationale*: Initially, CSV (Comma-Separated Values) files were chosen as the storage mechanism for their simplicity and ease of implementation during the early stages of development.
   * *Advantages*:
     + **Simplicity**: CSV provided a straightforward and lightweight method for storing structured data.
     + **Quick Implementation**: Rapid implementation allowed for immediate data storage and retrieval.
2. **Challenges with CSV Approach**
   * *Scalability Concerns*: As the dataset grew, the limitations of CSV files in handling complex and large-scale data became evident.
   * *Querying Complexity*: Retrieving specific data points and executing complex queries became cumbersome with the flat structure of CSV files.
3. **Transition to MongoDB**
   * *Motivation*: Recognizing the need for a more robust and scalable solution, the decision was made to transition from CSV to MongoDB, a NoSQL database.
   * *Advantages*:
     + **Structured Document Storage**: MongoDB's document-oriented structure offers a more natural representation of complex data relationships.
     + **Scalability**: MongoDB provides enhanced scalability, allowing for the efficient management of growing datasets.
     + **Querying Flexibility**: MongoDB's querying capabilities offer flexibility and efficiency in retrieving specific data attributes.
     + **JSON-Like Documents**: MongoDB's JSON-like document format aligns well with the structure of modern applications.
4. **Improved Data Management**
   * *Data Organization*: MongoDB's document-based approach allows for the organization of related data within nested structures, facilitating more efficient data management.
   * *Complex Data Handling*: The transition addresses the challenges posed by complex data handling, making it more manageable and scalable.

**Shifting from Selenium to Instagram Private API for Posting**

* 1. **Initial Approach: Selenium Automation**
     + *Motivation*: Selenium was initially employed for its versatility and ability to automate browser interactions. Logging into Instagram, navigating to the posting section via inspect element, and uploading pictures allowed for a flexible and adaptable approach.
  2. **Challenges with Selenium:**
     + *Dependency on Browser Automation*: Selenium's reliance on browser automation introduced challenges related to stability, performance, and potential changes in the Instagram web interface.
     + *Resource Intensiveness*: Browser automation can be resource-intensive, impacting the efficiency and speed of the posting process.
     + *Maintenance Overhead*: Frequent updates or changes to the Instagram web structure required constant adjustment of Selenium scripts, leading to maintenance overhead.
  3. **Transition to Instagram Private API:**
     + *Motivation*: The decision to shift to the Instagram Private API was driven by the need for a more stable, efficient, and reliable method of posting content on Instagram.
  4. **Advantages of Instagram Private API:**
     + *Direct Integration*: Instagram Private API allows direct integration with Instagram's backend, eliminating the need for browser automation.
     + *Efficiency*: The API streamlines the posting process, reducing resource consumption and enhancing overall system efficiency.
     + *Stability and Reliability*: By interacting with Instagram's official API, the system becomes less susceptible to changes in the web interface, ensuring greater stability and reliability.
  5. **Future-Proofing:**
     + *Scalability*: The shift to Instagram Private API positions the system for scalability, accommodating potential changes in Instagram's features and ensuring continued functionality.

**Transition from Local Hosting to Render and Netlify**

* 1. **Initial Hosting Plan: Client's Local System**
     + *Motivation*: Initially, the plan was to host the website on the client's local system. This choice might have been influenced by considerations of immediate accessibility and control.
  2. **Challenges with Local Hosting:**
     + *Limited Accessibility*: Hosting the website locally might provide immediate control but restricts accessibility to a local network, limiting the website's reach.
     + *Maintenance Challenges*: Maintenance and updates on a local system could pose challenges, especially if the client lacks dedicated IT resources for hosting.
  3. **Exploration of Hosting Services: Render and Netlify**
     + *Motivation*: The decision to explore hosting services like **Render and Netlify** aims to address limitations associated with local hosting and unlock additional benefits.
  4. **Advantages of A Render and Netlify Hosting:**
     + *Global Accessibility*: **Render and Netlify** hosting offer global accessibility, ensuring that the website is accessible to a wider audience over the internet.
     + *Scalability*: Hosting on A **Render and Netlify** allows for scalable infrastructure, adapting to varying loads and ensuring optimal performance during traffic spikes.
  5. **Security and Reliability:**
     + *Enhanced Security*: **Render and Netlify** hosting come with robust security measures, safeguarding the website and user data against potential threats.
     + *Reliability*: Leveraging established hosting services enhances the overall reliability of the website, with features like backup and redundancy.
  6. **Ease of Deployment and Management:**
     + *Streamlined Deployment*: **Render and Netlify** offer streamlined deployment processes, simplifying the setup and launch of the website.
     + *Centralized Management*: Centralized management tools provided by hosting services contribute to efficient website administration.