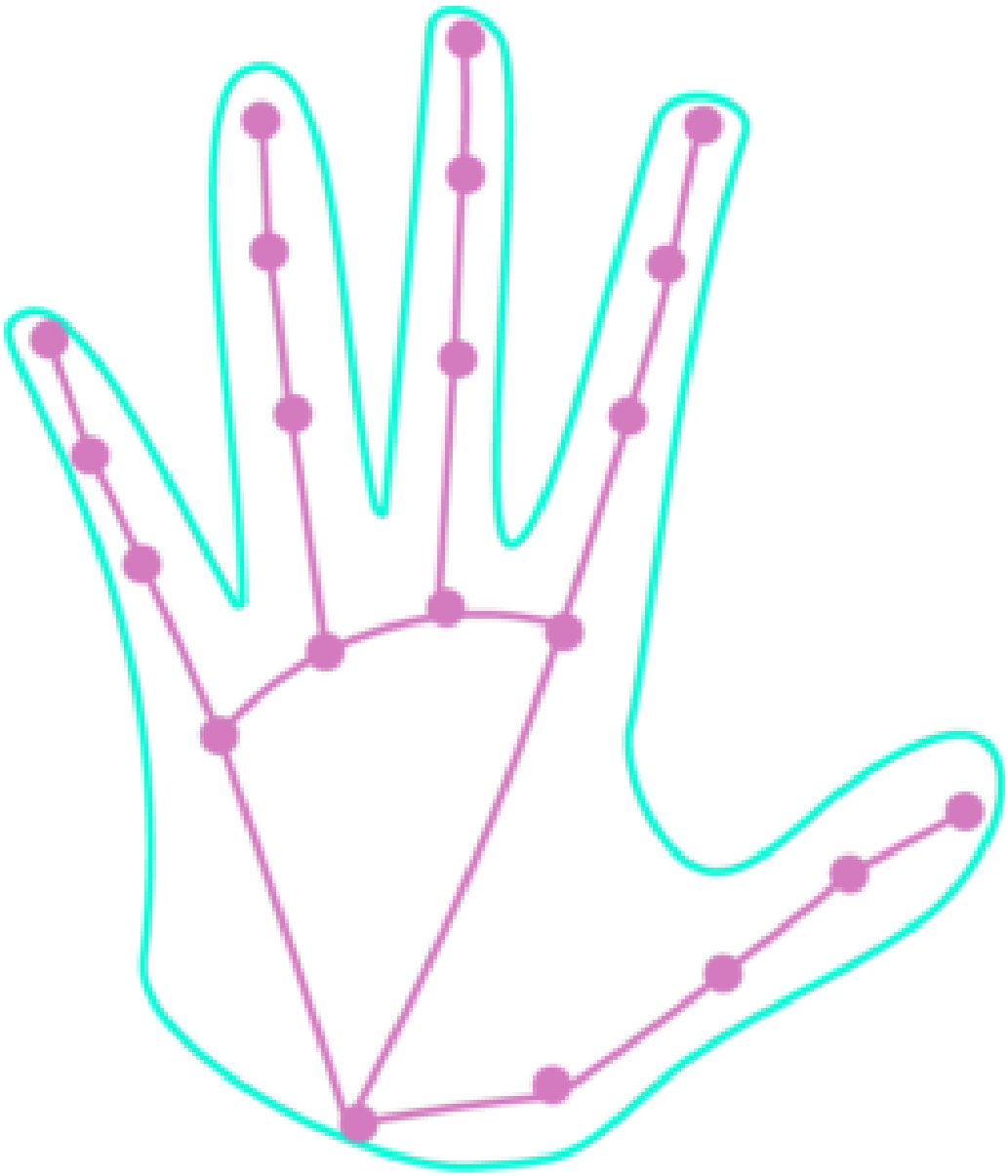
HandyRecipes



Av: Siri Nordstrand Eielsen, Sondre Langedal Ness og Jobjørn Røkenes Myren

Introduksjon

Denne rapporten er basert på rapportmalen gitt i DAT158, men siden den er tilpasset et prosjekt der en selv trener modellen, har vi gjort noen endringer. Siden vi bruker en ferdigtrent modell legger vi fram hvor mye data den er trent på og hva input og output er og hvordan vi har implementert det i vår løsning. Vi har ekskludert modellering, og heller fokusert på å få fram framgangsmåte på implementasjonen.

1.0 Scope

1.1 Prosjektets mål

HandyRecipes er en applikasjon som skal gjøre det enklere og mer hygienisk å følge matoppskrifter. Ved hjelp av computer vision vil applikasjonen gjenkjenne håndbevegelser som input, slik at brukeren enkelt kan bla frem og tilbake i oppskriften med enkle håndbevegelser, uten å måtte berøre skjermen.

Mobiltelefoner og tastaturer er blitt en naturlig del av hverdagen vår, men undersøkelser viser at disse ofte inneholder et betydelig antall bakterier [1]. Kombinasjonen av bakterier på enheter vi håndterer daglig og matlaging understreker behovet for en mer hygienisk tilnærming til matforberedelse. HandyRecipes sitt mål er derfor å bidra til tryggere og renere matlaging ved å redusere behovet for å berøre enheten underveis i matlagingsprosessen.

1.2 Nåværende løsninger

I dagens marked finnes det flere løsninger som sikter mot å gjøre matlaging enklere ved hjelp av teknologi. Apper som Recipe Keeper tilbyr oppskriftsorganisering med funksjoner som import av oppskrifter fra nettsteder, skanning av håndskrevne oppskrifter, måltidsplanlegging og handlelister[2]. Yummly er en annen tjeneste som tillater brukere å søke etter, og lagre matoppskrifter basert på personlige preferanser og kostholdsbehov[3]. Nettsider som godt.no tilbyr en rekke oppskrifter, inspirasjon, og faglige matartikler[4].

Handy Recipes skiller seg ut ved å satse på håndbevegelseskjenning for å navigere gjennom oppskrifter uten behov for fysisk berøring. Ved å kombinere funksjonaliteten til tradisjonelle oppskriftsapper med bevegelsesteknologi, adresserer HandyRecipes et nytt og spennende område i markedet. Denne løsningen tilbyr en mer praktisk og hygienisk løsning for hjemmekokker å følge oppskrifter på.

1.3 Business metrics

1.3.1 Brukerengasjement

Ytelsen til produktet skal måles på brukerengasjement og tilfredshet med gest-baserte interaksjoner. Høy frekvens av gest-basert navigasjon vil indikere at brukere foretrekker den berøringsfrie funksjonen. Dette vil indikere at prosjektet har verdi for brukere.

1.3.2 Brukertilfredshet

Ved å samle inn tilbakemeldinger fra brukere kan vi få innsikt i hvordan appen oppleves, spesielt hvor praktisk og pålitelig gest-gjenkjenningen er. Brukertilfredshet kan måles gjennom spørreundersøkelser og vurderinger og vil gi et godt grunnlag for videre utvikling av produktet.

1.4 Interessenter(Stakeholders)

1.4.1 Primære brukere

Hjemmekokker og matentusiaster er individer som kommer til være direkte brukere av appen. Disse verdsetter funksjonalitet og brukervennlighet. Det blir veldig viktig med tilbakemeldinger fra denne gruppen for å forbedre appens funksjoner og brukeropplevelser

1.4.2 Utviklere

Utviklere har ansvar for utvikling, vedlikehold og forbedring av appen. Disse har og et ansvar for å adressere etiske og tekniske utfordringer, som å sikre personvern og håndtere potensielle utfordringer.

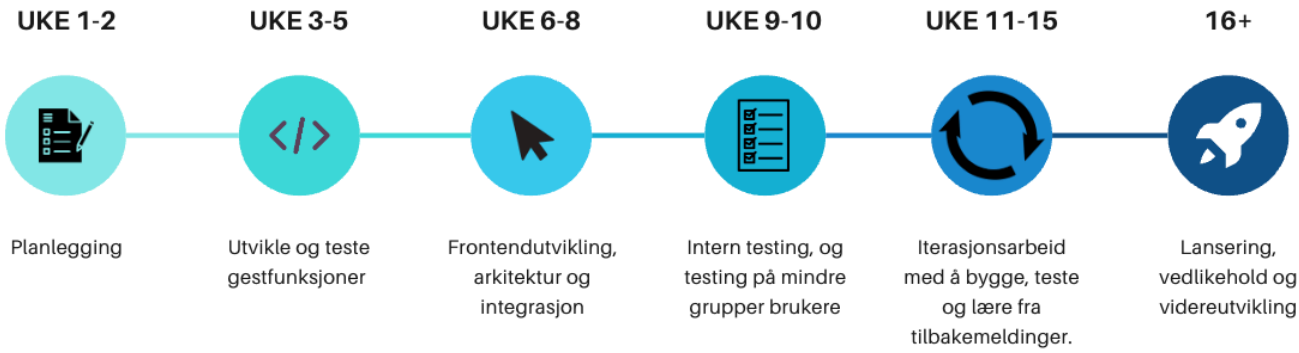
1.4.3 Forretningspartnere

For at HandyRecipes skal vokse og bli en kommersiell suksess, er det avgjørende å få med forretningspartnere som ser verdi i løsningen. Potensielle partnere kan inkludere matvaremerker og leverandører av smarthjemløsninger, som begge vil kunne dra nytte av å tilby en innovativ og hygienisk matlagingsopplevelse til sine kunder.

1.5 Tidslinje for prosjektet

Prosjektet vil innledningsvis følge en fossefallsmodell for å utvikle grunnleggende funksjonalitet som finnes i tradisjonelle oppskriftsapplikasjoner. To utviklingsteam vil bli etablert: det ene med fokus på nettsidens arkitektur og datalagring, og det andre på utforskning og implementering av OpenCV og MediaPipe i Python. Etter fullføring vil teamene samles for å integrere komponentene til en Minimum Viable Product (MVP). Deretter vil prosjektet gå over til en Lean Startup-metodikk, som innebærer kontinuerlig utvikling og testing av MVP-er for å validere sentrale hypoteser og tilpasse produktet basert på tilbakemeldinger[5].

Estimert tidslinje for prosjektlengde for eventuelt oppstart av full utvikling:



1.6 Nødvendige ressurser

1.6.1 Programvareverktøy

Google MediaPipe
Google MediaPipe er et rammeverk designet for å implementere maskinlærings- og kunstig intelligens-løsninger på ulike plattformer. Vi skal bruke MediaPipes hånd-løsning for å finne hender og tegne landmarks i sanntid.

OpenCV
OpenCV brukes for å få tilgang til kamera på enheten. Dette fungerer godt når programmet kjører direkte på enheten, men ikke når siden kjører i skyen som på Streamlit. Derfor blir OpenCV hovedsakelig brukt til testing før vi finner en passende browserløsning.

Streamlit
Streamlit benyttes til å lage brukergrensesnittet og for å deploye nettsiden. Streamlit har en brukervennlig og rask arbeidsflyt som gjør det enkelt for oss å jobbe iterativt, og regelmessig teste det deployede systemet mot brukere.

Snowflake
Snowflake integrerer streamlit i sine løsninger og har betalte planer for å hoste større plattformer.

Python
Vi har valgt Python som scriptspråk. Python tillater oss å utvikle løsninger raskt, og tilbyr et omfattende bibliotek, inkludert Googles MediaPipe-løsning, som vi bruker i prosjektet.

JSON

I startfasen bruker vi en statisk JSON database for å teste arkitekturen i systemet og for relevant innhold når vi utvikler sidene.

GitHub
For kodeversjonering bruker vi github. Det tillater oss å jobbe mer effektivt som en enhet, ved å organisere issues og tickets, håndtere pull requests og kjøre automatiske tester, gjennom github actions.

1.6.2 Maskinvare

Datamaskiner
Maskiner som tillater utvikling på alle plattformer. Minst en Mac for utvikling på iOS og eventuelt andre Apple-plattformer.

Testing-enheter
Smarttelefoner, nettbrett, og PC for å teste programvaren på ulike plattformer. Spesielt viktig for å lage universell webkameraløsning

1.6.3 Personnel

Utviklingsteam
For å utvikle produktet vil det være nødvendig med et utviklingsteam bestående hovedsakelig av fullstack–utviklere som kan bygge funksjonalitet på tvers av systemet. Applikasjonen har ingen store tekniske krav i oppstartsfasen; hovedfokuset vil være å bygge et system som kan testes av brukerne. I tillegg vil det være behov for frontendutviklere for å sikre en god og intuitiv brukeropplevelse.

1.6.4 Budsjett og tidsressurser

Lisenskostnader
Snowflake lisens: 3 dollar/kreditt [6]

Tidsestimat
22,5 t per utvikler per uke. Dette blir totalt 1012.5 timer fordelt på 15 uker med 3 utviklere.

2.0 Metrikker

2.1 Minimum business metric ytelse

For at vi kan erklære prosjektet som en suksess, har vi identifisert minimumskrav fra ulike "business metrics". Disse metrikkene gir oss en konkret målestokk for verdi og effektivitet.Vi har valgt ut viktige metrikker for å måle løsningens verdi, og satt mål som er viktig for å kartlegge løsningens verdi.

Brukertilfredshet
Minimumskravet for brukertilfredshet er viktig for å sikre verdi for sluttbrukere. Dette skal vi måle med spørreundersøkelser og tilbakemeldinger. Her setter vi en minimums gjennomsnittlig brukertilferdshetsscore på minst 80% på skjemaer som evaluerer hvor nyttig, intuitiv og pålitelig løsningen er

Presisjonsrate på Håndgjenkjenning
Presisjonen for håndgjenkjenning er viktig for å levere en god brukeropplevelse, da feiltolkede gester kan redusere brukeropplevelsen betydelig. Minimum 90% presisjon under normale lys- og kameraforhold.

Systemstabilitet
Applikasjonens stabilitet påvirker hvorvidt brukerne kan stole på løsningen. Høy stabilitet er avgjørende for å unngå frustrasjon og sikre god brukeropplevelse. Mindre enn én krasj per 100 brukerminutt er vårt mål for å oppnå suksess under denne metrikken.

Brukervekst
Løsningen har et kommersielt og brukerdrevet formål. Det er derfor viktig med jevn vekst i antall brukere. Det vil vise at flere ser verdi i løsningen og at den har mulighet for utvidelse. Vi har satt en vekst på minst 5% nye brukere per måned etter lansering som mål for suksess.

Retensjonsrate
Hvor mange brukere fortsetter å bruke løsningen over tid. Lav retensjonsrate kan indikere at brukere ikke finner vedvarende verdi i løsningen. Vi har satt minimumsmål på 50% retensjon over 3 måneder etter førstegangsbruk.

3.0 Data

3.1 MediaPipe

Vi bruker Google MediaPipe sin forhåndstrente håndmodell som identifiserer og sporer handlandmarks.[7] Den tar inn bildedata enten statisk eller fortløpende, og gir ut 21 landmarks i bilde- og virkelighetskoordinater. I [7] står det at denne modellen er trent på over 30 000 virkelige bilder, i tillegg til flere syntetiske hender med forskjellige bakgrunner.

3.2 Hand Estimation klasse

I programmet er det utviklet en handestimation basert på en mediumartikel av A. Guatam [8]. Artikkelen beskriver trinnvis hvordan en kan sette opp mediapipe sammen med OpenCV for å analysere videodata i sanntid. Den tillater oss å finne hvilke fingre som er oppe, som legger grunnlaget for funksjonaliteten i applikasjonen vår. Informasjonen bruker vi til å registrere ulike gester som da kan knyttes til kommandoer i applikasjonen.

3.3 Personvern

Lokal databehandling
For at løsningen skal fungere krever det tilgang til webkamera til brukeren. Dette skal kun skje lokalt og ingen videostrømming skal lagres på servere.

Ingen datalagring
Applikasjonen lagrer ikke video eller bilder. Dataene brukes kun i sanntid og slettes med en gang.

Sikker tilgang
Applikasjonen spør om tilgang til webkameraet på enheten, og dette kan når som helst trekkes tilbake av brukeren.

4.0 Implementasjon

4.1 Gestfunksjonalitet

For å lage livegjenkjenning var vi avhengig av å sette opp en loop som kjører når klassens attributt video_running er sann. Inne i denne løkka defineres en frame med OpenCV og bruker findPosition metoden fra hand_estimation klassen for å tegne opp hånda, i tillegg brukes find fingers for å registrere hvilke fingre som er oppe (==1) eller nede(==0).

Visst programmet registrerer at det er en eller flere fingre oppe vil den gå inn i en if-løkke. Denne løkka leter etter gester. Disse gestene er definert ved å angi for eksempel:

```
if fingers[1] == 1 and sum(fingers) == 1:
```

Denne linjen vurderer om pekefingeren er oppe, og at summen av fingrene som er oppe er lik null. Det betyr at en annen finger, som tommelen er oppe (fingers[0]==1), eller det er flere fingre oppe samtidig, vil den hoppe forbi. Vi satt pekefinger oppe til å inkrementere oppskriftssteg, og tommel opp til å hoppe tilbake.

Et problem som ble møtt var at estimeringen var svært sensitiv. Dette medførte at systemet plutselig kunne plukke opp at pekefinger og tommel var oppe, uten at brukerne mente det. Systemet inkrementerte også veldig raskt gjennom stegene dersom samme finger ble holdt oppe. For å løse dette ble det implementert en timer som krever at en holder en gest i 0.3 sekunder sammenhengende før if-løkken aktiveres. I tillegg ble siste aktive gest lagret, noe som førte til at brukeren måtte ta ned og opp fingeren for hvert steg som skulle navigeres frem eller tilbake.

Implementasjon hentet fra recipe_page.py i prosjektet [9]:

```
if len(lmsList) != 0:
    fingers = self.detector.findFingerUp()

    if fingers[1] == 1 and sum(fingers) == 1:
        # Start timing if this is a new gesture or continue if same
        if current_gesture != "next":
            gesture_start_time = time.time()
            current_gesture = "next"
        elif gesture_start_time and time.time() - gesture_start_time >= 0.3:  # Held for 0.3 second
            if self.current_step < len(self.recipe.steps) - 1:
                self.current_step += 1
                self.display_current_step()
            gesture_start_time = None  # Reset after action

    # Check if thumb is held up (for previous step)
    elif fingers[0] == 1 and sum(fingers) == 1:
        # Start timing if this is a new gesture or continue if same
        if current_gesture != "previous":
            gesture_start_time = time.time()
            current_gesture = "previous"
        elif gesture_start_time and time.time() - gesture_start_time >= 0.3:  # Held for 0.3 second
            if self.current_step > 0:
                self.current_step -= 1
                self.display_current_step()
            gesture_start_time = None  # Reset after action
    else:
        # Reset if no recognized gesture is held
        gesture_start_time = None
        current_gesture = None
```

4.2 Streamlit

Som nevnt i seksjon 1.6.1 Programvareverktøy ble rammeverket Streamlit benyttet for utvikling av brukergrensesnittet samt distribusjon av applikasjonen. For å gi grensesnittet et unikt og tilpasset preg, ble en kombinasjon av flerlinjestrenger (HEREDOC) og `st.markdown()` brukt for effektivt å integrere og presentere HTML-innhold. For eksempel ble HEREDOC benyttet til å skrive flerlinjet HTML-kode, som deretter ble rendret i applikasjonen ved hjelp av `st.markdown()`. Denne metoden forbedret lesbarheten i koden og forenklet håndteringen av komplekse tekstblokker.

For å organisere innholdet og gi sidene en enkel oversikt, ble Streamlits funksjoner for markdown og kolonner tatt i bruk. Ved hjelp av `st.markdown()` kunne teksten formateres med ønsket stil, inkludert overskrifter, lister og fremhevet tekst, noe som bidro til en klar og strukturert presentasjon av informasjonen. Videre ble st.columnd() benyttet for å dele siden inn i flere kolonner, slik at tekst og andre elementer kunne plasseres nøyaktig der det var ønskelig. Dette ga fleksibilitet til å arrangere innholdet på en brukervennlig måte og gjorde navigasjonen mer intuitiv.

I tillegg tilbyr Streamlit muligheten til å legge til egendefinerte temaer gjennom konfigurasjonsfilen `streamlit/config.toml`. Ved å tilpasse denne filen kunne applikasjonen få et utseende som samsvarte med prosjektets profil, og brukerne fikk muligheten til å justere temaet etter egne preferanser. Dette forbedret brukeropplevelsen og økte applikasjonens fleksibilitet.

Bruken av disse verktøyene i Streamlit bidro til rask utvikling av en applikasjon som kunne brukes til brukertesting i en tidlig fase av prosjektet.

5.0 Deployment

Å deploye en Streamlit-applikasjon er enkelt. Først opprettes en bruker, og deretter trykker du på *Deploy*-knappen øverst til høyre når applikasjonen kjører i nettleseren. Da kommer du til noen felt som må fylles ut:

1. Lim inn GitHub-URL-en til prosjektet du vil deploye.
2. Velg hvilken branch som skal brukes.
3. Angi hvilken fil som skal kjøres (i vårt tilfelle er det app.py).
4. Til slutt kan du bestemme URL-en til applikasjonen, som for eksempel kan være HandyRecipes.streamlit.app.

Under deployment ble det oppdaget et problem: Applikasjonen klarer ikke å finne webkameraet til brukeren. Dette skjer fordi Streamlit-applikasjonen kjører i skyen og dermed prøver å finne et kamera på serveren i stedet for på brukerens egen maskin. For å løse dette må vi finne et

alternativ til OpenCV når applikasjonen skal ut i produksjon. Derfor viser vi bare et eksempel på applikasjonen som kjører lokalt på vår maskin.

Referanser

[1] S. Pal mfl., «Mobile phones: Reservoirs for the transmission of nosocomial pathogens», Adv Biomed Res, bd. 4, nr. 1, s. 144, 2015, doi: 10.4103/2277-9175.161553.

[2] Yummly. «Yummly: Personalized Recipe Recommendations and Search.» Nettside. Hentet fra: https://www.yummly.com/mobile (Lastet ned: 8. november 2024).

[3] Tudorspan Limited. «Recipe Keeper App for iPhone, iPad, Android, Windows and Mac.» Nettside. Hentet fra: https://recipekeeperonline.com/ (Lastet ned: 8. november 2024).

[4] VG. «Godt.no – Oppskrifter og inspirasjon.» Nettside. Hentet fra: https://www.godt.no/ (Lastet ned: 8. november 2024).

[5] I. Whitestone. «Calculating cost per query in Snowflake.» SELECT.dev. Hentet fra: https://select.dev/posts/cost-per-query (Lastet ned: 8. november 2024).

[6] E. Ries. The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. New York, NY, USA: Crown Business, 2011.

[7] Google Edge AI. «Hand landmarks detection guide.» Nettside. Hentet fra: https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker (Lastet ned: 11. november 2024).

[8] A. Gautam, "Hand Detection Tracking in Python using OpenCV and MediaPipe," Medium, 16. mai 2024. Hentet fra : https://gautamaditee.medium.com/hand-recognition-using-opencv-a7b109941c88. (Lastet ned: 11. november 2024).

[9] J.Myren, “HandyRecipes” Github Nettside. Hentet: https://github.com/swampbear/HandyRecipes/edit/main/pages/recipe_page.py . (Lastet ned: 13. november 2024).