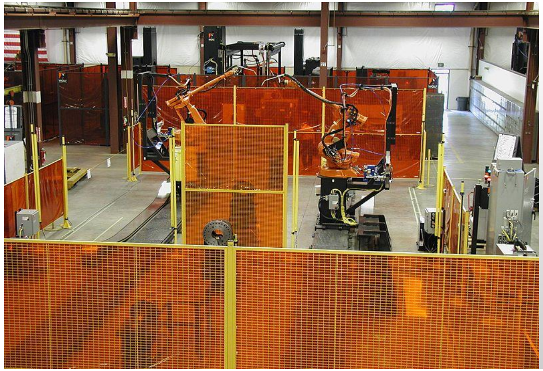
Preliminary Design and Implementation

Project 5: Industrial Robot Monitoring

...But first, some repetition: Our client

- OpiFlex Automation
- Sell industrial robots
  - Mobile platforms
  - Fenceless security
- Operators has to be on site to monitor robots

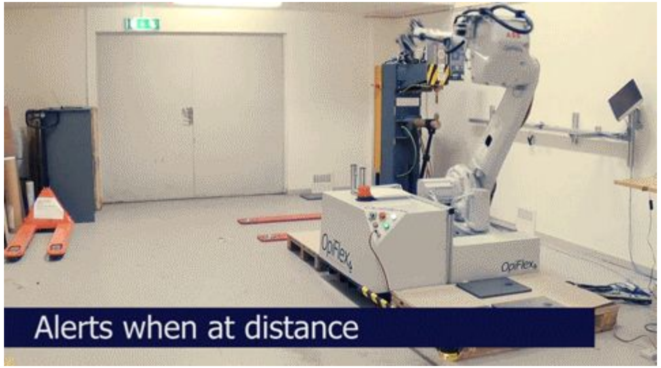


© WireCrafters / CC BY-SA 4.0

OpiFlex
We Move Automation

for increased productivity

info@opiflex.se www.opiflex.se



...But first, some repetition: Our product

- Remote monitoring of robots
- Gather relevant robot data
- Present data via android application

Important milestones since last presentation

- Design document
- Server
  - Had to reverse engineer simulator from OpiFlex
- Application prototype
  - First prototype created
- Demoed prototype for OpiFlex
  - No issues

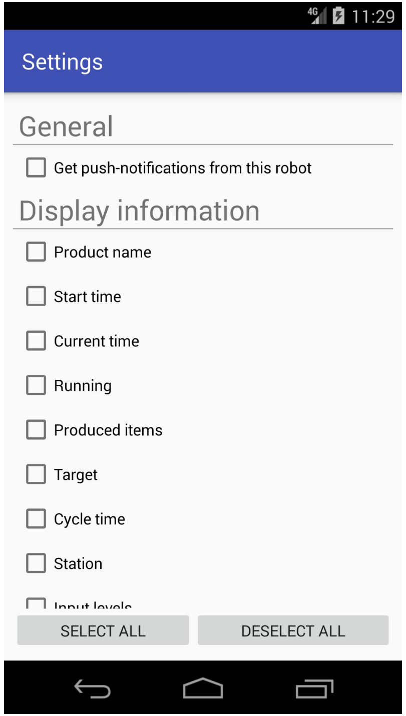
Features/Requirements

Android app

- Created pages for the app
  - Login
  - Robot Data
  - Settings Page
  - Robot list
  - Error log
- Created navigation between pages
- Push notification.

Server

- Storing data for each client
- Secure the data
- Accounts
- Emulating the robot.



Settings

General

□ Get push-notifications from this robot

Display information

□ Product name

□ Start time

□ Current time

□ Running

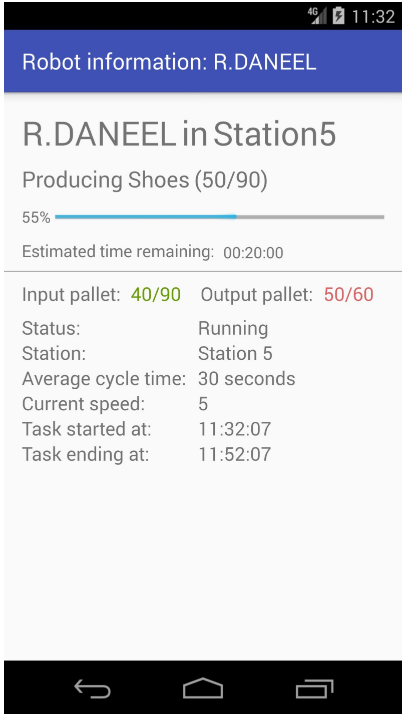
□ Produced items

□ Target

□ Cycle time

□ Station

□ Input levels



Robot information: R.DANEEL

R.DANEEL in Station5

Producing Shoes (50/90)

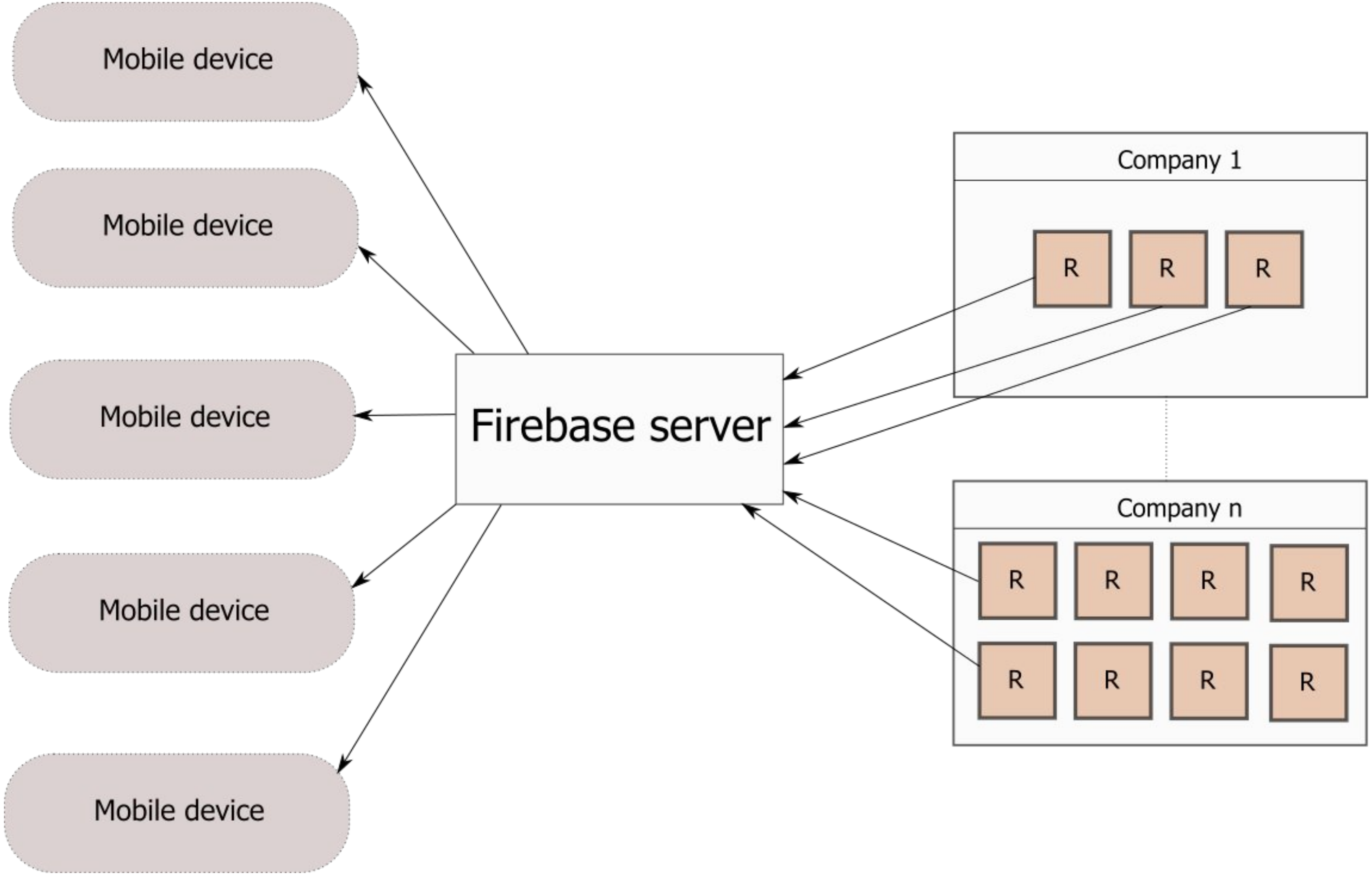
55%

Estimated time remaining: 00:20:00

Input pallet: 40/90 Output pallet: 50/60

Status:	Running
Station:	Station 5
Average cycle time:	30 seconds
Current speed:	5
Task started at:	11:32:07
Task ending at:	11:52:07

High-level design

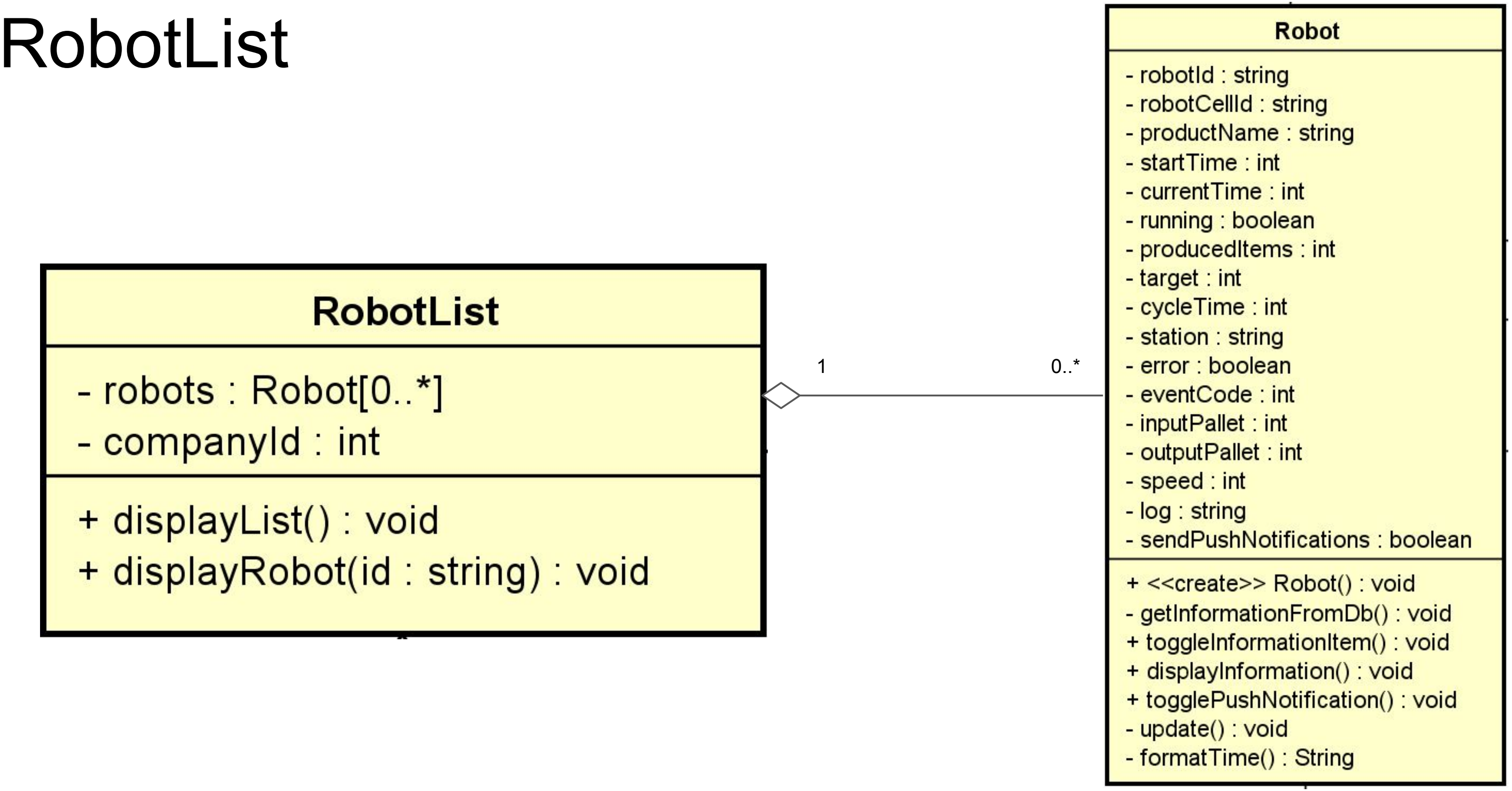


Detailed Design

Preliminary detailed design: Class diagram

- System consists of 7 classes...
- ...to support the main requirements
  - Have a list of robots
  - See detailed information about a robot
  - See list of (error) messages from a robot
  - Get notifications about important robot events
- Most important
  - RobotList
  - ErrorLog
  - PushNotification

RobotList



RobotList

- robots : Robot[0..*]
- companyId : int

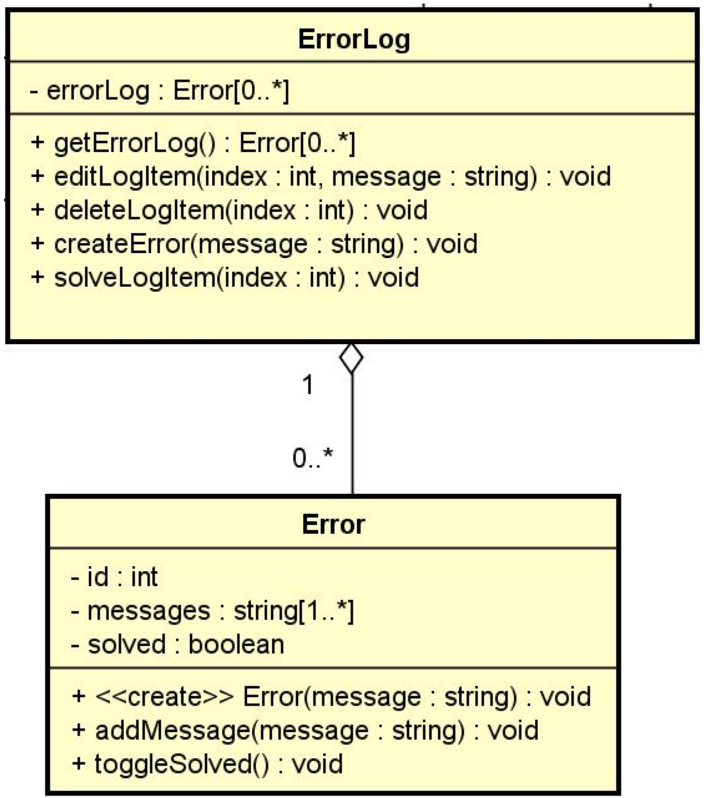
+ displayList() : void
+ displayRobot(id : string) : void

Robot

- robotId : string
- robotCellId : string
- productName : string
- startTime : int
- currentTime : int
- running : boolean
- producedItems : int
- target : int
- cycleTime : int
- station : string
- error : boolean
- eventCode : int
- inputPallet : int
- outputPallet : int
- speed : int
- log : string
- sendPushNotifications : boolean

+ <<create>> Robot() : void
- getInformationFromDb() : void
+ toggleInformationItem() : void
+ displayInformation() : void
+ togglePushNotification() : void
- update() : void
- formatTime() : String

ErrorLog



ErrorLog

- errorLog : Error[0..*]

+ getErrorLog() : Error[0..*]
+ editLogItem(index : int, message : string) : void
+ deleteLogItem(index : int) : void
+ createError(message : string) : void
+ solveLogItem(index : int) : void

1

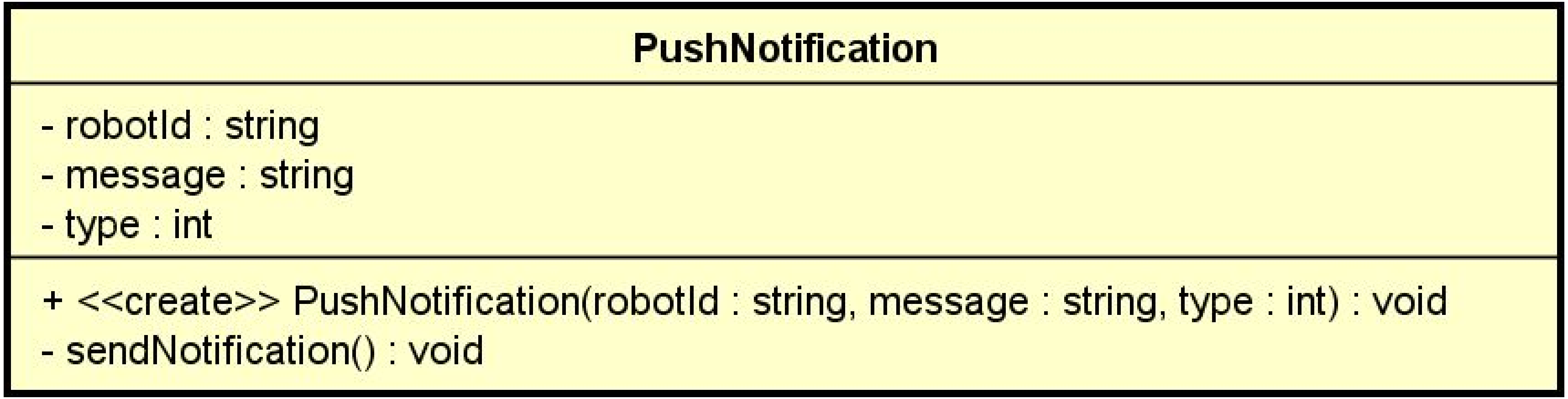
0..*

Error

- id : int
- messages : string[1..*]
- solved : boolean

+ <<create>> Error(message : string) : void
+ addMessage(message : string) : void
+ toggleSolved() : void

PushNotification



PushNotification

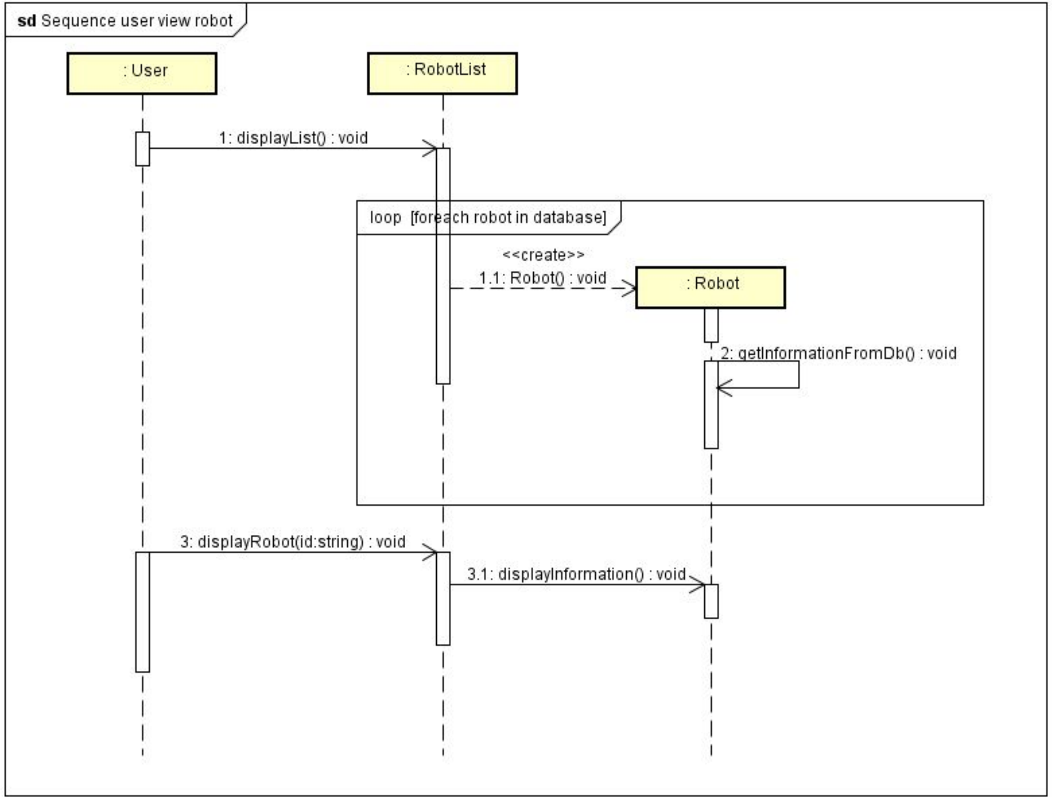
- robotId : string
- message : string
- type : int

+ <<create>> PushNotification(robotId : string, message : string, type : int) : void
- sendNotification() : void

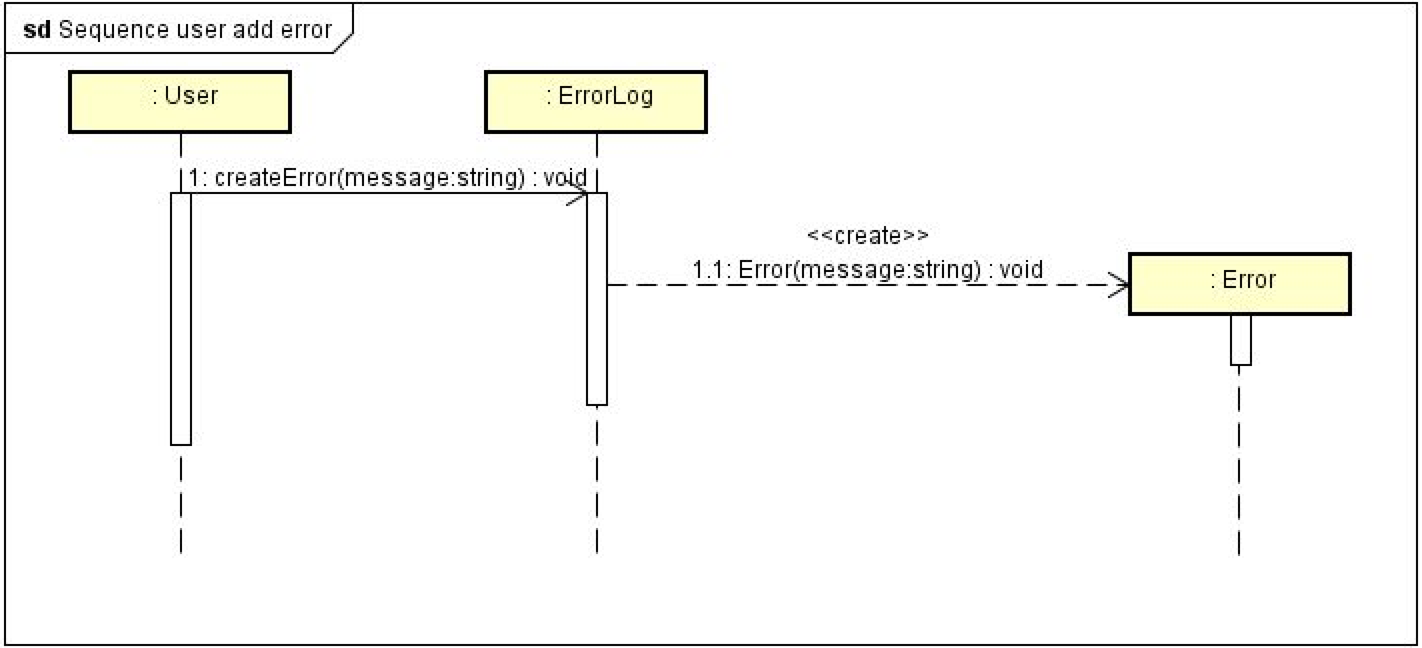
Preliminary detailed design: Sequence diagrams

- These are in an early stage
    - Will probably change a lot
- Display detailed robot information
- User add error
- User edit error
- Send notification

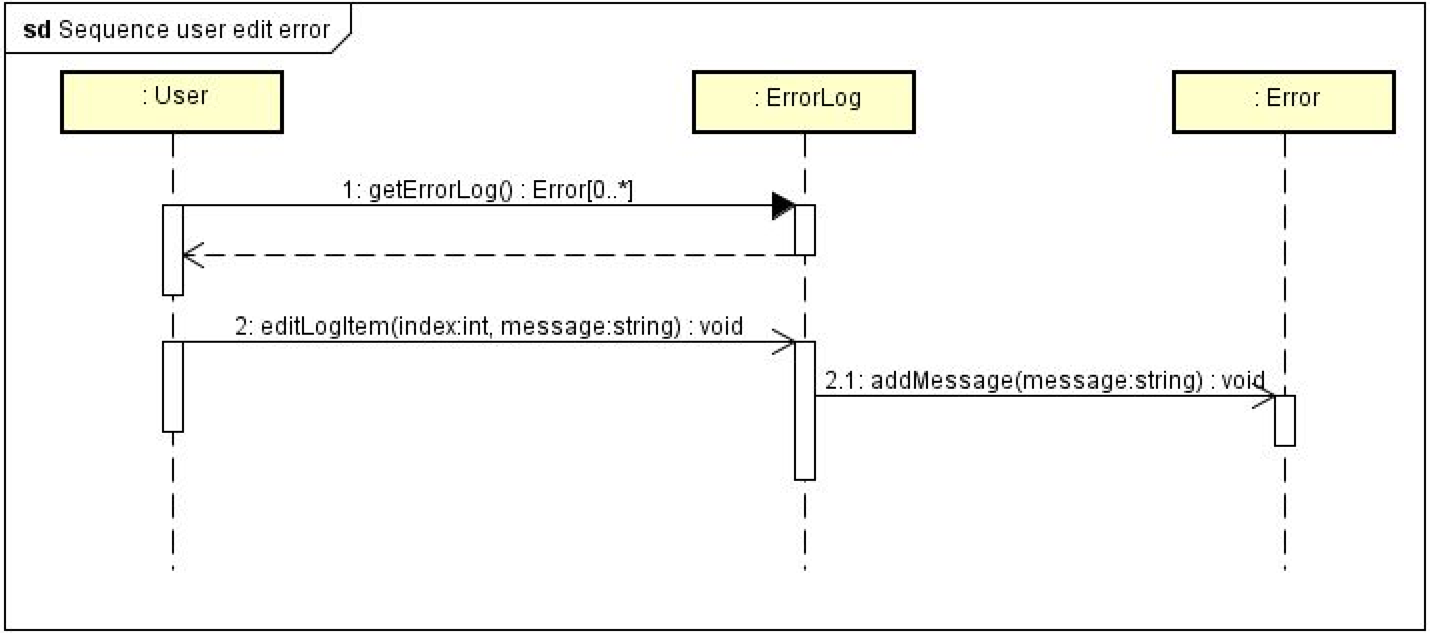
Viewing detailed robot information


sd Sequence user view robot
: User
: RobotList
1: displayList() : void
loop [foreach robot in database]
<<create>>
1.1: Robot() : void
: Robot
2: getInformationFromDb() : void
3: displayRobot(id:string) : void
3.1: displayInformation() : void

User adding an error

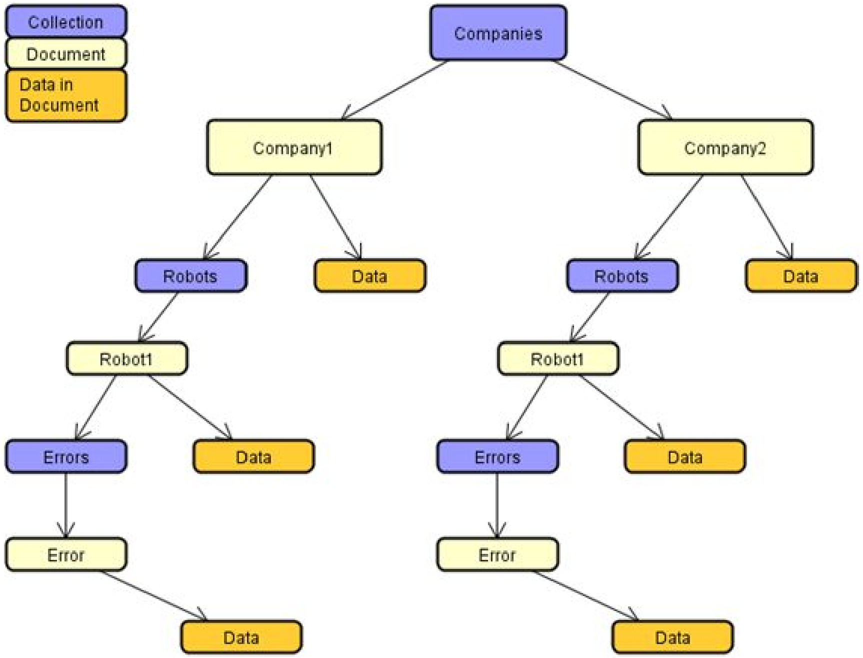

sd Sequence user add error
: User
: ErrorLog
1: createError(message:string) : void
<<create>>
1.1: Error(message:string) : void
: Error

User editing an error


sd Sequence user edit error
: User
: ErrorLog
: Error
1: getErrorLog() : Error[0..*]
2: editLogItem(index:int, message:string) : void
2.1: addMessage(message:string) : void

Preliminary detailed design: Server

- Firebase
- Real Time Database/Firestore
- Security

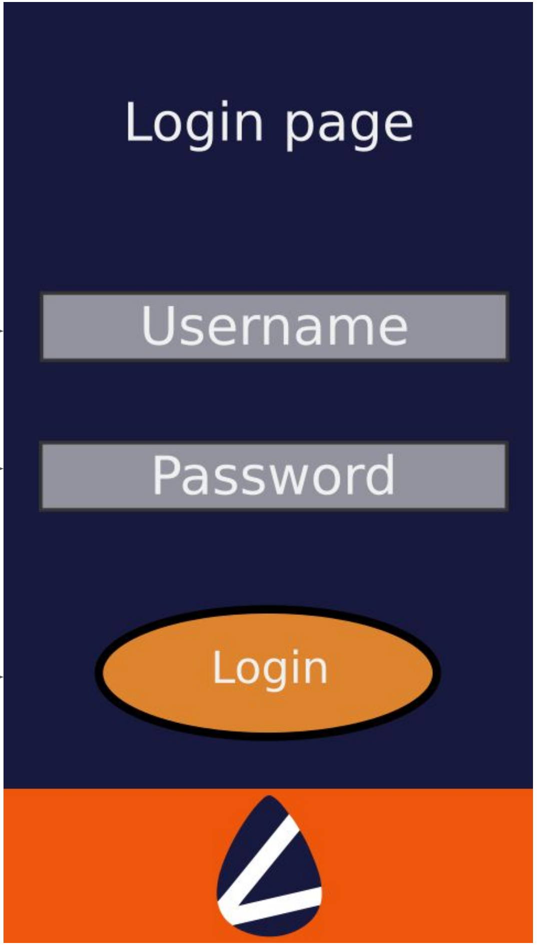


GUI Design

GUI design: context

- What goal? Display robot status
- In what situation? Anytime, anywhere (mobile app)
- What usage? Professional
- What user inputs? Touch screen (mobile app)

Login page

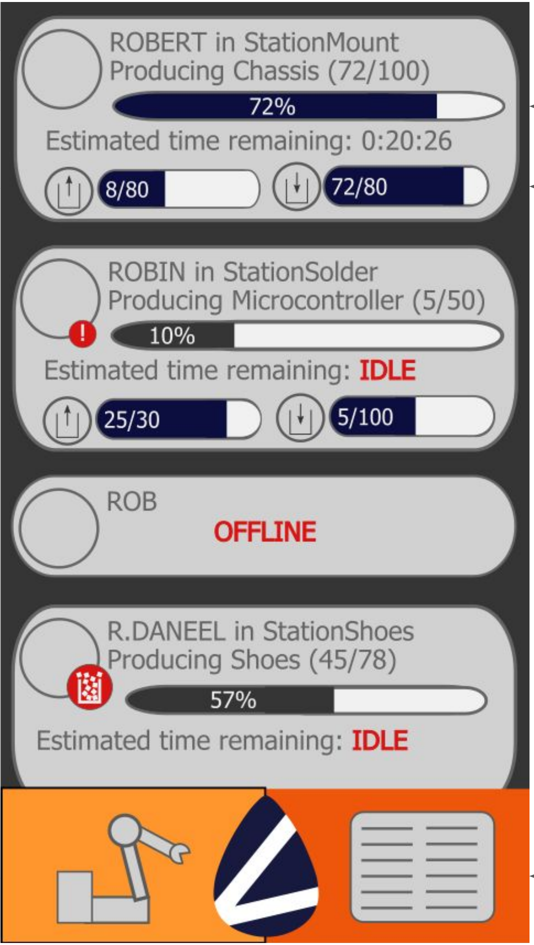


Username.

Password.

Button.

Robot list page



ROBERT in StationMount
Producing Chassis (72/100)
72%
Estimated time remaining: 0:20:26
8/80 72/80

ROBIN in StationSolder
Producing Microcontroller (5/50)
10%
Estimated time remaining: IDLE
25/30 5/100

ROB
OFFLINE

R.DANEEL in StationShoes
Producing Shoes (45/78)
57%
Estimated time remaining: IDLE

Name and station

Input pallet

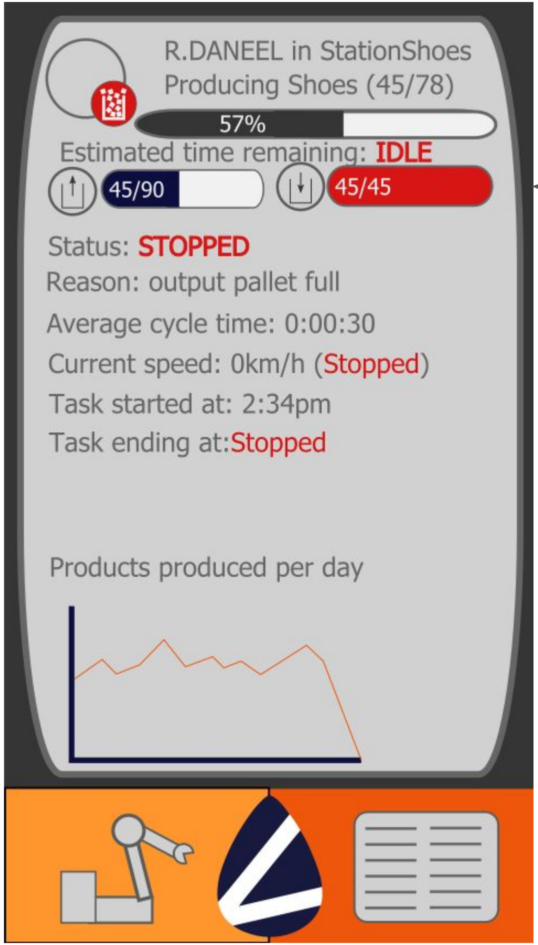
Robot list tab

Progress

Output pallet

Log tab

Robot detail page



R.DANEEL in StationShoes

Producing Shoes (45/78)

57%

Estimated time remaining: IDLE

45/90

45/45

Status: STOPPED

Reason: output pallet full

Average cycle time: 0:00:30

Current speed: 0km/h (Stopped)

Task started at: 2:34pm

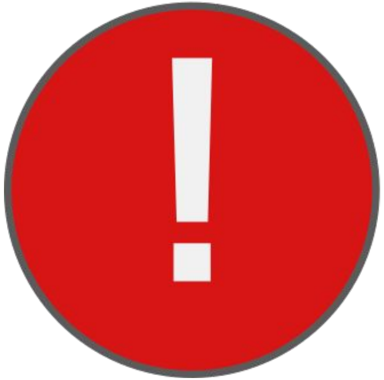
Task ending at:Stopped

Products produced per day

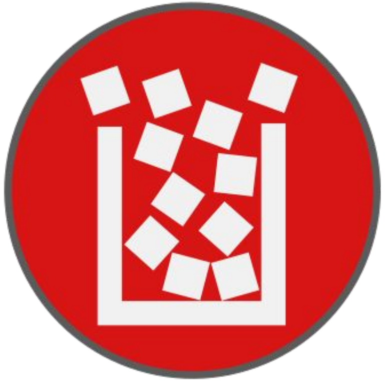
Red means BAD

Graph

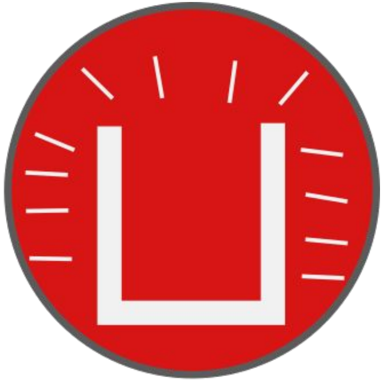
Bubbles



General error



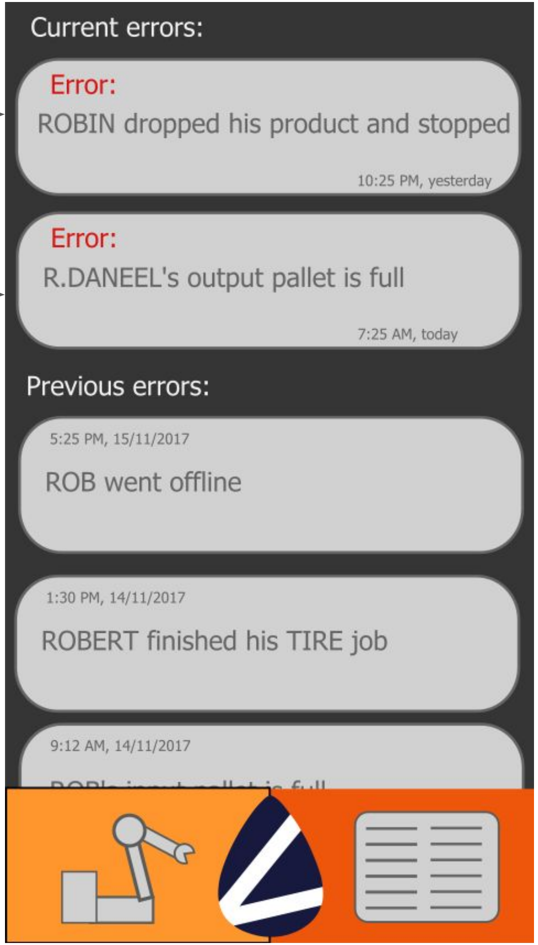
Output pallet full



Input pallet empty

Log page

Current errors



Current errors:

Error:
ROBIN dropped his product and stopped
10:25 PM, yesterday

Error:
R.DANEEL's output pallet is full
7:25 AM, today

Previous errors:

5:25 PM, 15/11/2017
ROB went offline

1:30 PM, 14/11/2017
ROBERT finished his TIRE job

9:12 AM, 14/11/2017
ROB's input pallet is full

Future plans

- Implementation
  - Connect firebase database to the application
  - Enable login authentication on the application
- Testing
  - Inhouse
  - OpiFlex
- If we have time
  - Improved design
  - Reward system