

AR-säkerhetskarta

Augmented reality-applikation för identifiering av
objekt i fordon

Cecilia Bergman
Fredrik Burmester
Hugo Dahl
Johnny Elmér
Simon Nilsson
Adrian Szuter

Examinator: Daniel Jönsson

Sammanfattning

När en fältarbetare inom räddningstjänsten anländer till en olycksplats, där en eller flera fordon har krockat, är det ytterst viktigt att veta hur de på ett säkert sätt ska kunna ta sig in i fordonen och eventuellt rädda livet på de olycksdrabbade. I nuläget används 2D-kartor för att identifiera viktiga objekt i fordon, som kan vara till nytt under räddningsprocessen. Denna metod begränsar fältarbetaren och kräver att fältarbetaren kan tyda kartan och sedan applicera det den ser på det verkliga fordonet, vilket kan ta tid och vara svårt.

Med hjälp av en AR-applikation som visar var dessa viktiga objekt befinner sig i fordonet, genom att visa en fordons-modell i 3D, skulle tiden det tar att rädda personer kunna minskas, jämfört med om 2D-kartor eller inga kartor används. Därför försöker denna rapport svara på om tiden det tar att identifiera objekt i fordon kan förkortas genom användning av en AR-applikation på olycksplatsen.

Utveckling av AR-applikationen skedde i Unity med Vuforia AR-bibliotek. Applikationen skulle tillåta användaren att söka upp ett registreringsplåtsnummer i en databas och visa tillhörande 3D-modell av fordonstypen genom kameran på en surfplatta. Personen kan sedan gå runt 3D-modellen som om den stod där i verkligheten och lättare identifiera exakt var objekt befinner sig. Applikationen är väl anpassad för att enkelt kunna användas även med vanar eller av personer med defekt färgseende genom att nyttja stora knappar med tydliga färger.

Tyvärr begränsades projektet av kunskap, Vuforia och tid vilket ledde till att endast en enda modell kunde visas istället för att användaren kunde söka efter en registreringsplåt. Detta begränsade dock inte resultatet av projektet då applikationen fungerade exakt som tänkt om den modellen valdes.

Efter utveckling av applikationen skedde användartester. Där undersöktes tid till identifiering (av ett objekt i fordonet), med hjälp av AR-modellen samt en 2D-karta. Jämförelse gjordes därefter för att avgöra vilken metod som gick snabbast att utföra. Vi såg då att tiden till identifiering av ett objekt (efter att AR-modellen placerats i applikationen) förkortas och förenklas jämfört med användning av en 2D-karta. Dock tog det extra tid för användaren att placera ut AR-modellen i applikationen, vilket ökade den totala tiden till identifiering. Detta betyder att om användaren är utbildad i applikationen, och kan därmed förkorta tiden det tar att placera ut AR-modellen, kan användning av denna typ av applikation främja räddningsprocessen av människoliv i bilolyckor, då tiden till identifiering av viktiga objekt förkortas.

Det finns dock tydliga begränsningar för en applikation som kräver en kamera. De är beroende av starkt ljus för att kameran ska kunna se och identifiera omgivningen. Detta betyder att applikationen ej kan användas på natten om inte hjälpmittel används som till exempel lampor eller UV-kamera. Utöver detta behöver användaren en plan yta för att kunna placera ut AR-modellen vilket begränsar användningen.

Slutsatsen av denna rapport är att användning av en AR-applikation i detta syfte kan underlätta identi-

fiering av viktiga objekt i ett fordon, samt förkorta tiden det tar att identifiera objekten. Applikationen bör dock vara ett komplement till 2D-kartor då det finns väldigt tydliga begränsningar. Användaren bör också vara utbildad i applikationen för att den faktiskt skall kunna underlätta för fältarbetaren.

Innehåll

Sammanfattning	i
Figurer	v
1 Introduktion	1
1.1 Bakgrund	1
1.2 Syfte	1
1.3 Frågeställning	2
1.4 Avgränsningar	2
2 Relaterat arbete	3
3 AR-applikation i Unity med hjälp av Vuforia	5
3.1 Design och prototyp	5
3.2 Användartester	6
3.3 Plattform och biblioteksväljning	6
3.3.1 Lagring av modeller	7
3.4 Appdesign och funktionalitet i Unity med UI-toolkit	9
3.5 AR-implementation och 3D-modellering	11
3.6 Systemtester	11
4 Resultat	12
4.1 Applikation	12
4.2 Användartester	16
4.3 Systemtester	17
5 Analys och diskussion	18
5.1 Metod	18
5.1.1 Användartester	18
5.1.2 Användargränssnitt och funktionalitet	18
5.1.3 Systemtester	19
5.1.4 Källkritik	19

5.2 Resultat	19
5.2.1 Hantering av 3D-modeller och databas	19
5.2.2 Användartester	19
5.2.3 Användsgränssnitt och funktionalitet	20
5.2.4 Systemtester	20
5.3 Etisk och samhällelig reflektion	20
6 Slutsatser	22
6.1 Hur kan man försäkra sig om en snabb responstid av systemet, med storleken på datafilerna i åtanke?	22
6.2 Kan man, med hjälp av AR, underlätta för en fältarbetare inom Räddningstjänsten Östra Götaland (RTÖG) att identifiera kritiska delar hos ett fordon?	22
6.3 Vidareutvecklingsmöjligheter	23
A Reflektion över systemutvecklingsprocessen	26
A.1 Cecilia Bergman	28
A.2 Fredrik Burmester	28
A.3 Hugo Dahl	28
A.4 Johnny Elmér	28
A.5 Simon Nilsson	29
A.6 Adrian Szuter	29
B Individuella bidrag	31
B.1 Cecilia Bergman	31
B.2 Fredrik Burmester	31
B.3 Hugo Dahl	32
B.4 Johnny Elmér	32
B.5 Simon Nilsson	32
B.6 Adrian Szuter	32

Figurer

2.1 Bild från Moditech Rescue Solutions som visar hur en 2D-karta för ett fordon ser ut.	4
3.1 En del utav applikationsdesignen från Figma som visar hur de tänkta sidorna i applikationen ska se ut.	5
3.2 Pseudokod för en fordons-klass i C# som gör det möjligt att länka modell och registreringsplåtnummer.	7
3.3 Pseudokod för en databas-klass med hjälpfunktioner i C#. Den här funktionen möjliggör hämtning av en modell-klass från databasen via antingen ID eller registreringssnummer.	8
3.4 Pseudokod för definition av en databas-klass i C#. Här definieras en klass med en lista som förvarar alla modell-klass-objekt.	8
3.5 Kod för applikationens meny-script som gör det möjligt att öppna och stänga menyn samt byta ikon beroende på dess tillstånd.	9
3.6 Kod för applikationens placering-script som gör det möjligt att placera ut en modell på ett plan.	10
3.7 Kod för applikationens skalning- och rotations-script som gör det möjligt att skala och rotera en modell.	10
4.1 Applikationens startsida som visas när applikationen startas. Användaren går vidare till nästa sida genom att trycka på den gröna 'Jag är redo'-knappen.	12
4.2 Applikationens vy för inmatning av registeringsnummer där man kan skriva in ett registeringsnummer och gå vidare genom att klicka på 'Klar'.	13
4.3 Applikationens vy för valet av karttyp där '2D' leder till 2D-kartan och 'AR' leder till AR-vyn.	14
4.4 Applikationens vy för 2D-kartan där man kan klicka på 'Sök igen'-knappen nere i högra hörnet för att komma till vyn i figur 4.2.	14
4.5 Applikationens AR-vy med stängd meny där man kan öppna menyn genom att klicka på knappen nere i hörnet.	15
4.6 Applikationens AR-vy med menyn öppen där man kan placera ut fordonet, söka igen och stänga menyn med respektive knappar.	15
4.7 Applikationens AR-vy med bilen utplacerad och menyn stängd. Dessutom visas sliders för rotation och skalning av fordonet.	16
A.1 GANTT-schemat till projektplanen. De orangemarkerade veckorna är när projektgruppen har uppehåll. Blåmarkerad ruta innebär arbetsvecka.	26

A.2 GANTT-schema som visar hur det faktiskt blev. De rosamarkerade veckorna är de som skiljer sig åt från originalplanen.

27

Kapitel 1

Introduktion

I detta kapitel presenteras rapportens syfte, en kort bakgrund kring arbetet samt frågeställningarna som försöker besvaras.

1.1 Bakgrund

Varje år dör cirka 40 personer i bilolyckor på svenska vägar [13]. När en fältarbetare inom Räddningstjänsten Östra Götaland (RTÖG) anländer till en olycksplats, där en eller flera fordon kolliderat, är det ytterst viktigt för dem att veta att det på ett säkert sätt går att ta sig in i fordonen och eventuellt rädda livet på de olycksdrabbade. I nuläget använder sig RTÖG av 2D-kartor som hjälper dem att identifiera placeringen av kritiska objekt inom fordonen såsom bränsletankar, elcablar, batterier, stolpar, med flera. Om ett fordon har krockat finns en risk att en dörr eller annan del behöver kapas för att komma åt den olycksdrabbade, vilket kan leda till vissa risker då vissa objekt utgör en fara ifall de blir kapade eller utsatta för till exempel lågor. Räddningspersonal måste mentalt övergå från 2D-kartan till verklighet, vilket kan ta tid och leda till skadliga misstag.

I detta projekt var kunden intern och därmed ej kopplad till RTÖG. Kunden hade ej några krav utöver att lösningen skulle vara AR-driven och visa positionen av kritiska objekt inom ett fordon.

1.2 Syfte

Syftet med denna rapport är att undersöka om en *Augmented reality*-applikation (AR) kan underlätta för en fältarbetare under en räddningsprocess där ett eller flera fordon kolliderat. Genom att utnyttja AR kan fältarbetaren se en 3D-modell av fordonet som förhoppningsvis kan underlätta att identifiera objekt i fordonet, kanske till och med snabbare än om en 2D-karta används. Objekten som behöver identifieras är oftast viktiga, så som gastankar, sladdar och krockkuddar, och måste hanteras varsamt utifall fordonet kolliderat.

Som redan nämnt används redan 2D-kartor, men genom att undersöka om AR kan vara en lämplig förbättring av räddningsprocessen för räddningstjänstens arbete, kan potentiellt fler liv räddas.

1.3 Frågeställning

1. Hur kan man försäkra sig om en snabb responstid av systemet, med storleken på datafilerna i åtanke?
2. Kan man, med hjälp av AR, underlätta för en fältarbetare inom Räddningstjänsten Östra Göta-land (RTÖG) att identifiera kritiska delar hos ett fordon?

1.4 Avgränsningar

Syftet är att underlätta för fältarbetare inom räddningstjänsten, därför har fokus legat på just de användarna gällande utvecklingen av applikationen. Plattformen som applikationen utvecklades till är Android, eftersom det är den plattform som utvecklarna har haft tillgång till. Istället för att kunna ladda in olika typer av 3D-modeller prioriterades att få in en 3D-modell av ett fordon i AR-scenen. Detta beslut togs för att visa på konceptet.

Under projektets gång fanns det hårda restriktioner i samhället kring hur många personer man fick träffa och hur. Detta gjorde det väldigt svårt att organisera användartester där personernas uppgift var att testa applikationen på plats. Valet gjordes då att bredda användartesterna till en större mängd män-niskor än tänkt. Detta gav i slutändan fler resultat, men inte från den demografin som applikationen är lämpad för.

Kapitel 2

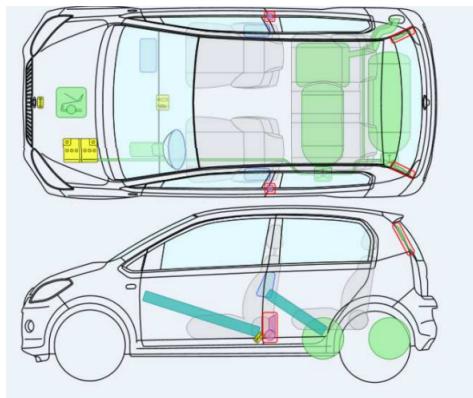
Relaterat arbete

AR bygger på idén att användaren kan interagera med virtuella objekt i den verkliga miljön. AR behöver inte i allmänhet arbeta med det visuella sinnet utan det kan även utnyttjas med haptik eller andra sinnen [23]. AR kan definieras med tre punkter [11]:

- Kombinerar verkliga med virtuella objekt.
- Interaktiv i realtid.
- Registrerad i 3-D.

Det finns flera olika system för att visa de interna delarna av ett fordon med hjälp av AR, till exempel har Mercedes-Benz ett eget system som använder AR [5]. Det finns ingen forskning i nuläget som visar att tillämpning av AR är fördelaktigt inom räddningstjänsten. Ett av de mest utforskade områdena inom AR idag är inom industrin för att visualisera manualer till tekniska enheter. Studier visar att det kan vara fördelaktigt att visa manualer via AR istället för traditionella metoder [18]. Brandkårer kan använda AR för att förenkla placering av olika objekt i brandbilen [12]. En annan tillämpning av AR är inom sjukvården men framför allt inom kirurgi där det kan hjälpa läkare visualisera interaktiv data i tre dimensioner [15].

Enligt Myndigheten för samhällsskydd och beredskap (MSB) i publikationen *Räddning vid trafikolycka — personbil* [25] så har räddningstjänst tillgång till *Moditechs Crash Recovery System* (CRS). Med hjälp av detta system kan räddningspersonal få fram tydliga 2D-bilder på ett fordons inre utifrån vilken modell bilen har, se figur 2.1. Bilderna visar på kritiska objekt i fordonet såsom elkablar, gastank, batteri m.m. Detta system är användbart för räddningstjänsten eftersom olika modeller av bilar har olika konstruktioner och placerar därför dessa olika objekt på olika ställen. För att motverka risken att någonting som klipps/sågas isär och leder till personskada så behöver alltså räddningspersonal veta var dessa objekt är placerade.



Figur 2.1: Bild från Moditech Rescue Solutions som visar hur en 2D-karta för ett fordon ser ut.

När räddningspersonal använder sig av 2D-kartor krävs det att de utför mental rotation på de objekt de vill identifiera. Resultatet av en studie utförd av Shepard och Metzler år 1971 vid namnet *Mental Rotation of Three-Dimensional Objects* [22] visade på att mental rotation av tredimensionella objekt tog upp mer tid ju mer den tvådimensionella representationen var roterad i förhållande till det verkliga objekten. Det framkom även att ett objekts komplexitet påverkade hur lång tid det tog att identifiera.

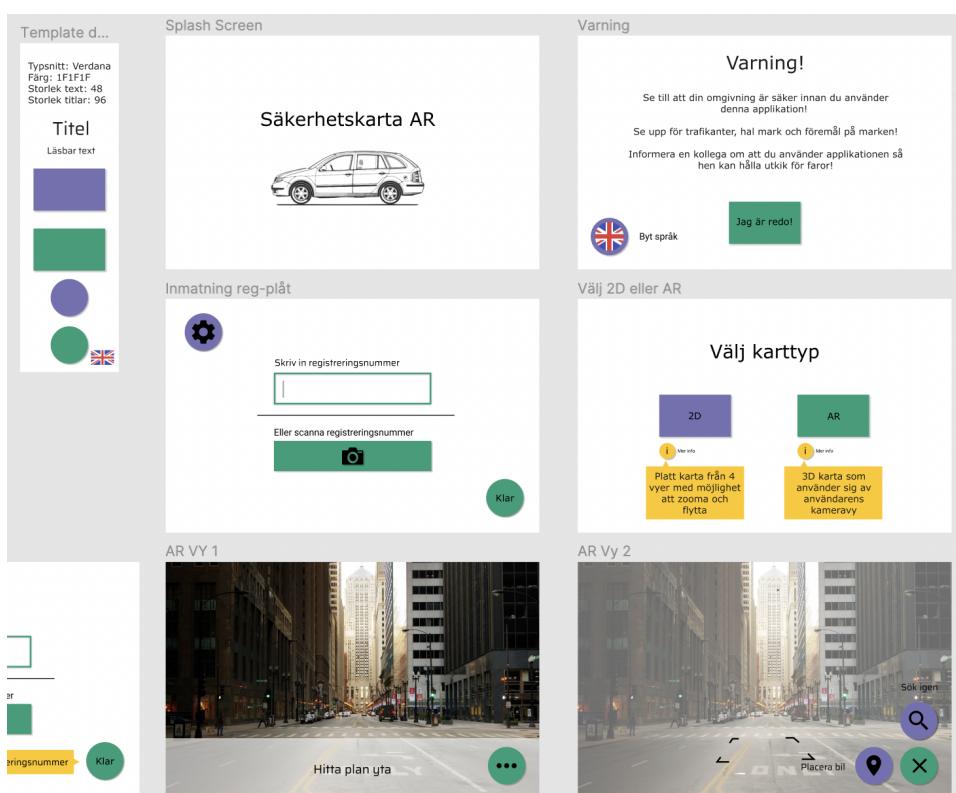
Kapitel 3

AR-applikation i Unity med hjälp av Vuforia

Detta kapitel beskriver metoden för hur AR-applikationen skapades i Unity med hjälp av Vuforia samt en beskrivning av förarbetet.

3.1 Design och prototyp

Designen av applikationen och första prototypen gjordes i *Figma* [16]. En del av designprototypen kan ses i figur 3.1. Genom att först designa applikationen fullt ut gick det sedan snabbt att implementera det i Unity. För att få fram designen diskuterade projektgruppen först tillsammans om färgval, hurdan stil applikationen skulle ha och hurdana visioner som gruppmedlemmarna hade. Sedan utvecklades olika designer och gruppen diskuterade tillsammans vad som var väsentligt för applikationen. Under denna diskussion togs även knappstorlek, form och placering fram.



Figur 3.1: En del utav applikationsdesignen från Figma som visar hur de tänkta sidorna i applikationen ska se ut.

Prototypen byggdes upp med hjälp av *frames*, häданefter kallade *ramar*, med en ram per skärm. De skärmar som togs fram var en *splash screen*, en varningssida, en för inmatningen av registreringsplåt, en för val av 2D- alternativt AR-karta samt flertalet ramar för att representera de olika stegen i AR-vyn. Färgvalet för knappar togs fram med hänsyn till färgblinda med hjälp av hemsidan *Colorbrewer* [14] och typsnitt valdes utifrån läsbarhet när prototypen visades på en surfplatta.

3.2 Användartester

Det första användartestet grundades i utvärdering av layout och appdesign. Detta användartest gjordes innan implementering hade påbörjats. I användartestet låg inte fokus på att en särskild målgrupp skulle utvärdera prototypen. Målet var att få så många åsikter som möjligt för att justera till en mer användarvänlig design. En nackdel med en bred testgrupp var att en del av testpersonerna inte förstod varför de som privatpersoner skulle behöva använda en sådan applikation. Användartestet gick till på så sätt att testpersonerna kunde klicka genom prototypen eftersom Figma, ett verktyg för mockups och prototyper, har möjligheten att koppla samman knappar och sidor så att det känns som om användaren har en riktig applikation framför sig. Testpersonerna fick instruktioner om att först klicka runt i prototypen och sedan besvara frågor som fanns i ett *Google Forms*-formulär. De flesta justeringar som åtgärdades efter detta användartest var kopplade till knapparnas placering, utseende och funktionalitet.

Det andra användartestet gjordes efter all implementering gjorts och projektgruppen hade en färdig applikation att testa. Detta användartest gick ut på att testpersonerna fick en färdig applikation på en surfplatta. Under dessa användartest spelades ljudet in för att projektgruppen inte skulle behöva lägga ner tid på att anteckna under själva användartestet, utan att fokus kunde ligga på att intervjuva testpersonerna och ställa följdfrågor till deras feedback och reaktioner. Först fick testpersonerna bekanta sig med applikationen för att lära sig hur systemet fungerar, sedan fick de uppgifter att genomföra och frågor att besvara. Målet med detta användartest var att se om AR-kartan faktiskt hjälpte användaren att identifiera ett objekt i bilen. Frågorna vi ställde syns nedan.

- **Fråga 1**

Vilken metod (AR eller 2D-karta) var lättast att använda?

- **Fråga 2**

Vilken metod känner du hade varit mest optimal i en kritisk situation där det finns ont om tid?

- **Fråga 3**

Är applikationens struktur överlag logisk och lätt att följa? Var det något som var oklart?

- **Fråga 4**

Till denna fråga har testpersonen fått ta på sig handskar.

Frågan lyder: Hur känns applikationens användargränssnitt när vantarna är på? Känns knapparna lätta att nå / är de tillräckligt stora för att applikationen lätt ska kunna användas?

3.3 Plattform och biblioteksväl

Till en början skulle ARCore[2] användas tillsammans med Unity, men eftersom det finns mer dokumentation om mjukvaruvecklingspaketet Vuforia, och Vuforia har enkla metoder för att placera virtuella objekt i AR utan att förankra objektet till en fysisk bild, bestämde projektgruppen sig för att använda sig utav Vuforia. All UI-implementering gjordes med hjälp av Unity-UI och *scripts* som

skrevs i Visual Studio, som är en texteditor och kompilator för bland annat programspråket C# [10]. Projektgruppen använde sig av Unity 2021.1.1f1 eftersom det var den nyaste versionen som fanns tillgänglig vid implementationsstadiet. Alla script är skrivna med C#.

Målplattformen för systemet är surfplattor med operativsystemet Android. Detta beslut är baserat på tillgängligheten till enheter med Android som projektgruppen har. Tillgänglighet till målplattformen är ett krav för att kunna göra systemtester och prova applikationens funktionalitet.

3.3.1 Lagring av modeller

För att förvara registreringsplatssnummer och dess motsvarande 3D-modell av bilmärket behövdes en databas. Denna databas kan sedan fyllas med registreringsplatssdata från Trafikverket samt 3D-modeller från biltillverkarna. Genom att skapa klass-typer för ett fordon med *Unity Scripts* kunde en modell länkas till ett registreringsplatssnummer. Detta visas i figur 3.2. Nedan syns C# pseudokod för en fordons-klass, en klass för en databas i figur 3.4 och en klass för en hjälpfil med funktioner för att hämta modeller ur databasen i figur 3.3.

```
public class Model : ScriptableObject {
    public string modelId;
    public string registrationNumber;
    public string modelName;
    public GameObject model;
}
```

Figur 3.2: Pseudokod för en fordons-klass i C# som gör det möjligt att länka modell och registeringsplattnummer.

Med hjälp av denna klass-typ kan en databas skapas.

```
public class DatabaseHelper {  
    public ModelDatabase models;  
    public static DatabaseHelper instance;  
  
    public void Awake() {  
        if(instance == null) {  
            dontDestroyGameObjectOnLoad();  
        } else {  
            destroyGameObject();  
        }  
    }  
  
    public static Model GetModelById(String id) {  
        return instance.models.allModels.Find(id);  
    }  
    public static Model GetModelByRegNum(String num) {  
        return instance.models.allModels.Find(num);  
    }  
}
```

Figur 3.3: Pseudokod för en databas-klass med hjälpfunktioner i C#. Den här funktionen möjliggör hämtning av en modell-klass från databasen via antingen ID eller registreringsnummer.

```
public class ModelDatabase: ScriptableObject {  
    public List<Model> allModels;  
}
```

Figur 3.4: Pseudokod för definition av en databas-klass i C#. Här definieras en klass med en lista som förvarar alla modell-klass-objekt.

Under utvecklingsprocessen av applikationen uppstod problem med Unity och Vuforia som begränsade dynamisk laddning av modeller under *Run-time*, alltså medan appen körs. Kunskapen saknades för att lösa detta problem och efter diskussion med kund bestämde projektgruppen att databasen inte ska användas. Detta diskuteras mer i kapitel 5. Istället laddas endast en modell in i applikationen, oavsett vilken registreringsplåt som anges. Det gjordes genom att förbestämma en modell som redan ligger importerad, men gömd, tills AR-delen av applikationen aktiveras.

Genom att utnyttja en lokal databas, till skillnad från en extern sådan med hjälp av till exempel ett API, kräver inte applikationen en internetuppkoppling och minimerar eventuella bristpunkter. Tiden det tar att ladda in en modell förkortas då också.

3.4 Appdesign och funktionalitet i Unity med UI-toolkit

Som ovan nämnt skapades en mockup och prototyp innan implementeringen började. Därefter inledes arbetet med att sätta ihop applikationen i Unity. Storleken på skärmen var förbestämd i Figma och kunde enkelt plockas därifrån. Samma gällde knapparnas storlek, textens typsnitt och dess korrekta storlek.

Användargränssnittet är uppbyggt med hjälp av Unitys inbyggda UI-system. Varje sida i applikationen är en självständig scen med ett så kallat kanvas och tillhörande element. Unity UI beskrivs enligt dokumentationen som ett *GameObject-based UI system* som använder olika komponenter och en vy för att justera användargränssnittet [9].

Scripts som implementerades för UI-objekten gav dem deras funktionalitet. Alltså kopplades minst ett script till varje knapp i applikationen då alla knappar skulle ha någon form av funktionalitet. De flesta scripts som skapades var relaterade till databasen och AR-vyn. Scripts för databasen togs upp under föregående kapitel 3.3.1 och för AR-vyn skapades det tre olika. Det första skapades för scenens meny och gjorde det möjligt att klicka på menyknappen för att visa dess innehåll. Det gjordes genom att aktivera eller avaktivera en panel med menyinnehållet beroende på dess dåvarande tillstånd. I samma script skapades även en funktion som byter knappens ikon beroende på vilken ikon som är aktiv. I figur 3.5 visas koden för detta script och i den ser man att vi använder oss av två *gameObjects* och två *Sprites* som då kopplas till respektive objekt och bild i Unity.

```

6  public class buttonInteractions : MonoBehaviour
7  {
8      public GameObject button; //Menyknappen
9      public GameObject panel; //Panelen som ska aktiveras/avaktiveras
10     public Sprite Icon_1; //Ikon för menyn
11     public Sprite Icon_2; //Ikon för att stänga menyn
12
13
14     public void OpenPanel() //Funktion för att öppna/stänga menyn
15     {
16         if (panel != null)
17         {
18             panel.SetActive(!panel.activeSelf);
19         }
20     }
21     public void changeIcon() //Funktion som byter knappens ikon
22     {
23         if (button.GetComponent<Image>().sprite == Icon_1)
24             button.GetComponent<Image>().sprite = Icon_2;
25         else
26         {
27             button.GetComponent<Image>().sprite = Icon_1;
28         }
29     }
30 }
```

Figur 3.5: Kod för applikationens meny-script som gör det möjligt att öppna och stänga menyn samt byta ikon beroende på dess tillstånd.

De två andra scripts som skapades var relaterade till själva 3D-modellen där ett script placerade ut den och det andra gjorde det möjligt att manipulera den. Det script som nämndes först gjorde det alltså möjligt att genom ett knapptryck placera ut modellen på ett plan som AR-kameran hittar. Mer om hur

AR-kameran implementeras och fungerar förklaras under kapitel 3.5. I figur 3.6 visas koden för detta script och även här används ett gameObject som då kopplas till rätt knapp i Unity.

```

8  public class PlaceModel : MonoBehaviour
9  {
10     public PlaneFinderBehaviour plane; //Planet
11
12
13     void Start() //Funktion som kallas när vi startat applikationen och
14                 //kollar på funktionen place
15     {
16         Button button = gameObject.GetComponent<Button>();
17         button.onClick.AddListener(place);
18     }
19
20     public void place() //Placerar ur objektet
21     {
22         Vector2 aPosition = new Vector2(0, 0);
23         plane.PerformHitTest(aPosition); //Kollar om AR-kameran har hittat
24                                         //ett plan
25     }
26 }
```

Figur 3.6: Kod för applikationens placering-script som gör det möjligt att placera ut en modell på ett plan.

Koden för det script som manipulerar modellen visas i figur 3.7 och som man kan se består den i huvudsak av två funktioner, en för skalning respektive rotation. Även här kopplas gameObjects till sina respektive objekt i Unity och genom detta möjliggörs att man kan manipulera modellens egenskaper genom att dra i respektive *slider*. Respektive sliders minimum- och maximumvärden definieras även i Unity.

```

6  public class ScaleAndRotate : MonoBehaviour
7  {
8      private Slider scaleSlider; //Slider för skalning
9      private Slider rotateSlider; //Slider för rotation
10
11     public float scaleMinValue; //Min värde för skalning
12     public float scaleMaxValue; //Max värde för skalning
13
14     public float rotMinValue; //Min värde för rotation
15     public float rotMaxValue; //Max värde för rotation
16
17     //Funktion som körs vid varje frame
18     void Update()
19     {
20         scaleSlider = GameObject.Find("ScaleSlider").GetComponent<Slider>(); //Hämtar objektet
21         scaleSlider.minValue = scaleMinValue;
22         scaleSlider.maxValue = scaleMaxValue;
23         scaleSlider.onValueChanged.AddListener(ScaleSliderUpdate); //Kollar på funktionen som
24                                         //skalar om modellen
25
26         rotateSlider = GameObject.Find("RotateSlider").GetComponent<Slider>(); //Hämtar objektet
27         rotateSlider.minValue = rotMinValue;
28         rotateSlider.maxValue = rotMaxValue;
29         rotateSlider.onValueChanged.AddListener(RotateSliderUpdate); //Kollar på funktionen som
29                                         //roterar modellen
30     }
31
32
33     void ScaleSliderUpdate(float value) //Skalar modellen
34     {
35         transform.localScale = new Vector3( (value), (value), (value) );
36     }
37     void RotateSliderUpdate(float value) //Roterar modellen OBS! endsat i y-axeln
38     {
39         transform.eulerAngles = new Vector3( (transform.rotation.x), (value), (transform.rotation.z) );
40     }
41 }
```

Figur 3.7: Kod för applikationens skalning- och rotations-script som gör det möjligt att skala och rotera en modell.

För resterande scener och deras knappar i applikationen skapades ett script för dessa som byter scen när man klickar på respektive knapp.

3.5 AR-implementation och 3D-modellering

Hur AR-implementationen skulle se ut kom fram efter förstudier. Tanken blev att användaren skulle klicka på en knapp för att lägga ut en 3D-modell på en bil. Modellen lades ut med hjälp av så kallad yt- eller plandetektion. I AR behövs en ankarpunkt för att lägga ut ett objekt, det kan till exempel vara ett fotografi som känns igen av kameran. Men som tidigare nämnts valdes plandetektion som metod, kameran känner alltså igen en plan yta och objekten kan läggas ut på den ytan.

Med en färdig prototyp och generell bild av hela applikationen implementerades AR-delarna. För att skapa en AR-scen i Unity användes Vuforia Engine SDK [24], då den enkelt kopplades med Unity och innehöll plandetektion, eller på engelska *Ground Plane Detection*[19], vilket var nödvändigt för applikationsiden. Vuforia valdes även för att det fungerar för både IOS och Android, detta var bra då målplattformen då det inte går att förutspå vilken enhet en användare äger.

För att kunna lägga ut en 3D-modell i AR med hjälp av Vuforia för Unity så lades först ett AR-kameraobjekt till. Det är det objekten som gjorde det möjligt för applikationen att använda sig av telefonens kamera. Därefter lades två objekt in i scenen som var nödvändiga för Ground Plane Detection, dessa var en plan-upptäckare och ett yt-scen-objekt, respektive *Plane Finder* och *Ground Plane Stage* på engelska. *Plane Finder* var det som hittade den plana ytan, *Ground Plane Stage* kopplades sedan till *Plane Finder*. På så sätt kunde *Ground Plane Stage* läggas ut på den plana ytan. En 3D-modell lades in i *Ground Plane Stage* vilket ledde till att applikationen kunde lägga ut ett 3D-objekt på en plan yta som kameran hittar när en knapp klickades. Till knappen skrevs ett script för att lägga ut ett 3D-objekt. Vuforias standardsätt för att lägga ut en modell i scenen var med ett tryck var som helst på skärmen, vilket betydde att applikationen hade två sätt att placera ut ett objekt. Det gick inte att ta bort den funktionaliteten.

Innan 3D-modelleringen kunde påbörja krävdes en 2D-karta för att ha som grund i modelleringen av de kritiska delarna i en bil. En sådan karta, skapad av Moditech Rescue Solutions, av en Skoda Citigo CNG hittades i en publikation av MSB vid namn *Räddningsinsatser med gasdrivna personbilar* [17].

För att underlätta arbetet valdes att använda en gratis och färdig modell av en bils yttre, i detta fall en Skoda Fabia på grund av dess liknande form till Citigo CNG. Modellen laddades ned och importerades sedan in i Unity och gjordes därefter transparent så att de kritiska delarna som därefter modellerades skulle vara synliga genom bilens yttre.

Genom att utgå från 2D-kartan modellerades de kritiska delarna i bilen såsom bensintank, batteri, A-, B- och C-stolpar, oljetank samt elkabel från batteriet. Dessa delar gavs olika material för att förtydliga vilka som var vad, t.ex. grön för allt bensin-relaterat, gult för batteri och lila för stolpar.

3.6 Systemtester

Systemets delar som testades var AR-relaterade eftersom AR var en stor del av applikationen och AR kan vara krävande för en enhet som kör programmet. För att testa dessa bitar gjordes en undersökning hur snabbt en plan yta upptäcktes efter att AR-scenen startades. Flera test gjordes på detta sätt med olika förutsättningar. Mitt på dagen och på kvällen var de förutsättningarna som testades för att observera hur snabbt den plana ytan hittas då det finns begränsat ljus. Applikationen byggdes även till IOS och kördes på en *iPad Pro* för att se hur hastigheten påverkades när systemet testades på annan hårdvara.

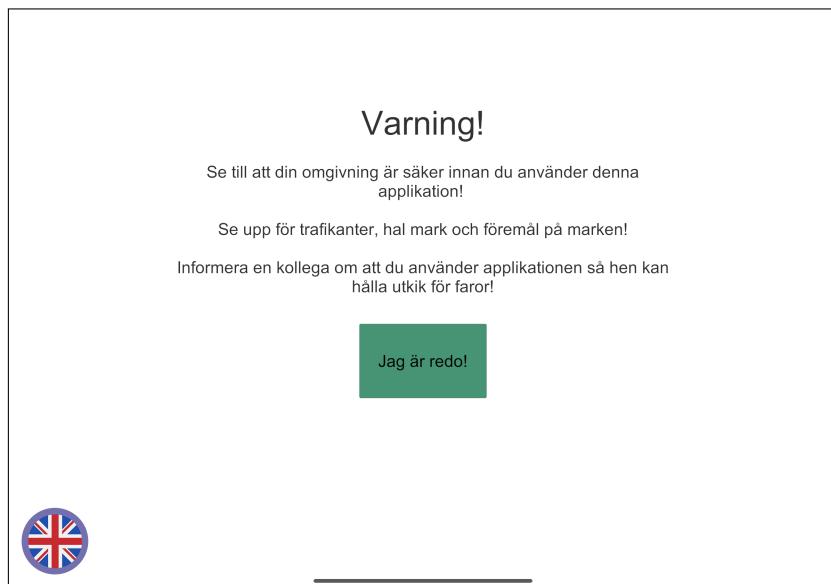
Kapitel 4

Resultat

Under detta kapitel presenteras resultaten som fåtts ut av arbetet.

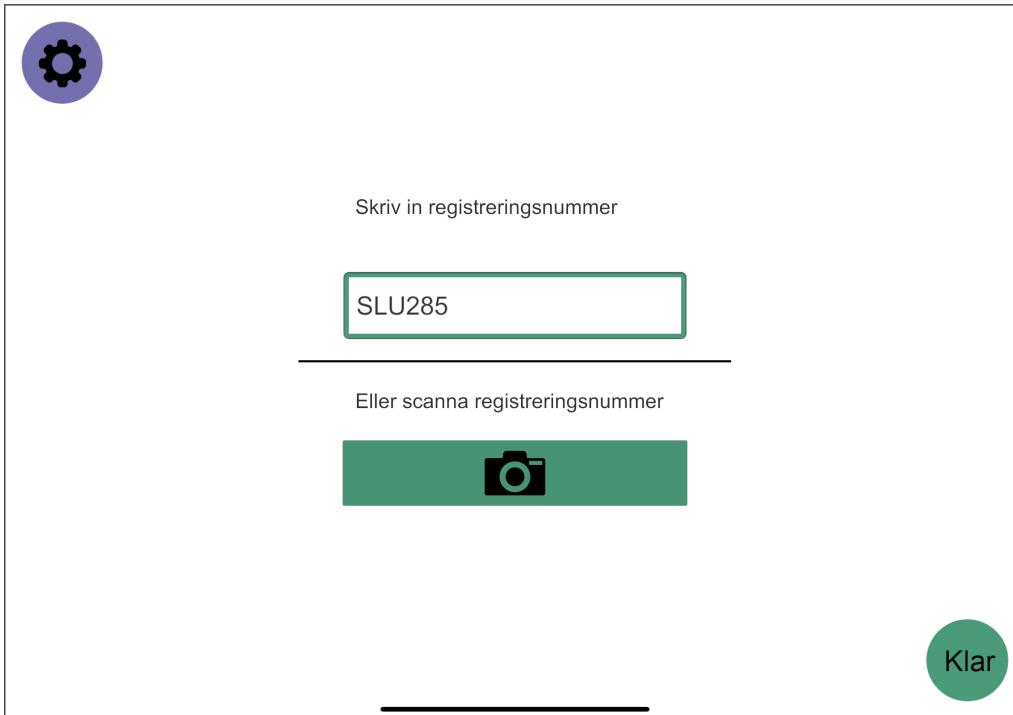
4.1 Applikation

Arbetet resulterade i en applikation som består av ett flertal scener och som kan placera ut en 3D-modell av ett fordon. Applikationens scener presenteras i ordning där första scenen är figur 4.1 som visas när applikationen startas. Som figuren visar är det en varningssida.



Figur 4.1: Applikationens startsida som visas när applikationen startas. Användaren går vidare till nästa sida genom att trycka på den gröna 'Jag är redo'-knappen.

För att gå vidare måste man klicka på '*Jag är redo!*'-knappen och då kommer man till nästa scen som visas i figur 4.2. I denna scen kan man skriva in ett fordons registreringsnummer och för att gå vidare måste man klicka på '*Klar*'. Skanning av registeringsnummer implementeras ej.



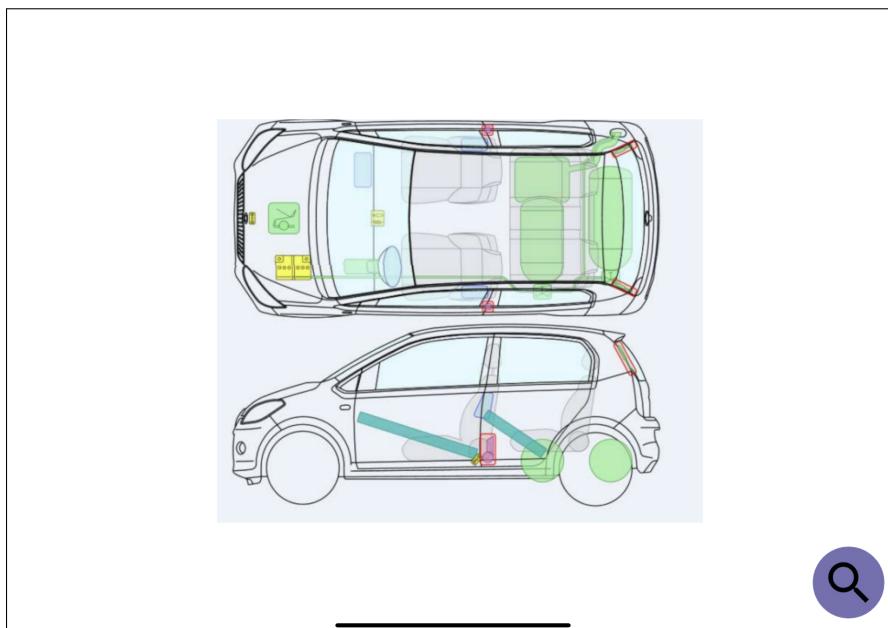
Figur 4.2: Applikationens vy för inmatning av registeringsnummer där man kan skriva in ett registeringsnummer och gå vidare genom att klicka på '*Klar*'.

Efter att användaren har klickat på '*Klar*' kommer hen till nästa sida som visas i figur 4.3. I denna scen kan användaren välja mellan att se en 2D-karta eller en 3D-modell i AR genom att klicka på respektive knapp.

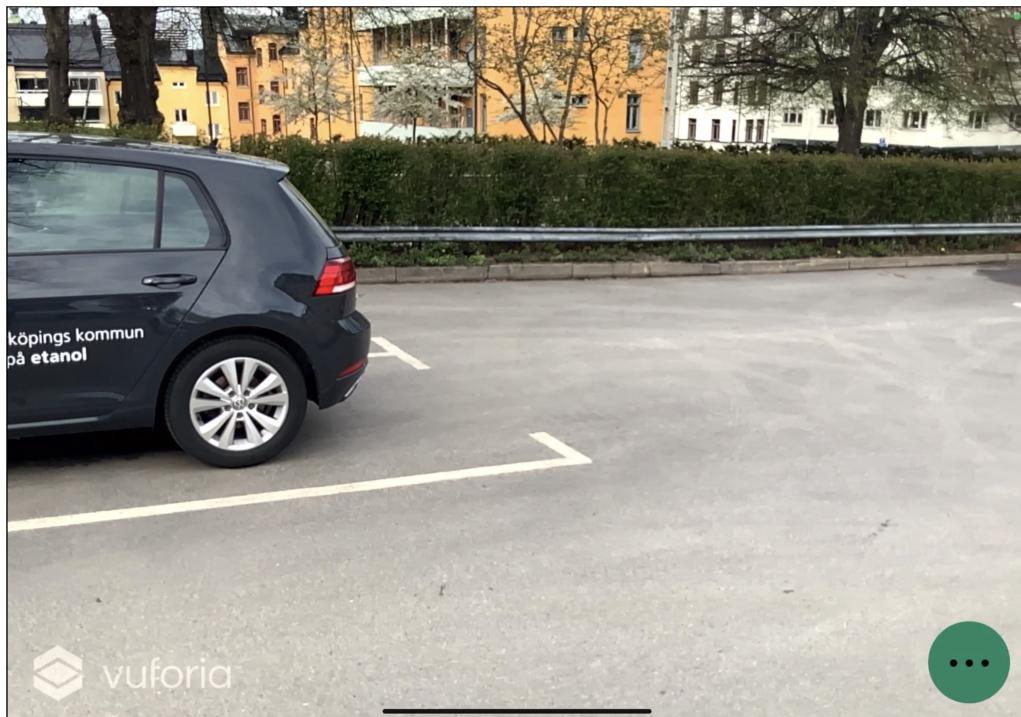


Figur 4.3: Applikationens vy för valet av karttyp där '2D' leder till 2D-kartan och 'AR' leder till AR-vyn.

Om användaren klickar på '2D'-knappen visas scenen i figur 4.4 och om man istället klickar på knappen 'AR' visas scenen i figur 4.5. I 2D-scenen kan man endast klicka på knappen nere till höger vilken skickar en till scenen i figur 4.2.

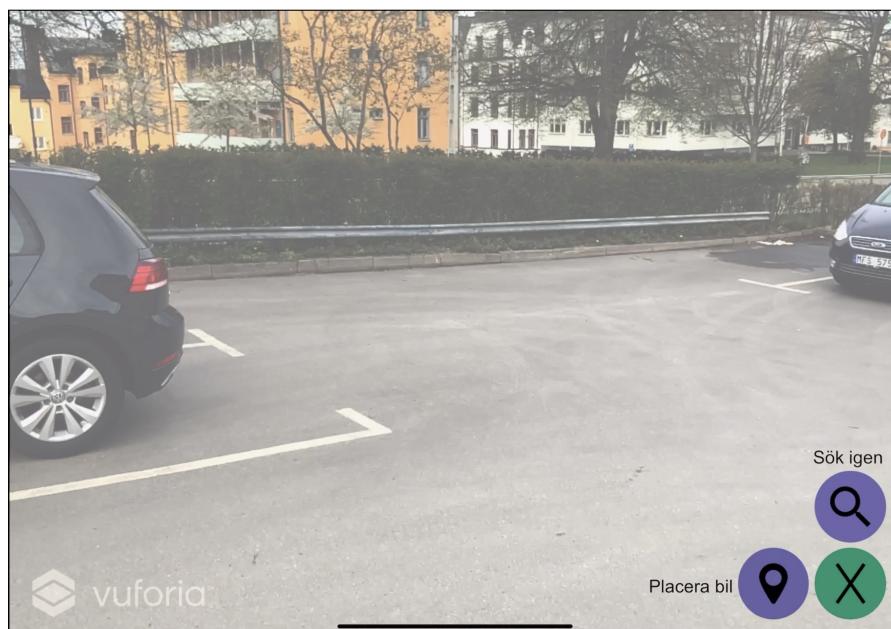


Figur 4.4: Applikationens vy för 2D-kartan där man kan klicka på 'Sök igen'-knappen nere i högra hörnet för att komma till vyn i figur 4.2.



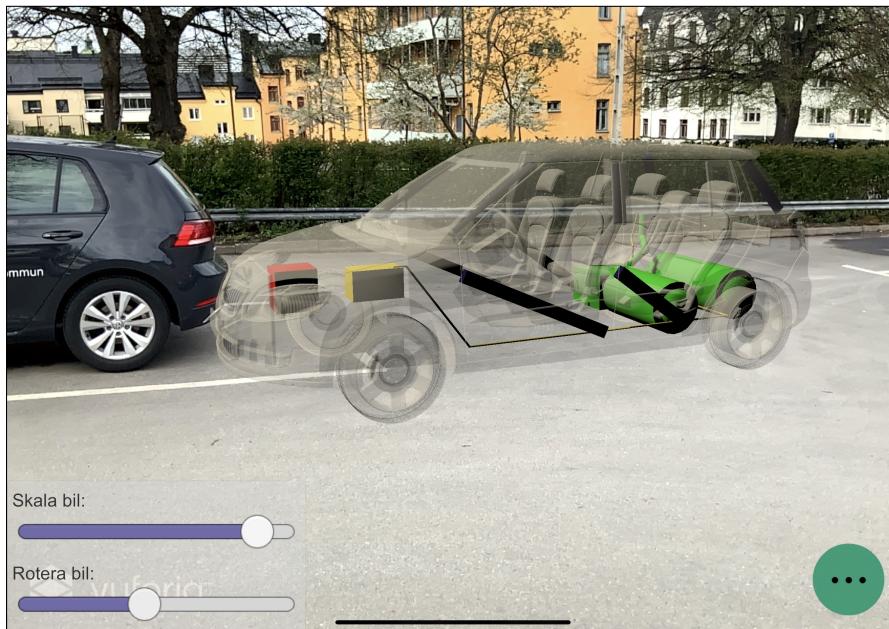
Figur 4.5: Applikationens AR-vy med stängd meny där man kan öppna menyn genom att klicka på knappen nere i hörnet.

I AR-scenen kan man alltså klicka på menyknappen för att aktivera eller avaktivera menyn som visas i figur 4.6. I menyn kan man som ses i figuren klicka på 'Placera bil'-knappen och placera ut en 3D-modell. Man kan även från menyn gå tillbaka till scen 4.2 genom att klicka på 'Sök igen'-knappen.



Figur 4.6: Applikationens AR-vy med menyn öppen där man kan placera ut fordonet, söka igen och stänga menyn med respektive knappar.

Efter att man har klickat på 'Placera bil'-knappen ser skärmen ut som i figur 4.7 med menyn stängd. I figuren kan man även se båda sliders för 3D-modellens skalnings- och rotationsfunktionalitet.



Figur 4.7: Applikationens AR-vy med bilen utplacerad och menyn stängd.
Dessutom visas sliders för rotation och skalning av fordonet.

4.2 Användartester

Från svaren i det första användartestet, som gjordes för att utvärdera designen, justerade projektgruppen layout vilket ledde till ökad användarvänlighet. Justeringarna gjorde att syftet för AR-vyns knappar var tydligare för användaren. Text på språk- och inställningsknapparna ändrades till symboler, vilket testpersonerna upplevde var mer intuitivt.

I användartest två låg fokus på AR-delen av applikationen. Totalt gjordes användartester med fem olika personer varav fyra stycken var civila och en var polis. Tyvärr var möjligheten begränsad att göra fler användartester eller fokusera på en specifik demografi, vilket diskuteras i kapitel 1.4.

Utifrån svaren från de två första testpersonernas kommentarer gjordes småjusteringar så att de efterkommande testpersonerna kunde ge feedback på andra aspekter än de två föregående. Justeringar som gjordes var bland annat skalning av en scen som hade annan storlek än övriga scener, skalning av bilen och omkoppling av knappar som antingen inte hade fungerat eller som ledde till fel scen.

Utöver justeringarna som nämns ovan upptäckte testpersonerna även några delar som ej fungerade i applikationen. Bland annat spelade det ingen roll vilket registerplåtsnummer som skrevs in i sökruutan, och scan-funktionen var inte implementerad. Dessa funktionaliteter var projektgruppen medveten om, men det var av intresse att testpersonerna faktiskt valde att trycka på scan-funktionens knapp före de valde att försöka skriva in ett registreringsnummer.

Utifrån alla användartester ser vi att AR-applikationen underlättar identifiering av objekt i ett fordon samt minskar tiden till identifiering. Vi ser också att AR-modellen är lättare att tyda då den extra dimensionen ger mer utrymme, och kartan är inte lika trång som 2D-kartan. Tyvärr krävs en inlär-

ningsprocess för att göra användningen smidig, då alla testpersoner kommenterade på att AR-biten av applikationen var lite svår att använda och tog lite tid.

4.3 Systemtester

Systemets huvuddel är AR, därför har olika scenarion prövats för att se begränsningar när det gäller AR-implementeringen. De scenarion som testades var mitt på dagen och natten. Applikationen hittade plana ytor mellan 0.5-2 sekunder med dagsljus. Under natten med begränsat ljus visade det sig vara svårare, systemet behöver någon typ av ljuskälla för att känna igen plana ytor. Alltså behöver systemet någon typ av ljuskälla för att fungera. Systemet testades även på olika plattformar och hårdvara. Applikationen byggdes även till IOS, då testades appen på en iPad Pro-surfplatta. I dagsljus tog det 0.5 sekunder för den att hitta en plan yta. Med en Androidtelefon av modellen Oneplus 6 tog det runt 2 sekunder på flera test. 3D-modellen som användes i applikationen blev cirka 30 megabyte i filstorlek och att placera ut ett objekt tog i princip ingen tid alls efter att en markplan funnits av Vuforia.

Kapitel 5

Analys och diskussion

Detta kapitel presenterar diskussion kring metod och resultat av arbetet. Här diskuteras vad som har gått bra eller dåligt i samtliga delar av arbetet.

5.1 Metod

Denna del fokuserar på diskussion kring metoden av arbetet.

5.1.1 Användartester

Optimalt hade användartester gjorts på räddningstjänstens personal samt andra personer inom liknande yrke och åldersgrupp. Detta hade gett den bästa inblicken i hur applikationen bör utvecklas för att bäst passa just dem.

Att testgruppen för det första användartestet var bred var både till för- och nackdel. Fördelen var att åsikter från personer med olika bakgrund och livssituation kunde samlas in, vilket var till nytta för projektgruppen eftersom det testet var till för att utvärdera designen och göra den så användarvänlig som möjligt. Eftersom projektgruppens kontaktnät till största del består av studenter kom dock återkopplingen till största del från studenter i samma ålder som projektgruppens medlemmar. En nackdel med den öppna testgruppen och projektgruppens begränsade kontaktnät var att projektgruppen inte fick återkoppling från den tänkta målgruppen, och det är i huvudsak deras åsikter som egentligen var sökta. Det begränsade kontaktnätet påverkade även tillgängligheten till testpersoner för det andra användartestet. Eftersom användartest två gjordes fysiskt begränsades även tillgängligheten till testpersoner ytterligare på grund av en pågående pandemi. Till följd av pandemins smittorisk och restriktionerna som nämns i kapitel 1.4 var kontakt med utomstående något som skulle undvikas i största möjliga mån.

5.1.2 Användargränssnitt och funktionalitet

Beslutet att använda Figma som ett designverktyg var fördelaktigt eftersom Unity har ett väldigt likt system för UI. Det gjorde att utvecklarna snabbt kunde implementera funktionerna. Det visade sig vara en stor fördel att utveckla hela projektet i Unity eftersom det finns ett gott stöd för Vuforia. Ett av problemen som uppstod kring Vuforia var att deras kod är låst för utvecklaren vilket gjorde det svårare att anpassa funktionaliteten utifrån våra behov. Dock sparade vi tid på att inte behöva programmera all AR-funktionalitet på en lägre nivå. En annan aspekt i arbetet är hur vi hanterade bilmodellerna.

I projektet hade vi tillgång till en 2D-karta och en 3D-modell för samma fordon. 3D-modellen blev tilldelad material för att skapa en transparent effekt. Med en större mängd fordon i databasen skulle en ökad mängd manuellt arbete krävas om varje bilmodell skulle genomgå samma process.

5.1.3 Systemtester

Metoden för att göra systemtester av en AR-applikation är inte så tydlig. Det är svårt att mäta prestanda när man använder externa bibliotek som Vuforia. Det vi gjorde var att testa hur lång tid saker tog genom Unitys egna verktyg samt extern klocka, men det gick inte att i koden placera mätverktyg av något slag.

Om applikationen i framtiden ska innehålla till exempel hundra tusen modeller, och tiden det tar att ladda in en modell är direkt beroende av mängden data. Då är tiden det tar att ladda in en modell otroligt viktig, och en differens på bara millisekunder kan orsaka större problem i en större applikation. Det var någonting som var väldigt svårt för oss att testa, och andra verktyg kanske måste erhållas för att göra bättre tester och en bättre bedömning.

5.1.4 Källkritik

I rapporten hänvisar vi främst till tre typer av källor. Teknisk dokumentation, akademiska texter och information från ett antal svenska myndigheter. Vi refererar även till hemsidor som hjälpt oss under utvecklingsprocessen som Figma eller Colorbrewer. Vi har även en källa från ett bilvarumärkes marknadsföring men vi använder den endast i syfte om att visa att det finns riktiga produkter på marknaden som innehåller AR i det här ändamålet. De akademiska texterna är hämtade från diverse tekniska tidskrifter, Konferenser och universitet.

5.2 Resultat

Här diskuteras och analyseras de resultat som togs fram. Det vill säga applikationen, användartester, systemtester och databas.

5.2.1 Hantering av 3D-modeller och databas

Som konstaterat i kapitel 3.3.1 gick det inte att dynamiskt ladda in modeller efter att applikationen var kompilerad. Det var bristande kunskap kring Vuforia och Unity som gjorde att vi inte kunde lösa detta problem och det gick tyvärr inte att inse svårigheten förens databasen var utvecklad. Efter ändring av implementationen begränsades applikationen till endast en bilmodell, vilket inte gjorde någonting för slutresultatet i detta skede, men för att applikationen faktiskt ska kunna användas är detta såklart en funktionalitet som måste implementeras.

5.2.2 Användartester

Som nämnt i kapitel 2 visade en studie [22] på att mental rotation av en tvådimensionell representation av ett tredimensionellt objekt tog mer tid ju mer komplext objektet var. Utifrån detta resultat är det möjligt att spekulera att ett objekt så komplext som en bil kan ta lång tid att mentalt rotera. Då en AR-version skulle hoppa över steget av mental rotation är det möjligt att tiden det tar att identifiera

objekt förminskas.

Som vi såg i kapitel 4.2 var tiden till identifiering kortare när användaren använde sig utav AR-kartan jämfört det 2D-kartan, vilket kan tyda på att det krävs extra arbete för en person att mentalt konvertera 2D-kartan till ett 3D-objekt. Under användartesten kontrollerades inte den exakta tidsskillnaden, utan resultaten baseras i stor grad på testpersonernas upplevelse av kartorna. För en större forskning skulle eventuellt tidtagning av samtliga test vara aktuella.

5.2.3 Användsgränssnitt och funktionalitet

Den resulterade applikation blev i huvudsak som tänkt dock saknas viss funktionalitet som inskanning av registreringsnummer eller hämtning av modeller från en databas. Dessutom saknas några sidor så som en sida för inställningar och en inladdningsskärm som skulle visas när man startar applikationen. Dessa funktioner och sidor valdes att prioriteras bort då de inte ansågs vara väsentliga för det slutliga målet med applikationen som då var att låta en användare placera ut en 3D-modell för att kunna identifiera farliga objekt inom den. Trots att dessa delar inte togs med var den slutliga applikationen tillräcklig för att besvara rapportens ena frågeställning men hade inte varit tillräcklig om den faktiskt skulle användas av personal från räddningstjänsten.

Användargränssnittet för applikationen blev även den ungefärligt som tänkt då de förändringar som gjorts jämfört med Figma prototypen var kopplade till hur man skalar och roterar 3D-modellen. I Figma så saknas ett gränssnitt för dessa då det inte var helt genorträknat hur detta skulle gå till eller se ut. Detta påverkade dock inte rapportens slutsatser under kapitel 6 då funktionaliteten fortfarande implementerades i applikationen.

5.2.4 Systemtester

Resultaten för systemtesterna visar att hårdvaran påverkar hur snabbt en plan yta hittas av applikationen. Detta kan betyda att Räddningstjänsten behöver en enhet som är tillräckligt snabb för att klara av applikationen. Detta gör att applikationen inte är lika tillgänglig, om en personal inte kan använda sin egen telefon blir Räddningstjänsten tvungna att tillhandahålla sin personal en sådan enhet.

Systemtesterna gav också information om hur stor påverkan ljuset har för att applikationen ska hitta en plan yta. Detta talar om att tekniken inte är så användbar inom detta område. Stor andel av trafikolyckor sker på grund av trötthet [26], detta kan betyda många olyckor sker under natten. Då kommer inte applikationen vara till någon hjälp, eftersom då har systemet svårare att hitta en plan yta. För att lösa detta problem skulle man behöva ha någon typ av kamera som ser bättre i mörkret. Då blir frågan om det värt att försöka lösa det problemet om systemet som räddningstjänsten redan använder fungerar, vilket beskrivs i kapitel 2.

5.3 Etisk och samhällelig reflektion

För att undvika orättvisor och utmattning inom projektgruppen har arbetet fördelats jämnt mellan gruppmedlemmarna. Genom avstämningssmöten har arbetsbördan för respektive gruppmedlem kontrollerats och det kommande arbetet har fördelats jämnt så att alla har lika mycket ansvar. Risken med att någon gruppmedlem överarbetar sig har inte varit stor, och med god kommunikation inom gruppen har arbetsbördan övervakats.

Applikationen är i första hand gjord för RTÖG att använda, och därmed ska information som är relevant för målgruppen finnas i systemet. Det kan eventuellt ses som känslig information, och då bör möjligheten för miss bruk av informationen noga ses över. För att förhindra att personer som vill missbruka applikationen för att skada andra skulle en sorts licens för användning av applikationen kunna göras. Denna licens skulle endast ges till räddningstjänsten som i sin tur har ansvar över att applikationen endast finns på mobila enheter som används i arbetssyften.

Det är även av stor vikt att informationen som finns i applikationen är korrekt, eftersom en felaktig karta kan skapa stora konsekvenser och försvåra Räddningstjänstens arbete enormt. Det finns kablar som kan vara ödesdigra om de skulle kapas, och eftersom syftet med applikationen är att hjälpa Räddningstjänsten från att undvika just sådana misstag, skulle det vara en stor brist i förtroendet för applikationen och dess skapare om applikationen innehöll felaktig information. För att systemet ska ha tillgång till relevant information så kräver det även att samtliga bilföretag kontinuerligt delar med sig av sina ritningar så att databasen alltid kan vara uppdaterad.

En viktig aspekt som projektgruppen behöver ta i beaktande är att säkerställa att användarna är uppmärksamma till sin omgivning. Det förutsätts att det aldrig kommer vara föraren i RTÖGs bil som hanterar surfplattan med applikationen, men också att användaren meddelar sina kolleger att dess uppmärksamhet kommer vara riktad mot surfplattan och därför inte har fullt fokus på omgivningen. För att försäkra sig om att användaren påminns om sitt ansvar mot kolleger och omgivning, har därför en informationsruta med varningstext implementerats som första sida i applikationen. För att komma vidare till nästa ruta i applikationen behöver användaren bekräfta att denne har läst varningsmeddelandet.

En aspekt som kan påverka den individuella användaren vid användning av applikationen är färgval. För att undvika problem för färgblinda har projektgruppen tagit fram en färgpalett som är anpassad för olika sorters färgblindhet. Färgpaletten används främst på knapparna i applikationen, medan färgerna i 3D-modellen är tagna från 2D-kartor som från förut är i bruk av Räddningstjänsten.

Givet att alla ovannämnda aspekter tas i beaktande och att varje användare av applikationen får en utbildning i hur den används, kan detta system påverka samhället positivt. Att Räddningstjänsten snabbare än förut får tillgång till informationen som behövs på en olycksplats kan göra att arbetet försnabbas och säkerheten för omgivningen kan återställs effektivt. Eftersom trafikolyckor, som är de mest relevanta olyckorna för denna applikations användning, ofta är vid vältrafikerade platser så är det till stor fördel att Räddningstjänsten snabbt kan bedöma situationen och agera därefter.

Kapitel 6

Slutsatser

Detta kapitel avslutar rapporten och presenterar slutsatser baserat på resultaten från projektet med syfte och frågeställningar i åtanke.

6.1 Hur kan man försäkra sig om en snabb responstid av systemet, med storleken på datafilerna i åtanke?

Eftersom att det uppstod problematik med in-laddning av modeller, som beskrivet i kapitel 3.3.1, gick det inte att konkret testa responstiden för att ladda in modeller ifrån en databas. Däremot ser vi i kapitel 4.3 att när en modell lagras direkt i Unity som en *asset* så går det i princip direkt att ladda in modellen i applikationen utan väntan. Denna modell var under cirka 30 megabyte i filstorlek som beskrivet i kapitel 4.3.

Baserat på den snabba responstiden bör inte tiden för laddning av modeller uppstå som ett problem om applikationen utvecklas fullständigt, baserat på att samma metod för laddning kan användas för samma typ av modeller. Skapas en extern databas eller en databas av en annan typ går det tyvärr inte att svara på om den responstiden hade varit lika snabb.

Det betyder att om applikationen fortsätter att använda samma metod för laddning samt samma modeller och filstorlek så kan man säkerställa en snabb responstid för laddning av modeller i applikationen.

6.2 Kan man, med hjälp av AR, underlätta för en fältarbetare inom Räddningstjänsten Östra Götaland (RTÖG) att identifiera kritiska delar hos ett fordon?

Baserat på resultatet vi fick från kapitel 4.2 ser vi att AR gör det lättare att identifiera kritiska delar i ett fordon för en person som inte är utbildad inom AR. Eftersom det inte gick att låta fältarbetare testa applikationen är det svårt att dra en korrekt slutsats angående om det hade underlättat för dem i deras arbete. Trots detta kan vi ändå se från användartesterna att det är noterbart lättare för en civil person att identifiera ett objekt i ett fordon med AR. Eftersom Räddningstjänstens fältarbetare är minst lika utbildade inom AR som testpersonerna, ty utbildningen nämnd i kapitel 5.3 bör resultaten ej vara sämre för en fältarbetare. Detta betyder att AR hade underlättat i att identifiera kritiskt objekt hos ett fordon.

6.3 Vidareutvecklingsmöjligheter

Applikationen har vidareutvecklingsmöjligheter inom flera olika områden. Ett utvecklingsområde gäller databasen som egentligen skulle implementeras under projektets gång men av tekniska skäl och uppmaning av kunden inte genomfördes. Databasen skulle alltså innehålla registreringsnummer, 2D-kartor och 3D-modeller för olika fordon. En databas som innehåller den informationen är en väsentlig del av applikationen om den skulle användas av personal från räddningstjänsten som är applikationens målgrupp. Det är en väsentlig del eftersom räddningstjänsten måste ha tillgång till dessa kartor och modeller för många, om inte alla fordon i Sverige eftersom vilket fordon som helst kan vara med om en olycka.

Hur en användare manipulerar och interagerar med en utplacerad modell är ett annat område som har hög vidareutvecklingspotential. Användaren kan redan manipulera modellen genom att skala den i ett intervall och rotera den runt sin egen y-axel. Dock så kan användaren inte exempelvis flippa modellen upp-och-ned eller lägga den på sidan för att reflektera hur en bil kan ligga i verkligheten. Detta hade kunnat möjliggöras genom att låta användaren rotera modellen i dess tre rotationsaxlar och inte bara y-axeln. Dessutom kan användaren endast interagera med modellen genom dess specifika sliders på gränssnittet och inte genom att dra i eller röra modellen med sina fingrar. Genom att implementera *touch* möjliggör man för användaren att snabbt kunna utan användningen av användargränssnittet manipulera modellen.

Litteraturförteckning

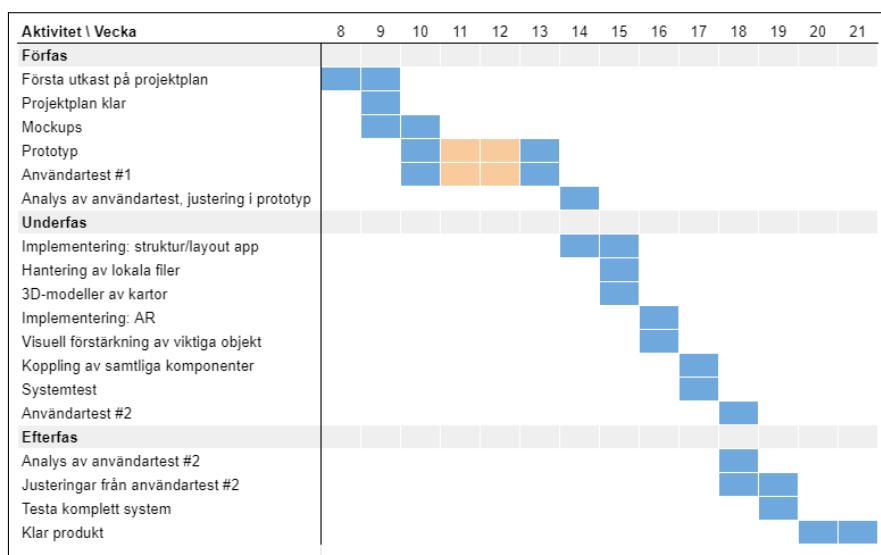
- [1] Applikationsutvecklare. <https://www.framtid.se/yrke/applikationsutvecklare/>.
- [2] Arcore. <https://developers.google.com/ar>.
- [3] Gantt. <https://projektledarbloggen.se/gantt-schema/>.
- [4] Korrekturläsare. <https://www.framtid.se/yrke/korrekturlasare>.
- [5] Mercedes-benz rescue assist now featuring ar to see inside crash vehicles. <https://www.re-flekt.com/portfolio-item/daimler-mb-rescue-assist>.
- [6] Scrum meeting. <https://www.productplan.com/glossary/scrum-meeting/>.
- [7] Sekreterare - förening.se. <https://forening.se/fortroendevalda/styrelsen/sekreterare/>.
- [8] Trello. <https://trello.com/sv>.
- [9] Unity ui: Unity user interface. <https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/index.html>.
- [10] Visual studio. <https://visualstudio.microsoft.com/>.
- [11] Ronald T. Azuma. A survey of augmented reality. In *Presence: Teleoperators and Virtual Environments*, 6(4):355–385, 1997.
- [12] Fahmi Bellalouna, Mika Luimula, Panagiotis Markopoulos, Evangelos Markopoulos, and Franco Zipperling. Fiaar: An augmented reality firetruck equipment assembly and configuration assistant technology. In *2020 11th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, pages 000237–000244, 2020.
- [13] Statistik över vägtrafikolyckor. <https://www.transportstyrelsen.se/sv/vagtrafik/statistik/olycksstatistik/statistik-over-vagtrafikolyckor/>.
- [14] Colorbrewer 2.0. <https://colorbrewer2.org/#type=qualitative&scheme=Set2&n=3>.
- [15] Mathilde R. Desselle, Ross A. Brown, Allan R. James, Mark J. Midwinter, Sean K. Powell, and Maria A. Woodruff. Augmented and virtual reality in surgery. *Computing in Science Engineering*, 22(3):18–26, 2020.
- [16] Figma. <https://www.figma.com/>.

- [17] Myndigheten för sammhällsskydd och beredskap. Räddningsinsatser med gasdrivna personbilar, 2010.
- [18] Michele Gattullo, Giulia Wally Scurati, Michele Fiorentino, Antonio Emmanuele Uva, Francesco Ferrise, and Monica Bordegoni. Towards augmented reality manuals for industry 4.0: A methodology. *Robotics and Computer-Integrated Manufacturing*, 56:276–286, 2019.
- [19] Ground plane. <https://library.vuforia.com/features/environments/ground-plane-guide.html>.
- [20] Karlsson M. Lennell R. Lundgren A. Nygren H. Lips - lätt interaktiv projektstyrning på linköpings universitet. <https://docplayer.se/2455996-Lips-latt-interaktiv-projektstyrning-pa-linkopings-universitet.html>, 2007.
- [21] Max Rehkopf. What is a kanban board? <https://www.atlassian.com/agile/kanban/boards>.
- [22] Roger N. Shepard and Jacqueline Metzler. Mental rotation of three-dimensional objects. *Science*, 171:701–703, 1971.
- [23] Rick Van Krevelen. Augmented reality: Technologies, applications, and limitations. 2007.
- [24] Getting started with vuforia in unity. <https://library.vuforia.com/articles/Training/getting-started-with-vuforia-in-unity.html>.
- [25] Dan Wargclou. Räddning vid trafikolycka - personbil, 2010.
- [26] Torbjörn Åkerstedt, David Hallvig, Anna Anund, Carina Fors, Johanna Schwarz, and Göran Kecklund. Having to stop driving at night because of dangerous sleepiness – awareness, physiology and behaviour. *Journal of Sleep Research*, 22(4):380–388, 2013.

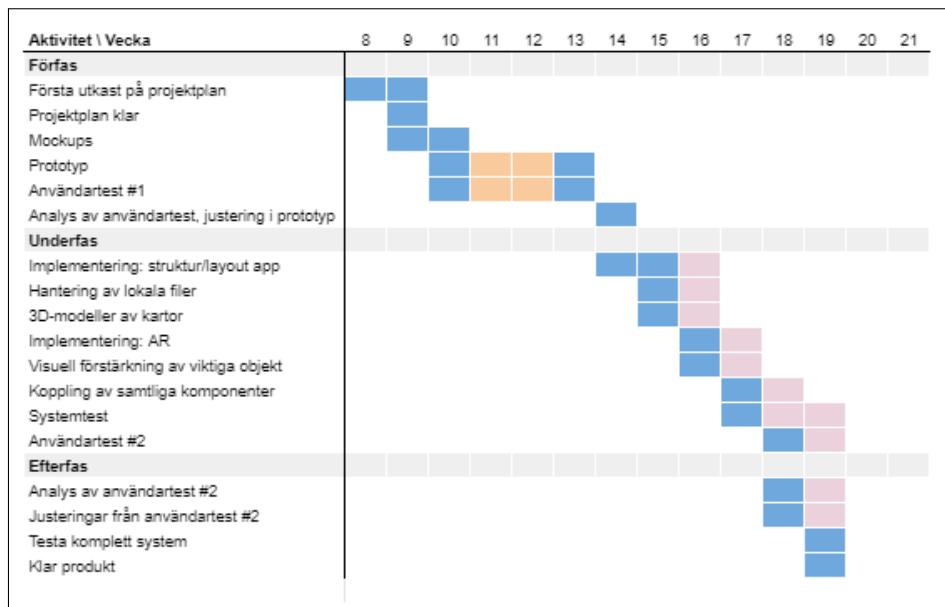
Bilaga A

Reflektion över systemutvecklingsprocessen

Att komma igång med projektet gick lite långsamt. Det var svårt att greppa var vi skulle börja arbetet, och att veta hur allting skulle planläggas. När vi väl satte igång med projektplanen och hade ett färdigt GANTT-schema[3], såg processen lite klarare ut. GANTT-schemat visas i figur A.1 och det var vår tänkta tidsplan för projektet. Hur vi egentligen arbetade visas istället i figur A.2 där de rosa markeringarna indikerar när saker blev klara. När vi jämför dessa ser vi att i början av projektet lyckades vi med att hålla oss till den planerade tidsplanen ganska bra.



Figur A.1: GANTT-schemat till projektplanen. De orangemarkerade veckorna är när projektgruppen har uppehåll. Blåmarkerad ruta innebär arbetsvecka.



Figur A.2: GANTT-schema som visar hur det faktiskt blev. De rosamarkerade veckorna är de som skiljer sig åt från originalplanen.

Som ses i figurerna var det efter första användartestet som saker inte längre gick som tänkt. Man ser då att alla delar blev klara en vecka efter den planerade deadlinen. Detta berodde till största del på att implementationen av Vuforia med Unity och Git inte var lika lätt som tänkt och tiden det tog att lösa var helt enkelt inte inplanerat i arbetsgången. Själva arbetsprocessen fungerade dock i huvudsak som tänkt då vi delade upp arbetet jämnt mellan oss och hjälpte till med varandras uppgifter om vi blev klara med våra egna. Detta resulterade i att några av oss tog del i flera av projektets delmoment och det tyckte vi fungerade bra.

Att direkt från början dela ut roller för alla visade sig vara svårt. Några i gruppen fick klara administrativa roller, medan andra blev utan. Tanken var att vi under arbetets gång skulle se vilka fler roller som behövdes. Eftersom vi inte följde någon utvecklingsmetodik som hade flera bestämda roller så fick vi inom gruppen bestämma några roller och ansvarsområden som vi kände var viktiga. Vi hade inga bestämda arbetsroller, såsom till exempel utvecklare, utan valde istället att ge alla medlemmar möjligheten att själva välja arbetsuppgifter utifrån intresse.

Användningen av Kanban[21] har fungerat halvbra. Ingen har kontinuerligt uppdaterat sina uppgifter där trots att det inom gruppen bestämdes att så skulle göras. Istället har ansvarsfördelningen skrivits upp i listformat som sedan har delats till alla gruppmedlemmar på Discord, och sen har det varit avstämning av arbetsframgång under varje möte. Detta system har fungerat bra för gruppen eftersom vi oftast haft möten flera gånger per vecka. Inför implementeringsarbetet fyllde alla i *Trello*, där framstegen i uppgifterna skulle uppdateras så att samtliga i projektgruppen kunde se hur arbetet framskred. Trello är en webapplikation i Kanban-stil, där flera användare samtidigt kan lägga till uppgifter i listor. I grund och botten fungerar det som en virtuell tavla där användarna kan flytta sina uppgifter mellan olika rubriker när arbetet framskrider, t.ex. från 'Ej påbörjat' till 'Klart' [8]. Om vi hade lagt större fokus på att under varje möte kontrollera Trello hade nog detta system fungerat bättre i längden.

Trots oklarheter i början, och aningen otydliga roller så har arbetet framskridit med god fart och framgång. Även om allting med implementeringen blev försenat har en färdig produkt framställts i tid enligt tidsplanen genom gemensamt arbete inom alla i gruppen.

A.1 Cecilia Bergman

Som sekreterare har det varit min uppgift att föra protokoll på möten[7] och se till att dessa finns tillgängliga för alla gruppmedlemmar på Google Drive. Arbetet som kommer med rollen har fungerat bra för mig, även om det ibland varit svårt att veta vad som varit viktigt att skriva ner. I sådana fall har jag bett övriga gruppmedlemmar komplettera mina anteckningar, så att inget blir bortglömt. Korta avstämningsmöten har inga protokoll eftersom gruppen under sådana möten gjort "Att göra"-listor som istället har skickats i Discord, där all vår kommunikation skett.

A.2 Fredrik Burmester

I detta projekt har min roll varit ordförande, en roll som varit mestadels ledande. Jag har hållit talan och drivit möten, planering och deadlines.

Rollen som ordförande ändrades inte under projektets gång och rollen har passat väl då jag ofta tar talan i övriga sammanhang. Rollen enligt teorin är en ledande roll med flertal arbetsuppgifter, så som att kalla till möte med en tydlig dagordning och se till att följa den, styra mötet och se till att inkludera alla, varav allt har följts.

Samarbetet med andra i gruppen har fungerat utan problem och eftersom Cecilia antecknat varje möte har det varit väldigt enkelt för mig att ha en god struktur på möten, då mitt minne inte är det bästa. Att hålla i möten har fungerat naturligt och jag tror alla har känt att de har fått sin vilja och talan igenom. Sedan början har vi haft ett anonymt kritiksystem där klagomål har kunnat skickas in till mig som ordförande eller till vice. Vi har dock inte fått någon anonym kritik, vilket jag hoppas talar till att allting fungerat bra.

A.3 Hugo Dahl

Jag har som uppgift att vara schemaläggare. Min uppgift var att hålla koll och skapa en Kanbantavla[21]. En Kanbantavla är till för att visualisera uppgifter som gruppen behöver göra, vilka uppgifter som gruppmedlemmarna jobbar på just nu och vilka uppgifter som är klara. På detta sätt har alla i arbetsgruppen koll på hur projektet och produkten ligger till.

I projektet användes bara Kanban väldigt lite i början av implementeringen av alla i gruppen inklusive mig. Kanbantavlan ersattes i princip av ett flertal möten i veckan där varje person gick igenom på vad de hade gjort sen sist och vad de håller på med eller ska göra. Allt detta antecknades av sekreteraren och gick sedan att läsa i gruppens delade Google Drive-mapp.

A.4 Johnny Elmér

Som kontaktansvarig var det mitt ansvar att ta hand om kontakt med kund, kursansvarig samt att förmedla information mellan dessa och de andra gruppmedlemmarna. De metodiker (en blandning av LIPS[20] och Kanban[21]) som valdes för projektarbetet gav ej några specifika riktlinjer för vad kontaktansvarig bör göra, därför framtogs dessa arbetsuppgifter inom gruppen under diskussion.

Min roll som kontaktansvarig fungerade bra, även om lärare vid ett tillfälle missförstod och istället tog kontakt med en gruppmedlem som ej hade denna roll. Utöver rollen som kontaktansvarig så tog

jag under projektet på mig att se till att vi bokade in kommande möten innan vi avslutade för dagen. Jag såg även till att dessa datum och tider skrevs ner i en delad textkanal i Discord så att alla hade tillgång till dem. Denna roll i sig hade inte något namn och var ej förbestämd men blev naturligt min då jag automatiskt tog ansvaret för detta. Man skulle kunna se detta som mer av en uppgift för en schemaläggare (Hugos roll).

Då det ej direkt finns någon teori att jämföra med (se första stycket i detta delkapitel) så är det svårt att säga om rollen som kontaktansvarig skiljer sig något från vad den teoretiskt bör innehålla. Den tillkommande rollen som framstod naturligt var ej baserad på en existerande teoretisk roll utan tillkom utifrån gruppens behov.

Jag har även varit ansvarig för att ta fram/skapa 3D-modellen, detta var dock ej en roll direkt utan endast ett ansvarsområde. Jag beskriver detta mer tydligt i bilaga B (se B.3).

A.5 Simon Nilsson

Jag har inte blivit tilldelad någon specifik arbetsroll i den traditionella meningen så som ordförande eller sekreterare. Under projektet arbetade jag som utvecklare. Kanban-systemet efterfrågar inte speciella arbetsroller men jag hade större ansvar inom UX och design av produkten. I efterhand känner jag att metodiken som vi valde fungerade eftersom projektets delmoment inte skiljde sig åt i en större grad. Det gjorde att vi kunde vara flexibla i vem som arbetade med vad. Eftersom vi alla arbetade parallellt med separata *git branches* var ett av mina ansvarsområden att säkerhetsställa att master-branchen låg i fas utan diverse buggar på grund av *merge-konflikter*. Git fungerade bra överlag dock valde vi att lägga upp alla tillhörande filer i projektet. I efterhand när jag reflekterar kring hur arbetet gått jämfört med projektplanen och de systemutvecklingsmetoderna vi valde anser jag att Kanban inte var det bästa för vårt projekt. Scrum kanske skulle varit fördelaktigt med dagliga Scrummöten, sprint-planering och *Sprint Retrospective* [6] då vi senare under projektet började med kortfattade möten som liknar Scrummöten för att hålla vår produktivitet uppe.

A.6 Adrian Szuter

I början av projektet blev jag tilldelad rollen som korrekturläsare men under projektets gång tog jag även en roll som utvecklare. Som korrekturläsare gick jag genom både projektplanen och kandidatrapporten och korrigrade stavnings- och grammatikmisstag innan dessa lämnades in. Som utvecklare arbetade jag med applikationen genom att skapa scener och funktionalitet för knappar med olika scripts.

Rollen som korrekturläsare är enligt teori att man kontrollerar att en text följer språkets regler och normer. Detta innefattar att en korrekturläsare går genom texten och markerar stavfel, grammatiska fel, särskrivningar och ser över meningsuppbryggnad samt punktsättning [4]. Rollen som utvecklare fungerade mer som ett ansvarsområde än en konkret roll vilket gör det svårt att koppla till konkret teori då den även var ganska generell.

Det är svårt att avgöra om min roll som utvecklare skilde sig mycket från de teoretiska beskrivningarna då det fungerade i huvudsak som ett ansvarsområde och var generell. Dock om man jämför med andra roller eller yrken som exempelvis en applikationsutvecklare som arbetar med design och implementation av applikationer så finner man likheter då jag arbetade med applikationens funktionalitet [1]. Dessutom skilde sig inte min roll som korrekturläsare speciellt mycket från teorin. Dock så blev korrekturläsningen på en låg nivå då jag inte är en expert i svenska språkregler eller språknormer. Detta resulterade i att vissa fel som exempelvis stavfel eller särskrivningar fanns kvar i rapporten när

den lämnades in och detta är definitivt ett förbättringsområde för mig.

Bilaga B

Individuella bidrag

En lista över vilka som varit involverade i projektet, inom teamet och utanför, med en beskrivning av någonting som de, genom individuellt, självständigt arbete, bidragit med.

B.1 Cecilia Bergman

Jag har tillsammans med två andra gruppmedlemmar ansvarat för att skapa sidor till applikationen och implementera funktioner för de knappar som finns på sidorna. Vad som ingick i detta arbete beskrivs i vår metod, i kapitel 3.4. Arbetet var uppdelat så att jag fick en tredjedel av sidorna vilket innebar 4-5 sidor som skulle skapas och sammanhangande script skulle implementeras, medan de två andra som implementerade även fick vardera 4-5 sidor. Implementeringen har flytit på, men ibland har det tagit mer tid än förutspått. Dessutom har jag haft delat ansvar för projektets två användartest som nämns i kapitel 3.2 och vars resultat presenteras i kapitel 4.2. Det ansvaret har inneburit att ta fram frågor som är aktuella för respektive test, tillvägagångssätt för respektive användartest och att sammanställa formulär och intervju-upplägg för testen.

B.2 Fredrik Burmester

Under projektet har jag bidragit i skapande av användartester, programmerat en databas och tillhörande hjälp-filer för sparandet av registreringsplatser och modeller för bilar i vår applikation. Jag har också samarbetat med Hugo kring hantering av våra projektfiler och hur vi hanterar kod på Github - vilket visade sig vara ett stort problem med kod från Unity och Vuforia - samt hjälpt till med mindre saker som att konvertera 3D-modeller till spel-objekt i Unity som sedan kunde användas i vår AR-vy.

På grund av motgångar med Vuforia och dynamisk laddning av modeller i Unity kommer databasen som jag skapade inte att användas. Vi bestämde oss istället, på uppmaning av vår kund, att endast skapa en enklare lösning för laddning av modell, som inte kräver en databas. Tyvärr hade det här inte gått att kringgå då problemet uppstod i samband med laddning av modeller från själva databasen och vi hade inget sätt att veta att dynamisk laddning inte funkade som vi ville.

B.3 Hugo Dahl

Jag har tillsammans med Fredrik satt upp filhantering och delad kod på Github, detta visade sig från början vara krångligare än först väntat, på grund av att Unity och Vuforia använder sig av väldigt många och stor filer. När väl all versionshantering var löst bestod min huvuduppgift av implementera AR-scenen. Jag lyckades lösa att en 3D-modell lades ut på en plan yta när en användare klickar på skärmen. Jag försökte även lösa dynamisk inladdning av 3D-modeller från databasen som skapades av Fredrik.

Den dynamiska inladdningen fick jag dock inte att fungera på grund av hur Vuforia fungerar. En 3D-modell var tvungen att initieras från appens start. Detta var inte möjligt då applikationen inte kunde veta från början vilken modell som skulle laddas in.

B.4 Johnny Elmér

Mitt (Johnnys) personliga bidrag till projektet är kopplat till 3D-modellen. Jag letade upp en redan framtagen modell för en bil på nätet och laddade ned denna. Sedan var det mitt ansvar att göra denna modell transparent och att modellera de invändiga delarna utifrån en 2D-karta som referens. Den färdiga modellen var den som sedan skulle användas av AR-applikationen.

B.5 Simon Nilsson

Jag har främst arbetat med UI-utveckling för programmet. Där skapade jag relevanta C#-script för att uppnå funktionaliteten som söktes från vår gemensamma design. Jag har också kopplat ihop de olika scenerna. Jag skapade sidan för inmatning av registreringsskylten, val mellan 2D och AR samt menyn med en 2D-karta. Jag har även arbetat med felsökning när projektet skulle byggas till Android och IOS.

B.6 Adrian Szuter

Jag har under projektets gång haft som uppgift att både skapa användargränssnittet för AR-vyn och dess relevanta funktioner i Unity. Alltså har jag bidragit med att placera ut rutor, knappar och texter utefter vår Figma och sedan skapat C#-script för deras funktioner. Specifikt sagt så har jag skapat funktionaliteten för skalning och rotation av den utplacerade 3D-modellen. Dessutom har jag hjälpt till med att ta fram relevanta frågor för det första användartestet och delat ansvar för hopkopplingen av alla sidor och då även sett till så att allt fungerar som tänkt.