

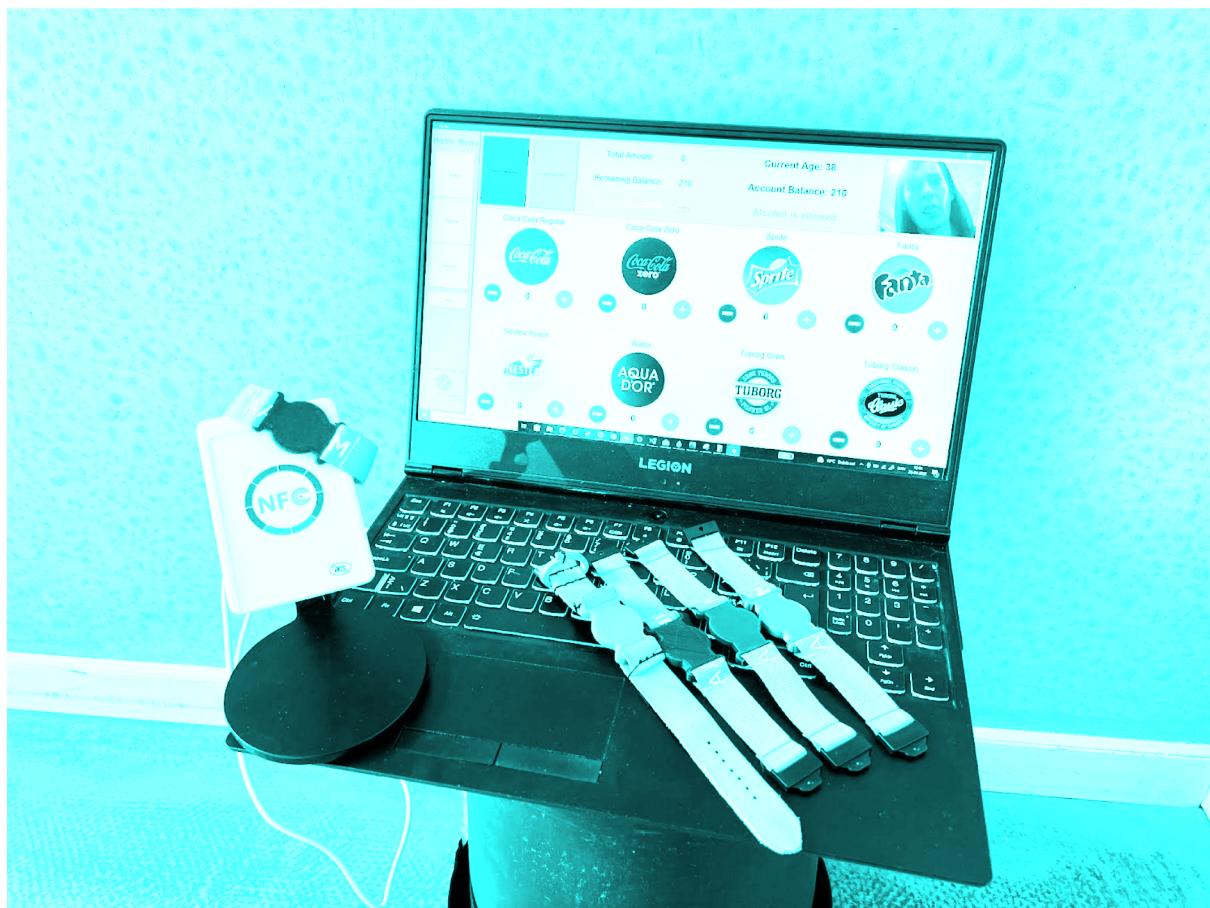
# “Bip-Bar”

Eksamensprojekt - Projektoplæg 1

*Digital Interaktion*

Lavet af: Viet, Lac, Kristian og Simon

Lærer: Mark Robert Nygaard Moore



# **Tro og Love-erklæring**

Undertegnede erklærer sig hermed bekendt med følgende:

Jeg bekræfter herved med min underskrift, at opgavebesvarelsen er udarbejdet af mig. Jeg har ikke anvendt tidligere bedømt arbejde uden henvisning hertil, og opgavebesvarel-sen er udfærdiget uden anvendelse af uretmæssig hjælp og uden brug af hjælpemidler, der ikke har været tilladt under prøven.

AARHUS GYMNASIUM, den \_\_\_\_\_

---

---

Underskrift

---

---

Underskrift

---

---

Underskrift

---

---

Underskrift

# Indholdsfortegnelse

<b>Tro og Love-erklæring</b>	<b>2</b>
<b>Indholdsfortegnelse</b>	<b>3</b>
<b>Introduktion/Resumé</b>	<b>5</b>
<b>Idégenerering</b>	<b>5</b>
Metodisk tilgang til idégerering	5
Første idégenerering til forskellige projektoplæg	5
Samling og beskrivelse af gode idéer	8
Dybere beskrivelse af interessante idéer	9
Lyskryds-skema	9
Skitsering af udvalgte idéer (Scenarier)	10
Lo-fi videoer til udvalgte idéer	12
Videre beskrivelse af teknologier til realisering af udvalgte idéer	13
Udvælgelse af idé-retning	13
<b>Endelig Kerneidé</b>	<b>14</b>
<b>Designspace</b>	<b>14</b>
<b>Projektplanlægning</b>	<b>18</b>
Prototype plan	18
Punkter i Trello	19
<b>Materialeliste</b>	<b>20</b>
<b>Målgruppeanalyse</b>	<b>20</b>
<b>Overordnet systemstruktur</b>	<b>21</b>
Systemets bestanddele	21
Flowchart over dataruter	22
Opdateret Flowchart over dataruter	23
UML over terminalstruktur	24
Opdateret UML over terminalstruktur	25
UML over server	26
<b>Prototype 1 - Form</b>	<b>27</b>
Skitsering og 3D-modeller af egne armbånd	27
Færdige Chip-designs	31
Design af rem til egne armbånd	32
Armbånd-designs	33
<b>Prototype 2 - Funktionalitet</b>	<b>34</b>
NFC-reader	34
Video med funktionalitet	34
Flowchart	35
Terminal	36
Introduktion til terminal	36
Bip Bar-siden	37
Edit User-siden	39

New User-siden	40
Server Status	41
Video af skabelse af bruger, redigering, samt første transaktioner:	41
Database	42
<b>Prototype 3 - Interaktion (ventes til at terminalen er færdig)</b>	<b>43</b>
<b>Prototype 4 - Sikkerhed</b>	<b>45</b>
Systemsikkerhed	45
Overblik	45
Brugersikkerhed	45
NFC sikkerhed	45
Serversikkerhed	46
Transaktionssikkerhed	46
Brugerdata	46
<b>Prototype 5 - App/Website</b>	<b>46</b>
<b>Prototype 6 - Samlet prototype</b>	<b>47</b>
<b>Det samlede system består af følgende primære elementer:</b>	<b>47</b>
<b>Test af prototyper</b>	<b>48</b>
Test af det fysiske design	48
Miljøtest	48
Brugertest af armbånd	49
Test af det digitale design	50
Brugertest	50
Komplet test	50
<b>Anatomy of prototypes</b>	<b>51</b>
<b>Ambient intelligence (Aml)</b>	<b>52</b>
<b>Affordances and Design + How Bodies Matter</b>	<b>54</b>
<b>Konklusion</b>	<b>56</b>
<b>Links</b>	<b>57</b>
<b>Litteraturliste</b>	<b>57</b>
<b>Bilag</b>	<b>58</b>
Brugertest resultater	58
Kildekode	60
Server kode (app.py)	60
Terminal kode (Main.py)	63
NFC-Reader.py	83
Pickletest.py	84
Camera test.py	84
entryWithPlaceHolder.py	86

# Introduktion/Resumé

I SUCK AT INTRODUCTIONS hehe

Gennem dette projekt har vi skabt et sikkert betalingssystem for alle aldre, som kan bruges primært på festivaler og andre store events, men også på lokationer som barer, hoteller og lignende.

I gennem det her projekt har vi undersøgt hvordan man kan lave et godt betaling system til bar og festivaler. Hvor vores koncept er at der skal være et nemt betalings system som er nemt at tage i brug. Det har vi bl.a. gjort ved at teste med flere forskelle prototyper at armbånd. Men også ved at lave et helt betalings system fra bunden af. Vi har også være ude og spørge hvad folk gerne vil se i sådan her et produkt. Det hele ud et i færdig bygget system som man kan tage i brug.

## Idégenerering

### Metodisk tilgang til idégenerering

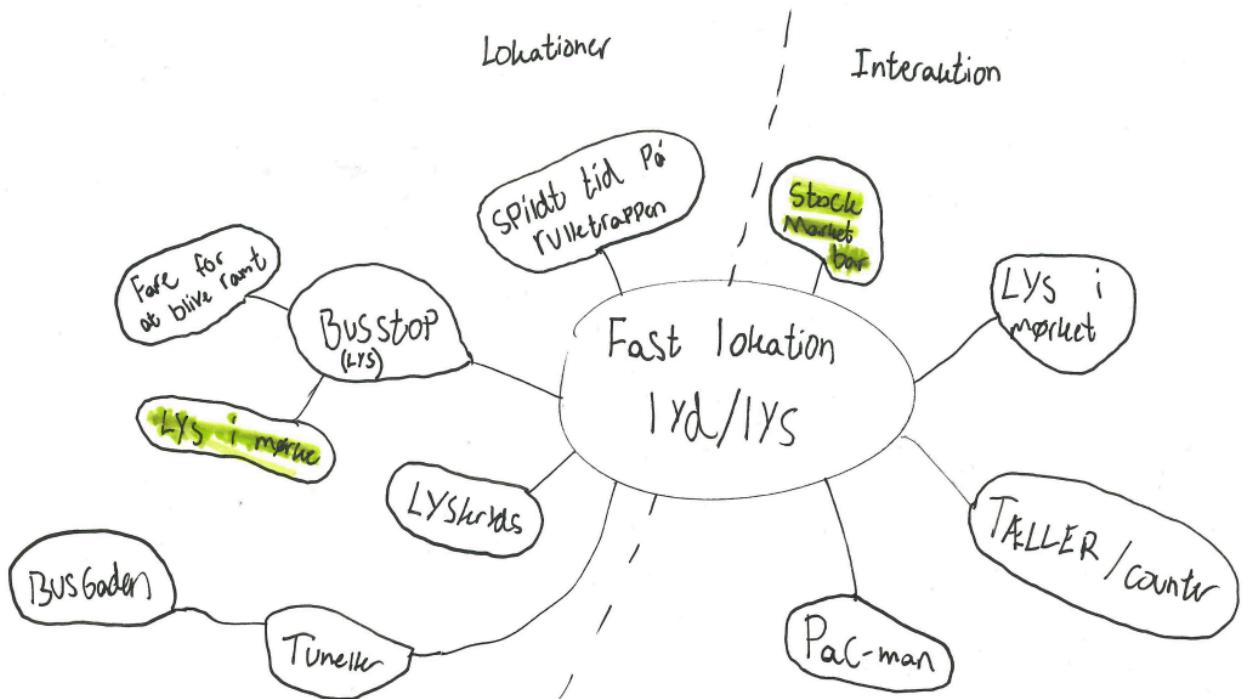
For at finde gode idéer, måder at sætte tankerne i gang, og måder at skabe diskussion i gruppen, for at skabe gode idéer til projektets retning, laves en stor idégenerering, som efterfølgende indsættes metodisk, for at den bedst mulige idé kan udvælges.

### Første idégenerering til forskellige projektoplæg

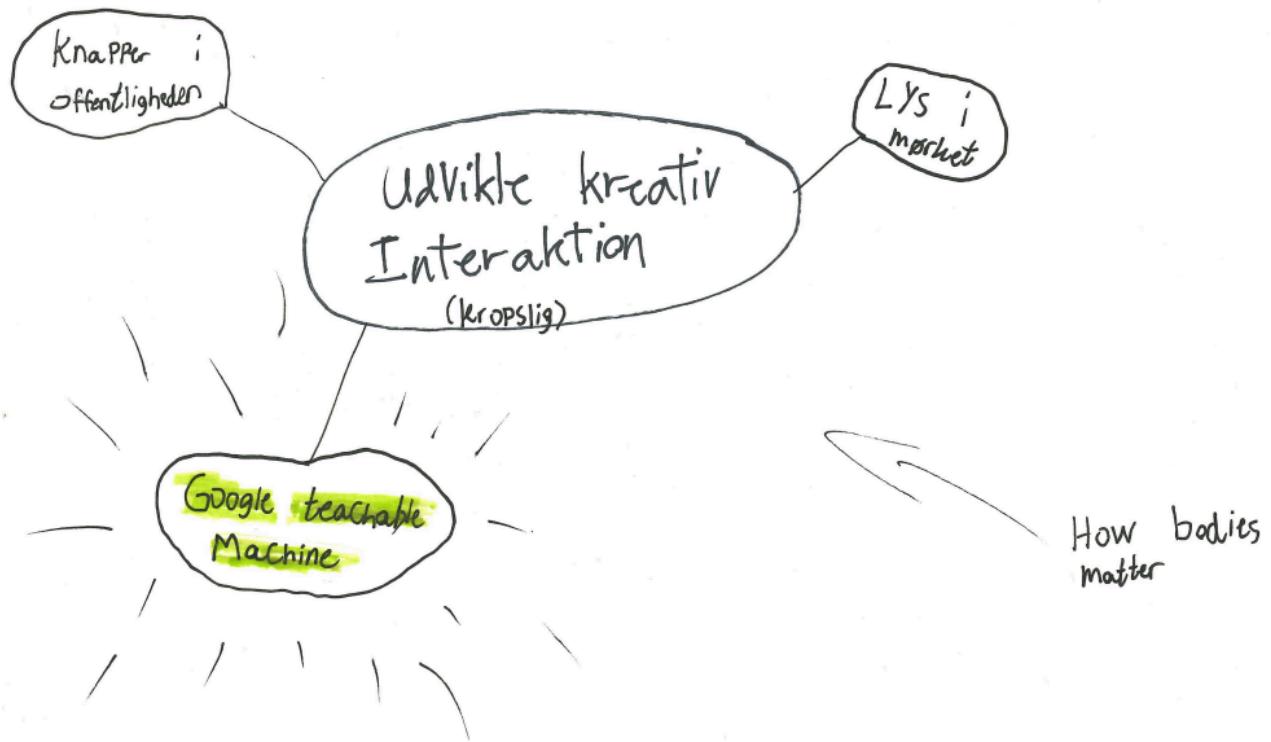
I den første del af dette projekt laves der idégenerering, hvor der genereres idéer ud fra de forskellige projektoplæg. Vi har i gruppen snakket og stemt på hvilke projekter ville være interessante at lave, vi har valgt at markere det med en grøn overstregningstusch. De følgende idéer gruppen, valgte at arbejde videre med beskrives i næste afsnit.



## Projektoplæg 2



## Projektoplæg 3



felles interaktioner

## Projektoplæg 4

Strøm gennem mennesker  
eller sensor fra Afdøde

### ESCAPE Room

eller felles  
interaktioner

Terminal-hacking

lysretrninger....

Wires  
(Angug os)

Bomb-defuse

Room

ESCAPE

## Projektoplæg 5

### IOT

Forbindelse  
digital og fysisk  
Verden

intet for  
fysisk verden  
internet

### Datavisualisering

Stock market  
bar

# Samling og beskrivelse af gode idéer

Herunder ses udvalgte idéer og koncepter fra den første brainstorm beskrevet. Det er de tanker vi har haft omkring idéer markeret med grøn, som vi har valgt at snakke mere om i gruppen, og skabe idéer, koncepter, tanker og forbehold gennem gruppesamtale, som gør at idéerne kan beskrives mere i dybden.

## Fjernstyrede biler med trafik - Projektoplæg 1

- Der laves en interaktion til en børnevenlig festival eller andet arrangement, hvor børn og unge kan lære regler i trafikken, og køre med fjernstyrede biler i en lille interaktiv verden.

## Color run interaktiv portal - Projektoplæg 1

- Dette idé er lavet i forbindelse med eventet color run, der afholdes i Aarhus. Her laves der en portal, som skyder farver ud når deltagerne fra løbet, passerer portalen.

## Bip-bar - Projektoplæg 1 (eller 5)

- I bip-baren kan man mere sikkert og med kundefordel betale for alkohol til festivaler, på barer eller til andre arrangementer, hvor det kan være en ulempe at medbringe fysiske betalingsmetoder. Man får et armbånd med en NFC-chip, som er tilknyttet en bruger på en centraliseret server, og kan bruge indsats kredit fra armbåndet. Sikkerhed ville være vigtigt i projektet, og der ville derfor skulle implementeres små informationer om brugeren som navn, alder og køn, sådan at svindel kan stoppes hurtigt. Derudover er det vigtigt at en chip kan spærres med det samme, sådan at stjålne og mistede armbånd hurtigt kan spærres.

## Teachable på festival - Projektoplæg 1

- En sjov (kropslig) interaktion baseret på Google Teachable, som ville kunne bruges på børnevenlig festival eller andet stort event. Her ville man eventuelt kunne vinde fede præmier, eller finde ud af hvad teknologien kan, som en introduktion til en verden med machine learning.

## Busstop lys i mørke - Projektoplæg 2

- Hvis man er på et busstop om natten så er der nogle lys som lyser hele stoppet op når bussen kommer tæt på. Det gør det nemmere for buschaufføren at se om der er nogle der skal med.

## Stock market bar - Projektoplæg 2 (eller 5)

- Det er en bar/fredagscafé, hvor alt efter hvor meget af de forskellige drikkevarer der bliver drukket, så stiger/falder prisen efter udbud/efterspørgsel. Derudover baseres prisen også på hvor mange folk der kommer til arrangementet. Så hvis der ikke er så mange så er prisen på drikkevarerne lav. Men hvis der er mange så stiger den til en højere pris pga af demand for det.

### **Google Teachable til kreativ kropslig interaktion - Projektoplæg 3**

- Her laves der en model ud fra Google Teachable, som kan genkende kropslige bevægelser. Der placeres en storskærm, hvor et videokamera opfanger bevægelserne og viser dem på skærmen. Brugerens bevægelser bliver nu vist på skærmen med forskellige baggrunde og udseende.

### **Escape Room - Projektoplæg 4**

- Vi havde tænkt på at designe et escape room hvis vi kunne få kontakt til et escape room virksomhed, hvor vi har forskellige idéer til escape roomet som kan blive inddragede.

## **Dybere beskrivelse af interessante idéer**

I denne sektion udvælges idéer med høj interesse, som derefter beskrives og researches yderligere. Først udvælges idéerne med høj interesse i et lyskryds-skema. Grønne idéer udforskes yderligere.

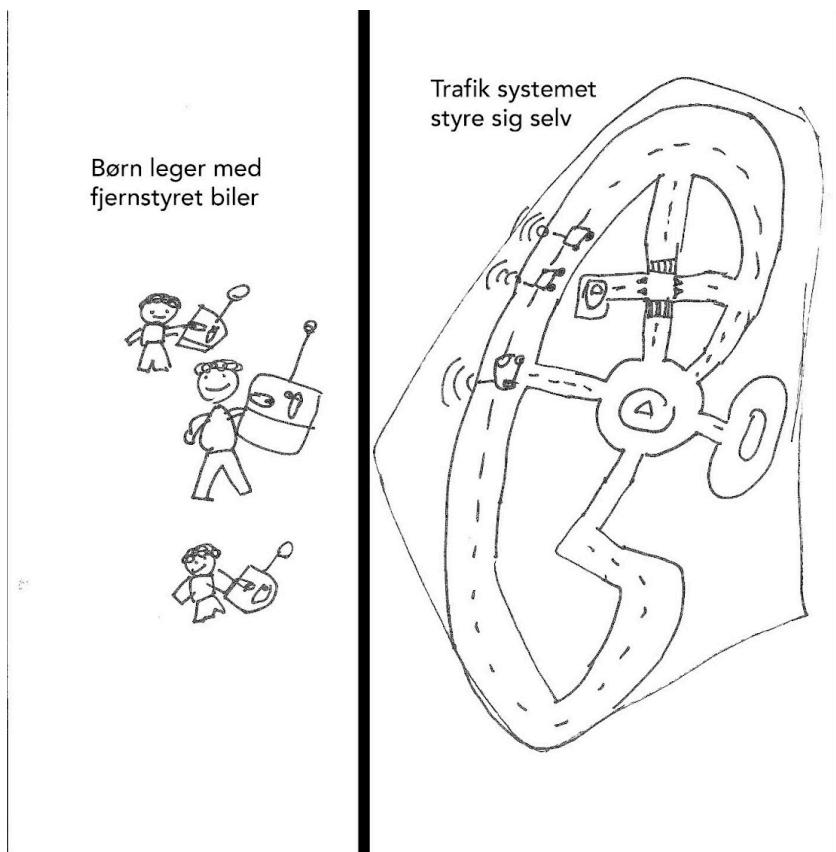
### **Lyskryds-skema**

Idé	Grøn	Gul	Rød
Fjernstyrede biler med trafik			
Color run interaktiv portal			
Bip-bar			
Teachable på festival			
Busstop lys i mørke			
Stock market bar			
Google Teachable til kreativ kropslig interaktion			
Escape Room			

## Skitsering af udvalgte idéer (Scenarier)

Herunder ses scenarier som skitserer principperne bag de tre udvalgte koncepter:

### Fjernstyrede biler med trafik



## Bip-bar

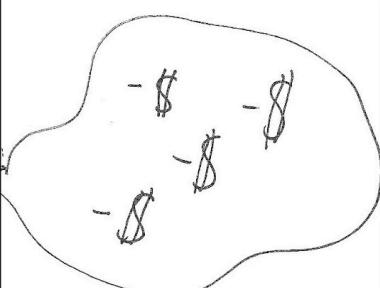
Erik er på festival, og er blevet meget tørstig



Han betaler med sit armbånd, der forbinder direkte til hans kroner



Modtageren tager imod hans kroner, og transaktionen bliver godkendt



Kronerne trækkes automatisk så Erik ikke skal bekymre sig om det

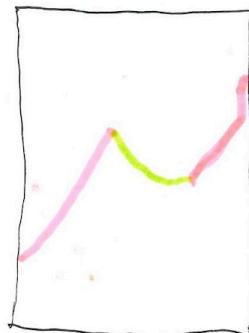


Eriks lillebror har også anskaffet sig et børnearmbånd, han bruger til betaling

## Stock market bar

### Cola

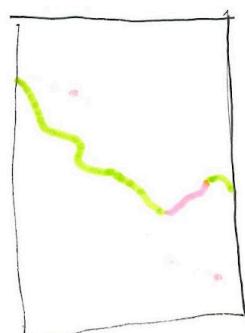
Priserne stiger  
"Venter med at købe"



21 kr

### Fanta

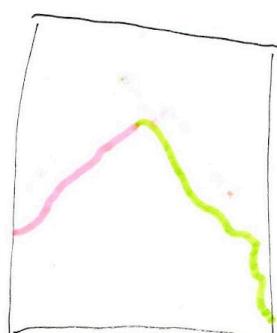
Priserne falder  
"Skynd jer at købe"



14 kr

### Sprite

"Godt tidspunkt at købe"

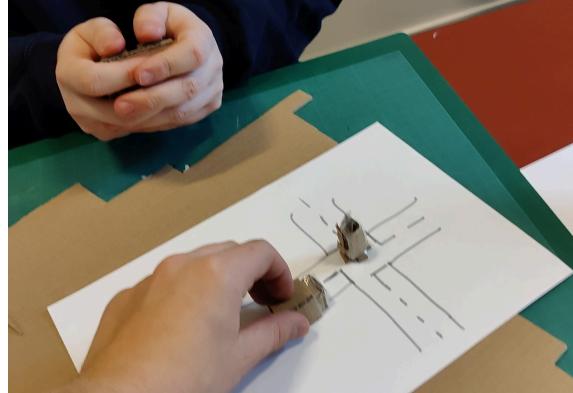


7 kr

## Lo-fi videoer til udvalgte idéer

### Fjernstyrede biler med trafik

Videoen kan ses ved linket til google drev i fodnoten<sup>1</sup>, eller vedhæftet i afleveringen på It's Learning under navnet; "Fjernstyrede biler med trafik Lo-Fi.mp4".



### Bip-bar

Videoen kan ses ved linket til google drev i fodnoten<sup>2</sup>, eller vedhæftet i afleveringen på It's Learning under navnet; "Bip-bar Lo-Fi.mp4".



### Stock market bar

Videoen kan ses ved linket til google drev i fodnoten<sup>3</sup>, eller vedhæftet i afleveringen på It's Learning under navnet; "Stock market bar Lo-Fi.mp4".



<sup>1</sup> <https://drive.google.com/file/d/1krLcM-4vaVkJCuuardcf1fP7sxrlCdfG/view?usp=sharing>

<sup>2</sup> <https://drive.google.com/file/d/1Es2aEkcEy8G2Psxz26rM7RNdJtXuacz8/view?usp=sharing>

<sup>3</sup> <https://drive.google.com/file/d/1kwq2h-jc0QTLMpYryaH1o8pDC6umLjCI/view?usp=sharing>

## Videre beskrivelse af teknologier til realisering af udvalgte idéer

### Fjernstyrede biler med trafik

Idéen ville kunne baseres på Arduino, sådan at man kan lave interaktive lyskryds og lignende med LED-dioder. Desuden ville man skulle skaffe biler til at køre rundt, og eventuelt udstyre dem med blinklys. I projektet ville der også være en masse praktisk arbejde, for at skabe den mest imponerende, interaktive og realistiske lille verden.

Herunder ses en opsummering af eventuelle materialer:

- Arduino
- LED-lys
- Små fjernstyrede biler
- Træplade, maling og dekorationer.
- Eventuelle automatiske livlige elementer (Mennesker, skibe, fly, osv.)

### Bip-bar

Idén vil være baseret omkring NFC-Chips som vil bruges sammen med reader og selve kassen. Hvor kassen henter data fra serveren for at se om det er den rigtige person.

Herunder ses en opsummering af eventuelle materialer:

- NFC-Chips
- NFC-Reader/Writer
- Armbånd som er tyverisikrede
- Server i python (køres på pc) (nfcpy-lib)

### Stock market bar

Idéen består af et algoritme baseret aktiesystem, hvor der bruges matematik til at vurdere priserne for drikkevarerne. Prisen påvirkes også af efterspørgslen, som systemet registrerer ved inputs. Der bruges Arduino til at samle inputs, hvorefter det sendes videre til systemet.

Herunder ses en opsummering af eventuelle materialer:

- Computer
- Skærm(e)
- Input-kilder
- Arduino
- Python og visualiserings-library. Evt. matplotlib.

## Udvælgelse af idé-retning

Efter de grønne idéer er blevet undersøgt yderligere, er det gennem gruppesamtale blevet valgt at arbejde videre med **Bip-bar**. Vi synes at det er mere spændende og mere relevant for os at arbejde med siden der er nogle teknologier vi har kendskab til i forvejen, og nogle som er nye for os. Denne laves efter **projektoplæg 1**.

# Endelig Kerneidé

Den udvalgte idé, Bip-bar, kan kort beskrives som et sikkert betalingssystem for alle aldre, som kan bruges primært på festivaler og andre store events, men også på lokationer som barer, hoteller og lignende.

I bip-baren kan man mere sikkert, og med kundefordele, betale for mad og drikke til festivaler, på barer eller til andre arrangementer, hvor det kan være en ulempe at medbringe fysiske betalingsmetoder. Man får et armbånd med en NFC-chip, som er tilknyttet en bruger på en centraliseret server, og kan bruge indsats kredit fra armbåndet. Sikkerhed ville være vigtigt i projektet, og der ville derfor skulle implementeres små informationer om brugeren som navn, alder og køn, sådan at svindel kan stoppes hurtigt. Derudover er det vigtigt at en chip kan spærres med det samme, sådan at stjålne og mistede armbånd hurtigt kan spærres.

## Designspace

Herunder opsættes et designspace, som bruges til at danne refleksion og perspektiv, og dermed udfolde idén. Til dette opstilles der en række kriterier, der skal hjælpe med at udforske hvilke interaktionsmuligheder, der er mulige. Derefter laves tekniske spørgsmål, som viser nogle af de vigtige overvejelser som skal tages i designprocessen.

### Kriterier:

Nu opstilles kriterier, som hjælper med at fastsætte vigtige parametre i produktudviklingen:

#### Fysisk:

- Vandtæt
- Sikkerhed fysisk
- Holdbarhed
- Forskellige størrelser

#### Digitalt:

- Sikkerhed digitalt
- Pålidelighed
- Brugerkontrol
- Benefits

Grafisk repræsentation i Miro<sup>4</sup>:



### Tekniske spørgsmål

Der opstilles en række tekniske spørgsmål for at danne en ramme om projektet.

- Hvordan gør man produktet behageligt at bære rundt på og benytte sig af?
- Hvordan sikrer man at der haves fysisk og digital sikkerhed i systemet?
- Hvordan gør man interaktionen simple og børnevenlig?
- Hvilke platforme kan anvendes til brugerkontrol?
- Hvilke fordele kan lokke brugeren til at anvende systemet?
- Hvilke situationer kan produktet blive utsat for, og hvordan sikrer man produktets holdbarhed?

### Afgrænsning

Ud fra de forrige opstillet spørgsmål, kan projektet nu afgrænses til at tage udgangspunkt i at skabe et betalingssystem. Der vil bliver undersøgt og testet, hvordan man kan lave et simpelt system, der også kan anvendes af børn. Fysisk og digital sikkerhed er essentielt for systemets funktionalitet. Derfor vil der blive undersøgt, hvordan man bedst muligt kan styrke systemets sikkerhed.

---

<sup>4</sup> Design Space i Miro  
([https://miro.com/app/board/uXjVOJfvjQ8=/?share\\_link\\_id=93355106485](https://miro.com/app/board/uXjVOJfvjQ8=/?share_link_id=93355106485))

## Analysefase

Der laves en undersøgelse af de muligheder af armbånde der allerede findes på markedet

Type	Beskrivelse
	<p>Her bruges der plastic til at fastholde chippen. Hvis det indeholder en chip bliver RFID chippen lamineret, og dette beskytter den dermed mod væske og stress.</p> <p><b>Fordele:</b> Nem og billig løsning og kan produceres i store mængder. Passer til de fleste håndled størrelser da den kan tilpasses. Fungerer godt, hvis begivenheden foregår over kort tid.</p> <p><b>Ulemper:</b> Den kan ikke om justeres, og er dermed permanent bundet til den indstillet størrelse. Plastikkens skarpe hjørner kan skabe irritation over længere periode. Kan ikke genbruges igen, og kan gå i stykker over tid.</p>
 <b>RFID stof armbånd</b>	<p>Her bruges der plastik til at beskytte hvis armbåndet indeholder RFID chip, hvor der anvendes stof til at spænde det fast til håndleddet.</p> <p><b>Fordele:</b> Dyre end plastik, men stadig billigt at fremstille i store mængder. Kan tilpasses efter de fleste håndled. De er mere behagelige og robuste i forhold til plastik.</p> <p><b>Ulemper:</b> De kan ikke justeres, og bliver låst til den indstillet størrelse. De kan ikke genbruges. Derudover suger stoffet vand, hvilket kan blive hygiejnisk over tid.</p>
	<p>Disse armbånd er bestående af papir med termo trykt plast, der beskytter papiret. Dette gøre armbåndet delvist modstandsdygtigt overfor vand og skarpe genstande. Derudover er det muligt at printe sit eget design.</p> <p><b>Fordele:</b> Billigt at fremstille i store mængder. Har bløde hjørner i forhold til plastik armbåndet, og kan også justeres så den passer til de fleste håndled.</p>

	<p><b>Ulemper:</b> Hvis plast forseglingen brydes, kan chippen og armbåndet udsættes for vand. Dermed er chippen mere utsat for fugt. Armbåndet kan heller ikke genbruges og justeres.</p>
 RFID Silikone armbånd	<p>Armbåndet er lavet i silikone, hvor det sætter sig fast på håndledet ved brug af elastisk kraft.</p> <p><b>Fordele:</b> Det er lavet i et mere premium materiale sammenlignet med de andre armbånd. Det kan tages af, og genbruges til flere arrangementer. Det lavet i et behageligt materiale, som er modstandsdygtigt for vand.</p> <p><b>Ulemper:</b> Armbåndet kan ikke justeres til alle håndleds størrelser, og kan dermed være en udfordring, hvis man er for stor eller for lille. Spændmekanismen gør det nemt for folk at stjæle det. Derfor har disse armbånd ikke samme sikkerhed som de forrige.</p>

Efter analysen kan, der nu igangsættes en brainstorm over idéer. Ud fra denne brainstorm opstilles der 3 forskellige idéer, som opstilles i et kravmatrix skema, hvor de vurderes ud fra de opstillede kriterier. Der giver hermed en score mellem 1-10, for hvordan de overholder kravene.

	Engangs armbånd NFC-chip med plastik rem	Engangs stof armbånd med NFC-chip dækket i plastik.	Genbrugeligt armbånd af silikone med indbygget NFC-chip	Genbrugelig NFC-chip forseglet i plastik med udskiftelig rem
Sikkerhed fysisk	7	8	3	8
Vandtæt	7	1	10	9
Holdbarhed	3	7	7	7
Justerbar	6	6	4	8
...				
Brugerkontrol	#	#	#	#
Pålidelighed	#	#	#	#
Digitalt	#	#	#	#

sikkerhed				
Benefits	#	#	#	#
Sum	23	22	24	32

De digitale parametre kan ikke vurderes på forhånd da de ikke er udviklet endnu. Derfor sættes disse værdier til blank, da de ville have værdi for alle de forskellige varianter.

# Projektplanlægning

## Prototype plan

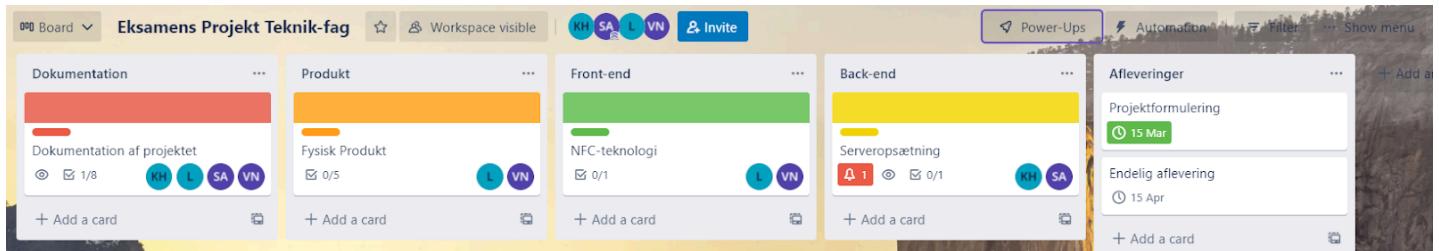
Der bliver lavet et skema over de dimensioner der er i det her projekt. Da vi regner med at vi har nok tid til at nå alle dimensioner så er de næsten alle lyseblå. Dette gør vi for at vi kan få skabt det bedste produkt.

Prototype af	Beskrivelse	Giver os	Ift. Design Space
Form	<ul style="list-style-type: none"> <li>• Plastik</li> <li>• Nfc Chip</li> <li>• Materialer</li> </ul>	Mulighed for at teste hvor behageligt at have og hvor sikert det sidder på armen	Sikkerhed fysisk, Vandtæt, Holdbarhed, Forskellige størrelser.
Funktionalitet	Nfc Chip som gør vi kan sende data til en computer og backend.	Giver et bruger-id til dem i baren så de kan se hvem det er og så personen kan betale.	Brugerkontrol, Sikkerhed fysisk, Sikkerhed digitalt, Pålidelighed,
Interaktion	En NFC-Chip reader som kan modtag data fra NFC-Chippen.	Tester interaktionen mellem form og funktionalitet i forskellige situationer.	Holdbarhed, Pålidelighed, (Sikkerhed digitalt), (Brugerkontrol)
Sikkerhed	<ul style="list-style-type: none"> <li>• Computer ved baren(Front end)</li> <li>• Serveren (Backend)</li> </ul>	Vi skal have testet hvor sikker vores Nfc-Chip er men også hvor sikker vores backend er.	Sikkerhed digitalt, Pålidelighed, Sikkerhed fysisk,
Evt. App	App til brugerstyring	Mulighed for at teste brugerkontrol, som ikke laves i baren.	Brugerkontrol, Sikkerhed digitalt,
Samlet prototype	En samling af alle 4 (5) ting	Vurdering bliver gjort fra hvert enkelt som så bliver samlet for at hvilke prototype der er bedst.	Alle parametre testes. (Måske på nær benefits)

## Punkter i Trello

Projektets tidsplan styres i programmet Trello. Her kan man inddele opgaver, sætte deadlines og få visuelt overblik over projektets gang.

For at der ikke kommer for mange tasks på Trello-boardet, laves der overordnede kategorier, hvori der laves checklists. Derudover laves der deadlines på afleveringer og deadlines gennem projektet. Alt dette kan ses herunder<sup>5</sup>:



<sup>5</sup> Trello link - (<https://trello.com/b/LHTyaqzR/eksamens-projekt-teknik-fag>)

# Materialeliste

Der opstilles en materialelister over de komponenter, der skal anvendes til at fremstille produktet. På denne måde skabes der et overblik over hvilke omkostninger, som er forbundet med produktionen af produktet.

Digital Interaktion Eksamensprojekt - 2022				
Antal gruppemedlemmer:		4		
Materialer:				
Antal:	Beskrivelse:	Pris/stk [DKK]:	Pris i alt:	Link?
10	NTAG215-NFC – RFID tag, Rund PVC	8,5	85	<a href="https://idekort.dk/vare/ntag215-nfc-rfid-tag-rund-pvc-oe25-mm-selvklaebende-hvid">https://idekort.dk/vare/ntag215-nfc-rfid-tag-rund-pvc-oe25-mm-selvklaebende-hvid</a>
10	NFC Label – NTAG RFID tag	9,85	98,5	<a href="https://idekort.dk/vare/nfc-label-ntag-rfid-tag-rund-pet-oe25-mm-selvklaebende-hvid">https://idekort.dk/vare/nfc-label-ntag-rfid-tag-rund-pet-oe25-mm-selvklaebende-hvid</a>
10	Papirarmbånd med tryk Design selv	0,86	93,6	<a href="https://jmband.dk/papirarmbånd/10-papirarmbånd-med-tryk-design-selv.html">https://jmband.dk/papirarmbånd/10-papirarmbånd-med-tryk-design-selv.html</a>
10	Plastarmbånd L farver På lager	1,38	13,8	<a href="https://jmband.dk/plastikarmbånd/18-plastarmbånd-l-farver-på-lager.html">https://jmband.dk/plastikarmbånd/18-plastarmbånd-l-farver-på-lager.html</a>
10	Silikone armbånd uden tryk På lager	4,94	28	<a href="https://jmband.dk/silikonearmbånd/217-silikone-armbånd-uden-tryk-på-lager.html">https://jmband.dk/silikonearmbånd/217-silikone-armbånd-uden-tryk-på-lager.html</a>
5	Bæredygtige Festivalarmbånd lavet af genbrugsplast	5,6	49,4	<a href="https://jmband.dk/stofarmbånd/282-Bæredygtige-Festivalarmband.html">https://jmband.dk/stofarmbånd/282-Bæredygtige-Festivalarmband.html</a>
4	Urspændende	39	185	<a href="https://www.evermart.dk/spaende-til-20-mm-urrem">https://www.evermart.dk/spaende-til-20-mm-urrem</a>
1	Touchskærm	2699	2699	<a href="https://www.elgiganten.dk/product/computer-kontor/skarme-tilbehør/monitor/aoc-16t2-156-transportabel-skarm-sort/225073">https://www.elgiganten.dk/product/computer-kontor/skarme-tilbehør/monitor/aoc-16t2-156-transportabel-skarm-sort/225073</a>
		TOTAL:	<b>3252,3</b>	
(Total / n gruppemedlemmer):			813,075	

# Målgruppeanalyse

Den primære målgruppe produktet, har vi tænkt os at det ville være primært til forældre som tager deres børn med på børnevenlige festivaler. Her ses det tit at børn får lov til at gå alene rundt på festivalpladsen, mens forældrene laver andre ting. Ofte får børn penge med på traditionelle måder på sådanne festivaler, for at de kan købe ting til dem selv. Dette løser vores armbånd, da børnene vil have sværere ved at miste deres armbånd, samt ikke vil have ægte valuta på sig, og dermed ikke have noget værdifuldt som kriminelle kan stjæle fra dem eller som de kan komme til at miste.

Vi har også tænkt på sekundære målgrupper, som vores produkt kunne interessere. De som tager på diskoteker, barer og klubber, ville kunne bruge teknologien som en mere permanent løsning. Her kan man samarbejde med allerede eksisterende firmaer, som prøver at gøre betaling i nattelivet mere sikker (f.eks. Nightpay), om armbånd som man beholder permanent, som er tilknyttet de alternative betalingssystemer. På den måde har man mindre chance for at miste sin telefon, sit Dankort, osv. Teknologien kunne også bruges på festivaler mere generelt for alle deltagere, og man ser også at lignende teknologier bruges på steder som for eksempel Smukfest.<sup>6</sup>

De ovennævnte målgrupper udgør et stort potentiel marked, både med armbånd til korttidsbrug og med armbånd som har til intention at blive brugt flere gange.

**For at samle op, kan teknologiens målgrupper umiddelbart beskrives som:**

**Primære målgruppeområder:**

- Børnevenlige festivaler

**Sekundære målgruppeområder:**

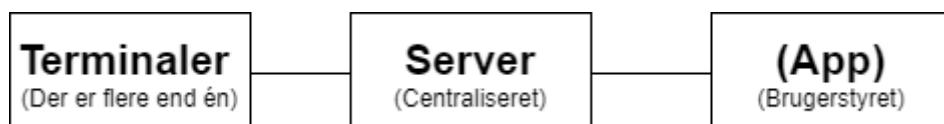
- Nattelivet
- Festivaler og lignende generelt

## Overordnet systemstruktur

### Systemets bestanddele

Systemet består overordnet set af tre systemdele, hvoraf det ene bliver brugerstyret.

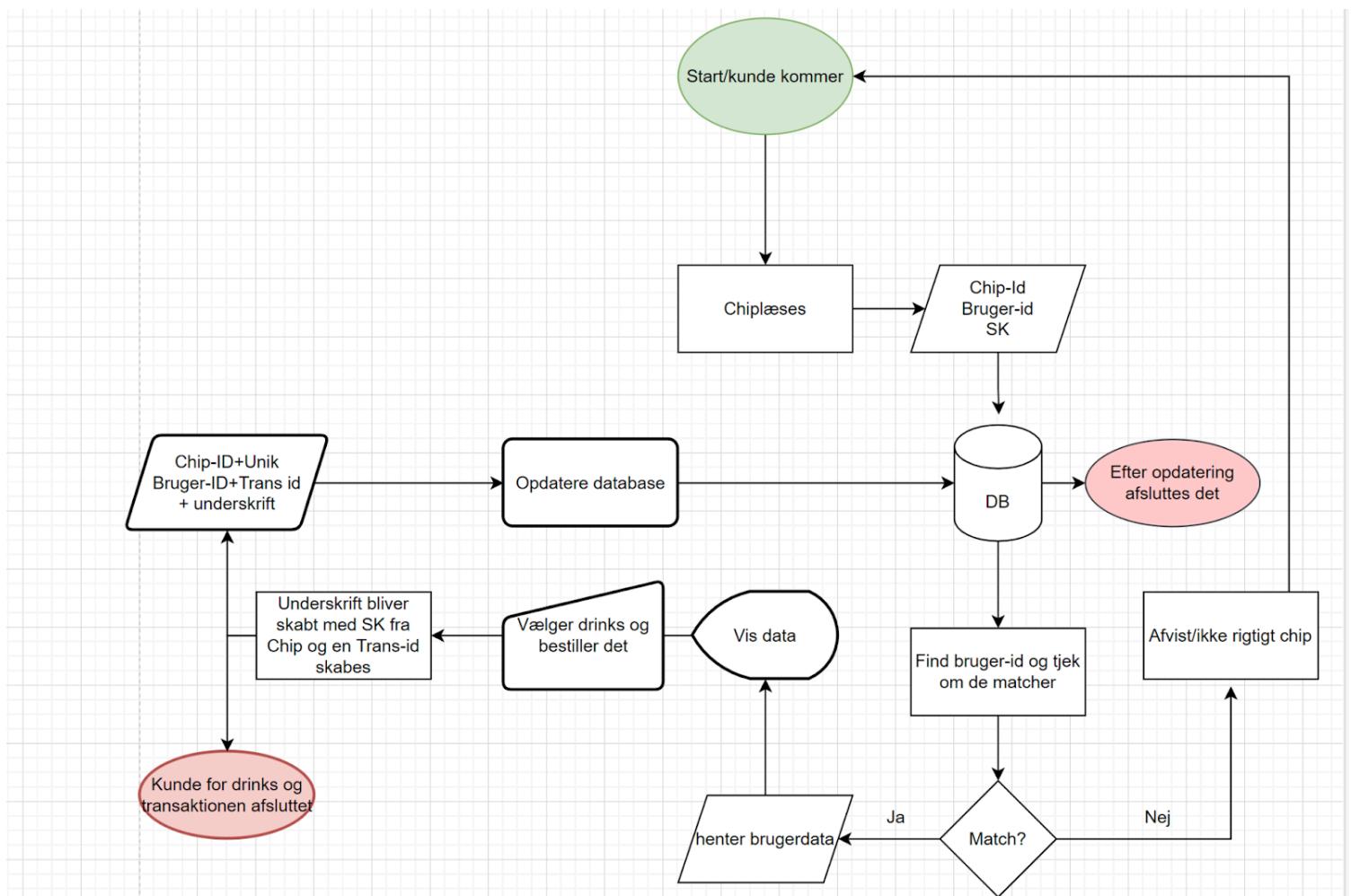
1. **Terminalerne** er de brugerflader som er i selve baren. Her kan man både redigere brugere, og skabe transaktioner; både ind og ud.
2. **Serveren** er det centraliserede system, som gemmer alt brugerdata, og håndterer requests fra terminaler og appen.
3. **Appen**, som kun måske laves i dette projekt, er det brugerstyrede system, hvor man kan redigere sin bruger, og eventuelt også håndtere transaktioner til systemet.



<sup>6</sup> INDSÆT KILDE <https://jyllands-posten.dk/jpaarhus/aarhus/article5801821.ece>

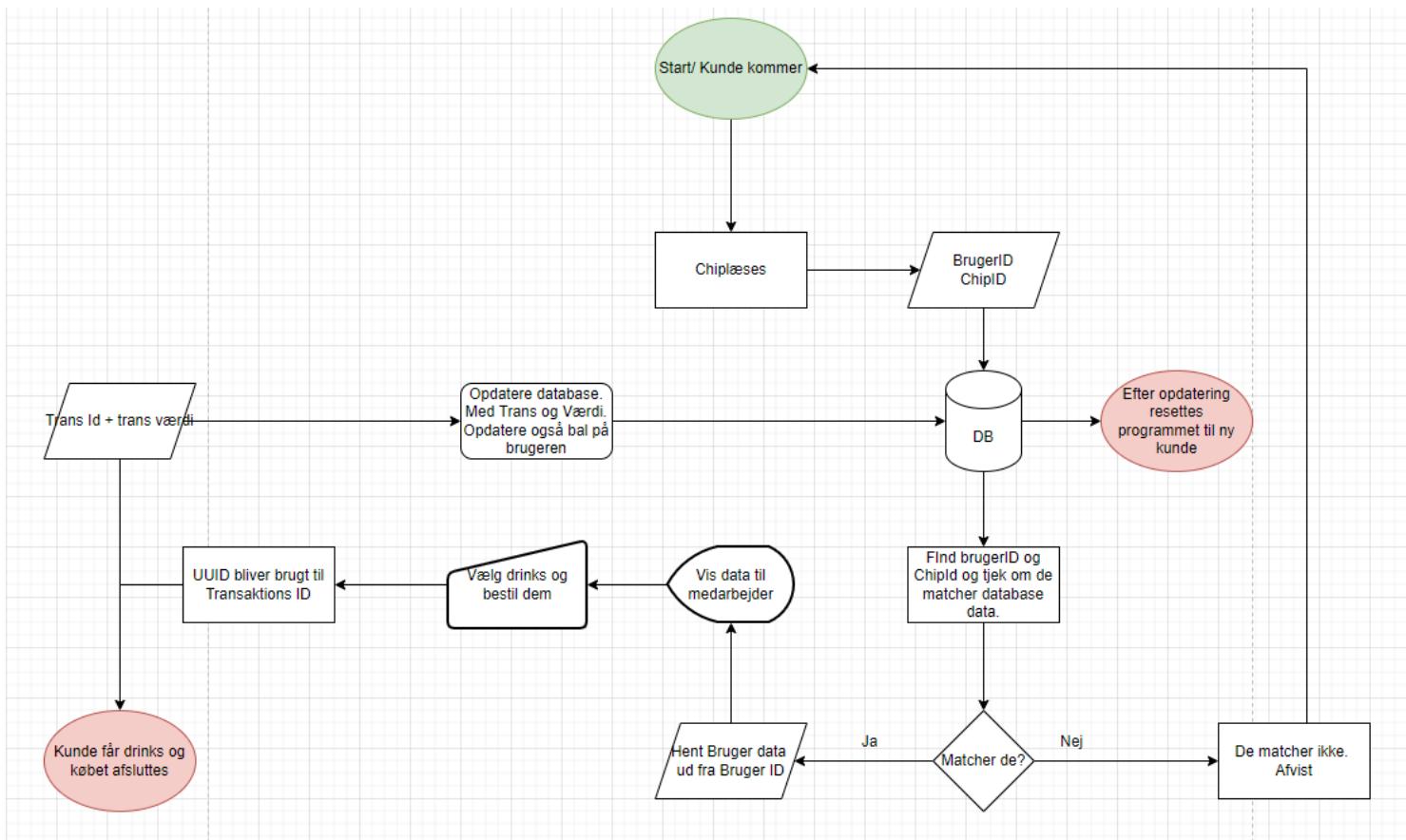
# Flowchart over dataruter

Herunder ses et flowchart over systemets kommunikationsveje. Her ser man de trin der sker under en transaktion med en kunde i baren. Først logger kunden ind i baren, hvorefter deres relevante data vises på terminalen. Herefter vælger bartender en vare, som kunden gerne vil have fra deres udvalg (baseret på alder). Når et valg er taget, opdateres databasen, efter sikkerhedstjek er sket. Der ses to stop-



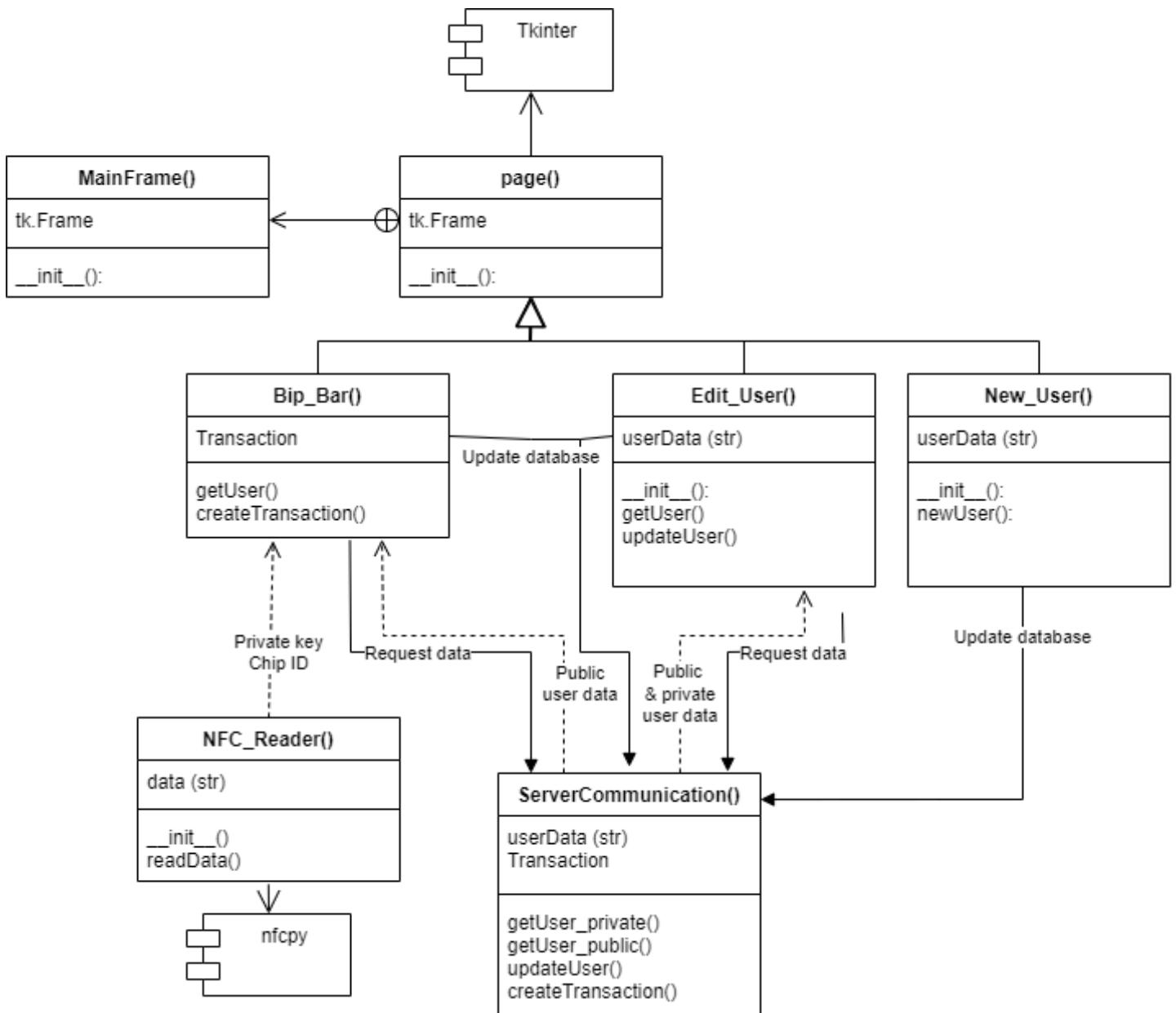
# Opdateret Flowchart over dataruter

Da vi har i løbet af projektet har fundet ud at der er nogle ting der er bedre at bruge; f.eks. at vi bruger websockets til at kommunikere, har vi lavet et opdateret flowchart over dataruter. Vi har også valgt at vi ikke har en public key mere. Vi har kun en secret key, som kun ligger på terminalen. Så når den skal læse vores chip's data så skal det låses op med vores secret key. Med den måde så hvis man prøver at læse noget fra chippen uden den kode kan du ikke. Den er låst og kan ikke unlockes. Vi også tilføjet "bal" som står for Balance. Det hvor mange penge der er tilbage på en bestemt bruger.



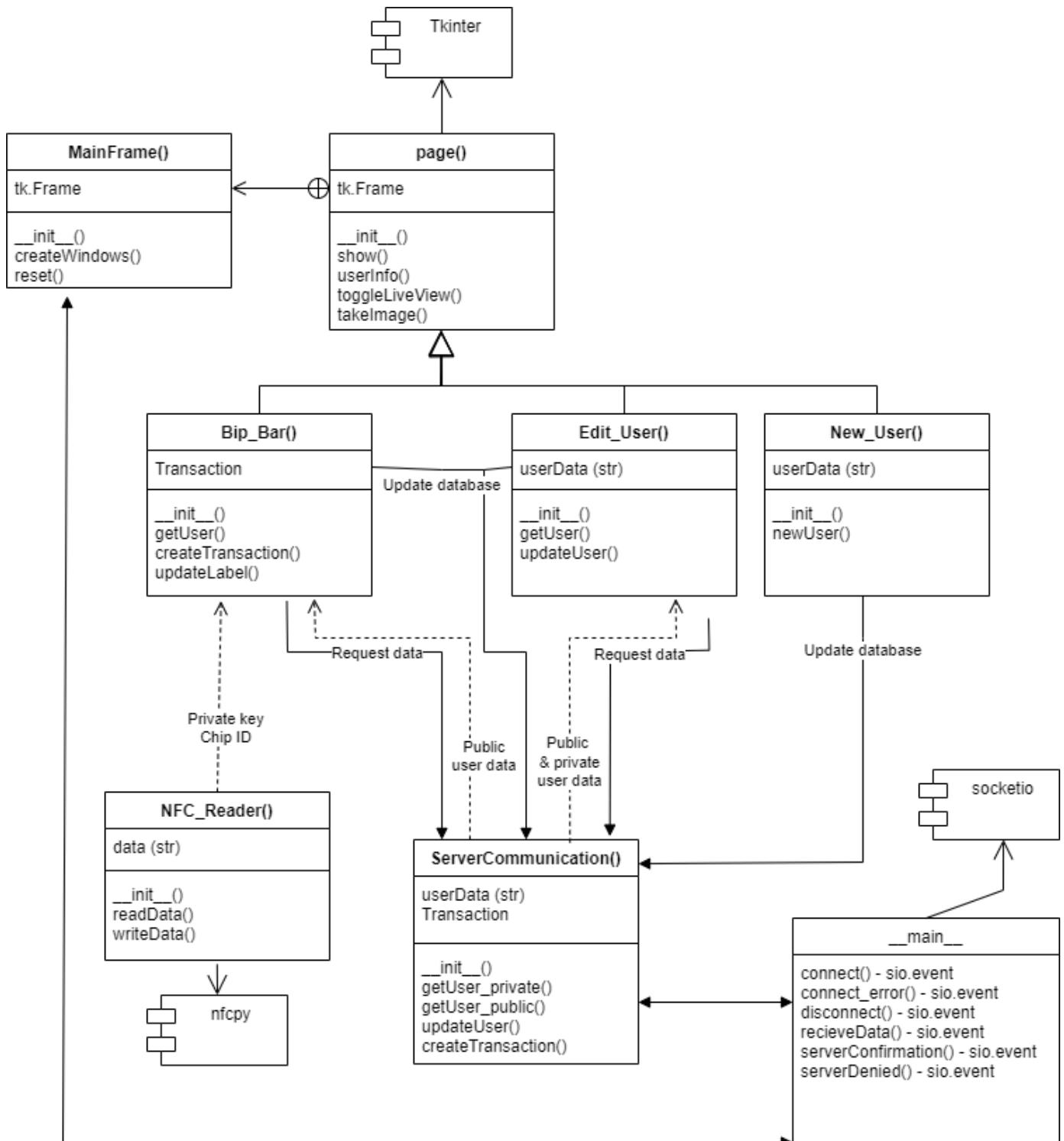
# UML over terminalstruktur

Herunder ses et UML-diagram over terminal-delens struktur, som er fremstillet før at terminalen kodes og rettes for fejl. Den består af to klasser som danner grunden for en GUI (MainFrame og page). MainFrame danner omridset for programmet, og page kan danne sider indenfor MainFrame. Page er en master class, som danner tre child classes. Bip\_Bar, New\_User og Edit\_User. Disse tre danner tre vinduer i GUI'en, hvor man kan benytte sig af transaktioner (Bip\_Bar) og kan redigere/slette brugere i systemet (Edit\_User) samt skabe nye brugere (New\_User). De sidste to klasser bruges til at scanne data fra NFC-chips og til at hente/sende til data til/fra den centrale server. (Der tilføjes en writeData til NFC\_Reader)



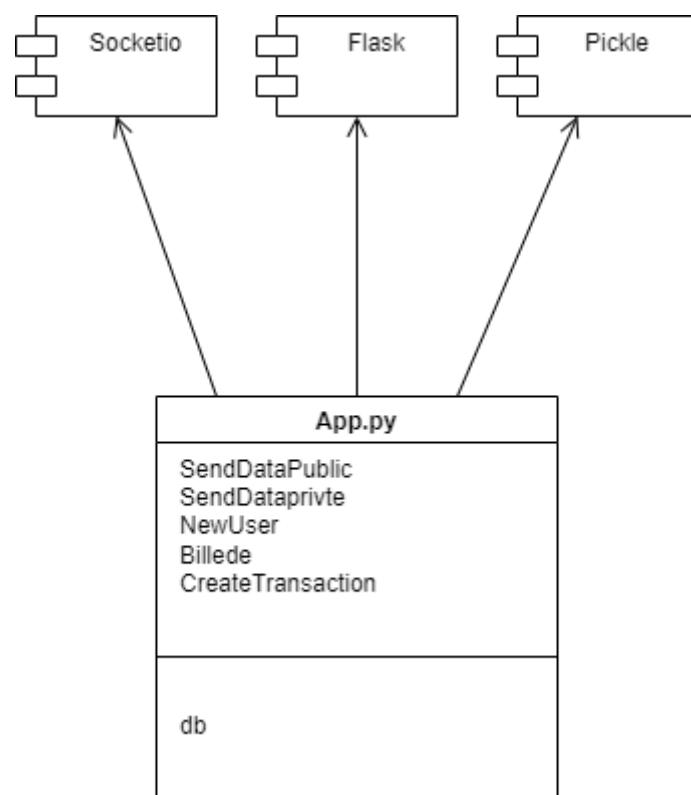
# Opdateret UML over terminalstruktur

Herunder ses en opdateret version af UML-diagrammet over terminalstrukturen. Gennem skabelsen af terminalen, har det været nødvendigt at lave ændringer til strukturen, hvilket er grunden til dette opdaterede diagram.



## UML over server

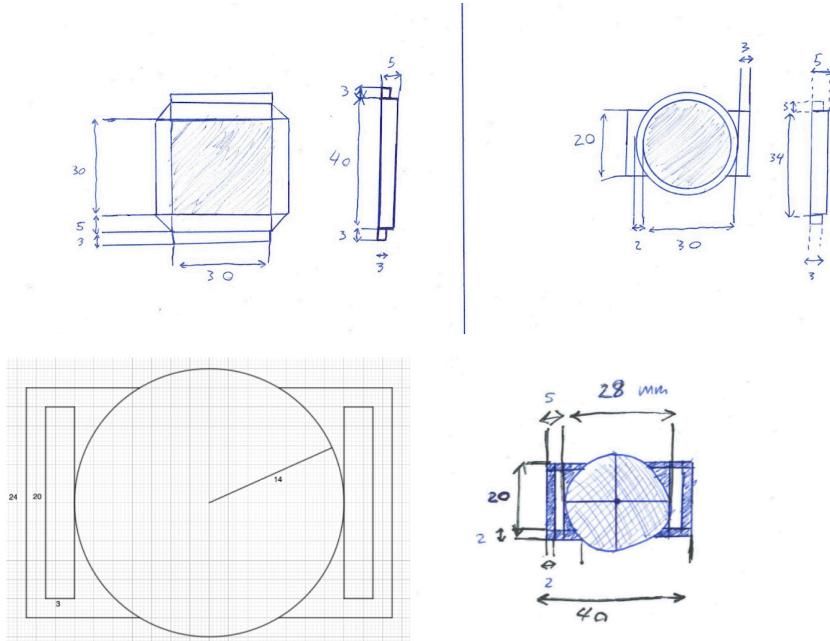
Serveren er meget simple fordi at den kun skal holde på det data den får fra terminalen. Så derfor har den ingen klasse men kun funktioner og moduler der hjælper til at modtage og sende det. Der er socketio som laver en websocket til at terminalen kan tilslutte sig. Derved lave en sikker forbindelse. Flask arbejde sammen med socketio for at lave en port til socketio så det kan tilslutte sig til. Så til at gemme alt vores data så bruger vi pickle. Det gør at når vi gemt vores data. Så ligger det i bytes og ikke i en teskt form så det er lidt ekstra sikkert. Men vi har 3 funktioner til at tage imod data. Det er "NewUser", "Billede" og "Createtransaktion". Det tager imod data og lagere det. Så har vi to funktioner som sender data. Som er "SendDataPublic" og SendDataPrivate". De sender henholdsvis alt data og vigtigt data.



# Prototype 1 - Form

## Skitsering og 3D-modeller af egne armbånd

Der laves en skitse for at udforske de forskellige muligheder produktet kan udformes.

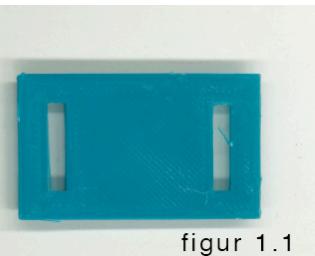
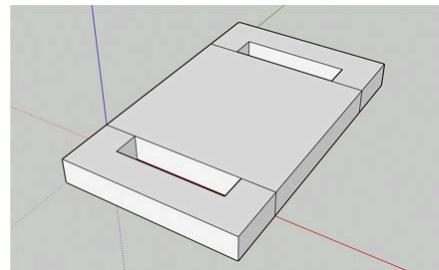
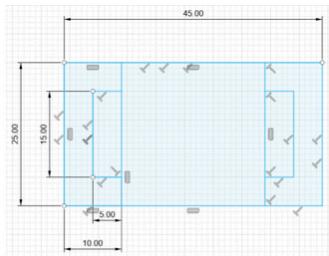


Til at udvikle produktet anvendes framework fra det videnskabelig blad *Anatomy of prototypes*. I den første del af udviklingsfasen anvendes frameworket som en metode til at filtrere og kigge på en lille del af produktet. Denne del af framework er bestående af forskellige *“Filtering Dimensions”* som er de dimensioner, der svarer til de aspekter af en designidé, som en designer forsøger at repræsentere i en prototype. Disse dimensioner er bestående af *appearance, data, functionality, interactivity og spatial structure*. I denne del af udviklingsfasen fokuseres der på *appearance* som er bestående af størrelse; farve; form; vægt; tekstur; mængde; hårdhed; gennemsigtighed; osv. Prototyperne testes dermed ud fra disse parametre.<sup>7</sup>

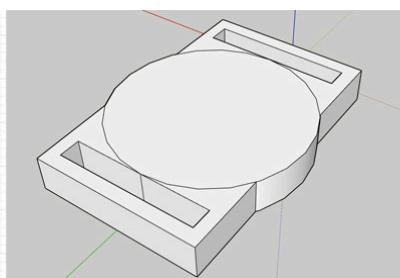
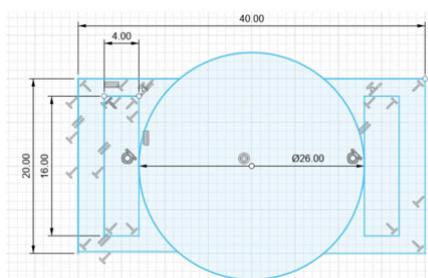
Da prototyperne er ufuldstændige åbner, det muligheden for nye design idéer. Der laves en prototype som tester en lille del af den færdige prototype. Det kendes at NFC chippen har en diameter på  $d = 25 \text{ mm}$  og en højde på  $h = 1 \text{ mm}$ . Der tegnes 2 skitser der tager hensyn til NFC-chippens størrelse. De 2 skitser tegnes i et 3D, hvorefter der printes en 3D model af udformningen. På de næste figurer vises designprocessen, hvor 3D modelleringen og prototyperne vises.

Der laves low-fidelity prototyper både plastik- og computerbaseret. Dette åbner muligheden for at teste de opstillede krav fra design-space. Først laves der en 2D-model i et computerprogram, hvor der hurtigt kan blive ændret på prototypens mål. Derefter udvides det til en 3D-prototype, der skal give en visualisering af modellen. 3D-prototypen fremstilles herefter i plastik, for at teste om det muligt at producere den.

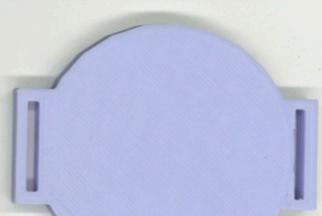
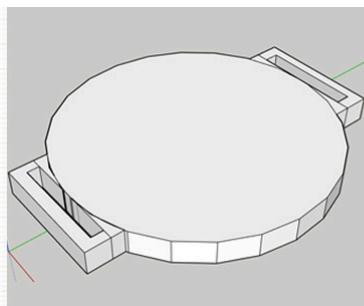
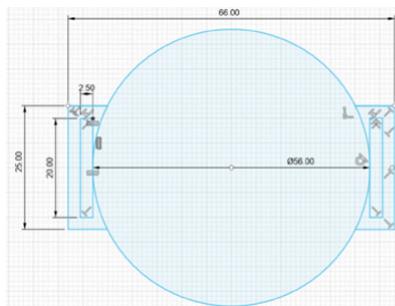
<sup>7</sup> (Lim and Stolterman; 7:11)



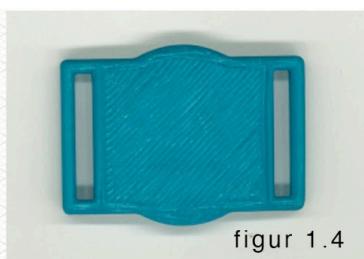
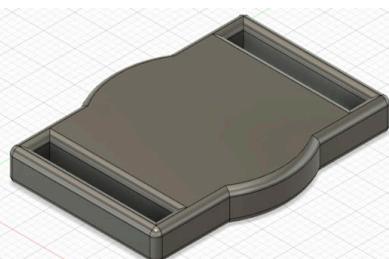
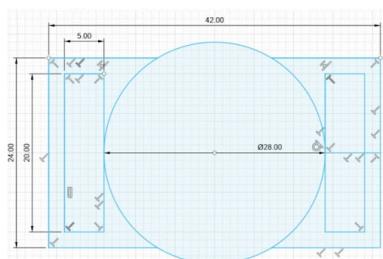
figur 1.1



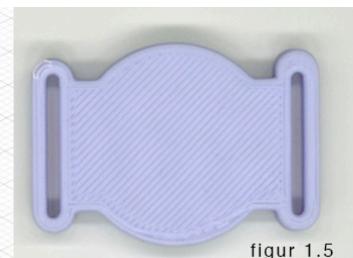
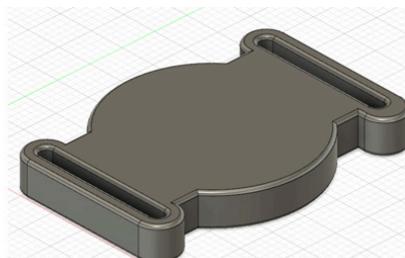
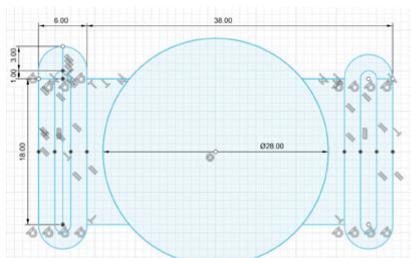
figur 1.2



figur 1.3



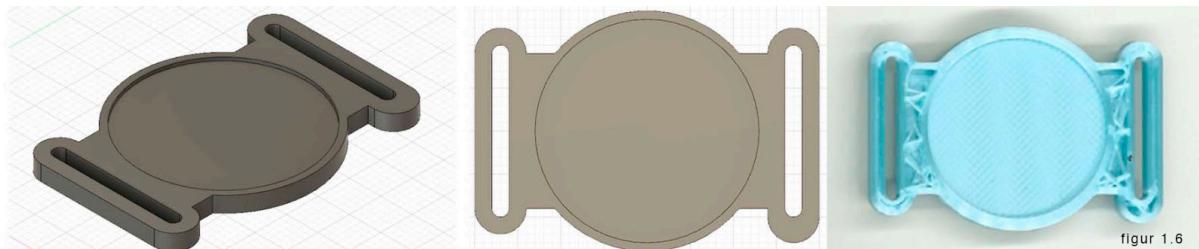
figur 1.4



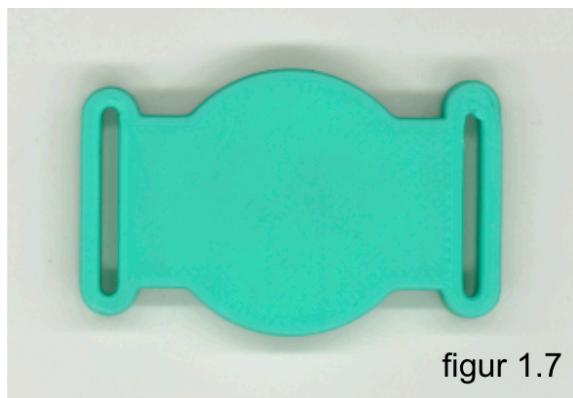
figur 1.5

Efter udviklingen af adskillige prototyper, er der kommet frem til en form med bløde hjørner og runde hængsler giver det mest ergonomiske design. Den er printet med 20% infill da det blot er en prototype, og har dermed et meget rå finish. Denne prototype er vist på figur 1.5.

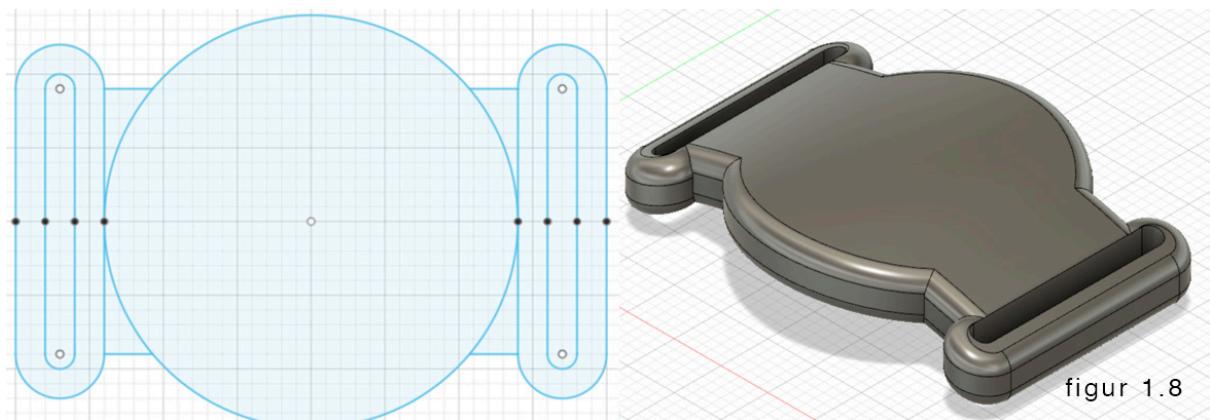
For at kunne lave en funktionel prototype skal der inkorporeres en NFC-chip. Da chippens størrelse kendes som  $d = 25 \text{ mm}$  og  $h = 1 \text{ mm}$  laves der hermed en udhulning i midten af modellen. På figur 1.6 ser man det indvendige design af modellen.



Ved fremstillingen indstilles 3D printeren til at pause halvvejs, hvorefter NFC-chippen indsættes og printet fortsætter. På denne måde forsegles chippen, så den bliver en fast del af prototypen. Den printes med 100% infill for maksimal styrke. På figur 1.7 vises den færdige form med NFC-chip

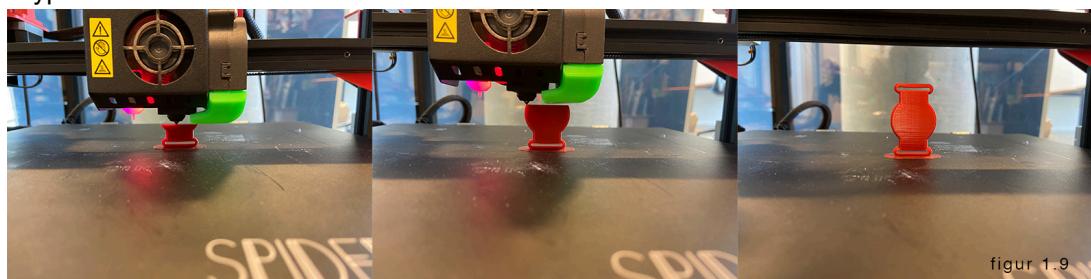


Med udgangspunkt i prototypen på figur 1.7, sættes der nu fokus på produktionen. Frameworket fra Anatomy of prototypes anvendes derfor igen, hvor prototyperne bruges som en metode til at filtrerer og fokusere på en lille del af projektet. Derefter undersøges brugerens oplevelse med prototypen. På figur 1.8 vises arbejdstegning og den computerbaseret prototype.



På grund af prototypens buer skaber det en række udfordringer i forhold til produktionen. Da denne prototype ikke kan printes med den brede flade nederst. Til dette printes modellen fra forskellige sider, for at finde den mest optimale.

Prototype 1.9



figur 1.9

**Beskrivelse**

Modellen printes lodret, hvor der tilføjes en flade(brim) for at stabilisere printet. Der tilføjes ikke support, da det sætter sig fast i hullerne, og ikke kan fjernes.

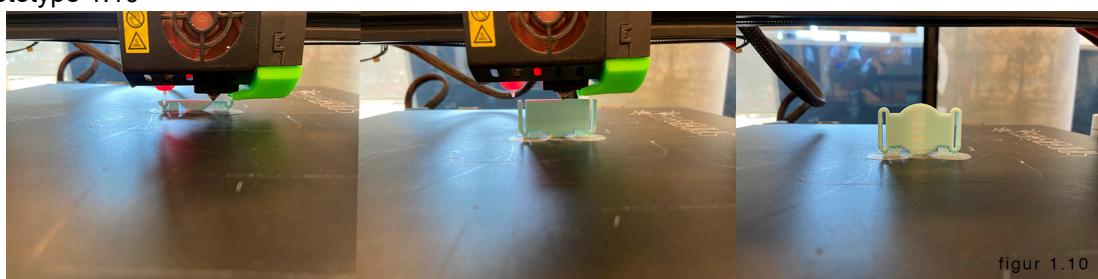
**Fordele**

- Glat overflade
- lange plastik lag, der styrker strukturen

**Ulemper**

- Svær at printe
- Mange mislykket forsøg
- Svær at inkorporerer NFC-chip

Prototype 1.10



figur 1.10

**Beskrivelse**

Modellen printes fra siden, og der tilføjes support under strukturen, da det nemt kan pilles af.

**Fordele**

- Nem at printe
- Glat overflade

**Ulemper**

- Knækker nemt ved hængslerne
- Skrøbelig
- Svær at indsætte NFC-chip

Prototype 1.12



figur 1.12

**Beskrivelse**

Her printes modellen fra bunden og op, hvor der indsættes support og en flade. Modellen efterbehandles ved at rense og skære den til

**Fordele**

- Nem at printe
- Nem at indsætte NFC chip
- Stærk struktur

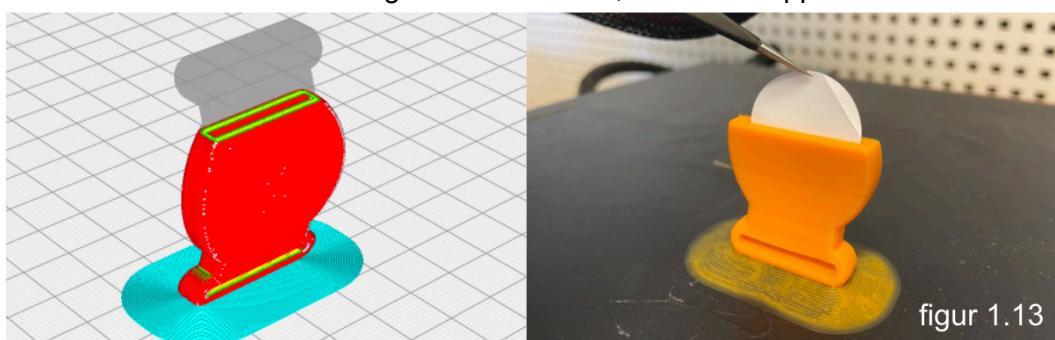
**Ulemper**

- Træls overflade

Til at undersøge hvilken printerindstilling der er bedst, inddrages brugeren til at hjælpe med at vurdere dem. Der opstilles et spørgeskema, hvor brugeren vurderer finish, behagelighed og udseende. Brugerne vurderer det på en pointskala fra 1-10, hvorefter summen af disse værdier beregnes. Det fulde spørgeskema kan findes i bilag(#).

Tabel 1.14	Finish	Behagelighed	udseende
Prototype 1.9	84	78	72
Prototype 1.10	45	55	38
Prototype 1.12	33	27	42

Det kan derfor vurderes at prototype 1.9 er den mest populære. Derfor vælges det at udvikle videre med den model. Med den nuværende størrelse af prototypen er det ikke muligt at indsætte en NFC-chip med den lodrette printer vinkel. Derfor skaleres modellen til 120% for at gøre det nemmere at indsætte chippen. Derefter programmeres printet til at stoppe, hvor chippen indsættes i modellen. På figur 1.13 vises det, hvordan chippen indsættes.

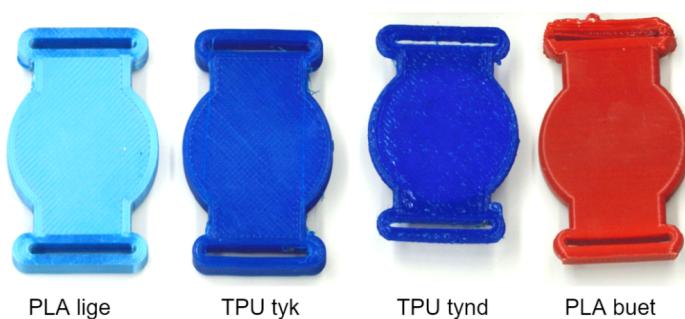


figur 1.13



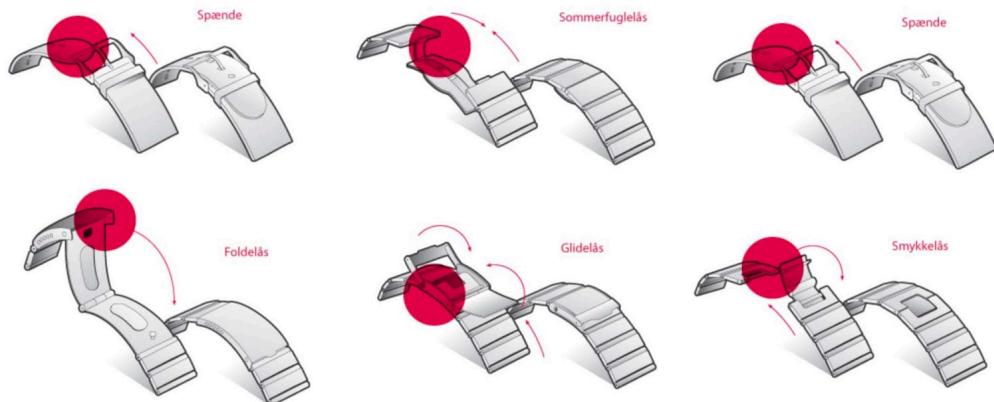
På figuren herover ses de forskellige prototyper ved siden af hinanden.

### Færdige Chip-designs



# Design af rem til egne armbånd

I dette afsnit kigges der på en anden del af den færdige prototype. Der tages udgangspunkt i den spændemekanisme, der skal bruges til at fastgøre armbåndet til brugerens. Til dette laves der en forlængelse af designspace, en undersøgelse af de spændemekanismer, der allerede findes på markedet.



Låse mekanismerne opdeles i 5 forskellige kategorier.

## Sommerfuglelås

Denne lås lukkes og åbnes som en sommerfugl og bruges typisk i dyre ure. Låsen kan bruges med metal og læderrem.

## Spænde

Denne type lås er en populær type lås, der bruges i mange ure. Den åbnes og spændes som et bælte, og er dermed nem og simpel at bruge. Spænde låsen benyttes typisk i billige ure, da den er nem at installere.

## Foldelås

Hvis der kigges på figuren fungerer låsemekanismen som en låge, der strammes når den foldes. Der findes forskellig varianter, hvor nogle af dem indeholder justerbare huller. Derudover holder indeholder nogle af en mekanisme der fastholder låsen, så den ikke åbner sig.

## Glidelås

Glidelåsen er en meget almindelig lås der bruges på de fleste ure. Denne lås hænger glider rundt på lænken, og kan dermed tilpasses til, hvor stram man gerne vil have. Låsen benyttes ofte på billige ure, da den er simpel og nem.

## Smykkelås

Denne lås er simpel og låses ved at klikke den fast. Den bliver dog kun udelukkende brugt på dameure.<sup>8</sup>

## Udvælgelse

På baggrund af de forskellige låse muligheder, er det blevet besluttet at bruge arbejde videre med spænde låsen, da den nem at montere, og bruger har muligheden for at justere størrelsen efter behov. Dog udvikles der også en række andre designs i næste afsnit.

<sup>8</sup> (Anytime)

## Armbånd-designs

Da spænde mekanismerne kendes, laves der en række forskellige designs. Herefter inddrages brugeren til at vurdere, hvilken type, som er mest komfortable. Der anskaffes også armbånd, der allerede eksisterer på markedet, til at sammenligne med de armbånd der testes. Der laves forskellige varianter, da det åbner muligheden tabel 1.15 vises de forskellige varianter.

<b>Tabel 1.15</b>			
<b>Armbånd</b>	<b>Materiale</b>	<b>Brugstype</b>	<b>Antal</b>
Plastarmbånd	blød PVC plastik	Engangsbrug	10 stk
Papirarmbånd	Tyvek® polyetylen	Engangsbrug	10 stk
Stofarmbånd (Bæredygtig)	Genbrugt PET polyester	Engangsbrug	5 stk
Silikonearmbånd	latex-fri gummi	Genanvendelig	10 stk
Stof - Klikspænde	BR-nøglering og klikspænde	Genanvendelig	1 stk
Stof - Spænde	BR-nøglering og urspænde	Genanvendelig	1 stk
Læder - Spænde	Læder og urspænde	Genanvendelig	1 stk
Stof - Velcro	BR-nøglering og velcro	Genanvendelig	1 stk
Stof - kliklås	BR-nøglering	Genanvendelig	1 stk



Figur 1.16

På figur 1.16 vises de forskellige låsemekanismer, der er blevet produceret.

# Prototype 2 - Funktionalitet

## NFC-reader

En af grundstenene i dette projekt, er evnen til at kunne læse brugerdata fra NFC-chips. For at skabe familiaritet med hvordan dette gøres i Python, laves et simpelt program, som kan læse den data som er på en NFC-chip gennem et library ved navn nfcpy. Dette program kaldes “**NFC-reader.py**”.

De elementer som bruges i dette testprogram, kan også bruges i Terminal-programmet som udarbejdes senere.

Koden kan ses afleveret på It's Learning under det nævnte navn; “**NFC-reader.py**”

## Video med funktionalitet

Herunder ses en video, som viser programmets funktionalitet. Man ser hvordan NFC-readeren automatisk er blevet registreret af programmet, og er klar til at læse fra chips. Herefter scannes en NFC-chip, og man får følgende output i terminalen, med informationer om chippen:

```
Type2Tag 'NXP NTAG215' ID=0481D601B64003
NDEF Text Record ID " Text 'Hello World' Language 'en' Encoding 'UTF-8'
NDEF Text Record ID " Text 'I am an NFC-chip' Language 'en' Encoding 'UTF-8'
```

Man ser først generel information om chippen, og derefter vises indholdet af to text-records, deres tekst og deres encoding (UTF-8).

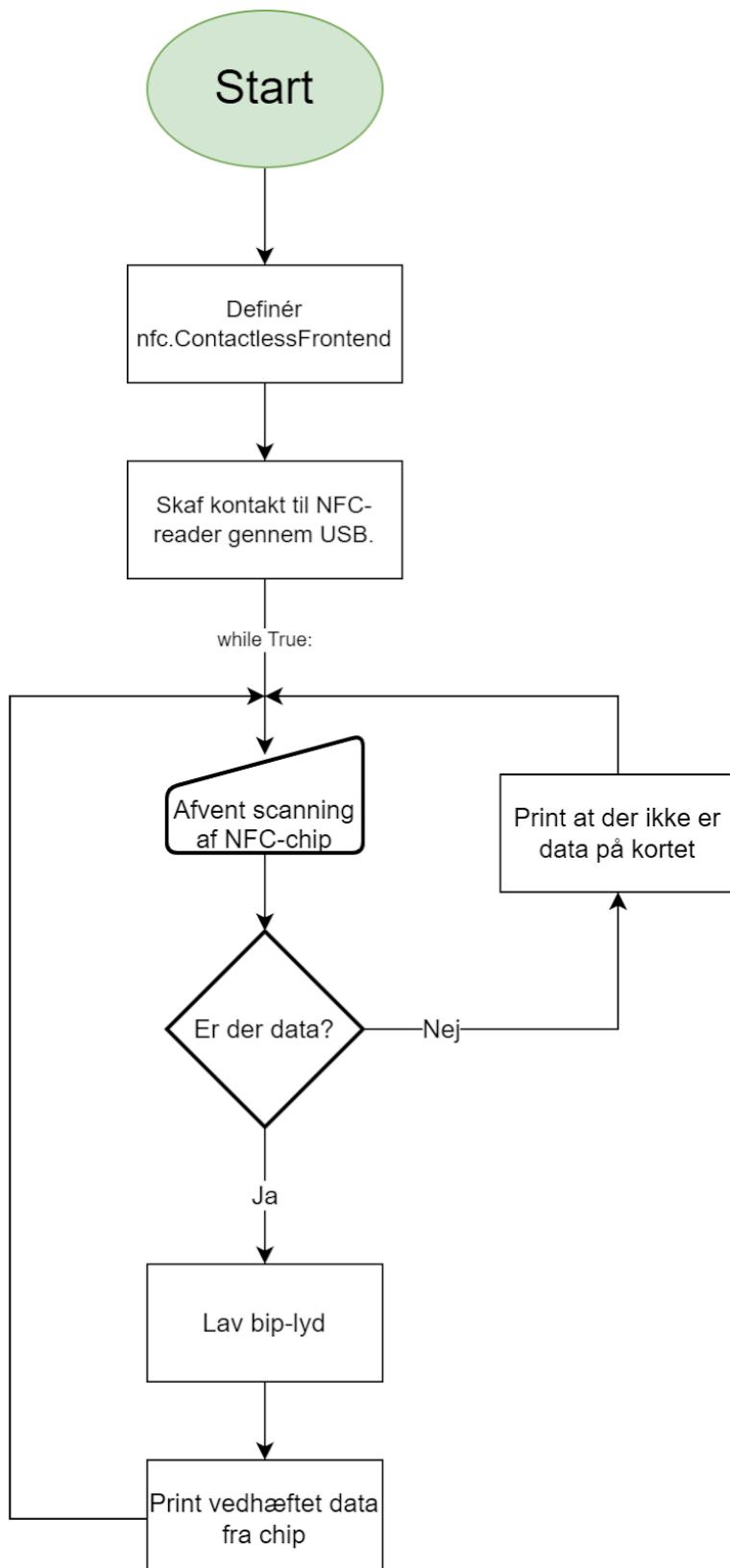
Videoen kan ses herunder:<sup>9</sup>



<sup>9</sup> [https://drive.google.com/file/d/1zFejQzp65DUuC\\_QHD-As5kuDzFj\\_ww5o/view?usp=sharing](https://drive.google.com/file/d/1zFejQzp65DUuC_QHD-As5kuDzFj_ww5o/view?usp=sharing)

## Flowchart

Herunder ses et flowchart over programmets gang:

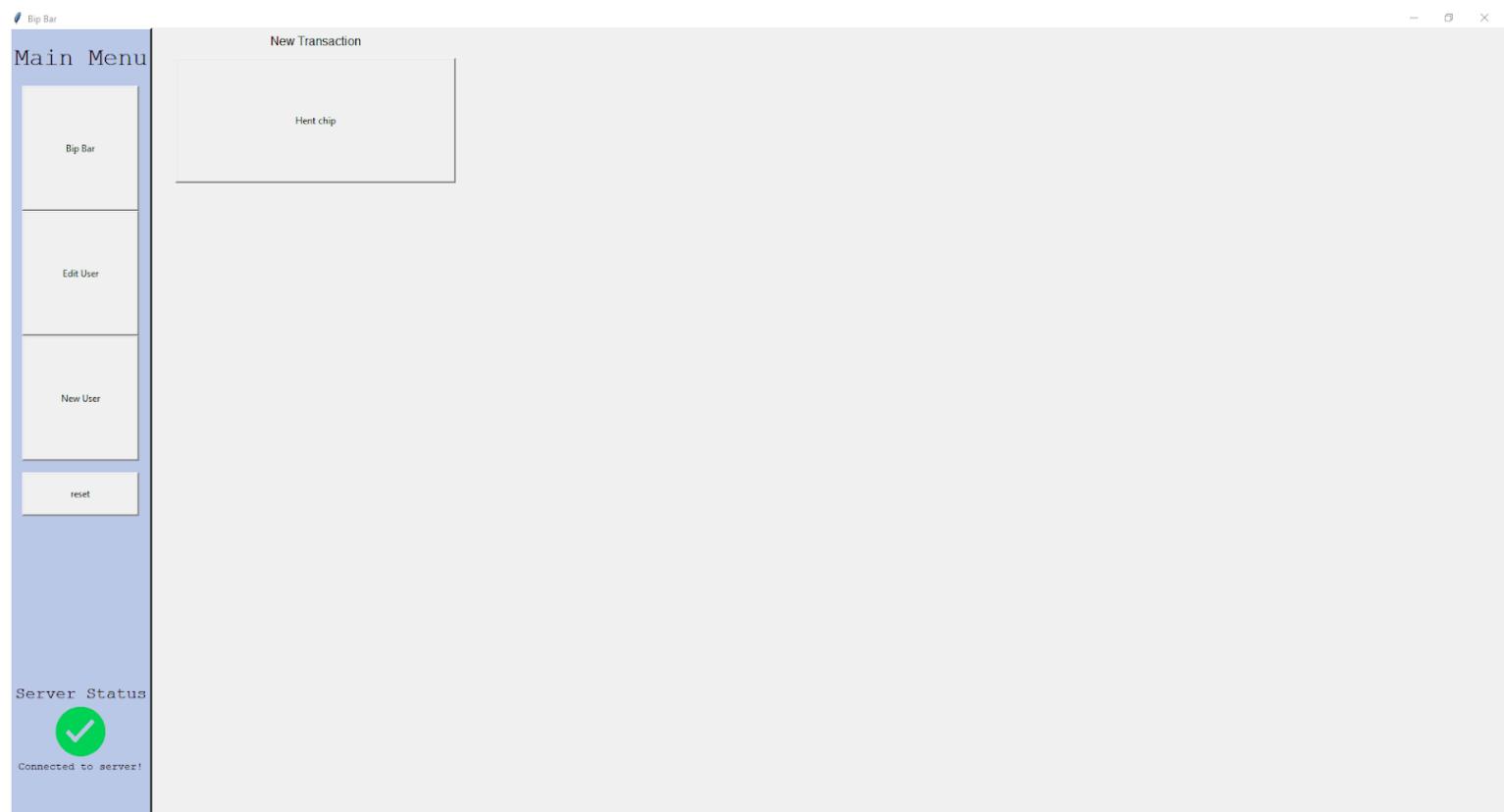


# Terminal

## Introduktion til terminal

Vi har tænkt at lave en terminal i python via tkinter. Strukturen for denne terminal kan ses på UML-diagrammet ved "UML over terminalstruktur". Terminalen udgør brugerfladen, hvor transaktioner og brugere skabes og redigeres. Vi har tre forskellige sider i vores Mainframe. Den første, "Bip Bar" er der hvor man primært vil bruge terminalen. På siden er selve barsystemet, hvor man skaber nye transaktioner til brugere. Den anden side er Edit user. Her kan man scanne en NFC-chip, og redigere de brugeroplysninger som er tilknyttet chippen. Den sidste side er New User. Her skaber man nye brugere, og tilknytter en NFC-chip til brugeren. Vi har en knap til hver eneste page, som bruges til at navigere igennem terminalen. Vi har også en ekstra knap, som bliver brugt til at resette siderne i programmet, hvis man nu kommer til at trykke forkert og man ikke kan gå tilbage. Under navigationsknapperne ses et område til "Server Status". Dette giver relevant information til brugeren om serverens kommunikation til terminalen.

Herunder ses et screenshot af terminalen, når man kommer ind i den:

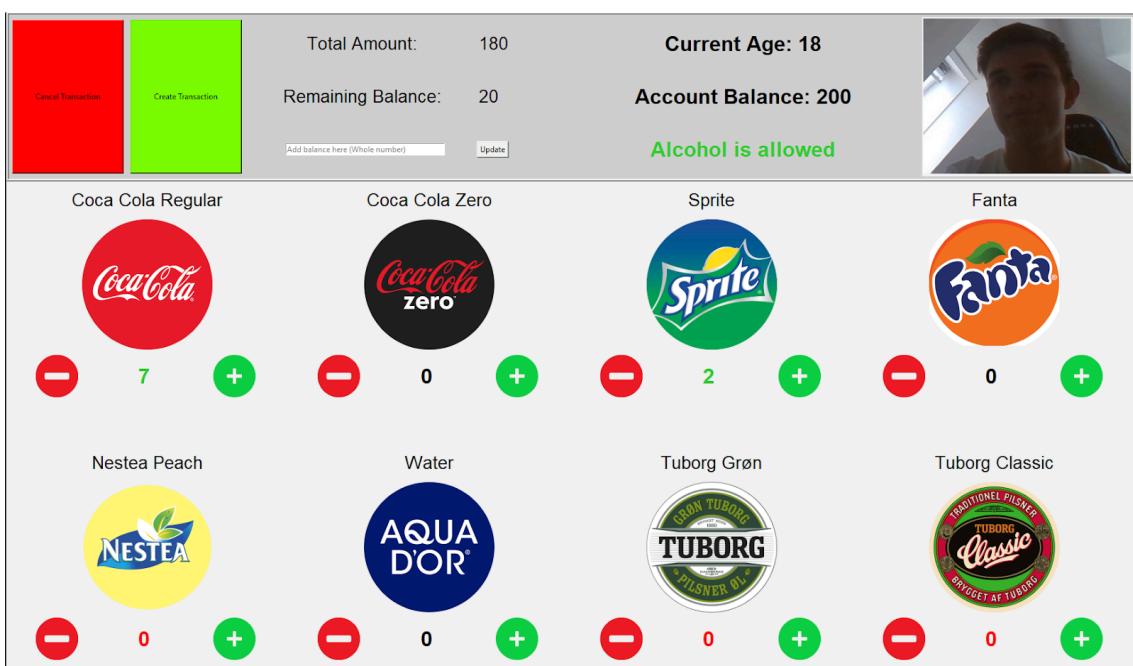


## Bip Bar-siden

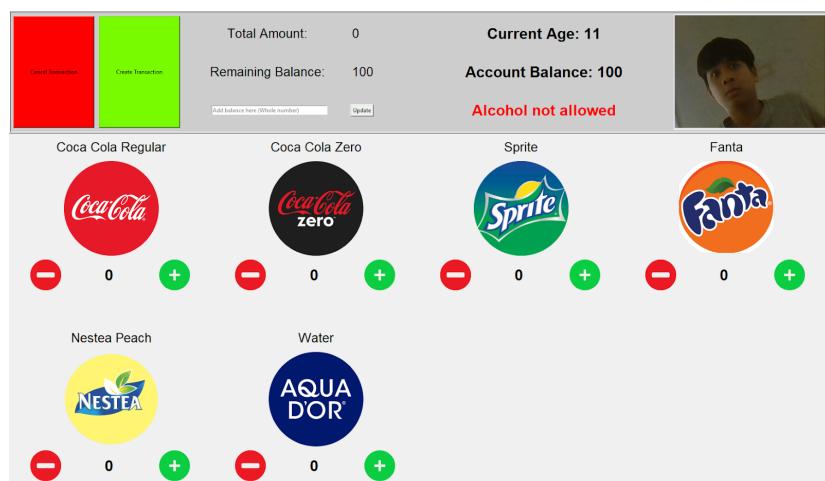
Trykker man på "Bip Bar"-knappen, kommer man ind på siden hvor man skaber transaktioner.

Øverst er knapper til at annullere eller fuldføre transaktioner. Derudover er der opdaterede tal for købets beløb, samt tilbageværende kredit på kontoen efter et eventuelt køb. Til højre for det ses informationer om alder, brugerens kredit før købet og hvorvidt det er tilladt at brugeren køber alkohol. Der er også et billede af brugeren til identifikation.

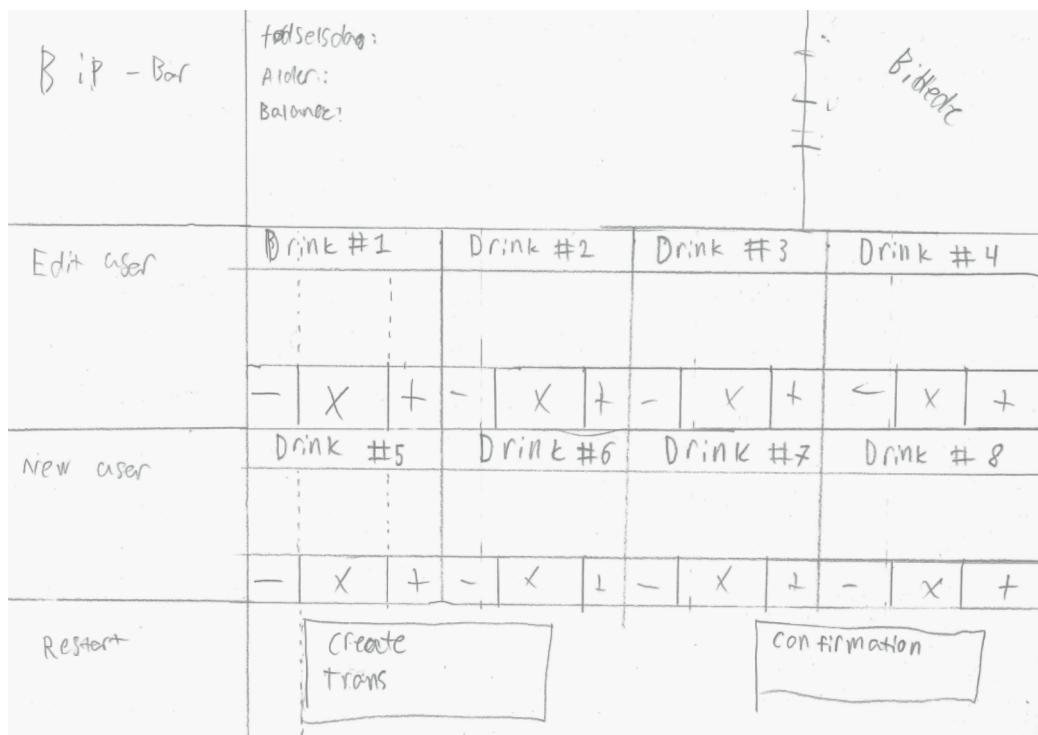
Herunder ses udvalget af drikkevarer som kan købes. Alle disse drikkevarer har forskellige priser, som er angivet i koden. Priserne for Cola Regular, Cola Zero, Sprite og Fanta er 20, Nestea er 25, vand er 10 og Tuborg Grøn samt Tuborg Classic er 30. Hvis brugerens tilbageværende kredit ikke er nok til at købe en ting på menuen, bliver tallet med antal rødt, for at vise at den ikke længere kan vælges. Skulle man prøve alligevel, vil programmet ikke tillade det, og antallet forbliver 0. Når en vare vælges, bliver antallet grønt, sådan at det er nemt at identificere hvor man har trykket. Alle de beskrevne ting kan ses herunder:



Det bør også noteres at varer med alkohol ikke vises hvis brugeren er under 18. Dette ses for eksempel herunder:



Transaktionsvinduet er lavet i et grid-format. Herunder ses den plan som var lavet for det grid:



Forskelligt fra planen, er transaktionsknapperne rykket op i toppen, da det gav mere mening. Dog er griddet nogenlunde det samme. Herunder er det indtegnet et grid ovenpå transaktionsvinduet, for at man kan se hvordan det endelige program er opsat:

		Total Amount:	180	Current Age: 18			
		Remaining Balance:	20	Account Balance: 200			
		Add balance here (Whole number)		Update	Alcohol is allowed		
Nestea Peach		Water		Tuborg Grøn		Tuborg Classic	

## Edit User-siden

På “Edit User”-siden skal man først scanne sin NFC-chip. Herefter vil alle brugerinformationer blive hentet ned til terminalen fra serveren, og de vil indsættes i samme tekstbokse, som når man laver en ny bruger. Man har nu muligheden for at redigere oplysningerne, tilknytte et nyt billede, samt tilknytte en ny NFC-chip. Dette gør man når man trykker “Opdatér bruger og scan NFC”. Her skal kunden igen scanne sin chip, og serveren opdateres.

Edit User Information

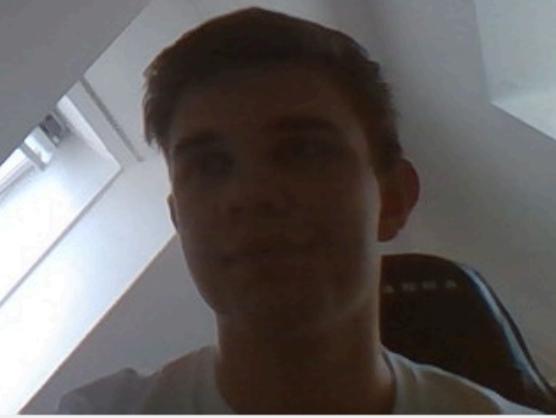
Fuld navn

E-mail adresse

Adresse

Fødselsdato

Billede til identifikation



## New User-siden

På "New User"-siden kan man oprette en ny bruger. Her skal man udfylde boksene, tage et billede af brugeren, ved at trykke på knappen, og til sidst trykke "Opret ny bruger og scan NFC". Her skal man scanne kundens armbånd, som nu er tilknyttet profilen.

Register New User

Fuldtnavn

Opret ny bruger og scan NFC

E-mail adresse

Adresse

Fødselsdato

dag  
månedstal (01-12)  
år

Billede til identifikation

Tag billede



## Server Status

For at give brugeren af terminalen en idé om, om kommunikation mellem server og terminal forløber rigtigt, sender serveren signaler og beskeder til terminalen, som vises i bunden af hovedmenuen. Der er ikoner til succes, fejl og når man skal afvente server-svar. De tre typer beskeder kan for eksempel se ud som herunder:



Følgende succes-/fejl-beskeder kan vises:

### Succeskoder:

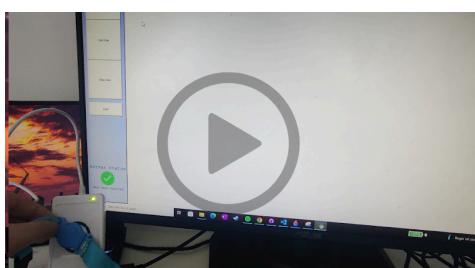
```
"I'm connected!"  
"Transaction received!"  
"Transaction received, but balance does not match list  
of transactions!"  
"User Updated!"  
"New user created!"
```

### Fejlkoder:

```
"The connection to\n server failed!"  
"Server Disconnected!"  
"Transaction failed\n for unknown reasons!"  
"FALSE CHIP-ID\n Public"  
"FALSE USER-ID\n Public"  
"FALSE CHIP-ID\n Private"  
"FALSE USER-ID\n Private"  
"User edit failed!"  
"New user failed!"
```

Video af skabelse af bruger, redigering, samt første transaktioner:

Vi har fremstillet en video, hvor man kan se terminalen blive brugt. Den kan ses herunder.<sup>10</sup>



<sup>10</sup> [https://drive.google.com/file/d/1FiS060Ri5fLoeF6\\_xVT3ESuqnnmNjHNG/view?usp=sharing](https://drive.google.com/file/d/1FiS060Ri5fLoeF6_xVT3ESuqnnmNjHNG/view?usp=sharing)

# Database

Vi har tænkt os at lave databasen på den måde, at der er en stor dictionary, som gemmes gennem pickle-biblioteket. Hver bruger i databasen tildeles en key, hvis value indeholder data for brugeren. Brugerens data gemmes i database-dictionariet med en form, som ser sådan her ud for en given bruger:

```
{UserID : {"name" : navn,
            "email" : email,
            "adress" : adresse,
            "birthday" : dd-mm-yyyy,
            "chipID" : chipID,
            "balance" : intBalance,
            "transactions" : {TransIDs... : intAmounts...} }}
```

Altså bruges UserID fra chippen som key til hele brugerens data. I "transactions" er der endnu en dictionary, som består af alle de transaktioner der er lavet, med ID og mængden af kreditændring.

Der er også et billede til hver bruger, som bliver modtaget som rå data til en .jpg-fil fra terminalen, når man opretter brugen. Så derefter bliver det sendt til serveren og den gemmer det i en mappe som gør at det er nemme at finde. Men billedet hedder hvad UserID det tilhøre. Så når vi skal have et billede til terminalen så skal vi bare finde det billede der passer til det UserID.

Følgende funktioner kan kaldes fra terminal (og website ved @app.route) til server:

```
@app.route("/")
def hello_world():

@socketio.on('PublicData')
def SendDataPublic(data):

@socketio.on("PrivateData")
def SendDataPrivate(data):

@socketio.on("NewUser")
def NewUser(data):

@socketio.on("Billed")
def Billed(data):

@socketio.on("Trans")
def CreateTransaction(data):
```

## Prototype 3 - Interaktion (ventes til at terminalen er færdig)

Denne "prototype" går ud på at teste de interaktioner som brugerne af systemet laver. Det gælder primært for den kunde som betaler gennem Bip-Bar. Derfor skal det undersøges hvordan man bedst betaler med et armbånd på en NFC-reader.

Til at teste hvordan brugeren vil interagere med produktet, anvendes forrige framework fra Anatomy of Prototypes. Her bruges det til at fokusere på en mindre del af den samlede prototype. Da prototypen anvendes til at "filtrere" dele af produktet, behøves der ikke at tage højde for at funktionalitet og udseende. Dette åbner muligheden for at teste interaktionen før terminalen færdiggøres. Der laves en low-fidelity prototype af papir og pap, der skal visualisere den betalingsterminal, som brugeren scanner chippen ved. Prototypen har en simpel kasse form, med 3 forskellige scanningsmuligheder: top, venstre og højre side. Der kan nu testes, hvilken side der mest behagelig for brugeren at scanne. Figur 3.1 viser den prototype der bruges til at teste interaktionen.



figur 3.1

Test resultaterne vises i tabel 3.2. Forsøgsdeltagerne blev også spurgt om deres begrundelse for deres valg, hvor der var bred enighed, at toppen føltes mest naturlig og dernæst den modsatte side af den hånd armbåndet sad på. F.eks. hvis armbåndet lå på højre hånd var toppen og venstre side mest naturlig at scanne. Derudover blev der også foreslået, at scanneren ville være mere behagelig med en lille hældning.

Tabel 3.2

Venstre	Top	Højre
4	16	1

Ud fra den data der blev opsamlet i forsøget, skabes der nu en prototype ud fra det. På figur 3.3 vises den prototype der blev fremstillet.

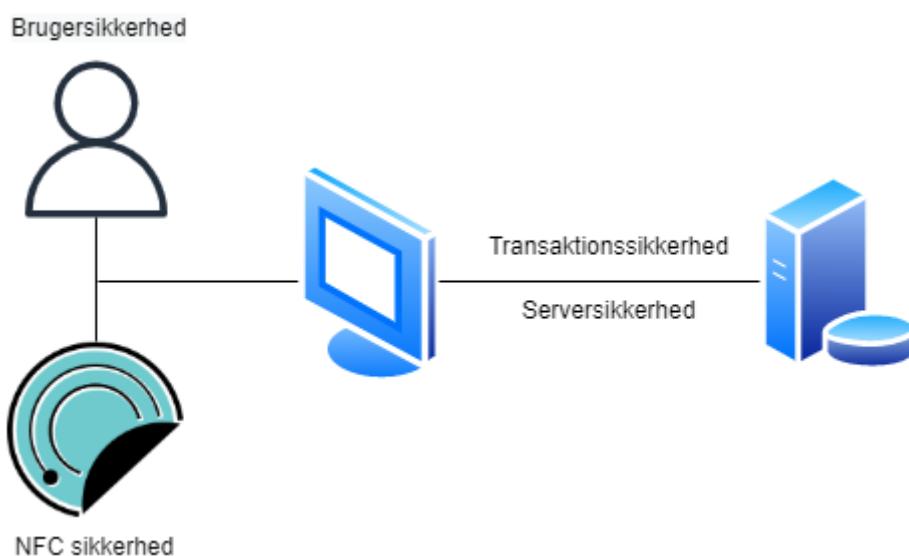


# Prototype 4 - Sikkerhed

## Systemsikkerhed

### Overblik

Der er en række områder som skal sikres i et sådant betalingssystem, som vi har lavet. De områder som vi har taget stilling til, kan beskrives som Brugersikkerhed, NFC sikkerhed, Serversikkerhed og Transaktionssikkerhed. De forskellige områder, og de sikkerhedstiltag vi har taget, beskrives i denne sektion.



### Brugersikkerhed

Med brugersikkerhed, menes der den sikkerhed og tillid som brugerne har til betalingssystemet. Vi har valgt at bruge **billede-identifikation** af brugerne, som den som betjener terminalen kan se hver gang man laver en transaktion. Dermed kan man afvise eventuelt stjålne armbånd.

Derudover har vi lavet **dybdegående tests af sikkerhed på armbåndene**, som gør at de er svære at tage af brugerens håndled, og dermed svære at stjæle.

### NFC sikkerhed

Da det brugerID der findes på chippen, i **kombination med chippens ID**, bruges som sikkerhed før transaktioner kan laves, er det vigtigt at NFC-chippen ikke er læselig for fremmede parter. Dette har vi løst ved at lave **password-protection** på vores chips, med et **128-bit UUID kodeord** som bruges til alle chips. Det betyder at man kun kan bryde sikkerheden i systemet, ved at aflæse kodeordet ved skabelsen af en ny bruger, samt at skabe en falsk NFC-chip med samme chip-id som en eventuel stjålen chip.

Sikkerheden kunne forbedres ved at skabe unikke kodeord til alle chips, og gemme dem i databasen, men vi har vurderet at andre punkter var vigtigere indtil videre.

## Serversikkerhed

For at fremmede entiteter ikke kan læse den data som sendes frem og tilbage mellem server og terminal benyttes **sockets**. Disse skaber en krypteret forbindelse mellem server og enhver som connecter til den server.

## Transaktionssikkerhed

Selvom forbindelsen mellem server og terminal er krypteret, skal transaktionerne som laves stadig sikres for replay attacks. Dette er et angreb hvor indholdet af en besked kopieres og afsendes igen. For at dette ikke kan gøres, er **hver transaktion forbundet med en UUID-key**, som gemmes i et dictionary. Derfor vil en kopieret transaktion ikke tælle som en ny transaktion, men blot opdatere samme transaktion med samme værdi. Derudover indeholder hver transaktionsbesked også data om brugerens *balance*.

## Brugerdata

Da det er vigtigt at minimere brugen af unødvendig brugerdata, har vi grundigt overvejet hvilke informationer som skal gemmes om vores brugere, samt hvilke informationer der skal vises på forskellige tidspunkter. **Vi har valgt at de relevante oplysninger er dem, som skal bruges når et mistet armbånd findes (navn, adresse, email), dem som skal bruges til identifikation (billede), samt dem som kan have indflydelse på ens udvalg i baren (fødselsdato).** Når baren bruges på normal vis, vises kun alder og billede af ejeren af chippen, da alt andet er irrelevant for den som betjener i baren. Disse brugerdata sendes altid krypteret gennem sockets, så derfor kan fremmede systemer ikke se dem.

## Prototype 5 - App/Website

For at gøre sådan at brugere kunne bruge vores system. Så ville vi tilføje en app eller en hjemmeside men selv kunne oprette sig på. Men også hvor man kan se hvor mange penge man har. Og hvor meget man har brugt.

Ideén var at vi havde en layout hvor man kan tilføje penge. Løse sin chip. evt. mere i den retning. Men vi ville gøre det nemt at sætte penge ind og ændre sine data.

Vi har en hjemmeside som er oppe, men den er ikke udviklet til at kunne gøre noget. Den viser at vi har muligheden til at videreudvikle det system, ovenpå den infrastruktur som allerede er opsat på serveren.



# Prototype 6 - Samlet prototype

Gennem skabelsen af de mange ovenstående prototyper, kan man skabe en samlet prototype af de systembestanddele som prototyperne udgør. Med denne samlede prototype vil man principielt være i stand til at betjene "Bip-Baren" og alle dens primære elementer (ikke den brugerstyrede side; App/Website).

**Det samlede system består af følgende primære elementer:**

NFC-armbånd



NFC-reader med optimal interaktion



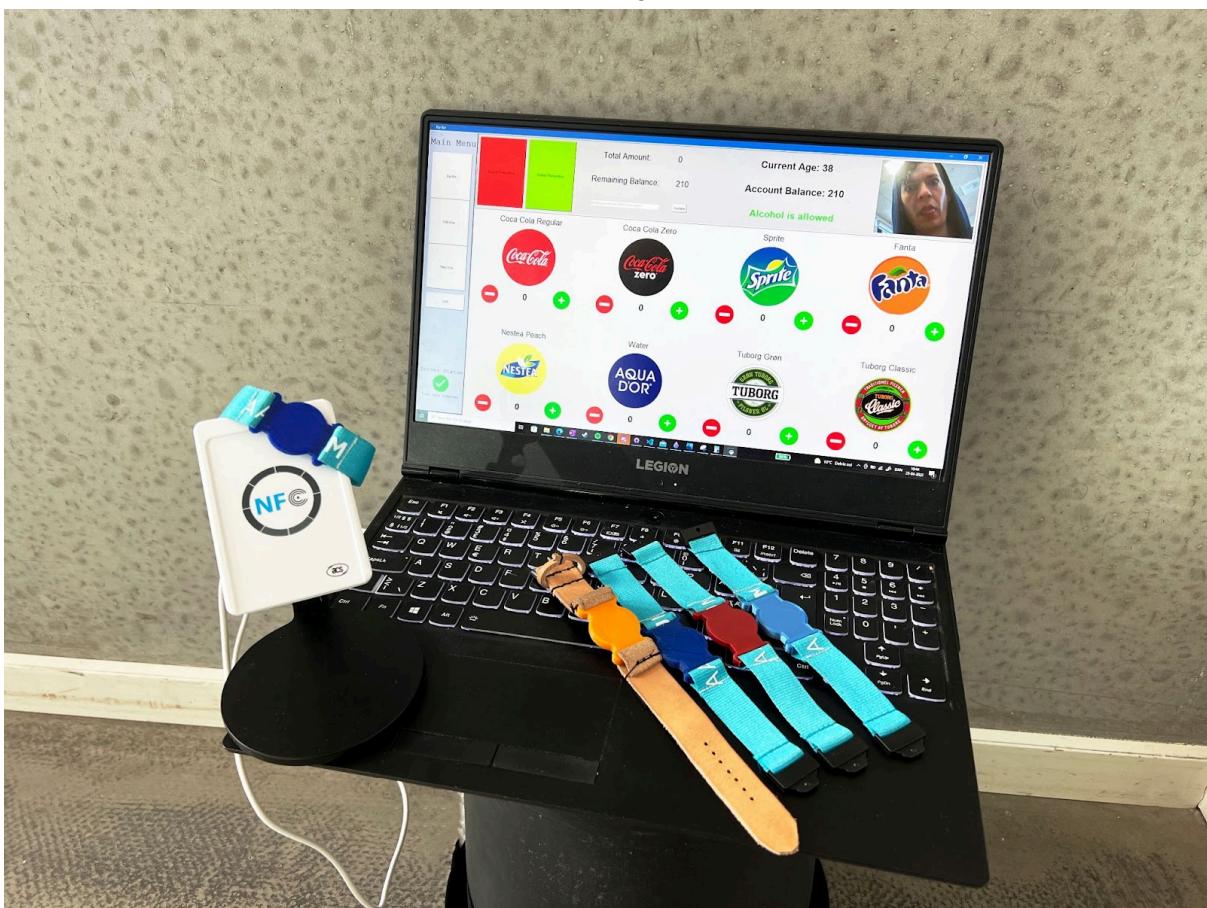
Terminal



Server



Alle elementerne samles i den samlede prototype, som kan ses herunder:



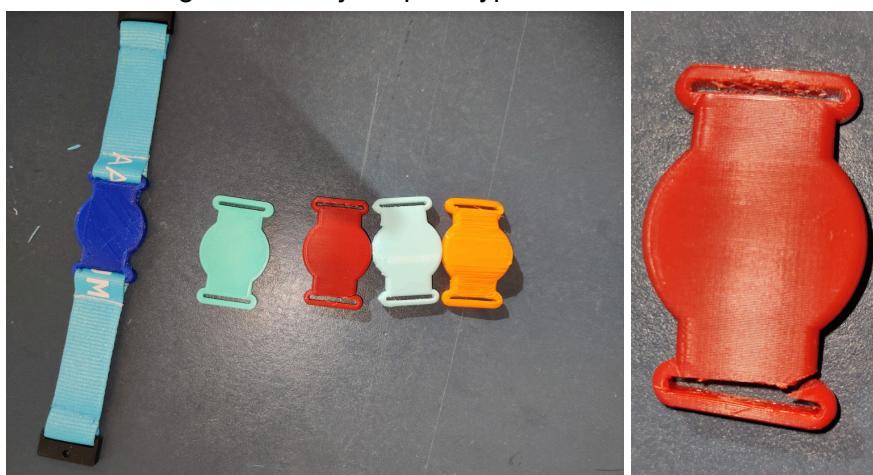
# Test af prototyper

## Test af det fysiske design

### Miljøtest

For at se hvordan vores prototyper klarer sig, vil vi sætte produktet i en igennem forskellige miljøtest, vi vil noter ned de vigtig observationer.

- Vandtest<sup>11</sup>
  - Vi havde sat to prototyper ned i en balje vand, da begge var lavet ud af forskelligt 3D-filament, et solidt PLA filament og TPU filament, vi havde dem nede i vandet i 3 timer hvor vi testede og vi kunne scanne nfc-chips mens de var under vandet.
  - Vi fandt frem til at begge arm er tilstrækkeligt vandtætte, de begge armbånd overlevede gennem de 3 timer i vand, hvor de blev scannet flere gange i processen.
- Træde test
  - Vi trådt på alle prototyper på begge sider da nogen af dem havde en bue. Der var kun to prototyper som gik i stykker.
  - Rød: Den røde gik i stykker ved remmen og splittede op efter vi trampede flere gange på den.
  - Babyblå: Kun ene side af en af uremmen knækkede, så den sidder stadig sammen.
- Drop test<sup>12</sup>
  - Alle vores prototyper var stadig intakte, da vi kastede dem ned fra altanen i annekts. Dog blev de bøjede prototyper mere skadede



<sup>11</sup> <https://drive.google.com/drive/folders/1foXSjFb6ERZnytnJDpG70T6sTZsDz5TS>

<sup>12</sup> <https://drive.google.com/drive/folders/1foXSjFb6ERZnytnJDpG70T6sTZsDz5TS>

## Brugertest af armbånd

Til at teste armbänder, anvendes forrige framework fra Anatomy of Prototypes til at filtrere dele af produktet, hvor der i dette tilfælde ikke behøves at tage forbehold for udseende og funktionalitet. Dette åbner muligheden for at teste prototypens komfort før, den digitale del af produktet virker. Der opstilles en test hvor deltagende modtager et armbånd inden de tager i byen. Deres bytur forløber, som den plejer, hvor der indtages alkoholiske drikkevarer.

For at få brugererfaring med de forskellige fremstillede designs, sættes de forskellige armbånd på syv forskellige testpersoner. Det gælder både de genbrugelige armbånd og dem til engangsbrug. Formålet ved brugertesten er både at få ærlige meninger og erfaringer fra vores testpersoner. Disse fremskaffes med åbne spørgsmål om brugeroplevelsen, hvor brugerne frit kan forklare om deres oplevelser med armbåndene.

Udover feedback fra brugerne, kan armbåndene også undersøges for slid efter brug. Dette er specielt relevant for armbåndene til engangsbrug, da slid kan vurderes efter kortere tids brug.

*Resultaterne af denne test kan ses i bilag under*

## Evaluering af resultater

# Test af det digitale design

## Brugertest

For at teste det digitale design, benyttes en testperson, som uden at få introduktion til det lavede GUI-interface, skal prøve at betjene baren, samt redigere en brugers oplysninger.

Følgende opgave blev stillet til testpersonen:

- "Prøv at sælg denne bruger to øl og en cola, og redigér derefter brugerens navn fra 'Mark Moore' til Johnny."

Uden yderligere introduktion til terminalen, var testpersonen i stand til, uden problemer, at sælge kunden to øl og en cola. Herefter var testpersonen i stand til at redigere brugerens navn. Testpersonen blev en smule forvirret over hvordan brugens navn blev opdateret i databasen.

Se herunder:

The screenshot shows a user interface titled 'Edit User Information'. It contains fields for 'Fuldtnavn' (Full name) with the value 'Johnny', 'E-mail adresse' (Email address) with the value 'mark@moore.dk', and 'Adresse' (Address) with the value 'Dollerupvej 2, 8000 Aar'. Below these are dropdown menus for 'Fødselsdato' (Birth date) with values '04', '04', and '1984'. At the bottom is a button labeled 'Nyt billede' (New picture). To the right of the address field is a button labeled 'Opdatér bruger og scan NFC' (Update user and scan NFC).

Brugeren havde to sekunders forvirring omkring hvorvidt de skulle trykke på "Nyt billede" eller på "Opret bruger og scan NFC". Dog kunne testpersonen læse sig frem til næste trin, og formåede at redigere brugerens oplysninger.

## Komplet test

Et system kan sjældent testes fuldstændigt, men man kan forsøge at teste så mange elementer som muligt. Derfor har vi brugt en passende mængde tid på at prøve alt funktionalitet af i terminalen og serveren. Alle fejl som kunne findes er nu rettet. Systemet kan kun fejle i øjeblikket, hvis en NFC-chip læses forkert. Dette er ikke et stort problem, men bør rettes i fremtiden.

# Anatomy of prototypes

I det videnskabelig blad "*Anatomy of Prototypes: Prototypes as filters, prototypes as manifestations of Design ideas*" foreslås anatomy of prototypes som et framework. Dette framework bygger på principperne, at prototyper både kan bidrage generativt til at udforske et designspace og analytisk til at reflektere og evaluere en design beslutning. Det vigtige ved dette framework er, at kan se prototyper som filtre og prototyper som manifestation af design idéer. Disse elementer er essentielle i teksten og bliver beskrevet med de engelske begreber som: *Prototypes as filters and prototypes as manifestations*.

## Prototypes as filters

Vi har brugt vores iterationer af vores armbånd til at reflekterer hvordan vores prototype skal designes, først så vi på formen af vores armbånd, vi havde kigget gennem nettet om hvilke typer af armbånd man kunne lave, først fandt vi frem til at fire forskellige armbånds designs, en af de designs var et selvlavet armbånd som havde en selvlavet NFC-holder spændt på sig selv, vi tænkte på selv at lave armbåndet da vi selv har kontrol over hvordan armbåndet skulle designes.

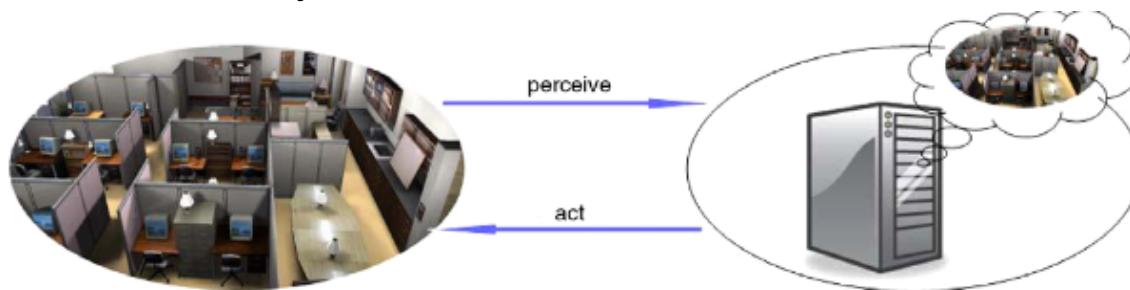
Vi startede først med at kigge på NFC-holderen, vi havde kiggede på selve figuren og størrelsen som holderen skulle have, så vi gik igennem iterationer af designs idéer. Efter vi havde kiggede på NFC-holder formen kiggede vi på ur lås, hvor vi i gruppen argumenterede for og imod de forskellige ur låse og låsemekanismer.

## Prototypes as manifestations of design ideas

# Ambient intelligence (Aml)

Principperne i "Ambient Intelligence: Technologies, applications, and opportunities" danner i høj grad bund for den digitale del af vores projekt. For at opsummere beskriver Aml-teksten hvilke faktorer som er vigtige når der sker en bevidst databehandling som aktiveres af en bruger i et system. Denne databehandling karakteriseres ved at ske fra én part til en central server, som behandler dataen og handler ud fra den. Dette ses i kommunikationsvejen for dataen:

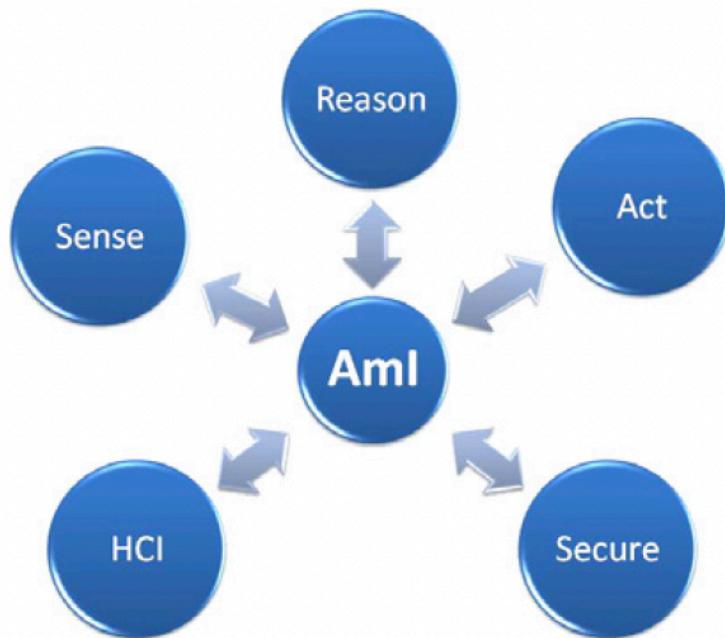
## Kommunikationsvejen



Gennem det forløb som sker fra noget data opsamles til det er færdigbehandlet, er der en række faktorer som er vigtige. Disse ses ved faktorer ved kommunikationen:

## Faktorer ved kommunikationen

D.J. Cook et al. / Pervasive and Mobile Computing 5 (2009) 277–298



Det vil nu beskrives hvordan vores Bip-Bar-system er bygget på de principper som disse faktorer består af.

### **Sensing**

Sensing-begrebet dækker over måden systemet indsamler data som behandles. I vores system dækker sensing-begrebet over det at indtaste brugeroplysninger og det at scanne sin NFC-chip. Her får man for første gang data fra en bruger som skal behandles.

### **Reasoning**

Når der er indsamlet data, er det op til systemet at beslutte hvad der skal ske med denne data, og at udføre den nødvendige behandling, for at dataen bliver som man vil have den. Dette sker for eksempel i vores terminal, hvor brugeroplysninger indsættes i den datastruktur som databasen består af. Når ens data er som man vil have den, kan man begynde at tage stilling til hvad der skal ske med den data.

### **Acting**

Acting-begrebet kommer i spil når den indsamlede data benyttes til noget. Lad os forestille os at vi har indsamlet data fra en NFC-chip for at lave en transaktion, og har sendt den i det rette format til serveren. Det er nu op til serveren at sende det relevante data for at skabe en transaktion (fødselsdato, billdele, balance, osv). Herefter benytter terminalen sig også af acting-begrebet ved at vise de modtagne data til terminalens bruger.

### **Human-computer interaction**

De to næste begreber er med til at underbygge de tre første. HCI-begrebet dækker over den brugerinteraktion som sker i ens system. Altså måden brugeren og computeren interagerer med hinanden. Alt dette er der taget stilling til med brugertests og viden om kropslig interaktion.

### **Secure**

For at have et godt Aml-system, er det vigtigt at systemet er sikkert og pålideligt. Derfor har sikkerhed også været en stor prioritet i dette projekt. Alle de sikkerhedstiltag vi har taget, kan ses i afsnittet om sikkerhed. ([Prototype 4 - Sikkerhed](#))

# Affordances and Design + How Bodies Matter

Når vi har designet det Human-computer interface som vores system består af, har vi haft teorien fra “Affordances and Design” samt “How Bodies Matter” i baghovedet. Dette gælder både for brugeren som skal scanne sin NFC-chip på det rigtige sted, samt den bruger som betjener terminalen. Teorien hvor hver af disse to diskuteres herunder:

## Terminal

Til designet af terminalen, er affordance-teorien og How Bodies Matter begge to meget vigtige. How Bodies Matter snakker om fem temaer: “Thinking through doing”, “Performance”, “Visibility”, “Risk” og “Thickness of practice”. Her indgår affordanceteorien i visibility-temaet. Af disse temaer

## NFC-reader

Til NFC-readeren benyttes primært teori fra Affordances and Design. For at informere kunden, og til dels også terminalens bruger, bruges følgende affordances:

1. Til NFC-readeren bruges først en visuel affordance, som viser hvor man kan scanne sin NFC-chip. Denne affordance består af et rundt NFC-logo med samme form som ens chip på armbåndet, og fungerer derfor som et match, som man intuitivt vil sætte sammen.
2. Udover logoet, er der også et indbygget grønt lys, som starter når læseren er klar til at aflæse en chip, og som slukker igen når læseren har læst chippen og slukker igen.
3. Til sidst har vi selv implementeret en auditiv affordance, da vi har kodet terminalen til at sige en Bip-lyd når en chip scannes. Denne lyd bruges også som affordance, til at vise brugeren at chippen er scannet, og at man gerne må fjerne sit armbånd fra scanneren igen.

Samlet set skulle disse tre affordances gerne informere kunden om alt de behøver at vide om NFC-scanneren. En eventuel forbedring kunne være matchende logoer på armbånd og NFC-scanneren, for at gøre deres forbindelse endnu mere tydelig.



## How bodies matter

### Five themes

1. Thinking through doing
2. Performance
3. Visibility
4. Risk
5. Thickness of practice

The first, thinking through doing, describes how thought (mind) and action (body) are deeply integrated and how they co-produce learning and reasoning.

- Når man interagerer med noget, samarbejder krop og sjæl for at forstå det. Man kommer derfor fremad i sin brugsproces ved at prøve sig frem.

The second, performance, describes the rich actions our bodies are capable of, and how physical action can be both faster and more nuanced than symbolic cognition.

- Dækker over hvilke ting kroppen er i stand til at gøre i interaktioner. Når man vænner sig til at bruge noget, smelter dine handlinger sammen med den ting du bruger. Mark giver et eksempel med hvordan man "styrer bassen" som lydmand i stedet for at rykke på et håndtag.

The first two themes primarily address individual corporeality; the next two are primarily concerned with the social affordances.

Visibility describes the role of artifacts in collaboration and cooperation.

- Dækker over de affordances som giver i en interaktion, samt hvordan man kan se hvad der sker i de forskellige dele af interaktionen med et system. Mark giver eksempel med papirstrips i air traffic control.

Risk explores how the uncertainty and risk of physical co-presence shapes interpersonal and human-computer interactions.

- **Når man vælger at interagere med noget, er der en risiko man tager, for at noget kan have en anden effekt end man troede. Det er hvad dette begreb dækker over.**

The final theme, thickness of practice, suggests that because the pursuit of digital verisimilitude is more difficult than it might seem, embodied interaction is a more prudent path

- **Man kan benytte øvelse til interaktion, da det er en mere stabil og nemmere vej, end at brugerens fra start skal have komplet selvsikkerhed i interaktionen.**

folk til at teste og se hvad de gøre/synes om produktet.

Få feedback af hvad kunne gøres bedre.

## Konklusion

# Links

Github - <https://github.com/froemosen/Bip-Bar>

Trello link: - <https://trello.com/b/LHTyaqzR/eksamens-projekt-teknik-fag>

Miro: - [https://miro.com/app/board/uXjVOJfvjQ8=/?share\\_link\\_id=93355106485](https://miro.com/app/board/uXjVOJfvjQ8=/?share_link_id=93355106485)

# Litteraturliste

Prof Bill Buchanan, OBE. (s.d.). If you're struggling picking a Crypto suite ... Fernet may be the answer. *Medium*. Lokaliseret den 5. 8 2018 på <https://medium.com/coinmonks/if-youre-struggling-picking-a-crypto-suite-fernet-may-be-the-answer-95196c0fec4b>

How secure is a nfc tag. (2017, 16 . 02). *NFC-TAG-SHOP*.

<https://www.nfc-tag-shop.de/info/how-secure-is-nfc.html>

Lamont Wood, LW. (2011, 21. 3). The Clock Is Ticking for Encryption. *Computerworld*.

<https://www.computerworld.com/article/2550008/the-clock-is-ticking-for-encryption.html>

Låstyper. (s.d.). *Anytimewatches*. <https://www.anytimewatches.com/da/laastyper.html>

Nae, Nae. (2017, 21. 12). How to add placeholder to an Entry in tkinter?. *Stackoverflow*.

<https://stackoverflow.com/questions/27820178/how-to-add-placeholder-to-an-entry-in-tkinter>

Stephen Tiedemann, S.T. (s.d.). Getting started with nfcpy. *nfcpy*.

<https://nfcpy.readthedocs.io/en/latest/topics/get-started.html#read-and-write-tags>

Søren Holm-Hansen, SHH. (2013, 9. 8). Kontantløs Smukfest ligner en succes. *JP*.

<https://jyllands-posten.dk/jpaarhus/aarhus/article5801821.ece>

# Bilag

## Brugertest resultater

### Dette var spørgsmålene som blev stillet:

"Hej folkens.

Nu da I var blevet valgt til at teste vores armbånds design, var vi et par spørgsmål vi vil stille jer.

- 1) Hvad synes I om jeres armbånds behagelighed?
- 2) Hvad synes I om jeres armbånds sikkerheds design?
- 3) Hvad vil I ændre på hvis I kunne ændre armbåndet?

Svar gerne dybdegående og ærligt :))"

### Feedback til gummiarmbånd:

"1) Armbåndet var super behagligt. Under hele aftenen lagde jeg stort set ikke mærke til at jeg havde det på, bortset fra når jeg brugt det som en form for fidget-toy, da det var lidt sjovt.  
2) Armbåndet virkede meget sikkert. Når armbåndet passer til ens håndled, er det svært at tage af, specielt uden at man opdager det.  
3) Hvis armbåndet var lidt mere strækbart, ville det blive lidt nemmere at tage af og på. Det var nemlig lidt svært. Udover det er der ikke meget at sige. Jeg kan dog forestille mig at andre med mindre håndled ville have et armbånd i mindre størrelse."

### Feedback til stof armbånd:

"1) Stof armbåndet var meget behagligt og jeg glemte hurtigt at jeg havde den på. Jeg tænkte ikke på at jeg havde den på  
2) Synes den var meget sikker fordi man skal have en saks for at få den af. Den sad også tæt på huden men heller ikke for tæt. Så den er svær at få af  
3) Jeg ville som såden ikke ændre noget da jeg synes den var god at havde på. Den var behaglig med stoffet og den kradsede heller ikke. Jeg lagde først mærke til den igen dagen efter. "

### Feedback til plastik armbånd:

"1) Min var både behagelig og ikke behagelig, altså i at man ikke rigtig lag mærke til man havde den på, og den dermed ikke irriterede på nogen måde.  
2) min var utrolig sikker, den var ikke løs eller jeg var bange for den vil falde af. Og den sad stram nok for at den ikke hang fast i noget  
3) ved min var der ikke noget at klage over, den var behagelig at have på, der var stor mulighed for at den skulle side på et meget stort eller tyndt håndled hvilket også var nice. Farven var måske en smule for "neon" til jeg selv vil gå med den i længere tid."

### Feedback til papirarmbånd:

"1) mit armbånd var ikke særligt behageligt, men på den anden side var den smule det irriterede ikke det store problem. Muligvis fordi jeg var stiv af helvede til.

- 2) mit armbånds sikkerhed var enligt fint, dog havde jeg det kun på en aften og det blev det ret slidt af. Så ved ikke om det ville gå i en længere periode.
- 3) det svært at sige, men muligvis i et andet materiale, så det ikke ser så slidt ud når det bliver brugt."

**Feedback til velcro armbånd:**

- "1) Mit armbånd var rigtigt behagelig med materialet remmen var lavet ud af og selve chip holderen var ikke for distraherende, da man stadig kunne mærke den i gang i mellem.
- 2) Velcro delen sidder meget godt fast på armen mens man danser og hopper, men hvis folk prøver at tag armbåndet vil det røge af meget nemt.
- 3) Måske en anden sikkerhedsmekanisme så folk ikke bare kan rive armbåndet af.

**Feedback til click armbånd?:**

1. Mit armbånd sad lidt løst og kunne ikke justeres i størrelsen hvilket gjorde det lidt ubehageligt.
2. Både båndet og brikken holdt meget godt, selvom jeg bankede den i bordet gentagende gange. låsen var lidt for let at åbne, hvilket godt kan forbedres.
3. Låsen må gerne være stærkere. Brikken var lidt stor og kunne derfor ændres i størrelsen."

**Feedback til spændelås armbånd:**

# Kildekode

## Server kode (app.py)

```
import pickle
from flask import Flask
from flask_socketio import SocketIO, emit

app = Flask(__name__)
socketio = SocketIO(app)

@app.route("/")
def hello_world():
    return "<h1>Hello, World!</h1>\n<h3>This is the Bip-Bar website running locally!</h3>"

@socketio.on('PublicData') #Socket public data request
def SendDataPublic(data):
    print("I HAVE RECIEVED THE 'PublicData' REQUEST")
    dbfile = pickle.load(open( "dbbb", "rb"))
    try:
        user = (dbfile[data[0]]) #All user data from userID

        #Slet unødvendig info
        user.pop("name")
        user.pop("email")
        user.pop("adress")

        print(user)

        #Find billede til identifikation og load bytes
        with open(f'ServerPhotos/{data[0]}.jpg', 'rb') as f:
            image = f.read()

        if user["chipID"] == data[1]: #Tjek om ChipID stemmer overens med userID
            emit("recieveData", [user, image])
        else: emit("serverDenied", "FALSE CHIP-ID\n Public")

    except: emit("serverDenied", "FALSE USER-ID\n Public") #Hvis der ikke er en bruger med det UserID

@socketio.on("PrivateData") #Socket private data request
def SendDataPrivate(data):
    print("I HAVE RECIEVED THE 'PrivateData' REQUEST")
    dbfile = pickle.load(open( "dbbb", "rb")) #load database
    try:
        user = (dbfile[data[0]]) #All user data from userID
        print(user)

        #Find billede til identifikation og load bytes
```

```

        with open(f'ServerPhotos/{data[0]}.jpg', 'rb') as f:
            image = f.read()

        if user["chipID"] == data[1]: #Tjek om ChipID stemmer overens med userID
            emit("recieveData", [user, image])
        else: emit("serverDenied", "FALSE CHIP-ID\n Private")

    except: emit("serverDenied", "FALSE USER-ID\n Private") #Hvis der ikke er en bruger
    med det UserID

@socketio.on("NewUser") ##Socket new user request
def NewUser(data):
    print("New user is being created...")

    db = pickle.load(open("dbbb", "rb")) #Load database
    dbUser = data #userData

    if list(dbUser.keys())[0] in db: userEdit = True #Check if user is already in
    database
    else: userEdit = False

    try:
        db.update(dbUser) #update userData in database

        #Write to database and save
        dbfile = open('dbbb', 'wb')
        pickle.dump(db, dbfile)
        dbfile.close()

        if userEdit == False: emit("serverConfirmation", "New user created!")
        elif userEdit == True: emit("serverConfirmation", "User Updated!")

    except:
        if userEdit == False: emit("serverConfirmation", "New user failed!")
        elif userEdit == True: emit("serverConfirmation", "User update failed!")

@socketio.on("Billede") #Socket image request
def Billede(data):
    with open(f"ServerPhotos/{data[0]}.jpg", "wb") as binary_file:
        binary_file.write(data[1]) # Write bytes to file


@socketio.on("Trans") #Socket transaction request
def CreateTransaction(data):
    try:
        db = pickle.load(open("dbbb", "rb")) #Load database
        dbUser = data[0] #userData
        transaction = data[1]
        newBalance = data[2]
        db[dbUser]["transactions"].update(transaction) #update userData in database
        db[dbUser]["balance"] = newBalance
    
```

```

print(db[dbUser])

#Check sum of transactions against newBalance
if sum(db[dbUser]["transactions"].values()) == newBalance:
    transAuthentication = True
else: transAuthentication = False

#Write to database and save
dbfile = open('dbbb', 'wb')
pickle.dump(db, dbfile)
dbfile.close()

if transAuthentication: emit("serverConfirmation", "Transaction received!")
else: emit("serverConfirmation", "Transaction received,\n but balance does\n not
match list\n of transactions!")
except Exception as e:
    print(e)
    emit("serverDenied", "Transaction failed\n for unknown reasons!")

if __name__ == '__main__':
    app.run()

```

## Terminal kode (Main.py)

```
import tkinter as tk # (pip install tkinter)
import entryWithPlaceholder #entryWithPlaceholder.py (local file)
import cv2 #Camera stuff (pip install opencv-python)
import PIL.Image, PIL.ImageTk #tkinter stuff with PIL (pip install Pillow)
import uuid #Random uuid-generation
import ndef # (pip install ndefpy)? - might be included in (pip install nfc)?
import socketio # (pip install "python-socketio[client]")
import time
from datetime import date
import shutil
import winsound

class MainFrame(tk.Frame):
    def __init__(self, *args, **kwargs):
        tk.Frame.__init__(self, *args, **kwargs, bg = '#bcc8e8')

        #Først laver vi kasser til selve knapperne.
        ButtonFrame = tk.Frame(self, bg = '#bcc8e8', borderwidth = 3, relief = 'raised')
        self.Box = tk.Frame(self)
        ButtonFrame.pack(side = "left", fill = "both", expand= False)
        self.Box.pack(side = "left", fill = "both", expand= True)

        mainMenu = tk.Label(ButtonFrame, text = "Main Menu", bg = '#bcc8e8')
        mainMenu.config(font=("Courier", 24), bg = '#bcc8e8')

        #Selv knapperne bliver lavet
        self.Bip_Bar_Button = tk.Button(ButtonFrame, text = "Bip Bar", height = 10,
width = 20)
        self.Edit_User_Button = tk.Button(ButtonFrame, text = "Edit User", height = 10,
width = 20)
        self.New_User_Button = tk.Button(ButtonFrame, text = "New User", height = 10,
width = 20)
        Reset_Button = tk.Button(ButtonFrame, text = "reset", height = 3, width = 20,
command = lambda: self.reset("all"))

        #Skab vinduer
        self.createWindows("all")

        #Knapper formatteres i tabel
        mainMenu.grid(row = 0, column = 0, padx = 0, pady = 15)
        self.Bip_Bar_Button.grid(row = 1, column = 0, padx = 0, pady = 0)
        self.Edit_User_Button.grid(row = 2, column = 0, padx = 0, pady = 0)
        self.New_User_Button.grid(row = 3, column = 0, padx = 0, pady = 0)
        Reset_Button.grid(row = 4, column = 0, padx = 0, pady = 15)

        #SERVER STATUS ELEMENTS!
        statusLabel = tk.Label(ButtonFrame, text = "Server Status", font=("Courier",
16), bg = '#bcc8e8')
```

```

        self.statusCanvas = tk.Canvas(ButtonFrame, width = 64, height = 64, bg =
 '#bcc8e8', highlightbackground = '#bcc8e8')
        self.statusMsg = tk.Label(ButtonFrame, text = "", font=("Courier", 10), bg =
 '#bcc8e8')

        self.succesIcon = PIL.ImageTk.PhotoImage(PIL.Image.open("Icons\succesIcon.png"))
        self.errorIcon = PIL.ImageTk.PhotoImage(PIL.Image.open("Icons\errorIcon.png"))
        self.waitIcon = PIL.ImageTk.PhotoImage(PIL.Image.open("Icons\waitIcon.png"))

        statusLabel.grid(row = 5, column = 0, padx = 0, pady = (200, 0))
        self.statusCanvas.grid(row = 6, column = 0)
        self.statusMsg.grid(row = 7, column = 0)

    def createWindows(self, windowsToCreate):
        if windowsToCreate == "all":
            self.Bip_Bar_Window = Bip_Bar(self)
            self.Edit_User_Window = Edit_User(self)
            self.New_User_Window = New_User(self)

            #Placering for kasserne
            self.Bip_Bar_Window.place(in_ = self.Box, x = 0, y = 0, relwidth = 1,
relheight = 1)
            self.Edit_User_Window.place(in_ = self.Box, x = 0, y = 0, relwidth = 1,
relheight = 1)
            self.New_User_Window.place(in_ = self.Box, x = 0, y = 0, relwidth = 1,
relheight = 1)

            self.Bip_Bar_Button.configure(command = self.Bip_Bar_Window.show)
            self.Edit_User_Button.configure(command = self.Edit_User_Window.show)
            self.New_User_Button.configure(command = self.New_User_Window.show)

            #Hvilken side programmet skal starte i
            self.Bip_Bar_Window.show()

        elif windowsToCreate == "Bip_Bar":
            self.Bip_Bar_Window = Bip_Bar(self)

            #Placering for kasserne
            self.Bip_Bar_Window.place(in_ = self.Box, x = 0, y = 0, relwidth = 1,
relheight = 1)

            self.Bip_Bar_Button.configure(command = self.Bip_Bar_Window.show)

            #Hvilken side programmet skal starte i
            self.Bip_Bar_Window.show()

        elif windowsToCreate == "Edit_User":
            self.Edit_User_Window = Edit_User(self)

            #Placering for kasserne

```

```

        self.Edit_User_Window.place(in_= self.Box, x = 0, y = 0, relwidth = 1,
relheight = 1)

        self.Edit_User_Button.configure(command = self.Edit_User_Window.show)

#Hvilken side programmet skal starte i
self.Edit_User_Window.show()

elif windowsToCreate == "New_User":
    self.New_User_Window = New_User(self)

#Placering for kasserne
self.New_User_Window.place(in_= self.Box, x = 0, y = 0, relwidth = 1,
relheight = 1)

    self.New_User_Button.configure(command = self.New_User_Window.show)

#Hvilken side programmet skal starte i
self.New_User_Window.show()

def reset(self, windowsToReset):
    if windowsToReset == "all": #Alle vinduer resettes
        self.Bip_Bar_Window.destroy()
        self.Edit_User_Window.destroy()
        self.New_User_Window.destroy()

        self.createWindows("all")

    elif windowsToReset == "Bip_Bar": #Bip Bar-vindue resettes
        self.Bip_Bar_Window.destroy()

        self.createWindows("Bip_Bar")

    elif windowsToReset == "Edit_User": #Edit User-vindue resettes
        self.Edit_User_Window.destroy()

        self.createWindows("Edit_User")

    elif windowsToReset == "New_User": #New User-vindue resettes
        self.New_User_Window.destroy()

        self.createWindows("New_User")

class page(tk.Frame):
    def __init__(self, *args, **kwargs):
        tk.Frame.__init__(self, *args, **kwargs)

    def show(self):
        self.lift()

```

```

def userInfo(self, placeholder): #Skaber widgets til at inputte brugerdata
    navnText = tk.Label(self, text = "Fuld navn")
    self.navnInput = tk.Entry(self)

    emailText = tk.Label(self, text = "E-mail adresse")
    self.emailInput = tk.Entry(self)

    adresseText = tk.Label(self, text = "Adresse")
    self.adresseInput = tk.Entry(self)

    birthdayText = tk.Label(self, text = "Fødselsdato")

    if placeholder == True:
        self.date = entryWithPlaceholder.EntryWithPlaceholder(self, "dag")
        self.month = entryWithPlaceholder.EntryWithPlaceholder(self, "månedstal
(01-12)")
        self.year = entryWithPlaceholder.EntryWithPlaceholder(self, "år")
    else:
        self.date = tk.Entry(self)
        self.month = tk.Entry(self)
        self.year = tk.Entry(self)

    billedeText = tk.Label(self, text = "Billede til identifikation")
    self.btn_billedet = tk.Button(self, text = "Tag billede",
command=self.toggleLiveView)

    navnText.grid(row = 1, column = 0, padx = 0, pady = 0)
    self.navnInput.grid(row = 2, column = 0, padx = 5, pady = 5)

    emailText.grid(row = 3, column = 0, padx = 30, pady = 5,)
    self.emailInput.grid(row = 4, column = 0, padx = 5, pady = 5)

    adresseText.grid(row = 5, column = 0, padx = 5, pady = 5)
    self.adresseInput.grid(row = 6, column = 0, padx = 5, pady = 5)

    birthdayText.grid(row = 7, column = 0, padx = 5, pady = 5)
    self.date.grid(row = 8, column = 0, padx = 5, pady = 5)
    self.month.grid(row = 9, column = 0, padx = 5, pady = 5)
    self.year.grid(row = 10, column = 0, padx = 5, pady = 5)

    billedeText.grid(row = 11, column = 0, padx = 5, pady = 5)
    self.btn_billedet.grid(row = 12, column = 0, padx = 30, pady = 5)

# -----
# CAMERA SETUP STARTS HERE
# -----


    self.vcap = cv2.VideoCapture(0)
    self.width = 400
    self.height = 400

```

```

#Canvas
self.canvas1 = tk.Canvas(self)
self.canvas1.configure( width= self.width, height=self.height)
self.canvas1.grid(column= 0, row=13,padx = 10, pady=10)

#Start-image
self.takeImage()
# -----
# CAMERA SETUP ENDS HERE
# -----


def toggleLiveView(self): #Om kamera har taget billede eller ej
    if self.liveView == False:
        self.liveView = True
        self.takeImage()
        self.btn_billedede.config(text = "Tag billede")
    else:
        self.liveView = False
        self.btn_billedede.config(text = "Nyt billede")

def takeImage(self): #Triggers image capture
    if self.liveView:
        _, self.frame = self.vcap.read()

        self.frame1 = cv2.cvtColor(self.frame, cv2.COLOR_BGR2RGB)

        self.photo = PIL.ImageTk.PhotoImage(image =
PIL.Image.fromarray(self.frame1))

        self.canvas1.create_image(0,0, image = self.photo, anchor = tk.CENTER)

        self.after(15, self.takeImage)

    else:
        pass


class Bip_Bar(page):
    def __init__(self, *args, **kwargs):
        page.__init__(self, *args, **kwargs)
        self.text = tk.Label(self, text = "New Transaction", width=20, font="FreeMono")
        self.text.grid(row = 0, column = 0, padx = 30, pady = 5)
        self.btn_getUser = tk.Button(self, text = "Hent chip", command = self.getUser,
width=50, height=10, activebackground="green yellow")
        self.btn_getUser.grid(row = 1, column = 0, padx = 30, pady = 5)

    def getUser(self): #Public data
        self.userID, chipID = nfcReader.readData() #Læser NFC-chip
        self.btn_getUser.destroy()
        self.text.destroy()

```

```

#Get relevant user data and image
serverComm.getUser_public(self.userID, chipID)
print("Awaiting data...")
global userData
global userImage
while userData == {}: #Await user data
    time.sleep(0.2)
print("Data ready to be inserted!")
main.statusCanvas.delete("all") #Reset canvas
main.statusCanvas.create_image(2, 2, image = main.succesIcon, anchor = "nw")
#Server status succes
main.statusMsg.config(text = "User data inserted") #Server status message

self.userData = userData #For using userdata in other functions

with open(f"currentImage.jpg", "wb") as binary_file: #Save image for later
    binary_file.write(userImage) # Write bytes to file

#calculate age
birthdayList = userData["birthday"].split("-")
today = str(date.today())
todayList = today.split("-")
todayList.reverse()
self.age = int(todayList[2]) - int(birthdayList[2]) - ((int(todayList[1]),
int(todayList[0])) < (int(birthdayList[1]), int(birthdayList[0])))

#Widget variables
informationWidth = 28
informationHeight = 1
informationFont = ("FreeMono", 24, "bold")
informationBG = "gray82"
drinkLabelWidth = 22
drinkFont = ("FreeMono", 20)
imageSize = 200

#9 rows and 12 coloumns

#Creation of GUI
infoBG = tk.Frame(self, bg = informationBG, height = 12, width = 200,
borderwidth=4, relief="ridge", pady=4)
cancelTransactionBtn = tk.Button(infoBG, text = "Cancel Transaction", height =
15, width = 23, bg = "red", command = lambda: main.reset("Bip_Bar"))
createTransactionBtn = tk.Button(infoBG, text = "Create Transaction", height =
15, width = 23, bg = "lawngreen", command = self.createTransaction)
totalAmountLabel = tk.Label(infoBG, text = "Total Amount: ", width = 21, bg =
informationBG, font = drinkFont)
self.remainBalanceLabel = tk.Label(infoBG, text = "Remaining Balance: ", width =
21, bg = informationBG, font = drinkFont)
self.totalAmountValue = tk.Label(infoBG, text = "0", width = 8, anchor='w', bg =
informationBG, font = drinkFont)
self.remainBalanceValue = tk.Label(infoBG, text = userData['balance'], width =
8, anchor='w', bg = informationBG, font = drinkFont)

```

```

        ageLabel = tk.Label(infoBG, text = f"Current Age: {self.age}",
width=informationWidth, height = informationHeight, font=informationFont, padx=6, bg =
informationBG)
        balanceLabel = tk.Label(infoBG, text = f"Account Balance:
{userData['balance']}]", width=informationWidth, height = informationHeight,
font=informationFont, padx=6, bg = informationBG)
        canvas = tk.Canvas(infoBG, width = 320, height = 240)
        if self.age >= 18: alcoholAllowedLabel = tk.Label(infoBG, text = "Alcohol is
allowed", width=informationWidth, height = informationHeight, font=informationFont,
padx=6, bg = informationBG, fg = "lime green")
        else: alcoholAllowedLabel = tk.Label(infoBG, text = "Alcohol not allowed",
width=informationWidth, height = informationHeight, font=informationFont, padx=6, bg =
informationBG, fg = "red")

        #Prototype version - Add credits to account
        self.insertMoneyEntry = entryWithPlaceholder.EntryWithPlaceholder(infoBG,
placeholder = "Add balance here (Whole number)", width = 40)
        self.updateInsert = tk.Button(infoBG, text = "Update", command = lambda:
self.updateLabel("nothing", "nothing"))

        #Labels
        colaregLabel = tk.Label(self, text = "Coca Cola Regular", width=drinkLabelWidth,
font=drinkFont, padx = 5, pady = 10)
        colazeroLabel = tk.Label(self, text = "Coca Cola Zero", width=drinkLabelWidth,
font=drinkFont, padx = 5, pady = 10)
        spriteLabel = tk.Label(self, text = "Sprite", width=drinkLabelWidth,
font=drinkFont, padx = 5, pady = 10)
        fantaLabel = tk.Label(self, text = "Fanta", width=drinkLabelWidth,
font=drinkFont, padx = 5, pady = 10)
        nesteaLabel = tk.Label(self, text = "Nestea Peach", width=drinkLabelWidth,
font=drinkFont, padx = 5, pady = 10)
        waterLabel = tk.Label(self, text = "Water", width=drinkLabelWidth,
font=drinkFont, padx = 5, pady = 10)

        if self.age >= 18: #Alkohol
            pilsnerLabel = tk.Label(self, text = "Tuborg Grøn", width=drinkLabelWidth,
font=drinkFont, padx = 5, pady = 10)
            classicLabel = tk.Label(self, text = "Tuborg Classic",
width=drinkLabelWidth, font=drinkFont, padx = 5, pady = 10)
        else: pass

        self.colaregImage =
PIL.ImageTk.PhotoImage(PIL.Image.open("MenuPhotos\CocaColaReg.png"))
        colaregCanvas = tk.Canvas(self, width = imageSize, height = imageSize)
        self.colazeroImage =
PIL.ImageTk.PhotoImage(PIL.Image.open("MenuPhotos\CocaColaZero.png"))
        colazeroCanvas = tk.Canvas(self, width = imageSize, height = imageSize)
        self.spriteImage =
PIL.ImageTk.PhotoImage(PIL.Image.open("MenuPhotos\Sprite.png"))
        spriteCanvas = tk.Canvas(self, width = imageSize, height = imageSize)
        self.fantaImage = PIL.ImageTk.PhotoImage(PIL.Image.open("MenuPhotos\Fanta.png"))
        fantaCanvas = tk.Canvas(self, width = imageSize, height = imageSize)

```

```

        self.nesteaImage =
PIL.ImageTk.PhotoImage(PIL.Image.open("MenuPhotos/Nestea.png"))
        nesteaCanvas = tk.Canvas(self, width = imageSize, height = imageSize)
        self.waterImage = PIL.ImageTk.PhotoImage(PIL.Image.open("MenuPhotos\Water.png"))
        waterCanvas = tk.Canvas(self, width = imageSize, height = imageSize)
        if self.age >= 18: #Alkohol
            self.pilsnerImage =
PIL.ImageTk.PhotoImage(PIL.Image.open("MenuPhotos\TuborgGrøn.png"))
            pilsnerCanvas = tk.Canvas(self, width = imageSize, height = imageSize)
            self.classicImage =
PIL.ImageTk.PhotoImage(PIL.Image.open("MenuPhotos\TuborgClassic.png"))
            classicCanvas = tk.Canvas(self, width = imageSize, height = imageSize)

        self.minusImage = tk.PhotoImage(file='MenuPhotos/minusBtn.png')
        self.plusImage = tk.PhotoImage(file='MenuPhotos/plusBtn.png')

        colaRegBtnDown = tk.Button(self, image = self.minusImage, borderwidth=0,
command=lambda: self.updateLabel("-", "Cola_Reg"))
        self.colaRegAmountLabel = tk.Label(self, text = "0", font = informationFont)
        colaRegBtnUp = tk.Button(self, image = self.plusImage, borderwidth=0,
command=lambda: self.updateLabel("+", "Cola_Reg"))
        colaZeroBtnDown = tk.Button(self, image = self.minusImage, borderwidth=0,
command=lambda: self.updateLabel("-", "Cola_Zero"))
        self.colaZeroAmountLabel = tk.Label(self, text = "0", font = informationFont)
        colaZeroBtnUp = tk.Button(self, image = self.plusImage, borderwidth=0,
command=lambda: self.updateLabel("+", "Cola_Zero"))
        spriteBtnDown = tk.Button(self, image = self.minusImage, borderwidth=0,
command=lambda: self.updateLabel("-", "Sprite"))
        self.spriteAmountLabel = tk.Label(self, text = "0", font = informationFont)
        spriteBtnUp = tk.Button(self, image = self.plusImage, borderwidth=0,
command=lambda: self.updateLabel("+", "Sprite"))
        fantaBtnDown = tk.Button(self, image = self.minusImage, borderwidth=0,
command=lambda: self.updateLabel("-", "Fanta"))
        self.fantaAmountLabel = tk.Label(self, text = "0", font = informationFont)
        fantaBtnUp = tk.Button(self, image = self.plusImage, borderwidth=0,
command=lambda: self.updateLabel("+", "Fanta"))
        nesteaBtnDown = tk.Button(self, image = self.minusImage, borderwidth=0,
command=lambda: self.updateLabel("-", "Nestea"))
        self.nesteaAmountLabel = tk.Label(self, text = "0", font = informationFont)
        nesteaBtnUp = tk.Button(self, image = self.plusImage, borderwidth=0,
command=lambda: self.updateLabel("+", "Nestea"))
        waterBtnDown = tk.Button(self, image = self.minusImage, borderwidth=0,
command=lambda: self.updateLabel("-", "Water"))
        self.waterAmountLabel = tk.Label(self, text = "0", font = informationFont)
        waterBtnUp = tk.Button(self, image = self.plusImage, borderwidth=0,
command=lambda: self.updateLabel("+", "Water"))

        if self.age >= 18: #Alkohol
            pilsnerBtnDown = tk.Button(self, image = self.minusImage, borderwidth=0,
command=lambda: self.updateLabel("-", "Pilsner"))
            self.pilsnerAmountLabel = tk.Label(self, text = "0", font = informationFont)
            pilsnerBtnUp = tk.Button(self, image = self.plusImage, borderwidth=0,
command=lambda: self.updateLabel("+", "Pilsner"))

```

```

        classicBtnDown = tk.Button(self, image = self.minusImage, borderwidth=0,
command=lambda: self.updateLabel("-", "Classic"))
        self.classicAmountLabel = tk.Label(self, text = "0", font = informationFont)
        classicBtnUp = tk.Button(self, image = self.plusImage, borderwidth=0,
command=lambda: self.updateLabel("+", "Classic"))

#Packing of GUI
infoBG.grid(row = 0, column = 0, columnspan = 12, rowspan = 3)
cancelTransactionBtn.grid(row = 0, column = 0, rowspan = 3, padx = 5)
createTransactionBtn.grid(row = 0, column = 1, rowspan = 3, padx = (5, 18))
totalAmountLabel.grid(row = 0, column = 2, columnspan = 3)
self.remainBalanceLabel.grid(row = 1, column = 2, columnspan = 3)
self.insertMoneyEntry.grid(row = 2, column = 2, columnspan = 3) #PROTOTYPE ENTRY
self.updateInsert.grid(row = 2, column = 5, sticky = "w") #PROTOTYPE BUTTON
self.totalAmountValue.grid(row = 0, column = 5)
self.remainBalanceValue.grid(row = 1, column = 5)
ageLabel.grid(row = 0, column = 6, columnspan = 3, pady = 5)
balanceLabel.grid(row = 1, column = 6, columnspan = 3, pady = 5)
alcoholAllowedLabel.grid(row = 2, column = 6, columnspan = 3, pady = 5)
canvas.grid(column = 9, row = 0 , rowspan = 3, columnspan = 3)

colaregLabel.grid(row = 3, column = 0, columnspan = 3)
colazeroLabel.grid(row = 3, column = 3, columnspan = 3)
spriteLabel.grid(row = 3, column = 6, columnspan = 3)
fantaLabel.grid(row = 3, column = 9, columnspan = 3)
nesteaLabel.grid(row = 7, column = 0, columnspan = 3)
waterLabel.grid(row = 7, column = 3, columnspan = 3)

if self.age >= 18: #Alkohol
    pilsnerLabel.grid(row = 7, column = 6, columnspan = 3)
    classicLabel.grid(row = 7, column = 9, columnspan = 3)

colaregCanvas.grid(row = 4, column = 0, columnspan = 3)
colazeroCanvas.grid(row = 4, column = 3, columnspan = 3)
spriteCanvas.grid(row = 4, column = 6, columnspan = 3)
fantaCanvas.grid(row = 4, column = 9, columnspan = 3)
nesteaCanvas.grid(row = 8, column = 0, columnspan = 3)
waterCanvas.grid(row = 8, column = 3, columnspan = 3)

if self.age >= 18: #Alkohol
    pilsnerCanvas.grid(row = 8, column = 6, columnspan = 3)
    classicCanvas.grid(row = 8, column = 9, columnspan = 3)

colaRegBtnDown.grid(row = 5, column = 0) #Første række starter her
self.colaRegAmountLabel.grid(row = 5, column = 1)
colaRegBtnUp.grid(row = 5, column = 2)
colaZeroBtnDown.grid(row = 5, column = 3)

```

```

        self.colaZeroAmountLabel.grid(row = 5, column = 4)
        colaZeroBtnUp.grid(row = 5, column = 5)
        spriteBtnDown.grid(row = 5, column = 6)
        self.spriteAmountLabel.grid(row = 5, column = 7)
        spriteBtnUp.grid(row = 5, column = 8)
        fantaBtnDown.grid(row = 5, column = 9)
        self.fantaAmountLabel.grid(row = 5, column = 10)
        fantaBtnUp.grid(row = 5, column = 11)

        #Indsættelse af en frame til afstand
        distanceFrame = tk.Frame(self)
        spacing = tk.Label(distanceFrame, width = 120, height = 4)
        distanceFrame.grid(row = 6, column = 0, columnspan = 12)
        spacing.pack()

        nesteaBtnDown.grid(row = 9, column = 0) #Anden række starter her
        self.nesteaAmountLabel.grid(row = 9, column = 1)
        nesteaBtnUp.grid(row = 9, column = 2)
        waterBtnDown.grid(row = 9, column = 3)
        self.waterAmountLabel.grid(row = 9, column = 4)
        waterBtnUp.grid(row = 9, column = 5)

        if self.age >= 18: #Alkohol
            pilsnerBtnDown.grid(row = 9, column = 6)
            self.pilsnerAmountLabel.grid(row = 9, column = 7)
            pilsnerBtnUp.grid(row = 9, column = 8)
            classicBtnDown.grid(row = 9, column = 9)
            self.classicAmountLabel.grid(row = 9, column = 10)
            classicBtnUp.grid(row = 9, column = 11)

        #Indsæt billeder
        colaregCanvas.create_image(2, 2, image = self.colaregImage, anchor = "nw")
        colazeroCanvas.create_image(2, 2, image = self.colazeroImage, anchor = "nw")
        spriteCanvas.create_image(2, 2, image = self.spriteImage, anchor = "nw")
        fantaCanvas.create_image(2, 2, image = self.fantaImage, anchor = "nw")
        nesteaCanvas.create_image(2, 2, image = self.nesteaImage, anchor = "nw")
        waterCanvas.create_image(2, 2, image = self.waterImage, anchor = "nw")

        if self.age >= 18: #Alkohol
            pilsnerCanvas.create_image(2, 2, image = self.pilsnerImage, anchor = "nw")
            classicCanvas.create_image(2, 2, image = self.classicImage, anchor = "nw")

        self.photo = PIL.ImageTk.PhotoImage(PIL.Image.open(f"currentImage.jpg")) #Image
        to ImageTK object
        canvas.create_image(0,0, image = self.photo, anchor = tk.CENTER) #Show image on
        screen

        #Reset globale variable
        userData = {}
        userImage = None

    def createTransaction(self): #Skaber transaktion

```

```

        transEntry = {str(uuid.uuid1()) : -self.totalToPay} #Add balance
        newBalance = self.remainBalance

        print([self.userID, transEntry, newBalance])
        serverComm.createTransaction([self.userID, transEntry, newBalance]) #Send
transaktion til server
        main.statusCanvas.delete("all")
        main.statusCanvas.create_image(2, 2, image = main.waitIcon, anchor = "nw")
        main.statusMsg.config(text = "Please Wait...") #Afvent svar - server status

    def updateLabel(self, operation, item): #Opdaterer labels
        ops = {"+": (lambda x,y: x+y), "-": (lambda x,y: x-y)} #Used as ops["+"] (x,y)
        or ops["-"] (x,y)

        #First evaluation
        self.totalToPay = 0

        labels = [[self.colaRegAmountLabel, 20], [self.colaZeroAmountLabel, 20],
                  [self.spriteAmountLabel, 20], [self.fantaAmountLabel, 20],
                  [self.nesteaAmountLabel, 25], [self.waterAmountLabel, 10]] #labels are
built like [[label1, price], [label2, price]...]

        if self.insertMoneyEntry.get() != "Add balance here (Whole number)": #Prototype
money insert
            try: self.totalToPay -= int(self.insertMoneyEntry.get())
            except: pass

        if self.age >= 18:
            labels.append([self.pilsnerAmountLabel, 30])
            labels.append([self.classicAmountLabel, 30])

        for label in labels:
            self.totalToPay += int(label[0].cget("text"))*label[1] #Current total

        self.remainBalance = int(self.userData['balance'])-self.totalToPay
#self.remainBalance before new item

        #Update items if user can afford
        if item == "Cola_Reg" and self.remainBalance >= 20 or item == "Cola_Reg" and
operation != "+":
            self.colaRegAmountLabel.config(text = ops[operation]
(int(self.colaRegAmountLabel.cget("text")), 1))
            if int(self.colaRegAmountLabel.cget("text")) < 0:
self.colaRegAmountLabel.config(text = "0")
        elif item == "Cola_Zero" and self.remainBalance >= 20 or item == "Cola_Zero" and
operation != "+":
            self.colaZeroAmountLabel.config(text = ops[operation]
(int(self.colaZeroAmountLabel.cget("text")), 1))
            if int(self.colaZeroAmountLabel.cget("text")) < 0:
self.colaZeroAmountLabel.config(text = "0")
        elif item == "Sprite" and self.remainBalance >= 20 or item == "Sprite" and
operation != "+":

```

```

        self.spriteAmountLabel.config(text = ops[operation]
(int(self.spriteAmountLabel.cget("text")), 1))
        if int(self.spriteAmountLabel.cget("text")) < 0:
self.spriteAmountLabel.config(text = "0")
    elif item == "Fanta" and self.remainBalance >= 20 or item == "Fanta" and
operation != "+":
        self.fantaAmountLabel.config(text = ops[operation]
(int(self.fantaAmountLabel.cget("text")), 1))
        if int(self.fantaAmountLabel.cget("text")) < 0:
self.fantaAmountLabel.config(text = "0")
    elif item == "Nestea" and self.remainBalance >= 25 or item == "Nestea" and
operation != "+":
        self.nesteaAmountLabel.config(text = ops[operation]
(int(self.nesteaAmountLabel.cget("text")), 1))
        if int(self.nesteaAmountLabel.cget("text")) < 0:
self.nesteaAmountLabel.config(text = "0")
    elif item == "Water" and self.remainBalance >= 10 or item == "Water" and
operation != "+":
        self.waterAmountLabel.config(text = ops[operation]
(int(self.waterAmountLabel.cget("text")), 1))
        if int(self.waterAmountLabel.cget("text")) < 0:
self.waterAmountLabel.config(text = "0")
    elif item == "Classic" and self.remainBalance >= 30 or item == "Classic" and
operation != "+":
        self.classicAmountLabel.config(text = ops[operation]
(int(self.classicAmountLabel.cget("text")), 1))
        if int(self.classicAmountLabel.cget("text")) < 0:
self.classicAmountLabel.config(text = "0")
    elif item == "Pilsner" and self.remainBalance >= 30 or item == "Pilsner" and
operation != "+":
        self.pilsnerAmountLabel.config(text = ops[operation]
(int(self.pilsnerAmountLabel.cget("text")), 1))
        if int(self.pilsnerAmountLabel.cget("text")) < 0:
self.pilsnerAmountLabel.config(text = "0")
    else: pass

#New evalutation
self.totalToPay = 0

if self.insertMoneyEntry.get() != "Add balance here (Whole number)": #Prototype
money insert
    try: self.totalToPay -= int(self.insertMoneyEntry.get())
    except: pass

for label in labels:
    self.totalToPay += int(label[0].cget("text"))*label[1]

self.remainBalance = int(self.userData['balance'])-self.totalToPay
#self.remainBalance after new item

```

```

        for label in labels: #Make red labels on items not affordable, black on items
with 0 selected and green on selected items
            if label[1] > self.remainBalance: label[0].config(foreground = "red")
            elif int(label[0].cget("text")) == 0: label[0].config(foreground = "black")
            else: label[0].config(foreground = "lime green")

            self.totalAmountValue.config(text = str(self.totalToPay))
            self.remainBalanceValue.config(text = str(self.remainBalance))

class Edit_User(page):
    def __init__(self, *args, **kwargs):
        page.__init__(self, *args, **kwargs)
        text = tk.Label(self, text = "Edit User Information", width=20, font="FreeMono")
        text.grid(row = 0, column = 0, padx = 30, pady = 5)
        self.btn_getUser = tk.Button(self, text = "Hent chip", command = self.getUser,
width=50, height=10, activebackground="green yellow")
        self.btn_getUser.grid(row = 1, column = 0, padx = 30, pady = 5)

        #LiveView Boolean
        self.liveView = False

    def getUser(self): #Private and public data
        #Read NFC
        userID, chipID = nfcReader.readData()
        self.btn_getUser.destroy()

        #Creates boxes for input
        self.userInfo	placeholder=False)
        self.btn_billede.config(text="Nyt billede")

        serverComm.getUser_private(userID, chipID) #Get private user data from server

        print("Awaiting data...")

        global userData
        global userImage

        while userData == {}: #Await userdata
            time.sleep(0.2)

        print("Data ready to be inserted!")
        main.statusCanvas.delete("all")
        main.statusCanvas.create_image(2, 2, image = main.succesIcon, anchor = "nw")
#Server status
        main.statusMsg.config(text = "User data inserted")

        #Inserting data in boxes
        self.navnInput.insert(0, userData["name"])
        self.emailInput.insert(0, userData["email"])
        self.adresseInput.insert(0, userData["adress"])

```

```

#Format birthday
dateList = userData["birthday"].split("-")
self.date.insert(0, dateList[0])
self.month.insert(0, dateList[1])
self.year.insert(0, dateList[2])

#Save image to file
with open(f"currentImage.jpg", "wb") as binary_file:
    binary_file.write(userImage) # Write bytes to file

self.photo = PIL.ImageTk.PhotoImage(PIL.Image.open(f"currentImage.jpg")) #Image
to ImageTK object
    self.canvas1.create_image(0,0, image = self.photo, anchor = tk.CENTER) #Show
image on screen

editUserButton = tk.Button(self, width = 30, height = 2, text = "Opdatér bruger
og scan NFC", command = lambda: self.updateUser(userID, userData["balance"],
userData["transactions"]))
editUserButton.grid(row = 2, column = 1, padx = 0, pady = 0)

def updateUser(self, UserID, balance, transactions):
    global userData
    global userImage

ID = nfcReader.writeData(UserID) #WRITE UserID TO NFC-CHIP AND SECURE

try: cv2.imwrite(f"IDPhotos/{UserID}.jpg", self.frame) #Save image of user for
identification
except: shutil.copyfile("currentImage.jpg", f"IDPhotos/{UserID}.jpg")

#Get userdata and format in order to send to server
userData = {UserID : {"name" : str(self.navnInput.get()),
                      "email" : str(self.emailInput.get()),
                      "adress" : str(self.adresseInput.get()),
                      "birthday" :
str(self.date.get()+'-'+self.month.get()+'-'+self.year.get()),
                      "chipID" : ID,
                      "balance" : balance,
                      "transactions" : transactions
                     }
            }
print(userData)

self.photo = PIL.ImageTk.PhotoImage(PIL.Image.open(f"IDPhotos/{UserID}.jpg"))
#Image to ImageTK object
    self.canvas1.create_image(0,0, image = self.photo, anchor = tk.CENTER) #Show
image on screen

image_data = [UserID]

#Load image again to send in right format
with open(f'IDPhotos/{UserID}.jpg', 'rb') as f:
    image_data.append(f.read())

```

```

#Send UserData and Image to Server
serverComm.updateUser(userData, image_data)
main.statusCanvas.delete("all")
main.statusCanvas.create_image(2, 2, image = main.waitIcon, anchor = "nw")
main.statusMsg.config(text = "Please Wait...")

#Reset global variables
userData = {}
userImage = None

class New_User(page):
    def __init__(self, *args, **kwargs):
        page.__init__(self, *args, **kwargs)

        newUserText = tk.Label(self, text = "Register New User", width=20,
font="FreeMono")
        newUserText.grid(row = 0, column = 0, padx = 0, pady = 0)

        #LiveView Boolean
        self.liveView = True

        #Creates boxes for input
        self.userInfo(placeholder=True)

        newUserButton = tk.Button(self, width = 30, height = 2, text = "Opret ny bruger
og scan NFC", command = self.newUser)
        newUserButton.grid(row = 2, column = 1, padx = 0, pady = 0)

    def newUser(self): #Private and public data
        #GENERATION OF USERID
        UserID = str(uuid.uuid1()) #Generate User ID

        ID = nfcReader.writeData(UserID) #WRITE UserID TO NFC-CHIP AND SECURE

        cv2.imwrite(f"IDPhotos/{UserID}.jpg", self.frame) #Save image of user for
identification

        #Get userdata and format in order to send to server
        userData = {UserID : {"name" : str(self.navnInput.get()),
                           "email" : str(self.emailInput.get()),
                           "adress" : str(self.adresseInput.get()),
                           "birthday" :
str(self.date.get()+'-'+self.month.get()+'-'+self.year.get()),
                           "chipID" : ID,
                           "balance" : 0,
                           "transactions" : {}}
                    }
        print(userData)

```

```

        self.photo = PIL.ImageTk.PhotoImage(PIL.Image.open(f"IDPhotos/{UserID}.jpg"))
#Image to ImageTK object
        self.canvas1.create_image(0,0, image = self.photo, anchor = tk.CENTER) #Show
image on screen

        image_data = [UserID]

#Load image again to send in right format
with open(f'IDPhotos/{UserID}.jpg', 'rb') as f:
    image_data.append(f.read())

#Send UserData and Image to Server
serverComm.updateUser(userData, image_data)
main.statusCanvas.delete("all")
main.statusCanvas.create_image(2, 2, image = main.waitIcon, anchor = "nw")
main.statusMsg.config(text = "Please Wait...")

```

```

class NFC_Reader():
    def __init__(self):
        import nfc

        self.clf = nfc.ContactlessFrontend() #NFC-reader object

    def readData(self):
        #Check NFC-reader connection
        try:
            assert self.clf.open('usb') is True #Determined in cmd by command: "python
-m nfc"
            print(f"Using device: {self.clf}")

        except AssertionError:
            print("NFC-reader not set up correctly. Try again! (Maybe the error is in
the code!)")
            self.clf.close() #Test over - New connection will be needed

        tag = self.clf.connect(rdwr={'on-connect': lambda tag: False})
        print("Tag found!")
        print(tag)

        chipID = tag.identifier.hex()
        try:
            if tag.ndef == None: #If tag is authenticated or carries no data.
                tag.authenticate(b"203ec79c-9288-4612-bac3-9e827d43c5d3")

            if not tag.ndef == None: #If tag carries data
                record = tag.ndef.records[0]
                userID = record.resource.uri #Gets the userID from the NFC-tag records

            self.clf.close()
            winsound.Beep(frequency=2000, duration=400) #Beep-sound

```

```

        return(userID, chipID)

    else:
        print(tag.dump())
        print("Card Carries No Data!\n")
        self.clf.close()
        return(None, None)
except:
    self.clf.close()
    return(None, None)

def writeData(self, inputData):
    #Check NFC-reader connection
    try:
        assert self.clf.open('usb') is True #Determined in cmd by command: "python
-m nfc"
        print(f"Using device: {self.clf}")

    except AssertionError:
        print("NFC-reader not set up correctly. Try again! (Maybe the error is in
the code!)")
        self.clf.close() #Test over - New connection will be needed

    tag = self.clf.connect(rdwr={'on-connect': lambda tag: False})
    if tag.ndef == None: #If tag is authenticated or carries no data.
        tag.authenticate(b"203ec79c-9288-4612-bac3-9e827d43c5d3") #Unlocks tag
    print("Tag found!")
    print(tag)

    chipID = tag.identifier.hex()
    print("unique chip id:", chipID)

    tag.ndef.records = [ndef.SmartposterRecord(inputData)]
    tag.protect(password = b"203ec79c-9288-4612-bac3-9e827d43c5d3", read_protect =
True)

    self.clf.close()
    winsound.Beep(frequency=2000, duration=400) #Beep-sound

    return chipID

class ServerCommunication():
    def __init__(self):
        global sio

    def getUser_private(self, userID, chipID):
        print("Making private user data request...")
        data = [userID, chipID]
        sio.emit("PrivateData", data)
        sio.emit("GetBillede", userID)

```

```

def getUser_public(self, userID, chipID):
    print("Making public user data request...")
    sio.emit("PublicData", [userID, chipID])
    sio.emit("GetBillede", userID)

def updateUser(self, userData, imageData):
    print("Making updateUser request...")
    sio.emit("NewUser", userData)
    sio.emit("Billede", imageData)

def createTransaction(self, transInfo):
    print("Making transaction request...")
    sio.emit("Trans", transInfo)

if __name__ == "__main__":
    nfcReader = NFC_Reader()
    serverComm = ServerCommunication()
    base = tk.Tk()
    base.title("Bip Bar")
    base.state('zoomed') #Laver windowed fullscreen
    main = MainFrame(base)
    main.pack(side = "top", fill = "both", expand = True)

    userData = {}
    userImage = None

#base.mainloop() #USE ONLY FOR TESTING WITHOUT SERVER

#Setup of socketio
sio = socketio.Client()

@sio.event
def connect():
    print("I'm connected!")
    main.statusCanvas.delete("all")
    main.statusCanvas.create_image(2, 2, image = main.succesIcon, anchor = "nw")
    main.statusMsg.config(text = "Connected to server!")

@sio.event
def connect_error(data):
    print("The connection failed!\n", str(data))
    main.statusCanvas.delete("all")
    main.statusCanvas.create_image(2, 2, image = main.errorIcon, anchor = "nw")
    main.statusMsg.config(text = "The connection to\n server failed!")

@sio.event
def disconnect():
    print("I'm disconnected!")
    main.statusCanvas.delete("all")
    main.statusCanvas.create_image(2, 2, image = main.errorIcon, anchor = "nw")
    main.statusMsg.config(text = "Server Disconnected!")

```

```

@sio.on("recieveData")
def recieveData(data):
    print("USER DATA RECIEVED!")
    global userData
    global userImage

    userImage = data[1]
    userData = data[0]

    print(data[0])

@sio.on("serverConfirmation")
def serverConfirmation(data):
    print(data)
    if data == "Transaction recieved!": main.reset("Bip_Bar")
    elif data == "Transaction recieved,\n but balance does\n not match list\n of
transactions!": main.reset("Bip_Bar")
    elif data == "User Updated!": main.reset("Edit_User")
    elif data == "New user created!": main.reset("New_User")
    else: main.reset("all") #Reset windows

    main.statusCanvas.delete("all")
    main.statusCanvas.create_image(2, 2, image = main.succesIcon, anchor = "nw")
    main.statusMsg.config(text = data)

@sio.on("serverDenied")
def serverDenied(data):
    print(data)

    global userData
    userData = {"failure" : "failure"} #To break wait loop in tkinter

    if data == "Transaction failed\n for unknown reasons!": main.reset("Bip_Bar")
    elif data == "FALSE CHIP-ID\n Public": main.reset("Bip_Bar")
    elif data == "FALSE USER-ID\n Public": main.reset("Bip_Bar")
    elif data == "FALSE CHIP-ID\n Private": main.reset("Edit_User")
    elif data == "FALSE USER-ID\n Private": main.reset("Edit_User")
    elif data == "User edit failed!": main.reset("Edit_User")
    elif data == "New user failed!": main.reset("New_User")
    else: main.reset("all") #Reset all windows

    userData = {} #reset

    main.statusCanvas.delete("all")
    main.statusCanvas.create_image(2, 2, image = main.errorIcon, anchor = "nw")
    main.statusMsg.config(text = data)

#print("Connecting to server... please wait up to 60 seconds")
#sio.connect('https://froemosen.pythonanywhere.com', wait_timeout = 60) #Connect to
online server
sio.connect('http://127.0.0.1:5000/') #Connect to local server

base.mainloop() #TKINTER MAIN LOOP

```



## NFC-Reader.py

```
#Testprogram til NFC-reader. Brugt i Main.py
import nfc
import winsound

clf = nfc.ContactlessFrontend()

data = None

#Check NFC-reader connection
try:
    assert clf.open('usb') is True #Determined in cmd by command: "python -m nfc"
    print(f"Using device: {clf}")

except AssertionError:
    print("NFC-reader not set up correctly. Try again! (Maybe the error is in the code!)")
    clf.close()  #Test over - New connection will be needed

def getData():
    tag = clf.connect(rdwr={'on-connect': lambda tag: False})
    print("Tag found!")
    print(tag)

    print(tag.identifier)

    if not tag.ndef == None:
        winsound.Beep(frequency=2000, duration=400) #Beep-sound (Make this on the nfc-reader if possible)
        for record in tag.ndef.records:
            print(record)
        return(tag.ndef.records)

    else:
        print(tag.dump())
        print("Card Carries No Data!\n")
        return(None)

while True:
    try:
        if data == None:
            data = getData()

        else:
            #Do stuff with data
            data = None

    except Exception as errorCode:
        print(errorCode)
        print("Exception caught - Closing device")
        clf.close()  #Test over - New connection will be needed
        break
```

## Pickletest.py

```
import pickle

favcolor = pickle.load( open( "dbbb", "rb" ) )
print(favcolor)
```

## Camera test.py

```
import tkinter as tk
from tkinter import ttk
import cv2
import PIL.Image, PIL.ImageTk
from tkinter import font
import time

class Application(tk.Frame):
    def __init__(self,master, video_source=0):
        super().__init__(master)

        self.master.geometry("700x700")
        self.master.title("Tkinter with Video Streaming and Capture")

        # -----
        # Font
        # -----
        self.font_frame = font.Font( family="Meiryo UI", size=15, weight="normal" )
        self.font_btn_big = font.Font( family="Meiryo UI", size=20, weight="bold" )
        self.font_btn_small = font.Font( family="Meiryo UI", size=15, weight="bold" )

        self.font_lbl_bigger = font.Font( family="Meiryo UI", size=45, weight="bold" )
        self.font_lbl_big = font.Font( family="Meiryo UI", size=30, weight="bold" )
        self.font_lbl_middle = font.Font( family="Meiryo UI", size=15, weight="bold" )
        self.font_lbl_small = font.Font( family="Meiryo UI", size=12, weight="normal" )

        # -----
        # Open the video source
        # -----
        self.vcap = cv2.VideoCapture( video_source )
        self.width = self.vcap.get( cv2.CAP_PROP_FRAME_WIDTH )
        self.height = self.vcap.get( cv2.CAP_PROP_FRAME_HEIGHT )

        # -----
        # Widget
        # -----
        self.create_widgets()
```

```

# -----
# Canvas Update
# -----

    self.delay = 15 #[mili seconds]
    self.update()

def create_widgets(self):

    #Frame_Camera
    self.frame_cam = tk.LabelFrame(self.master, text = 'Camera',
font=self.font_frame)
    self.frame_cam.place(x = 10, y = 10)
    self.frame_cam.configure(width = self.width+30, height = self.height+50)
    self.frame_cam.grid_propagate(0)

    #Canvas
    self.canvas1 = tk.Canvas(self.frame_cam)
    self.canvas1.configure( width= self.width, height=self.height)
    self.canvas1.grid(column= 0, row=0,padx = 10, pady=10)

    # Frame_Button
    self.frame_btn = tk.LabelFrame( self.master, text='Control',
font=self.font_frame )
    self.frame_btn.place( x=10, y=550 )
    self.frame_btn.configure( width=self.width + 30, height=120 )
    self.frame_btn.grid_propagate( 0 )

    #Snapshot Button
    self.btn_snapshot = tk.Button( self.frame_btn, text='Snapshot',
font=self.font_btn_big)
    self.btn_snapshot.configure(width = 15, height = 1,
command=self.press_snapshot_button)
    self.btn_snapshot.grid(column=0, row=0, padx=30, pady= 10)

    # Close
    self.btn_close = tk.Button( self.frame_btn, text='Close', font=self.font_btn_big
)
    self.btn_close.configure( width=15, height=1, command=self.press_close_button )
    self.btn_close.grid( column=1, row=0, padx=20, pady=10 )

def update(self):
    #Get a frame from the video source
    _, frame = self.vcap.read()

    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    self.photo = PIL.ImageTk.PhotoImage(image = PIL.Image.fromarray(frame))

    #self.photo -> Canvas

```

```

        self.canvas1.create_image(0,0, image= self.photo, anchor = tk.NW)

        self.master.after(self.delay, self.update)

    def press_snapshot_button(self):
        # Get a frame from the video source
        _, frame = self.vcap.read()

        frame1 = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        cv2.imwrite( "frame-" + time.strftime( "%Y-%d-%m-%H-%M-%S" ) + ".jpg",
                    cv2.cvtColor( frame1, cv2.COLOR_BGR2RGB ) )

    def press_close_button(self):
        self.master.destroy()
        self.vcap.release()

def main():
    root = tk.Tk()
    app = Application(master=root)#Inherit
    app.mainloop()

if __name__ == "__main__":
    main()

```

## entryWithPlaceHolder.py<sup>13</sup>

```

#LIBRARY FROM HERE
https://stackoverflow.com/questions/27820178/how-to-add-placeholder-to-an-entry-in-tkinter
import tkinter as tk

class EntryWithPlaceholder(tk.Entry):
    def __init__(self, master=None, placeholder="PLACEHOLDER", color='grey', width=20):
        super().__init__(master, width = width)

        self.placeholder = placeholder
        self.placeholder_color = color
        self.default_fg_color = self['fg']

        self.bind("<FocusIn>", self.foc_in)
        self.bind("<FocusOut>", self.foc_out)

        self.put_placeholder()

    def put_placeholder(self):

```

---

<sup>13</sup> Kode lavet af bruger NAE på Stackoverflow(Se litteraturliste)

```
    self.insert(0, self.placeholder)
    self['fg'] = self.placeholder_color

def foc_in(self, *args):
    if self['fg'] == self.placeholder_color:
        self.delete('0', 'end')
        self['fg'] = self.default_fg_color

def foc_out(self, *args):
    if not self.get():
        self.put_placeholder()

if __name__ == "__main__":
    root = tk.Tk()
    username = EntryWithPlaceholder(root, "username")
    password = EntryWithPlaceholder(root, "password", 'blue')
    username.pack()
    password.pack()
    root.mainloop()
```