



A V  F Y . M E



Avafy

- BLIV GJORT OPMÆRKSOM PÅ NÆRLIGGENDE UHELD

62522 Mobilapplikationsudvikling - Android F17

Afleveringfrist: 16-5-2017

Gruppe: 1

Medlemmer:

Adis Begic - s133889

Benjamin Fester Henningsen - s124317

Kontekst	3
Begrænsninger	3
Principper	4
For at kunne levere et velfungerende og effektivt produkt er der opsat nogle principper som udgangspunkt for udviklingen og fremgangsmåden.	4
Funktionel oversigt	5
1.1 Brugsmønstre	5
1.1.1 Havari	5
1.1.2 Glatbane**	6
1.1.3 Udrykning**	6
1.2 Brugergrænseflade	6
1.3 Skærmbilleder	7
Softwarearkitektur	10
Afprøvning	11
Enheds specifikke undersøgelser:	12
Netværksbaseret undersøgelser:	13
Applikations relateret undersøgelser:	14
Konklusion	15

Kontekst

Moderne biler genererer store mængder data der kan benyttes mere optimalt, hvis det sendes til det rigtige sted på det rigtige tidspunkt. Avafy er en Android mobil applikation der skal hjælpe bilister i trafikken med at være opmærksomme på eventuelle ulykker som er på den samme rute som de selv kører ved at få biler til at kommunikere. Hermed kan sikkerheden i trafikken øges, hvis bilister får en advarsel i god nok tid.

Dette kan lade sig gøre ved at identificere relevant data der bliver genereret af en bil ved diverse aktioner og levere den til bilister på relevante lokationer. Information vedrørende uheld, glatte veje, udrykning, bilkøer, huller etc. kan, hvis leveret i tide nok, øge sikkerheden på vejene.

Vidensdeling mellem biler er dermed hovedfokus i dette projekt og ideen kan implementeres i bilens egen software i fremtiden. Problemformulering til projektet lyder således:

Hvordan kan man via CAN-bus datadeling mellem biler, øge sikkerheden på vejene?

Begrænsninger

Under udviklingen af vores applikation, har vi måttet være nødsaget til at tage forbehold for nogle essentielle begrænsninger som vi har måttet acceptere og arbejde ud fra. Disse begrænsninger er beskrevet herunder:

- Tidsmæssigt har det været lidt en udfordring at arbejde med projektet, da det har krævet at man booker bilen på forhånd. Herudover er der altid andre der også booker bilen så det ikke altid man kan få den når det passer en selv.
- Der er kun 1 Smart bil i DTU's garage, hvilket vil sige at man ikke rigtigt kan teste applikationens formål med flere bilister.
- Kun en velfungerende OBDLink MX til rådighed, resten er enten defekte eller af andet mærke hvilket er imod krav.
- OBDLink MX sender med ~100 PIDs/sekund for Android, en hastighed så hurtig at de fleste android telefoners buffers vil blive fyldt straks som besværliggøre hacking af bilens data.

- Det er umuligt at emulere en virtuel maskine i Android Studio der indeholder en Bluetooth adapter så alle vores tests/undersøgelser har derfor foregået tilkoblet til bilen.
- Applikationen skulle udvikles til det mobile operativsystem Android og virke på enheder til og med nyere software version end 5.0.
- Enheds mæssigt så er applikationen kun blevet eksperimenteret med/testet på Samsung s5, s6, s8 samt OnePlus.
- Ud af de 5 mulige kommunikations protokoller der er understøttet af OBD-II, er vi nødsaget til at bruge ISO15765-4 (CAN) da vi har med en elektrisk smart bil at gøre.

Principper

For at kunne levere et velfungerende og effektivt produkt er der opsat nogle principper som udgangspunkt for udviklingen og fremgangsmåden.

- Angående vores design er der simple og klare guidelines. Farvetemaet, rød og hvid der er genkendeligt fra trafikken, skal gå igennem alle aktiviteter, fragmenter, dialoger o.l. Simple menuer, knapper og tekstfelter af nævnte, der gør applikationen overskuelig og intuitiv.
- Der bliver ikke benyttet funktioner til at lave animationer på knapper inde i appen. Dette er for at gøre applikationen så let og flydende som muligt. Derfor har vi forsøgt at gøre temaet, logoet og knapperne både neutrale og pæne, uden at gå på kompromis med hastighed og funktionalitet.
- Brugere skal optimalt kun parre med OBD modulet en enkelt gang - første gang den tages i brug. Derefter oprettes forbindelsen hurtigere næste gang. Dette gør det hurtigere at benytte vores app hver gang brugeren sætter sig i bilen og skal indtaste en rute.
- Måden (protokollen/metoden) hvorpå OBD modulet og telefonen kommunikerer er der faste regler for hvordan skal udføres. Dette gøres med henblik på at få en sikker og stabil kommunikation af den real-time data der skal overføres mellem enhederne. I forlængelse af dette skal der på samme måde være en stabil forbindelse mellem telefon og Firebase database.

- Advarselsbeskeden der kommer når bilisten skal være opmærksom på eventuelle skader på vejen skal være en AlertDialog.
- Avafy benytter ikke splash screens for at minimere ventetiden før applikationen starter og samtidigt gøre den lettere.
- Information og data vedrørende vores brugere skal forblive sikkert og utilgængeligt for uvedkommende. Relevant information og data skal dog deles og distribueres til det rigtige formål for at øge sikkerheden for trafikanter.

Funktionel oversigt

1.1 Brugsmønstre

Følgende afsnit vil belyse hvilke problemer applikationen ville kunne hjælpe med at løse. Dette vil blive eksemplificeret gennem tre brugsmønstre, hvoraf én er blevet udviklet mens de to andre skal ses som udvidelser til den kommende fremtid og er derfor markeret med **.

1.1.1 Havari

Vi vil først tage et kig på et scenerie som involverer brugen af havariblink. En familie er desværre kørt i grøften efter at have undvejet en spøgelsesbilist og kan nu ikke få deres bil fri. Det er sket på en uklar, mindre belyst, landevej og der kommer biler kørende i fuld fart i begge retninger. Hændelsen er sket meget pludseligt og de hurtigt kørende bilister vil ikke rigtigt blive opmærksomme på nogle usædvanlige hændelser, medmindre familien tænder for havariblinket eller placere en trekant ude på vejen.

Selve situationen kræver en hurtig reduktion af hastighed for bilisternes vedkommende og en hurtig handling fra familien af. En pludselig opbremsning af en hurtigt kørende bil på en tungere belastet vej, er meget farligt og det øger risikoen for ulykker drastigt. Havariblinket eller trekanten, bruges her til at signalere at der desværre er sket noget med køretøjet. Desværre er problemet bare at fornævnte signaler oftest ikke er synlige nok, hurtigt nok. Med vores applikation vil man blive advaret fra en specifik afstand og på forhånd være informeret om at der er sket en hændelse i nærheden.

1.1.2 Glatbane**

Det er en varmere dag midt i November med meget kulde. Vinteren har ikke rigtigt ramt Danmark dette år og det har derfor ikke været nødvendigt at skifte til vinterdæk, selvom loven bemander dem påført. Kombinationen af varme og kulde har gjort at der ligger sort is på vejene som selv for det vågne øje næsten er umuligt at se. Du kører langs Frederikssundsvej mod DTU Ballerup og nærmer dig det sidste kryds. Der er massere cyklister og du skal snart dreje til højre.

100 meter inden du drejer til højre bliver du gennem applikationen gjort opmærksom på, at der ligger sort is på vejen og ved derfor at du skal kører ekstra langsomt igennem svinget, for ikke at miste kontrol over dit køretøj.

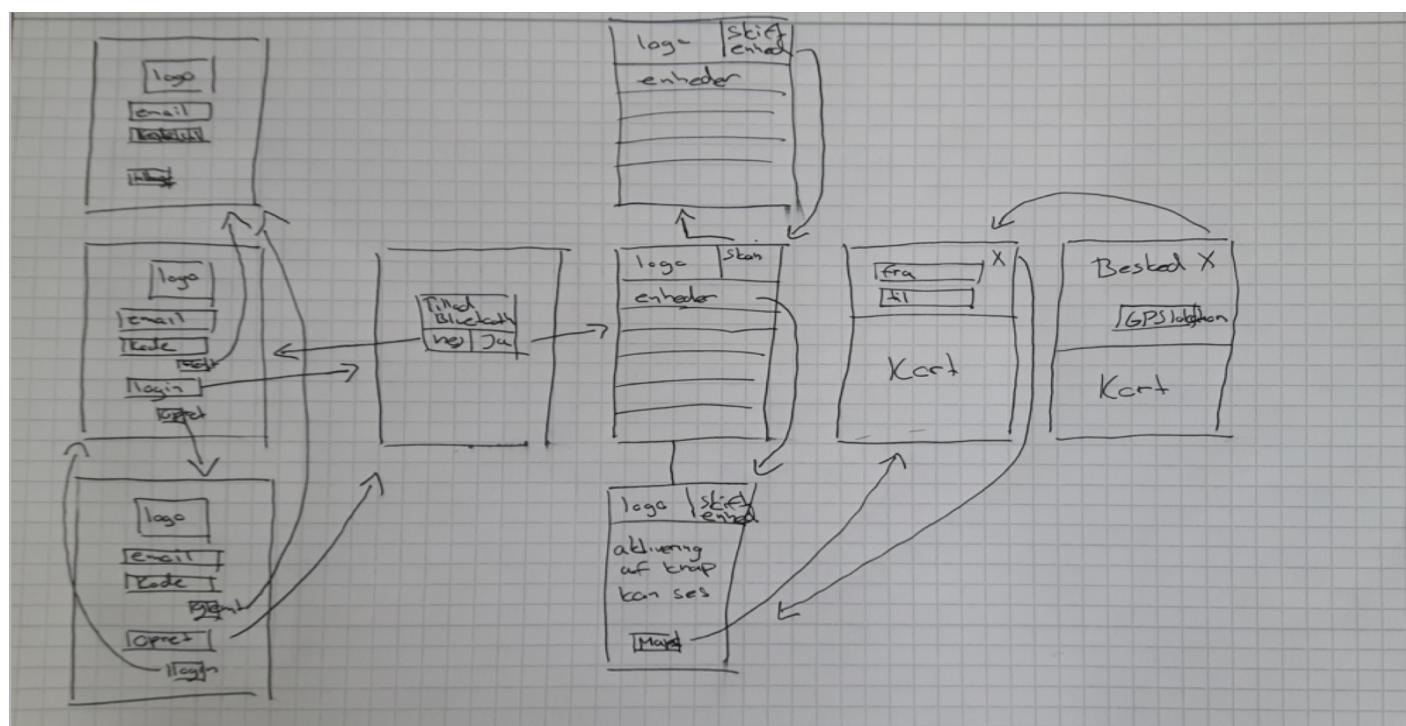
1.1.3 Udrykning**

Du er nu på vej hjem fra DTU. Du kører endnu engang ude på Frederikssundsvej men nu mod København. Klokken er 16.00, din radio er tunet ind på The Voice, hvor der afspilles Top 30 hits af Chriz & Heino. Du elsker hvad du hører og skruer helt op for volumen. Du kan nu ikke hører dine omgivelser, da din bil er ekstra godt isoleret udefra og dine højtalere yderst gode. Du kører stille og roligt i en langsom kø der bevæger sig meter for meter.

Et udrykningskøretøj er på vej bagfra i høj hastighed, andre bilister flytter sig men din opmærksomhed er et helt andet sted. Du modtog en påmindelse igennem vores applikation om at udrykningskøretøjet kører på samme veje som dig og derudover at det nærmer sig, hvilket resultere at du kan handle ligesom de andre bilister.

1.2 Brugergrænseflade

Efter brugsmønstrene var blevet fastlagt og undersøgelserne færdiggjort, var tiden inde til at visualisere vores resultater. Vi startede med at udarbejde konceptuelle wireframes, hvilket tillod os at have nogle fælles retningslinjer at følge. Dette gjorde arbejdet meget mere transparent og fastlagde skelettet for applikationens brugergrænseflade.

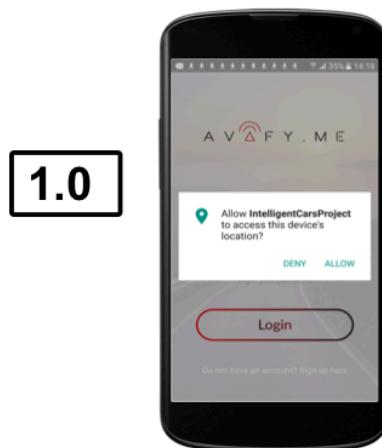


(Ovenstående billede illustrerer wireframe af konceptuel tankegang)

Herefter skabte vi et visuelt tilfredsstillende moderne design i 'Sketch' hvilket er et software design værktøj målrettet brugergrænseflade design. Da det endelig design var blevet fastlagt brugte vi dem som retningslinjer til at omsætte det til XML i Android Studio.

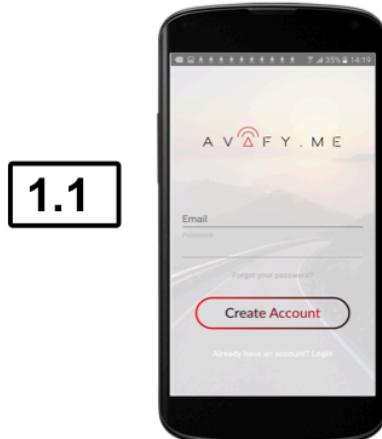
1.3 Skærmbilleder

Nedenstående ses en oversigt over applikations endelige brugergrænseflade design. Til højre for skærmbilledet vil de kunne finde en beskrivelse af skærmens funktionalitet.



1.0

For at anvende applikationens fulde funktionalitet korrekt, kræves det at man tillader applikationen adgang til enhedens placering ved at trykke 'Allow'.



1.1

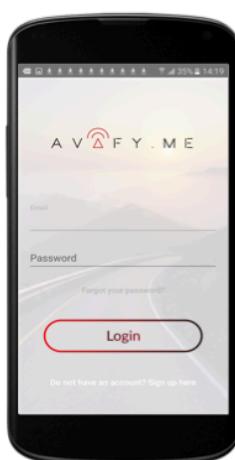
Oprettelse af bruger finder sted ved at indtaste sin email adresse samt kodeord og derefter trykkelse på 'Create Account'. Dette er essentielt for applikationen da det tillader os at adskille brugere fra hinanden.

1.2



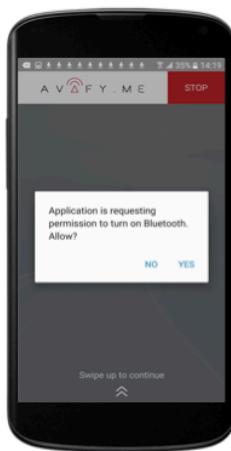
**Hvis du har glemt dit kodeord men vil
bruge samme email, nulstil da dit kodeord.
Dette gøres ved at indtaste email adresse
og derefter trykkelse på 'Reset Password'.**

1.3



**Indtast nu dine legitimations- oplysninger
som du tidligere oprettede for at kunne
anvende appen og tryk derefter 'Login'.**

1.4



**Da det er en Bluetooth baseret applikation,
er det altså altid vigtigt at der svarer 'JA' til
den fremtrædende dialog boks. Dette vil
slå Bluetooth til på din enhed. Hvis der
svares 'No' vil applikationen aflevere
brugeren tilbage til login skærmen.**

1.5



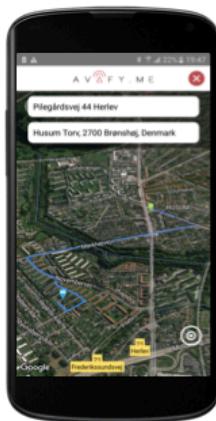
**Når man er forbundet til Bluetooth er
næste skridt at skanne for enheder. Når
enheden er fundet, klikkes der simpelthen
bare på enheden og der vil så blive oprettet
en parring imellem de to enheder.**

1.6



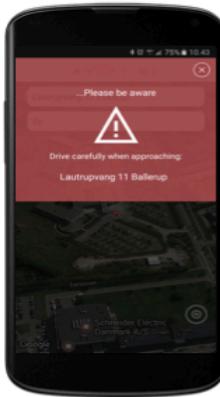
Det vil herefter blive vist på skærmen om dataforbindelsen er igangsat, hvilket gerne skulle efterlade et tomt felt og ved aktivering af havari-knappen vise en besked om at det er slæt til. Hvis andet vises er man desværre ikke forbundet korrekt og man må derfor trykker på 'change device' for at opdatere forbindelsesproceduren.

1.7



Ved at swipe op, kan man igangsætte en rutevejledning og få vist hvilken vej man skal kører. På kortet vil der blive vist igennem iconer; Hvor man er, hvor man skal hen, hvor man starter fra, samt hvis der er en aktiv havari i nærheden.

1.8

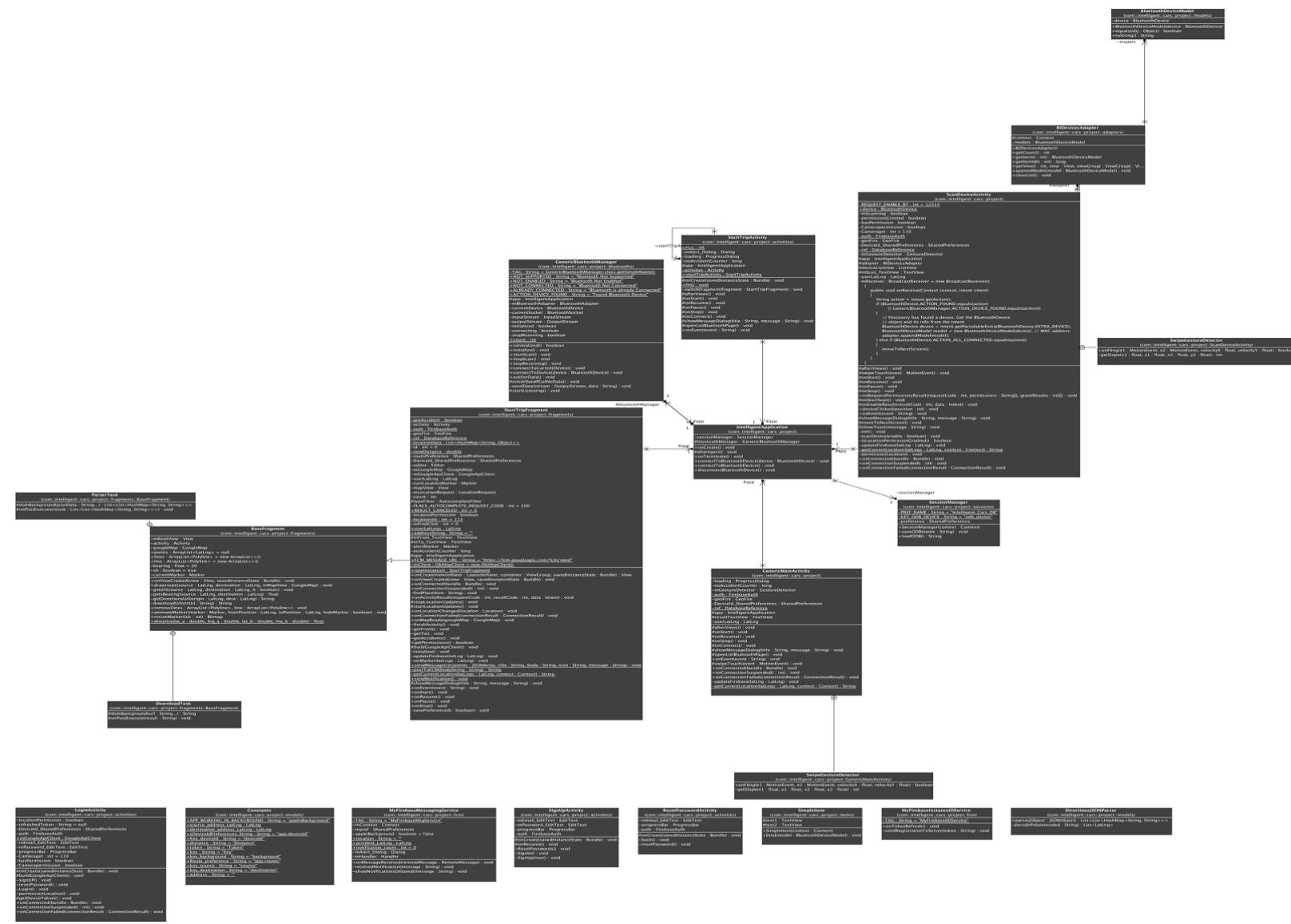


Når Havari knappen trykkes i bilen, vil nærliggende brugere modtage en besked om ovenstående samt placeringen på hændelsen. Udover beskeden, vil et havari ikon også blive vist på ens rute/kort. Bruger'en vil blive opmærksomgjort på beskeden igennem en lyd og kan derfra selv vælge om de vil åbne beskeden eller ikke.

Softwarearchitektur

Nedenstående billede er et UML klassediagram. De mørkegrå kasser repræsenterer her klasser og deres dertilhørende attributter og operationer/metoder.

(For at se billedet tydeligt se AvafyUML.png vedlagt i projekt mappen.)



Den første klasse som brugeren støder på er *SignUpActivity*. Denne klasser er uafhængig af andre klasser, da information vedrørende brugerens e-mail og password ikke skal benyttes efterfølgende. Det samme gælder for *LoginActivity*, *ResetPasswordActivity*, *Constants*, *MyFirebaseMessagingService*, *SimpleItem*, *MyFirebaseInstanceIdService* og *DirectionsJSONParser*. Disse klasser er ikke forbundet med pile da de ikke afhænger af andre klasser og ingen klasser afhænger af dem.

IntelligentApplication er klassen der holder et singleton objekt der skal være tilgængeligt fra overalt. Singleton objektet bliver oprettet som `sessionManager`.

SessionManager er den klasse der holder vedvarende information vedrørende OBD modulet igennem hele applikationen, selvom applikationen lukker. Derfor anvendes SharedPreferences. *SessionManager* klassen er ejet af *IntelligentApplication* klassen.

Klassen *ScanDeviceActivity* opretter en BroadcastReceiver der kan finde signaler fra enheder med Bluetooth. Herfra nås klassen *BtDevicesAdapter*, der viser en liste over tilgængelige Bluetooth enheder. Fra *BtDevicesAdapter* nås klassen *BluetoothDeviceModel* der returnerer navnet på den valgte enhed med Bluetooth. *ScanDeviceActivity* er ikke navigerbar fra *BtDevicesAdapter* og *BtDevicesAdapter* er ikke navigerbar fra *BluetoothDeviceModel*.

ScanDeviceActivity og *GenericMainActivity* bliver realiseret af *SwipeGestureDetector* klassen (interface realization).

GenericBluetoothManager er klassen som står for håndtering af Bluetooth enheder.

GenericMainActivity er den klasse der viser respons fra OBD modulet.

DownloadTask og *ParserTask* klasserne henholdsvis downloader lokation fra Google API og bliver efterfølgende parset til java syntaks der kan benyttes. Disse to klasser er forudsætning for at *BaseFragment* kan igangsættes. Igen er der her tale om en interface realization. *BaseFragment* er selve ruten på kortet samt en udvidelse af *StartTripFragment*.

StartTripFragment er klassen der viser kortet samt giver mulighed for at zoome ind på nuværende placering ved tryk på ikonet lignende en hvid skydeskive.

StartTripActivity er den klasse der starter *StartTripFragment*. *StartTripActivity* er afhængig af *IntelligentApplication* klassen der indeholder singleton objektet. Derudover står førstnævnte aktivitet også for at åbning af Alertdialog.

Afprøvning

Eftersom applikationen er Bluetooth baseret, kan man ikke foretag automatiserede tests igennem android eller anvende android test frameworks som f.eks. Robotium. Dette skyldes at virtuelle maskiner ikke har en Bluetooth adapter, hvilket vil sige at Bluetooth ikke kan tilsluttes. Herudover, er der også tale om en applikation som skal forbindes med hardware, hvilket yderligere umuliggøre en fuldt automatiseret test af applikationen. En fuldt automatiseret test vil kun teste *LoginActivity*, *SignUpActivity*, samt *ResetPasswordActivity* aktiviteterne hvorefter testen vil være fuldendt, hvilket ikke er tilstrækkeligt nok. I forlængelse af ovenstående har det betydet at applikationen er blevet testet manuelt igennem brugertest.

Nedenstående tjekliste er blevet lavet til at teste de væsentlige karakteristikker af applikationen.

Enheds specifikke undersøgelser:

#	Beskrivelse	B/IB	Bemærkninger
1.1	Kan applikationen installeres på enheden?	B	Applikationen installeres uden problemer.
1.2	Fortsætter applikationen som den skal efter enheden har været låst?	B	Efter at havde været låst, virker applikationen som tidligere.
1.3	Modtager applikationen data fra GPS sensoren i enheden korrekt?	B	GPS data fra sensoren er modtaget korrekt.
1.4	Opfører applikationen sig som den skal når lyden er slået fra enheden?	B	Uden lyd opfører applikationen som den skal. Anbefales ikke da lyd giver påmindelse om besked, hvilket er hoved funktionaliteten af applikationen.
1.5	Kan applikationen blive af-installeret fra enheden?	B	Applikationen afinstalleres helt uden problemer.
1.6	Virker applikationen efter en frisk installation på enheden som den skal?	B	Applikationen virker perfekt efter frisk installation.

Netværksbaseret undersøgelser:

#	Beskrivelse	B/I/B	Bemærkninger
2.1	Opfører applikationen sig som den skal når den er forbundet til internettet igennem Wi-Fi?	B	Applikationen virker som den skal når forbundet igennem Wi-Fi.
2.2	Opfører applikationen sig som skal når den er forbundet til internettet igennem 4G/3G?	B	Applikationen virker som den skal når forbundet igennem 4G/3G.
2.3	Opfører applikationen sig som den skal når den er uden for netværks rækkevidde?	B	Google Maps services virker ikke uden for netværks rækkevidde, hvilket er bevidst.
2.4	Fortsætter applikationen automatisk når man er inden for netværks rækkevidde igen?	B	Google Maps services er anvendelige igen når inden for netværks rækkevidde.
2.5	Opdatere applikationen automatisk netværksorienteret handlinger i applikationen ved tilbage etableret forbindelse?	B	Google Maps opdateres automatisk når applikationen har internet etableret forbindelse.
2.6	Forbindes applikations enheden med OBD-enheten over Bluetooth?	B	Applikationen forbindes med OBD-enheden over Bluetooth som den skal.
2.7	Forbindes applikations enheden med OBD-enheten over Bluetooth automatisk når de to enheder er indenfor rækkevidde?	B	Automatisk etablering af forbindelse sker når de to enheder er inden for rækkevidde.

2.8	Virker dataoverførslen ved havari aktivering over Bluetooth automatisk ALTID, når de to enheder er indenfor rækkevidde?	IB	Der kan opstå forstyrrelse hos OBD-enhed hvilket kræver at man nulstiller den ved at holde den ene knap som der er på OBD-en inde til forbindelses lampen på enheden blinker.
-----	---	----	---

Applikations relateret undersøgelser:

#	Beskrivelse	B/IB	Bemærkninger
2.1	Er applikationen blevet afprøvet på forskellige type enheder og forskellige versioner af android operativsystem?	B	Applikationen er blevet gennemtestet på Samsung Galaxy S5 Neo, Samsung Galaxy S6, samt OnePlus One. Ergo er den blevet testet på v.5.0.1 +
2.2	Virker swipeop- gesturen som den skal når personen hiver fingeren op?	B	Swipeop- gesturen virker som den skal når brugeren skal ind i aktiviteten indeholdende Google Maps.
2.3	Renser applikationen den indkommende data fra havari knappen når knappen slukkes?	B	Når havari knappen slukkes, skyldes data. Ergo forsvinder data.
2.4	Kan brugeren gå tilbage til tidligere aktivitet igennem en eventuel lukning/tilbage knap?	B	Brugeren kan vende tilbage til tidligere aktiviteter igennem luknings/tilbage knap for det er ment at det skal fungere.
2.5	Er det tydeligt fra første anvendelse hvad hoved funktionaliteten af applikationen er ?	B	Det er tydeligt fra start at det er en Bluetooth baseret applikationen der kræver en OBD for at kunne blive anvendt. Hvis man svarer nej til en fremtrædende dialog om tilladelse til at tænde for Bluetooth, vil man blive hjemsendt til login aktiviteten.

2.6	Hvis ydeevnen er langsom, bliver der så vist en status med en meddelelse om den igangværende handling?	B	Der bliver vist en status. Eksempelvis når man til at modtage data fra havari knappen vil der blive vist "Initializing data".
2.7	Svarer applikationen som den skal til ændringer i enheds orientering ifølge designet?	B	Applikationen er kompatibel i både landskab og porträts tilstand.
2.8	Handlinger i applikationen er ikke blevet genskabte som har en standard mening (f.eks. swipe fra toppen til bunden for at se notifikationer)	B	Det er undladt at ændre i standard handlinger som eksempelvis, swipe fra top til bund.

Konklusion

Vi har formået at bekræfte 'proof of concept' igennem en fungerende prototype. Dog ville vi også gerne have implementeret vores udvidelser iht. glatbane og udrykning men løb desværre tør for tid, da vi hele tiden støtte på nye problemer under udviklingen af applikationen. Derudover, lærte vi i praksis hvor drilsk det kan være at arbejde med dynamisk data og at man virkelig skal gennemtænkte ens næste træk når man arbejder med trådløs kommunikation, da der let kan forekomme forstyrrelser hvis man arbejder med flere teknologier.

I fremtiden ville det være oplagt at implementere glatbane og udryknings scenariet i applikationen men sikkert også mange andre scenarier. Det kunne også være smart at lave en HUD-Display version af applikationen som kan blive vist mod front glasset i bilen da 'Augmented Reality' begynder at blive implementeret mere og mere i de nye, dyrere biler.

Kurset har været lærerigt og har givet den nødvendige baggrundsviden for at selv kunne lave en applikation. Programmering i sig selv er et teknisk fag og man lærer derfor bedst i praksis. I fremtiden ville det måske være en ide at starte timen ud med at lave en multiple choice opsummerings test på CampusNet som tager max 10 minutter, hvor man svarer på multiple choice spørgsmål fra emnet gennemgået forrige gang. Dette vil give eleverne en bedre forståelse men også læreren et indblik af hvor godt klassen er med.

Udarbejdelsen af projektet har haft sine -op og nedture samt været meget tidskrævende men i sidst ende har det været en god oplevelse.