Risikoer kan ha negativ effekt på en programvares kvalitet og kan påvirke tidsplanen negativt. Derfor gjorde vi en risikoanalyse i starten av prosjektet med hensikt om å gi grunnlag for gode beslutninger. Det kan ofte forekomme mye endringer underveis i et slikt prosjekt, noe som kan være en risiko i seg selv. Det er derfor nyttig å ha en plan for å håndtere dette. I denne hensikt satte vi opp en Risikotabell (se vedlegg nr.?). Tabellen viser sannsynligheten for at en risiko ville skje, risikoens konsekvens på en skala fra 1-5, tiltak for å unngå at en risiko skulle skje og reparerende tiltak dersom risikoen faktisk skulle inntreffe. I tabellen valgte vi å ikke ta med oversikt over hvem som hadde ansvar for hver risiko, fordi vi mente at hvert punkt var noe alle burde ta ansvar for dersom det skulle oppstå problemer.

Underveis i prosjektet har vi prøvd å forholde oss til risikoanalysen og følge de avvergende tiltakene for å unngå at noe skulle gå galt. Spesielt for de punktene med høy sannsynlighet og stor konsekvens. VI utarbeidet den vedlagte tabellen ved starten av prosjektet og har revidert analysen jevnlig. Da den originale tabellen stort sett har vært oppfyllende gjennom hele prosjektet har store forandringer ikke vært nødvendige.

I tabellen står manglende kompetanse oppført som det mest kritiske punktet, det er også på dette området av prosjektet vi har møtt de største problemene. Ingen av oss hadde prøvd android utvikling da vi startet prosjektet, det har derfor krevd en del fordypelse å sette seg inn i dette. Til tross for diverse utfordringer har oppgaven vært spennende og lærerikt. Mot slutten har vi hatt problemer med out of memory feil på enheter med mindre RAM. Dette har vi ikke vært i stand til å rette opp i ettersom vi mangler erfaring og kunnskap med hvordan android håndterer bilder og mellomlagring. Vi har måttet gi slipp på ideer vi har hatt fordi vi ikke har hatt kunnskapen til å gjennomføre dem eller hatt tid til å lære teorien rundt. For eksempel så ønsket vi å kunne krysse av og velge flere sauer samtidig i sauelisten, noe som ble for omfattende i denne omgang. I tillegg syntes vi at koden vår etter hvert ble stor og rotete om følge av manglende struktur i begynnelsen. Da vi i starten manglet erfaring med utvikling over en større skala, hadde vi ikke den rette innsikten i hvordan planlegge en kodeoppbygging. Dette medførte mye unødvendig kode som kunne blitt unngått, dersom vi i starten hadde hatt en bedre forståelse av prosessen.

Det har ikke vært mange problemer i løpet av perioden i forhold til punktene vi satte opp i risikoanalysen, og av de få vi har kommet over har de fleste vært kompetanserelaterte, som vil bli tatt opp i neste avsnitt. Utover dette har det vært problemer med at vi ikke har overholdt avtaletider, fordi vi hadde satt opp så lange arbeidsøkter på de dagene vi skulle ha møte. Dersom vi hadde satt kortere møtetid hadde mest sannsynlig flere kommet tidsnok. Vi valgte til tross for dette å fortsette med lengre arbeidsøkter som ga oss bedre mulighet til å jobbe sammen og til å hjelpe hverandre.

**Verktøy  
Scrumwise**En av fordelene ved scrum skal være at hele teamet skal kunne observere og ha oversikt over hva som skjer i prosjektet. Vi hadde ikke mulighet til å ha stand up møte hver dag, ettersom alle har forskjellige timeplaner og ikke har tid til å møtes hver dag. Derfor mente vi at vi trengte et verktøy til å holde oversikt over hvem som holdt på med hva og hvordan vi lå an med sprinten til enhver tid. Allerede første møte bestemte vi oss for å prøve scrumwise som er et verktøy der all prosessinformasjon ligger lagret på et sted. Den gir god oversikt over både backlog, sprinter, taskboard, estimater og genererer et burndownchart. Vi valgte på slutten av første sprint å slutte å bruke scrumwise, fordi det ble lite brukt og utover en måneds prøveperiode ville vi måtte betale. Vi følte dette var unødvendig når vi brukte det så lite, og vi ville heller bruke tid på å lære oss Android programmering.

**Trello**Som et alternativ til scrumwise begynte vi å bruke trello som task board. Trello er et samarbeidsverktøy som organiserer prosjekter inn i boards, eller tavler. Vi laget et nytt board for hver sprint, og la tilsvarende oppgaver inn i disse. På denne måten har vi hatt oversikt over hvilke oppgaver som var påbegynt, hvilke som var ferdige og hvilke vi ikke hadde startet på. I tillegg har vi kunnet se hvem som jobbet med hva, legge til kommentarer til oppgavene og vise eventuelle oppgaver som måtte flyttes til neste sprint. Flere på gruppa hadde erfaring med bruk av trello fra før og det har derfor blitt mer brukt, i tillegg til at trello har vært mye mer oversiktlig enn scrumwise.

**Github**   
Github er hovedsaklig et webbasert versjonskontrollsystem. I en slik utviklingsprosess som vi har hatt, der flere personer koder på samme kodebase samtidig, er et versjonskontrollsystem veldig viktig. Et slikt system passer på at ingen kode går tapt, man vet hvem som har gjort hvilke endringer, og den vil prøve å forhindre konflikter hvis to personer har jobbet på samme fil. Måten man jobber med github på er at hovedkodebasen ligger i skyen på github sine servere og når man skal endre på kodebasen lager man en forgreining av denne. Dvs. at du kopierer kodebasen ned til din lokale datamaskin. Her kan du gjøre alle endringene du vil for så å dytte koden opp til github. Når du har gjort dette kan du slå sammen dine endringer med den originale koden. Github vil nå prøve å flette dette sammen automatisk, noe som vanligvis går veldig bra, men i ny og ne klarer den det ikke og du må da selv si hvilke endringer som skal være med.

Github har også andre fine funksjoner som et hvert versjonskontrollsystem burde ha: den lagrer alle endringer siden du startet prosjektet, og du kan derfor rulle tilbake til en tidligere versjon av koden om noe skulle gå galt. I tillegg til å være et versjonskontrollsystem tilbyr Github en del andre funksjonaliteter. Du har muligheter for å legge til såkalte "issues" og gi dem til folk som jobber på prosjektet. På denne måten kan du se hva hver person skal gjøre. Hver eneste kodelinje i kodebasen kan spores til en person, og man vet da hvem som har skrevet hva.

Vi fikk tilbud om en SVN server fra studass, men siden noen på gruppa hadde erfaring med Github fra før bestemte vi oss for å bruke dette siden det er gratis.

**Eclipse**  
Da vi bestemte oss for å gå for androidutvikling ble vi rådet til å laste ned ADT Bundle som egner seg veldig bra for uerfarne androidutviklere. ADT Bundle inkluderer viktige Android SDK komponenter og en versjon av Eclipse, et kodebehandlingsprogram, med innebygd ADT (Android Developer Tools). Fordelen med bruk av Eclipse fra ADT Bundle pakken var at det var raskt og enkelt å komme i gang. De innebygde ADT funksjonene gjorde det lett for oss å opprette et nytt android prosjekt, lage UI til applikasjonen og debugge applikasjonen ved hjelp av Android SDK-verktøyene. I tillegg til at alt koderelatert var lett tilgjengelig innenfor programmet, noe som gjorde det veldig oversiktlig å jobbe med.

**PostgreSQL**  
For å lagre data fra flere bønder var i nødt til å ha en database. Valget vårt havnet på PostgreSQL siden det var det eneste databasehåndteringssystemet noen i gruppa hadde brukt før. I tillegg har PostgreSQL åpen kildekode som er veldig solid og stabil, og det er ingen restriksjoner ved bruk av den. Vi har valgt å ta backup av all dataen i databasen hver natt slik at hvis en systemkrasj skulle inntreffe vil vi ha all dataen lagret på en annen server. Databasen er ikke tilgjengelig utenifra, dette fordi da har vi ikke noen mulighet for å sanitere sql spørringene som kommer inn. Dette er viktig siden ellers ville vi vært svært åpne for SQLInjections. Vi har derfor valgt å ha et mellomlag mellom database og applikasjon som saniterer data og håndterer eventuelle feil.

**Dropbox**  
For at alle på gruppa skulle ha tilgang til alt vi har laget av diagrammer, dokumenter og bilder opprettet vi en felles dropbox mappe. Dropbox er et delingsverktøy der man lagrer filer i mapper som man har mulighet til å dele med andre, noe som gir god støtte for samarbeid. En ulempe med dropbox har vært at når vi skulle jobbe med dokumentasjonen har ikke flere kunnet endre på samme fil samtidig uten å overskrive det den siste personen har gjort. Vi løste dette problemet ved å opprette en ny tekstfil når vi skulle legge til noe i rapporten. Deretter fikk dokumentasjonsansvarlig ansvar for å sette de forskjellige tekstene inn i hovedrapporten. Ellers er dropbox veldig lett å håndtere og forstå, som var grunnen til at vi valgte å bruke det som delingsverktøy.

**Google Maps API**  
En av hovedgrunnene til at vi valgte Googles karttjeneste, var at å implementere Google Maps på Android er ganske enkelt og sømløst, i forhold til alternativene. Vi så også på muligheten for å erstatte kartene fra Google med kartverkets kart, og da samtidig beholde Google API sine kartfunksjoner. Tanken bak å gjøre dette var at vi ønsket å sikre et mer detaljert kart, men etter dialog med kunden valgte vi å prioritere andre funksjonaliteter siden kartet til Google fungerte, og det gir mulighet for å se sauene i en type topografisk kart.

**Office Word**Til å håndtere tekst, tabeller og bilder i rapporten og til møtereferater har vi stort sett brukt Office Word. Dette har ikke vært helt optimalt for de på gruppa som ikke har windows-pc da visningen ikke ble helt lik, men alle har fortsatt kunnet lese og endre tekst. Vi valgte å bruke word ettersom det gjorde det lett for oss å opprette, redigere og formatere både tekst, bilder og tabeller.

**Office Excel og OmniGraffle Pro**I tillegg til word har vi brukt Office Excel og OmniGraffle Pro til å lage tabeller og diagrammer. Vi valgte OmniGraffle som er et gratis UML-behandligsprogram fordi det gjorde det lettere og raskere for oss å lage UML-diagrammer.

**Photoshop, Illustrator**  
For å lage ikoner, knapper og bakgrunner til applikasjonens grafiske brukergrensesnitt har vi brukt programmene Photoshop og Illustrator. Illustrator er et redigeringsverktøy for vektorgrafikk og Photoshop er et bilderedigeringsprogram. Grunnen til at vi brukte disse programmene til design er at designansvarlig har god erfaring med dem, og for at designet skulle bli best mulig valgte vi å holde oss til det vi kjente fra før.

**Facebook**Mye av kommunikasjonen i gruppa, utenom møtetider, har foregått på facebook. Facebook er noe vi alle sjekker jevnlig, og det ble derfor naturlig for oss å bruke dette til å avtale møter, gi hverandre beskjeder og spørre om ting vi lurte på.

**Nettbrett og android telefon til testing**  
Vi syntes android-emulatoren som er innebygd i Eclipse ATD var veldig sen til å starte opp og til å kjøre, derfor har vi brukt android telefoner og nettbrett til å teste applikasjonsfunksjonene underveis. Scrum master Fredrik Borgen Tørnvall gikk til innkjøp av et android nettbrett til prosjektet for å ha en debuggingsenhet. Det er denne vi har lagt hovedfokuset vårt på, ettersom kunden forsikret oss om at applikasjonen kun trengte å virke på en enhet. Derfor har vi hovedsakelig konsentrert oss om at alt skal virke på nettbrettet, men har fått det til å virke på andre enheter også.

**Teknisk brukermanual:**Ettersom vi har valgt å konsentrere oss om å utvikle til nettbrettet, ser vi helst at testing foregår på denne. Men for å kjøre programmet vårt på en annen enhet eller på en emulator gjøres dette lettest ved hjelp av Eclipse med ADT-plugin og SDK-verktøy, ADT Bundle anbefales. Ettersom programmet ikke ligger ute på google play store må det gjøres på denne måten.

For å laste inn koden i Eclipse går man inn på *file > import > android > existing android code into workspace*, og importerer mappen som heter Find\_my\_sheep. For å kjøre programmet må man koble en androidenhet til datamaskinen og skru på USB-feilsøking i enhetens innstillinger. Man kan også opprette en emulator i Eclipse med Android Virtual Device Manager under *window*. Emulatoren burde ha en størrelse på 1204 x 600. Etter å ha satt opp en enhet er det bare å trykke på run. For å få opp andorid debug output-vindu trykker man på *window > show view > other > android > log cat*.

For å kunne se kartet når man kjører programmet må man laste ned Google Play Services fra Android SDK manager, også under *window*. Etter at denne er lastet ned må man trykke på *file > import > android >existing android code into workspace* igjen og importere mappen *sdk >extras > google > google\_play\_services > libproject > google-play -services\_lib*. Når denne er importert høyreklikker man på prosjektmappa og velger *properties (eller preferences) > android >add > google-play-services\_lib*. Dersom denne ikke er her er den allerede lagt til. Til slutt må man legge til debug.keystore for å få riktig Google Maps API Key. Dette gjøres ved å gå inn på *window > preferences > android > build* velge browse på costum debug keystore og legge til filen debug.keystore copy som ligger vedlagt vårt prosjekt.