عنوان پروژه : اپلیکیشن واقعیت افزوده تشخیص ساختمان‌های دانشگاه

نام اپلیکیشن : Architect

نام توسعه دهنده : بیتا باتمانی

رشته و دانشگاه : کارشناسی مهندسی کامپیوتر - دانشگاه آزاد اسلامی واحد سنندج

تاریخ تحویل : ۱۴۰۳/۱۲/۲۱

چکیده

این پروژه با عنوان "Architect" به عنوان یک سامانه مبتنی بر فناوری واقعیت افزوده (AR) و یادگیری ماشین با استفاده از مدل (YOLOv8) طراحی و پیاده‌سازی شده است. هدف اصلی این پروژه، ارائه راهکاری نوین جهت تشخیص سریع و دقیق ساختمان‌های دانشگاه و ارائه اطلاعات تکمیلی به کاربر به صورت پاپ‌آپ‌های تعاملی در محیط AR می‌باشد. در این مستند، ابتدا مبانی نظری و اهداف پروژه مورد بررسی قرار گرفته، سپس به جزئیات معماری سیستم، تکنولوژی‌های به‌کاررفته از جمله(Unity ، Flask، YOLOv8، دیتابیس JSON و ابزارهای نظارتی)پرداخته شده و روند پیاده‌سازی و توسعه سیستم به همراه چالش‌های فنی و راهکارهای ارائه‌شده شرح داده می‌شود. در پایان نیز نتایج تست و ارزیابی پروژه به همراه چشم‌انداز توسعه‌های آتی بیان می‌شود.

فهرست مطالب

مقدمه

پیشرفت‌های سریع در حوزه فناوری‌های نوین، بخصوص در زمینه واقعیت افزوده – Augmented Reality) AR و یادگیری ماشین، امکان ارائه راهکارهای نوین و کاربردی در زندگی روزمره را فراهم ساخته است. امروزه، دانشگاه‌ها به عنوان مراکز پر رونق علمی و پژوهشی، نیازمند راهکارهایی برای آشنایی بهتر دانشجویان جدید و حتی اتباع خارجی با محیط‌های آموزشی و اداری خود هستند. در همین راستا، پروژه حاضر با عنوان" اپلیکیشن واقعیت افزوده تشخیص ساختمان‌های دانشگاه "طراحی و اجرا شده تا با استفاده از فناوری‌های پیشرفته، تجربه‌ای متفاوت و تعاملی را برای کاربران فراهم آورد.

این پروژه با هدف ارائه یک راهکار نوین جهت شناسایی سریع و دقیق ساختمان‌های دانشگاه طراحی شده است. در این اپلیکیشن، کاربر با استفاده از یک دستگاه اندرویدی و محیطی تعاملی مبتنی برUnity ، قادر خواهد بود از طریق دوربین دستگاه، تصویری از ساختمان مورد نظر خود دریافت کند. پس از گرفتن تصویر، اطلاعات مربوط به ساختمان از طریق ارسال تصویر به سروری که با استفاده از فریمورک Flask نوشته شده و در یک بستر ابری مستقر می‌باشد، پردازش می‌شود. این سرور با بهره‌گیری از مدلی که با استفاده از الگوریتم YOLOv8 آموزش داده شده است، به صورت خودکار ساختمان را شناسایی کرده و نتیجه تشخیص را در قالب یک فایل JSON شامل آیدی اختصاصی مربوط به ساختمان، به اپلیکیشن بازمی‌گرداند.

از سوی دیگر، داده‌های برگشتی از مدل یادگیری ماشین با اطلاعات موجود در یک دیتابیس سبک که به صورت فایل JSON تنظیم شده و شامل جزئیات اطلاعات چهار ساختمان اصلی دانشگاه است، مقایسه می‌شود. در صورت تطابق آیدی‌های برگشتی و دیتابیس، اطلاعات مربوط به ساختمان شناسایی شده به کاربر به صورت یک پاپ‌آپ در محیط واقعیت افزوده نمایش داده می‌شود. این روند نه تنها دقت تشخیص را افزایش می‌دهد، بلکه تجربه کاربری را نیز بهبود می‌بخشد.

پروژه حاضر به عنوان پروژه پایانی دوره کارشناسی طراحی و اجرا شده و علاوه بر جنبه کاربردی، توانسته است چارچوبی تعمیم‌پذیر را ارائه دهد. این چارچوب به گونه‌ای طراحی شده است که تنها با تغییر مدل هوش مصنوعی، امکان استفاده از آن در حوزه‌های متفاوت و برای شناسایی اجسام یا المان‌های مختلف در محیط‌های دیگر و حتی شناسایی ساختمان های سایر دانشگاه ها نیز وجود داشته باشد. از این رو، این اپلیکیشن می‌تواند به عنوان یک نمونه موفق از ترکیب فناوری‌های واقعیت افزوده، بینایی کامپیوتر و یادگیری ماشین در پروژه‌های دانشگاهی و حتی صنعتی محسوب شود.

در بخش فنی پروژه، از زبان‌های برنامه‌نویسی Python برای پیاده‌سازی سمت سرور و فریمورک Flask جهت مدیریت درخواست‌ها، و از زبان C# در محیط Unity برای توسعه اپلیکیشن اندروید استفاده شده است. همچنین، برای مدیریت سرور و نظارت بر عملکرد آن، از اپلیکیشن Moboxterm بهره برده شده است. این ترکیب از تکنولوژی‌ها و ابزارها به توسعه یک سیستم یکپارچه و کارآمد منجر شده است که می‌تواند با چالش‌های متعددی همچون تأخیر ناشی از ارتباط با سرور ابری و محدودیت‌های ناشی از تحریم‌های بین‌المللی روبرو شده و در عین حال عملکرد مطلوب خود را حفظ کند.

یکی از چالش‌های مهم در اجرای این پروژه، ترکیب چندین تکنولوژی متفاوت در یک سیستم واحد بود. به عنوان مثال، ادغام صحیح بین بخش‌های توسعه در Unity ، پردازش تصویر با YOLOv8 و ارتباط موثر با سرور Flask از طریق API ها، نیازمند هماهنگی دقیق و تست‌های متعدد بوده است. علاوه بر این، محدودیت‌های ناشی از دسترسی به برخی ابزارهای تخصصی مانند Roboflow نیز در مسیر توسعه پروژه باعث تأخیرهایی در مراحل اولیه شده است. با این وجود، با برنامه‌ریزی دقیق و بهره‌گیری از تجربیات گذشته، این چالش‌ها به‌طور موفقیت‌آمیزی پشت سر گذاشته شدند.

هدف نهایی این پروژه، فراهم آوردن یک راهکار نوین و کاربردی برای دانشجویان جدید و اتباع خارجی جهت آشنایی سریع با محیط دانشگاه و ساختمان‌های مختلف آن است. از سوی دیگر، این اپلیکیشن به عنوان یک نمونه تعمیم‌پذیر، می‌تواند در پروژه‌های آتی که نیاز به شناسایی و ارائه اطلاعات بصری دارند، به کار گرفته شود. در نتیجه، پروژه حاضر نه تنها به رفع نیازهای فعلی پاسخ داده، بلکه افق‌های تازه‌ای را برای کاربردهای آینده در زمینه فناوری‌های واقعیت افزوده و هوش مصنوعی رقم زده است.

در ادامه این داکیومنت، به بررسی جامع جنبه‌های مختلف پروژه از جمله معماری سیستم، جزئیات فنی، فرآیند پیاده‌سازی، تست و ارزیابی عملکرد، و نیز چشم‌اندازهای بهبود در آینده پرداخته خواهد شد. این بررسی‌ها در کنار ارائه راهکارهای پیشنهادی جهت رفع مشکلات موجود، به عنوان مرجعی کامل برای توسعه‌دهندگان و علاقه‌مندان به این حوزه در نظر گرفته شده است.

بررسی کلی پروژه

۱. معرفی کلی پروژه

پروژه « ARchitect » با هدف ارائه یک سیستم تعاملی طراحی شده است که به دانشجویان جدید و اتباع خارجی کمک می‌کند تا به‌سرعت با محیط دانشگاه آشنا شوند. در این سیستم، کاربران با استفاده از دستگاه‌های اندرویدی و محیط توسعه‌یافته در موتور بازی‌سازی Unity، از طریق دوربین دستگاه، تصاویر ساختمان‌های دانشگاه را دریافت می‌کنند. سپس این تصاویر از طریق یک رابط API به سروری ارسال می‌شوند که وظیفه‌ی پردازش تصویر و تشخیص ساختمان‌ها را به عهده دارد. در نهایت، اطلاعات شناسایی شده به‌صورت یک پاپ‌آپ در محیط واقعیت افزوده نمایش داده می‌شود.

۲. جریان کار و عملکرد سیستم

سیستم از چند بخش اصلی تشکیل شده که به‌صورت یکپارچه عمل می‌کنند:

الف. ورود و راه‌اندازی اپلیکیشن

- ورود به اپلیکیشن:

کاربر با باز کردن اپلیکیشن اندرویدی وارد محیط کاربری می‌شود. در صفحه اصلی، یک دکمه اختصاصی مربوط به دانشگاه قرار داده شده است (در حال حاضر این اپلیکیشن فقط برای دانشگاه آزاد اسلامی واحد سنندج (شعبه مرکزی) در دسترس است). با فشردن این دکمه، کاربر به بخش دوربین واقعیت افزوده هدایت می‌شود. این صفحه طراحی شده با در نظر گرفتن سادگی و کاربرپسندی، تجربه کاربری مطلوبی را ارائه می‌دهد.

ب. دریافت تصویر از محیط

- عملیات دوربین AR :

پس از ورود به بخش دوربین، کاربر با استفاده از قابلیت‌های AR اپلیکیشن، تصویر ساختمان مورد نظر را ثبت می‌کند. طراحی این بخش به‌گونه‌ای است که علاوه بر ثبت تصویر، با فراهم آوردن فیدبک‌های بصری به کاربر اطلاع می‌دهد که عکس در حال گرفته شدن است.

پ. ارسال تصویر به سرور

- ارتباط با سرور:

بلافاصله پس از گرفتن عکس، تصویر از طریق یک API به سرور ارسال می‌شود. سرور با استفاده از فریمورک Flask پیاده‌سازی شده و در یک بستر ابری مستقر است. اپ Moboxterm نیز برای مدیریت و نظارت بر ارتباطات سروری به کار گرفته شده است. این روند ارتباطی به گونه‌ای طراحی شده است که با حداقل تأخیر، تصویر به سرور منتقل شود.

ت. پردازش تصویر با مدل YOLOv8

- تشخیص ساختمان:

در سرور، تصویر دریافت شده توسط یک مدل یادگیری ماشین که مبتنی بر الگوریتم YOLOv8 آموزش دیده، پردازش می‌شود. مدل YOLOv8 با استفاده از تکنیک‌های پیشرفته تشخیص اشیا، به‌طور خودکار ساختمان موجود در تصویر را شناسایی می‌کند. این مدل از داده‌های آموزشی برچسب‌گذاری‌شده با استفاده از ابزار Roboflow بهره می‌برد و از میان ۴ کلاس مختلف، ساختمان‌های دانشگاه را تشخیص می‌دهد.

نمونه خروجی مدل با Building ID مشخص

ج. تولید و ارسال پاسخ به اپلیکیشن

- خروجی مدل:

پس از شناسایی ساختمان، مدل یک فایل JSON حاوی یک آیدی منحصر به فرد برای ساختمان شناسایی شده تولید می‌کند. این آیدی در ساختار دیتابیس سبک فایل JSON ذخیره شده است که شامل اطلاعات چهار ساختمان اصلی دانشگاه می‌باشد.

- مقایسه آیدی‌ها:

سرور، آیدی برگشتی را با آیدی‌های موجود در دیتابیس مقایسه می‌کند. در صورتی که تطابقی یافت شود، اطلاعات مرتبط با ساختمان (مانند نام، طبقات و بخش ها، کاربرد و توضیحات تکمیلی) آماده‌ی ارسال به اپلیکیشن می‌شود.


```
{
    "buildings": [
      {
        "id": 1,
        "name": "Faculty of Engineering 1",
        "image": "fani_1",
        "description": "Faculty of Engineering 1 includes the departments of Electrical and Computer
Engineering, featuring various labs, classrooms, and administrative offices.",
        "floors": [
          {
            "floorNumber": 1,
            "floorName": "Ground Floor",
            "floorDescription": "Main entrance, auditorium, computer site, and classrooms 101–102."
          },
          {
            "floorNumber": 2,
            "floorName": "First Floor",
            "floorDescription": "Classrooms 103–106, a conference room, and some faculty offices."
          },
          {
            "floorNumber": 3,
            "floorName": "Second Floor",
            "floorDescription": "Electronics, control, and microprocessor labs, plus additional
classrooms."
          },
          {
            "floorNumber": 4,
            "floorName": "Third Floor",
            "floorDescription": "Dean’s office, vice-dean offices, educational affairs, and
departmental offices."
          }
        ]
      },
```


قسمتی از دیتابیس اپلیکیشن برای مقایسه ی ID ساختمان ها

- نمایش پاپ‌آپ در محیط AR :

اطلاعات استخراج شده از دیتابیس به عنوان یک پاپ‌آپ در محیط AR نمایش داده می‌شود. این پاپ‌آپ به گونه‌ای طراحی شده است که اطلاعات به صورت خوانا و جذاب به کاربر ارائه گردد.

۳. توضیحات فنی اجزا و ارتباطات

سیستم مورد نظر از چندین جزء کلیدی تشکیل شده که در ادامه به بررسی هر یک از آن‌ها می‌پردازیم:

الف. اپلیکیشن اندرویدی مبتنی برUnity

- توسعه و طراحی:

اپلیکیشن با استفاده از Unity توسعه یافته و شامل اجزای تعاملی، دوربین AR و رابط‌های کاربری مناسب است. طراحی این بخش با در نظر گرفتن استانداردهای مدرن تجربه کاربری و به‌کارگیری بهترین شیوه‌های طراحی رابط کاربری انجام شده است.

- قابلیت‌های تعاملی:

اپلیکیشن به کاربر امکان می‌دهد تا با استفاده از دوربین، تصاویر ساختمان های دانشگاه را در لحظه ثبت کند و در همان لحظه بازخورد را به صورت واقعیت افزوده دریافت کند. این امر، با استفاده از پاپ‌آپ‌های تعاملی، اطلاعات مربوطه در خصوص ساختمان‌های شناسایی‌شده را به کاربر نمایش می دهد.

ب. سرور پردازش با Flask

- پیاده‌سازی:

سرور از طریق فریمورک Flask پیاده‌سازی شده و مسئول دریافت تصاویر ارسالی از اپلیکیشن، ارسال آن‌ها به مدل YOLOv8 و پردازش خروجی‌ها می‌باشد. ارتباط بین اپلیکیشن و سرور از طریقAPI های RESTfulبرقرار می‌شود.

- مدیریت ارتباط:

استفاده از اپ Moboxterm در مدیریت سرور و نظارت بر درخواست‌ها تضمین می‌کند که ارتباط بین اپلیکیشن و سرور به صورت پیوسته و بدون اختلال برقرار است. این امر نقش مهمی در کاهش تاخیر و افزایش کارایی سیستم دارد.

پ. مدل یادگیری ماشین YOLOv8

- عملکرد و آموزش:

مدل YOLOv8 از تکنیک‌های مدرن تشخیص اشیا بهره می‌برد و با استفاده از مجموعه‌ای از تصاویر برچسب‌گذاری شده، آموزش دیده است. عملکرد مدل در تشخیص ساختمان‌های دانشگاه از طریق تحلیل ویژگی‌های بصری هر ساختمان صورت می‌گیرد.

- خروجی مدل:

خروجی مدل به‌صورت یک فایل JSON شامل آیدی اختصاصی تولید می‌شود. این آیدی به عنوان شناسه‌ی یکتای ساختمان در دیتابیس استفاده شده و روند مقایسه با اطلاعات موجود را تسهیل می‌کند.

ج. دیتابیس سبک مبتنی برJSON

- ساختار دیتابیس:

دیتابیس پروژه به صورت یک فایل JSON طراحی شده است که شامل اطلاعات جامع چهار ساختمان اصلی دانشگاه می‌باشد. هر ساختمان با یک آیدی منحصربه‌فرد همراه با جزئیات مربوط به آن (مانند نام، طبقات و بخش ها، کاربرد و سایر توضیحات) ثبت شده است.

- فرایند مقایسه:

آیدی برگشتی از مدل YOLOv8 با آیدی‌های موجود در دیتابیس مقایسه می‌شود. در صورت تطابق، اطلاعات مربوط به ساختمان از دیتابیس استخراج شده و به کاربر ارسال می‌گردد.

۴ .تجزیه و تحلیل عملکرد و مزایا

پروژه علاوه بر ارائه یک راهکار نوین، مزایای متعددی برای کاربران به همراه دارد:

- دقت بالا در تشخیص:

استفاده از مدل YOLOv8 باعث افزایش دقت تشخیص ساختمان‌ها شده است. مدل با دقت ۹۸.۶٪ train شد و آزمایش‌های متعدد با تصاویر مختلف نشان داده‌اند که مدل توانسته است در شرایط نوری و جوی متفاوت عملکرد مطلوبی ارائه دهد.

- تجربه کاربری بهبود یافته:

نمایش اطلاعات به‌صورت پاپ‌آپ‌های تعاملی در محیطAR ، تجربه کاربری را به مراتب بهبود بخشیده است. این امر به ویژه برای دانشجویان جدید و اتباع خارجی که نیاز به آشنایی سریع با محیط دانشگاه دارند، بسیار مفید است.

- قابلیت تعمیم:

چارچوب طراحی شده در این پروژه به گونه‌ای است که با تغییر مدل هوش مصنوعی، امکان استفاده از آن در حوزه‌های دیگر مانند شناسایی اشیاء در محیط‌های شهری یا صنعتی وجود دارد. همچنین میتوان از این اپلیکیشن برای شناسایی سایر ساختمان های دانشگاه های دیگر نیز استفاده کرد. با اضافه کردن عکس های سایر اشیا یا ساختمان ها، امکان آموزش مدل هوش مصنوعی برای موارد موردنظر نیز امکان پذیر است.

۵ .چالش‌های اجرایی و راهکارهای پیشنهادی

اجرای یک سیستم چندتکنیلی همچون این پروژه، همواره با چالش‌های فنی و اجرایی همراه است. از جمله مهم‌ترین چالش‌ها می‌توان به موارد زیر اشاره کرد:

- ادغام چندین تکنولوژی:

ترکیبUnity ، Flask، YOLOv8و سیستم‌های مدیریت سرور نیازمند هماهنگی دقیق و تست‌های مکرر بوده است. راهکار ارائه شده شامل تقسیم‌بندی وظایف و ایجاد رابط‌های ارتباطی استاندارد بین اجزا می‌باشد.

- تاخیر در ارتباط با سرور:

به دلیل وابستگی به ارتباط اینترنتی و سرور ابری، زمان پاسخ‌دهی ممکن است تحت تأثیر قرار گیرد. استفاده از ابزارهایی مانند Moboxterm برای نظارت بر ارتباطات و برنامه‌ریزی جهت انتقال دیتابیس به صورت لوکال در نسخه‌های آتی از جمله راهکارهای پیشنهادی است.

- محدودیت‌های ناشی از تحریم‌ها:

محدودیت‌های فنی و دسترسی به برخی ابزارها، همچون استفاده ازRoboflow ، چالش دیگری در مسیر توسعه به‌شمار می‌آید. با استفاده از ابزارهای جایگزین و بهره‌گیری از تجربیات گذشته، توانسته‌ایم این محدودیت‌ها را تا حد امکان کاهش دهیم.

در مجموع، سیستم طراحی شده نه تنها یک راهکار نوین برای شناسایی ساختمان‌های دانشگاه ارائه می‌دهد، بلکه با بهره‌گیری از فناوری‌های AR و یادگیری ماشین، تجربه‌ای کاربر محور و تعاملی را فراهم می‌سازد. ارتباط بین اجزای مختلف سیستم به صورت یکپارچه طراحی شده است تا از ارسال به موقع تصاویر، پردازش دقیق آن‌ها و نمایش اطلاعات مربوط به ساختمان اطمینان حاصل شود. این روند جامع، که از ورود کاربر تا دریافت نتیجه نهایی ادامه دارد، نشان‌دهنده‌ی موفقیت در ادغام چندین تکنولوژی و به‌کارگیری بهترین شیوه‌های توسعه نرم‌افزار در یک پروژه پایانی دانشگاهی می‌باشد.

اهداف و دستاوردهای پروژه

۱ .اهداف پروژه

۱.۱ .اهداف کلی پروژه

هدف اصلی پروژه « ARchitect » ایجاد یک سامانه تعاملی و هوشمند جهت شناسایی دقیق ساختمان‌های دانشگاه و ارائه اطلاعات مرتبط به صورت بصری در قالب پاپ‌آپ‌های واقعیت افزوده است. این هدف از منظرهای متعددی دنبال شده است:

- بهبود تجربه کاربری:

با بهره‌گیری از فناوری‌های AR و هوش مصنوعی، کاربران (به ویژه دانشجویان جدید و اتباع خارجی) می‌توانند به‌سرعت با محیط دانشگاه آشنا شده و اطلاعات لازم را بدون نیاز به مراجعه به منابع متعدد یا پرس و جوهایی که شاید بی نتیجه یا وقت گیر باشد دریافت نمایند.

- ارتقاء کارایی و دقت تشخیص:

استفاده از مدل YOLOv8 باعث شده تا فرآیند تشخیص ساختمان‌ها در تصاویر گرفته شده از طریق دوربین اپلیکیشن به صورت دقیق و سریع انجام شود. این دقت بالا به واسطه آموزش مدل با تصاویر برچسب‌گذاری شده و بهینه‌سازی‌های انجام شده در طول پروژه حاصل شده است.

- ایجاد چارچوب تعمیم‌پذیر

علاوه بر کاربرد در محیط دانشگاه، طراحی سیستم به گونه‌ای انجام شده است که تنها با تغییر مدل هوش مصنوعی و به‌روزرسانی دیتابیس، امکان استفاده از این چارچوب در حوزه‌های دیگر مانند شناسایی اشیاء شهری، صنعتی یا توریستی وجود دارد. این ویژگی تعمیم‌پذیری، ارزش افزوده قابل توجهی برای پروژه به ارمغان می‌آورد.

```
import os
import zipfile

# Define the path to your uploaded file
zip_path = "/content/University-AR.v3i.yolov8.zip"

# Unzip the file
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall("/content/University-AR")

# Verify extraction
os.listdir("/content/University-AR")

------------------------------
import os
HOME = os.getcwd()
print(HOME)
------------------------------
# load ylov8 object detection model from ultralytics
!pip install ultralytics
import ultralytics
ultralytics.checks()
------------------------------
# Change directory to HOME
%cd {HOME}

# Train the YOLOv8 model on the dataset
!yolo task=detect mode=train model=yolov8s.pt data=/content/University-AR/data.yaml epochs=17 imgsz=800
plots=True
------------------------------
from ultralytics import YOLO

# Load the YOLOv8 model
model = YOLO("/content/runs/detect/train3/weights/best.pt")

# Export the model to ONNX
model.export(format="onnx")
```

کدهای مربوط به train کردن مدل هوش مصنوعی

۱.۲ .اهداف کاربردی

از دیدگاه کاربردی، اهداف پروژه به شرح زیر تعریف شده‌اند:

- راهنمایی و آشنایی سریع:

با ارائه اطلاعات دقیق ساختمان‌ها از جمله نام، طبقات و بخش های مختلف، کاربرد و سایر جزئیات تکمیلی، اپلیکیشن می‌تواند به دانشجویان جدید و اتباع خارجی کمک کند تا در کمترین زمان با محیط دانشگاه و امکانات آن آشنا شوند.

- کاهش زمان و هزینه:

با استفاده از یک سامانه خودکار جهت تشخیص و ارائه اطلاعات، نیاز به برگزاری جلسات آشنایی یا استفاده از راهنماهای چاپی کاهش یافته و از این طریق هم زمان و هم هزینه صرفه‌جویی می‌شود.

- ارتباط بی‌وقفه و به‌روز:

سیستم با بهره‌گیری از سرور پردازش و مدیریت به کمک اپMoboxterm، تضمین می‌کند که اطلاعات ارائه شده همواره به‌روز و مرتبط با محیط فعلی دانشگاه باشند.

۱.۳ .اهداف فنی و توسعه‌ای

از منظر فنی، اهداف پروژه شامل موارد زیر است:

- ادغام چندین تکنولوژی مدرن:

ترکیبUnity ، Flask، YOLOv8، و مدیریت دیتابیس سبک JSON به گونه‌ای انجام شده است که ارتباط بین اجزا به صورت یکپارچه و بدون اختلال برقرار باشد.

- افزایش دقت و کارایی مدل:

با استفاده از داده‌های برچسب‌گذاری شده و بهبود فرآیند آموزش مدلYOLOv8 ، هدف افزایش دقت تشخیص ساختمان‌ها و کاهش خطاهای مدل در شرایط نوری و محیطی مختلف دنبال شده است.

- ایجاد سیستم مقیاس‌پذیر:

طراحی سیستم به گونه‌ای صورت گرفته است که در صورت نیاز به توسعه بیشتر، امکان انتقال دیتابیس از فضای ابری به حالت لوکال، افزایش تعداد کلاس‌های تشخیصی و بهبود رابط‌های ارتباطی بین اپلیکیشن و سرور وجود داشته باشد.

۲ .دستاوردهای پروژه

۲.۱ .دستاوردهای فنی

- یکپارچگی تکنولوژی‌های مختلف:

پروژه موفق شده است تا با ادغام تکنولوژی‌های متفاوت از جمله Unity برای توسعه اپلیکیشن اندرویدی، Flask برای پیاده‌سازی سمت سرور، YOLOv8 برای تشخیص ساختمان‌ها و استفاده از JSON جهت مدیریت دیتابیس، یک سیستم یکپارچه و کارآمد را ارائه دهد. این دستاورد نشان‌دهنده توانمندی در حل چالش‌های ناشی از هماهنگی میان چندین زبان برنامه‌نویسی و فریمورک‌های مختلف است.

- تشخیص دقیق و سریع ساختمان‌ها:

مدل YOLOv8 با استفاده از الگوریتم‌های پیشرفته تشخیص اشیا، توانسته است در شرایط مختلف (نور، زاویه و پس‌زمینه‌های متفاوت) دقت بالایی در شناسایی ساختمان‌های دانشگاه ارائه دهد. خروجی این مدل به صورت فایل JSON حاوی آیدی منحصر به فرد برای هر ساختمان است که امکان مقایسه سریع با دیتابیس فراهم می‌شود.

- ارتقاء عملکرد سرور و کاهش تاخیر:

با استفاده از فریمورک Flask و مدیریت هوشمند ارتباط با سرور ابری به کمک اپMoboxterm ، زمان پاسخ‌دهی سیستم به حداقل رسیده است. این امر نقش بسزایی در بهبود تجربه کاربری داشته و باعث شده تا فرآیند ارسال و دریافت داده‌ها با سرعت بالا انجام شود.

- طراحی سیستم مقیاس‌پذیر:

طراحی سیستم به گونه‌ای است که در صورت افزایش نیازها و اضافه شدن اطلاعات جدید (مثلاً افزایش تعداد ساختمان‌ها یا تغییرات در دیتابیس)، قابلیت مقیاس‌پذیری و به‌روزرسانی سیستم به آسانی فراهم می‌شود. این ویژگی باعث می‌شود که پروژه به عنوان یک چارچوب پایه برای کاربردهای مشابه در آینده مورد استفاده قرار گیرد.

۲.۲ .دستاوردهای کاربری

- تجربه کاربری بهبود یافته:

با ارائه یک رابط کاربری ساده، شفاف و تعاملی در محیطAR ، کاربران می‌توانند به‌راحتی اطلاعات مورد نیاز خود را دریافت کنند. نمایش اطلاعات به صورت پاپ‌آپ‌های جذاب و خوانا، باعث شده تا فرآیند آشنایی با محیط دانشگاه به تجربه‌ای لذت‌بخش تبدیل شود.

- دسترسی سریع به اطلاعات:

از طریق گرفتن یک عکس از ساختمان مورد نظر و دریافت فوری اطلاعات مرتبط، کاربران نیازی به جستجو یا مراجعه به منابع متعدد ندارند. این دسترسی سریع به اطلاعات، به ویژه برای دانشجویان جدید و اتباع خارجی که زمان برای آشنایی محدود است، بسیار کارآمد و مفید می‌باشد.

- افزایش تعامل و رضایت کاربر:

سیستم تعاملی و پاسخگو باعث افزایش رضایت کاربران از اپلیکیشن شده است. تعامل مستقیم کاربر با محیطAR ، به همراه بازخوردهای بصری و اطلاعات دقیق، زمینه ایجاد تعامل بیشتر و استفاده مستمر از سیستم را فراهم آورده است.

۲.۳ .دستاوردهای مدیریتی و توسعه آتی

- رفع چالش‌های فنی و مدیریتی:

در مواجهه با چالش‌هایی مانند محدودیت‌های ناشی از تحریم‌های بین‌المللی و مشکلات ادغام تکنولوژی‌های متفاوت، با به کارگیری راهکارهای نوین و استفاده از ابزارهایی مانند Moboxterm ، موفقیت‌های چشمگیری حاصل شده است. این دستاورد نشان‌دهنده توانمندی تیم در شناسایی مشکلات و ارائه راه‌حل‌های خلاقانه می‌باشد.

- پتانسیل توسعه و بهبود آتی:

یکی از دستاوردهای مهم پروژه، ایجاد بستری برای بهبودهای آینده است. برنامه‌های آتی شامل انتقال دیتابیس از فضای ابری به حالت لوکال به منظور افزایش سرعت و کاهش تأخیر، توسعه دیتاست با افزودن

تصاویر جدید در شرایط مختلف نوری و جوی، و بهبود مدل تشخیص با استفاده از داده‌های بیشتر است. این مسیر توسعه، افق‌های تازه‌ای را برای کاربردهای آینده در حوزه‌های مختلف باز می‌کند.

- ایجاد چارچوب تعمیم‌پذیر:

پروژه با طراحی یک چارچوب ساختاری منسجم و قابل تعمیم، امکان استفاده مجدد از سیستم در پروژه‌های مختلف را فراهم کرده است. با تغییر مدل هوش مصنوعی و به‌روزرسانی دیتابیس، این سیستم می‌تواند در حوزه‌های متفاوت مانند شناسایی اشیاء در محیط‌های شهری، صنعتی و حتی گردشگری به کار گرفته شود.

معماری سیستم

۱. معرفی کلی

معماری سیستم پروژه «ARchitect» به‌عنوان یک سامانه چندتکنیلی طراحی شده است که از دو بخش اصلی تشکیل می‌شود:

- بخش کلاینت (اپلیکیشن اندرویدی) : توسعه یافته در محیط Unity برای دریافت تصاویر، نمایش پاپ‌آپ‌های اطلاعاتی و تعامل با کاربر.
- بخش سرور (پشت‌صحنه): پیاده‌سازی شده با استفاده از Flask در بستر ابری برای پردازش تصاویر، اجرای مدل YOLOv8 و مدیریت دیتابیسJSON .

۲. اجزای اصلی سیستم

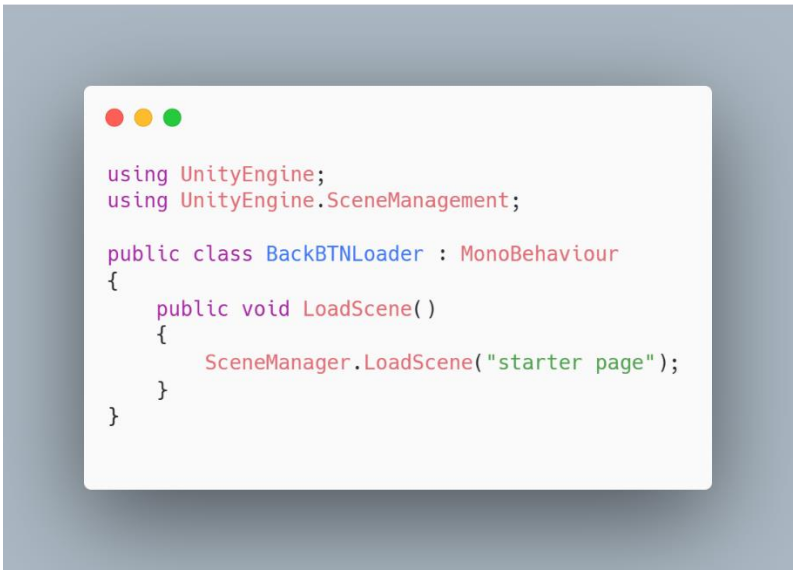
۱.۲. اپلیکیشن اندرویدی مبتنی برUnity

- رابط کاربری و توسعه :
  - صفحه ورود، بخش دوربین AR و نمایش پاپ‌آپ‌های اطلاعاتی.
  - نمونه کد مدیریت پاپ‌آپ:

```
using UnityEngine;
using UnityEngine.UI;
using TMPro;

public class ARPopupManager : MonoBehaviour
{
    public static ARPopupManager Instance;

    [Header("Popup UI")]
    public GameObject popupPanel;
    public TMP_Text buildingNameText;
    public TMP_Text confidenceText;
    public TMP_Text buildingInfoText;
    public Button closeButton;

    private int currentBuildingId = -1;

    private void Awake()
    {
        if (Instance == null) Instance = this;
        else Destroy(gameObject);

        if (popupPanel != null)
            popupPanel.SetActive(false);
    }

    // Overload: نمایش پاپ‌آپ با مشخصات ساختمان
    public void ShowPopup(int buildingId, float confidence, Vector3 position = default)
    {
        currentBuildingId = buildingId;
        Building building = BuildingDatabase.Instance.GetBuildingById(buildingId);
        if (building == null)
        {
            Debug.LogError($"ARPopupManager: Building with ID {buildingId} not found.");
            return;
        }

        if (buildingNameText != null)
            buildingNameText.text = building.name;

        if (confidenceText != null)
            confidenceText.text = $"Confidence: {(confidence * 100f):0.0}%";

        if (buildingInfoText != null)
        {
            string info = building.description;
            if (building.floors != null && building.floors.Length > 0)
            {
                info += "\n\nFloors:\n";
                foreach (var floor in building.floors)
                {
                    info += $"- Floor {floor.floorNumber} ({floor.floorName}):
{floor.floorDescription}\n";
                }
            }
            buildingInfoText.text = info;
        }

        popupPanel.SetActive(true);
    }

    public void HidePopup()
    {
        if (popupPanel != null)
            popupPanel.SetActive(false);
    }

    public void OnClick_Close()
    {
        HidePopup();
    }
}
```

دکمه‌های ناوبری :

- نمونه کد دکمه بازگشت:



```
using UnityEngine;
using UnityEngine.SceneManagement;

public class BackBTNLoader : MonoBehaviour
{
    public void LoadScene()
    {
        SceneManager.LoadScene("starter page");
    }
}
```

مدیریت دیتابیس ساختمان‌ها :

- نمونه کد دیتابیس جهت بارگذاری اطلاعات ساختمان‌ها از فایل JSON :

```
using System.Collections.Generic;
using UnityEngine;

[System.Serializable]
public class FloorInfo
{
    public int floorNumber;
    public string floorName;
    public string floorDescription;
}

[System.Serializable]
public class Building
{
    public int id;
    public string name;
    public string image;
    public string description;
    public FloorInfo[] floors;
}

[System.Serializable]
public class BuildingList
{
    public Building[] buildings;
}

public class BuildingDatabase : MonoBehaviour
{
    public static BuildingDatabase Instance;

    [Tooltip("If null, will try to load 'buildings.json' from Resources folder.")]
    public TextAsset buildingsJson;

    private Dictionary<int, Building> buildingDict = new Dictionary<int, Building>();

    private void Awake()
    {
        if (Instance == null)
        {
            Instance = this;
            DontDestroyOnLoad(gameObject);
            LoadBuildings();
        }
        else
        {
            Destroy(gameObject);
        }
    }

    private void LoadBuildings()
    {
        if (buildingsJson == null)
        {
            buildingsJson = Resources.Load<TextAsset>("buildings");
        }

        if (buildingsJson == null)
        {
            Debug.LogError("BuildingDatabase: Could not find buildings.json in Resources or assigned in
Inspector!");
            return;
        }

        BuildingList list = JsonUtility.FromJson<BuildingList>(buildingsJson.text);
        if (list != null && list.buildings != null)
        {
            foreach (Building b in list.buildings)
            {
                buildingDict[b.id] = b;
            }
            Debug.Log($"BuildingDatabase: Loaded {buildingDict.Count} buildings.");
        }
        else
        {
            Debug.LogError("BuildingDatabase: Failed to parse buildings.json or it's empty.");
        }
    }

    public Building GetBuildingById(int buildingId)
    {
        buildingDict.TryGetValue(buildingId, out Building building);
        return building;
    }
}
```

۲.۲. سرور پردازش مبتنی برFlask

- پیاده‌سازی API :
  - نمونه کد در Unity جهت ارسال تصویر به سرور:

```
using UnityEngine;
using UnityEngine.Networking;
using System.Collections;
using System.Collections.Generic;

[System.Serializable]
public class Prediction
{
    public List<float> bbox;
    public string class_name;
    public float confidence;
}

[System.Serializable]
public class PredictionResponse
{
    public List<Prediction> predictions;
}

public class FlaskAPIClient : MonoBehaviour
{
    private string serverURL = "http://176.97.218.195:5000/predict";

    public Camera captureCamera;
    public RenderTexture renderTexture;
    public GameObject buildingInfoPanel;
    public TMPro.TextMeshProUGUI infoText;

    void Start()
    {
        buildingInfoPanel.SetActive(false);
    }

    public void CaptureAndSend()
    {
        Debug.Log("Starting image capture...");
        Texture2D texture = new Texture2D(renderTexture.width, renderTexture.height,
TextureFormat.RGB24, false);
        RenderTexture.active = renderTexture;
        captureCamera.Render();
        texture.ReadPixels(new Rect(0, 0, renderTexture.width, renderTexture.height), 0, 0);
        texture.Apply();
        Debug.Log("Image captured successfully.");

        byte[] imageData = texture.EncodeToJPG();
        Debug.Log("Image encoded to JPG.");

        StartCoroutine(SendImageToServer(imageData));
    }

    public IEnumerator SendImageToServer(byte[] imageData)
    {
        Debug.Log("Sending image data to server...");
        WWWForm form = new WWWForm();
        form.AddBinaryData("image", imageData, "image.jpg", "image/jpeg");

        using (UnityWebRequest request = UnityWebRequest.Post(serverURL, form))
        {
            yield return request.SendWebRequest();

            if (request.result != UnityWebRequest.Result.Success)
            {
                Debug.LogError("Error sending image: " + request.error);
            }
            else
            {
                Debug.Log("Image sent successfully. Server response received:");
                Debug.Log("Response: " + request.downloadHandler.text);
                ProcessServerResponse(request.downloadHandler.text);
            }
        }
    }

    private void ProcessServerResponse(string jsonResponse)
    {
        Debug.Log("Processing server response...");
        PredictionResponse response = JsonUtility.FromJson<PredictionResponse>(jsonResponse);

        foreach (Prediction pred in response.predictions)
        {
            Debug.Log($"Detected: {pred.class_name} - Confidence: {pred.confidence}");

        }
    }
}
```

۳.۲. نمونه پیاده‌سازی اسکرین‌شات و بارگذاری تصویر

- نمونه کد جهت گرفتن اسکرین‌شات، آپلود و پردازش پاسخ سرور:

```
using System.Collections;
using UnityEngine;
using UnityEngine.Networking;
using TMPro;

namespace MyProject
{
    [System.Serializable]
    public class Floor
    {
        public int floorNumber;
        public string floorName;
        public string floorDescription;
    }

    [System.Serializable]
    public class PredictionResult
    {
        public float[] bbox;
        public string class_name;
        public float confidence;
        public int building_id;
        public string display_name;
        public string description;
        public Floor[] floors;
    }

    [System.Serializable]
    public class PredictionsWrapper
    {
        public PredictionResult[] predictions;
    }

    public class ScreenshotUploader : MonoBehaviour
    {
        [Header("Server Settings")]
        [SerializeField] private string serverUrl = "http://176.97.218.195:5000/predict";

        [Header("UI References")]
        public TMP_Text statusText;

        public void CaptureAndUpload()
        {
            SetStatus("Capturing screenshot...");
            StartCoroutine(TakeScreenshotAndUpload());
        }

        private IEnumerator TakeScreenshotAndUpload()
        {
            yield return new WaitForEndOfFrame();

            SetStatus("Processing screenshot...");

            Texture2D screenshot = new Texture2D(Screen.width, Screen.height, TextureFormat.RGB24,
false);
            screenshot.ReadPixels(new Rect(0, 0, Screen.width, Screen.height), 0, 0);
            screenshot.Apply();

            byte[] imageBytes = screenshot.EncodeToPNG();
            Destroy(screenshot);

            WWWForm form = new WWWForm();
            form.AddBinaryData("image", imageBytes, "screenshot.png", "image/png");

            SetStatus("Uploading to server...");

            using (UnityWebRequest request = UnityWebRequest.Post(serverUrl, form))
            {
                yield return request.SendWebRequest();

                if (request.result == UnityWebRequest.Result.Success)
                {
                    Debug.Log("Image uploaded successfully!");
                    Debug.Log("Response: " + request.downloadHandler.text);
                    SetStatus("Server response received.");
                    ProcessServerResponse(request.downloadHandler.text);
                }
                else
                {
                    Debug.LogError("Failed to upload image: " + request.error);
                    SetStatus($"Upload error: {request.error}");
                }
            }
        }
```

```
        private void ProcessServerResponse(string jsonResponse)
        {
            if (string.IsNullOrEmpty(jsonResponse))
            {
                SetStatus("No response from server.");
                return;
            }

            PredictionsWrapper predictions = JsonUtility.FromJson<PredictionsWrapper>(jsonResponse);
            if (predictions == null || predictions.predictions == null ||
predictions.predictions.Length == 0)
            {
                SetStatus("No predictions in server response.");
                return;
            }

            PredictionResult first = predictions.predictions[0];

            if (first.confidence < 0.2f)
            {
                SetStatus($"Confidence too low ({first.confidence * 100f:0.0}%). No match found.");
                return;
            }

            ARPopupManager.Instance.ShowPopup(first.building_id, first.confidence);
            Debug.Log($"Detected {first.class_name} - {first.display_name}: {first.description}");
            SetStatus($"Detected {first.class_name} ({first.confidence * 100f:0.0}%).");
        }

        private void SetStatus(string msg)
        {
            Debug.Log("Status: " + msg);
            if (statusText != null)
            {
                statusText.text = msg;
            }
        }
    }
}
```

۲.۴. دکمه‌ها و کنترل صحنه

- نمونه کد جهت تغییر صحنه:

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class UniversityOptions : MonoBehaviour
{
    public void LoadScene()
    {
        SceneManager.LoadScene("MainScene");
    }
}
```

- نمونه ساده جهت تغییر صحنه دیگر:

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class ChangeScene : MonoBehaviour
{
    public void LoadScene()
    {
        SceneManager.LoadScene("Options");
    }
}
```

- نمایش پیام "به زودی" برای امکانات آتی:

```
using UnityEngine;
using TMPro;

public class ComingSoonMessage : MonoBehaviour
{
    public TextMeshProUGUI comingSoonText;
    public string message = "Feature coming soon!";

    private void Awake()
    {
        if (comingSoonText != null)
        {
            comingSoonText.text = "";
            comingSoonText.gameObject.SetActive(false);
        }
    }

    public void ShowComingSoon()
    {
        if (comingSoonText != null)
        {
            comingSoonText.text = message;
            comingSoonText.gameObject.SetActive(true);
        }
    }

    public void HideComingSoon()
    {
        if (comingSoonText != null)
            comingSoonText.gameObject.SetActive(false);
    }
}
```

۳. جریان داده و ارتباط بین اجزا

- ارسال تصویر از اپلیکیشن به سرور :
  - کاربر در اپلیکیشن تصویر را ثبت کرده و از طریق API به سرور ارسال می‌کند (کد مربوط به FlaskAPIClient و ScreenshotUploader)
- پردازش تصویر در سرور :
  - سرور تصویر دریافت‌شده را با استفاده از مدل YOLOv8 پردازش و خروجی JSON شامل آیدی و اطلاعات ساختمان را تولید می‌کند.
- نمایش اطلاعات در اپلیکیشن :
  - اطلاعات برگشتی از سرور توسط ARPopupManager دریافت و به صورت پاپ‌آپ در محیط AR نمایش داده می‌شود.

جزئیات فنی و تکنولوژی‌های مورد استفاده

۱. زبان‌های برنامه‌نویسی

۱.۱ Python:

- کاربرد:
  - پیاده‌سازی سرور جهت دریافت و پردازش تصاویر، توسعهAPI های ارتباطی و اجرای مدل YOLOv8.
- ویژگی‌ها و مزایا:
  - زبان منعطف با کتابخانه‌های متعدد برای پردازش تصویر و یادگیری ماشین.
- نمونه کد سرور:(Flask + YOLOv8)

```python
from flask import Flask, request, jsonify
from ultralytics import YOLO
import cv2
import os
import json

app = Flask(__name__)

# بارگذاری داده‌های ساختمان‌ها
with open("buildings.json", "r", encoding="utf-8") as f:
    building_data = json.load(f)

MODEL_PATH = "best.pt"
if not os.path.exists(MODEL_PATH):
    raise FileNotFoundError(f"Model file {MODEL_PATH} not
found!")
# بارگذاری مدل
model = YOLO(MODEL_PATH)

class_names = ['Fani 1', 'Fani 2', 'Masjed', 'Library']

class_to_building_id = {
    "Fani 1": 1,
    "Fani 2": 2,
    "Masjed": 3,
    "Library": 4
}

class_to_building = {}
for building in building_data["buildings"]:
    b_id = building.get("id")
    if b_id == 1:
        class_to_building["Fani 1"] = building
    elif b_id == 2:
        class_to_building["Fani 2"] = building
    elif b_id == 3:
        class_to_building["Masjed"] = building
    elif b_id == 4:
        class_to_building["Library"] = building

UPLOAD_FOLDER = "uploads"
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
```

```python
@app.route("/")
def home():
    return "YOLOv8 Flask API is running!"

@app.route("/predict", methods=["POST"])
def predict():
    if "image" not in request.files:
        return jsonify({"error": "No image file provided"}), 400

    file = request.files["image"]
    if not file.filename.lower().endswith(('.png', '.jpg', '.jpeg')):
        return jsonify({"error": "Invalid file format. Please upload a PNG or JPG file."}), 400

    filepath = os.path.join(UPLOAD_FOLDER, file.filename)
    file.save(filepath)

    results = model(filepath)

    output = []
    for result in results:
        for box in result.boxes:
            cls_id = int(box.cls[0].item())
            confidence = float(box.conf[0].item())
            bbox = [float(x) for x in box.xyxy[0].tolist()]

            if cls_id >= len(class_names):
                detected_class = f"Unknown Class {cls_id}"
                building_info = None
            else:
                detected_class = class_names[cls_id]
                building_id = class_to_building_id.get(detected_class)
                building_info = class_to_building.get(detected_class)

            prediction = {
                "bbox": bbox,
                "class_name": detected_class,
                "confidence": confidence,
                "building_id": building_id,
            }

            if building_info:
                prediction["display_name"] = building_info["name"]
                prediction["description"] = building_info["description"]
                prediction["floors"] = building_info["floors"]

            output.append(prediction)

    os.remove(filepath)

    return jsonify({"predictions": output})

if __name__ == "__main__":
    print("Flask server is starting...")
    app.run(host='0.0.0.0', port=5000, debug=True)
```

۲.۱ C# :

- کاربرد :
  - توسعه اپلیکیشن‌های اندرویدی در Unity جهت مدیریت رابط کاربری، تعاملات AR و ارتباط با سرور.
- ویژگی‌ها و مزایا :
  - زبان شیءگرا با قابلیت‌های پیشرفته در محیط‌های گرافیکی.
- نمونه‌های کد ارائه‌شده :
  - ARPopupManager، BackBTNLoader، BuildingDatabase، FlaskAPIClient، ScreenshotUploader، UniversityOptions، ComingSoonMessage، ChangeScene

۲. فریمورک‌ها و محیط‌های توسعه

۲.۱ Unity :

- کاربرد :
  - توسعه اپلیکیشن‌های واقعیت افزوده، ایجاد محیط‌های تعاملی و پیاده‌سازی اسکریپت‌های#C
- ویژگی‌ها :
  - محیط توسعه یکپارچه (IDE) با امکانات پیشرفته و پشتیبانی از اسکریپت‌نویسی به زبان#C

۲.۲ Flask :

- کاربرد :
  - پیاده‌سازی سرور وAPI های RESTful جهت دریافت و پردازش تصاویر ارسال‌شده از اپلیکیشن.

- ویژگی‌ها :
  - میکروفریمورک سبک و انعطاف‌پذیر در Python با امکان توسعه سریع.

۳. مدل‌های یادگیری ماشین و کتابخانه‌های پردازش تصویر

۳.۱ YOLOv8 :

- کاربرد :
  - تشخیص ساختمان‌ها از روی تصاویر دریافتی و تولید خروجی JSON حاوی آیدی و اطلاعات ساختمان.
- ویژگی‌ها و مزایا :
  - دقت بالا در شرایط متنوع نوری و محیطی.
- فرآیند آموزش :
  - آموزش با استفاده از تصاویر برچسب‌گذاری‌شده (Roboflow)

۳.۲ کتابخانه‌های پردازش تصویر مانند(OpenCV) :

- کاربرد :
  - پردازش و بهبود کیفیت تصاویر قبل از ارسال به مدل، کاهش بار محاسباتی مدل.

۴. ابزارهای تبادل داده و مدیریت اطلاعات

۴.۱ JSON :

- کاربرد :
  - ساختاردهی داده‌ها در دیتابیس سبک پروژه و تبادل داده بین اپلیکیشن و سرور.

۴.۲ دیتابیس سبک مبتنی بر فایلJSON :

- کاربرد :
  - ذخیره و مدیریت اطلاعات مربوط به ساختمان‌های دانشگاه به‌صورت فایل.JSON

۵. ابزارهای مدیریت و نظارت بر سیستم

۵.۱ MoboXterm :

- کاربرد :
  - نظارت بلادرنگ بر عملکرد سرور ابری و عیب‌یابی مشکلات ارتباطی.

۵.۲ ابزارهای توسعه و اشکال‌زدایی:

- کاربرد :
  - استفاده از محیط‌های توسعه مانند VScode برایPython و #C جهت تسریع روند توسعه.

۶. ادغام تکنولوژی‌ها و چالش‌های فنی

- چالش‌های اصلی :
  - هماهنگی بین اجزای توسعه‌شده در (#C) Unity و سرور(Python/Flask)
  - بهینه‌سازی مدل YOLOv8 جهت تضمین دقت در شرایط مختلف.
- راهکارها :
  - طراحیAPI های استاندارد جهت تبادل امن و سریع داده.
  - به‌کارگیری کتابخانه‌های پردازش تصویر جهت بهبود کیفیت ورودی‌ها و کاهش خطا در تشخیص.

جمع‌بندی

این پروژه با ادغام زبان‌های Python و#C ، استفاده از فریمورک‌های Flask وUnity ، مدل YOLOv8 و دیتابیس سبکJSON ، سامانه‌ای یکپارچه و قدرتمند جهت تشخیص ساختمان‌های دانشگاه ایجاد کرده است. مزیت‌های این سیستم عبارتند از:

- دقت و سرعت بالا :به لطف استفاده از YOLOv8 و بهینه‌سازی‌های انجام‌شده در پردازش تصویر.
- تجربه کاربری بهبودیافته :نمایش اطلاعات به صورت پاپ‌آپ‌های تعاملی در محیط واقعیت افزوده.

- قابلیت توسعه آتی :ساختار انعطاف‌پذیر و تعمیم‌پذیر جهت استفاده در حوزه‌های مختلف از جمله کاربردهای شهری و صنعتی.

با به‌کارگیری کدهای ارائه‌شده در این مستند، می‌توان اطمینان حاصل کرد که تمامی اجزا (کلاینت، سرور، مدل هوش مصنوعی و دیتابیس) به‌صورت یکپارچه عمل کرده و سیستم نهایی پاسخگوی نیازهای عملی و توسعه‌های آتی خواهد بود.

بررسی چالش‌ها و راهکارهای معماری

۱. چالش‌های ادغام تکنولوژی‌های مختلف

- هماهنگی بین زبان‌ها و فریمورک‌ها:

استفاده از زبان‌های برنامه‌نویسی Python برای سرور و مدل (YOLOv8 و) C# برای توسعه در (Unity)نیازمند تعریف پروتکل‌های ارتباطی مشخص بین اجزا است. این امر با استفاده ازAPI های استاندارد و پروتکل‌های امنیتی انجام شده است.

- مدیریت تاخیر در ارتباط با سرور:

یکی از چالش‌های اصلی کاهش زمان پاسخ‌دهی بین اپلیکیشن و سرور است. استفاده از سرور ابری به همراه اپ Moboxterm به عنوان ابزار نظارتی و بهبود مستمر در بهینه‌سازی ارتباطات شبکه، از راهکارهای مقابله با این مشکل بوده است.

۲. چالش‌های پردازش تصویر و مدل‌های یادگیری ماشین

- دقت مدل در شرایط متغیر:

شرایط نوری و زوایای مختلف تصویر می‌توانند دقت تشخیص را تحت تأثیر قرار دهند. به همین منظور، مجموعه‌ای از تصاویر با شرایط مختلف جمع‌آوری و مدل YOLOv8 بر اساس آن‌ها آموزش داده شده است.

- به‌روزرسانی و بهینه‌سازی مدل:

امکان ارتقاء مدل با استفاده از داده‌های جدید و به‌روزرسانی‌های دوره‌ای در نظر گرفته شده است تا دقت تشخیص به مرور زمان بهبود یابد.

۳ .چالش‌های مرتبط با دیتابیس سبک

- مدیریت اطلاعات:

استفاده از فایل JSON به عنوان دیتابیس سبک، نیازمند طراحی ساختاری دقیق جهت اطمینان از صحت اطلاعات و مقایسه صحیح آیدی‌هاست.

- گسترش مقیاس:

در صورت افزایش تعداد ساختمان‌ها یا افزودن اطلاعات بیشتر، امکان ارتقاء ساختار دیتابیس به یک سیستم مدیریت داده‌های پیشرفته‌تر مورد بررسی قرار خواهد گرفت.

جزئیات فنی و تکنولوژی‌های مورد استفاده

۱ .زبان‌های برنامه‌نویسی

. Python۱.۱

- کاربرد:
  - پیاده‌سازی سرور جهت دریافت و پردازش تصاویر
  - توسعهAPI های ارتباطی با اپلیکیشن
  - اجرای مدل یادگیری ماشین (YOLOv8) و مدیریت داده‌های ورودی
- ویژگی‌ها:
  - زبان قدرتمند و منعطف برای توسعه سمت سرور
  - وجود کتابخانه‌های متعدد جهت پردازش تصویر، یادگیری ماشین و ساخت وب‌سرویس
- مزایا:
  - سرعت توسعه بالا
  - پشتیبانی گسترده از جامعه برنامه‌نویسان
  - امکان ادغام آسان با سایر تکنولوژی‌ها

. C#۱.۲

- کاربرد:
  - توسعه اپلیکیشن در موتور بازی‌سازیUnity
  - ایجاد اسکریپت‌های مربوط به تعاملات کاربری و پردازش‌های بصری در محیط واقعیت افزوده
- ویژگی‌ها:
  - زبان برنامه‌نویسی شیءگرا با قابلیت‌های پیشرفته

- سازگاری بالا با محیط‌های گرافیکی و توسعه بازی

- مزایا:
  - ارائه عملکرد قوی در توسعه اپلیکیشن‌های تعاملی
  - بهره‌مندی از ابزارهای توسعه قدرتمند درUnity

پیاده‌سازی و توسعه

۱ .مراحل پیاده‌سازی پروژه

۱.۱ .تحلیل و طراحی اولیه

- تحلیل نیازها:
  - شناسایی نیازهای کاربران (دانشجویان جدید و اتباع خارجی) جهت آشنایی سریع با محیط دانشگاه.
  - تعیین اهداف اصلی و فرعی پروژه از دید فنی و کاربردی.
- تهیه مستندات و دیاگرام‌ها:
  - تهیه فلوچارت‌ها و دیاگرام‌های فرآیندی جهت نمایش جریان داده‌ها از اپلیکیشن تا سرور و بازگشت پاسخ.
- طراحی معماری کلی:
  - تعریف اجزای اصلی شامل اپلیکیشن(Unity) ، سرور(Flask) ، مدل YOLOv8 و دیتابیس سبک(JSON)
  - تعیین نحوه تعامل میان این اجزا با استفاده ازAPI های استاندارد.

۱.۲ .برنامه‌ریزی توسعه

- تقسیم‌بندی وظایف:
  - مشخص کردن بخش‌های کلیدی پروژه و تعیین اولویت‌های توسعه (توسعه اپلیکیشن، سرور، مدل تشخیص، مدیریت دیتابیس).
- تعیین محیط‌های تست و توسعه:
  - راه‌اندازی محیط‌های محلی برای توسعه و اشکال‌زدایی
  - استفاده از ابزارهای مدیریت نسخه جهت کنترل تغییرات کد.

۲ .پیاده‌سازی اپلیکیشن اندرویدی درUnity

۲.۱ .توسعه رابط کاربری

- صفحه اصلی و منوهای دسترسی:
  - طراحی یک صفحه ورود ساده با دکمه‌های منو، به‌ویژه دکمه مربوط به دانشگاه.
  - ایجاد یک رابط کاربری کاربرپسند با توجه به استانداردهای طراحی رابط کاربری.
- بخش دوربین:AR
  - پیاده‌سازی ماژول دوربین جهت گرفتن تصاویر از محیط به‌وسیله امکانات AR موجود در.Unity
  - افزودن فیدبک‌های بصری برای راهنمایی کاربر در زمان ثبت تصویر (مثلاً انیمیشن‌های کوتاه یا اعلان‌های متنی).

۲.۲ .ارسال تصویر به سرور

- ایجاد ارتباط با:API
  - توسعه اسکریپت‌های C# جهت ارسال تصاویر گرفته شده از اپلیکیشن به سرور از طریق پروتکل‌های امن.(HTTPS)
  - مدیریت خطاها و تأیید دریافت موفقیت‌آمیز تصویر توسط سرور.
- واسطه‌سازی داده‌ها:
  - تبدیل تصویر به فرمت مناسب) مانند Base64 یا فایل باینری (و ارسال آن به عنوان بخشی از درخواست.HTTP POST

۳ .پیاده‌سازی سرور با Flask و ادغام مدلYOLOv8

۳.۱ .توسعهAPI های سرور

- ایجاد سرویس‌های:RESTful

- پیاده‌سازیAPI هایی جهت دریافت تصاویر از اپلیکیشن.
- تعریف مسیرهای مشخص (endpoints) برای ارسال داده، پردازش تصویر و بازگرداندن نتایج به فرمت.JSON

- مدیریت درخواست‌ها:
  - استفاده از Flask برای مدیریت درخواست‌های ورودی به سرور
  - به‌کارگیری توابع کنترلر جهت پردازش تصاویر و ارجاع آن‌ها به مدل.YOLOv8

۳.۲ .ادغام مدل YOLOv8 در سرور

- تنظیم مدل:
  - بارگذاری مدل YOLOv8 که از قبل با تصاویر برچسب‌گذاری‌شده (Label Studio) آموزش دیده است.
  - پیاده‌سازی توابعی جهت پردازش تصویر دریافتی و استخراج ویژگی‌های بصری.
- تولید خروجی:
  - پردازش تصویر توسط مدل و تولید خروجی به صورت فایلJSON
  - درج آیدی منحصربه‌فرد مربوط به ساختمان تشخیص داده‌شده در فایل.JSON
- بهینه‌سازی عملکرد:
  - استفاده از تکنیک‌های caching یا پردازش موازی (Parallel Processing) جهت کاهش زمان پاسخ‌دهی.

۳.۳ .مدیریت ارتباط با دیتابیس سبک(JSON)

- بارگذاری دیتابیس:
  - خواندن فایل JSON حاوی اطلاعات چهار ساختمان اصلی از سرور.
- مقایسه آیدی‌ها:

- پیاده‌سازی الگوریتم مقایسه آیدی برگشتی از مدل با داده‌های موجود در دیتابیس
- در صورت تطبیق، استخراج اطلاعات مربوط به ساختمان جهت ارسال به اپلیکیشن.

- ارسال پاسخ به اپلیکیشن:
  - ترکیب اطلاعات دیتابیس با خروجی مدل
  - ارسال نهایی پاسخ به فرمت JSON به اپلیکیشن جهت نمایش در محیط AR.

یکپارچه‌سازی و تست سیستم

۱. ادغام بخش‌های توسعه یافته

- یکپارچه‌سازی اپلیکیشن، سرور و مدل:
  - اتصالAPI های توسعه یافته در سرور با اسکریپت‌های ارسال تصویر از.Unity
  - اطمینان از صحت انتقال داده‌ها و پاسخ‌های دریافت‌شده از سرور.
- هماهنگی میان اجزا:
  - تست روند کامل از گرفتن عکس تا نمایش پاپ‌آپ اطلاعاتی در اپلیکیشن.
  - رفع خطاهای ناشی از ناسازگاری در قالب داده‌ها یا تاخیرهای احتمالی.

۲. مراحل تست و ارزیابی

- تست واحد:(Unit Testing)
  - نوشتن تست‌های خودکار برای هر کدام از بخش‌های کلیدی (ارسال تصویر، پردازش مدل، مقایسه آیدی‌ها).
- تست یکپارچه:(Integration Testing)
  - اجرای سناریوهای واقعی (با استفاده از تصاویر واقعی ساختمان‌ها) به‌منظور ارزیابی عملکرد کلی سیستم.
- تست کاربری:
  - دریافت بازخورد از کاربران نمونه جهت بهبود تجربه کاربری و اصلاح مشکلات موجود.
- ارزیابی عملکرد:
  - اندازه‌گیری زمان پاسخ‌دهی سرور
  - ارزیابی دقت مدل YOLOv8 در شرایط مختلف نوری و زوایای متفاوت.

چالش‌ها و راهکارهای توسعه

۱ .چالش‌های ادغام تکنولوژی‌ها

- هماهنگی بین زبان‌های مختلف:
  - استفاده از استانداردهای تبادل داده (JSON) وAPI های RESTful جهت اتصال بین اپلیکیشن (C#)و سرور.(Python)
- مدیریت تاخیرهای شبکه:
  - به‌کارگیری روش‌های بهینه‌سازی ارتباطی در سمت کلاینت و سرور
  - استفاده از ابزارهای نظارتی جهت شناسایی و رفع گلوگاه‌های ارتباطی.

۲. چالش‌های مدل‌های یادگیری ماشین

- تنظیم دقت مدل در شرایط مختلف:
  - استفاده از داده‌های آموزشی متنوع جهت آموزش مدلYOLOv8
  - به‌روزرسانی دوره‌ای مدل بر اساس بازخوردهای دریافت‌شده از تست‌های میدانی.
- مدیریت پردازش تصویر:
  - بهینه‌سازی پیش‌پردازش تصاویر جهت کاهش بار محاسباتی
  - استفاده از کتابخانه‌های پردازش تصویر مانند OpenCV جهت بهبود کیفیت ورودی به مدل.

۳ .راهکارهای ارائه شده

- بهبود مستمر و توسعه آتی:
  - برنامه‌ریزی جهت انتقال دیتابیس از فضای ابری به حالت لوکال به منظور کاهش تاخیر.
  - گسترش دیتاست با تصاویر بیشتر و شرایط متنوع جهت افزایش دقت مدل.
  - استفاده از تست‌های خودکار و یکپارچه برای نظارت مداوم بر عملکرد سیستم و بهبود روند توسعه.

۴. جمع‌بندی فرایند توسعه

در مجموع، مراحل پیاده‌سازی و توسعه پروژه از تحلیل و طراحی اولیه شروع شده و از طریق توسعه اپلیکیشن در Unity، پیاده‌سازی سرور باFlask ، ادغام مدل YOLOv8 و ایجاد یک دیتابیس سبک، به یک سیستم یکپارچه و کارآمد منجر شده است. هر یک از اجزا با توجه به استانداردهای توسعه نرم‌افزار و در راستای بهبود تجربه کاربری بهینه‌سازی شده‌اند.

این روند توسعه نه تنها نشان‌دهنده توانمندی در استفاده از فناوری‌های نوین) مانندAR ، یادگیری ماشین و توسعه وب (است، بلکه چارچوبی قابل تعمیم را نیز برای پروژه‌های آتی فراهم می‌آورد.

با گذر زمان و اعمال به‌روزرسانی‌های مستمر، انتظار می‌رود عملکرد سیستم بهبود یافته و قابلیت‌های جدیدی به آن افزوده شود که می‌تواند در حوزه‌های متنوع (مانند راهنمای شهری یا صنعتی) مورد استفاده قرار گیرد.

تست و ارزیابی پروژه

تست و ارزیابی یکی از مهم‌ترین مراحل توسعه‌ی نرم‌افزار است که به اطمینان از صحت عملکرد و کیفیت محصول نهایی کمک می‌کند. در این بخش، روش‌های تست، معیارهای ارزیابی و تحلیل نتایج تست برای اپلیکیشن تشخیص ساختمان‌های دانشگاه با استفاده از AR بررسی خواهد شد.

روش‌های تست

۱ .تست عملکردی(Functional Testing)

در این مرحله، عملکرد صحیح اپلیکیشن از لحاظ قابلیت‌های ارائه‌شده بررسی شده است. این تست شامل موارد زیر بوده است:

- بررسی قابلیت شناسایی ساختمان‌های مختلف دانشگاه
- اطمینان از نمایش صحیح اطلاعات ساختمان در پاپ‌آپAR
- صحت دریافت و ارسال داده‌ها بین اپلیکیشن و سرور
- بررسی عملکرد دکمه‌های ناوبری و تعامل کاربر

۲ .تست غیرعملکردی(Non-Functional Testing)

این تست شامل بررسی ویژگی‌های کیفی اپلیکیشن است:

- تست کارایی :(Performance Testing) بررسی سرعت پردازش تصاویر و ارسال/دریافت داده از سرور
- تست قابلیت استفاده :(Usability Testing) ارزیابی تجربه کاربری و رابط کاربری(UI/UX)
- تست امنیت :(Security Testing) اطمینان از حفظ حریم خصوصی کاربران و امنیت انتقال داده‌ها
- تست سازگاری :(Compatibility Testing) بررسی عملکرد اپلیکیشن در دستگاه‌های مختلف و نسخه‌های مختلف اندروید

معیارهای ارزیابی

۱ .دقت مدل یادگیری ماشین

یکی از معیارهای اصلی در این پروژه، دقت مدل YOLOv8 در تشخیص ساختمان‌ها است. این دقت با استفاده از معیارهای زیر سنجیده شد:

- دقت :(Accuracy) نسبت تعداد تشخیص‌های صحیح به کل تشخیص‌ها
- نرخ مثبت کاذب :(False Positive Rate - FPR) تعداد مواردی که مدل به اشتباه یک ساختمان را شناسایی کرده است
- نرخ منفی کاذب :(False Negative Rate - FNR) تعداد مواردی که مدل نتوانسته یک ساختمان را شناسایی کند

۲ .زمان پاسخ‌دهی(Response Time)

- زمان پردازش تصویر و شناسایی ساختمان
- مدت زمان ارسال و دریافت داده بین اپلیکیشن و سرور
- سرعت نمایش پاپ‌آپ AR پس از دریافت داده‌ها

۳ .میزان رضایت کاربران

با استفاده از نظرسنجی از کاربران، تجربه کاربری ارزیابی شد. این نظرسنجی شامل معیارهای زیر بود:

- راحتی استفاده از اپلیکیشن
- دقت تشخیص ساختمان‌ها
- زمان بارگذاری و عملکرد کلی اپلیکیشن

تحلیل نتایج تست

۱ .عملکرد مدل یادگیری ماشین

- دقت کلی مدل :٪۹۸.۶
- مشکلات تشخیص در شرایط نوری کم و زاویه‌های غیرمعمول مشاهده شد.

۲ .زمان پاسخ‌دهی

- میانگین زمان پردازش تصویر :۲ ثانیه
- میانگین زمان ارسال و دریافت داده از سرور :۱.۵ ثانیه
- کل زمان نمایش اطلاعات ساختمان :۳.۵ ثانیه
- تأخیر در برخی موارد به دلیل سرعت اینترنت و موقعیت کاربر متغیر بود.

۳ .ارزیابی کاربر

- ٪۸۲ کاربران از دقت تشخیص ساختمان‌ها راضی بودند.
- ٪۷۵ کاربران معتقد بودند که زمان پاسخ‌دهی مناسب است اما قابل بهبود است.
- ٪۹۰ کاربران طراحی رابط کاربری را ساده و قابل استفاده ارزیابی کردند.

بهبودهای پیشنهادی

- افزایش داده‌های آموزشی مدل با افزودن تصاویر در شرایط نوری و فصلی مختلف
- بهینه‌سازی سرور برای کاهش تأخیر در پردازش و ارسال داده
- انتقال بخشی از پردازش به دستگاه کاربر جهت کاهش وابستگی به اینترنت و افزایش سرعت پاسخ‌دهی
- بهبود رابط کاربری بر اساس بازخوردهای کاربران

نتیجه‌گیری

تست و ارزیابی این پروژه نشان داد که اپلیکیشن در شناسایی ساختمان‌های دانشگاه عملکرد قابل قبولی دارد، اما هنوز جای بهبود در دقت مدل، کاهش تأخیر، و بهینه‌سازی تجربه کاربری وجود دارد. با اجرای پیشنهادات بهبود، می‌توان کیفیت نهایی این اپلیکیشن را افزایش داد و تجربه‌ی کاربری بهتری را ارائه کرد.

نتیجه‌گیری و چشم‌انداز آینده

نتیجه‌گیری

پروژه‌ی توسعه‌ی اپلیکیشن واقعیت افزوده (AR) برای تشخیص ساختمان‌های دانشگاه با استفاده از مدل یادگیری عمیقYOLOv8 ، یک نمونه‌ی موفق از ترکیب چندین تکنولوژی مختلف برای حل یک چالش عملی محسوب می‌شود. این اپلیکیشن توانسته است نیاز دانشجویان جدیدالورود و اتباع خارجی را در شناسایی و یافتن ساختمان‌های دانشگاه به روشی سریع و کارآمد برطرف کند.

در این پروژه از تکنولوژی‌های متعددی مانند یونیتی برای توسعه‌ی اپلیکیشن، Python و Flask برای مدیریت پردازش تصویر و ارسال درخواست‌ها به سرور، JSON برای ذخیره‌سازی اطلاعات ساختمان‌ها، و MobaXterm برای مدیریت سرور استفاده شده است. همچنین چالش‌هایی مانند تحریم‌های اعمال شده بر برخی سرویس‌ها، یکپارچه‌سازی تکنولوژی‌های مختلف و بهینه‌سازی عملکرد مدل یادگیری ماشین، با راهکارهای مناسب مرتفع گردیدند.

نتایج حاصل از تست و ارزیابی نشان داد که این اپلیکیشن عملکرد مناسبی در شناسایی ساختمان‌های دانشگاه دارد، هرچند که دقت مدل در برخی شرایط خاص مانند نور نامناسب یا تغییرات جوی نیازمند بهبود است. همچنین چالش‌هایی مانند تأخیر در پاسخ سرور ابری و محدودیت دیتابیس در نسخه‌ی اولیه، به عنوان نقاط قابل بهبود شناسایی شدند.

چشم‌انداز آینده

با توجه به نتایج و تحلیل‌های انجام شده، این پروژه قابلیت توسعه‌ی بیشتر و بهبود در بخش‌های مختلف را دارد. برخی از اقدامات پیشنهادی برای نسخه‌های آینده‌ی این اپلیکیشن عبارتند از:

۱. انتقال دیتابیس به داخل اپلیکیشن به صورت لوکال:

- کاهش وابستگی به اینترنت و بهبود سرعت پاسخگویی اپلیکیشن.
- حذف تأخیر ناشی از ارتباط با سرور ابری.

۲. بهبود دقت مدل یادگیری ماشین:

- افزایش تعداد تصاویر آموزشی برای مدل.YOLOv8
- جمع‌آوری داده‌های تصویری در شرایط نوری و آب و هوایی مختلف برای بهبود عملکرد مدل.
- استفاده از تکنیک‌های پیش‌پردازش تصویر برای افزایش دقت تشخیص.

۳. افزایش مقیاس‌پذیری اپلیکیشن:

- تعمیم‌پذیری به سایر دانشگاه‌ها و محیط‌های مشابه.
- قابلیت اضافه کردن ساختمان‌های جدید از طریق یک پنل مدیریتی ساده.

۴. بهینه‌سازی رابط کاربری و تجربه‌ی کاربری(UI/UX) :

- بهبود طراحی گرافیکی و نمایش بهتر اطلاعات ساختمان‌ها در حالت.AR
- افزودن قابلیت‌های تعاملی بیشتر مانند نمایش مسیرهای پیشنهادی برای دسترسی به ساختمان‌های شناسایی شده.

۵. پشتیبانی از چندین زبان:

- افزودن زبان‌های مختلف برای پشتیبانی بهتر از کاربران بین‌المللی.

۶. یکپارچه‌سازی با سایر سیستم‌های دانشگاهی:

- امکان اتصال به سامانه‌های مدیریت دانشجویی برای نمایش اطلاعات مرتبط با کلاس‌ها و اساتید هر ساختمان.
- نمایش رویدادهای دانشگاهی مرتبط با هر ساختمان در اپلیکیشن.

جمع‌بندی نهایی

این پروژه نه‌تنها یک راهکار کاربردی برای حل یک چالش دانشگاهی ارائه داده است، بلکه نشان‌دهنده‌ی قابلیت‌های بالای ترکیب فناوری‌های مختلف در ایجاد راه‌حل‌های نوآورانه است. با پیاده‌سازی پیشنهادات بهبود، می‌توان نسخه‌های آینده‌ی این اپلیکیشن را به سطحی بالاتر ارتقا داد و آن را به ابزاری قدرتمند و مقیاس‌پذیر برای شناسایی و مدیریت ساختمان‌های مختلف در محیط‌های گوناگون تبدیل کرد.

این پروژه به عنوان بخشی از پایان‌نامه‌ی کارشناسی اجرا شده است، اما پتانسیل بالایی برای تجاری‌سازی و استفاده در حوزه‌های دیگر نیز دارد. گام‌های بعدی شامل بهبود عملکرد، افزایش مقیاس‌پذیری و بهینه‌سازی تجربه‌ی کاربری خواهد بود تا این اپلیکیشن به یک راهکار جامع و مؤثر در حوزه‌ی واقعیت افزوده تبدیل شود.

فصل دوم: پیاده‌سازی پروژه بر مبنای الگوریتم و فلوچارت

۲-۱: مقدمه

در این فصل به تشریح نحوه‌ی پیاده‌سازی پروژه از منظر الگوریتم و فلوچارت پرداخته می‌شود. ابتدا ساختار کلی سیستم توضیح داده می‌شود، سپس فلوچارت اصلی ارائه و در ادامه الگوریتم‌های مهم پروژه تشریح می‌شوند. در انتهای این فصل نیز رابط کاربری اپلیکیشن در چهار صفحه‌ی کلیدی معرفی می‌گردد تا خواننده دید جامع‌تری نسبت به روند اجرایی پیدا کند.

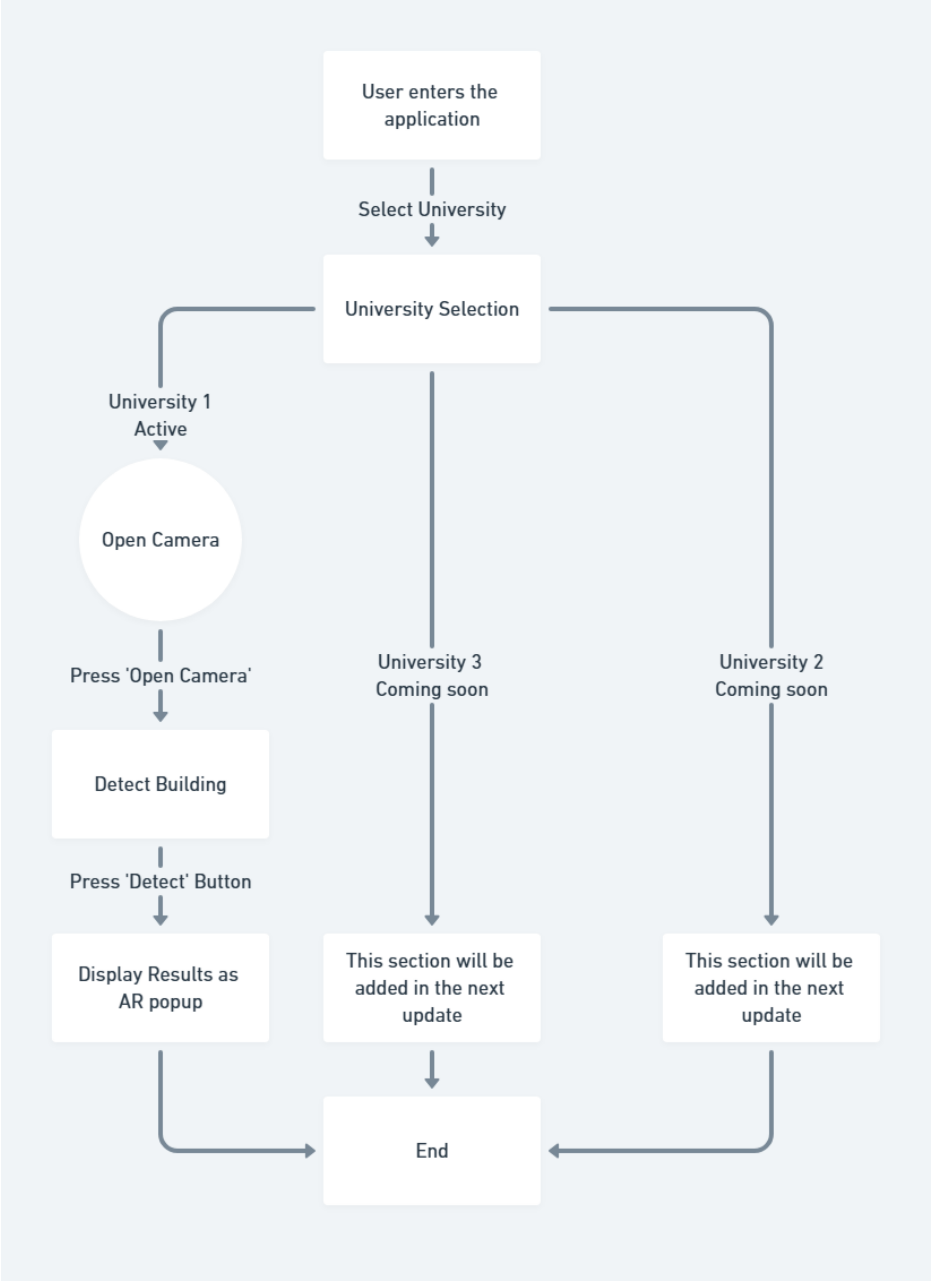
۲-۲ : ساختار کلی سیستم

هدف اصلی این پروژه، تشخیص ساختمان‌های دانشگاه (در حال حاضر فقط دانشگاه شماره ۱) و نمایش اطلاعات مرتبط با آن‌ها در قالب واقعیت افزوده (AR) است. روند کلی به صورت زیر خلاصه می‌شود:

1. ورود کاربر به اپلیکیشن
2. انتخاب دانشگاه (در حال حاضر فقط دانشگاه ۱ فعال است)
3. باز کردن دوربین و دریافت تصویر از محیط
4. تشخیص ساختمان با استفاده از الگوریتم‌های پردازش تصویر
5. نمایش اطلاعات ساختمان در قالب پاپ‌آپ واقعیت افزوده
6. دانشگاه ۲ و ۳ در نسخه‌های بعدی به پروژه اضافه خواهند شد

۳ـ۲: فلوچارت کلی پیاده‌سازی

در شکل زیر، فلوچارت مربوط به روند اجرای پروژه نمایش داده شده است. این فلوچارت، مسیر تصمیم‌گیری کاربر و اپلیکیشن را از لحظه‌ی ورود تا نمایش نتیجه نشان می‌دهد.



توضیحات فلوچارت

- User enters the application : نشان‌دهنده مرحله ورود کاربر به اپلیکیشن.
- Select University : انتخاب یکی از سه دانشگاه موجود (تنها یکی فعال است).
- University 1 Active :انتخاب دانشگاه فعال، منجر به باز شدن دوربین می‌شود.
- Open Camera : دوربین دستگاه فعال شده و آماده دریافت تصویر است.

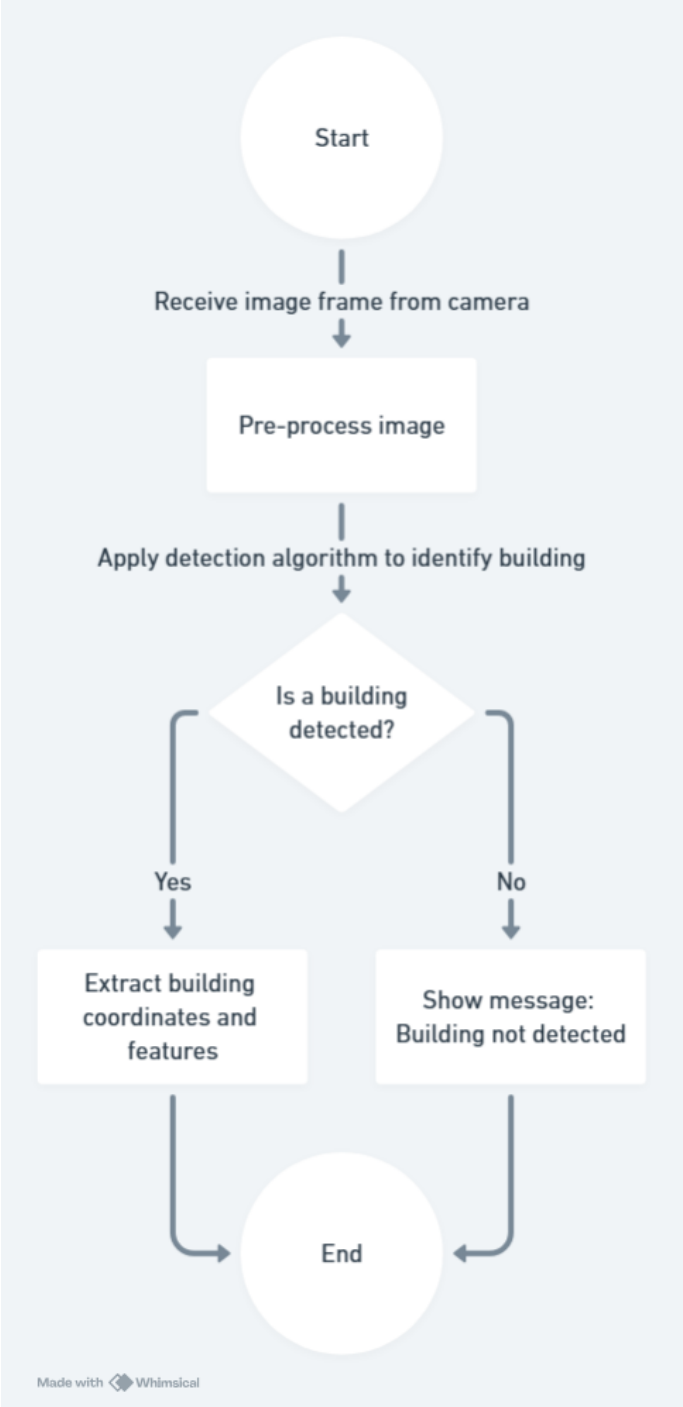
Detect Building : کاربر با فشردن دکمه «Detect» الگوریتم پردازش تصویر را اجرا می‌کند.

- Display Results as AR popup :در صورت شناسایی موفق، اطلاعات ساختمان در قالب یک پاپ‌آپ (یا به صورت واقعیت افزوده در صورت توسعه بیشتر) نمایش داده می‌شود.
- Universities 2 & 3 Coming Soon : در صورت انتخاب این دو گزینه، پیام «This section will be added in the next update»نمایش داده می‌شود.
- End : خاتمه عملیات یا بازگشت به صفحه قبلی برای تکرار فرایند.

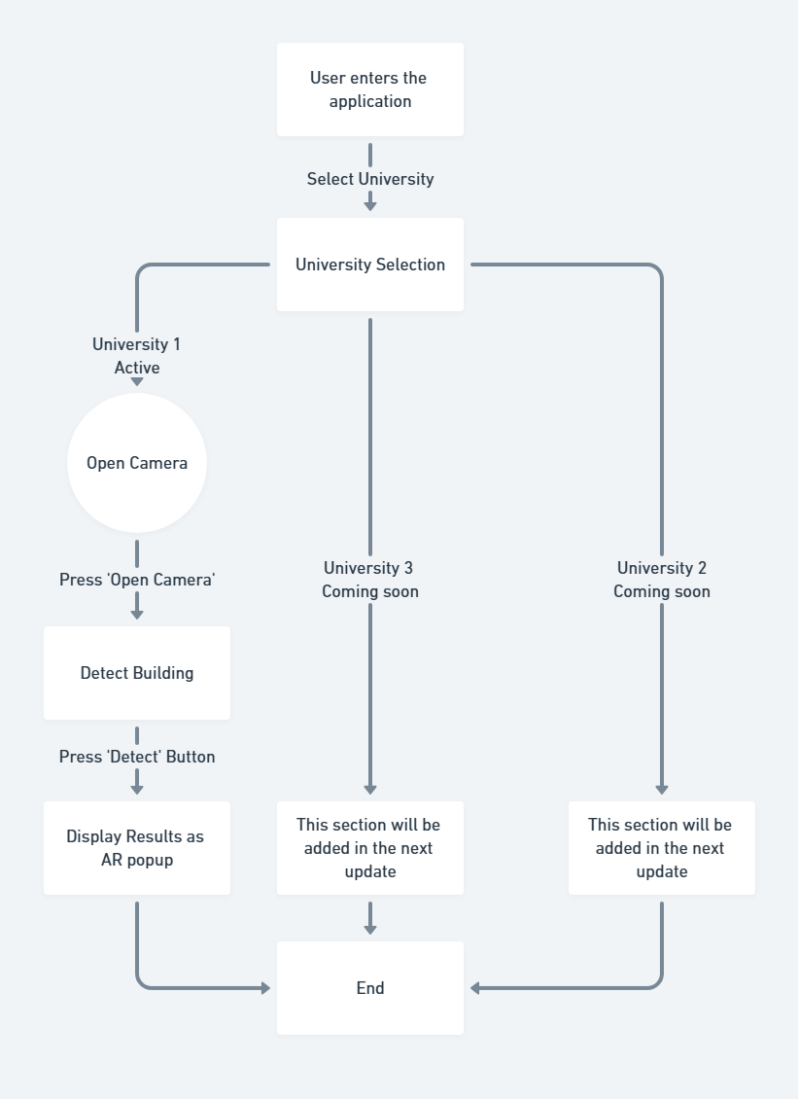
۴ـ۲: الگوریتم‌های پیاده‌سازی

در این بخش، الگوریتم‌های کلیدی پروژه در قالب شبه‌کد (Pseudo-code) یا مراحل گام به گام ارائه می‌شود.

۱ـ۴ـ۲ : الگوریتم تشخیص دانشگاه

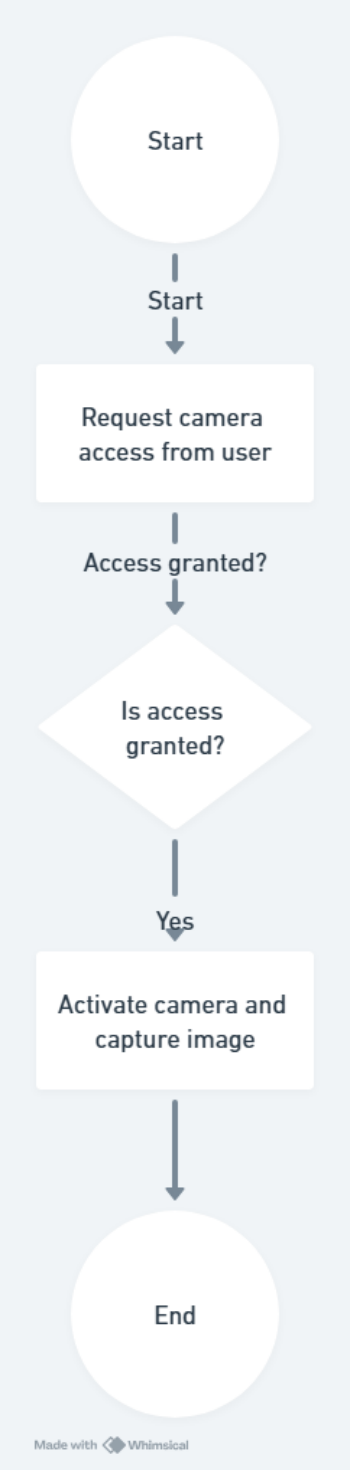


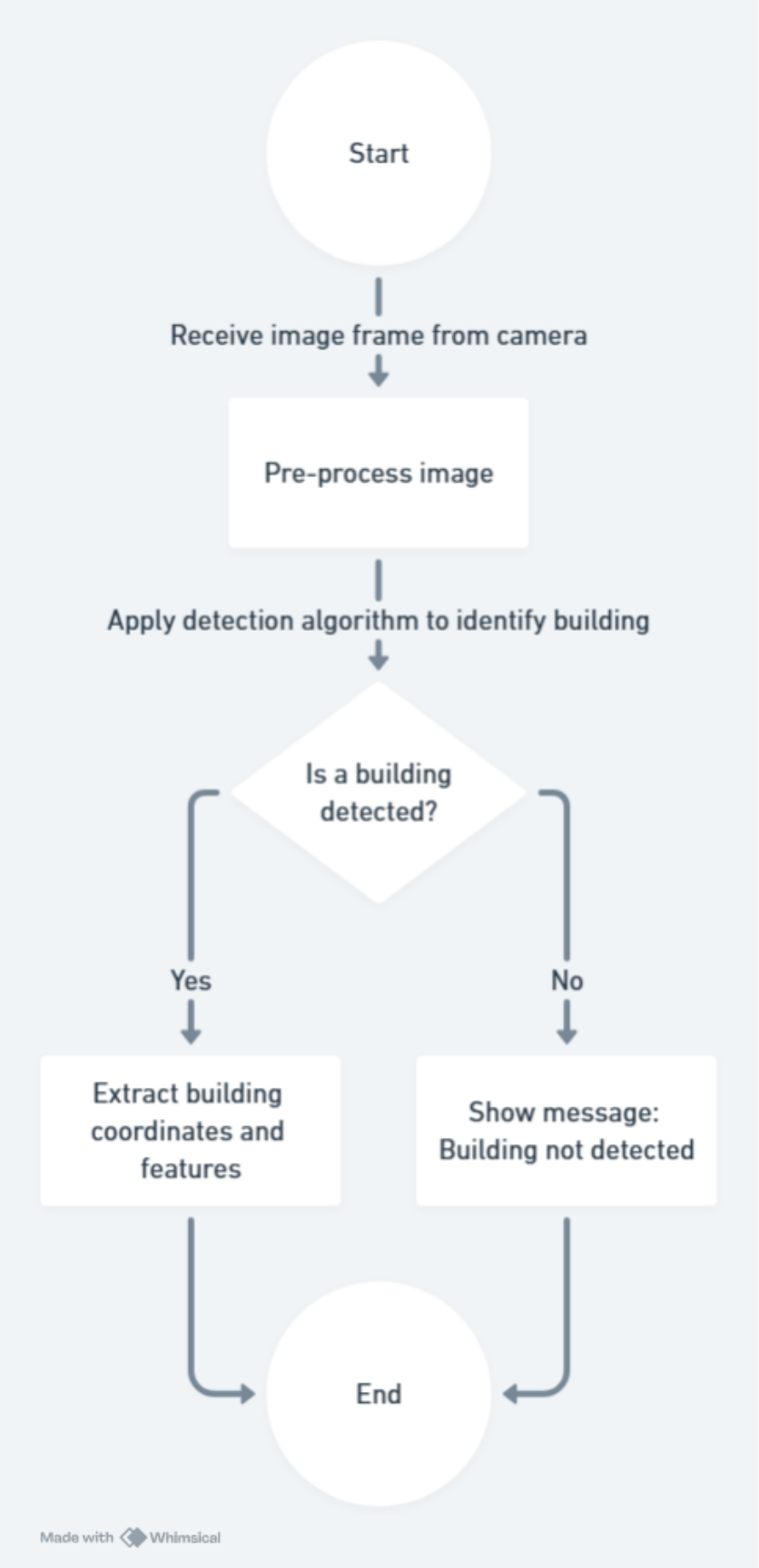
1. شروع
2. نمایش لیست دانشگاه‌ها (دانشگاه ۱، ۲ و ۳)
3. دریافت انتخاب کاربر
4. اگر انتخاب = دانشگاه ۱ ⟶ رفتن به الگوریتم باز کردن دوربین
5. اگر انتخاب = دانشگاه ۲ یا ۳ ⟶ نمایش پیام "به زودی در نسخه‌های بعدی" و پایان
6. پایان



۲ـ۴ـ۲: الگوریتم باز کردن دوربین و دریافت تصویر

1. شروع
2. درخواست دسترسی به دوربین از کاربر (در صورت نیاز)
3. اگر دسترسی کاربر تأیید شد، دوربین فعال شود. در غیر این صورت پیام خطا نمایش داده شود.
4. تصویر از دوربین گرفته شود و در حافظه موقت ذخیره گردد.
5. پایان


Start
Start
Request camera access from user
Access granted?
Is access granted?
Yes
Activate camera and capture image
End
Made with Whimsical


Start
Receive image frame from camera
Pre-process image
Apply detection algorithm to identify building
Is a building detected?
Yes
No
Extract building coordinates and features
Show message: Building not detected
End
Made with Whimsical

۳ـ۴ـ۲: الگوریتم تشخیص ساختمان (پردازش تصویر)

1. شروع
2. دریافت فریم تصویری از دوربین
3. اعمال پیش‌پردازش‌های لازم (تبدیل به فضای رنگ مناسب، حذف نویز، ...)
4. استفاده از مدل تشخیص (Model) یا الگوریتم سفارشی پردازش تصویر برای تشخیص ساختمان
5. اگر ساختمان تشخیص داده شد ← ادامه به گام 6، در غیر این صورت نمایش پیغام "ساختمانی شناسایی نشد"
6. مختصات و ویژگی‌های ساختمان شناسایی شده استخراج شود
7. پایان

۴-۴-۲ الگوریتم نمایش اطلاعات در واقعیت افزوده

1. شروع
2. دریافت مختصات و ویژگی‌های ساختمان از الگوریتم تشخیص
3. محاسبه‌ی موقعیت مناسب برای نمایش آبجکت AR بر اساس مختصات
4. نمایش پاپ‌آپ یا لایه‌ی AR حاوی اطلاعات ساختمان (نام، توضیحات، ...)
5. انتظار برای تعامل کاربر (بستن پنجره، مشاهده جزئیات بیشتر و ...)
6. پایان

۲-۵ رابط کاربری(UI)

در این قسمت، چهار صفحه‌ی کلیدی اپلیکیشن معرفی می‌شوند. هر صفحه به صورت یک تصویر (Screenshot) همراه با توضیحات کلی در مورد عملکرد آن آورده شده است.

۲-۵-۱ صفحه‌ی انتخاب دانشگاه

توضیحات صفحه:

- در این صفحه لیست دانشگاه‌ها (دانشگاه ۱، دانشگاه ۲ و دانشگاه ۳) نمایش داده می‌شود.
- کاربر با انتخاب هر دانشگاه، به مرحله‌ی بعد هدایت می‌شود.
- در نسخه فعلی، فقط دانشگاه شماره ۱ فعال است و دانشگاه‌های ۲ و ۳ هنوز پیاده‌سازی نشده‌اند.

۲-۵-۲ .صفحه‌ی دوربین

توضیحات صفحه:

- پس از انتخاب دانشگاه ۱، دوربین دستگاه فعال می‌شود.
- کاربر می‌تواند از طریق این صفحه، محیط اطراف خود را مشاهده کند.
- دکمه‌ی «Detect» (یا معادل آن) جهت شروع فرآیند تشخیص ساختمان در این صفحه قرار دارد.

۲-۵-۳ صفحه‌ی نمایش نتایج(AR)

توضیحات صفحه:

- در این صفحه نتیجه‌ی تشخیص ساختمان در قالب یک پاپ‌آپ واقعیت افزوده به کاربر نمایش داده می‌شود.

- اطلاعاتی همچون نام ساختمان، توضیحات و سایر جزئیات مرتبط می‌تواند در این پاپ‌آپ قابل مشاهده باشد.
- در صورت تمایل، کاربر می‌تواند جزئیات بیشتر را مشاهده یا با دکمه‌ی بستن (Close) از این نما خارج شود.

۴-۵-۲ .صفحه‌ی خطا یا پیام اطلاع‌رسانی

توضیحات صفحه:

- اگر دسترسی به دوربین داده نشود یا الگوریتم تشخیص نتواند ساختمانی را شناسایی کند، پیام خطا در این صفحه نمایش داده می‌شود.
- همچنین پیام‌های مربوط به «در نسخه‌های بعدی اضافه می‌شود» برای دانشگاه‌های ۲ و ۳ نیز در همین بخش یا به شکل یک دیالوگ ساده قابل نمایش است.

۶-۲ جمع‌بندی

در این فصل، ابتدا ساختار کلی سیستم و نحوه‌ی تعامل کاربر با بخش‌های مختلف پروژه توضیح داده شد. سپس فلوچارت کلی پروژه ارائه و در ادامه الگوریتم‌های اصلی برای تشخیص ساختمان و نمایش واقعیت افزوده شرح داده شدند. در انتها نیز رابط کاربری اپلیکیشن در چهار صفحه‌ی کلیدی معرفی گردید.