## Slide 1: Introduction to GUI Development

"Now, I will be presenting our project on the development of the front-end components for our application. This includes the HTML structure, CSS styling, and integration with Flask for backend functionality.

Our goal was to create a user-friendly interface that is both visually appealing and functional. We'll cover the HTML structure for navigation and content organization, the CSS styling for responsiveness and design, the JavaScript functions for interactivity, and finally, how Flask ties everything together on the backend."

## Open Web Application

"Let's start with the HTML structure and CSS styling.

Firstly, we created a fixed navigation bar at the top of the page. This bar includes links to different sections of the application, such as Home, Converted File, and Historical Conversions, ensuring easy navigation for users.

We then divided the main content into three containers:

- The `home-container` is the default view where users can upload an image or capture a photo.

- The `converted-file-container` displays the result after OCR processing.

- The `historical-conversions-container` shows previously converted texts.

For CSS, we focused on enhancing visual appeal and ensuring the application is responsive across various devices. We implemented a responsive design where elements are resized according to the screen size. Buttons were styled consistently with hover effects to improve interactivity. Additionally, we mirrored the video feed horizontally when using the camera to facilitate easier preprocessing."

## Show JavaScript Functions as background

"Next, let's talk about the JavaScript functionality that adds interactivity to our web page.

For camera access, we have three key functions:

- `openCamera()`: This function accesses the user's camera, creates a video element, and displays the video feed.

- `capturePhoto()`: It captures an image from the video feed, displays the captured image, and removes the video element.

- `stopCamera()`: This stops the camera stream and removes the video element from the DOM.

For file uploads, we use the `convertImage()` function. This handles image uploads, converts the image format, sends the image to the server via a POST request, displays the uploaded image, and shows the converted string results after receiving the server response.

Lastly, for navigation, we use functions like `showHome()`, `showConvertedFile()`, and `showHistoricalConversions()` to switch between different content containers based on user actions."

## Slide 2: Integration with Flask

"Finally, let's discuss the integration with Flask and the results of our project.

Flask is a WSGI *application*. A WSGI *server* is used to run the application, converting incoming HTTP requests to the standard WSGI environ, and converting outgoing WSGI responses to HTTP responses.

Flask plays a crucial role in our project by rendering the main HTML page, handling image processing requests, and managing different views based on user interactions. When an image is uploaded or captured, Flask processes it uses our OCR model and returns the extracted text.

The results of our application are very promising. Users can easily upload images or capture photos directly from their devices. The responsive design ensures the application is accessible on various devices, providing a consistent user experience.

## Slide 3: Results and Challenges

However, we faced some challenges:

- Implementing camera functionality and ensuring the video feed was mirrored correctly required careful handling of video streams and CSS transformations.

- Ensuring the layout was fully responsive involved extensive testing and adjustments

- Integrating the front-end with the Flask backend required a clear understanding of both client-side and server-side programming.

Despite these challenges, we successfully developed a user-friendly and functional application that meets our project goals."