Emotion Monitor IDA Media Foundry

Realisaties

Bachelor in de Toegepaste Informatica
keuzerichting AI

Ruben Lievens

Academiejaar 2020-2021

Campus Geel, Kleinhoefstraat 4, BE-2440 Geel

INHOUDSTAFEL

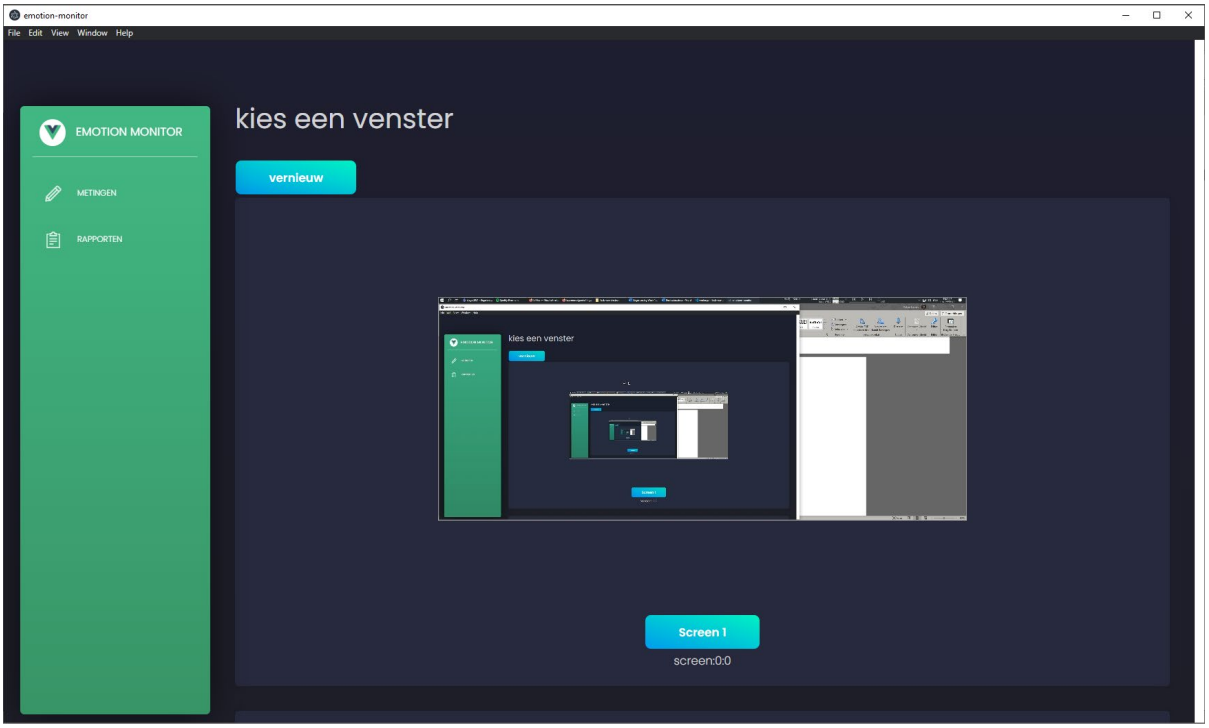
1 INLEIDING

Ik heb van 1 maart tot 27 mei bij IDA Media Foundry stage gelopen. Daar heb ik de gedurende de hele periode zelfstandig en autonoom aan een applicatie gewerkt die het makkelijker moet maken om de emotionele staat van deelnemers van een online meeting meetbaar te maken.

Voor de applicatie heb ik natuurlijk een aantal dingen gerealiseerd, dewelke ik zal toelichten in dit document.

2 VENSTER SELECTIE SCHERM

Het eerste wat men ziet als men de applicatie opent is het venster “selectie scherm”, op dit scherm krijgt men een lijst van alle gedetecteerde schermen op je computer. Met een live feed van elk scherm kan je dan kiezen op welk scherm je de vergadering op gaat zetten zodat de AI van start kan gaan. Als je achteraf een scherm zou aansluiten kan je de schermen opnieuw ophalen met de “vernieuw” knop.

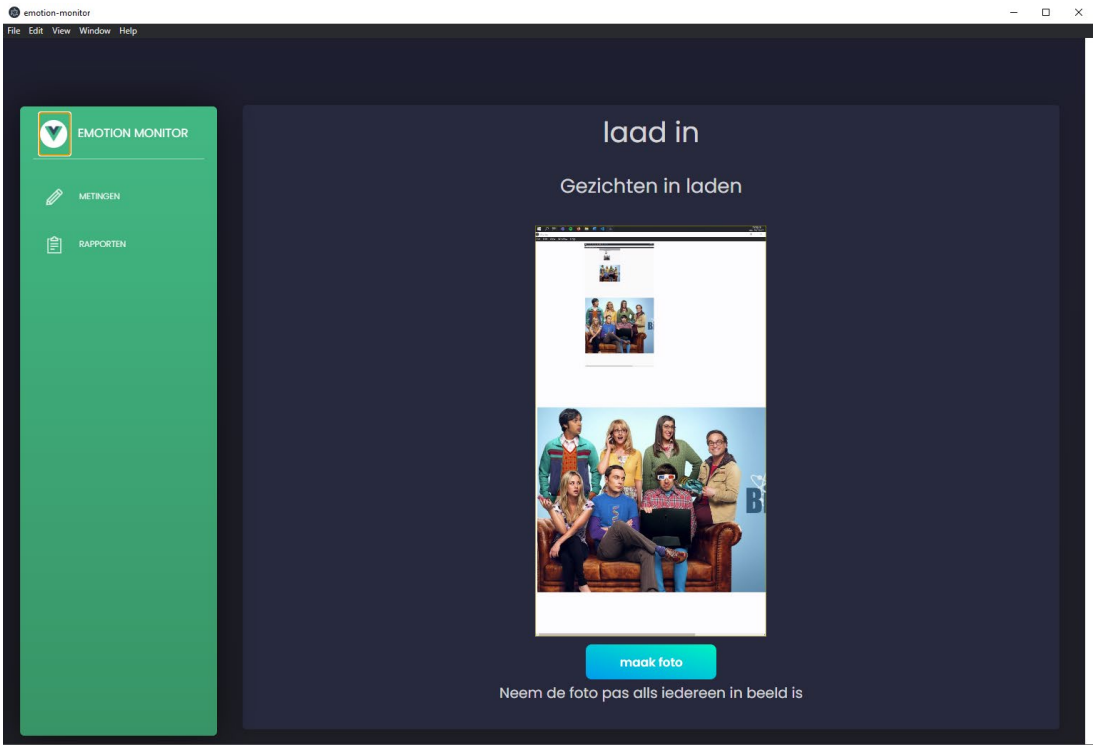


Hier wordt de code getoond voor het ophalen en tonen van een scherm:
Er wordt ook een webcam opgezet met een Chrome Media Source. Eens dat die klaar is wordt die opgezet en krijgt de gebruiker de preview te zien.

```
async Preview(width, height)  {
console.log(this.obj)
  let cam = this.$refs.cam;
  cam.width = width;
  cam.height = height;

  const stream = await navigator.mediaDevices.getUserMedia({
      audio: false,
video: {
  mandatory: {
    chromeMediaSource: 'desktop',
    chromeMediaSourceId: this.obj.id
  },


}
  });
  cam.srcObject = stream;

  return new Promise((resolve) => {
    cam.onloadedmetadata = () => {
      resolve(cam);
    };
  });
}
```

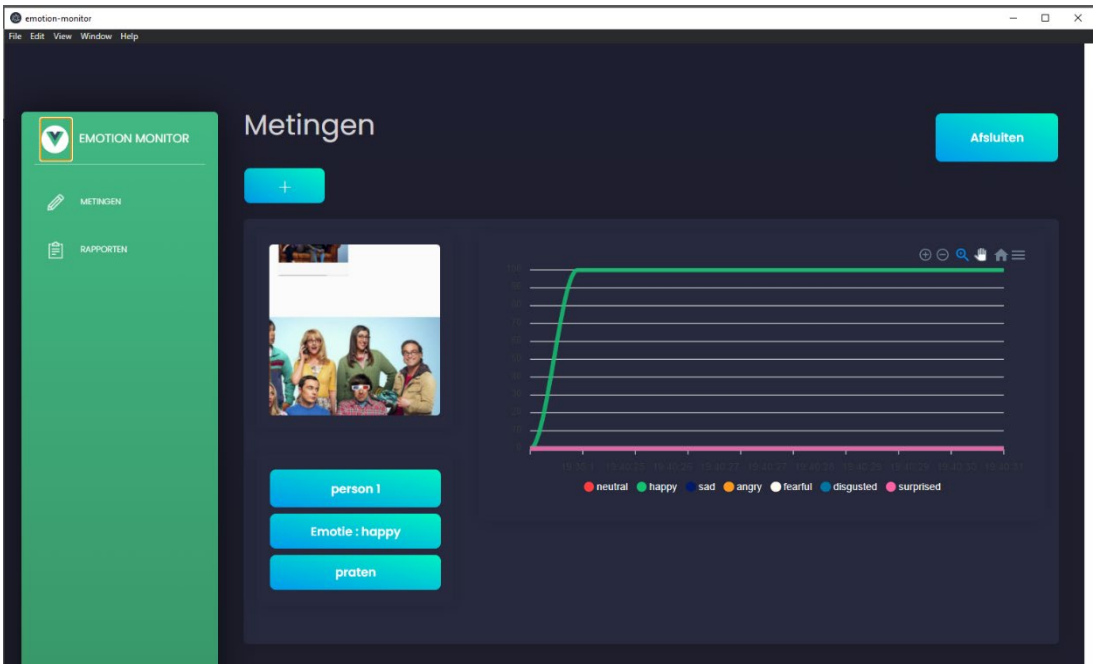
3 LAAD IN SCHERM

Als het laad in scherm getoond wordt, dan wordt er in de achtergrond de AI ingeladen die later de emotiedetectie gaat doen. In deze stap wordt er ook gevraagd om een foto te trekken van alle deelnemers. Deze foto gaat dan in de back-end gebruikt worden om de verschillende deelnemers te herkennen.



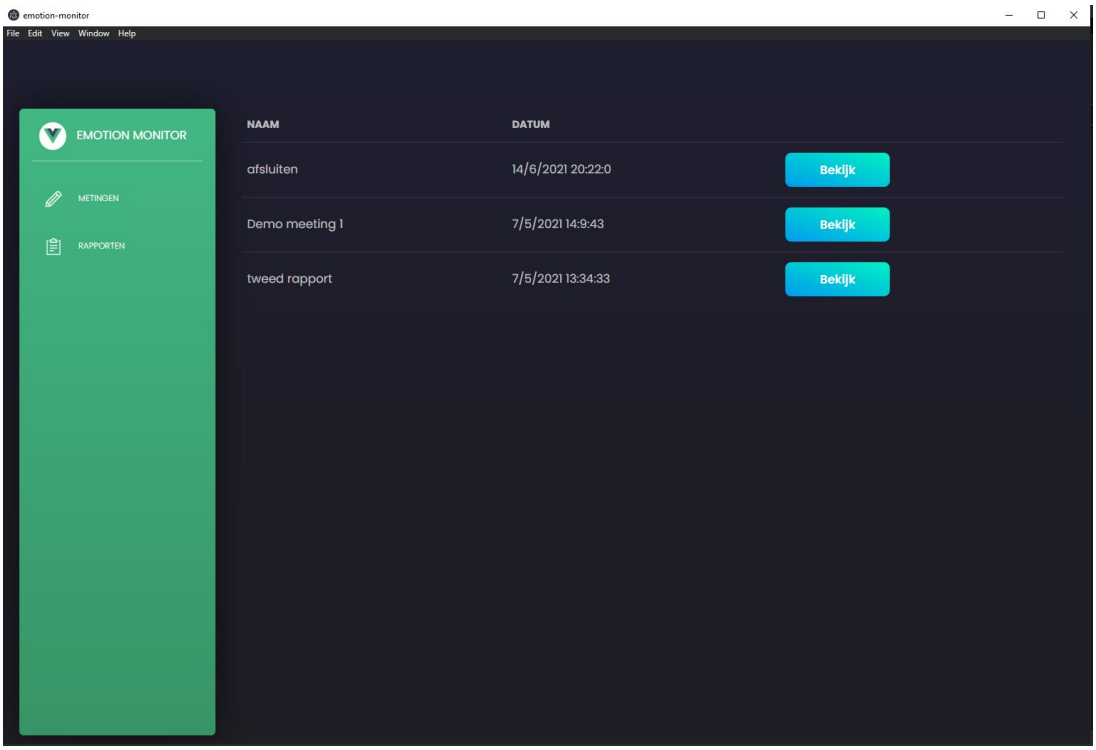
4 METINGEN SCHERM

Nadat het laad in scherm is uitgevoerd komt men vervolgens terecht op het metingen scherm. Hier gaat men de real time resultaten zien die de AI doorgeeft. Het moment dat de nieuwe metingen worden ontvangen, worden de laatste gemeten emoties getoond en deze metingen worden dan ook toegevoegd aan de grafiek. Zo wordt de grafiek opgebouwd met data van continue metingen. Op deze pagina kan men dan ook notities toe voegen en de namen van de verschillende deelnemers aanpassen. Uiteindelijk kan men hier ook de meeting afsluiten en de gegevens opslaan in een rapport dat men later kunt herbekijken. Hiermee kan men dan een analyse doen van de metingen op elk gewenst tijdstip van de totale meting, men kan een selectie maken van metingen of de gehele grafiek terug herbekijken.



5 RAPPORT SELECTIE SCHERM

Op het rapport selectie scherm worden alle opgeslagen rapporten getoond, deze lijst wordt opgehaald van een folder in het project met de naam /rapporten/. Om een rapport in te kijken, selecteer je de bekijk knop van het gewenste rapport, het rapport wordt dan geladen in het rapport scherm.



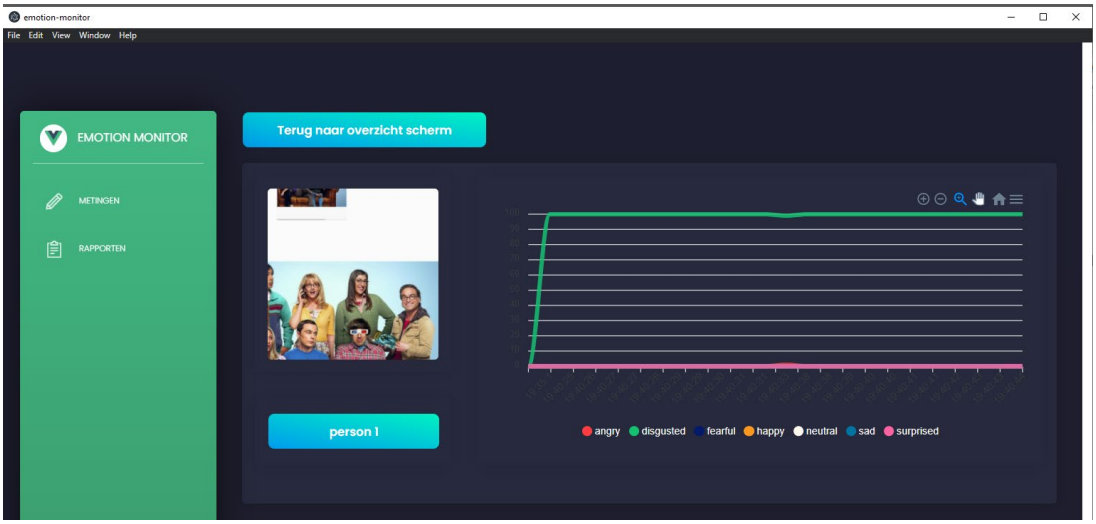
De volgende code haalt de verschillende bestanden op van de map:

Eerst wordt het pad naar de rapporten opgezet en toegewezen aan de path constante. Vervolgens wordt de folder uitgelezen met hulp van het node.js fs pakket. Als alle bestanden zijn ingeladen wordt het pakket terug gebruikt om de aanmaak datum van de bestanden op te halen. Als laatste worden de datums toegevoegd aan de bestand namen en het resultaat word vervolgens aan de lijst toegevoegd.

```
   async Getfiles() {
      const path = process.cwd() + "/Raporten/";
      console.log(path);
      const bestanden = fs.readdirSync(path);
      const result= []
      bestanden.forEach((item, index, base) => {
        console.log(item)
        console.log(path+item)
         const stat =  fs.statSync(path+item, function(err  , stats){
            console.log(stats)
            result.push("test")
          })
        console.log(stat)
        const naam = item.replace('.json', '')
        const today= new Date(stat.birthtime)
       var date = today.getDate()+'/'+(today.getMonth()+1)+'/'+today.g
etFullYear();

        var time = today.getHours() + ":" + today.getMinutes() + ":" +
 today.getSeconds();
        const object = { naam: naam, date:date +" "+ time, filename: i
tem}
        result.push(object)
        console.log(index)
        console.log(base.length)
         if(index === base.length-1)
         {
           this.lijst = result
         }
      });

      //console.log(bestanden);

    },
```

6 RAPPORT SCHERM

Op het rapport scherm kan je de data zien van een afgelopen meeting



De volgende code toont hoe de data van het rapport wordt opgehaald:

Als eerste wordt het pad terug gedeclareerd in de constante path, vervolgens wordt de naam van het bestand opgehaald en dit met de path constante samengevoegd om zo het pad naar het gekozen bestand te generen. Als laatste wordt de data van het bestand opgehaald en ingelezen.

```
   async mounted() {
         const path = process.cwd() + "/Raporten/";
      const file = this.$store.getters["Python/GetRapport"];
      //console.log(file)

        const filepath = path+file
           const  testt = await fs.readFileSync(filepath, 'utf-8',
           function (err,data) {
  if (err) {
    return console.log(err);
  }

return data
})
const waarden = await JSON.parse( testt)
this.metingen = waarden.metingen

this.Notes= waarden.notes


}
```