

# Final documentation

[Team ChargePoint]

Teamleden:

1. Jens Geenen (IOT)
2. Reno Goeyvaerts (CSS)
3. Bryan Van Herck (AI)
4. Milan Keersmaekers (APP)
5. Seppe Meeus (APP)

## Contents

<b>1</b>	<b>INLEIDING.....</b>	<b>6</b>
<b>2</b>	<b>WAT WORDT ER VERWACHT .....</b>	<b>7</b>
<b>3</b>	<b>APP .....</b>	<b>8</b>
<b>3.1</b>	<b>Pagina's.....</b>	<b>8</b>
3.1.1	Welcome.....	8
3.1.2	Sign in.....	10
3.1.3	Sign up.....	12
3.1.4	Account .....	15
3.1.5	Chat.....	24
<b>4</b>	<b>CLOUD .....</b>	<b>32</b>
<b>4.1</b>	<b>AWS-topologie.....</b>	<b>32</b>
4.1.1	AWS Cognito .....	33
4.1.2	AWS elastic Beanstalk .....	33
4.1.3	AWS EC2 .....	33
4.1.4	AWS api-gateway (rest api).....	33
4.1.5	AWS api-gateway (websocket).....	33
4.1.6	AWS lambda.....	34
4.1.7	AWS S3 .....	34
4.1.8	AWS documentdb .....	34
4.1.9	AWS Cloudwatch .....	34
4.1.10	AWS elastic container registry .....	34
4.1.11	AWS VPC .....	35
<b>4.2</b>	<b>Pipeline.....</b>	<b>35</b>
<b>4.3</b>	<b>IAC .....</b>	<b>36</b>
<b>5</b>	<b>AI .....</b>	<b>37</b>
<b>5.1</b>	<b>Inhoud.....</b>	<b>37</b>
<b>5.2</b>	<b>Gebruikte technieken.....</b>	<b>37</b>
5.2.1	OpenCV .....	37
5.2.2	Yolov8 (You Only Look Once) .....	37
5.2.3	Documentdb .....	37
5.2.4	AWS S3 .....	38
5.2.5	Werking .....	38
<b>6</b>	<b>IOT .....</b>	<b>39</b>
<b>6.1</b>	<b>Inhoud.....</b>	<b>39</b>
<b>6.2</b>	<b>Setup .....</b>	<b>39</b>
6.2.1	GoPro Hero 4 .....	39
6.2.2	Raspberry pi 3B .....	39
6.2.3	Code .....	39
6.2.4	Schematische voorstelling.....	40
<b>7</b>	<b>BESLUIT.....</b>	<b>41</b>
<b>8</b>	<b>BRONNEN .....</b>	<b>42</b>

---

## Figurenlijst

Figuur 1: Welcome (Desktop) .....	8
Figuur 2: Welcome(Mobile).....	9
Figuur 3: Sign in (Desktop) .....	10
Figuur 4: Sign in (Mobile).....	10
Figuur 5: Sign up (Desktop) .....	12
Figuur 6: Sign up (Mobile).....	12
Figuur 7: Sign up EULA (Desktop) .....	13
Figuur 8: Sign up EULA (Mobile).....	13
Figuur 9: Account (Desktop) .....	15
Figuur 10: Account (Mobile).....	15
Figuur 11: Home (Desktop) .....	17
Figuur 12: Home (Mobile).....	17
Figuur 13: Queue (Desktop) .....	19
Figuur 14: Queue (Mobile).....	19
Figuur 15: Queue (Desktop) .....	20
Figuur 16: Queue (Mobile).....	20
Figuur 17: Queue (Desktop) .....	21
Figuur 18: Queue (Mobile).....	21
Figuur 19: Chat (Desktop).....	24
Figuur 20: Chat (Mobile) .....	24
Figuur 21: Knop om te reageren op een bericht (Desktop).....	25
Figuur 22: Reageren op een bericht (Mobile) .....	25
Figuur 23: Reactie op een bericht (Mobile) .....	26
Figuur 24: Confirmatie voor verzoek om te veranderen van plaats in queue (Mobile) .....	26
Figuur 25: Verzoek om te veranderen van plaats in queue (Mobile).....	27
Figuur 26: Afgewezen verzoek (Mobile) .....	27
Figuur 27: Geaccepteerd verzoek (Mobile) .....	28
Figuur 28: Reageren op verzoek (Mobile).....	28
Figuur 29: Reageren op verzoek (Mobile).....	29
Figuur 30: Bericht na het afwijzen of accepteren (Mobile).....	29
Figuur 31: Topologie .....	32
Figuur 32: pipeline flow.....	35
Figuur 33: Discord notifications .....	35
Figuur 34: IOT schema .....	40

---



## **1 INLEIDING**

We kregen de opdracht van Elision in Hasselt om de manier waarop zij hun oplaadpalen gebruiken te optimaliseren door middel van een app.

In dit document gaan wij toelichten hoe de app eruit zal zien en hoe deze zal functioneren.

---

## **2 WAT WORDT ER VERWACHT**

Het probleem op de Corda campus is dat er te weinig oplaadpalen zijn voor het aantal werknemers en dat mensen hun auto aan de laadpalen laten staan ook al is hun auto volledig opgeladen.

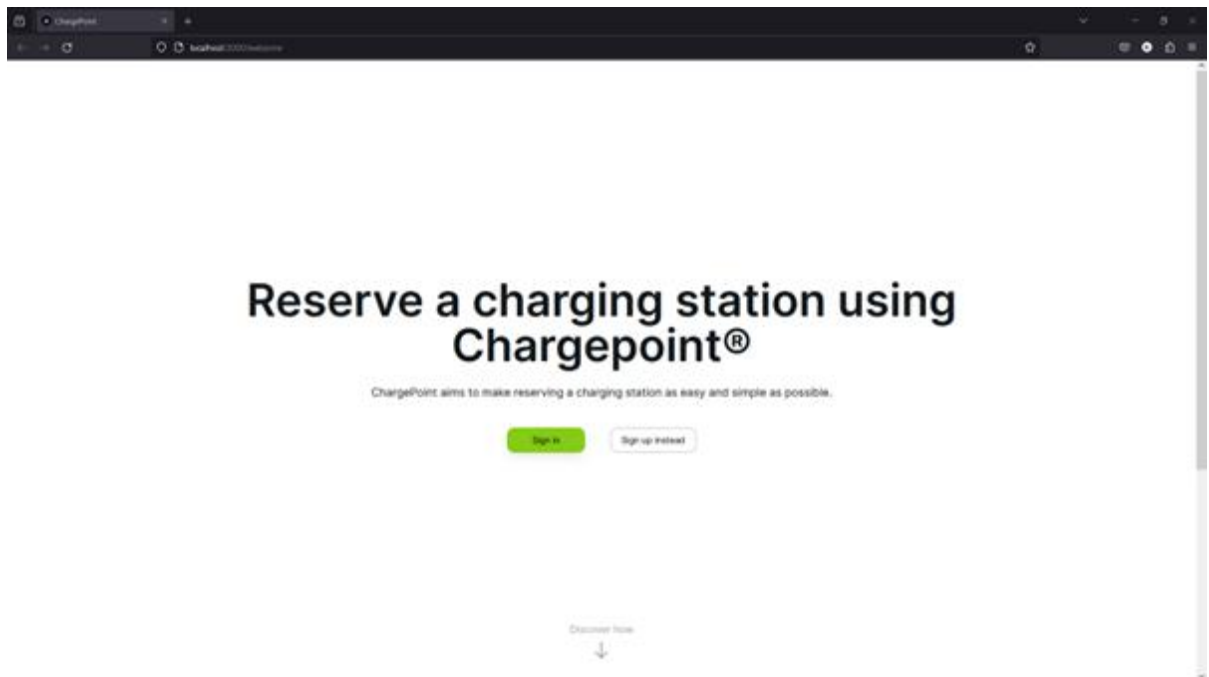
Ze hebben ons de opdracht gegeven om een app te maken waarbij mensen die willen opladen zichzelf in de wachtrij kunnen zetten en een melding/berichtje krijgen op het moment dat er een laadpaal vrijkomt.

## 3 APP

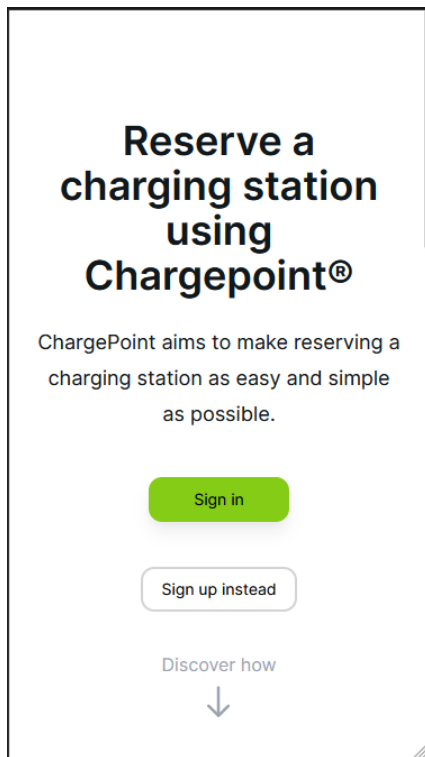
### 3.1 Pagina's

Hier worden de verschillende pagina's beschreven. Eerst wordt het design op zowel mobile als PC getoond. Vervolgens wordt elke component van de pagina individueel besproken.

#### 3.1.1 Welcome



*Figuur 1: Welcome (Desktop)*



*Figuur 2: Welcome (Mobile)*

#### 3.1.1.1 Sign in knop

- Doel: De gebruiker naar de sign in pagina brengen.
- On click: Herleid naar sign in pagina.
- On hover: Toont hover state (enkel op desktop).

#### 3.1.1.2 Sign up knop

- Doel: De gebruiker naar de sign up pagina brengen.
  - On click: Herleid naar sign up pagina.
  - On hover: Toont hover state (enkel op desktop).
-

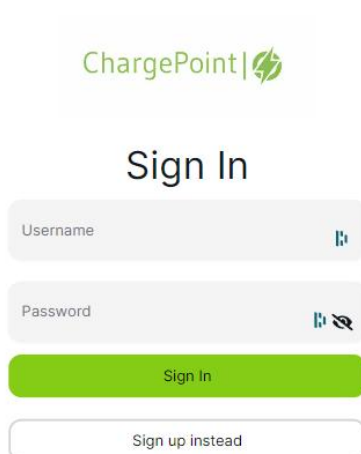


### 3.1.2 Sign in



The desktop sign-in form is a compact, rectangular layout. At the top left is the ChargePoint logo. To its right, the text "Sign In" is centered. Below this, there are two input fields: "Username" and "Password", each with a small icon on the right. A prominent green "Sign In" button is positioned below the password field, and a smaller, rounded "Sign up instead" button is at the bottom.

*Figuur 3: Sign in (Desktop)*



The mobile sign-in form is designed for a smaller screen. It features the ChargePoint logo at the top. Below it, the "Sign In" text is centered in a larger font. The "Username" and "Password" input fields are stacked vertically, each with a small icon on the right. A large green "Sign In" button is centered below the password field, and a rounded "Sign up instead" button is at the bottom.

*Figuur 4: Sign in (Mobile)*

---

#### 3.1.2.1 Username input veld

- Doel: De gebruiker de mogelijkheid geven om zijn gebruikersnaam in te geven.
- On click: De gebruiker kan starten met typen in het veld.

#### 3.1.2.2 Password input veld

- Doel: De gebruiker de mogelijkheid geven om zijn wachtwoord in te geven.
- On click: De gebruiker kan starten met typen in het veld.
- On click van het oogje: Toont of verbergt het wachtwoord.

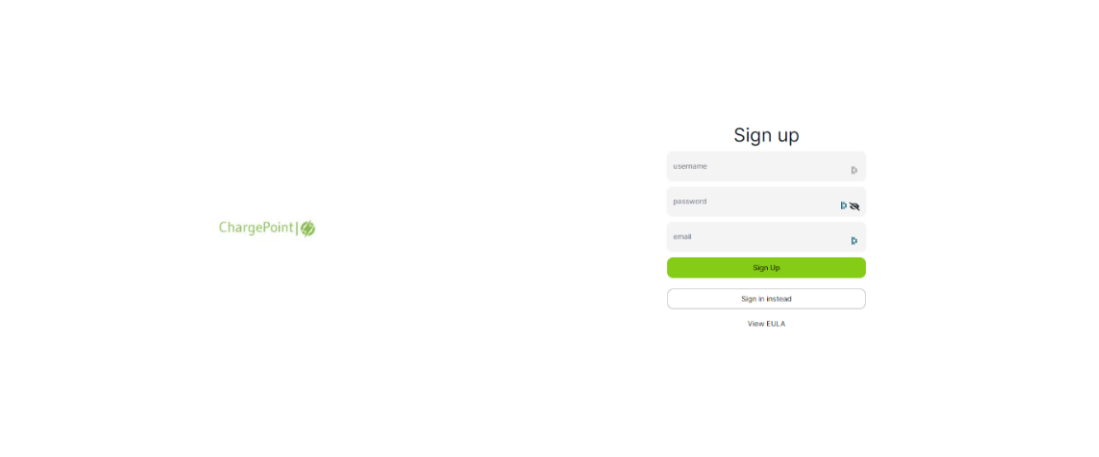
#### 3.1.2.3 Sign in knop

- Doel: De gebruiker inloggen via AWS Cognito. Vervolgens naar de homepagina brengen.
- On click: Logt de gebruiker in en herleid naar homepagina.
- On hover: Toont hover state (enkel op desktop).

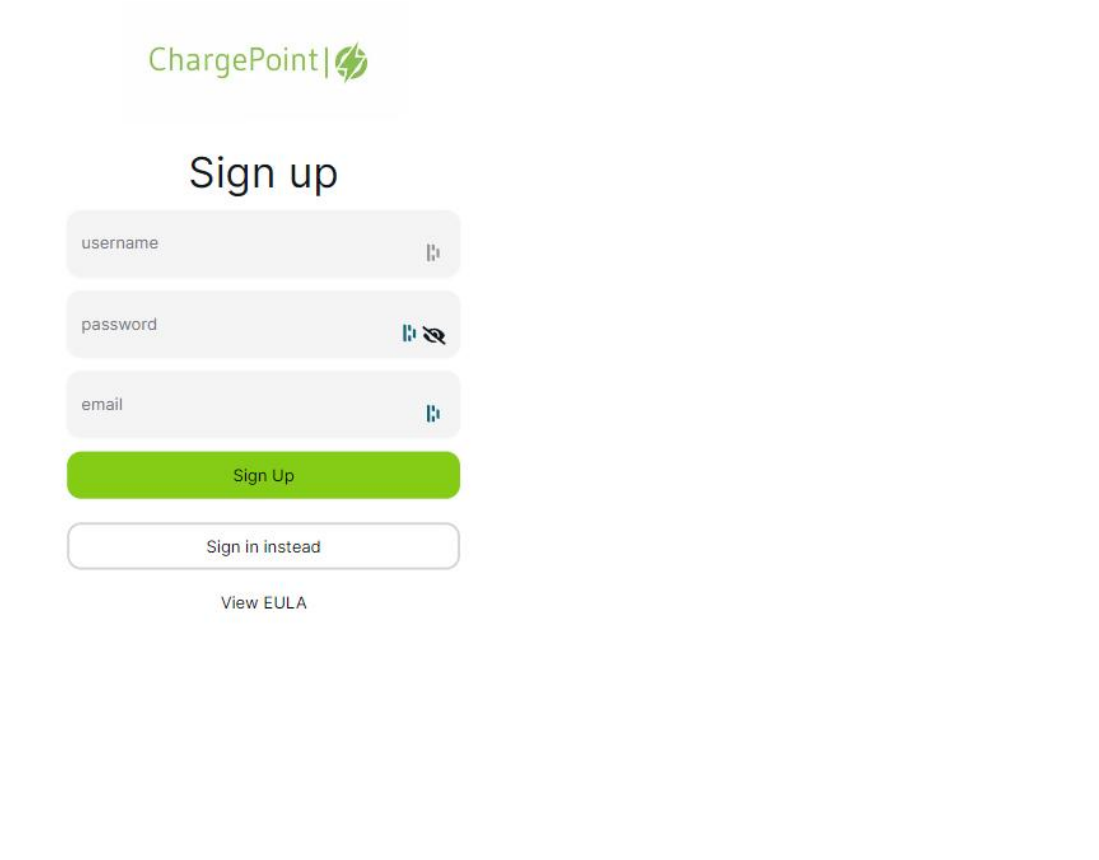
#### 3.1.2.4 Sign up instead knop

- Doel: Als de gebruiker toch niet wil inloggen, of nog geen account heeft, hun de mogelijk geven om te registreren.
  - On click: Leidt de gebruiker naar de sign up pagina.
  - On hover: Toont hover state (enkel op desktop).
-

### 3.1.3 Sign up

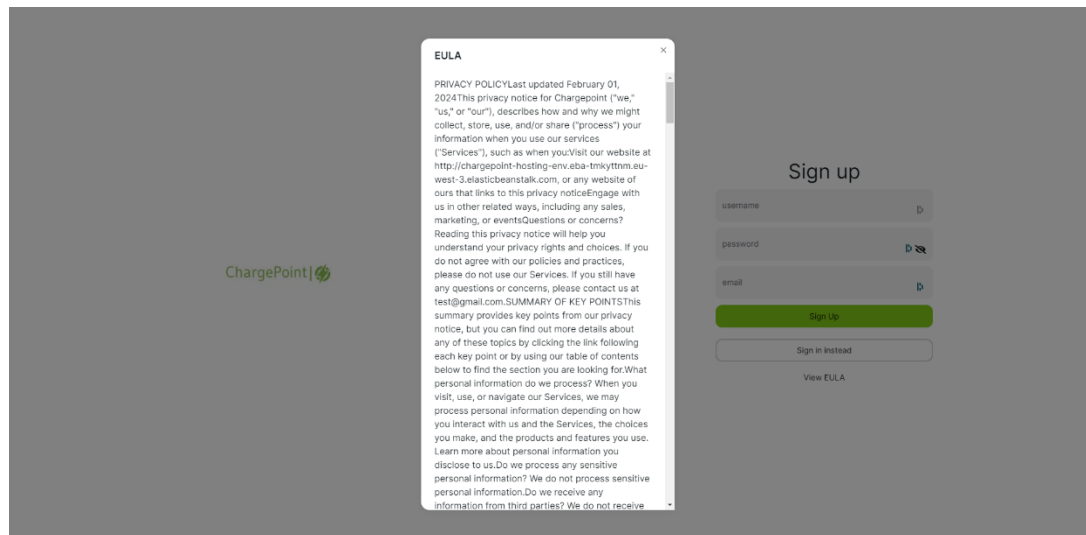


*Figuur 5: Sign up (Desktop)*

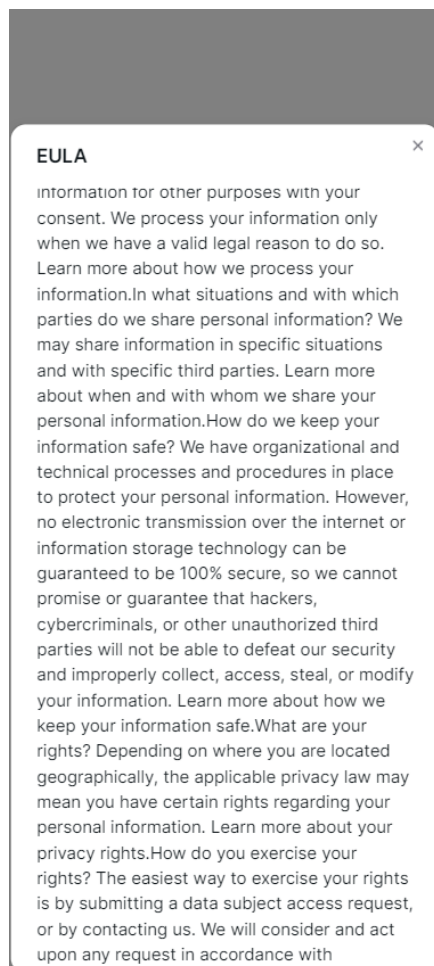


*Figuur 6: Sign up (Mobile)*

---



*Figuur 7: Sign up EULA (Desktop)*



*Figuur 8: Sign up EULA (Mobile).*

#### 3.1.3.1 Username input veld

- Doel: De gebruiker de mogelijkheid geven om zijn gebruikersnaam in te geven.
- On click: De gebruiker kan starten met typen in het veld.

#### 3.1.3.2 Password input veld

- Doel: De gebruiker de mogelijkheid geven om zijn wachtwoord in te geven.
- On click: De gebruiker kan starten met typen in het veld.
- On click van het oogje: Toont of verbergt het wachtwoord.

#### 3.1.3.3 Email input veld

- Doel: De gebruiker de mogelijkheid geven om zijn email in te geven.
- On click: De gebruiker kan starten met typen in het veld.

#### 3.1.3.4 Sign up knop

- Doel: De gebruiker registreren via AWS Cognito. Vervolgens moet de gebruiker zijn email confirmeren en opnieuw inloggen.
- On click: Registreert de gebruiker.
- On hover (enkel op desktop): Toont hover state.

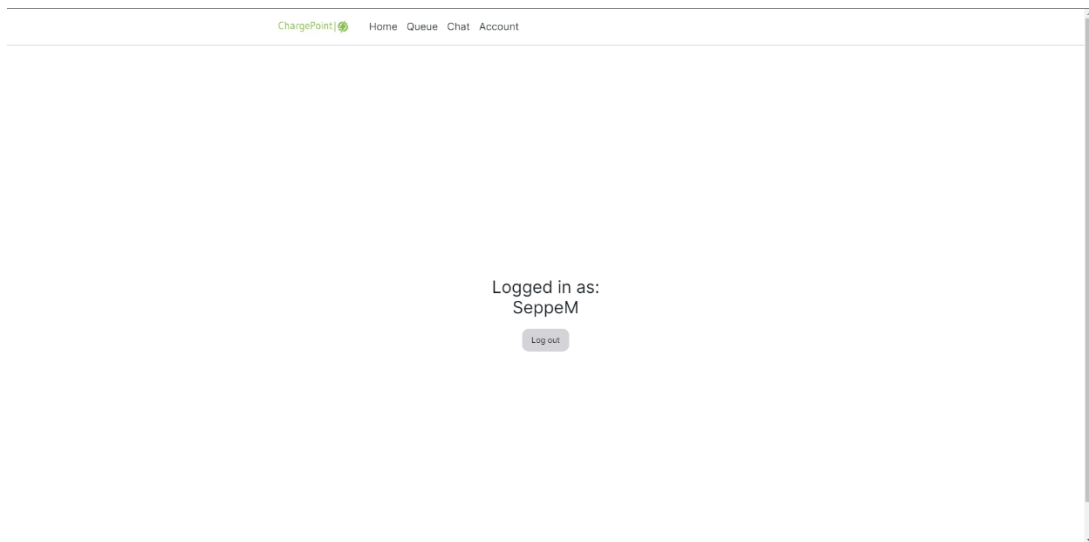
#### 3.1.3.5 Sign in instead knop

- Doel: Als de gebruiker toch niet wil registreren, of al een account heeft, hun de mogelijk geven om in te loggen.
- On click: Leidt de gebruiker naar de sign in pagina.
- On hover (enkel op desktop): Toont hover state.

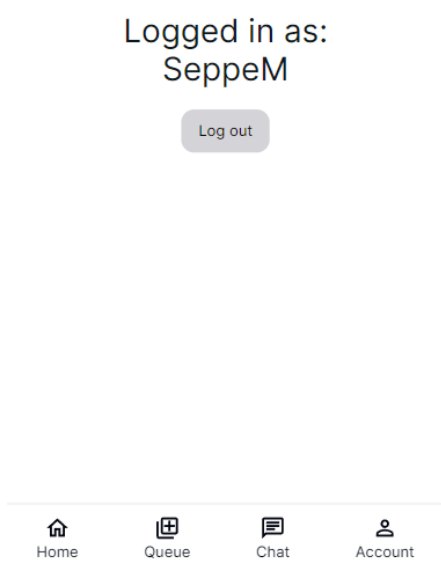
#### 3.1.3.6 View EULA knop

- Doel: De gebruiker de mogelijkheid geven om de EULA van de applicatie te bekijken.
  - On click: Opent een modal waarop de EULA getoond wordt.
  - On hover (enkel op desktop): Toont hover state.
-

### 3.1.4 Account



*Figuur 9: Account (Desktop)*



*Figuur 10: Account (Mobile)*

#### 3.1.4.1 Logged in as tekst

- Doel: Duidelijk maken aan de gebruiker als wie hij is ingelogd.

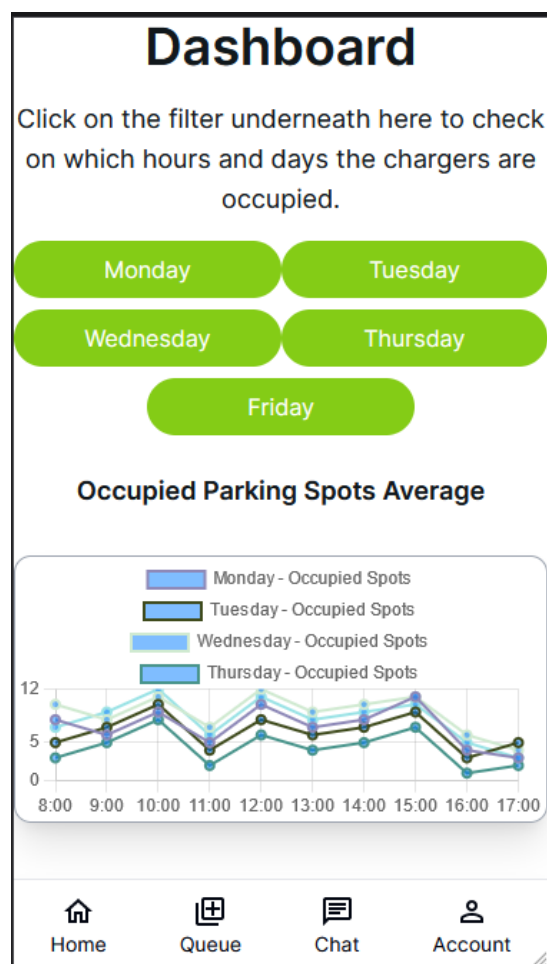
#### 3.1.4.2 Log out knop

- Doel: De gebruiker de mogelijkheid geven om uit te loggen.
- On click: De gebruiker via AWS Cognito uitloggen en herleiden naar de welcome pagina.
- On hover (enkel op desktop): Toont hover state.

### 3.1.5 Home



Figuur 11: Home (Desktop)



Figuur 12: Home (Mobile)



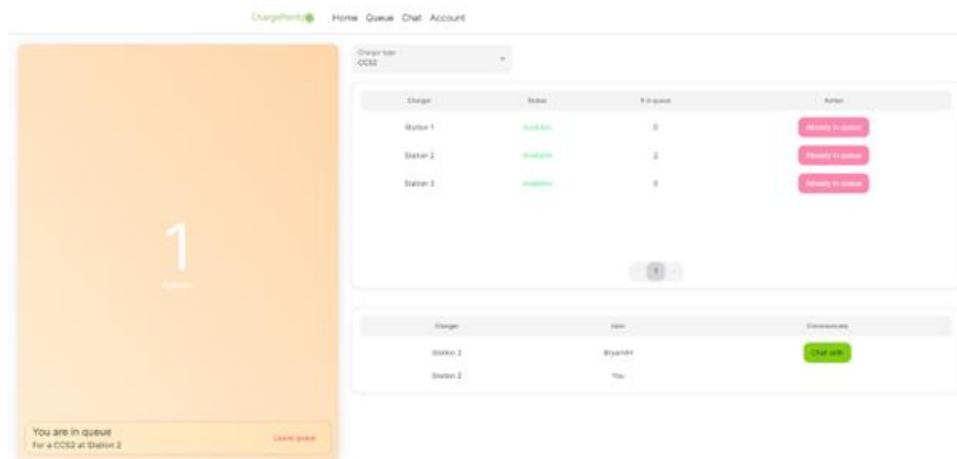
#### 3.1.5.1 Toggle dag knop

- Doel: Gebruikers de mogelijkheid geven een dag wel of niet te tonen op de grafiek.
- On click: Verbergt of toont de data voor de dag.
- On hover (enkel op desktop): Toont hover state.

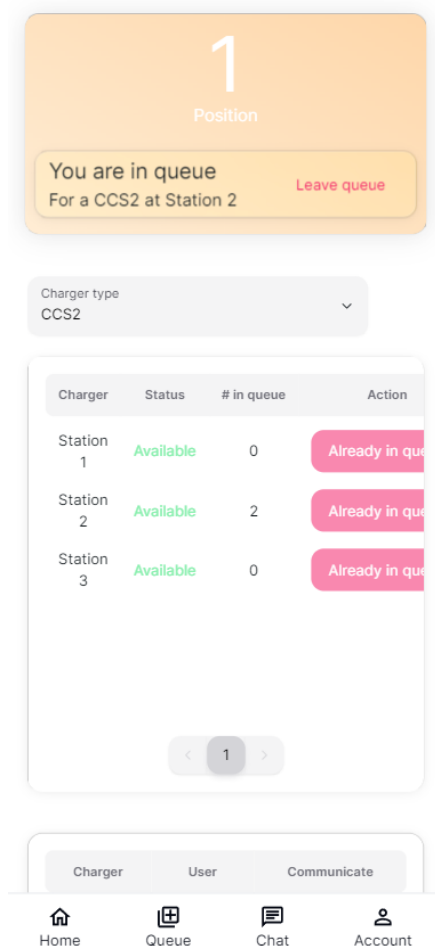
#### 3.1.5.2 Grafiek

- Doel: Gebruikers informatie bieden over het aantal ingenomen plaatsen over de dag heen.

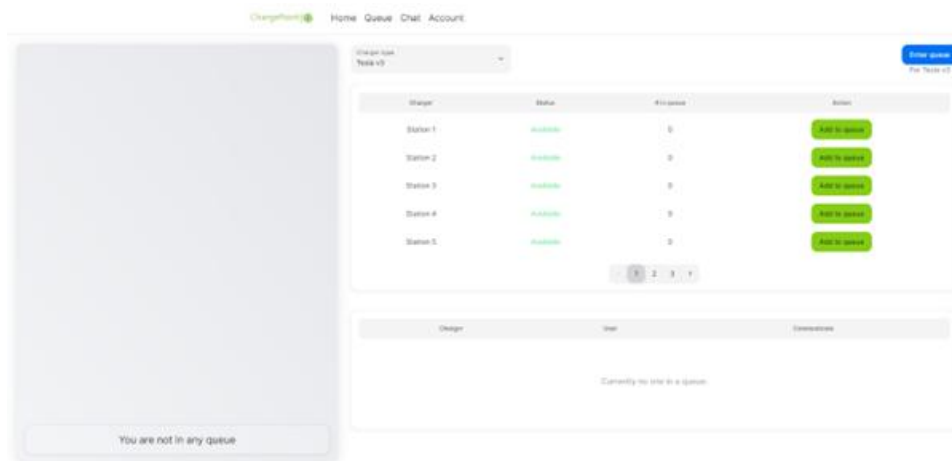
### 3.1.6 Queue



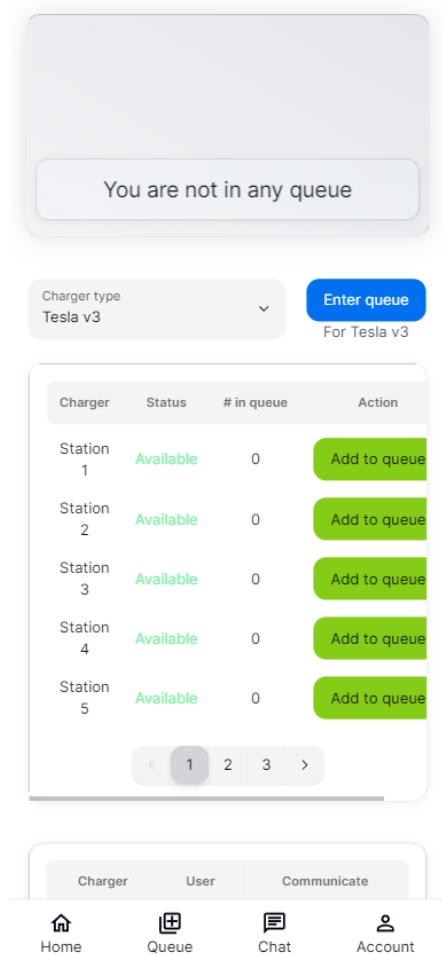
Figuur 13: Queue (Desktop)



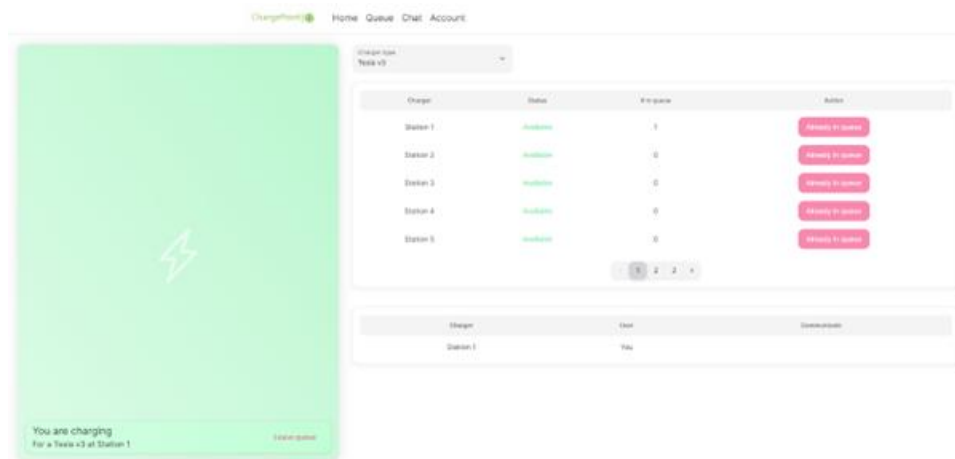
Figuur 14: Queue (Mobile)



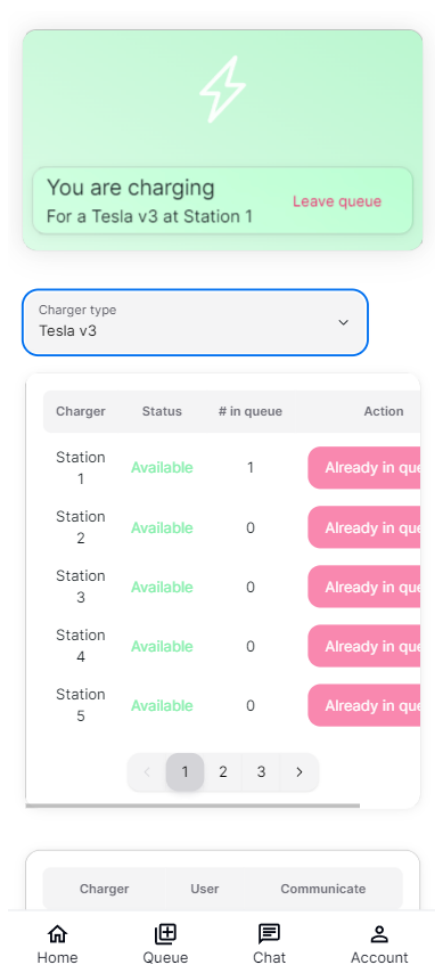
Figuur 15: Queue (Desktop)



Figuur 16: Queue (Mobile)



Figuur 17: Queue (Desktop)



Figuur 18: Queue (Mobile)

#### 3.1.6.1 Laadstatus informatie voor gebruiker

- Doel: De gebruiker informeren over hun huidige status in de wachtrij (niet in wachtrij, in wachtrij, aan het opladen).

#### 3.1.6.2 Wachtrij verlaten knop

- Doel: De gebruiker de mogelijkheid geven om de wachtrij te kunnen verlaten.
- On click: De gebruiker verlaat de wachtrij.
- On hover (enkel op desktop): Toont hover state.
- Hidden: Als de gebruiker nog niet in een wachtrij zit, wordt deze knop niet getoond.

#### 3.1.6.3 Type laadpaal filter

- Doel: De gebruiker de mogelijkheid geven te filteren op een bepaald type van lader, zodat enkel de plaatsen met dit type getoond worden.
- On click: Klapt open en toont de verschillende laadtypes.
- On select: Filtert de verschillende plaatsen op het geselecteerd laadtype.
- On hover (enkel op desktop): Toont hover state.

#### 3.1.6.4 Enter queue (voor het laadtype) knop

- Doel: De gebruiker de mogelijkheid geven zich snel in de wachtrij te plaatsen voor het geselecteerde laadtype.
- On click: Er wordt gekeken naar een optimale plaats om in de wachtrij te staan (minste aantal gebruikers in wachtrij). De gebruiker komt hierbij in te staan.
- On hover (enkel op desktop): Toont hover state.
- Hidden: Als de gebruiker al in een wachtrij zit, wordt deze knop niet getoond.

#### 3.1.6.5 Add to queue (voor een laadplaats) knop

- Doel: De gebruiker de mogelijkheid geven zich in de wachtrij te plaatsen voor een bepaalde laadplaats/ laadstation.
  - On click: De gebruiker komt in de wachtrij terecht van het geselecteerde laadstation.
  - On hover (enkel op desktop): Toont hover state.
  - Disabled: Al is de gebruiker al in een wachtrij zit, wordt deze knop disabled.
-

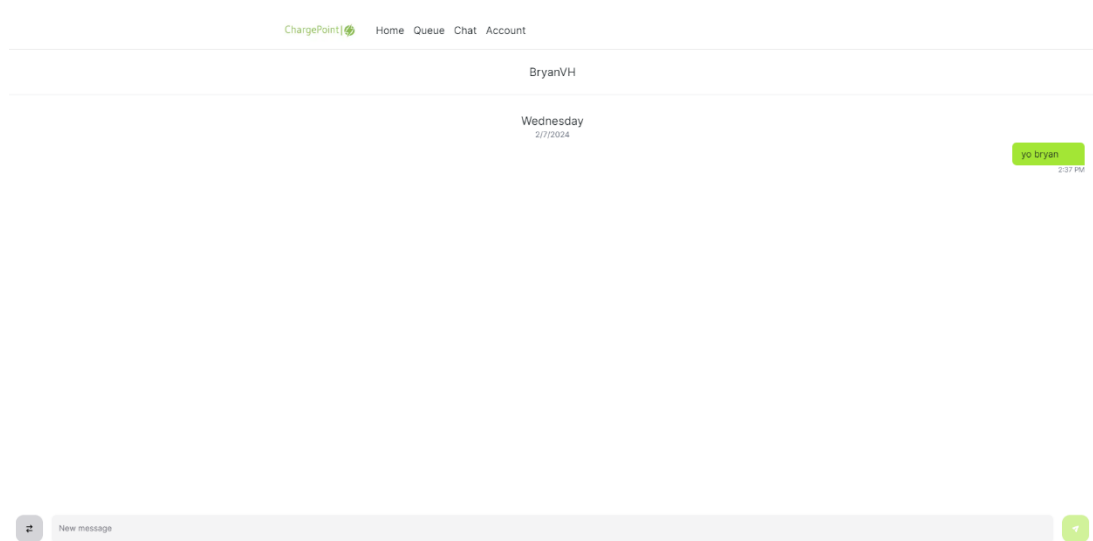
#### 3.1.6.6 Tabel paginatie knoppen:

- Doel: De gebruiker de mogelijkheid geven op andere pagina's van de tabel te gaan kijken.
- On click van pijlen: De gebruiker gaat 1 pagina vooruit of achteruit (als dat mogelijk is), afhankelijk van op welke knop gedrukt wordt.
- On click van een nummer: De gebruiker gaat rechtstreeks naar die pagina.
- On hover (enkel op desktop): Toont hover state.

#### 3.1.6.7 Chat with knop

- Doel: De gebruiker de mogelijkheid geven om te chatten met iemand in de wachtrij. Dit kan gebruikt worden om te vragen te wisselen van plaats in de wachtrij.
  - On click: De gebruiker wordt naar de chat herleid.
  - On hover (enkel op desktop): Toont hover state.
  - Hidden: Als de gebruiker niet in een wachtrij zit, wordt deze knop niet getoond.
-

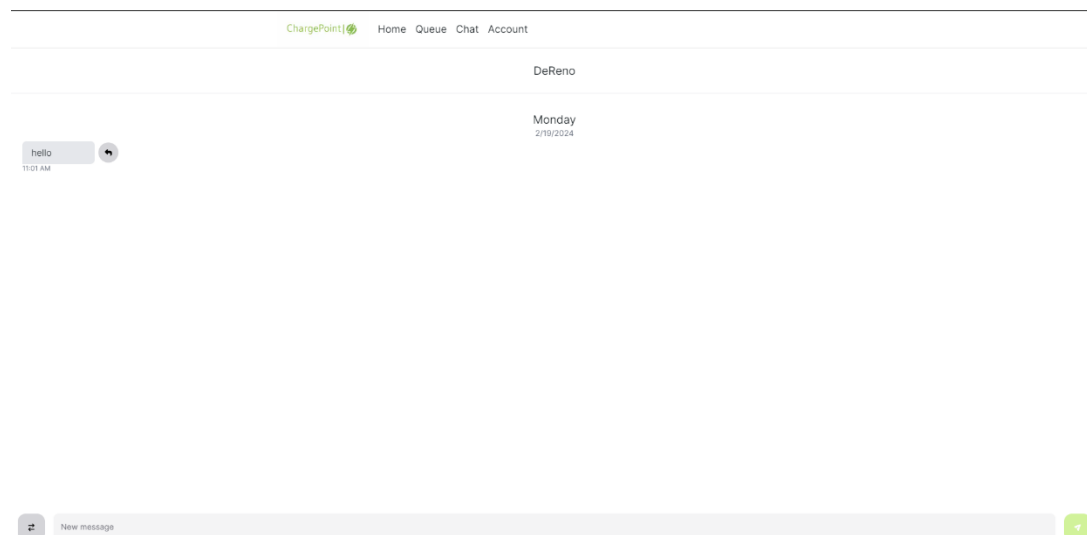
### 3.1.7 Chat



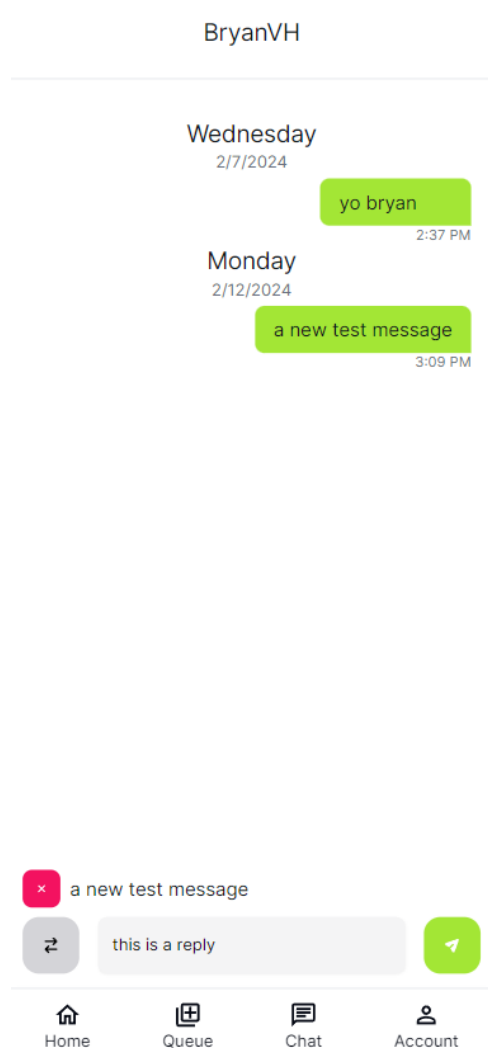
*Figuur 19: Chat (Desktop)*



*Figuur 20: Chat (Mobile)*

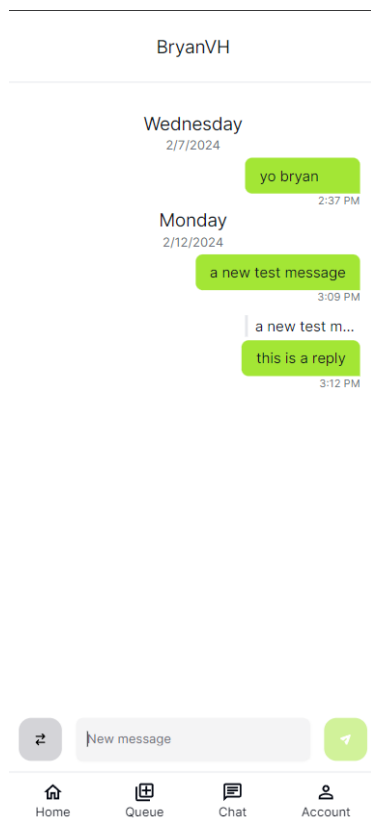


Figuur 21: Knop om te reageren op een bericht (Desktop)

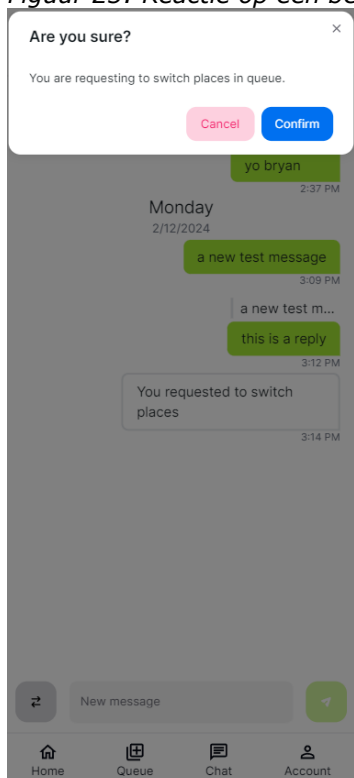


Figuur 22: Reageren op een bericht (Mobile)

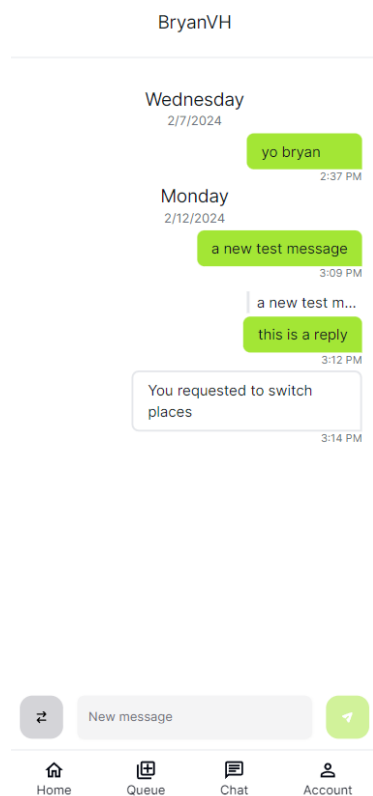




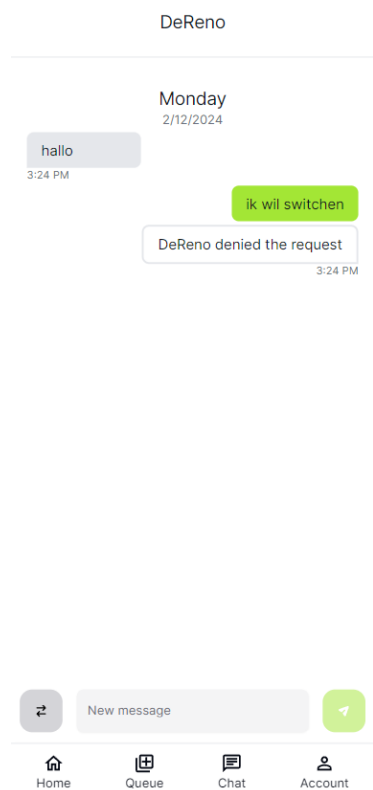
Figuur 23: Reactie op een bericht (Mobile)



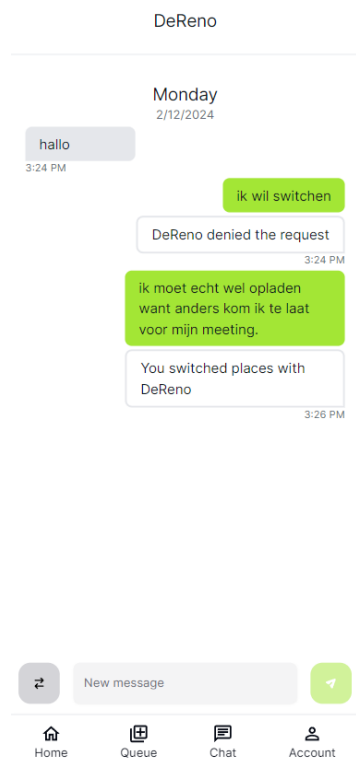
Figuur 24: Confirmatie voor verzoek om te veranderen van plaats in queue (Mobile)



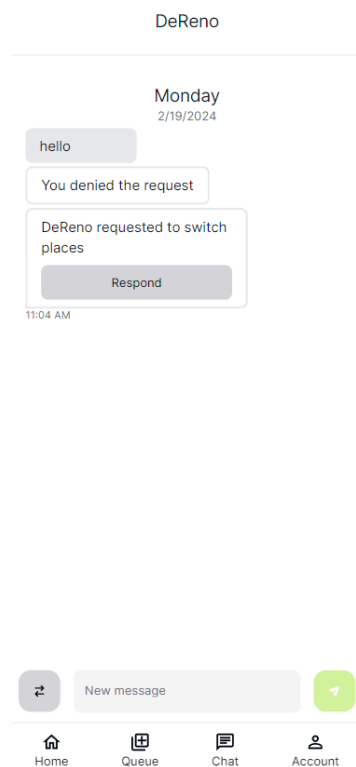
*Figuur 25: Verzoek om te veranderen van plaats in queue (Mobile)*



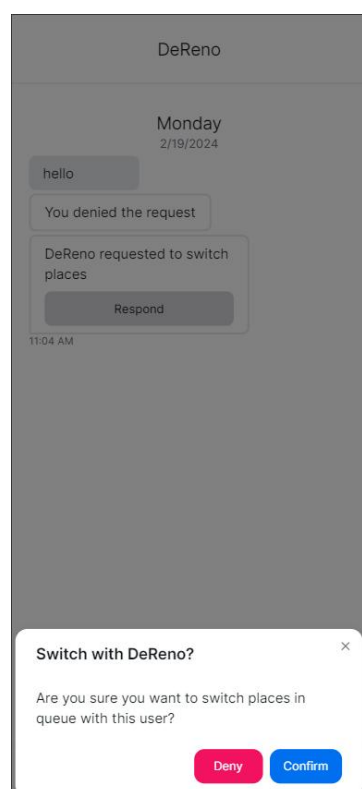
*Figuur 26: Afgewezen verzoek (Mobile)*



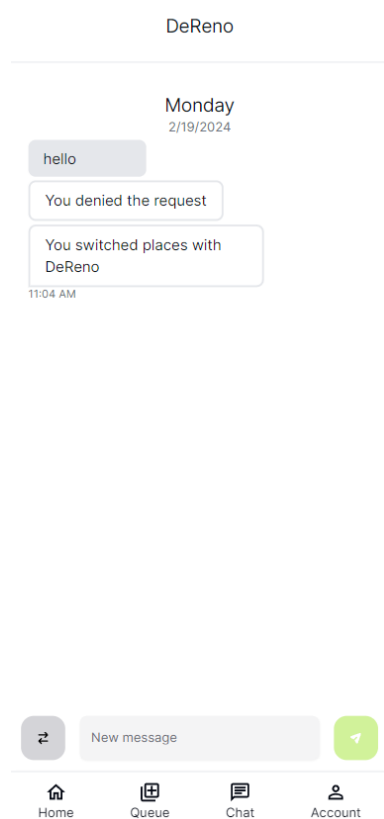
*Figuur 27: Geaccepteerd verzoek (Mobile)*



*Figuur 28: Reageren op verzoek (Mobile)*



*Figuur 29: Reageren op verzoek (Mobile)*



*Figuur 30: Bericht na het afwijzen of accepteren (Mobile)*

#### 3.1.7.1 Normaal bericht

- Doel: Deze geven weer aan de gebruiker wat er gestuurd is in de chat. Deze hebben een gevulde achtergrond, groen of grijs.
- On hover (enkel op desktop): Toont de reactie knop, zie figuur 15
- On swipe (enkel op mobile): Na een swipe is de gebruiker aan het reageren op het bericht, zie figuur 11. Voor een bericht gestuurd door de gebruiker zelf moet deze swipe naar links gebeuren, anders naar rechts.

#### 3.1.7.2 Input veld

- Doel: De gebruiker de mogelijkheid geven een bericht in te geven.
- On click: De gebruiker kan starten met typen in het veld.

#### 3.1.7.3 Verstuur knop

- Doel: De gebruiker de mogelijkheid geven hun bericht te versturen.
- On click: Het bericht wordt verstuurd over de websocket connectie. Enkel als er tekst in het input veld staat kan hier pas op geklikt worden.
- On hover (enkel op desktop): Toont hover state.

#### 3.1.7.4 Verwisselen van plaats verzoek knop

- Doel: Opent een modal dat de gebruiker de mogelijkheid geeft een verzoek te sturen naar de andere gebruiker om te verwisselen van plaats in de wachtrij.
- On click: Opent een modal voor confirmatie, zie figuur 13.
- On hover (enkel op desktop): Toont hover state.

#### 3.1.7.5 Verwisselen van plaats verzoek cancel knop

- Doel: De gebruiker de mogelijkheid geven het verzoek toch niet te sturen. Denk aan een perongelukke klik of van gedachte veranderen.
- On click: Sluit het verzoek modal.
- On hover (enkel op desktop): Toont hover state.

#### 3.1.7.6 Verwisselen van plaats verzoek confirm knop

- Doel: De gebruiker de mogelijkheid geven het verzoek te bevestigen, waardoor er weldegelijk een verzoek verstuurd zal worden.
  - On click: Verstuurd een bericht over de websocket. Dit bericht is een verzoek om te veranderen van plaats in de wachtrij, zie figuren 14 en 22. De modal wordt gesloten.
  - On hover (enkel op desktop): Toont hover state.
-

#### 3.1.7.7 Reageren op verzoek knop

- Doel: Opent een modal dat de gebruiker de mogelijkheid geeft het verzoek af te wijzen of te accepteren.
- On click: Opent een modal waarin de gebruiker kan kiezen om het verzoek te accepteren of afwijzen.
- On hover (enkel op desktop): Toont hover state.

#### 3.1.7.8 Reageren op verzoek deny knop

- Doel: De gebruiker de mogelijkheid geven het verzoek af te wijzen.
- On click: Wijst het verzoek af. Het verzoeksbericht wordt geupdated om te reflecteren dat het afgewezen is. De modal wordt gesloten.
- On hover (enkel op desktop): Toont hover state.

#### 3.1.7.9 Reageren op verzoek accept knop

- Doel: De gebruiker de mogelijkheid geven het verzoek te accepteren.
- On click: Accepteerd het verzoek. Het verzoeksbericht wordt geupdated om te reflecteren dat het geaccepteerd is. De modal wordt gesloten.
- On hover (enkel op desktop): Toont hover state.

#### 3.1.7.10 Afgewezen of geaccepteerd verander verzoek bericht

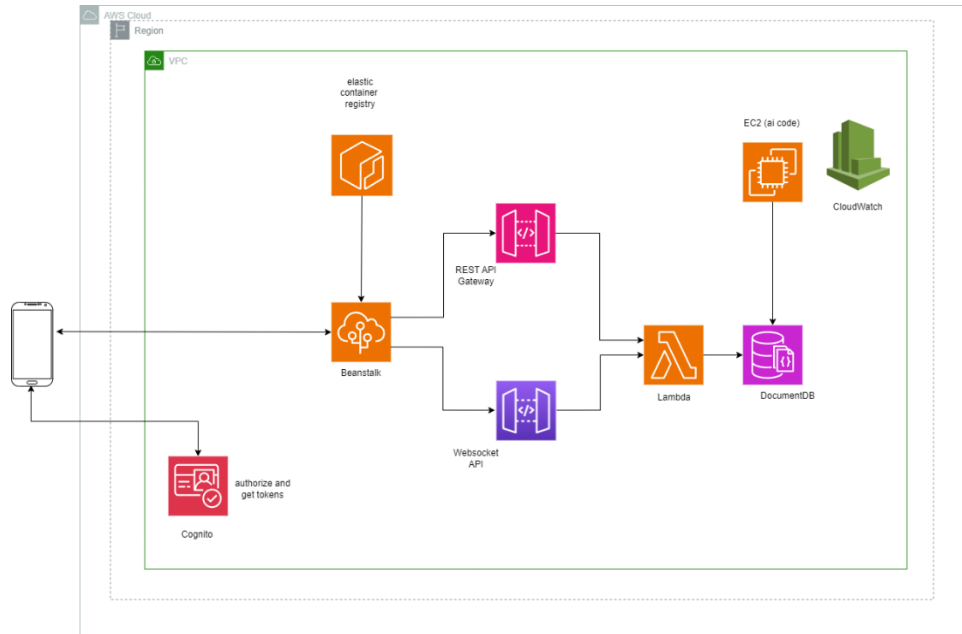
- Doel: Tonen aan de gebruiker wat de reactie was op het verzoek. Deze hebben een grijze border en transparante achtergrond.
-

## 4 CLOUD

In dit deel bespreken we de resources die we hebben opgezet in AWS om onze applicatie in de cloud te kunnen gebruiken.

### 4.1 AWS-topologie

Hier ziet uw een schematische voorstelling van de infrastructuur.



Figuur 31: Topologie

We maken gebruik van de volgende services:

- AWS Cognito
- AWS elastic Beanstalk
- AWS EC2
- AWS API-Gateway
- AWS Lambda
- Amazon S3
- Amazon Documentdb
- Amazon Cloudwatch
- AWS Elastic Container Registry
- AWS VPC

#### **4.1.1 AWS Cognito**

Amazon Cognito is een beheerde service van Amazon Web Services (AWS) die identiteits- en toegangsbeheer (IAM) biedt voor applicaties. Het maakt het ook mogelijk voor gebruikers om te kunnen inloggen via andere media door middel van OAuth & SAML. Wij maken gebruik van het identiteitsbeheer met de userpools waarin wij password policies definiëren en users beheren. We maken ook gebruik van de ingebouwde lambda trigger die getriggerd wordt wanneer iemand zijn account confirmation mail accepteert.

#### **4.1.2 AWS elastic Beanstalk**

AWS Elastic Beanstalk is een service waarmee je automatisch schaalbare applicaties kunt implementeren op AWS. Het maakt het eenvoudig om applicaties te deployen en te beheren zonder dat je je zorgen hoeft te maken over de onderliggende infrastructuur. Wij maken hier gebruik van om de dockerfile van de applicatie te deployen.

#### **4.1.3 AWS EC2**

Het is een managed service binnen de Amazon Web Services (AWS) cloud die virtuele machines/instances aanbiedt. Deze instances zijn als individuele computers met verschillende configuraties van CPU, geheugen en opslagruimte, zodat je de perfecte optie kunt kiezen voor je behoeften. We maken van deze resource gebruik om de code voor de ai bewerkingen op de foto's te kunnen uitvoeren en de data hiervan naar de database te sturen.

#### **4.1.4 AWS api-gateway (rest api)**

AWS API Gateway is een service waarmee je REST API's kunt creëren, publiceren, beheren, monitoren en beveiligen. Je kunt API's maken die toegang hebben tot AWS-services of andere webservices, evenals tot gegevens die zijn opgeslagen in de AWS-cloud.

Dit gebruiken we voor de lambda functie's te triggeren die onze backend code bevatten.

#### **4.1.5 AWS api-gateway (websocket)**

AWS WebSocket API Gateway is een service waarmee je WebSocket API's kunt creëren, publiceren, beheren, monitoren en beveiligen. WebSocket API's zijn ideaal voor toepassingen die realtime communicatie vereisen, zoals chat-apps. Wij gebruiken dit voor de chat functie in de applicatie, de routes die in de api worden aangemaakt triggeren bij connectie of actie's tijdens een lopende connectie lambda functie's waardoor de gebruiker zijn berichten ziet en ontvangt, maar ook bij het heropenen van een chat al zijn vorige berichten terugziet.

---



#### **4.1.6 AWS lambda**

AWS Lambda is een serverloze compute service waarmee je code kunt uitvoeren zonder dat je servers hoeft te provisioneren of te beheren. Lambda runt je code op een hoog beschikbare compute-infrastructuur en voert alle beheer van de compute-resources uit, inclusief server- en besturingssysteemonderhoud, capaciteitsprovisionering en automatic schaling, en logging. Met Lambda hoeft je alleen maar je code te leveren in een van de door Lambda ondersteunde talen.

Wij gebruiken lambda functie's voor:

- Backend code
- Functie's websocket
- Chat wipe function

#### **4.1.7 AWS S3**

Amazon S3 is een cloudservice voor object-opslag die grote hoeveelheden gegevens betrouwbaar, veilig en schaalbaar kan opslaan. Het is als een enorme digitale opslagruimte waar je al je bestanden, afbeeldingen, video's en andere digitale middelen kunt bewaren. S3 is gebouwd op dezelfde betrouwbare infrastructuur die Amazon zelf gebruikt om zijn wereldwijde online winkel te runnen. Dit wordt gebruikt om de foto's van de camera in op te slaan.

#### **4.1.8 AWS documentdb**

Amazon DocumentDB is een gemanaged NoSQL-databasedienst binnen de Amazon Web Services (AWS) cloud die gespecialiseerd is in het opslaan, ophalen en indexeren van JSON-documenten. Het heeft een hoge mate van compatibiliteit met MongoDB, een populaire open-source NoSQL-database, waardoor het gemakkelijk is om bestaande MongoDB-applicaties naar AWS te migreren of nieuwe applicaties te bouwen met de vertrouwde tools en code.

#### **4.1.9 AWS Cloudwatch**

CloudWatch is een monitoring- en observatieservice van Amazon Web Services (AWS) die u helpt de prestaties en gezondheid van uw AWS-resources en -applicaties te bewaken.

Wij maken hier gebruik van om de dagelijkse chat wipe lambda functie te triggeren.

#### **4.1.10 AWS elastic container registry**

AWS Elastic Container Registry (ECR) is een gemanageerde service binnen de Amazon Web Services (AWS) cloud die speciaal is ontworpen voor het opslaan, beheren en implementeren van containerimages. Het is als een veilige digitale repository waar je al je Docker-images, OCI-images en andere compatibele artefacten kunt bewaren. Je hoeft je geen

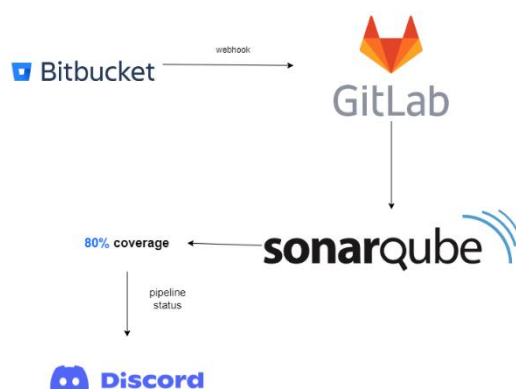
---

zorgen te maken over het opzetten en onderhouden van je eigen containerregistry, want AWS regelt dat allemaal voor je.

#### 4.1.11 AWS VPC

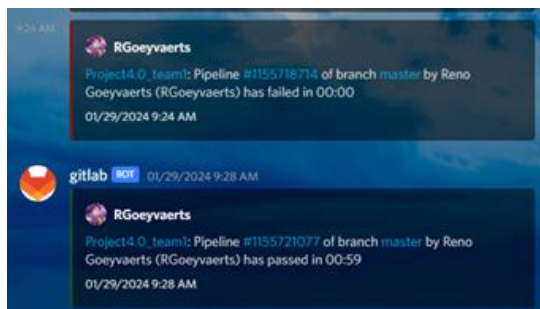
Amazon Virtual Private Cloud (VPC) is een service waarmee je een virtueel netwerk in de cloud kunt creëren en aanpassen. Hiermee kun je je AWS-resources, zoals Amazon EC2-instances, isoleren en op een veilige manier met elkaar verbinden. We maken dan gebruik van de subnets, internet gateway, nat gateway, security groups en ACL's. Om ervoor te zorgen dat onze applicatie op een veilige manier verbinding heeft met het internet.

## 4.2 Pipeline



Figuur 32: pipeline flow

We maken gebruik van gitlab pipeline om de testen te kunnen uitvoeren met sonarcloud. Dit hebben we niet volledig kunnen toepassen aangezien we niet voor elk deel van de frontend testen heb kunnen schrijven. De pipline kan getriggerd worden door nieuwe dingen toe te voegen aan de gitlab repo of een pull request te mergen op bitbucket dat door middel van een webhook die de pipeline in gitlab gaat triggeren. We maken gebruik van status berichten die op het einde van de pipeline een bericht sturen in een bepaald tekst-channel in onze discord server om te zien of de pipeline succesvol was.



Figuur 33: Discord notifications

### 4.3 IAC

Terraform is een open-source tool van HashiCorp die je helpt om infrastructuur te definiëren, beheren en provisioneren met behulp van code. In plaats van handmatig servers te configureren en te beheren, kun je met Terraform je infrastructuur beschrijven in declaratieve configuratiebestanden. We hebben terraform code voorzien voor de belangrijkste resources voor de applicatie, deze kan u terugvinden in de gitlab repo. We maken gebruik van Terraform omdat dit wel wat voordelen met zich meebrengt namelijk:

- Eenvoudig beheer
  - Verhoogde efficiëntie
  - Verbeterde consistentie: Vermindert menselijke fouten en garandeert consistente configuraties.
  - Schaalbaarheid: Breid je infrastructuur eenvoudig uit met behulp van modules en herbruikbare code.
  - Betere samenwerking: Deel configuraties met teamleden en stakeholders.
  - Open source
-

## 5 AI

### 5.1 Inhoud

In dit deel bespreken we het achterliggende AI-model. Dit model heeft als doel het implementeren van een AI-gebaseerd systeem voor het monitoren van parkeerplaatsen. Het systeem maakt gebruik van YOLO (You Only Look Once) voor objectdetectie, AWS S3 voor het opslaan van afbeeldingen, MongoDB voor het opslaan van parkeerinformatie, en OpenCV voor beeldverwerking.

### 5.2 Gebruikte technieken

#### 5.2.1 OpenCV

OpenCV wordt gebruikt voor beeldverwerking, waaronder het roteren en vergroten van afbeeldingen, het tekenen van rechthoeken en tekst, en het uitvoeren van geometrische operaties. Wij hebben in onze code gebruik gemaakt van OpenCV om de parkeerplaatsen te kunnen indelen. Wij hebben gebruik gemaakt van coördinaten om rechthoeken rond de parkeerplaatsen te kunnen tekenen. Daarnaast gebruiken we het middelpunt van de parkeerplaats. Dit om te kijken of deze plaats leeg of bezet is.

#### 5.2.2 YOLOv8 (You Only Look Once)

YOLO is een geavanceerd objectdetectiemodel dat is ontwikkeld voor computer vision-toepassingen. Het wordt specifiek gebruikt voor het detecteren en lokaliseren van objecten in afbeeldingen en video's. Wij hebben gebruik gemaakt van versie 8. Dit omdat YOLOv8 sneller is dan YOLOv5 en dus meer geschikt is voor realtime object detectie. Ook is er gebruik gemaakt van het large model van YOLOv8. Dit model gaf de nauwkeurigste resultaten weer.

Er wordt ook gebruik gemaakt van de "Ultralytics" package. Deze wordt gebruikt voor het integreren van YOLO in het project. Hiermee kunnen eenvoudig voorspellingen worden gedaan op basis van het getrainde YOLO-model.

#### 5.2.3 Documentdb

Documentdb wordt gebruikt als de database voor het opslaan van parkeerinformatie. Er zijn drie collecties: collection voor de actuele parkeerinformatie, queueentries voor wachtrijinformatie en chargertypes voor informatie over laadstations.

---

### 5.2.4 AWS S3

Amazon Simple Storage Service (S3) wordt gebruikt om afbeeldingen op te slaan en op te halen. De AWS SDK voor Python (Boto3) wordt gebruikt om toegang te krijgen tot S3-buckets.

### 5.2.5 Werking

Het model wordt geactiveerd zodra er een foto wordt opgeslagen in de AWS S3-bucket. De afbeelding wordt uit de S3-bucket opgehaald en opgeslagen om bewerkingen uit te voeren. OpenCV en het YOLOv8-model worden vervolgens gecombineerd. Eerst wordt OpenCV geactiveerd om te controleren of er een object binnen de getekende rechthoek aanwezig is. Als er een object binnen de rechthoek wordt gedetecteerd, bepaalt OpenCV het middelpunt ervan. Daarna wordt het YOLOv8-model geactiveerd om te controleren of het gedetecteerde object daadwerkelijk een auto is, en geen bijvoorbeeld een persoon die over de parkeerplaats loopt. Het YOLOv8-model herkent uitsluitend auto's die zich op de parkeerplaatsen bevinden, omdat deze gebieden als de belangrijkste oppervlakten op de foto worden beschouwd.

We hebben gebruikgemaakt van een voorgetraind YOLOv8-model, omdat dit al zeer nauwkeurig is. Om de gegevens op te slaan, is DocumentDB, een NoSQL-database, gebruikt. Wanneer een auto op een parkeerplaats staat, worden de gegevens naar de database gestuurd. Hierbij worden de Parkingspot ID, Occupancy Status, Start Time en End Time bijgehouden. De Parkingspot ID geeft aan welke plaats bezet of vrij is, waarbij nummer 1 zich het dichtst bij de ingang van Corda Campus gebouw 1 bevindt. Occupancy Status geeft de beschikbaarheid van de parkeerplaats weer, met mogelijke waarden "Available" of "Occupied". Start Time verwijst naar het tijdstip waarop een parkeerplaats als "Occupied" wordt gemarkeerd, terwijl End Time verwijst naar het moment waarop een parkeerplaats van "Occupied" naar "Available" verandert.

---

## **6 IOT**

### **6.1 Inhoud**

Voor het IOT-gedeelte was de opdracht om een GoPro Hero 4 te koppelen aan een raspberry pi 3b en dan met die GoPro een foto maken die doorsturen naar de raspberry pi en dan die foto in de Cloud plaatsen zo dat de AI student hier bezettings detectie van de parkeerplaatsen kan op toe passen.

### **6.2 Setup**

Hier zal ik kort de instelling van de benodigde apparatuur toelichten.

#### **6.2.1 GoPro Hero 4**

Voor het instellen heb je 3 dinge nodig: de GoPro zelf een sd kaartje en een GoPro houder met een zuignap om het aan de raam te monteren. Steek de sd kaart in de camera en zet de camera aan, zet nu de camera in wifi modus zo dat we de GoPro met de raspberry pi kunnen verbinden.

#### **6.2.2 Raspberry pi 3B**

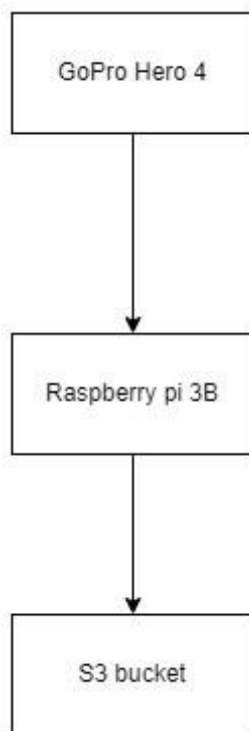
Zet de pi aan en connecteer met de wifi van de GoPro je kunt nu de code aanzetten.

#### **6.2.3 Code**

De code kan teruggevonden worden in de zipfile.

---

#### 6.2.4 Schematische voorstelling



*Figuur 34: IOT-schema*

Even samenvatten een foto wordt vastgelegd met een GoPro en vervolgens opgeslagen op het apparaat. Daarna wordt deze meest recente foto door een Raspberry Pi opgehaald en onder een consistente bestandsnaam opgeslagen. Vervolgens wordt de foto naar wens van de AI student bijgesneden en gedraaid, waarna deze naar een AWS S3 Bucket wordt verzonden voor opslag. Hier kan de AI-student zijn of haar code uitvoeren op de verwerkte afbeelding.

---

## **7 BESLUIT**

We vonden dit een zeer leerzaam en uitdagend project waar we met vallen en opstaan doorgekomen zijn. We zijn als team gegroeid en hebben geleerd om samen aan iets te werken in plaats van ieder zijn vakgebied.

De documentatie bevat gedetailleerde informatie over de architectuur, de gebruikte technologieën, de implementatieprocedure, en de gebruikershandleiding. Dit document zal als waardevolle referentie dienen voor toekomstige onderhouds- en uitbreidingswerkzaamheden.

We willen onze opdrachtgevers, docenten en medestudenten bedanken voor hun voortdurende ondersteuning en feedback gedurende dit project. We zijn trots op wat we hebben bereikt en zijn ervan overtuigd dat deze webapplicatie een positieve impact zal hebben op de overgang naar efficiënt laden.

---



## 8 BRONNEN

*API Management - Amazon API Gateway - AWS.* (z.d.). Amazon Web Services, Inc.

<https://aws.amazon.com/api-gateway/>

*Authentication Service - Customer IAM (CIAM) - Amazon Cognito - AWS.* (z.d.).

Amazon Web Services, Inc. <https://aws.amazon.com/cognito/>

*AWS re:Invent 2023 - Compute Innovation Talk.* (z.d.). [Video]. Amazon Web

Services, Inc. <https://aws.amazon.com/ec2/>

*Cloud Object Storage - Amazon S3 - AWS.* (z.d.). Amazon Web Services, Inc.

<https://aws.amazon.com/s3/>

*Container Registry - Amazon Elastic Container Registry (Amazon ECR) - AWS.*

(z.d.). Amazon Web Services, Inc. <https://aws.amazon.com/ecr/>

*Introduction to Amazon CloudWatch.* (z.d.). [Video]. Amazon Web Services, Inc.

<https://aws.amazon.com/cloudwatch/>

*Introduction to AWS Elastic Beanstalk (2:14).* (z.d.). [Video]. Amazon Web

Services, Inc. <https://aws.amazon.com/elasticbeanstalk/>

*JSON Document Database - Amazon DocumentDB - AWS.* (z.d.). Amazon Web

Services, Inc. <https://aws.amazon.com/documentdb/>

*OpenCV: OpenCV modules.* (z.d.). <https://docs.opencv.org/4.x/index.html>

*Private Cloud - Amazon Virtual Private Cloud (VPC) - AWS.* (z.d.). Amazon Web

Services, Inc. <https://aws.amazon.com/vpc/>

*Serverless Function, FAAS Serverless - AWS Lambda - AWS.* (z.d.). Amazon Web

Services, Inc. <https://aws.amazon.com/lambda/>

*What is Amazon API Gateway? - Amazon API Gateway.* (z.d.).

<https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html>

*What is Amazon EC2? - Amazon Elastic Compute Cloud. (z.d.).*

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>

*What is Amazon Elastic Container Registry? - Amazon ECR. (z.d.).*

<https://docs.aws.amazon.com/AmazonECR/latest/userguide/what-is-ecr.html>

*What is Amazon S3? - Amazon Simple Storage Service. (z.d.).*

<https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html>

*What is AWS Lambda? - AWS Lambda. (z.d.).*

<https://docs.aws.amazon.com/lambda/latest/dg/welcome.html>

*YOLOV8: a new State-of-the-Art Computer Vision model. (z.d.).* <https://yolov8.com/>

---