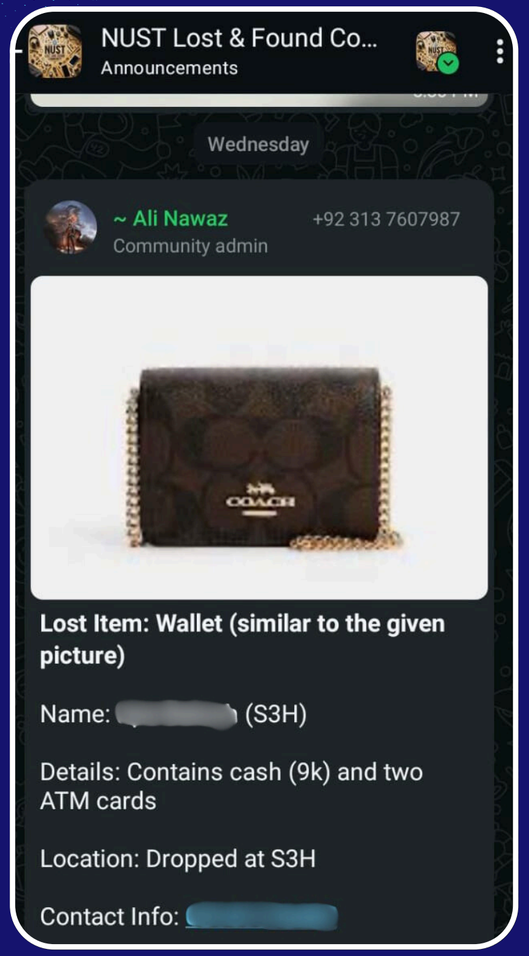
NATIONAL UNIVERSITY OF
SCIENCE & TECHNOLOGY

NUST LOST & FOUND

TEAM MEMBERS:

- Munha Shahzad
- Gulwarina Muska Saleem
- Maha Mohsin

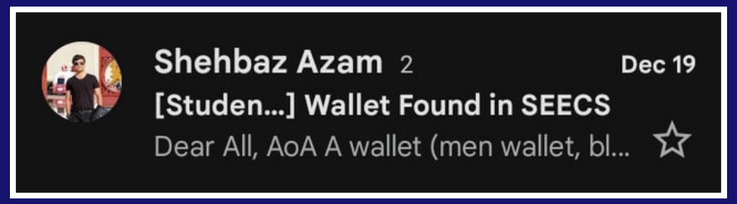


WHY WE CHOSE TO TACKLE THIS PROBLEM

1. **FREQUENCY OF LOSSES**
   People lose important things like ID cards, Wallets and other valuables on a daily basis

2. **LACK OF A PROPER SYSTEM**
   All current systems are manual and time-consuming.

3. **INEFFICIENT COMMUNICATION**
   No proper communication between the finders and original owners.



TECHNICAL IMPLEMENTATION

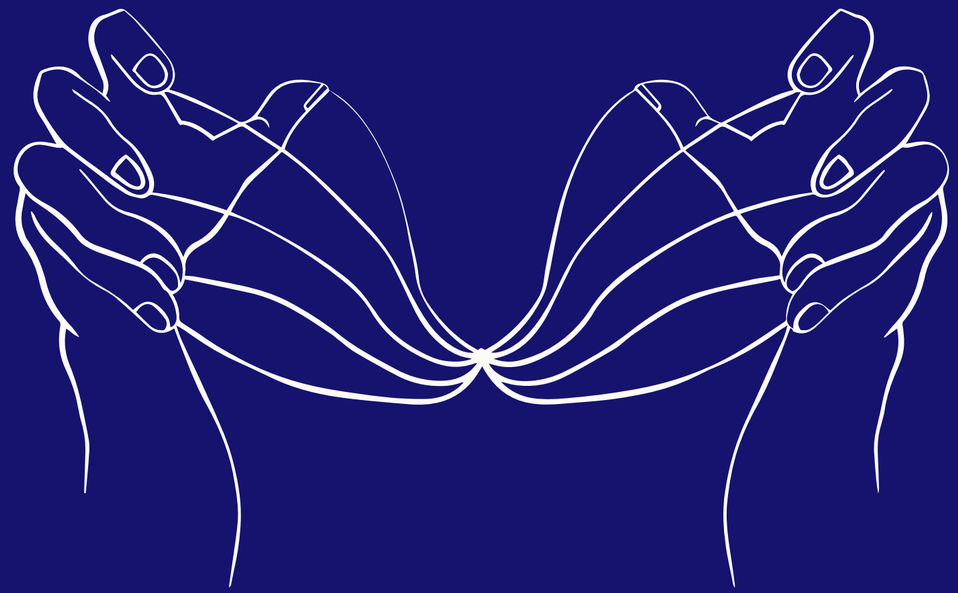
CLIENT-SIDE

- **index.html:** Defines the user interface, including forms, input fields, and display sections for items.
- **styles.css:** Controls the visual presentation, ensuring a user-friendly and visually appealing interface.
- **scripts.js:** Handles user interactions like form submissions, filtering, and chatbot interactions. Implements input validation, data storage, and updates the UI dynamically.

TECHNICAL IMPLEMENTATION

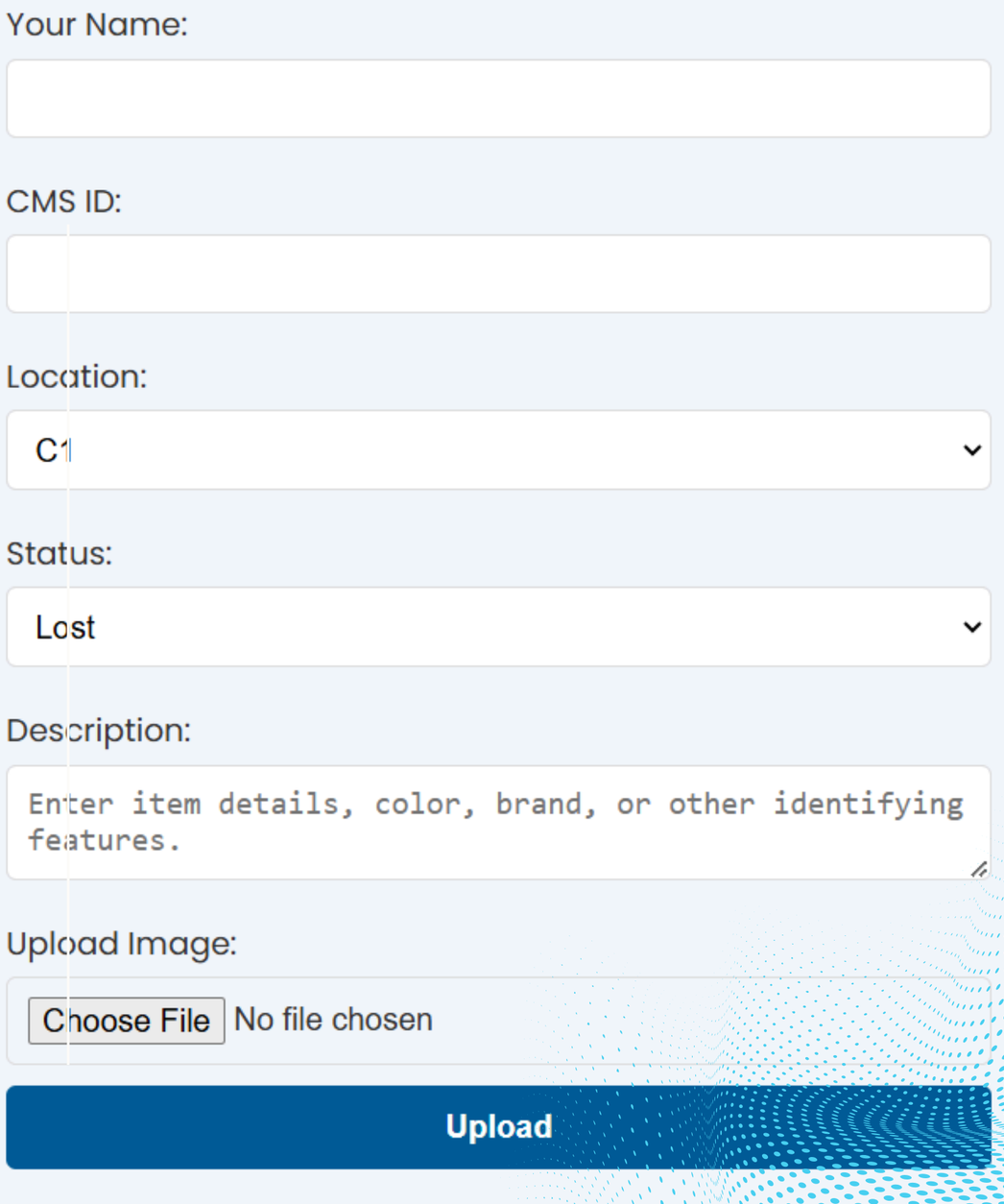
SERVER-SIDE

- **server.js:** Acts as the backend, handling requests from the client-side.
- Uses **Express.js** for routing and handling HTTP requests.
- Employs **multer** middleware to efficiently handle file uploads (images).



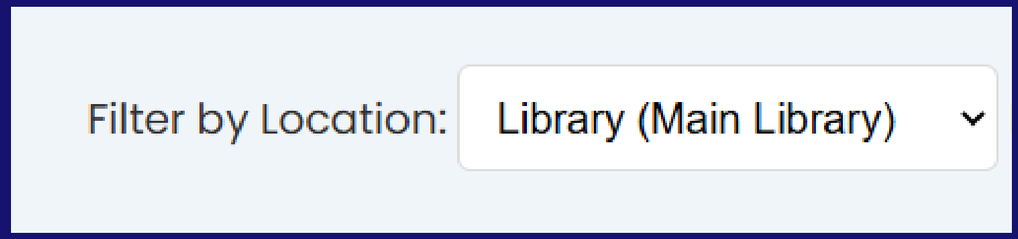
UPLOAD FUNCTIONALITY

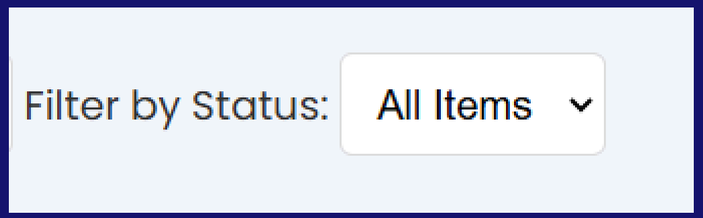
- **Form Handling:** The "Upload" button triggers the form submission. JavaScript collects form data using FormData, including user input and the selected image file. Input validation checks are performed, such as verifying the validity of the CMS ID.
- **Image Handling:** The selected image is converted into a URL using URL.createObjectURL(), allowing it to be displayed on the page before the server-side upload completes.
- **Server-Side Upload:** The server receives the upload request. multer middleware processes the image file and saves it to the uploads folder. The server sends a success response back to the client.
- **Data Storage:** The uploaded item data (including the image URL) is stored in the items array in the client-side JavaScript.



FILTERING AND DISPLAY

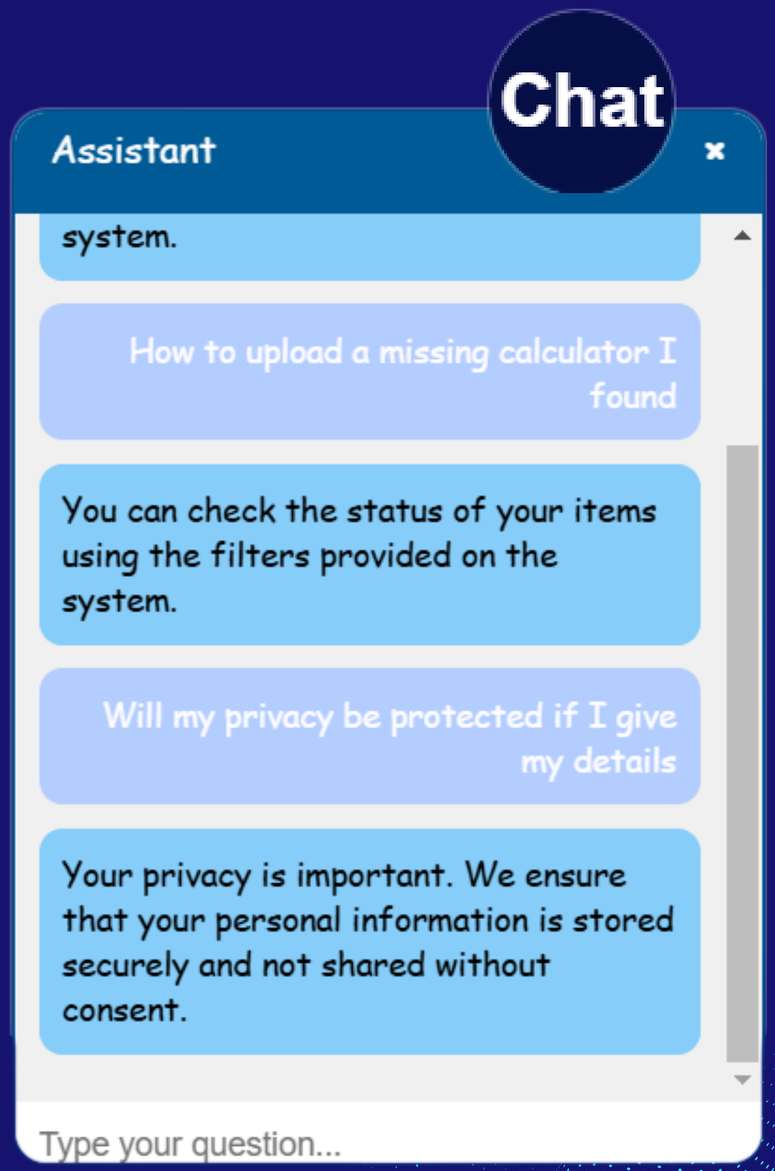
- **Filtering Logic:** Users can filter the list of items by location and status using dropdown menus. JavaScript filters the items array based on the selected criteria.
- **Item Rendering:** The renderItems() function dynamically creates HTML elements to display each filtered item. This includes displaying the item's name, status, location, description, and the uploaded image.





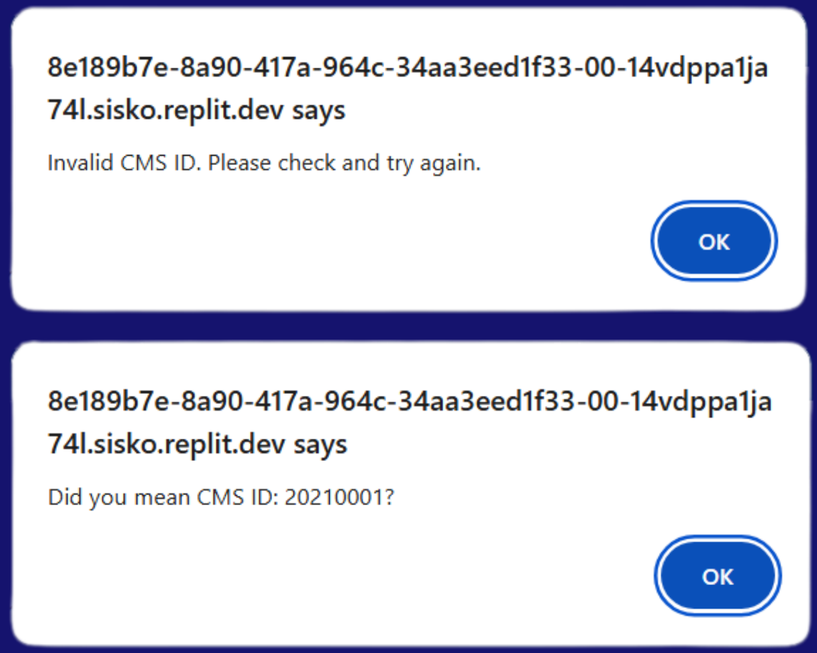
CHATBOT IMPLEMENTATION

- **Core Logic:** The chatbot handles user input through text messages. It uses a combination of keywords and conditional logic to determine appropriate responses. It can answer questions about reporting items, locations, contact information, and basic help.
- **Conversation State:** The conversationState variable tracks the context of the conversation, allowing the chatbot to maintain a more coherent and helpful interaction.



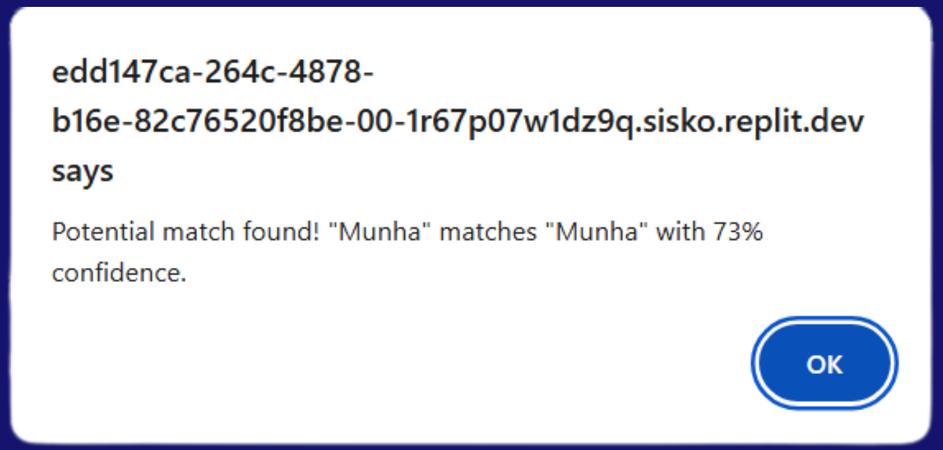
CMS ID VALIDATION

- Implements a robust check for the validity of the entered CMS ID.
- Employs the Levenshtein distance algorithm to calculate the minimum number of edits (insertions, deletions, substitutions) required to transform the entered CMS ID into a known valid ID.
- A lower Levenshtein distance indicates a higher degree of similarity between the two strings.
- If a close match for the CMS ID is found within a predefined threshold, the user is presented with the potential match and prompted to confirm its accuracy. This helps to minimize errors and ensure data integrity.

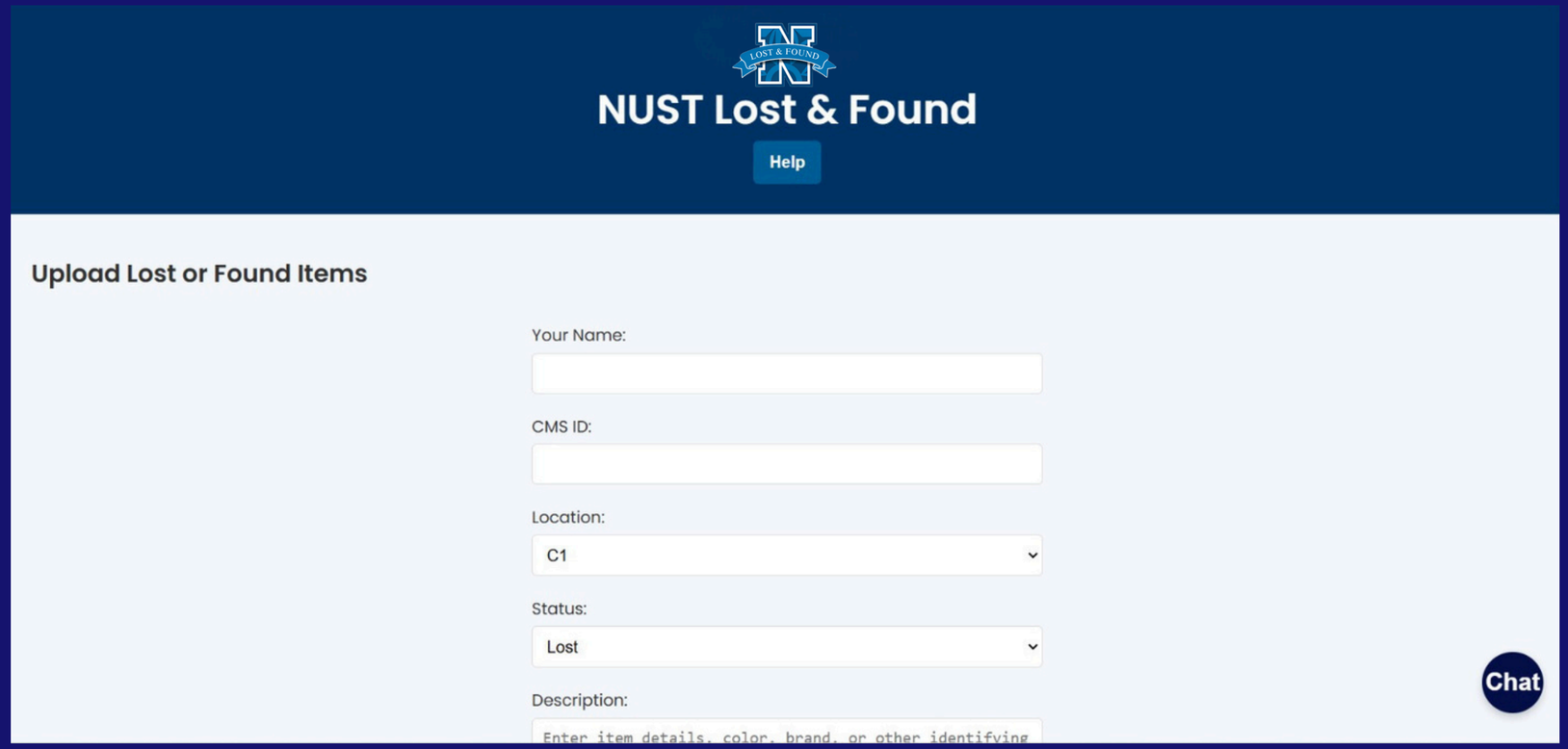


POTENTIAL MATCH IDENTIFICATION

- **Semantic Matching:** Uses Universal Sentence Encoder model from TensorFlow.js. This AI model generates embeddings for text, which are dense vector representations that capture the semantic meaning of the text. When a new item is reported, the model generates an embedding for the item's description. It then compares this embedding with the embeddings of previously reported items.
- **Cosine Similarity:** The code calculates the cosine similarity between the embeddings of the new item's description and existing items. Cosine similarity measures the cosine of the angle between two vectors in a multi-dimensional space. If the cosine similarity between the new item's description and an existing item's description exceeds a predefined threshold, the system alerts the user about a potential match.



LOST AND FOUND WEBSITE IN ACTION

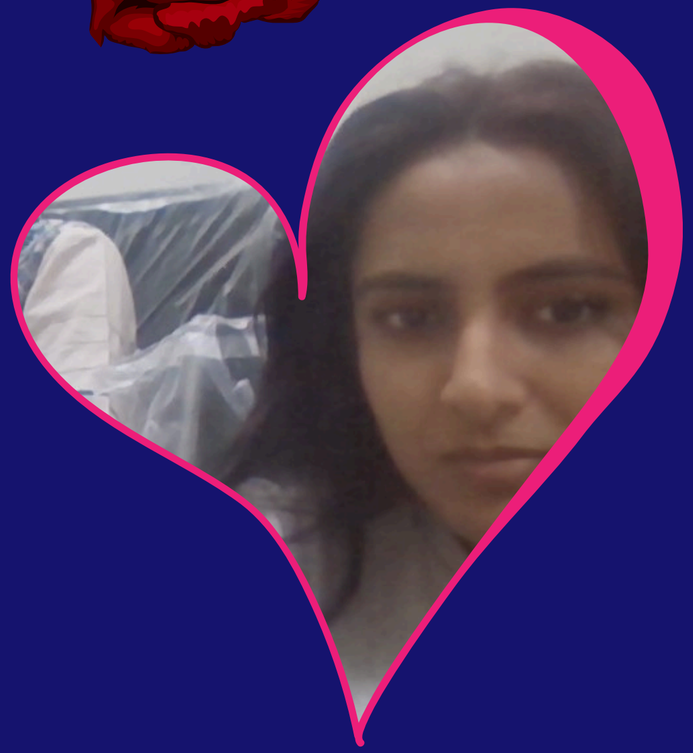


Future Enhancements & Conclusion

Integrating with the NUST database for automatic CMS ID validation.

Real-time notifications for potential matches.

Enhanced chatbot capabilities using GPT models.



THANK YOU FOR YOUR TIME!