Tolkning av oppgaven:

For å klare å ikke gå ekstremt langt over anbefalt tidsbruk har jeg valgt å ikke implementere noe innloggingsløsning på dette. Dette har to viktige betydninger for oppgaven.

1. Det er mulig for deltakere å registrere seg, men det er også mulig for en deltaker å registrere andre deltakere. For å endre dette måtte det vært en eller annen valideringsløsning av den hver deltaker som registrerer seg for å sjekke at det er seg selv de registrerer. Dette har jeg ikke fått til på såpass kort tid. I forsøk på å gjøre dette litt sikrere så har jeg valgt at ett telefonnummer kan kun bli benyttet en gang per arrangement. Altså jeg kan ikke registrere 2 personer på mitt telefonnummer, eksempelvis.
2. Det er den enkle implementeringen mulig for alle og se alle som deltar på hvert arrangement. Dette er i utgangspunktet gjort slik for å kunne vise fram at dette fungerer. Spesielt siden alle kan registrere hvem som helst, så hadde det her vært naturlig å gjemme dette bak en innloggingsløsning, slik at kun administratorer/operatører kan se deltakere.

Hva jeg ville gjort med bedre tid på dette:

De to viktigste tingene som måtte ha blitt forbedret dersom man hadde bedre tid her er sikkerhet og complience. Spesielt i backend er det nå manglende sikkerhet. API ’et er nå åpent for alle som vet IP og port (URL holder dersom man tilknytter dette), noe som ikke vil være helt heldig dersom man publiserer det. Det ville da vært naturlig med flere ting som ratelimit, recaptcha, API-nøkkel, autentisering, tilgangskontroll.

For complience så er det litt forskjellig som avtale med forbruker som må på plass, som også inneholder relevant info og andre nødvendigheter som gjelder GDPR. Videre hadde det også vært nødvendig med en cookie-policy, da det hadde vært naturlig å implementere cookies i forbindelse med autentisering. Til slutt så hadde det også vært viktig å sørge for å opprettholde GDPR lovgivning.

Med mer tid etter dette så hadde det også vært en fordel å oppgradere UX’en i frontend, og sørge for en mer sømløs brukeropplevelse, hvor programmet hadde vært brukervennlig og intuitivt.

Med tilgangskontroll, så hadde det også vært naturlig å refaktorere den koden som allerede er, slik at man gjemmer noe bak en innloggingsløsning, som nevnt over.

Oppgave 1.

For å lage en slik løsning så tenker jeg at de viktigste tingene å tenke på vil være at det følger lovgivning, sikkerhet for dataene og en enkel og sømløs brukeropplevelse.

I ett slik system så vil det være en god del lover og regler å forholde seg til. Dette er spesielt siden man behandler persondata for brukerne som registrerer seg. Man vil da være blant annet bundet av GDPR lovgivning. Videre er sikkerhet i systemet en forlengelse av dette. Dette er for å sørge for at ikke data kommer på avveie, men også for å sørge for at data er korrekt. Sikkerhet vil også være en viktig del for å overholde punktet i oppgaveteksten som sier at «det er viktig at risikoen for feil i systemet er så liten som mulig».

Fra dette tenker jeg at det er viktig å sette seg klare retningslinjer for hvordan man ønsker at programmet skal være organisert. Hva skal være bak innloggingsløsninger, hvem skal kunne registrere hvem osv. Under dette punktet tenker jeg også at det naturlig at man har en plan på hvordan man har tenkt til å tjene penger på programmet (forutsatt at det er dette som er målet). Organiseringen av programmet vil kunne være forskjellig basert på dette, alt etter om å selge reklameplasser, eller om arrangementene betaler for å benytte programmet etc. I min lille versjon så har jeg valgt at alle skal kunne registrere hvem som helst. Jeg har også valgt at telefonnummer skal være unikt for hver som registrerer seg. Dette er for at man ikke skal kunne registrere seg selv mange ganger, eller registrere mange personer på samme telefonnummer. Poenget her ser jeg for meg at er at man skal kunne ta kontakt med hver og en som har registrert seg i etterkant, og da vil det være naturlig at hver har forskjellige telefonnummer. Jeg har ikke her valgt å ta hensyn til dette med inntjening.

Videre er det viktig å tenke på hvordan en database for dette er strukturert. Hvilke valg man tar her ser jeg for meg er avhengig av hvordan man ser for seg at programmet skal være strukturert, og derav hvilken informasjon som skal lagres.

Alt i alt syns jeg altså det viktigste man bør tenke på for å lage et slikt system er å ha et klart mål for hvilket problem programmet skal løse, og ut fra dette sette seg tydelige visjoner for hvordan programmet skal struktureres. I tillegg er det altså viktig at man har god sikkerhet i systemet, og at man overholder lover og regler.

Oppgave 2.

Det er mange rammeverk som kan brukes for å lage ett slikt system. Rammeverk er selvfølgelig også avhengig av hvilket språk man skriver i. I min lille implementasjon har jeg skrevet alt i JavaScript. Jeg tar derfor dette som utgangspunkt videre i oppgaven. I frontend er jeg personlig veldig fan av React. Her kan man lage komponenter som man gjenbruker flere steder i layout, og det er en ganske enkel syntaks. For å skrive CSS koden i frontend, så liker jeg også å benytte Tailwindcss, fordi dette gir en ganske enkel oversikt over hvilke tags som bruker hvilken CSS kode, samt at Tailwindcss er bygd opp slik at de har en del kommandoer som egentlig består av flere CSS linjer. Min opplevelse er derfor at man kan skrive CSS raskere ved bruk av dette, enn uten. For å få maksimalt utbytte av Tailwindcss lønner det seg å benytte seg mye av komponenter. Grunnen til dette er at man må skrive CSS koden (i Tailwindcss) på nytt alle steder, hvis man eksempelvis prøver å lage to sider som skal se veldig like ut, uten at man benytter komponenter, må man da skrive all CSS koden på nytt på begge steder. Jeg vet også mange liker å benytte Bootstrap for sin CSS kode.

På backend siden så liker jeg å benytte Express som rammeverk for å bygge en server. Dette forutsetter igjen at man også benytter JavaScript til å skrive backend koden. Express gjør at man kan sette opp en server som er funksjonell ganske raskt, og det er også her en veldig enkel syntaks for hvordan man benytter dette rammeverket.

Oppgave 3.

Jeg tenker at den enkleste løsningen for å lage en liste som skal brukes flere steder i layout er å lage en komponent med listen, som man så kan gjenbruke flere steder i layout. Dette forutsetter riktignok at den biten av listen man lager en komponent av, skal være likt strukturert alle steder hvor den vises fram. Dersom man ønsker store forskjeller mellom hvordan listen framstilles, og kanskje også i hvordan informasjon som framkommer av listen på forskjellige steder, så tenker jeg at det kanskje er mer naturlig å lagre informasjonen som listene bruker på ett felles sted for alle komponenter. En løsning på dette er eksempelvis ved å bruke React context eller Redux for å ha en universal state i programmet. Dersom all info som listene skal bruke kommer fra backend, er det også mulig å lagre informasjonen i en cookie.