Høgskulen på Vestlandet

Fakultet for ingeniør- og naturvitskap

Institutt for datateknologi, elektroteknologi og realfag

Geppetto - Fra Windows til Web

Systemdokumentasjon

Versjon 4.0

Dokumentet er basert på Systemdokumentasjon utarbeidet ved NTNU. Revisjon og tilpasninger til bruk ved IDER, DATA-INF utført av Carsten Gunnar Helgesen, Svein-Ivar Lillehaug og Per Christian Engdal. Dokumentet finnes også i engelsk utgave.

REVISJONSHISTORIE

Dato	Versjon	Beskrivelse	Forfatter
25.04.2023	1.0	Innledning, arkitektur, server-tjenester	Håkon Grimen
04.05.2023	1.9	2.utkast rapport, sensur gjennomført	Håkon Markhus
09.05.2023	3.0	Komponentdiagram og prosjektstruktur oppdatert. Informasjon om avhengigheter	Håkon Grimen
18.05.2023	4.0	Sensur kildekode	Håkon Markhus

INNHOLDSFORTEGNELSE

1 INNLEDNING

Dette dokumentet inneholder nødvendig dokumentasjon på hvordan web-applikasjonen som er bygget i denne bacheloroppgaven fungerer, hvordan arkitekturen er og andre relevante detaljer for prosjektet. Det inkluderer også noe informasjon om API-endepunkter og kode med veiledning for installasjon og bruk.

Dokumentet er laget som et støttedokument til oppgaven «Geppetto – fra Windows til Web», skrevet som en bacheloroppgave ved Høgskolen på Vestlandet, våren 2023.

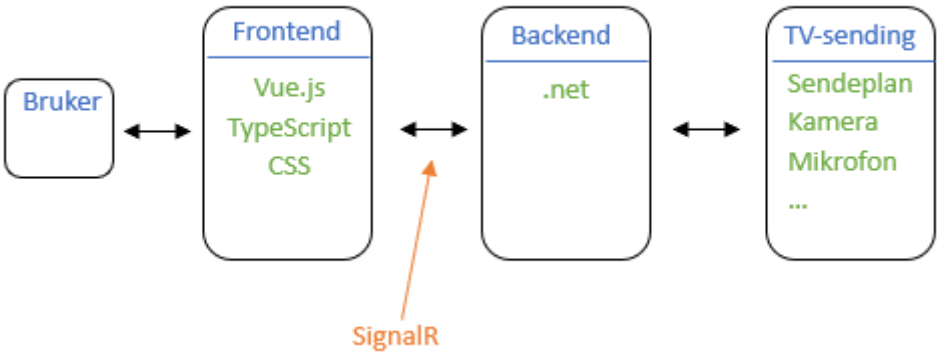
2 ARKITEKTUR

Denne løsningen er bygget som en web-applikasjon som tilbyr et brukergrensesnitt hvor brukeren utfører handlinger og får presentert informasjon. Systemet har kontakt med en backend-løsning via WebSocket-protokollen og SignalR (Microsoft, 2023). Backend-løsningen er en automasjonsbackend som utfører bestemte handlinger i forbindelse med kjøring av en TV-sending, og leverer informasjon tilbake til frontend-løsningen om status på denne. Blant noe av de som tilbys er en «puls», som gir status på flere forskjellige komponenter som backend har kontakt med. Denne informasjon brukes og presenteres i brukergrensesnittet som er bygget i denne oppgaven.

I en Vue-applikasjon (Vue.js, n.d.), som denne oppgaven er, foregår svært mye av logikken på frontend. Det er bygget opp som en SPA (Single Page Application) og bestående av flere SFC (Single File Component) som baserer seg på visse komponenter, som deretter fører til handlinger i backend. Backend-løsningen står for all videre kommunikasjon ut mot TV-studioet der en sending pågår.

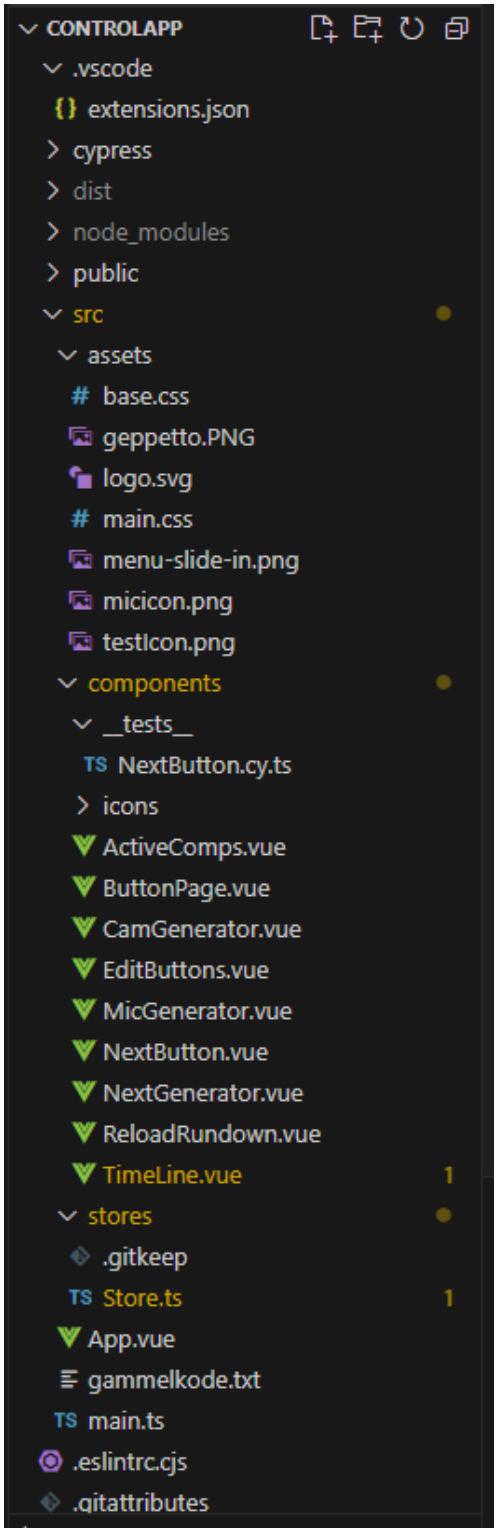
Løsningen presenterer en tidslinje, hvor dataen som fyller opp denne er hentet fra backend-løsningen. Denne dataen er lagret i en XML-fil som leses inn og behandles av nevnte backend. I web-applikasjonen kan man under en aktiv sending, hele tiden se status på hver enkelt historie som pågår ved å se på denne tidslinjen.

Brukergrensesnittet tilbyr også knapper som utfører handlinger i backend, f.eks å hoppe videre til neste historie. En slik handling vil vises i tidslinjen ved at viser statusen, hentet fra backend.



Figur 2.1 arkitekturnivåene i applikasjonen

3 PROSJEKTSTRUKTUR



Figur 3.1 Filstrukturen til prosjektet i VSCode

Figur 3.1 viser fil- og katalogstrukturen for dette prosjektet slik det er vist utviklingsverktøyet.

ActiveComponents.vue inneholder koden som viser navnet på innslaget som kjøres, samt tiden den har kjørt.

ButtonPage.vue inneholder koden som tegner opp knappen brukeren har lagt til på hovedsiden.

CamGenerator.vue og **MicGenerator.vue** har begge funksjonalitet som oppretter nevnte knapper.

EditButtons.vue styrer visning av slette-knapp på de brukergenererte knappene.

NextButton.vue styrer Next-knappen som hopper til neste innslag. Visning av selve knappen styres fra **NextGenerator.vue**.

ReloadRundown.vue generer og styrer funksjonalitet som lar brukeren restarte en sendeplan (rundown).

TimeLine.vue inneholder kjernefunksjonaliteten knytte til tidslinjen.

Store.ts er filen hvor Pinia-storen er implementert og definert. Her ligger mye av funksjonaliteten og tilstanden lagret. (Morote, 2019).

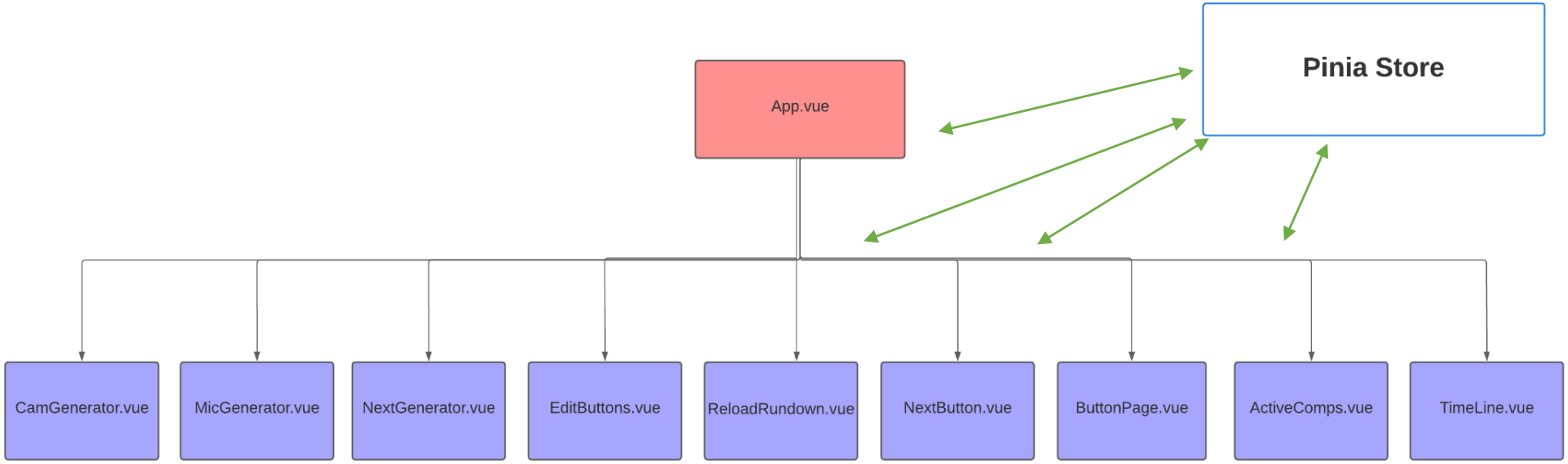
App.vue er selve roten i en Vue-applikasjon. Her importeres de andre komponentene og plasseres der det skal være. Her ligger også koden som tegner opp menyen i applikasjonen.

I **main.ts** og **main.css** finnes import av biblioteker som Vuetfiy og Pinia, samt nødvendig CSS-styling.

4 KLASSEDIAGRAM

Grunnet at utviklingen av prosjektet vårt har vært fokusert på frontend og er bygget som en Single-Page Application, eksisterer det ikke klare klasser som kan dokumenteres på tradisjonell form. Selv om TypeScript har støtte for klasser, er dette ikke direkte brukt. Vi har heller benyttet allerede eksisterende klasser, egendefinerte funksjoner og definert grensesnitt til bruk ved datasett som ikke skal behandles videre.

De forskjellige komponentene i Vue-applikasjonen har likevel et forhold som kan illustreres. Dette er vist i Figur 4.1.



Figur 4.1 Diagram som viser Vue-komponentene

Som vist i Figur 4.1, har en applikasjon i Vue av denne begrensede størrelsen, en ganske flat struktur. App.vue er forelder, eller roten, i selve applikasjonen og stedet hvor det importeres de andre komponentene. Komponentene vil da være barn av komponenten over, i dette tilfellet App.vue. I større applikasjoner vil det være flere nivåer nedover med barn/forelder-forhold.

I tillegg til dette er Pinia, som er hvor mye funksjonalitet og tilstandsdata lagres, være knyttet til komponentene via en import-setning. En komponent som er barn av et annet trenger ikke å gå via sin forelder for å kontakte Pinia, men kan gjøre det selv via et direkte kall. Dette er illustrert med piler fra Pinia-storen og ut til de forskjellige komponentene i figuren.

5 DATABASEMODELL

Dette kapittelet er utelatt da denne løsningen ikke har en database.

6 Server-tjenester

Oppdragsgiver sin automasjonsbackend tilbyr en rekke endepunkter som kan benyttes. Ettersom dette er interne systemer som benyttes av Vizrt, vil gruppen bare kort beskrive hovedfunksjonaliteten i de endepunktene som er benyttet. Endepunktene har andre navn internt, men har fått navn «Endepunkt *N*» i denne teksten.

Endepunkt 1
Dette er handlingen som trigger et hopp til neste del-historie. Ved trykk på «Next»-knappen i applikasjonen, sendes det et metodekall til backen med informasjon om å hoppe til neste enhet på kjøreplanen. Backend-løsning utfører så denne handlingen. Resultatet av denne vil være synlig i tidslinjen 1 sekund etter klikket og handlingen er utført.

Dette endepunktet tilbyr funksjonalitet som er kritisk for denne applikasjonen.

Endepunkt 2:
Handling som henter ut en sendeplan (i backend heter dette rundown). Denne sendeplanen er en XML-fil som først leses inn av Viz Mosart, som er selve automasjonsbackenden. Når Mosart har lest inn en slik, kan den igjen hentes ut av andre applikasjoner via dette endepunktet.

Vår applikasjon bruker denne til å hente ut sendeplanen ved oppstart av applikasjon. En rundown inneholder et antall historier, som igjen består av flere del-historier. Hver av disse historiene som hentes vil så danne en tidslinje i applikasjonen gruppen har utviklet. For hver del-historie bygges det progress-bars.

Endepunkt 3:

Handling som laster inn sendeplan på nytt og setter tiden til null.

Endepunkt 4:
Dette er en event som backend-løsningen tilbyr applikasjon å abonnere på. Når man har opprettet et abonnement på denne, vil den sende en samling med øyeblikks-informasjon, hvert eneste sekund. Det er blant annet informasjon om aktiv historie, ID, teller for antall sekunder i historie, osv.

Informasjonen og «pulsen» denne sender benyttes til å gi liv til tidslinjen når sendingen er aktiv.

7 SIKKERHET

Sikkerhet som er implementert i denne løsningen består generelt sett på sikkerhetsprinsipper, og ikke eksklusive sikkerhetsfunskjonalitet, da denne ikke er kritisk for kjøring av programmet. Den sikkerheten som er tilstede er blant annet at all informasjon som sendes til og fra Mosart går via WebSockets som bare tillater lokale IP-adresser. Dette kommer av CORS godkjenning (også kjent som «whitelisting» av adresser). Videre vil heller ikke hostserver for nettsiden kunne nås fra adresser utenfor et lokalt nettverk. Derfor vil brannmur og gode nettverks vaner være et viktig på det nettverket tjeneren kjører. Nettsiden i seg selv skal i utgangspunktet kunne kjøre ulike typer servere, men vil i komme via en https-server, som vil kjøre regelmessige sikkerhets oppdateringer, og oppsett av automatisk og regelmessig backup.

Ved en eventuell videreutvikling av produktet vil også pålogging mot Vizrt sine servere være nødvendig dersom webapplikasjonen skal brukes opp mot produksjon. Produktet vil også måtte validere brukere og enheter på en vitelistet måte, og derav varsle via satt opp kommunikasjonskanaler (epost, mobil 2-faktor autentisering), dersom ukjente enheter forsøker innlogging.

8 INSTALLASJON OG KJØRING

8.1 Avhengigheter og bibliotek

De viktigste avhengigheter og biblioteker som benyttes av denne web-applikasjon er følgende:

Vue.js (https://vuejs.org/):
Et progressivt JavaScript-rammeverk som er benyttet i utviklingen av web-applikasjonen

TypeScript (https://www.typescriptlang.org/)**:**
Supersett av JavaScript, som gir mer robust kode og typesikkerhet. Dette er benyttet sammen med Vue i denne applikasjonen.

Node.js (https://nodejs.org/en)**:**
Et run-time system for server- og nettverksapplikasjoner som kjører JavaScript-kode. Muliggjør kjøring av JavaScript-programmer på servere.

Vite (https://vitejs.dev/)**:**
Byggeverktøy som brukes for å sette sammen kode og filer til noe som kan kjøres.

Vuetify (https://vuetifyjs.com/en/)**:**
Komponentbibliotek for utvikling av brukergrensesnitt, med blant annet knapper, tidslinje og menyer.

Pinia (https://pinia.vuejs.org/):
Rammeverk som styrer lagring og tilstandshåndtering i en Vue-applikasjon

SignalR (https://dotnet.microsoft.com/en-us/apps/aspnet/signalr)**:**
Bibliotek som tillater realtime-kommunikasjon mellom server og klient. Benyttes mellom frontend og backend.

Node Package Manager, npm (https://www.npmjs.com/):
En pakke-manager for JavaScript, som blant annet er brukt av Node.js for installasjon av andre avhengigheter og bibliotek.

I tillegg til dette er Microsoft Visual Studio Code benyttet som utviklingsverktøy underveis i dette prosjektet. Programvaren kan hentes fra https://code.visualstudio.com/

En komplett oversikt over avhengigheter (her kalt for «dependencies») er vist i Figur 8.1 Utdrag fra filen "package.json". Her ser man blant annet versjonsnummer for de forskjellige avhengighetene.

```
},
"dependencies": {
  "@mdi/font": "^7.2.96",
  "@microsoft/signalr": "^7.0.4",
  "pinia": "^2.0.30",
  "vue": "^3.2.45",
  "vuetify": "^3.1.1"
},
"devDependencies": {
  "@fortawesome/fontawesome-free": "^6.3.0",
  "@rushstack/eslint-patch": "^1.1.4",
  "@types/node": "^18.11.12",
  "@vitejs/plugin-vue": "^4.0.0",
  "@vue/eslint-config-prettier": "^7.0.0",
  "@vue/eslint-config-typescript": "^11.0.0",
  "@vue/tsconfig": "^0.1.3",
  "cypress": "^12.0.2",
  "eslint": "^8.22.0",
  "eslint-plugin-cypress": "^2.12.1",
  "eslint-plugin-vue": "^9.3.0",
  "npm-run-all": "^4.1.5",
  "prettier": "^2.7.1",
  "start-server-and-test": "^1.15.2",
  "typescript": "~4.7.4",
  "vite": "^4.0.0",
  "vue-tsc": "^1.0.12"
},
```

Figur 8.1 Utdrag fra filen "package.json"

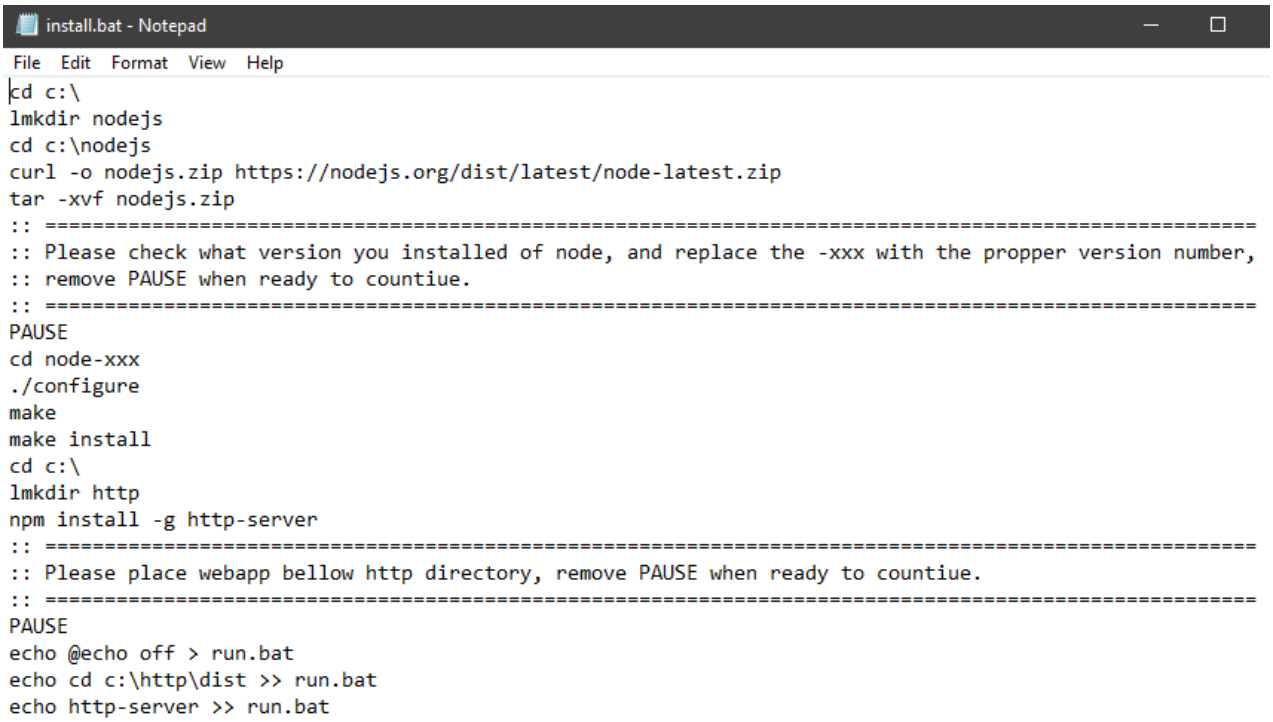
8.2 Installasjon og oppsett

Det er to deler denne installasjonen av web-applikasjonen består av. Først, må en installere en http-server, som kjører på det lokale nettverket. Dette kan være hvilket som helst server, men på Windows er det enkelt å installere ved å skrive følgende kommandoer inn i terminal, endre *xxx* til versjonsnummer som hentes. (Windows + R, skriv inn *cmd* i vinduet som popper opp, for å åpne terminalen):

```
cd c:\nodejs
curl -o nodejs.zip https://nodejs.org/dist/latest/node-latest.zip
```

tar -xvf nodejs.zip
cd node-*xxx*
./configure
make
make install
npm install -g http-server
cd *path**til**webapp*\dist
http-server

Før du kjører siste kommando, kopier mappen *dist* til der du vil starte din instans av serveren fra. Dette er også lagt inn i et *installScript.bat*, som er kjørbart ved nedlastning av denne web-applikasjonen, men også her må man være obs på å endre til riktig versjon av httpserver, etter dette kan en kjøre *run.bat* for startup av server.



Figur 8.2innhold i filen "Install.bat"

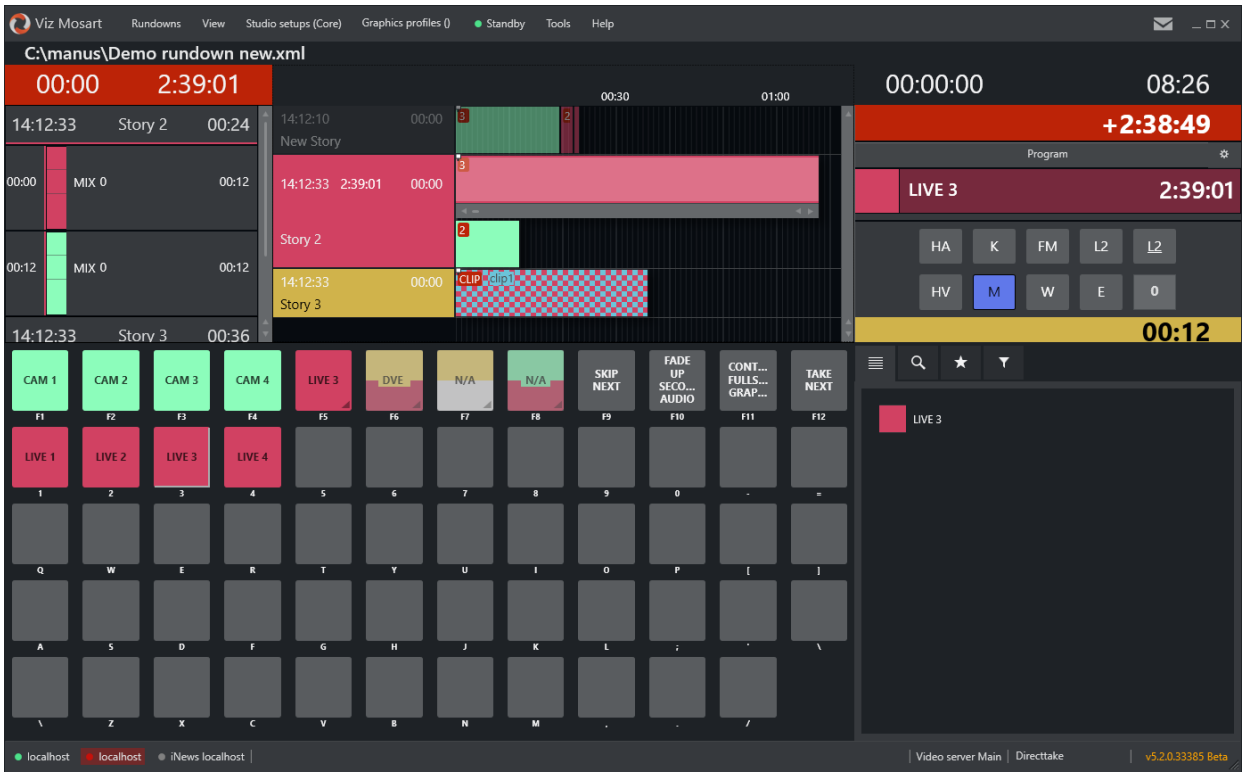
Skriv følgende inn i browser på enhet på samme nettverk.
http:/serverip:8080/
Nå kjører webappen uten backend på maskinen.

Steg 2: Installer Viz Mosart.

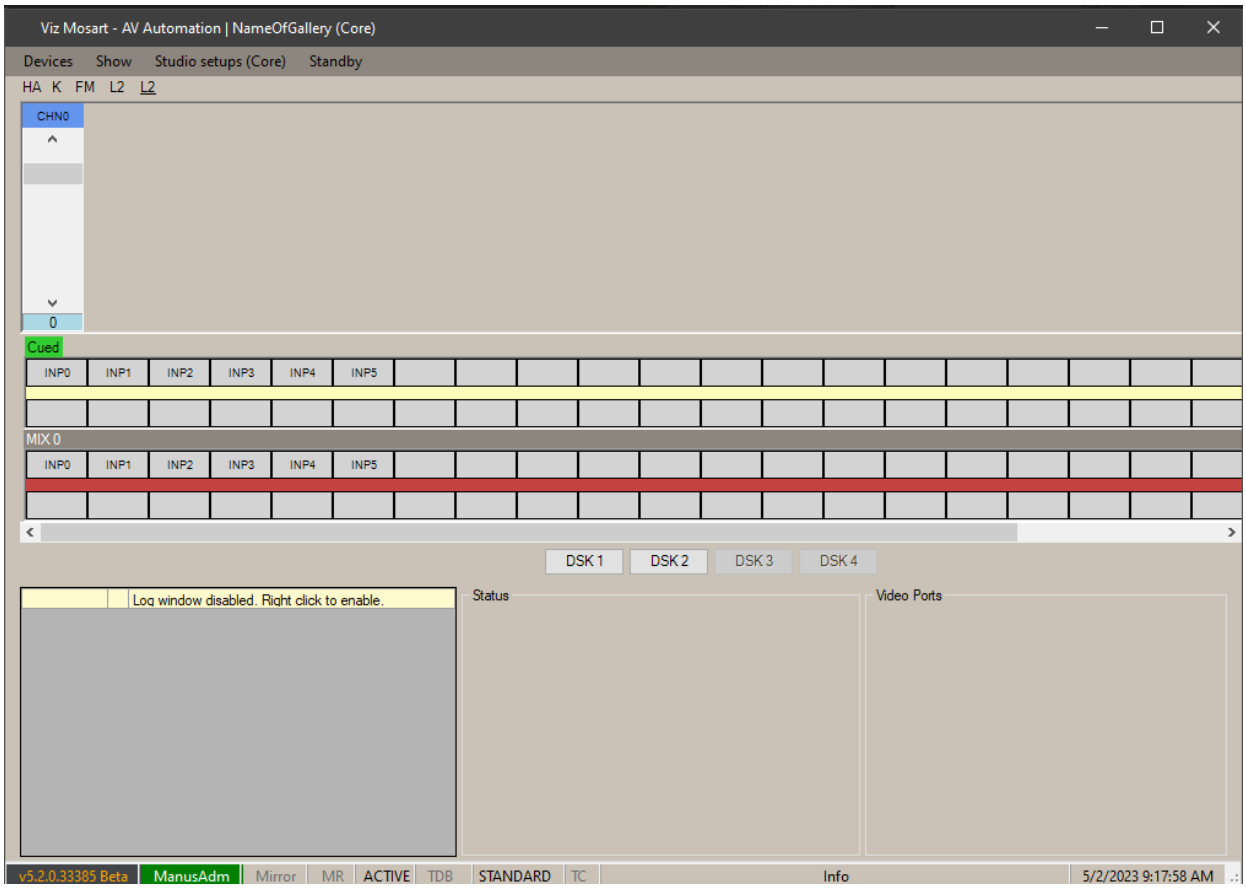
Dette steget er nødvendig for å få tilgang til all kontakt som har basefunksjonalitet i Mosart, som er backend-løsningen denne applikasjonen benytter seg av. Dermed kreves en avtale med Vizrt for å få dette til, noe som ikke er distruerbart av studentgruppen.
Ettersom disse pakkene ikke er offentlig tilgjengelig og ikke distribueres av gruppen, vil det ikke være mulig å kjøre web-applikasjonen med full funksjonalitet.

Dersom det er mulig å få tak i en installasjon av Mosart-suiten, kan man installere denne i henhold til instruksjonene der.

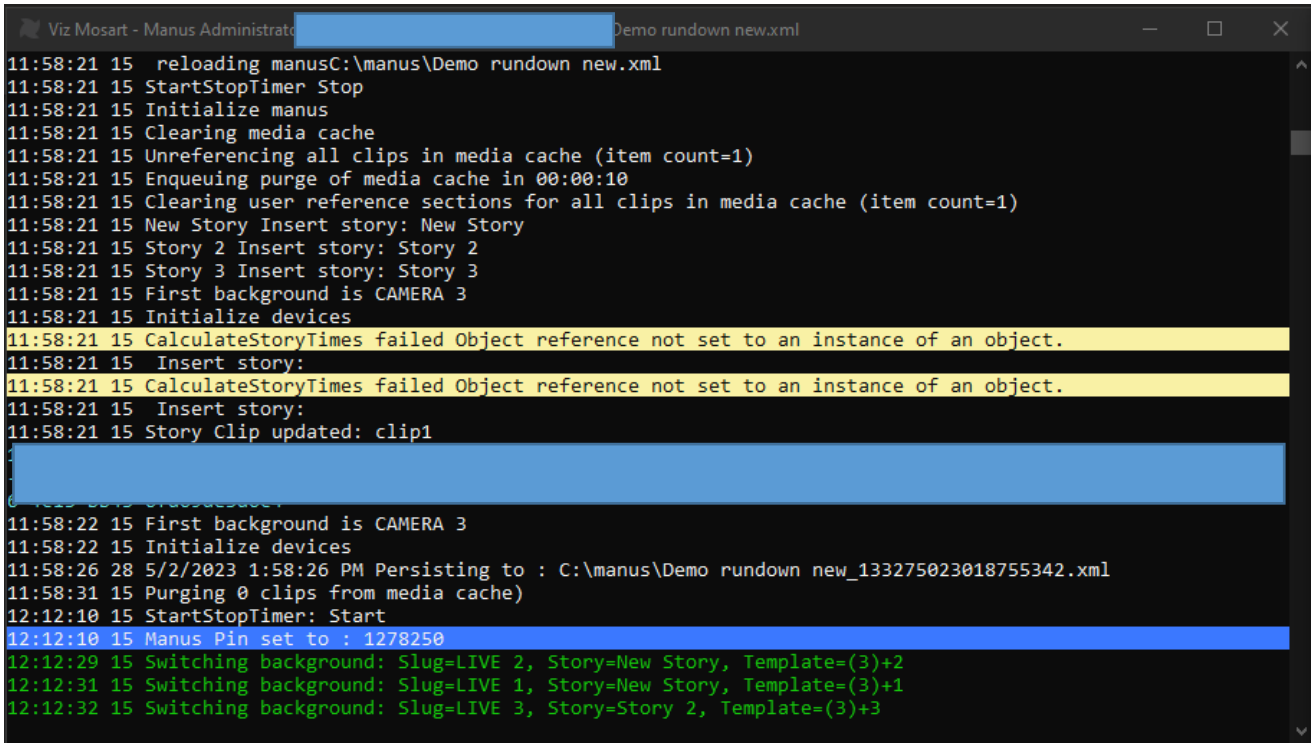
Deretter kjøres serveren (manus admin), GUI og AV Automation for kjøring av applikasjon.



Figur 8.3 Viz Mosart (produkt eies av Vizrt)



Figur 8.4 Viz Mosart - AV Automation. Produktet eies av Vizrt



Figur 8.5 Viz Mosart -Manus Administrator. Produktet eies av Vizrt

9 DOKUMENTASJON AV KILDEKODE

Kildekoden utviklet i dette prosjektet er oppdragsgivers eierskap og kan dermed ikke distribueres.

10 KONTINUERLIG INTEGRASJON OG TESTING

Ikke relevant.

11 REFERANSER

Microsoft, 2023. *Microsoft dot.net.* [Internett]
Available at: https://dotnet.microsoft.com/en-us/apps/aspnet/signalr

Morote, E. S. M., 2019. *Pinia Vue.JS.* [Internett]
Available at: https://pinia.vuejs.org/
[Funnet 27 April 2023].

Vue.js, n.d.. [Internett]
Available at: https://vuejs.org/guide/introduction.html
[Funnet 21 02 2023].