FAGPRØVE I IT-UTVIKLERFAGET

Planlegging

David Hasselø

Dato/tid for prøve

Start: 04.06.25 kl. 08:00
Slutt: 13.06.25 kl. 16:00

David Hasselø
david.hasselo@omega365.com

Innhold

Tolkning av oppgave 2
Funksjonelle krav 2
Drøfting av løsninger 3
Framdriftsplan 6
Valg av teknologier 7
Design 10
Kilder 12

Tolkning av oppgave

Jeg har fått i oppdrag å utvikle en applikasjon for interne konkurranser i Omega 365, hvor ansatte registrerer stikk-ut turer. Brukeren skal kunne opprette og delta i konkurranser, registrere egne turer, samt se aggregert rangering. Det skal også være en funksjon for å gjennomføre en flaxlodd-trekning som belønner innsats med høyere vinnersjanse. Løsningen skal inkludere både frontend og backend, og følge designprinsippene i Omega 365s brand-manual.

Funksjonelle krav

Konkurranser

Administratorer kan opprette konkurranser med start/sluttdato og beskrivelse.

Kun én aktiv konkurranse om gangen.

Tidligere konkurranser skal vise vinner og premie.

Brukere og innlogging

Brukere registrerer seg med e-post, fornavn, etternavn og passord.

Innlogging skjer med e-post og passord

Turrapportering

Innloggede brukere kan se, legge til, redigere og slette egne turer.

Turer vises i “Mine turer”, sortert nyeste først.

Hver tur har dato, sted og hvem som deltok.

Rangering

Viser aggregert antall turer per bruker i en konkurranse.

Kun egne turer vises, rangering er readonly.

Rangering vises som tabell eller diagram.

Trekning

Admin kan gjennomføre trekning etter konkurransens slutt.

Én tur = ett flaxlodd (vektet sannsynlighet).

Resultat lagrer vinner, premie og tidspunkt.

Mobilvennlighet

Applikasjonen skal fungere godt på mobil.

Det skal være enkelt å registrere turer rett etter man har gått dem.

Drøfting av løsninger

Frontend:

Jeg ønsker å benytte ren HTML og CSS sammen med bootstrap 5 for å sikre rask og responsiv oppbygging av brukergrensesnittet. Bootstrap gjør det enkelt å følge prinsipper for universell utforming og gir god mobilstøtte, som er viktig for denne løsningen. Brukergrensesnittet skal være intuitivt, med tyldeig navigasjon mellom konkurranser, egne turer og rangering. Javascript brukes til å håndtere dynamiske elementer som skjemaer, hente data fra API-et via fetch, samt validere brukerinput før innsending.

Backend:

Jeg bruker Node.js med Express for å bygge et REST-basert API. Backend vil ha ansvar for å håndtere følgende hovedfunksjoner:

- CRUD-operasjoner for brukere, turer og konkurranser
- Autentisering med e-post og passord. Ved registrering vil passordet bli hashet med bcrypt og lagret sikkert. Ved innlogging verifiseres passordet, og ved gyldig innlogging utstedes en JWT-token som klienten bruker for autentisering i videre forespørsler.
- Aggregert rangering, basert på antall turer per deltaker, beregnes og eksponeres via en readonly-rute

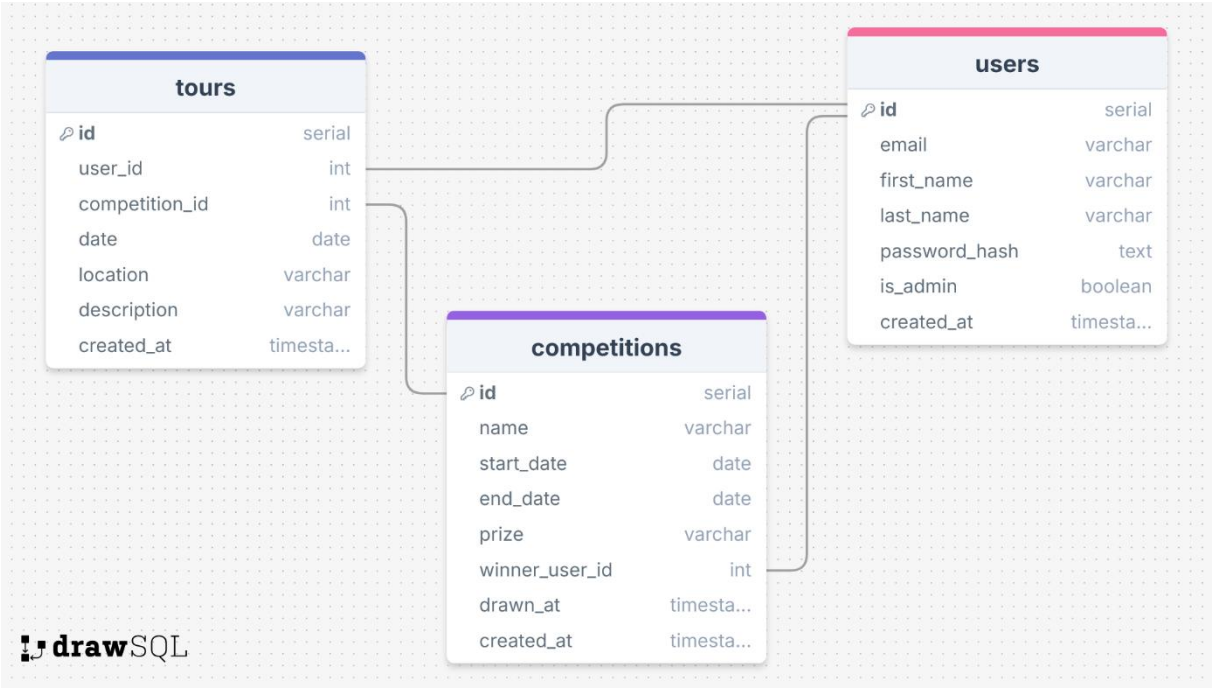
- Trekning av vinner, som kun kan utføres av admin, og som vekter antall turer som antall flaxlodd per bruker.

Backend vil også validere alle innkommende data og returnere meningsfulle feilmeldinger ved feil.

Database:

Jeg benytter PostgreSQL som relasjonell database til å lagre all data, inkludert brukere, turer og konkurranser. Jeg modellerer med relasjoner mellom brukere og turer, og inkluderer felter som data, sted og hvem som deltok.

Datamodellen vil består av disse tabellene:



Hver bruker (users) har en unik ID, navn, e-post og passord som er hashet med bcrypt.

En konkurranse (competitions) har navn, start- og sluttdato, premie, samt hvem som vant og når trekningen skjedde.

En tur (tours) er knyttet til både en bruker og en konkurranse. Hver tur lagrer informasjon som dato, sted og beskrivelse.

Relasjonene mellom tabellene er implementert med foreign keys, slik at hver tur refererer til en bruker og en konkurranse. I tillegg refererer *competitions.winner_user_id* tilbake til *users.id*, noe som gjør det enkel å hente ut tidligere vinnere og deres premie.

Link til diagramverktøyet: (drawsql)

https://drawsql.app/teams/omega-365-1/diagrams/david-fagproeve-stikkut

API:

Jeg bygger et REST-API hvor hver ressurs (bruker, tur, konkurranse) har dedikerte ruter, Ruter som krever autentisering vil være beskyttet med JWT-verifisering via middleware. Eksempler på planlagte endepunkter:

- POST /register – registrering av bruker med e-post og passord
- POST /login – innlogging og utstedelse av JWT token
- GET/me/tours – henter alle turer til innlogget bruker
- POST /tours – registrerer ny tur
- PUT /tours/:id – oppdaterer eksisterende tur (bare egne)
- DELETE /tours/:id – sletter egen tur
- GET /ranking – henter aggregert rangering (readonly)
- POST /draw – admin-trigger for flaxlodd-trekning, basert på antall turer

Alle API-kall vil returnere JSON og benytte passende HTTP-statuskoder

Link til API dokumentasjon: (Postman)
https://documenter.getpostman.com/view/44720323/2sB2qi9dh3

Autentisering og sikkerhet:

Brukerne registrerer seg med fullt navn, e-post og passord. Passordet blir både hashet og saltet med bcrypt (kryptografisk hash-funksjon designet for passord-hashing) før de lagres i databasen, slik at det ikke er mulig å hente ut råpassord. Ved innlogging sammenlignes innskrevet passord med det hashede passordet i databasen.

Dersom innloggingen er vellykket, genereres en JWT-token (JSON Web Token) som sendes tilbake til klienten. Denne tokenen brukes for å identifisere fremtidige forespørsler mot API-et, for eksempel ved å inkludere tokenen i Authorization-header. På backend verifiseres tokenen for hver beskyttet rute, og tilgang gis kun dersom tokenen er gyldig.

Framdriftsplan

Jeg planlegger å bruke de første 4,5 dagene til selve utviklingen, med en strukturert fremdrift hvor backend og database prioriteres tidlig, og frontend og integrasjon kommer deretter. Deretter settes det av god tid til testing, finishing touches, samt egenvurdering og dokumentasjon. Dersom jeg blir ferdig med enkelte deler tidligere, går jeg videre til neste punkt i planen for å ligge foran. Jeg planlegger også å skrive en kort daglig logg ved slutten av hver gjennomføringsdag, hvor jeg reflekterer over hva som ble gjort, hvordan det gikk, hva som kunne vært gjort annerledes, og hva som er neste steg. Denne loggen vil brukes som grunnlag for egenvurderingen, og bidra til mer presis og ærlig refleksjon.

Dag 1 (04.06) – Planlegging og prosjektstart

- Skrive plan til gjennomføring av oppgaven
- Lage databasediagram, wireframe og API-dokumentasjon

Dag 2 (05.06) – Sende inn planlegging, database, backend og enkel frontend

- CRUD for «users»-tabell
- Se over og send inn planlegging til prøvenemnd,
- Oppsett av PostgreSQL og backend (Express, JWT, bcrypt, baseruter)
- Enkel frontend med skjemaer for å verifisere postgresql tilkobling og oppførsel

Dag 3 (06.06) – CRUD og autentisering

- CRUD for «tours» og «competitions»-tabell
- Implementere JWT-verifisering for beskyttede ruter
- Oppsett av http kall
- Teste med postman

Dag 4 (10.06) – Aggregert rangering og rangering API

- Aggregert rangering
- Admin funksjoner
- Opprette frontend for visning av konkurranser og rangering

Dag 5 (11.06) – Full frontend og mobiltilpasning

- Fokus på frontend
- Implementere dynamisk oppdatering med Javascript og fetch()
- Validere skjemaer og feilmeldinger
- Responsivt design
- Verifisere brukere kun ser egne turer og aggregert data

Dag 6 (12.06) – Finishing touches, testing og egenvurdering

- Gjennomgå all funksjonalitet, gjøre siste forbedringer
- Gjennomføre et testskjema hvis det er tid til det, gjerne få andre til å gjennomgå skjema med all funksjonalitet
- Når gjennomføringen er fullført start på egenvurdering

Dag 7 (13.06) – Egenvurdering og dokumentasjon

- Skrive ferdig egenvurdering
- Ferdigstille teknisk dokumentasjon
- Brukerveiledning

Valg av teknologier

Backend:

Node.js og Express benyttes for å bygge backend-løsningen. Express gir en fleksibel struktur for å lage REST-API-er, og Node.js er godt egnet til raske og skalerbare webtjenester. Det er i tillegg en teknologi som jeg har tatt i bruk i egne prosjekter tidligere noe som gjør at jeg er kjent med det framfor andre runtime-environments.

PostgreSQL brukes som relasjonsdatabase. Den har høy ytelse, støtte for relasjoner og komplekse spørring – som er hensiktsmessig i dette tilfellet for å hente ut aggregert rangeringsdata og utføre vektede trekninger. Jeg har tidligere brukt mye SQL Server på jobb relaterte oppgaver, men jeg valgte postgresql siden den er gratis uten begrensninger, mye mindre tungt å kjøre, mye bedre støttet for deployment i docker og railway og er generelt bedre å bruke for slike små fullstack apper.

Bcrypt benyttes for å hashe og salte brukernes passord på en sikker måte før de lagres i databasen, som øker sikkerheten og gjør det ekstremt vanskelig for hackere ved en datalekkasje å gjette hva det originale passordet er, selv med tilgang til hashene.

JWT (JSON Web Tokens) brukes for autentisering. Ved innlogging får brukeren en token som benyttes i videre forespørsler for å identifisere brukeren sikkert. Det gjør også livet for databasen enklere siden den lagrer brukerinfo som id i selve tokenen, slik at serveren slipper å slå opp brukeren i databasen for hver forespørsel og reduserer belastning.

Frontend:

HTML, CSS og Javascript er valgt for å utvikle brukergrensesnittet. Dette gir meg full kontroll og enkle tilpasningsmuligheter. Dersom jeg hadde mer forståelse for React.js hadde jeg nok brukt det med tailwind, siden det er enklere å skalere med f.eks komponentbasert struktur som gjør det enklere å organisere og gjenbruke, mer oversiktlig og ryddig. Jeg velger bevisst å bruke html, css og vanilla javascript da det er det jeg har jobbet med mest og føler meg mest komfortabel med, samtidig så er det greit for slike små prosjekt og gjør at jeg ikke trenger å innføre nye ukjente teknologier under fagprøven. Disse er lettvekts teknologier uten noe særlig rammeverk som gir raskere lasting og lavere strømbruk hos brukeren.

Bootstrap 5 brukes som css-rammeverk for å lage et responsivt og tilgjengelig grensesnitt som fungerer godt på mobil, nettbrett og desktop.

Testing og dokumentasjon

Postman brukes til å teste og dokumentere API-endepunktene. Samlingen publiserer som en lenke, og beskriver hvordan API-et kan brukes.

Excalidraw er et veldig enkelt tegneverktøy som brukes til å lage skisser og brukergrensesnittutkast som gjør det enklere å visualisere ideer og tanker rundt appen for å få det fullstendige bildet. Jeg har tidligere brukt interne ressurser som designere til å skissere en tegning som har blitt brukt hos kunder med min funksjonalitet, men jeg velger å skissere wireframes selv da dette krever tid av andre i et lite tidsrom.

Logger og notater føres daglig i løpet av gjennomføringen og danner grunnlag for egenvurderingen. Jeg bruker hovedsakelig Microsoft Word til dokumentasjon og loggføring, og Obsidian (notatapplikasjon) til arbeidsoppgaver, to-do og lignende.

Verktøy og utviklingsmiljø

Microsoft Visual Studio Code brukes som utviklingsmiljø fordi det er lettvektig, raskt og har gode utvidelser for både backend og frontend-utvikling. Det er en IDE som jeg har mye erfaring med og perfekt til slike fullstack applikasjoner.

Git og GitHub brukes til versjonskontroll og hele prosjektet gjøres tilgjengelig for prøvenemnda via GitHub. Fordelen med github er at man kan spole tilbake til tidligere versjoner av koden, lage eksperimentelle grener (branches) uten å påvirke hovedkoden og har god støtte til vscode som gjør at man kan utføre endringer til github rett fra terminalen i vscode. I tillegg gjør det at jeg jobber mer effektivt uten å duplisere innsats.

Distribusjon og drift

Docker bruker jeg for å containerisere applikasjonen, noe som gir forutsigbart oppsett og enkel lokal testing uavhengig av operativsystem. Dette gjør det enkelt for andre utviklere å kjøre appen ved at docker håndterer alt av pakking av applikasjonen.

Railway brukes som deployment-plattform for backend og database. Fordelen med dette er at man kan koble til et github repo og når man pusher ny kode vil den bygge og deployer appen automatisk. Dette er perfekt for små prosjekter da man får backend og database oppe i løpet av minutter uten behov for avansert DevOps-oppsett. Den har innebygd postgresql støtte, og kan enkelt åpnes via en url og brukes av alle med tilgang til linken, dataen som blir sendt gjennom appen vil man kunne se i postgresql databasen på dashboardet i railway. Jeg valgte denne fordi det også speiler hvordan jeg jobber i Omega 365. Når man får en forespørsel fra en kunde som krever at man f.eks skal skrive en ny funksjon, vil man ikke skrive dette direkte i samme plattformen som kundene bruker, da det kan skape problemer for dem som gjør de misfornøyde med produktet. Derfor har man egne dedikerte «sites» eller miljøer der man gjør testing og deretter pusher kode til produksjon etter testingen. I mitt tilfelle har jeg mitt lokale miljø i vscode med en egen localhost server som utviklingsmiljø, og railway som produksjonsmiljø med sin egen dedikerte database slik at det ikke blir påvirket av utviklingen. En annen fordel med railway er at den blir automatisk nedskalert når applikasjonen ikke er i bruk, som sparer serverressurser og bidrar til økt bærekraft.

Design

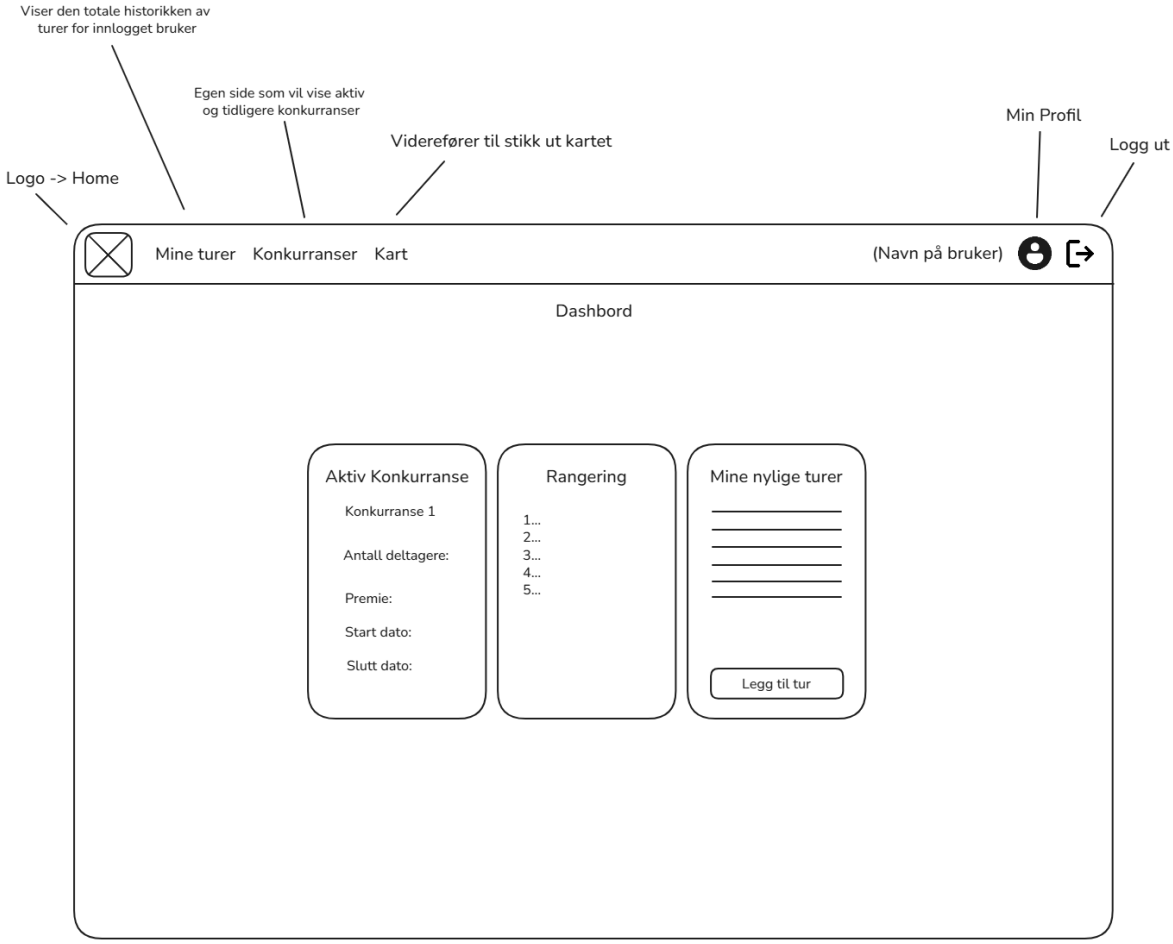
Brukergrensesnitt

Brukergrensesnittet bygges med HTML, CSS og Bootstrap 5, som gir ferdige komponenter med god støtte for universell utforming og mobilvennlighet.

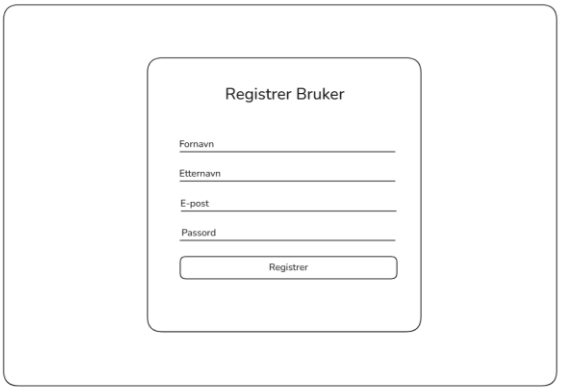
Jeg forsøker å følge Omega 365 sin visuelle profil i farger, fontvalg og knapper, i tråd med brand-manualen som er lenket i oppgaveteksten. Dette gir applikasjonen et profesjonelt og helhetlig uttrykk som passer inn i bedriftens eksisterende design.

Tilgjengelighet og universell utforming

For å opprettholde god tilgjengelighet slik at appen kan brukes av flest mulig skal jeg sørge for at knapper og lenker er godt beskrevet med klare kontraster, skjemaer skal ha tilknyttede labels for skjermleserstøtte og at alle interaktive elementer skal kunne navigeres med kun tastatur.



Jeg kom fram til dette enkle dashbord inspirerte designet, der fordelen er at brukere kan se alt av nødvendig data uten å måtte gå innpå andre sider. Dersom de ønsker en mer detaljert og oversiktlig visning over ulike element kan de åpne den dedikerte siden for tilhørende element som gir full oversikt og historikk.



Ren og enkel registrerings- og innloggingsside for å opprettholde lett tilgjengelighet og brukervennlighet.

Kilder

Express.js dokumentasjon – https://expressjs.com/

JWT – https://jwt.io/introduction

WCAG retningslinjer – https://www.w3.org/WAI/WCAG21/quickref/

Omega 365 Brand Manual - https://www.omega365.com/media-resources/brand-manual