TDT4140 - Programvareutvikling

Leveranse 2

Antall ord utenom forside og referanser: XXXX

Gruppenummer: 10

Produktnavn: GjørNo’

Medlemmer som har bidratt til leveransen:

Fornavn	**Etternavn**	**Studmail**
Elise Adriane	Telje	eliseate@stud.ntnu.no
Mikal Hobbelstad	Viga	mikalhv@stud.ntnu.no
Johanne	Glende	johagl@stud.ntnu.no
Kristine	Larssen	krlarss@stud.ntnu.no
Iver Olai Lade	Gjørvad	iogjorva@stud.ntnu.no
Jørgen Vartdal	Halse	jorgevh@stud.ntnu.no
Ask Fresvig	Heggen	askfh@stud.ntnu.no

Problembeskrivelse

Produkteier ønsker et produkt som gjør det enklere å finne på aktiviteter å gjøre. Sammen med produkteier har vi blitt enige om å lage en nettside hvor brukere og organisasjoner kan legge ut slike aktiviteter. Videre kan brukere filtrere og melde seg på det de ønsker. Det skal i tillegg være mulig for et admin-team å slette, opprette og moderere forespørsler, samt hente ut statistikk.

Valg av teknologi-stack

Vi har valgt å benytte Python og TypeScript som programmeringsspråk til henholdsvis server- og web-delen av prosjektet. Årsaken til dette er at disse er open source og flertallet har kjennskap til syntaks. I tillegg valgte vi TypeScript foran JavaScript fordi vi mener at det gjør det enklere å forstå og bygge videre på hverandres koder når elementtypene spesifiseres eksplisitt. CSS og HTML brukes for å bygge opp og designe nettsiden.

For å effektivisere produktutviklingen, har vi valgt å benytte rammeverkene Django, Django REST og React. Dette gjør vi fordi det i artikkel 4 (Suschevich & Birukova, 2020) står at disse passer godt for mindre prosjekter, slik som vi skal lage. I tillegg er disse ready solutions for vårt prosjekt, som det i samme artikkel sies å være et godt valg når man har et prosjekt som skal utvikles over kort tid. I tillegg skrives det i artikkelen at man skal velge det man er komfortabel med. Etter diskusjon i gruppa og på bakgrunn av tidligere erfaringer kom vi fram til at det var disse to rammeverkene vi var mest komfortable med.

Et annet argument for å velge Django er at man får med SQLite, som er en innebygd databaseteknologi vi kan bruke. Alt i alt mener vi at dette tilfredsstiller både våre egne og produkteiers ønsker om teknologiske egenskaper tilknyttet produktutviklingen.

Standarder

Etter en diskusjon rundt standarder og konvensjoner vi ønsker å følge under produktutviklingen, har vi kommet fram til følgende:

- For kodeformatering i Python og TypeScript, vil vi laste ned plugins for henholdsvis *black* og *Prettier*. Vi har valgt disse fordi de er mye brukt i praksis, og gjør at alle i teamet skriver på samme format.
- Som navngivingskonvensjoner har vi i Python primært valgt å bruke lowercase + underscore. I React skal vi bruke PascalCase på filnavn og komponenter, og camelCase på instanser. React-komponenter skal samles i en mappe kalt 'components'. Hver enkelt komponent skal ligge i en fil kalt 'index.tsx' under en mappe med samme navn som komponenten selv (PascalCase). Tilsvarende vil gjelde for annet enn komponenter. *(src/components/Navn/index.tsx)*
- Komponenter i React skal hovedsakelig kodes som funksjonelle komponenter.

GitLab:

Hver sprint starter med en sprintplanlegging og slutter med en milepæl ("milestone") i GitLab. Under planleggingen blir brukerhistoriene gjort om til oppgaver ("issues"). Det vil også være oppgaver, blant annet "tech stories", som ikke tilhører noen brukerhistorier eller milepæler, men som likevel må gjøres. Å ta med en oppgave i en sprint skjer ved at den blir flyttet fra det man kaller **product backlog** til **sprint backlog** på issue boardet.

Hver sprint starter med en sprintplanlegging og slutter med en milepæl ("milestone") i GitLab. Under planleggingen blir brukerhistoriene gjort om til oppgaver ("issues") og disse sammen med "tech stories" blir lagt inn i product backlog. Deretter blir oppgavene vi har valgt å ta med til først sprint flyttet til sprint backlog på issue boardet.

Når det gjelder utviklingen får personer en oppgave i GitLab og lager en egen gren ("branch") for å arbeide på denne oppgaven. Da skal oppgaven flyttes til "doing" på boardet. Alt fra grennavn til format på commit-melding følger en fast struktur og blir bedre forklart i et eget dokument CONTRIBUTING.md som kommer til å ligge i repoet.

Etter en oppgave er ferdig utviklet skal det lages en fletteforespørsel ("merge-request") fra utviklingsgrenen til hovedgrenen. Denne forespørselen skal noen andre enn utvikleren selv se på, for å sørge for at all kode som går på hovedgrenen er i orden.

For å sikre kvalitet skal prosjektet bruke en GitLab-pipeline som automatisk kjører testene i utviklingsgrenen, kodekvalitet og formatering samt rapporterer testdekning. Dersom noen av testene feiler, vil ikke forespørselen gå gjennom.

Testplan

Hvitboks-testing:

I følge Kniberg (2015, s.115) er det essensielt at man utfør manuell testing av systemet, og at dette bør gjøres av teamets egne medlemmer. Derfor har vi bestemt at enhetstesting skjer manuelt gjennom hele sprinten, der utviklerne selv står ansvarlig for å teste egen kode. Vi har derimot nedprioritert å skrive egen testkode for enhetene pga mangel på tid. Funksjonaliteten blir videre testet av en med-utvikler under code-review. Dersom koden har bugs må utvikleren rette det opp, men dersom det ikke finnes noen åpenbare feil er koden klar for merging. Etter merging til master testes systemet manuelt om alt fungerer som det skal, og om ytelsen og robustheten fortsatt er slik man ønsker.

På slutten av hver sprint utføres systemtesting, hvor det er fokus på å finne bugs i systemet som bør rettes opp i før scenario-testing (Sommerville, 2015, s.245). Systemtestingen utføres at et valgt test-team med to personer. Vi ønsker å være et kryssfunksjonelt team og velger derfor et test-team innad i gruppen (Kniberg, 2015, s.117). Systemtestingen er ferdig dersom man tilsynelatende ikke klarer å finne flere bugs.

Svartboks-testing:

Første arbeidsdag i ukene hvor det er produktgjennomgang skal scenario-testing gjennomføres. Testingen gjennomføres av test-teamet som skal klikke seg gjennom systemet som om man var en sluttbruker (Sommerville, 2015, s.246). På denne måten kan man teste at brukerhistoriene tilfredsstiller de gitte kravene. Derfor er det sentralt at alle issues tilknyttet brukerhistoriene må ha vært gjennom code-review før testingen. Scenario-testene er ferdig dersom systemet tilfredsstiller produkteiers krav.

Releaseplan

I samsvar med det Kniberg skriver om iterasjonslengde på rundt tre uker, skal vi under produktutviklingen ha to iterasjoner på tre til fire uker hver (Kniberg, 2015, s.22). Andre iterasjon blir litt lengre enn den første, men fordi påske innfaller i andre iterasjon, blir denne

forskjellen neglisjerbar. Ettersom vi er studenter og ikke jobber fulltid som et normalt utviklingsteam, vil vi få gjort mindre arbeid. Derfor mener vi at det er lurt med en lengre sprint, selv om dette strider mot Knibergs anbefalinger.

Første iterasjon er planlagt fra uke 5 til uke 8, med avslutning på samme dag som gjennomgang 1. Vi setter da av tid til retrospektiv, og reflekterer rundt gjennomført iterasjon. Retrospektiv 1 skal leveres uke 9, og vi setter i gang med iterasjon 2 rett etter retrospektiv-møtet. Andre, og siste, iterasjon er ferdig med gjennomgang og retrospektiv i henholdsvis uke 14 og 15.

I Tabell 1 står brukerhistoriene i prioritert rekkefølge. Prioriteringen er gjort på bakgrunn av diskusjon med produkteier og det vi i fellesskap kom frem til var viktigst. Vi har valgt å ta med de fire første brukerhistoriene i iterasjon 1. Flertallet av brukerhistorier blir da i iterasjon 2, men som Kniberg sier tar vi heller med for lite enn for mye ved usikkerhet for første iterasjon. For estimering av Story Points for brukerhistoriene brukte vi Planning poker, som er en god måte å unngå at folk blir påvirket av andres tanker (Kniberg, 2015, s.38-39). Med løpende kontakt med produkteier i både første og andre iterasjon, kan vi ta høyde for uforutsette problemer som for eksempel feil i satt story points eller endring i prioriteringsrekkefølge.

ID	**Navn**	**Brukerhistorie**	**Story Points**
Sprint 1			
A	Se aktivitet	Som en bruker, ønsker jeg å kunne se aktiviteter, slik at jeg kan gjennomføre disse	5
B	Legge til aktivitet	Som en bruker ønsker jeg å kunne legge ut aktiviteter offentlig slik at andre kan bli inspirerte til å gjøre det samme	8
C	Lag profil	Som en bruker/organisasjon ønsker jeg å kunne lage en egen profil, slik at jeg kan delta/få innsyn på plattformen	13

D	Innloggingssystem	Som en bruker/organisasjon ønsker jeg å kunne logge inn i egen profil, slik at jeg får oversikt over mine aktiviteter	5
Sprint 2			
E	Opprette organiserte aktiviteter	Som en organisasjon ønsker jeg å kunne opprette organiserte aktiviteter slik at andre brukere kan melde seg på	13
F	Melde seg på aktiviteter	Som en bruker ønsker jeg å kunne melde meg på aktiviteter i regi av organisasjoner slik at jeg kan delta på aktuelle turer	8
G	Markere aktiviteter	Som en bruker/organisasjon, ønsker jeg å kunne markere aktiviteter slik at de blir en del av en aktivitetslogg	13
H	Skille aktiviteter	Som en bruker ønsker jeg å kunne skille mellom aktiviteter lagt ut av organisasjoner og andre brukere, slik at jeg kan få opp det jeg ønsker å se	3
I	Filtrere/sortere aktiviteter	Som en bruker, ønsker jeg å kunne filtrere/sortere aktiviteter, slik at jeg kan finne de turene jeg har interesse for	13
J	Moderator-tilgang	Som en admin, ønsker jeg å kunne slette, opprette og moderere aktiviteter slik at aktivitetene er kvalitetssikret	13
K	Statistikkside	Som en admin ønsker jeg en statistikkside slik at jeg kan ha oversikt over hvilke aktiviteter som er mest populære	20

Ekstra			
L	Deltakere på private aktiviteter	Brukere kan organisere turer med et spesifisert antall deltakere.	13
M	Chat-funksjon	Som bruker av applikasjonen, ønsker jeg å kunne kommunisere direkte med organisasjoner,slik at jeg kan melde meg på aktiviteter eller få mer informasjon om interessante aktiviteter.	100

Tabell 1: Oversikt over brukerhistorier med beskrivelse og story points

Målet for første iterasjon er å levere en fungerende utgave av applikasjonen, men med noe begrenset funksjonalitet. Det viktigste blir å implementere grunnleggende funksjoner som reflekterer de viktigste elementene produkteier ga uttrykk for.

Referanseliste

Kniberg, H. (2015). *Scrum and XP from the Trenches*. InfoQ.

Sommerville, I. (2015). *Software Engineering*. Pearson

Artikkel 4:
Suschevich, A. & Birukova, D. (2020). *What Is a Technology Stack? Choosing the Right Tech Stack For Your Web Project.* Hentet 04.02.21 fra https://www.intexsoft.com/blog/post/tech-stack.html