



بنام معمارهستی

پروژه تحلیل و پایش محیط با استفاده از دوربین ESP32 به کمک
مدل زبانی بزرگ Gemini

نام استاد:

دکتر بهلولی

اعضای گروه:

حسن رجائی ۴۰۰۳۶۱۳۰۳۰

مبینا مومنزاده ۴۰۰۳۶۲۳۰۳۶

رادمهر آقاخانی ۴۰۰۳۶۶۳۰۰۲

فهرست مطالب

۱- انتخاب مدل زبانی	۳
۲- تحلیل کد	۴
۱-۲- پیکربندی دوربین	۴
۲-۲- حلقه loop و خواندن Http request	۴
۱-۲-۲- تابع handleRoot	۴
۲-۲-۲- تابع handleCaptureImages	۵
۳-۲-۲- تابع handleGetAnalysis	۶
۴-۲-۲- تابع sendToAPI	۸
۵-۲-۲- تابع flushCameraBuffer	۸
۶-۲-۲- تابع handleResetData	۹
۳- نمونه سناریوهای اجرای سامانه ESP32-CAM AI Monitor	۱۰
۱-۳- سناریو ۱: تشخیص حضور یک شخص در محیط خالی	۱۰
۲-۳- سناریو ۲: تغییر نور و نحوه قرار گیری فرد	۱۱
۳-۳- سناریو ۳: شناسایی فعالیت و شیء مشکوک در دست شخص	۱۳
۴-۳- سناریو ۴: شناسایی تغییر شدید نور در محیط	۱۴
۴- مراجع	۱۶

۱- انتخاب مدل زبانی

در این پروژه از مدل هوش مصنوعی Gemini 1.5 Flash استفاده شده که از طریق سرویس دهنده‌ی Avalapis.ir در دسترس قرار گرفته است. این مدل قابلیت تعامل چندرسانه‌ای متن و تصویر را دارد و از طریق ارسال درخواست POST به آدرس <https://api.avalapis.ir/v1/chat/completions> به کار گرفته می‌شود. فرمت درخواست‌ها به صورت JSON شامل یک پرامپت متنی و دو تصویر Base64 است.

از بین مدل‌های موجود، Gemini 1.5 Flash بهترین گزینه‌ای بود که با سطح دسترسی پایه می‌توانستیم در این پروژه استفاده کنیم. این انتخاب به ما این امکان را داد که با کمترین هزینه ممکن و با حفظ کیفیت تحلیل، از توانایی‌های بصری (vision) بکار بگیریم.

با توجه به این که در پروژه ما فقط دو تصویر و یک پرامپت متنی ارسال می‌شود، مصرف توکن پایین بوده و هزینه اجرای هر درخواست بسیار ناچیز است. همچنین استفاده از پرامپت‌های خلاصه و تکرار درخواست‌های مشابه، منجر به کش شدن پاسخ‌ها و کاهش بیشتر هزینه خواهد شد.

جدول جزییات مدل بصورت زیر می‌باشد:

هزینه (واحد اعتبار به ازای هر ۱ میلیون توکن)	مقدار/محدودیت	نوع مصرف
—	۵۰۰ درخواست در دقیقه (RPM)	حداکثر نرخ درخواست
—	۴ میلیون توکن در دقیقه (TPM)	حداکثر نرخ توکن
۰/۰۷۵	تا ۱۲۸ هزار توکن	ورودی (prompt)
۰/۱۵	بیشتر از ۱۲۸ هزار توکن	
۰/۳	تا ۱۲۸ هزار توکن	خروجی (پاسخ مدل)
۰/۶	بیشتر از ۱۲۸ هزار توکن	
۰/۰۳۷۵	هر مقدار توکن	درخواست کش شده

توجه: منظور از «درخواست کش شده» پاسخ‌هایی است که قبلاً پردازش شده‌اند و از کش بازگردانی می‌شوند، نه پردازش جدید. هزینه‌ی آن نصف هزینه‌ی prompt سطح پایین است.

۲- تحلیل کد

در این قسمت بخش هایی از کد ارائه شده است که در پیاده سازی پروژه نقش بسزایی داشته‌اند:

۲-۱- پیکربندی دوربین

این قسمت از کد که از خط ۶۵ شروع شده و تا خط ۹۵ ادامه دارد. مهم‌ترین پارمتر آن را سایز فریم است که بصورت پیش‌فرض بر روی QVGA که متناسب با رزولوشن ۲۴۰ * ۳۲۰ است تنظیم شده است که ما این را برای صرفه جویی در توکن مصرفی و کاهش دادن حجم پیلود درخواست به API، به QQVGA متناسب با یک چهارم رزولوشن VGA اصلی (۱۲۰*۱۶۰) کاهش دادیم. نمونه کد در پایین آمده است:

```
config.frame_size = FRAMESIZE_QQVGA;
```

۲-۲- حلقه loop و خواندن Http request

این بخش در واقع شامل توابع اصلی: ساختن صفحه اصلی وب سرور، عکس گرفتن، مهندسی پرامپت و فرستادن درخواست به API و نمایش آن است:

۲-۲-۱- تابع handleRoot:

این تابع از لحظه‌ای که مرورگر کاربر آدرس / را درخواست می‌کند، آغاز به ساخت پاسخ HTTP می‌کند. ابتدا در سه `println` متوالی، کد وضعیت OK ۲۰۰، نوع محتوا `text/html` و دستور بستن اتصال پس از ارسال پاسخ فرستاده می‌شود؛ سپس یک خط خالی می‌آید که پایان سرآیند HTTP را علامت می‌زند. بعد از آن، با `client.print` یک رشته خام (raw string literal) ارسال می‌شود که تمام اسناد HTML، CSS و JavaScript صفحه را شامل می‌شود.

منطق رابط کاربر تماماً در JavaScript تعبیه شده است:

۱. تابع `updateStatus` پیام وضعیت را با رنگ سبز یا قرمز نمایش می‌دهد.
۲. `showPromptSelection` پس از دریافت دو تصویر، فرم «گرفتن عکس» را پنهان و بخش انتخاب نوع تحلیل را آشکار و تصاویر Base64 را در تگ‌های `` قرار می‌دهد.
۳. `resetPage` صفحه را به حالت اولیه بازمی‌گرداند.

مراحل انجام درخواست‌ها بصورت زیر می‌باشد:

- (۱) هنگام ارسال فرم `capture` ، کد با `fetch` به مسیر `/capture_images?delay=` روی خود `ESP32` درخواست می‌فرستد، انتظار دارد پاسخ `JSON` حاوی `success` و دو رشته `Base64` تصویر را بگیرد.
- (۲) با موفقیت، وضعیت به «تصاویر گرفته شد» تغییر کرده و `showPromptSelection` فراخوانی می‌شود.
- (۳) برای تحلیل، کاربر می‌تواند یک یا چند تیک بزند؛ حلقه‌ای روی هر تیک یک `POST` به `/get_analysis? type=` می‌زند.
- (۴) پاسخ این درخواست، متنی خام است که در `updateResponse` به `JSON` تبدیل می‌شود.
- (۵) مسیر `choices[0].message.content` استخراج و در یک جعبه رنگی زیر صفحه نشان داده می‌شود. اگر خطا باشد، همان متن یا پیام خطا در جعبه قرمز می‌آید.
- (۶) سرانجام، فرم `reset` با `POST /reset_data` بفر تصاویر را پاک می‌کند و رابط را ریست می‌کند.
- (۷) در `DOMContentLoaded` صفحه بلافاصله `resetPage` را اجرا می‌کند تا همه چیز آماده ثبت عکس تازه باشد.

ر نتیجه `handleRoot` فقط یک صفحه ایستا با منطق کامل فرانت‌اند می‌سازد؛ که کار واقعی گرفتن عکس، ذخیره `Base64`، تماس با مدل زبانی برای تحلیل و پاک‌سازی همگی در مسیرهای `/capture_images` ، `/get_analysis` و `/reset_data` که بر روی مازول `ESP32-CAM` پیاده‌سازی شده است.

۲-۲-۲- تابع `handleCaptureImages`

زمانی که مرورگر از مسیر `/capture_images?delay=...` استفاده می‌کند. این تابع وظیفه دارد دو تصویر را با تاخیر زمانی دلخواه بگیرد، آن‌ها را به `Base64` تبدیل کند، و همراه با پیامی در قالب `JSON` برای مرورگر ارسال کند.

خلاصه از عملکرد این تابع:

- گرفتن اولین عکس و تبدیل آن به رشته `Base64`
 - اگر گرفتن عکس موفق نبود (طول رشته صفر بود)، پاسخ `JSON` برای مرورگر ارسال می‌شود که موفقیت `false` بوده است و پیام خطا دارد، و تابع پایان می‌یابد.
- تأخیر برحسب ثانیه‌ای که از کلاینت (مرورگر) آمده
- گرفتن عکس دوم و تبدیل به `Base64`
- ارسال `JSON` حاوی هر دو عکس و پیام موفقیت یا خطا به مرورگر

تابع `handleCaptureImages` دقیقاً مثل یک API تحت وب رفتار می‌کند. که به ما امکان می‌دهد از راه دور دو تصویر بگیریم (با فاصله زمانی دلخواه) و خروجی‌ها را در فرمت JSON دریافت کنیم. این داده‌ها مستقیماً در کد جاوااسکریپت سمت کلاینت نمایش داده می‌شوند (`showPromptSelection(...)` در `handleRoot`).

۲-۲-۳ تابع `handleGetAnalysis`

این تابع مسیر `/get_analysis` را مدیریت می‌کند و برای هر درخواست، بسته به نوع تحلیل خواسته شده «alert»، «formal» یا «friendly» متن مناسب را می‌سازد، آن را همراه با دو تصویر Base64 به‌سوی API مدل زبانی استفاده شده می‌فرستد و پاسخ خام را برای کلاینت باز می‌گرداند:

۱. در آغاز با پیام سریال اعلام می‌شود که یک درخواست تحلیل با نوع مشخص دریافت شده است.
۲. اگر هنوز هیچ تصویری گرفته نشده باشد (طول `img1_base64` یا `img2_base64` صفر است)، سرور با کد ۴۰۰ و متن خطا پاسخ می‌دهد و تابع پایان می‌یابد.
۳. بر اساس `promptType`، یکی از سه متن زیر ساخته می‌شود (**Prompt Engineering**):

▪ «alert» برای هشدار فوری و بسیار کوتاه:

```
prompt_to_send = "URGENT ALERT: Immediate action required. Analyze and identify critical changes between the two images that demand immediate attention. Focus on potential threats, security breaches, or critical system failures. Respond with a very concise, actionable alert message.";
```

▪ «formal» برای گزارش رسمی، مفصل و ساختاریافته:

```
prompt_to_send = "Generate a comprehensive, formal analytical report comparing the two provided images. Detail all observed changes, including subtle differences in lighting, object positions, and the operational status of any visible devices. Analyze potential causes and implications of these changes. Present findings in a structured, objective manner.";
```

- «friendly» برای توضیح خودمانی و مختصر ویژه کاربر خانگی:

```
prompt_to_send = "Hey there! I've taken two pictures of your room a little while apart. Could you tell me in a friendly, easy-to-understand way if anything looks different? Like, did someone move something, or is a light on that wasn't before? Just a quick summary for a home user, please!";
```

۴. متن استاندارد base_analysis_prompt به انتهای پیام انتخابی افزوده می‌شود تا مطمئن شویم مدل زبانی همیشه چهار مورد کلیدی (تغییر اشیاء، وضعیت دستگاه‌ها، حضور انسان/حیوان و تغییرات مشکوک) را بررسی می‌کند:

"Please analyze and describe:

- Any noticeable changes in objects or lighting.**
- If any devices appear to have turned ON or OFF.**
- Any signs of human or animal presence or movement.**
- Any suspicious or unexpected changes.**

Please respond with a clear and concise summary."

این متن تضمین می‌کند که صرف‌نظر از نحوه نگارش اولیه پیام «alert»، «formal» یا «friendly» چهار پرسش اصلی همیشه مطرح شوند و پاسخ نهایی از مدل لزوماً آن‌ها را پوشش دهد.

۵. تابع sendToAPI با سه آرگومان فراخوانی می‌شود: متن کامل پرامپت، تصویر اول و تصویر دوم. این تابع باید پاسخ مدل زبانی را معمولاً به‌صورت JSON برگرداند. پاسخ خام در سریال لاگ می‌شود تا در زمان خطایابی بتوان محتوای دقیق API را دید.

۶. سرانجام سرور با وضعیت OK ۲۰۰ و تیپ text/plain پاسخ می‌دهد و همان متن خام API را به کلاینت می‌فرستد؛ جاوااسکریپت سمت مرورگر مسئول تجزیه JSON و نمایش آن است.

۲-۲-۴- تابع sendToAPI

این تابع وظیفه دارد درخواست چندرسانه‌ای متن + دو تصویر Base64 را به سرور مدل زبانی — که در متغیر `api_url` مشخص شده — ارسال کند و پاسخ خام را برگرداند. ابتدا یک شی `HttpClient` ساخته می‌شود و به آدرس سرور متصل می‌گردد. سپس زمان پایان کار روی سی ثانیه تنظیم شده و دو سرآیند `Content-Type: application/json` و `Authorization: Bearer ...` افزوده می‌شود.

۱. بدنه JSON دستی رشته‌سازی می‌شود.

- کلید `model` نام مدلی است که در متغیر `model` تعیین می‌شود.
- آرایه `messages` دو پیام دارد:

- نقش `system` هویت کلی دستیار را تعریف می‌کند. «You are an AI monitoring assistant...»
- پیام `user` شامل یک آرایه `content` است که نخستین عضو آن متن پرامپت انتخابی (`prompt_text`) و دو عضو بعدی `URL` های داده `URI` حاوی تصویر اول و دوم است (زیرا `Gemini` باید متوجه شود که ساختار مربوطه `base64` می‌باشد).
- این ساختار همان چیزی است که `Gemini` برای یک درخواست مولتی‌مودال انتظار دارد.

۲. تابع سپس `payload` را با `http.POST` می‌فرستد و کد پاسخ `HTTP` را چاپ می‌کند.

۳. اگر کد بالاتر از صفر باشد، بدنه پاسخ به رشته `response` ریخته می‌شود؛ در غیر این صورت، `errorToString` پیام خطا و کد را کنار هم می‌سازد. در پایان اتصال `http` بسته و همان `response` به فراخواننده برگردانده می‌شود.

۲-۲-۵- تابع flushCameraBuffer

این تابع برای «پاک‌سازی بافر دوربین» طراحی شده است. `esp_camera_fb_get()` یک فریم (تصویر) از بافر دوربین می‌گیرد و اشاره‌گر به آن را در `fb` ذخیره می‌کند:

- اگر تصویر معتبر باشد (`fb != NULL`)، با `esp_camera_fb_return(fb)` آن فریم آزاد می‌شود.

این فرآیند ۳ بار تکرار می‌شود تا مطمئن شویم بافرهای باقی‌مانده (در صورت وجود چند فریم قدیمی) کاملاً تخلیه شده‌اند. یک `delay(50)` برای پایداری میان هر تکرار قرار داده شده است.

۲-۲-۵-۱- تاخیر فریم (frame lagging)

دوربین ممکن است چند فریم قبلی را نگه دارد، به‌ویژه اگر بین دو عکس وقفه زیاد باشد. این تابع کمک می‌کند دفعه بعد که عکس می‌گیرید، حتماً فریم به‌روز و واقعی گرفته شود و نه یک فریم با تاخیر.

۲-۲-۶- تابع `handleResetData`

این تابع هنگام دریافت درخواست از مسیر `/reset_data` اجرا می‌شود:

۱. پیامی در مانیتور سریال چاپ می‌کند.
۲. دو متغیر `img1_base64` و `img2_base64` که قبلاً تصاویر Base64 را نگه می‌داشتند (پاک می‌شوند).
۳. تابع `flushCameraBuffer()` برای پاک‌سازی بافرهای دوربین فراخوانی می‌شود.
۴. در نهایت، پاسخ JSON به کلاینت (مرورگر) بازگردانده می‌شود:

```
{
  "success": true,
  "message": "Data reset successfully"
}
```

بعد از تحلیل تصاویر، اگر کاربر روی دکمه «Reset» کلیک کند، این تابع اجرا می‌شود تا تصاویر قبلی از حافظه پاک شوند و سیستم برای گرفتن عکس جدید آماده شود. این کار از بروز تداخل، مصرف زیاد حافظه، یا تحلیل اشتباه روی تصاویر قدیمی جلوگیری می‌کند.

۳- نمونه سناریوهای اجرای سامانه ESP32-CAM AI Monitor

در این بخش از گزارش، ۴ نمونه از اجرای واقعی سامانه اینترنت اشیا بر مبنای ماژول ESP32-CAM آورده شده است. هر سناریو نشان می‌دهد که یک ویژگی مختلف از سامانه تحلیل تصویری بر پایه Gemini-1.5-flash به شکل عملی به کار گرفته شده و خروجی متنی شفاف و قابل فهمی از تغییرات ارسال می‌کند.

۳-۱- سناریو ۱: تشخیص حضور یک شخص در محیط خالی

شرح: در این سناریو، تصویر اول محیطی خالی را نشان می‌دهد، در حالی که در تصویر دوم یک فرد در صحنه حضور پیدا می‌کند.

پرامپت منتخب : Formal Analytical Report

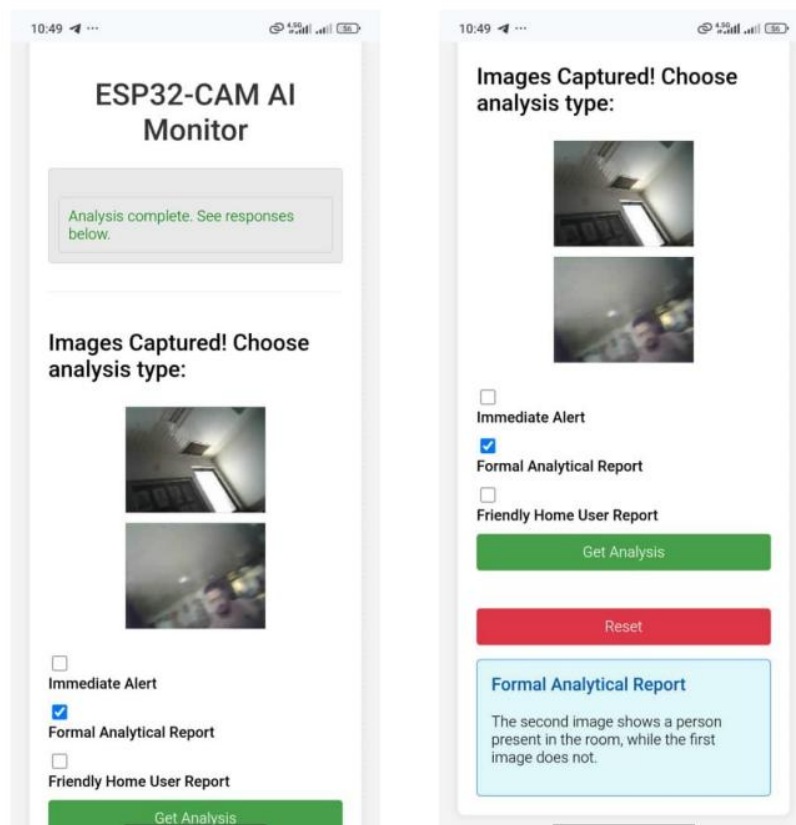
خروجی مدل زبانی:

“The second image shows a person present in the room, while the first image does not.”

تحلیل:

سیستم با دقت توانست تغییر وضعیت و حضور شخص را شناسایی کند و خروجی رسمی و قابل اسنادی ارائه دهد.

اسکرین شات:



توجه: برای بازنمایی بهتر زوم نمایید

۳-۲- سناریو ۲: تغییر نور و نحوه قرار گیری فرد

شرح: در تصویر دوم شخصی با پوشش مشکی و نوری متفاوت در صحنه حضور دارد. تغییر شدید زاویه دوربین و شدت روشنایی مشهود است.

پرامپت منتخب: Formal Report

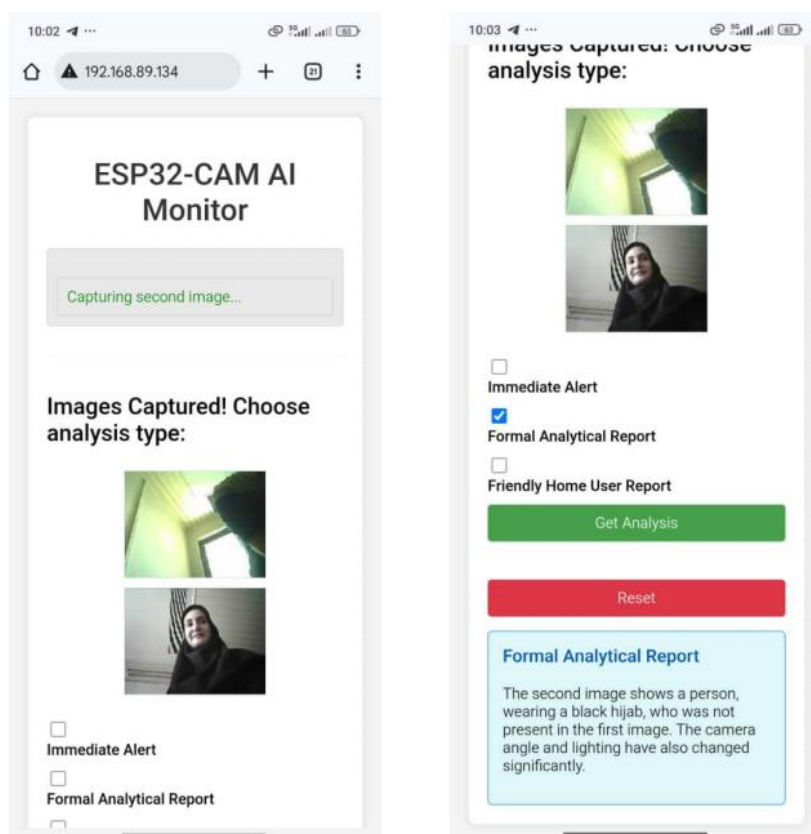
خروجی رسمی:

“The second image shows a person, wearing a black hijab, who was not present in the first image. The camera angle and lighting have also changed significantly.”

تحلیل :

مدل تغییرات هم در شرایط محیطی و هم در وضعیت افراد را تحلیل کرده است.

اسکرین شات:



توجه: برای بازنمایی بهتر زوم نمایید

۳-۳- سناریو ۳: شناسایی فعالیت و شیء مشکوک در دست شخص

شرح: در تصویر اول فردی در صحنه هست، اما فعالیتی ندارد. در تصویر دوم، همان شخص در حال نگاه داشتن یک شیء مستطیل تیره رنگ است.

پرامپت منتخب: Immediate Alert + Friendly Home Report

خروجی هشدار:

“ALERT: Unauthorized human presence detected; subject manipulating an unknown object. Immediate investigation required.”

خروجی خانگی:

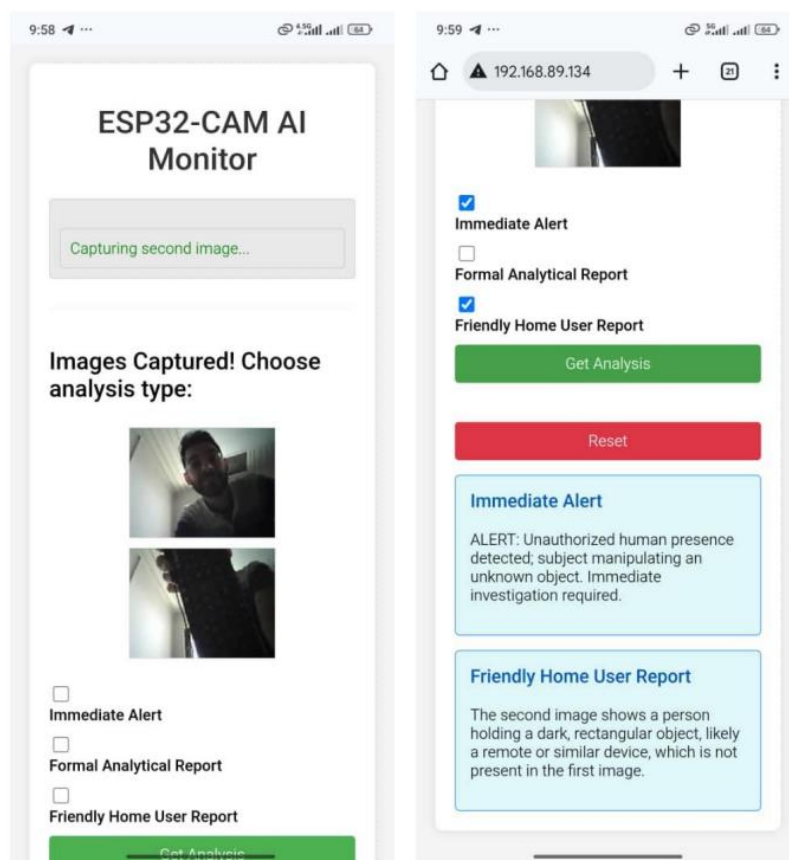
“The second image shows a person holding a dark, rectangular object, likely a remote or similar device, which is not present in the first image.”

تحلیل:

در این سناریو، سیستم توانست نه تنها حضور یک فرد، بلکه تغییر در وضعیت آن فرد (داشتن یا نداشتن یک شیء خاص) را نیز تشخیص دهد. این ویژگی برای کاربردهایی مثل کنترل از راه دور، تشخیص فعالیت غیرمجاز یا نظارت خانگی بسیار ارزشمند است. همچنین خروجی دو پرامپت مختلف، نشان دهنده‌ی توانایی سیستم در تطبیق لحن و عمق تحلیل با مخاطب هدف است:

- هشدار فوری برای امنیت
- گزارش دوستانه برای کاربران خانگی

اسکیرین شات:



توجه: برای بازنمایی بهتر زوم نمایید

۳-۴- سناریو ۴: شناسایی تغییر شدید نور در محیط

شرح: فردی در صحنه نیست، اما روشنایی محیط در تصویر دوم بطور واضح کاهش یافته است.
پرامپت منتخب: Immediate Alert

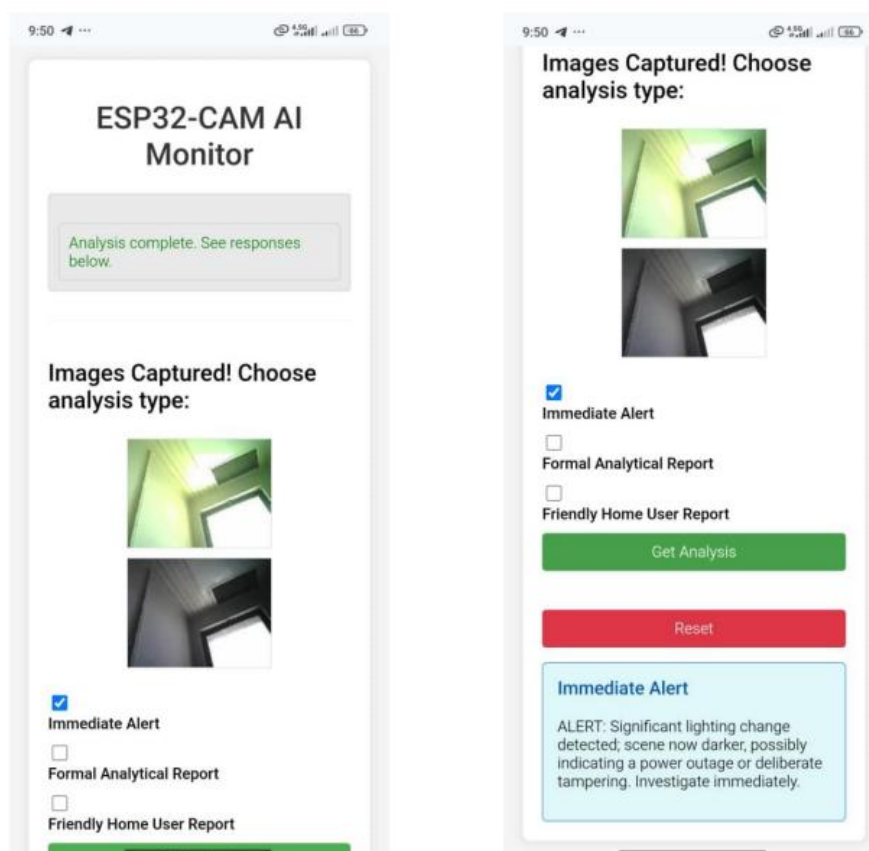
خروجی:

“ALERT: Significant lighting change detected; scene now darker, possibly indicating a power outage or deliberate tampering. Investigate immediately.”

تحلیل:



سیستم قادر به شناسایی تغییرات ساختاری و مهم محیطی نظیر قطع برق، افت نور یا نادرست شدن مسیر دوربین بوده و برای آنها هشدار تولید می‌کند.

اسکرین‌شات:



توجه: برای بازنمایی بهتر زوم نمایید

٤- مراجع

 <https://chatgpt.com/>
 <https://gemini.google.com>