TDT4140 - Programvareutvikling

Leveranse 3

Antall ord utenom forside og referanser

1780

Gruppenummer

22

Lenke til gruppens kodebase på GitLab

https://gitlab.stud.idi.ntnu.no/tdt4140-2022/landsby-2/gruppe_22/groupup

Produktnavn

GroupUp

Medlemmer som har bidratt til leveransen

Nummer	***Fornavn***	***Etternavn***	***Studentmail***
527892	*Line*	*Rosland*	*linehro@stud.ntnu.no*
543888	*Peter Johan*	*Flått-Bjørnstad*	*peterjf@stud.ntnu.no*
558229	*Joachim*	*Fredheim*	*joachibf@stud.ntnu.no*
527217	*Camilla*	*Kopperud*	*camillkk@stud.ntnu.no*
766739	*Thorbjørn*	*Lundin*	*tslundin@stud.ntnu.no*
543919	*Jørgen*	*Sandhaug*	*jorgeksa@stud.ntnu.no*
525194	*Martin*	*Skatvedt*	*martskat@stud.ntnu.no*

Problembeskrivelse

Produkteier har sett et ønske hos studenter ved NTNU om å kunne møtes og bli kjent med andre vennegjenger etter flere år med strenge koronarestriksjoner. Vi har valgt å ta for oss den delen av problemet som omfatter å koble sammen vennegjenger over 18 år med like interesser, slik at de kan avtale arrangementer. Sammen med produkteier har vi kommet fram til at dette skal løses med en webapplikasjon som kan brukes både på pc og mobil.

Valg av teknologistakk

For å lage produktet produkteier ønsker trenger vi en frontend i form av en nettside og backend med mulighet for persistens. Teamet har ulike erfaringer med programmeringsspråk, men en fellesnevner er at flertallet har arbeidet med Python eller Java, samt at flere har erfaring med språk for webutvikling. Vi velger mobilvennlig nettside, da studenter trolig vil foretrekke mobil. Denne kan senere tjenes som en mobilapplikasjon med en progressiv webapp, strukturert i en *model-view-controller* tankegang for mulighet til modularitet.

Frontend: React med TypeScript, styling med Chakra

React er et JavaScript-bibliotek med verktøy for å bygge avanserte grensesnitt. En av grunnene til at vi valgte React over lignende biblioteker som Vue og Angular, er at flere i teamet har erfaring med det. Dessuten har React en større brukerbase med gode ressurser for dokumentasjon, og det eksisterer mye gjenbrukbar kode og delte komponenter.

Videre ønsket vi å skrive koden i TypeScript, som introduserer strenge typer i JavaScript. TypeScript gir oss muligheten til å oppdage feil i koden tidligere, altså vi får mindre teknisk gjeld, slik at kodebasen vår blir mer robust, og vi er mindre utsatt for uventede problemer (Finnegan, 2019). Å velge TypeScript over JavaScript vil gi oss lengre opplæringsperiode, allikevel vil det spare oss tid på sikt.

For å effektivisere design av produktet bruker vi Chakra-UI som er et komponentbibliotek. Chakra-UI gir oss funksjonalitet som temaer og gjenbrukbare komponenter. Det legger til rette for å gjøre webapplikasjonen responsiv på alle enheter. Funksjonaliteten og tidligere erfaring med Chakra-UI gjorde at vi valgte dette biblioteket.

Backend: Django REST, med SQLite

For å håndtere persistens valgte vi Django, et python-bibliotek for å opprette, kjøre og teste databaser, med et REST API gjennom Django REST. Vi vurderte fordelen nevnt i forelesningen med å bruke samme programmeringsspråk i frontend og backend, der Strapi og Mongoose er gode alternativer bygget med henholdsvis TypeScript og JavaScript. Felles for disse er at de tilbyr en forenklet måte å sette opp backend på, men med visse begrensninger når det gjelder funksjonalitet. Grunner til at vi heller valgte Django er at det er et anerkjent rammeverk som enkelte teammedlemmer hadde kunnskap til fra før. Dette legger til rette for å lære biblioteket gjennom teamarbeid og parprogrammering. Dessuten er Django godt dokumentert, og dermed tilpasningsdyktig om det oppstår behov for å utvide

standard funksjonalitet. SqLite er standard med Django og er tilstrekkelig for et enkleste fungerende produkt. Skulle vi senere trenge å optimalisere lagringen vil det enkelt kunne byttes ut.

Standarder og test

For å tilrettelegge for god kodekvalitet i hele kodebasen vil vi innføre en felles struktur. For å oppnå dette bruker vi et felles IDE, integrated development environment, slik at formatering og kodeanaylseringsverktøy, kalt linting, skal være likt mellom maskinene. Vi bruker VSCode på grunn av tilgjengeligheten og utvalget av støttebibliotek/plugins. Vi har valgt noen av de mest kjente formateringsstandardene og linterne, slik at det går raskt å implementere dem:

TypeScript
- Formatering: Google TSGuide[1]
- Linter: ESLint[2]

Python
- Formatering: Black[3]
- Linter: Pylint[4]

Disse implementeres ved hjelp av en felles konfigurasjonsfil i VSCode, og skal ligge i kodelageret slik at eventuelle endringer påvirker alle utviklere.

Vi vil bruke følgende standarder til Git og GitLab for arbeidsprosessen vår, inspirert av scrumstandardene og Gitflow Workflow (Bitbucket, n.d.):
- For hver sprint lages det en *milestone*, som skal inneholde overordnet beskrivelse av sluttmålet for sprinten og dens deadline.
- Brukerhistorier og tilhørende utviklingsoppgaver blir lagt inn som *issues*, og disse skilles henholdsvis med en “userstory” og “task” label.
    - Alle utviklingsoppgaver lenkes til deres brukerhistorie, og brukerhistorien til dens utviklingsoppgaver.
- Kjente feil vil legges inn som *issues* på GitLab med taggen “bug”
    - Slik har vi en enkel oversikt der man kan avgjøre om det er tid til å fikse feilene i løpet av sprinten (Kniberg, 2015, s. 123)
- Under arbeid av *issue* skal man alltid branche fra utviklingsbranchen

Alle commit-meldinger skal være på engelsk i et standardisert format for å sikre konsis leselighet på hele historikken. Følgene mal vil ligge i kodelageret:

> Commit title (Imperative, e.g. “Add user model to backend”)
>
> Commit body (Optional for small commits)
>
> Co-authored-by: name <additional-dev-1@example.com>

[1] https://google.github.io/styleguide/tsguide.html
[2] https://eslint.org/
[3] https://github.com/psf/black
[4] https://pylint.org/

For å gjøre det mulig å måle den interne kvaliteten til koden vil gruppen teste enheter og komponenter med automatiserte tester, i samsvar med første kvadrant i testkvadrantmodellen. De automatiserte testene skal lages med verktøyet Jest for React og verktøy i Django Rest for å teste apiet, fordi dette er verktøy noen av utviklerne har lykkes med tidligere. Vi har valgt å ikke følge testdrevet utvikling, TDD. Dette gjøres fordi gruppen ikke har erfaring med TDD, og praksisen ville da ha økt kostnaden (Perforce, 2016), spesielt når testbibliotekene er fremmed for mange. Dessuten mener Kniberg at TDD er vanskelig å lære seg, og at erfarne medlemmer er essensielt for at resten lærer det, noe vi ikke har (Kniberg, 2015, s.105 - 107).

Tredje kvadrant av testkvadrantmodellen handler om brukervennlighet og funksjonalitet. Her vil gruppen bruke *User Acceptance Testing*, UAT og *Exploratory testing*, ET, for å å avdekke feil og mangler i produktet som er vanskelig å oppdage med kode. UAT handler om å gi kunden/brukeren anledning til å prøve produktet og dets funksjoner, slik at de kan kritisere og foreslå endringer eller nye funksjoner. Målet med UAT er å forsikre seg om at produktet virker slik produkteier ønsker for personer i målgruppen. ET er en metodisk analyse av produktets funksjoner ved å fortløpende lage og gjennomføre tester av funksjonalitet. Analysen bidrar til å finne bugs og uønsket funksjonalitet, men krever at testeren har god forståelse for produktet. (Crispin & Gregory, 2008)
I slutten av hver sprint vil vi benytte oss av UAT og ET for å oppdage problemer og feil i og utenom grensesnittet. Resultatet av testingen vil bli brukt til å opprette nye issues i neste eller nåværende sprint dersom det trengs.

Når er oppgaven ferdigstilt?

Når en utviklingsoppgave har blitt skrevet, med både tester og dokumentasjon, oppretter man en *merge request*. For å sikre produktkvalitet vil alle enhetstester kjøres i GitLab CI pipeline ved hver merge request. Samtidig skal det gjøres *code review* av minst to andre personer, for å sørge for at kodekvaliteten følger standardene og at det er skrevet tilstrekkelige tester. Først etter koden har blitt godkjent og merget i utviklingsbranchen kan utviklingsoppgaven anses som ferdig. Når alle utviklingsoppgaver til en brukerhistorie er ferdige, skal minst én person verifisere at ferdigstilt-kriteriet er oppfylt, som vi definerer til å være at man i applikasjonen kan gjennomføre "*vil jeg"*-delen av brukerhistorien. Etter dette er brukerhistorien ferdig. Denne prosessen sikrer at funksjonaliteten i brukerhistoriene blir oppfylt.

Lanseringsplan

Tabell 1. Oversikt over produktkø med tidsestimat i form av estimeringspoeng.

Identifikator	**Tittel**	**Brukerhistorie**	**Estimeringspoeng**
Sprint 1: 7. - 27. februar			

Mål: Produsere et enkleste fungerende produkt i form av en webapplikasjon hvor man kan lage en profil, bli medlem av grupper og “matche” med andre grupper. Gruppene kommuniserer med kontaktinformasjonen knyttet til brukerne.			
A	Brukerprofil	Som en bruker, vil jeg registrere en profil, slik at jeg kan bruke nettsiden.	10
B	Log-in	Som en bruker, vil jeg kunne logge inn på min konto, slik at jeg kan bruke nettsiden.	2
C	Gruppeprofil	Som en bruker, vil jeg kunne opprette en gruppe med en gruppebeskrivelse, slik at jeg har en gruppe hvor jeg kan legge til vennene mine.	5
D	Legge til venner	Som en gruppe-administrator, vil jeg kunne legge til venner i en gruppe, slik at vi kan møte andre grupper sammen.	3
E	Fjerne bruker/gruppe	Som en nettside-administrator, vil jeg kunne fjerne en bruker og/eller gruppe, for å unngå upassende oppførsel.	1
F	Slette gruppe	Som en gruppe-administrator, vil jeg kunne slette en gruppe hvis den ikke brukes, slik at man ikke kan sende “match” forespørsel.	1
G	Søke etter nye grupper	Som et gruppemedlem, vil jeg kunne gjøre et filtrert søk etter andre grupper, slik at jeg kan sende dem en “match”-forespørsel.	5
H	Liste på “matches”	Som et gruppemedlem, vil jeg kunne se en liste på alle “match”-forespørselene gruppen har fått, slik at gruppen vår kan komme i kontakt med disse gruppene.	3
I	Redigere gruppeprofil	Som gruppe-administrator, vil jeg kunne redigere en gruppeprofil, slik at profilen speiler gruppens interesser.	3
Sprint 2: 7. - 27. mars			
J	Kommunikasjon	Som en bruker/gruppe, vil jeg kunne kommunisere med grupper man har “matchet” med, slik at jeg kan planlegge aktiviteter.	10
K	Redigere brukerprofil	Som en bruker, vil jeg kunne redigere min brukerprofil, slik at den reflektere mine nye interesser.	2

L	Rapportere en bruker/gruppe	Som en bruker, vil jeg kunne rapportere en bruker/gruppe, slik at upassende oppførsel ikke unngås.	3
M	Liste rapporteringer	Som en nettside-administrator, vil jeg kunne se en liste på rapporterte brukere og grupper, slik at jeg kan vurdere rapporteringens relevans.	4
N	Forlate gruppe	Som en bruker, vil jeg kunne forlate en gruppe, slik at jeg ikke er medlem av grupper jeg ikke lenger identifiserer meg med.	
O	Fjerne bruker fra gruppe	Som en gruppe-administrator, vil jeg kunne fjerne en bruker fra gruppen min, slik at jeg kan fjerne medlemmer som ikke ønsker å være med.	
Hvis vi får ekstra tid:			
P		Som en bruker, vil jeg kunne gjøre et filtrert søk på grupper, slik at jeg kan finne grupper som interesserer meg.	
Q		Som en gruppe-administrator, vil jeg kunne akseptere medlemsforespørsler, slik at alle mine venner kan bli med.	

Tabell 2. Oversikt over utviklingsoppgavene for de tre første brukerhistoriene.

Brukerhistorie	**Utviklingsoppgave (som skrevet i GitLab)**	**Estimeringspoeng**
A	Create Django Development Environment	0.5
	Create frontend development environment	0.5
	Create a Django profile model	2.5
	Configure Django to authenticate and register users	3
	Create register user form	3

	Implement user persistence	1
	Configure Docker-compose	Deprioritized. To be done if there is time.
B	Create login page/form	1.5
	Connect the login functionality to the backend	0.5
C	Create Django group model	2
	Create group creation frontend page/form	3

Hver sprint skal vare i 3 uker. Mandagen og fredagen mellom sprintene skal henholdsvis brukes til forberedelse til gjennomgang og refleksjon, samt planlegging til neste sprint. I arbeidslivet er det vanlig å bruke 2 uker per Sprint, vi jobber ikke 100% og tenker derfor at 3 uker vil passe bedre for timeantallet gruppen skal legge ned i prosjektet.

Kilder

Bitbucket, n.d. Gitflow Workflow [WWW Document]. URL https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow) (accessed 2.4.22).

Crispin, L., Gregory, J., 2008. Agile testing: a practical guide for testers and agile teams, The Addison-Wesley signature series. Addison-Wesley, Upper Saddle River, NJ.

Finnegan, C., 2019. The Major Benefits of Using Typescript [WWW Document]. URL https://medium.com/swlh/the-major-benefits-of-using-typescript-aa8553f5e2ed (accessed 2.4.22).

Kniberg, H., 2015. Scrum and XP from the trenches. C4Media.