**گزارش مربوط به نسخه های پایدار و ویژگی هایی که در هر نسخه اضافه یا حذف شده است.**

**نسخه صفر:**

در این نسخه ui به صورت قابل استفاده برای تعداد نامحدود دوربین طراحی شده که در آن میتوان دوربین را در tab ، add camera در سرور تعریف کرد و در tab ، show camera به نمایش گزاشت همچنین میتوان با انخاب فیلتر مورد نظر در tab ، show camera پردازشی ساده را بر روی تصاویر انجام داد. همچنین بستری برای شرح اتفاقی که در پشت انجام خواهد شد در پایین صفحه به نمایش در خواهد آمد.

**عملکرد کلی:**

در این برنامه با فشردن کلید add camera دستگاه اطلاعات خود را بر روی یک exchange مشخص ریخته و یک صف می سازد و صف مربوطه را به آن exchange مشخص منتقل میکند. به همین جهت این نسخه تنها توانایی پردازش اطلاعات یک دوربین را دارد. همچنین با فشردن این دگمه دو qprocess بر روی رم تخصیص داده می­شوند که برنامه ارسال کننده اطلاعات و برنامه دریفات کننده اطلاعات در آن اجرا خواهند شد.

با فشردن کلید add camera بر اساس فیلتری که انتخاب شده، برنامه sender.py و receiver.py در qprocess های ایجاد شده در مرحله قبل اجرا خواهند شد. برنامه ارسال کننده اطلاعات به مشابه گزارش های قبلی نوشته شد. و برنامه دریافت کننده اطلاعات بسته یک ماتریس از یک رنگ ثابت با یک ضریب با تصویر جمع میکند تا تصویر کمی تغییر کند.

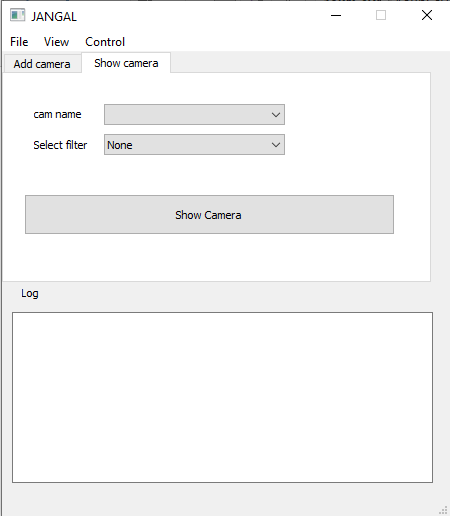
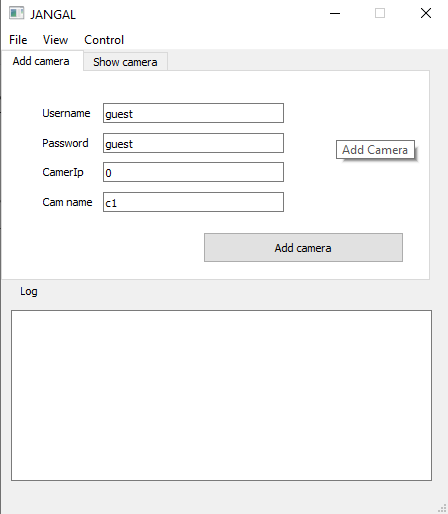
همچنین در برنامه gui که اطلاعات دوربین را به نمایش در می­آورد میتوان نوع فیلتر تعیین شده را تغییر داد.

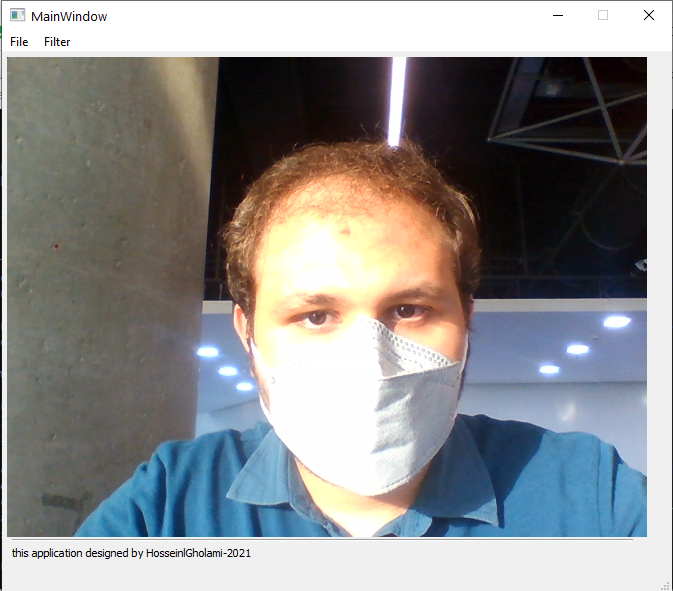
این دو برنامه توسط هندل هایی که برای آن ها توسط gui.py تعریف می­شوند اجرا خواهند شد.

**برای اجرای این برنامه:**

برای اجرای این برنامه ابتدا دستور docker-compose up را اجرا کرده و در ادامه برنامه پایتون gui.py را اجرا کنید.

**نمایی از این نسخه از برنامه:**





**نسخه یک:**

در این نسخه ui به صورت قابل استفاده برای تعداد نامحدود دوربین طراحی شده که در آن میتوان دوربین را در tab ، add camera ، ارزیابی و وضعیت سرور را بررسی و در نهایت دوربین ها را در سرور با اکسچنجی مشخص تعریف کرد و در tab ، sender controllerبرنامه ارسال اطلاعات به سرور برای دوربین هایی که در مرحله قبل تعریف شده اند را توسط کلید های start sender و stop sender کنترل کرد. و در tab، Reciever کنترلر، برنامه دریافت اطلاعات از دوربین را به اجرا درآرود تا اطلاعاتی که به سرور فرستاده شده اند را بتوان دریافت نمود.

**عملکرد کلی:**

در این برنامه با فشردن کلید add camera دستگاه وضعیت دوربین وارد شده را بررسی میکند و سپس احراز هویت برای اتصال به سرور را بررسی کرده میکند سپس در صورتی که موارد فوق مورد تایید باشد یک exchange با نام camera name بر روی سرور ایجاد میکند و در برنامه ارسال کننده و دریافت کننده اطلاعات این جزئیات را وارد میکند و نتیجه بررسی خود را در بخش پایین صفحه اعمال می­کند . همچنین با فشردن این کلید دو qprocess بر روی رم تخصیص داده می شوند تا برنامه ارسال کننده اطلاعات و برنامه دریافت کننده اطلاعات در آن اجرا شوند.

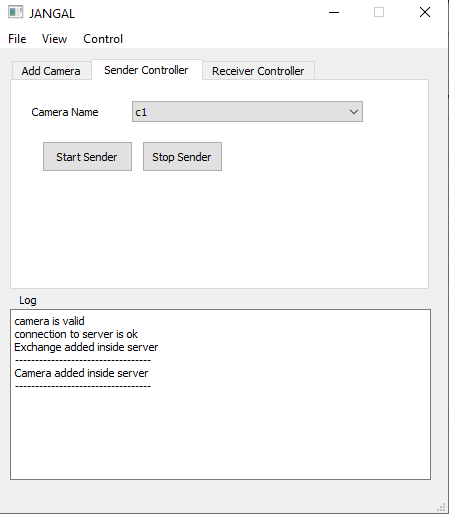
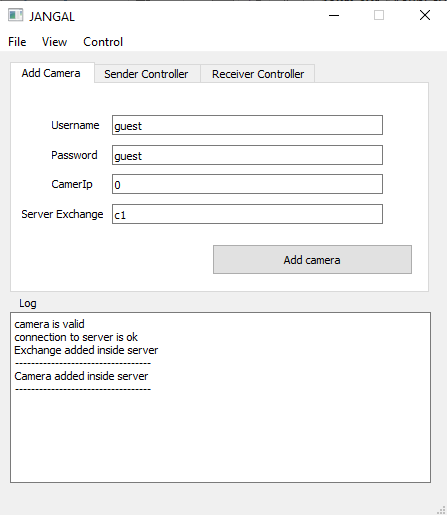
در tab، Sender controller ، ابتدا باید دوربین مد نظر را انتخاب و سپس با فشردن کلید start Sender برنامه ارسال اطلاعات دوربین در qprocess که در مرحله قبل ایجاد شد اجرا میشود. و با فشردن کلید stop sender ارسال اطلاعات دوربین به سرور متوقف خواهد شد.

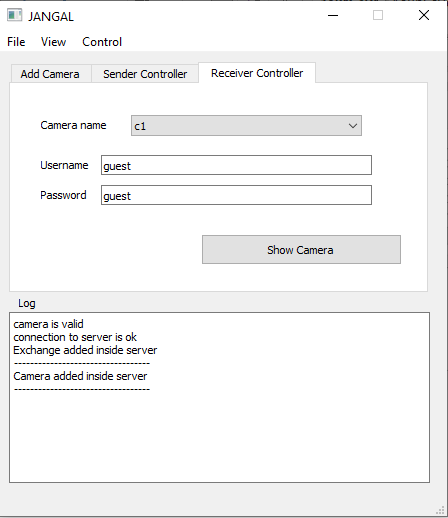
در tab، Reciever Controller ، ابتدا باید شرایط احزار هویت وارد شده و دوربین مد نظر انتخاب شود سپس با فشردن کلید show camera دوربین مد نظر اجرا خواهد شد.[[1]](#footnote-1)

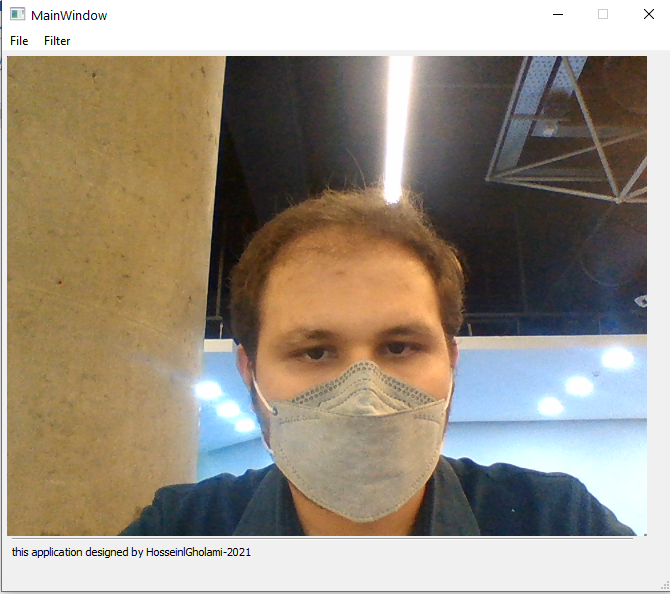
**برای اجرای این برنامه:**

برای اجرای این برنامه ابتدا دستور docker-compose up را اجرا کرده و در ادامه برنامه پایتون gui.py را اجرا کنید.

**نمایی از این نسخه از برنامه:**







**نسخه دوم:**

در این نسخه ui به صورت قابل استفاده برای تعداد نامحدود دوربین در دو برنامه طراحی شده که یکی برنامه بر روی سرور اجرا شده و برنامه ای در سمت کلاینت در شبکه اجرا خواهد شد. برنامه سمت سرور اطلاعات را از دوربین ها دریافت کرده و در سرور تجمیع می­کند و برنامه سمت کاربر به سرور متصل شده و اطلاعات را دریافت می­کند. در برنامه سمت سرور میتوان دوربین را در tab ، add camera ، ارزیابی و وضعیت سرور را بررسی و در نهایت دوربین ها را در سرور با اکسچنجی مشخص تعریف کرد. و در برنامه سمت کاربر، پس لاگین کردن به سرور ، اطلاعات دوربین های ساخته شده را دریافت میکند و با فشردن کلید show camera اطلاعات دوربین در یک پنجره مجزا به نمایش در خواهد آمد. در این برنامه اطلاعات دوربین ها بر روی برنامه سرور به صورت فایل سیستم ذخیره می­شوند این بدان معناست که در صورتی که برنامه سرور بسته شود، تنظیمات دوربین ها از بین نخواهند رفت.

**عملکرد کلی:**

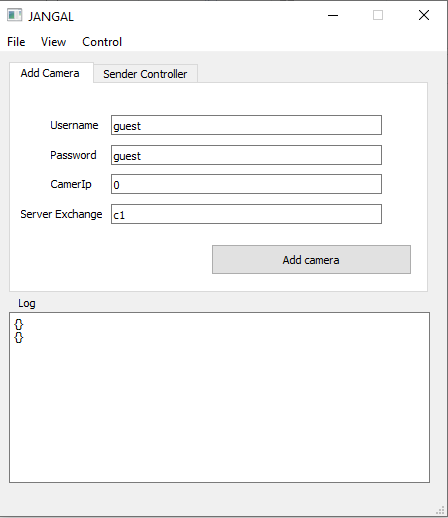
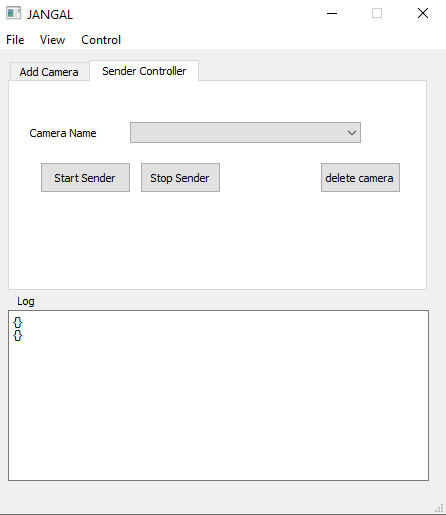
در برنامه سمت سرور، در tab، add camera با فشردن کلید add camera دستگاه وضعیت دوربین وارد شده را بررسی میکند و سپس احراز هویت برای اتصال به سرور را بررسی کرده میکند سپس در صورتی که موارد فوق مورد تایید باشد یک exchange با نام camera name بر روی سرور ایجاد میکند و در برنامه ارسال کننده و دریافت کننده اطلاعات این جزئیات را وارد میکند و نتیجه بررسی خود را در بخش پایین صفحه اعمال می­کند و همچنین اطلاعات دوربین وارد شده را در یک فایل ذخیره میکند تا در صورت بسته شدن اطلاعات دوربین های وارد شده حذف نشوند. همچنین با فشردن این کلید دو qprocess بر روی رم تخصیص داده می شوند تا برنامه ارسال کننده اطلاعات در آن اجرا شوند. سپس در tab، Sender controller ، ابتدا باید دوربین مد نظر را انتخاب و سپس با فشردن کلید start Sender برنامه ارسال اطلاعات دوربین در qprocess که در مرحله قبل ایجاد شد اجرا میشود. و با فشردن کلید stop sender ارسال اطلاعات دوربین به سرور متوقف خواهد شد. همچنین با فشردن کلید delete camera دوربین ها از لیست دوربین های موجود حذف شده و از فایل سیستم نیز حذف خواهد شد.

در برنامه سمت کاربر ابتدا باید به سرور متصل شده و پس بررسی شرایط احزار هویت ، وارد صفحه نمایش اطلاعات دوربین شود در ادامه با فشردن کلید show camera دوربین مد نظر اجرا خواهد شد.[[2]](#footnote-2)

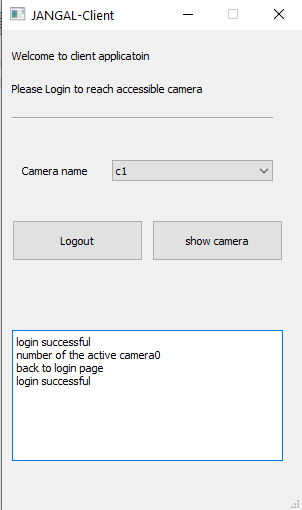
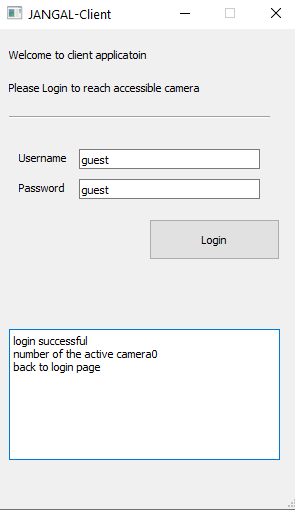
**برای اجرای این برنامه:**

برای اجرای این برنامه ابتدا دستور docker-compose up را بر روی سرور اجرا کرده و در ادامه برنامه پایتون gui.py را که در پوشه Server application اجرا شود. سپس در کامپیوتر هایی که بر روی شبکه سرور قرار گرفته اند، برنامه client.py که پوشه Client application قرار گرفته اجرا شوند.

**نمایی از این نسخه از برنامه سرور:**

**نمایی از این نسخه از برنامه کاربر:**



1. در این نسخه پردازش حدقلی به دلیل تغییر در معماری حذف شد تا برنامه دریافت کننده اطلاعات نسبت برنامه ارسال کننده اطلاعات مستقل شود تا بتوان آن ها را به سادگی بتوان داخل شبکه اجرا کرد. [↑](#footnote-ref-1)
2. در این نسخه پردازش حدقلی به دلیل تغییر در معماری حذف شد تا برنامه دریافت کننده اطلاعات نسبت برنامه ارسال کننده اطلاعات مستقل شود تا بتوان آن ها را به سادگی بتوان داخل شبکه اجرا کرد. [↑](#footnote-ref-2)