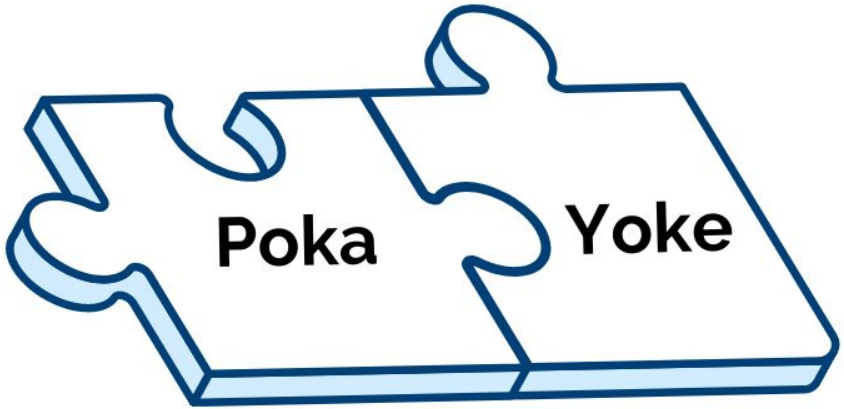
# Gruppe 5:
# Poke Yoke sorting system using a UR3e robot

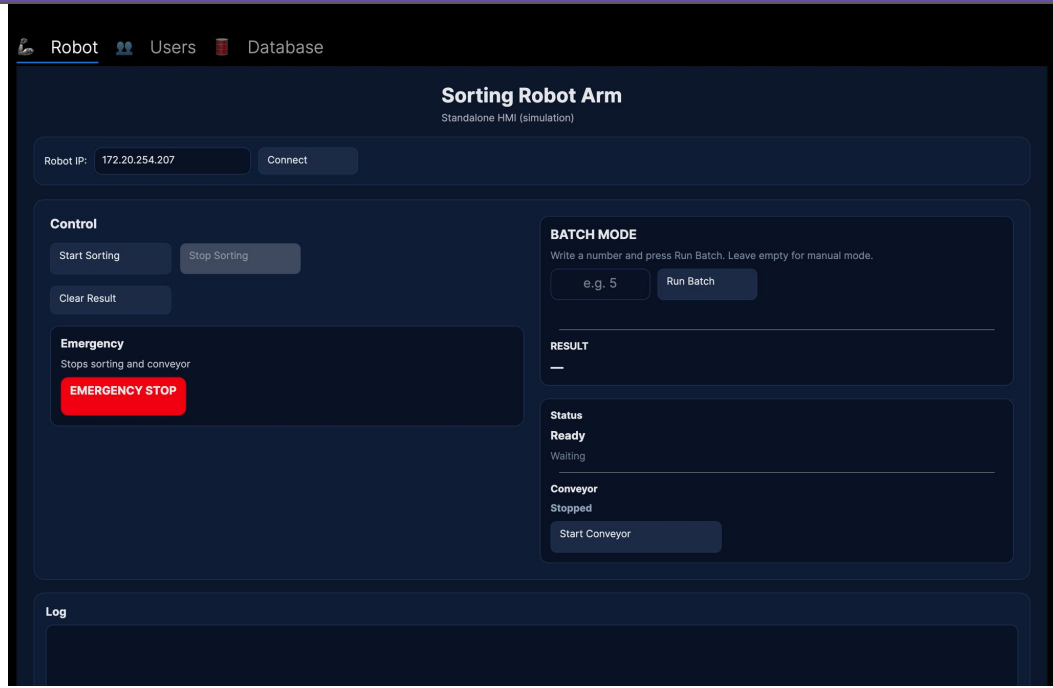
# Problem

- Poka-yoke
- Industry 4.0 setting
- Consistency, reliability & stability
- Representative case -> real production environment
- Changed several times



# Løsning

- Kiss principle
- Desktop GUI
- Login as safety measure
- Role based access
- Easy to use
- Robotic side -> motion pendant
- Stops when criteria is met



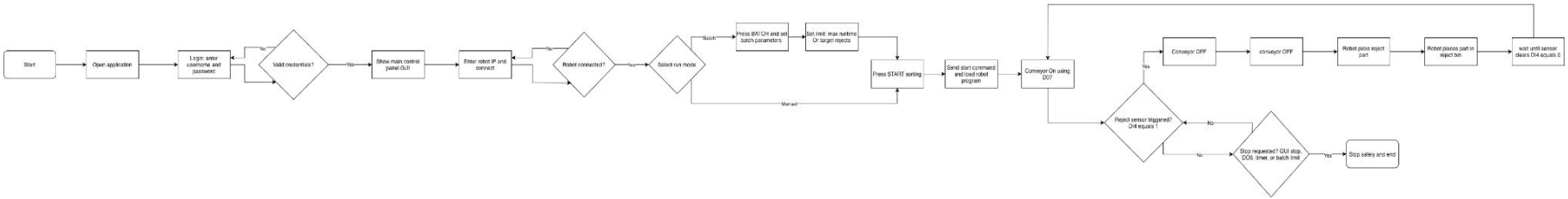
# Video af projekt:

https://www.youtube.com/watch?v=CZpJa-QoSUE

# Design og implementering: System Overview:

- PC side: Login + GUI + Database
- Network: TCP/IP to Ur robot
- Robot side: UR program + Input/Output
- Hardware: Conveyor + Reject sensor (DI4)

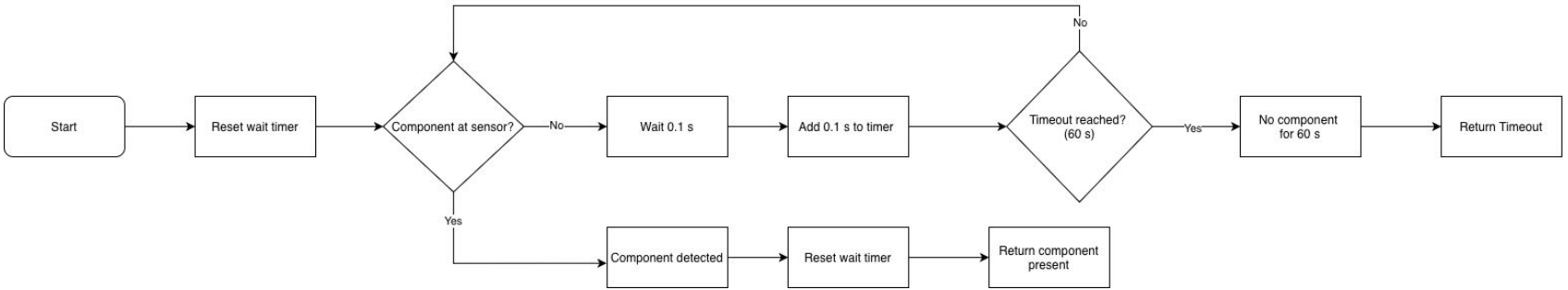
Operator -> Login -> GUI -> (dashboard 29999 + UR Script 30002) -> Robot program -> DI4 sensor + Conveyor -> Reject bin /Output.



# Design og implementering: Sorting logic on robot (Event-driven):

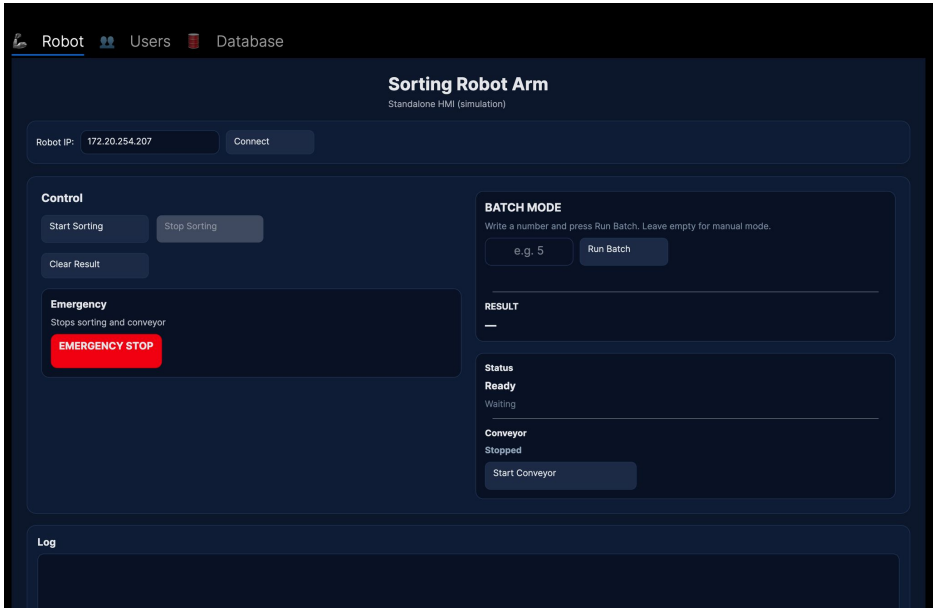
- Event-driven loop
- DI4 triggers only rejects
- Conveyor OFF -> pick reject -> place reject -> resume
- Correct parts pass through (no DI4)

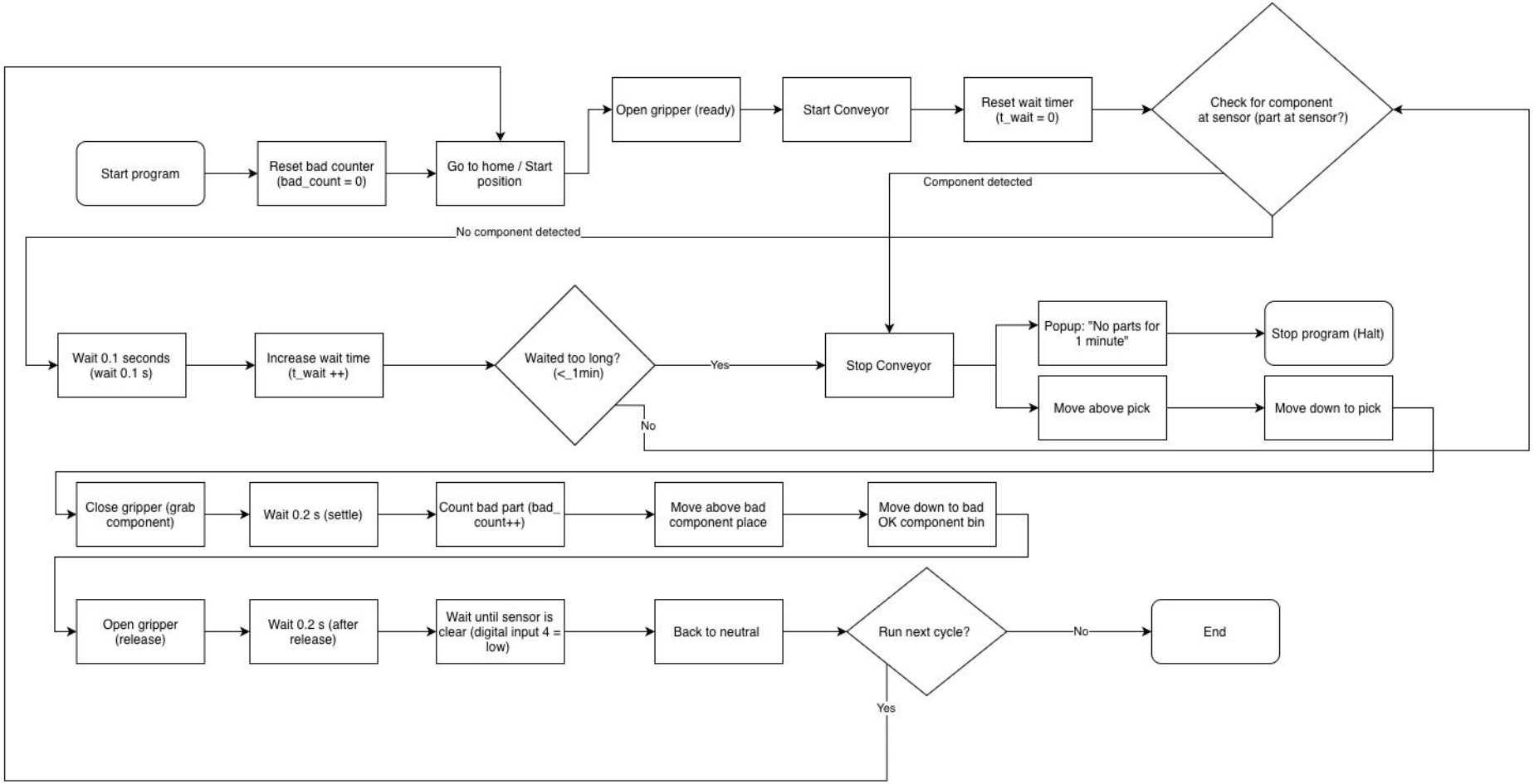
Conveyor ON -> wait DI4 -> if DI4=1: stop belt -> pick -> place -> home -> wait clear -> resume.



# Design og implementering: GUI design: minimal + access control:

- Minimal operator panel (not programming)
- Login + roles: Operatør vs Admin
- Admin: brugeroprettelse og database-vedligehold
- Controls: Connect/Start/Stop/Status.




Start program
Reset bad counter (bad_count = 0)
Go to home / Start position
Open gripper (ready)
Start Conveyor
Reset wait timer (t_wait = 0)
Check for component at sensor (part at sensor?)
Component detected
No component detected
Wait 0.1 seconds (wait 0.1 s)
Increase wait time (t_wait ++)
Waited too long? (<_1min)
Yes
No
Stop Conveyor
Popup: "No parts for 1 minute"
Stop program (Halt)
Move above pick
Move down to pick
Close gripper (grab component)
Wait 0.2 s (settle)
Count bad part (bad_ count++)
Move above bad component place
Move down to bad OK component bin
Open gripper (release)
Wait 0.2 s (after release)
Wait until sensor is clear (digital input 4 = low)
Back to neutral
Run next cycle?
No
End
Yes

# Design og implementering: Trade-off + forbedring

- Trade-off: no validated live counter in GUI
  - reason: robot loop runs robot-side -> PC needs confirmed feedback
- Batch/timeout added for predictable demos
- Next. 2nd sensor (count all parts) + robot -> PC feedback

# Programmering

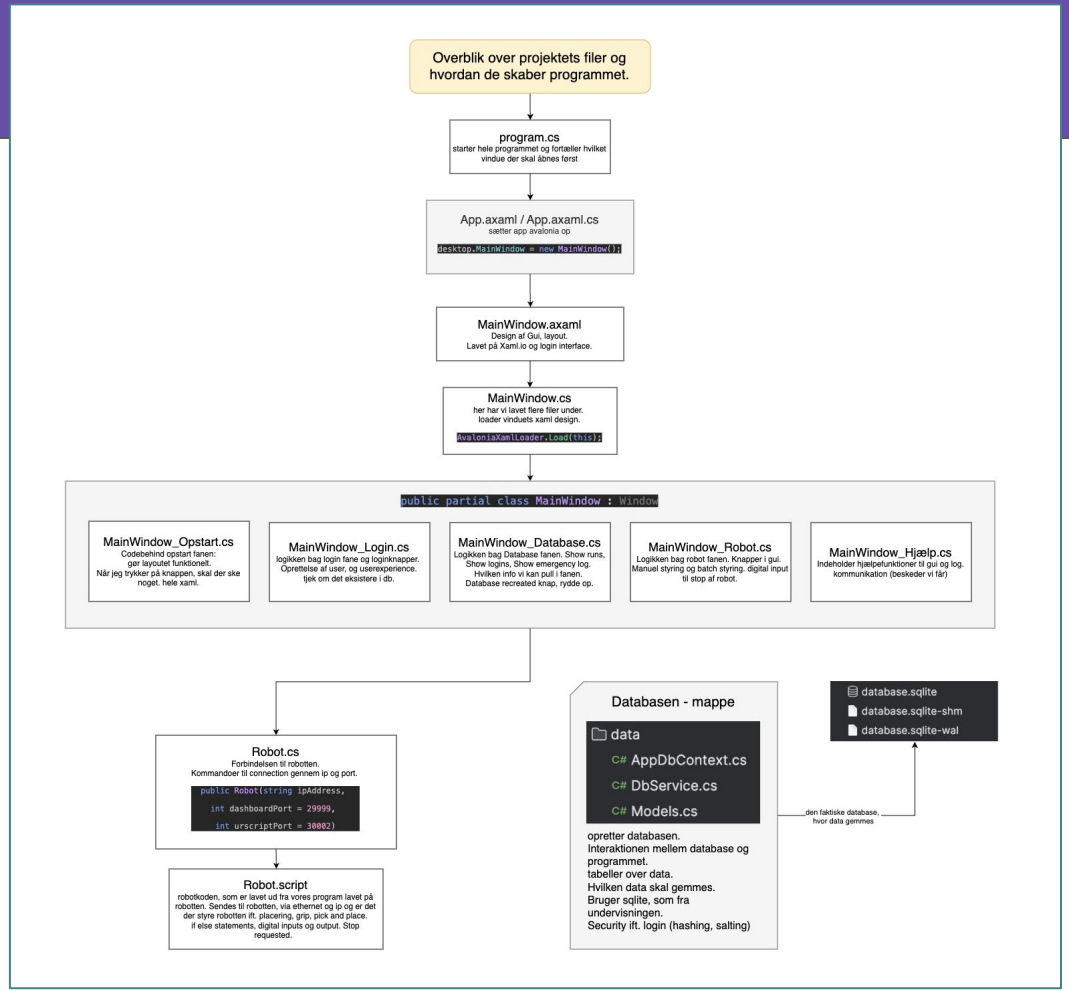
Practice makes perfect
simplicity and competence.

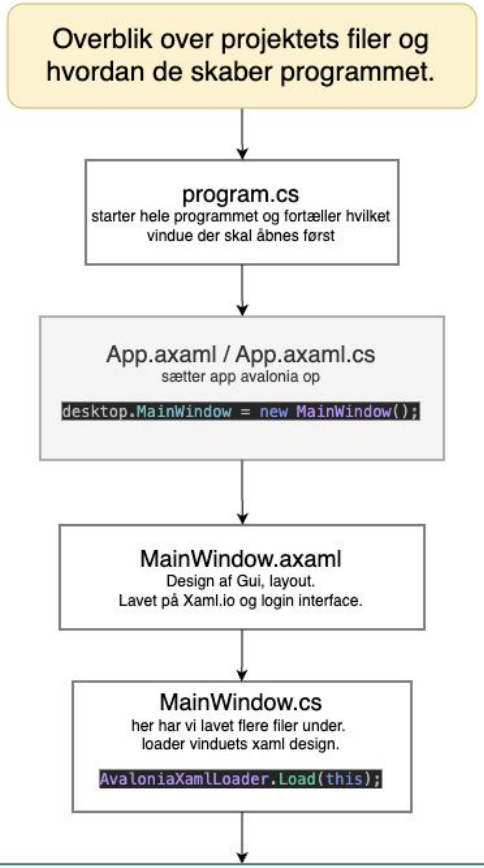
Opbygning af vores
solution

Mainwindow class
classes

Overblik og overskuelighed

Målet og det store billede




Overblik over projektets filer og hvordan de skaber programmet.
program.cs
starter hele programmet og fortæller hvilket vindue der skal åbnes først
App.axaml / App.axaml.cs
sætter app avalonia op
desktop.MainWindow = new MainWindow();
MainWindow.axaml
Design af Gui, layout.
Lavet på Xaml.io og login interface.
MainWindow.cs
her har vi lavet flere filer under.
loader vinduets xaml design.
AvaloniaXamlLoader.Load(this);

```
public partial class MainWindow : Window
```

### MainWindow_Opstart.cs
Codebehind opstart fanen:
gør layoutet funktionelt.
Når jeg trykker på knappen, skal der ske noget. hele xaml.

### MainWindow_Login.cs
logikken bag login fane og loginknapper.
Oprettelse af user, og userexperience.
tjek om det eksistere i db.

### MainWindow_Database.cs
Logikken bag Database fanen. Show runs, Show logins, Show emergency log.
Hvilken info vi kan pull i fanen.
Database recreated knap, rydde op.

### MainWindow_Robot.cs
Logikken bag robot fanen. Knapper i gui.
Manuel styring og batch styring. digital input til stop af robot.

### MainWindow_Hjælp.cs
Indeholder hjælpefunktioner til gui og log.
kommunikation (beskeder vi får)

### Robot.cs
Forbindelsen til robotten.
Kommandoer til connection gennem ip og port.

```
public Robot(string ipAddress,
    int dashboardPort = 29999,
    int urscriptPort = 30002)
```

### Robot.script
robotkoden, som er lavet ud fra vores program lavet på robotten. Sendes til robotten, via ethernet og ip og er det der styre robotten ift. placering, grip, pick and place.
if else statements, digital inputs og output. Stop requested.

### Databasen - mappe
- data
  - C# AppDbContext.cs
  - C# DbService.cs
  - C# Models.cs

opretter databasen.
Interaktionen mellem database og programmet.
tabeller over data.
Hvilken data skal gemmes.
Bruger sqlite, som fra undervisningen.
Security ift. login (hashing, salting)

den faktiske database, hvor data gemmes

- database.sqlite
- database.sqlite-shm
- database.sqlite-wal

# Koder og logikken bag

Logik fra undervisning

foreach statement:
foreach (var r in batchRuns)
går igennem hver gemt batch-kørsel én ad gangen.

Branching and boolean logic:
if
else
&&//

Branching bruges til at sikre, at robotten reagerer korrekt på stop, sensorer og fejl.

```
foreach (var r in batchRuns)
        {
            var endedLocal = r.EndedAt.ToLocalTime();

            _ = TryParseBatchMeta(r.Username, out var who, out var batchCycles, out var secStartUtc, out var secEndUtc);

            Log($"{endedLocal:yy-MM-dd HH:mm:ss} | Batch finished");
            Log($"    User:         {who}");
            Log($"    Batch cycles:  {batchCycles}");
            Log($"    Security:      {secStartUtc:yyyy-MM-dd HH:mm:ss}Z  ->  {secEndUtc:yyyy-MM-dd HH:mm:ss}Z");
        }

        Log("----------------------------");
    }
    catch (Exception ex)
    {
        Log("Show runs error: " + ex.Message);
    }
}
```

```
if (_robot == null || !_robot.Connected)
```

```
if (u.IsAdmin)
return "YES";
else
return "NO";
```

```
public async void ShowAllLoginsButton_OnClick(object? sender, RoutedEventArgs e)
{
    try
    {
        var users = await _db.Accounts
            .OrderBy(a => a.Username)
            .ToListAsync();

        Log("---- All logins ----");
        if (users.Count == 0)
        {
            Log("(no users)");
        }
        else
        {
            foreach (var u in users)
                Log($"{u.Username,-18} | admin={(u.IsAdmin ? "YES" : "NO")}");
        }
        Log("--------------------");
    }
    catch (Exception ex)
    {
        Log("Show all logins error: " + ex.Message);
    }
}
```

# Loops og kontinuerlige processer

Brug ift. automation

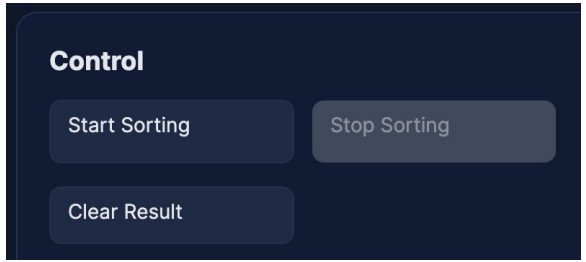
Loops ift. manual sorting og batch mode.

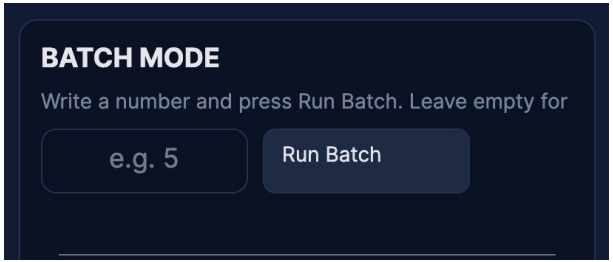
Batch cycles

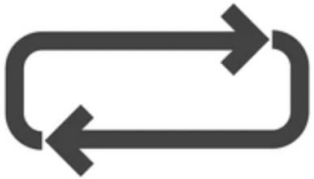
Sikkerhed ift. conveyor, spild af ressource, forebyggende håndtering

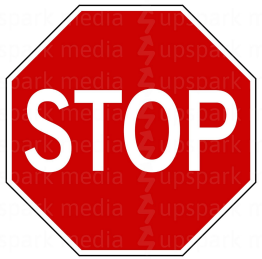
Loop+sensor= realistisk industri

Gentagne flow.






```
# Kør conveyor 60 sek og stop ALT (ingen popups)
def security_stop():
  set_standard_digital_out(7, True)
  sleep(60.0)
  set_standard_digital_out(7, False)
  halt
end
```

```
# 0 = manuel mode (kører uendeligt)
# >0 = batch mode (kører præcis N cycles)
max_cycles = {{MAX_CYCLES}}
cycles_done = 0  # tæller hvor mange cycles der er kørt
```

# Sikkerhed og database

Gemme data pga sporbarhed, log, håndtere fejl og sikkerhedshændelser.

sikkerhed ift. login puzzle

samme kode = forskellige hashes pga. salt.

beskyttelse mod angreb

- vi gemmer ikke kodeord, kun hash og salt, for at beskytte mod data beskadigelse.

```
public AccountService(AppDbContext db, PasswordHasher hasher)
{
    _db = db;
    _hasher = hasher;
}
```

```
26-01-21 09.34.54 I ---- All logins ----
26-01-21 09.34.54 I admin          I admin=YES
26-01-21 09.34.54 I ayse           I admin=NO
26-01-21 09.34.54 I hej            I admin=NO
26-01-21 09.34.54 I hejalle        I admin=NO
26-01-21 09.34.54 I jacob          I admin=NO
26-01-21 09.34.54 I prathiga       I admin=NO
26-01-21 09.34.54 I ratatouille    I admin=NO
26-01-21 09.34.54 I shamla         I admin=NO
26-01-21 09.34.54 I user           I admin=NO
26-01-21 09.34.54 I ----------------
```

```
public async Task NewAccountAsync(string username, string password, bool isAdmin = false)
{
    var (salt :byte[] , saltedPasswordHash :byte[] ) = _hasher.Hash(password);

    _db.Add(new Account
    {
        Username = username,
        Salt = salt,
        SaltedPasswordHash = saltedPasswordHash,
        IsAdmin = isAdmin
    });

    await _db.SaveChangesAsync();
}
```

# Diskussion

**Design og scope:**

- Sensor registrerer kun rejects → begrænset tælle logik
- Manglende totaltælling påvirker batch-definition
- GUI mangler bekræftet feedback fra robot
- Afhængighed af implicitte hændelser

**Forbedring**

- Ekstra entry-/exit-sensor til total flow
- Good parts = total − rejects
- Robot → PC feedback (URScript / digital output)
- Validerede tællere og data i database