Refleksjonsrapport for Hædda Project Simulator

Innledning

I dette prosjektet har vi utviklet en webbasert applikasjon som skal støtte studenter i faget *Prosjektledelse 2*. Løsningen er en dialogdrevet forhandlingssimulator knyttet til byggingen av en ny bareneskole, Hædda barneskole. Studenten skal fungere som "prosjektleder" og kunne føre samtaler med flere KI-agenter som representerer sentrale aktører i et reelt prosjektscenario. KI agentene skal representere for eksempel totalentreprenør, arkitekt og VVS-ingeniør.

Gjennom samtale med disse agentene skal brukeren kunne utforske problemstillinger, forhandle og forme en prosjektplan som automatisk oppdaterer både *Precedence diagram* og *Gantt plan*. Målet er å skape et læringsverktøy som gjør det mulig for studenter å øve på prosjektledelse ved å simulere kommunikasjon med ekte interessenter, og samtidig forstå hvordan dialog og forhandling påvirker planlegging, økonomiske utfordringer, avhengigheter, tidslinjer og prioriteringer.

I denne rapporten skal vi reflektere over utviklingsprosessen, bruken av KI, utfordringer vi har møtt på, og de etiske og teknologiske konsekvensene prosjektet reiser.

Utviklingsprosess

Vi er fire studenter med svært ulike faglige bakgrunner, noe som både har bidratt til mangfold i perspektiver og utfordringer i kommunikasjon. Vi har forsøkt å holde hyppige møter gjennom hele prosjektet. En utfordring her var at burde vært mer konsekvente på å planlegge og sette opp faste tidspunkter for møtene ukentlig. Vi endte ofte opp med en sen varsling opp mot statusmøtene våre som resulterte i at ikke alle hadde mulighet til å stille.

Oppgavene ble fordelt slik at tre gruppemedlemmer fokuserte på selve utviklingsarbeidet, mens jeg hadde hovedansvar for refleksjonsrapporten og dokumentasjonen. Denne strukturen gjorde det mulig å jobbe parallelt, men avdekket også behovet for tett kommunikasjon for å sikre at dokumentasjon, designvalg og implementasjon hang sammen.

Vi må være ærlige på at vi fikk god hjelp fra læreren vår til å utarbeide proposal.md – produktspesifikasjonen som definerer MVP-kravene, teknologistakken, og de overordnede

brukerflytene. Dette ga oss et solid fundament å bygge videre på, men betyr også at vi har mindre eierskap til den overordnede arkitekturen enn vi ville hatt om vi hadde utviklet den helt selv fra bunnen av. Samtidig har denne hjelpen vært verdifull for læringsprosessen. Vi har fått se hvordan en profesjonell produktspesifikasjon ser ut, og har kunnet fokusere på å forstå og implementere kravene fremfor å bruke all tid på å definere dem.

Metode og arbeidsform

KI verktøy som Gemini, Gemini CLI, Claude Code og ChatGPT ble brukt gjennom hele prosessen. Både til ideutvikling, planlegging, design, kodegenerering, feilsøking og dokumentasjon. Dette gjorde KI til en integrert del av vår arbeidsflyt.

Bruk av BMAD-metodikken

I arbeidet med prosjektet har vi fulgt BMAD-metodikken (Business
Model Assisted Development), slik den er introdusert i emnet. I stedet for å «bare» begynne å kode, tok vi først utgangspunkt i den overordnede forretningsmodellen: hvem brukerne er (LOG565-studenter), hvilken verdi simulatoren skal gi (trening i planlegging, forhandling og bruk av WBS), og hvilke rammer som gjelder (budsjett, tidsfrist og vurderingsform).

Dette ser vi igjen i repository strukturen. Brainstorming dokumentene og workflow-/statusfilene våre viser hvordan vi først jobbet bredt med idéer og mulige brukerreiser, før vi snevret inn til konkrete funksjoner og en tydelig definert MVP. Project-plan, proposal.md og de ulike «concept»/«workflow»-filene henger sammen som spor av denne prosessen. BMAD har dermed fungert som en rød tråd fra de første diskusjonene om behov og læringsmål, via kravspesifikasjon, til de tekniske beslutningene vi tok underveis.

Teknisk løsning og verktøy

Prosjektet krever en moderne og bred teknologistakk, og vi tok i bruk både tradisjonelle utviklingsverktøy og nyere KI baserte løsninger.

Tradisjonelle programmer og utviklingsverktøy

Vi arbeidet med følgende verktøy:

- **Operativsystem:** Windows
- **Kommandolinje:** CMD, Powershell, Terminal
- **Kodeeditor:** VSCode med relevante utvidelser (Windsurf, Cursor ol)
- **Versjonskontroll:** Git og GitHub (repo, branches, pull requests)
- **Programmeringsspråk:** Node.js og Python
- **Infrastruktur:** Docker Containers
- **Kontinuerlig integrasjon:** GitHub Actions
- **Publisering:** Vercel (tett integrasjon med Next.js)

Disse verktøyene utgjorde kjernen i utviklingsmiljøet og representerte en betydelig del av læringskurven i prosjektet.

Planlagt teknologistakk *(fra proposal.md)*

Komponent	Teknologi
Frontend	Next.js (App Router), TypeScript, Tailwind CSS, shadcn/ui, Zustand
Backend	FastAPI (Python) – valgt for AI/LLM-økosystem
Database	Supabase (PostgreSQL) – Auth, persistens, real-time
AI-tjeneste	Gemini 2.5 Pro/Flash – norskspråklige agenter
Hosting	Vercel (frontend + backend)

KI verktøy

I tillegg brukte vi en rekke KI programmer som:

- Nettbaserte KI-tjenester: Gemini, Claude Pro og ChatGPT

- Kodeagenter: Gemini CLI og Claude Code

KI ble brukt til:

- generering av kode
- debugging
- planlegging av systemarkitektur
- utvikling av prompts og agentlogikk
- casetekst og dokumentasjon
- designforslag
- problemløsing
- Deploy

Vært å merke seg. KI var for oss ikke et tillegg, det var et sentralt arbeidsverktøy.

Kompleksitet i verktøylandskapet

En av de største utfordringene var å forstå når og hvordan hvert verktøy skulle brukes. Kombinasjonen av tradisjonelle verktøy (Git, Docker, Node, Next.js) og KI-verktøy (Gemini, CLI, Claude Pro/Code) gjorde arbeidsflyten kompleks og skapte usikkerhet i starten.

Dette utfordret spesielt:

- skillet mellom frontend og backend
- riktig bruk av Gemini CLI / Claude Code i utviklingsprosessen
- forståelse av Next.js sin filstruktur
- oppsett av GitHub Actions og Vercel
- samspill mellom KI og manuelt skrevet kode

Likevel ga dette oss god innsikt i hvordan moderne utviklingsprosesser faktisk fungerer i praksis.

Utfordringer og løsninger

Selv om prosjektet fortsatt er under utvikling, har vi møtt flere sentrale utfordringer:

Forståelse av alle verktøyene

Å jobbe med så mange ulike teknologier samtidig gjorde det krevende å vite hva hvert verktøy egentlig skulle brukes til. Det oppstod ofte forvirring mellom:

- Backend vs Frontend
- Docker vs lokal kjøring
- GitHub Actions vs Vercel deploy
- manuell kode vs KI generert kode

KI bidro med forklaringer, men vi måtte likevel bruke tid på å forstå konseptene selv.

Lage en god case beskrivelse

Agentene er avhengig av et godt faktagrunnlag. Å skrive en konsistent og faglig troverdig case beskrivelse tok mye tid. Dette inkluderte:

- rollene til totalentreprenør, arkitekt og VVS-ingeniør
- prosjektmål
- rammer, krav, forventninger
- læringsmål
- milepæler og planstruktur

Uten dette ville agentene gitt irrelevante eller feilaktige svar.

Visualisere sluttproduktet

Å forstå hvordan dialog med agentene skulle generere endringer i Precedence og Gantt ble vanskelig i starten. Vi måtte jobbe mye med å forestille oss konkrete brukerreiser og visuelle komponenter før utviklingen kunne starte.

Variasjon i forkunnskaper

Ulike bakgrunner gjorde samarbeidet mer krevende. Vi brukte tid på å sikre at alle hadde nødvendig forståelse av både teknologien og prosjektets mål.

Kritisk vurdering av KI sin rolle

Positive effekter

- Raskere prototyping
- Kodegenerering som sparte tid
- Automatiske forklaringer på komplekse konsepter
- Bidro til bedre dokumentasjon
- God sparringspartner for idéutvikling

Utfordringer og risiko

- KI genererte kode som ofte måtte omskrives
- Hallusinasjoner, for eksempel fant KI på fakta
- Risiko for overavhengighet
- Agentene blandet roller uten strenge systemprompter
- Vanskelig å validere svar uten solid fagforståelse

KI sin påvirkning på sluttproduktet

KI akselererte utviklingen, men sluttkvaliteten avhenger helt av menneskelig vurdering, fagkunnskap og kvalitetssikring.

Etiske og teknologiske implikasjoner

Etiske hensyn

- Agentene må ikke fremstå som absolutte fasiter
- Risiko for feilinformasjon i studentverktøy
- Bruk av tidligere “star answers” krever anonymitet på elever
- Transparens om at agentene er KI og ikke eksperter

Teknologiske hensyn

- Precedence grafen må være en gyldig DAG

- Gantt planen må alltid være konsistent
- Feilhåndtering for KI svar må bygges inn
- Systemet trenger sporbarhet (historikk)

Konklusjon

Prosjektet har gitt oss innsikt i hvordan KI kan integreres i både utvikling og læring. Selv om teknologien skapte utfordringer, ga den oss også nye muligheter for effektivitet og kreativitet. Arbeidet har vært krevende, men har bidratt til å utvikle våre ferdigheter innen både programmering, KI forståelse og tverrfaglig samarbeid.

På bakgrunn av at vi fremdeles er i en utviklingsfase, har vi måtte gjøre noen prediksjoner opp mot problemer vi ser for oss å møte på. Blant annet hallusinasjoner og agenter som blander roller er typiske eksempler på predikasjoner vi har gjort i denne refleksjonsrapporten.