TDT4140 - Programvareutvikling
Leveranse 8

Antall ord utenom forside og referanser: 1500 + 1958

Gruppenummer: 63

Produktnavn: Exercise it!

Medlemmer som har bidratt til leveransen:

Fornavn	**Etternavn**	**Studmail**
Thomas	Karud	thomabk@stud.ntnu.no
Kristoffer	Nyvoll	kristnyv@stud.ntnu.no
Vemund	Eggemoen	vemundeg@stud.ntnu.no
Hjalti	Hjaltason	hphjalta@stud.ntnu.no
Erlend	Fredborg	epfredbo@stud.ntnu.no
Gard	Drag-Erlandsen	gardd@stud.ntnu.no
Khadija	Laajab	khadijal@stud.ntnu.no

Gruppereflekskjon

Innledning

Gjennom utviklingen av treningstjenesten “Exercise It” har gruppen tilegnet seg flere erfaringer med å jobbe i team og utvikle en tjeneste i samarbeid med en kunde. Fagets overordnede læringsmål vil i denne oppgaven bli diskutert punktvis ut fra våre erfaringer.

Gruppedynamikk og psykologi (PRF.psy)

Gruppen har erfart ulike aspekter av smidig programvareutvikling. Flat gruppestruktur og uformelle relasjoner medførte få kommunikasjonsledd og høy interaksjon mellom gruppemedlemmene. Strukturvalget lærte gruppen at tett kommunikasjon fasiliterer involvering. Dette underbygges av Sommerville som hevder god kommunikasjon er essensielt i teamarbeid.[1] Dog krever flat struktur selvstendige og strukturerte gruppemedlemmer. Dette erfarte vi gjennom forsentkomminger og uferdig arbeid. Sanksjoner som kakekryss kan hindre dette ved å nedfelles i gruppekontrakten. Flat struktur fasiliterte også idéutveksling om produktets funksjonalitet. Eksempelvis var uerfarne programmerere mer bidragsytende på funksjonalitetsidéer kontra implementasjon.

Gruppens medlemmer går forskjellige studieprogram og har ulikt erfaringsgrunnlag. I retrospekt er vi enige med Sommerville at mangfold støtter effektiv kommunikasjon og dynamikk.[1] Dette viste seg da medlemmene tok ansvar og bidro til fremdrift på sine fagfelt. Følgelig bidro mangfoldet til skjevfordelt arbeidsmengde. Skjevfordelingen ble innført med intensjon om å øke effektiviteten, men enkelte gruppemedlemmer mente skjevfordelingen var urettferdig og påvirket arbeidsmoralen. En løsning, og punkt til forbedring, er å skape konsensus rundt strategien ved å nedfelle den i gruppekontrakten.[2] I senere prosjekter blir da gruppemedlemmene forberedt på at skjevfordeling av arbeidsmengde kan forekomme, og forståelsen for tiltaket styrkes.

Kommunikasjon spesifikt til programvareutvikling (PRF.com)

Produkteier hadde mindre tid til møter enn hva Kniberg anbefalte.[3] Dette løste vi ved å inkludere henne i gruppens slack. Produkteier var dermed mer tilgjengelig, og bidro tett på reevaluering av brukerhistorienes prioriteringer underveis. Gruppen erfarte at nær kundekontakt øker arbeidets effektivitet og styrker beredskapen for uforutsette situasjoner da det ikke ble behov for nye rutiner etter skolen stengte. Det gjorde også utviklingen smidigere, siden vi kunne gjøre fortløpende vurderinger ved behov.

I henhold til Kniberg kan grupper lide av for lite kontakt med produkteier.[3] Gruppen erfarte dette, da en misforståelse om tjenesten skulle poste enkeltøvelser eller økter oppstod. Situasjonen tydeliggjorde viktigheten av korrekt og tydelig informasjon om brukerhistoriene

[1] Sommerville, s. 658, Teamwork
[2] Arbeid i team, s. 108, Teamkontrakt som hjelpeverktøy
[3] Kniberg, s. 71, Keep the product owner at bay

og produktkrav, særlig da produkteier var mindre fysisk tilstedeværende enn ønskelig. I senere prosjekter vil vi etterstrebe bedre kjennskap til, og bruk av, akseptansetester[4] for å kunne teste kvalitet og hindre misforståelser.

Pensum ga mindre føring på den daglige kommunikasjonen med produkteier. Vi måtte selv finne passende formalitetsnivå. Dette erfarte vi ved at kommunikasjonen begynte formelt på mail, til å senere ha vennligere og avslappet tone på slack. Gruppen lærte at den avslappede tonen var en fordel da det bygget opp under den flate strukturen, få kommunikasjonsledd og dermed effektiv samhandling.

Programvareutvikling som profesjon(PRF.pr)

Under utviklingen av “Exercise It!” erfarte gruppen å måtte ta stilling til mengden personinformasjon som innhentes. I tråd med GDPR gikk vi gjennom hva vi skulle bruke personinformasjonen til, som videre ga grunnlaget for å kun be brukeren om epost. Målet var å minimere brukerdata uten å påvirke produktets funksjonalitet.

Gruppen vurderte formålet med tjenesten, og opplevde den, gitt sitt fokus på helse, som positiv for samfunnet. Til gjengjeld kan tjenester med fokus på kropp og trening medføre skadelig kroppspress. I tråd med Sommerville, lærte gruppen at man som programvareutvikler kommer overfor etiske dilemmaer hvor man som fagperson har ansvar for å opptre forsvarlig.[5]

Prosesskonsepter og implementasjon (PRO.con/PRO.imp)

Scrum og XP er nyttige verktøy i et gruppeprosjekt. Ulike timeplaner gjorde at gruppen ikke kunne møtes hver dag. Det førte til at daily meetings med fysisk oppmøte, slik Kniberg anbefaler, var vanskelig å gjennomføre. Alternativet ble GitLab som scrum-board og slack til daily scrum. Selv om slack sentraliserte kommunikasjonen følte gruppen likevel at slackbaserte daily meetings ga mindre utbytte sammenlignet med fysisk oppmøte. Det ble vanskeligere å formulere spørsmål og kritikk som medførte mangel på diskusjon. Gruppen reflekterte over at tidligere bruk av videomøter kan fasilitere diskusjoner ved at samtalen flyter lettere.

Gruppen ble godt kjent med parprogrammering[6] i Sprint 2. Ulempen var økt tidsbruk, men økt veiledning var en fin måte å igangsette gruppemedlemmene med mindre programmeringserfaring. Gruppen erfarte økt tidsbruk som en investering som ga avkastning senere i form av mindre tid brukt på veiledning. Da skolene stengte var det en fordel at alle fikk god veiledning under parprogrammeringen og kunne jobbe selvstendig. I fremtiden vil dypere læring om effektiv parprogrammering kunne forbedre utviklingsmiljøet.

Etter releaseplanen var ferdigstilt fikk gruppen et nytt medlem. Det nye medlemmet måtte forholde seg til en gruppekontrakt hun ikke hadde vært med å utforme. Gruppen løste dette

[4] Sommerville, s. 250, Acceptance test
[5] Sommerville, s. 28, Software engineering ethics
[6] Kniberg, s. 104, Pair programming

ved at vedkommende kom med endringsforslag som gruppen stemte over. Dette sparte tid samtidig som alle fikk ta del i utformingen.

Prosjektplanlegging (PRO.pp)

Faglitteraturen har fasilitert en effektiv utviklingsprosess. Sprinter og retrospektiv ga gruppen kontrollerte rammer for hvordan prosjektet skulle utføres og tiden kunne brukes til utførelse av brukerhistorier fremfor å organisere arbeidsrammene. Grundig egenevaluering og gjennomførbare forbedringstiltak tilknyttet retrospektivmøtene forbedret gruppens innsats og fremgangsmåte underveis. Et konkret eksempel var innføringen av parprogrammering etter Retrospektiv 1. Tiltaket økte integrering av medlemmene og førte til en lærerik start på Sprint 2.

Til første demonstrasjon klarte ikke gruppen å følge releaseplanen. Dette skjedde hovedsakelig fordi gruppen ikke forstod omfanget av alle brukerhistoriene ved utforming, og dermed bommet på tidsestimeringen. Dette lærte gruppen at scrums rammer bidrar til ekstraarbeid dersom de brukes unøyaktig. Planleggingspoker[7] kan i fremtiden benyttes for å begrense tidsestimeringsavvik. Metoden fremmer diskusjon rundt tidsbruk på brukerhistoriene, og dermed et bedre vurderingsgrunnlag for tidsestimeringen. Som et resultat av avvikene måtte brukerhistoriene omorganiseres i den neste sprintplanen. Dette ga gruppen erfaring med prioriteringer og endringer via iterativ arbeidsmetode.

Versjonshåndtering (PRO.cm)

Gruppen benyttet Gitlab, som effektiviserte arbeidet gjennom prosjektet. Vi brukte Boards-funksjonen til å tydeliggjøre statusen til issues. Gruppen erfarte at utvikling i branches, kombinert med hyppige commits, gjorde det lettere å avdekke og korrigere feil. Vi fikk godt utbytte av å gjennomføre Code Review ved hver Merge-Request, både for kunnskapsoverføring og kvalitetssikring. Vi fastsatt handlemåten for versjonskontroll i wikien[8] vår. Behovet for dette var tydelig, da ulike konvensjoner førte til feil og misforståelser.

Det oppsto problemer mot slutten av Sprint 2 da vi forplantet og pushet ulike endringer i lokale databaser. Vi løste problemet ved å wipe databasen og resette migrations, men dette kunne vi løst ved bedre konfigurasjon av .gitignore[9] og dermed hensiktsmessig tracking av filer og endringer i GitLab.

Evolusjonsprosess (PRO.evo)

Produktet er utviklet med fokus på kundens ønsker. En tydelig beskrivelse av installasjonsprosessen og detaljert dokumentasjon vil minimere sannsynligheten for utfordringer ved overtagelse, som videre bidrar til at ny funksjonalitet kan prioriteres. Videre har gruppen skrevet anbefalinger for videre utvikling i Gitlab-Wiki'en[10], som gir en pekepinn på naturlig fortsettelse av prosjektet. En utfordring med produktet er mangelfull testing av profesjonelle brukere. Dette kan medføre "broken code" som blir et problem for senere

[7] Kniberg, s. 38, How we do sprint planning
[8] Workflow, https://gitlab.stud.idi.ntnu.no/tdt4140-2020/63/-/wikis/Workflow
[9] Gitignore, versjon 2.26.1, https://git-scm.com/docs/gitignore
[10] Gruppe 63, Wiki, https://gitlab.stud.idi.ntnu.no/tdt4140-2020/63/-/wikis/Product-Roadmap

utviklere. For å unngå forvirring ved overtakelse, vil tett oppfølging bidra til å kunne løse problemer og gi innføring i systemet.

På grunn av prioriteringene om funksjonelle krav og at utviklingsperioden var kort, satte vi ikke av egen tid til refaktorering[11]. Runder med refaktorering kunne avklart svakheter, men gruppen vurderte at tilfredsstillende kodekvalitet kan oppnås ved normalt bruk av parprogrammering og Code Reviews. Metodene har ikke medført komplikasjoner, men det kan skape problemer for fremtidige grupper dersom kodebasen viser seg å ha uforutsette kvalitetsusikkerheter.[12] Videre vil det faktum at prosjektet baseres på Djangos velkjente arkitektur og gode dokumentasjon bidra til at overføringen kan få færre komplikasjoner.

Testing, verifisering og validitet (VAV.fnd/VAV.tst)

Under prosjektarbeidet har gruppen aktivt testet produktet, og gjennomført akseptansetesting med produkteier. Vi har kontinuerlig testet og verifisert koden ved hjelp av GitLabs Pipeline. Dette testet koden hver gang vi pushet eller merget, og garanterte at koden i master til enhver tid var fungerende.[13] Det tok derimot lang tid før gruppen begynte å skrive egne tester. Dette valget begrunnes med at sofistikert testskriving, i likhet med testdrevet utvikling, stiller høyere krav til kompetanse, og gruppen håper på å tilegne seg kjennskap til TDD i senere prosjekter. Mangelfull forkunnskap gjorde det utfordrende å implementere testskriving fra starten av, og vi skrev heller konsise enkelttester i etterkant som konsekvens. Nå som prosjektet er ferdig, har gruppen sterkere testskrivingsferdigheter som kan utnyttes i senere prosjekter.

COVID-19

Læringsmålene ble i ulik grad påvirket av korona-pandemien. Læring rundt bruk av Scrum-prosessen ble svekket da Demo 2 ikke ble gjennomført. Det medførte at produkteier fikk redusert adgang til produktet og svekket tilbakemeldingsgrunnlaget. For å mestre læringsmålet kunne gruppen arrangert et videomøte med gjennomgang via skjermdeling for produkteier. Fordelen med en slik gjennomgang er at den er mindre omstendelig, kan arrangeres hyppigere og føre til tettere kontakt med produkteier.

Covid-19-situasjonen gjorde at gruppen ikke fikk full utnyttelse av Scrum-prosessen da fysiske scrum-møter ble vanskelig å opprettholde. Kniberg er tydelig på at man bør etterstrebe fysisk interaksjon mellom gruppemedlemmene for å sikre aspektet “team gel”.[14] Flere fysiske møter ville ført til bedre “group flow”[14] og høyere produktivitet. For å sikre “team gel” i fremtiden kan videosamtaler med et sosialt fokus, eksempelvis “zoom-pils”, styrke relasjonene og gruppedynamikken.

[11] Kniberg, s. 47, Tech stories
[12] Kniberg, s. 18, Why quality is not negotiable
[13] Sommerville, s. 735, version management
[14] Kniberg, s. 139, Rearrange teams between sprints -or not?

Individuell refleksjon

Erlend

Min rolle har bestått av å være administrativ, hjelpe scrum-master, men også bidra som front-end utvikler. Jeg har arbeidet med appens brukervennlighet, men også bidratt med struktur og refleksjon under prosjektets tekstlige oppgaver. Som utvikler har jeg videreutviklet mine ferdigheter i Python og lært hvordan rammeverk som django forenkler utviklingsprosessen. Gjennom oppgaverefleksjon har jeg fått innblikk i hvordan kunnskapsvariasjon og personer med ulik bakgrunn kan effektivisere og kvalitetssikre et prosjekt.

Oppgaveskrivingen passet meg best. Der bidro jeg med både struktur og refleksjon. Å stille spørsmål ved gruppens tidligere valg og tydeliggjøre utfordringer har vært med på at gruppen har lært underveis. Det ble for eksempel enighet om at gruppekontrakten burde inneholdt sanksjoner mot forsentkomming. Videre har jeg lært at en utviklingsprosess er så mye mer enn koding. Oppgavene må organiseres, ferdigheter må struktureres og man må hjelpe hverandre. På dette området har scrum hjulpet til å tydeliggjøre viktige aspekter som kan være utfordrende. Bruk av scrum-teknikker som burndown-chart[15] har videre hjulpet å løse disse problemene og lært meg at bruk av kreative løsninger kan være gode alternativer.

Prosjektet har også bydd på utfordringer og kodingen er det jeg synes har vært vanskeligst. Jeg har ingen kode-erfaring utenom fag på NTNU. Dette gjorde det å starte på brukerhistorier var vanskelig. Da jeg satt fast kjente jeg at mestringsfølelsen og motivasjonen falt. Det var frustrerende å ikke klare å bidra like mye til gruppen som ønskelig. Jeg mener selv at en slik situasjon kan unngåes eller gjøres mildere ved å gi undervisning om hvordan typiske brukerhistorier løses. Mangelen på følelse av bidrag har til gjengjeld ført til større motivasjon for å bidra i oppgaveskrivingen og fordype meg i pensum.

[15] Kniberg, s.56, Estimating days VS hours

Gard

Under prosjektarbeidet har min rolle vært administrativ gjennom å fasilitere godt gruppearbeid, og samtidig hjelpe til i gruppen der det har vært størst behov. Videre har jeg i stor del bidratt i oppgaveskrivingen hvor jeg har supplert med formuleringer, refleksjon, og rettskriving. Jeg har funnet meg selv i rollen som en igangsetter, som stiller mye spørsmål og prøver å føre arbeid fremover. Rollen opplevde jeg som naturlig, og trivdes godt med. Tidlig tok jeg ansvar for å utforme møtereferat, booke rom, skaffe oversikt over timeplaner og inkludere medlemmer i gruppediskusjoner. I starten av prosjektet tok jeg også del i front-end utviklingen av prosjektet. Jeg opplevde prosessen å være delaktig i utviklingen av et produkt i samarbeid med mer erfarne utviklere å være veldig lærerikt, men samtidig utfordrende. Det ga meg et perspektiv på hvordan man jobber sammen og går frem i løsning av sammensatte kodeprosjekter.

Gjennom min rolle som gruppemedlem har jeg lært at Scrum prosessen består av mye mer enn kun koding. Jeg har i praksis sett hvordan sprint-planlegging, retrospektivmøter, daily-scrum og tydelig rollefordeling bidrar til en mer effektiv utviklingsprosess dersom tilpasset på riktig måte. Lærdommen jeg tar med meg fra denne prosessen ser jeg på som utrolig verdifull med tanke på fremtidige gruppesamarbeid.

Mitt kompetansenivå påvirket gruppen i form av at jeg brukte mye tid på å lære meg både Scrum og Django. Jeg endte opp med å hindre medlemmer i å jobbe videre med prosjektet ettersom jeg ofte spurte dem om hjelp. I starten av prosjektet opplevde jeg dette som utfordrende da jeg ikke følte jeg fikk bidratt like mye som ønsket. Med mye erfaring fra gruppearbeid gjennom verv, var det derimot naturlig at jeg tok en større del i administrative oppgavene og oppgaveskrivingen.

Khadija

Min rolle gjennom prosjektarbeidet har vært å hjelpe til med å utvikle produktet. Jeg har også vært med på å sikre gode refleksjoner i henhold til pensum. Som følge av at jeg ikke hadde så mye erfaring i programmering innenfor webutvikling gikk mye tid på starten i å komme i gang. Gruppemedlemmene som hadde mye erfaring gikk grundig gjennom Git/Gitlab. Dette førte til at jeg ble tryggere på verktøyet og kunne bidra ytterligere.

Selv om rollen var litt krevende til tider, trivdes jeg med nye utfordringer. Det å se endringer i funksjonalitet på nettsiden ga meg mestringsfølelse. Tidligere hadde jeg heller ikke erfaring innenfor det å jobbe smidig i team. Dette gjorde at jeg erfarte hvordan det er å jobbe sammen, men også individuelt på ulike brukerhistorier. Da jeg ble med i gruppen var gruppekontrakten og releaseplan allerede ferdig utarbeidet. Disse ble redebattert slik at jeg fikk være med på å endre det jeg var uenig i. Å hoppe inn når teamet var godt i gang førte til at startfasen ble tidkrevende.

Gjennom prosjektet erfarte jeg en del problemer underveis med kodingen hvor andre gruppemedlemmer med mer erfaring var til stor hjelp. Samtidig var gode IT-supporter som gurutjenesten til hjelp når feilmeldinger var vanskelig å tolke. Da skolen stengte og det ikke var mulig å møtes ble det vanskeligere å få hjelp fra de andre i gruppen. Dette førte til at jeg ikke klarte å ferdigstille brukerhistorien jeg arbeidet med i Sprint 2.

Vemund

Dette var ikke mitt første programmeringsprosjekt, noe som førte til at jeg var en av de som tok ansvar på bakgrunn av tidligere erfaring. Dette var en av grunnene til at jeg ble valgt som Scrum-master. Under oppstartsfasen holdt de av oss som kunne Git og Gitlab kurs for resten av gruppa. I tillegg tok jeg ansvar for å finne eksterne program og utvidelser til VSCode for å forenkle startfasen for gruppemedlemmer med mindre erfaring.

Som Scrum-master ble det naturlig å ta initiativ til daily-meetings, programmeringsdelen og til å hjelpe gruppemedlemmer. Med mye ansvar ble det vanskelig å være en oppfølgende Scrum-master, og i sprint 2 bestemte gruppen å bytte scrum-master i håp om en mer tilstedeværende leder. Det var en positiv endring der jeg følte jeg fikk arbeidet bedre med produktet og kunne hjelpe andre.

Rapportskriving er ikke min sterkeste side der det har vært utfordrende å reflektere over læringsmål. samtidig som å knytte opp mot pensum. Det har det resultert i at jeg har tatt mer ansvar over programmeringsdelen og vært litt i bakgrunnen under skrive delen. I fremtiden kan jeg bli flinkere til å stå på egne ben. Jeg har lært mye av de andre på gruppa og tar med meg det videre.

Som utvikler har jeg hatt ansvar for mye funksjonalitet, administrativt på Gitlab og testing. Jeg har lært et nytt rammeverk; Django, og forstått oppkobling mot database og hvordan dette påvirker hverandres kode med migrations. I tillegg har jeg lært å skrive gode tester noe jeg ikke hadde gode kunnskaper om tidligere. Til slutt har jeg erfart hvordan det er å jobbe i et team der alle ikke har like mye erfaring og satt pris på forskjellige egenskaper og evner som fører til et bra team med ulike oppgaver.

Kristoffer

I likhet med Vemund hadde jeg erfaring fra foregående programmeringsprosjekter, hvor jeg lærte veldig mye om det tekniske som tilrettelegger for utviklingsprosessen. Ettersom jeg i tillegg hadde forkunnskap med smidig utvikling, tok jeg en naturlig lederrolle de første to månedene. Jeg lærte mye av ansvaret dette medførte. Tidligere programmeringsprosjekter har jeg lent meg på mer erfarne medlemmer, men denne gangen fikk jeg erfare hvordan det er å bygge et prosjekt fra bunnen av. Det var særlig utfordrende å sette opp GitLab-repoet og gjøre et veloverveid og begrunnet valg av teknologistakk basert på produkteiers ønsker.

Å skrive dokumentasjon og tilpasse prosjektet til Open Source-modellen hadde jeg aldri gjort før, men dette lærte jeg mye av da jeg arbeidet med wikien og dokumentasjonen til prosjektet. Gjennom arbeid med prosjektet har jeg utviklet god kjennskap til GitLab, hvor jeg investerte mye tid til å sørge for at vi ikke tok noen store feilvurderinger.

Tidlig i prosjektet tok jeg ansvar for å få gruppen i gang. Dette medførte sterk eierskapsfølelse, som gjorde meg lite mottakelig for endringer. Samtidig foretrekker jeg at det jobbes i høyt tempo, som ofte resulterer i at jeg tar styringen hvis jeg føler ting går tregt. Dette kan ha redusert involveringen til andre gruppemedlemmer. I fremtidige prosjekter må jeg bli flinkere til å inkludere de rundt meg.

Aspektet ved programvareutvikling som jeg ønsket å fordype meg i, var hvordan gruppen skulle jobbe smidig sammen på tross av våre forskjellige kompetansenivå. Jeg ville også lære mer om versjonskontroll. Læringsutbyttet i dette faget har vært veldig stort for min del, ettersom de andre på gruppa har vært flinke til å stille masse spørsmål. Det er overraskende hvor læringsrikt det er å lære bort!

Thomas

Som en som kom inn i faget med lite programmeringserfaring, så jeg på prosjektet som en mulighet til å lære masse om hvordan man jobber med programvareutvikling. Motivasjonen for å ta del i kodingen var svært høy. Det var derfor viktig for meg å sette meg godt inn i hvordan de mer erfarne programmererne jobbet fra start, stille mange spørsmål, og lytte til deres tips. Dette bidro til at jeg raskt lærte å bruke Git/GitLab/GitKraken og fikk god forståelse for front-end- og back-end-koden, som etter hvert førte til at jeg fikk en viktig rolle i å utvikle funksjonalitet for produktet. Dette opplevde jeg som svært givende, og gjorde at jeg fikk stor tilhørighet til det vi jobbet med. Ved å tidlig ta del i kodingen, var det også enklere å ta større ansvar senere. På denne måten har jeg fått betydelig større kjennskap til Django-rammeverket og spesielt front-end-delen av koding gjennom prosjektet, deriblant HTML som jeg tidligere aldri har vært borti.

Da enkelte på gruppen skulle på permisjon, endte jeg opp som scrum-master til Sprint 2, ettersom jeg hadde god oversikt over hele prosjektet. Med den rollen erfarte jeg sider av å ha mye ansvar, blant annet gjennom delegering og daily-meetings. I starten syntes jeg at dette var lærerikt og interessant, men da koronaepidemien brøt ut sank motivasjonen. Dette førte til at jeg ikke greide å ta det ansvaret jeg burde ha tatt som scrum-master for å opprettholde produktiviteten i gruppen. Videomøter var her et viktig virkemiddel som burde blitt tatt i bruk raskere. Jeg synes det er viktig å ta med seg disse erfaringene videre, slik at jeg er bedre forberedt for lignende roller senere.

Hjalti

Som utvikler har jeg noe tidligere erfaring med programmering i gruppe, via andre fag her på NTNU. Men dette er den første situasjonen hvor vi har hatt et slikt fokus på å opprettholde gode rutiner med hensyn til Scrum og Extreme Programming. Dette prosjektet ble da en motivasjon for å lære meg mer om disse arbeidsmåtene og rutinene. I tillegg til at jeg kunne sette meg inn i ny teknologi under utviklingen av applikasjonen. Django var et interessant rammeverk å jobbe med siden dette er noe jeg aldri har brukt selv, men er noe linjeforeningen min har benyttet til å utvikle sine programvareløsninger tidligere.

Min rolle i prosjektet har i hovedsak handlet om å utvide produktets funksjonalitet ved å arbeide på noen av produktets mer ambisiøse brukerhistorier. Nemlig å legge inn søkefunksjonalitet på nettsiden og muligheten for brukere til å favorisere øvelser. Dette var i retrospekt et uklokt valg ettersom jeg ikke hadde benyttet django-rammeverket tidligere. Hver av disse funksjonalitetene viste seg også mer tidkrevende enn gruppen hadde estimert samt at de viste seg vanskelige å implementere med mine begrensede forkunnskaper. For å støtte meg med dette fikk jeg hjelp av Vemund til å få disse implementert. Her fikk jeg erfare hvor mektig parprogrammering som verktøy kan være og jeg lærte veldig mye i denne delen av utviklingsperioden. Parprogrammering er ikke noe jeg har vært borti tidligere, men er absolutt noe jeg kommer til å benytte i fremtidige prosjekter.

Ved siden av dette har jeg deltatt ved å arbeide med utformingen av applikasjonen og strukturering av gruppens skriftlige innleveringsoppgaver. Disse oppgavene har jeg tatt på meg ettersom disse tillot meg å benytte mine forkunnskaper fra andre prosjekter og sette mitt preg på produktet utenfor de mer tidkrevende brukerhistoriene.

Kildeliste

- Sommerville, I. (2016). Software engineering. Boston: Pearson.
- Kniberg, H., Cohn, M., & Sutherland, J. (2015). Scrum and Xp from the Trenches: how we do Scrum. C4Media.
- Levin, Rolfsen(2015), Arbeid i team - læring og utvikling i team, Fagbokhandelen
- Gruppe 63, Wiki, hentet 24.04.2020 fra https://gitlab.stud.idi.ntnu.no/tdt4140-2020/63/-/wikis/home
- Gitignore, versjon 2.26.1, hentet 24.04.2020 fra https://git-scm.com/docs/gitignore