



Cedric Vanhaegenberg

Juan José Muñoz

Arteveldehogeschool-Grafische & Digitale Media

New Media Development II

2018-2019

Inhoudstafel

Briefing	3
Competitieve Analyse	5
Persona sheet	10
Empathy map	13
Ideaboard	14
Moodboard	15
Sitemap	16
Wireframes	17
Wireflow	27
Style tile	28
Style Guide	29
Mockup	30
Visual designs	31
Backend	40
Database	42
Frontend	48
User handleiding	50

Discover

Briefing

Opdrachtgever

Arteveldehogeschool/ Timelab

Voor het vak mobdev-II moeten we een oplossing vinden om de buurt hechter te maken (cultureel, sociaal, ...) via een app / webapp, enkele suggesties:

- Maatschappelijk relevant
- Buurt dichter bij elkaar brengen

Agenda

- Start: Ma 01-04
- Milestone 1 (+feedback): Ma 29-04
- Milestone 2 (+feedback): Ma 13-05
- Presentation in Amsterdam: eind mei
- Deadline: Zo 16-06-2019 om 20u
- Examens: Ma 17-06-2019 of Di 18-06-2019

Doelpubliek

Mensen van de buurt van Timelab (Mogelijk om uit te breiden)

Geslacht : 0 ,1 , 2 , 9

Leeftijd: 20 tot 50 jaar

talenkennis : Engels

Inkommen: niet van toepassing

Geboorteplaats : Gent-België

Define

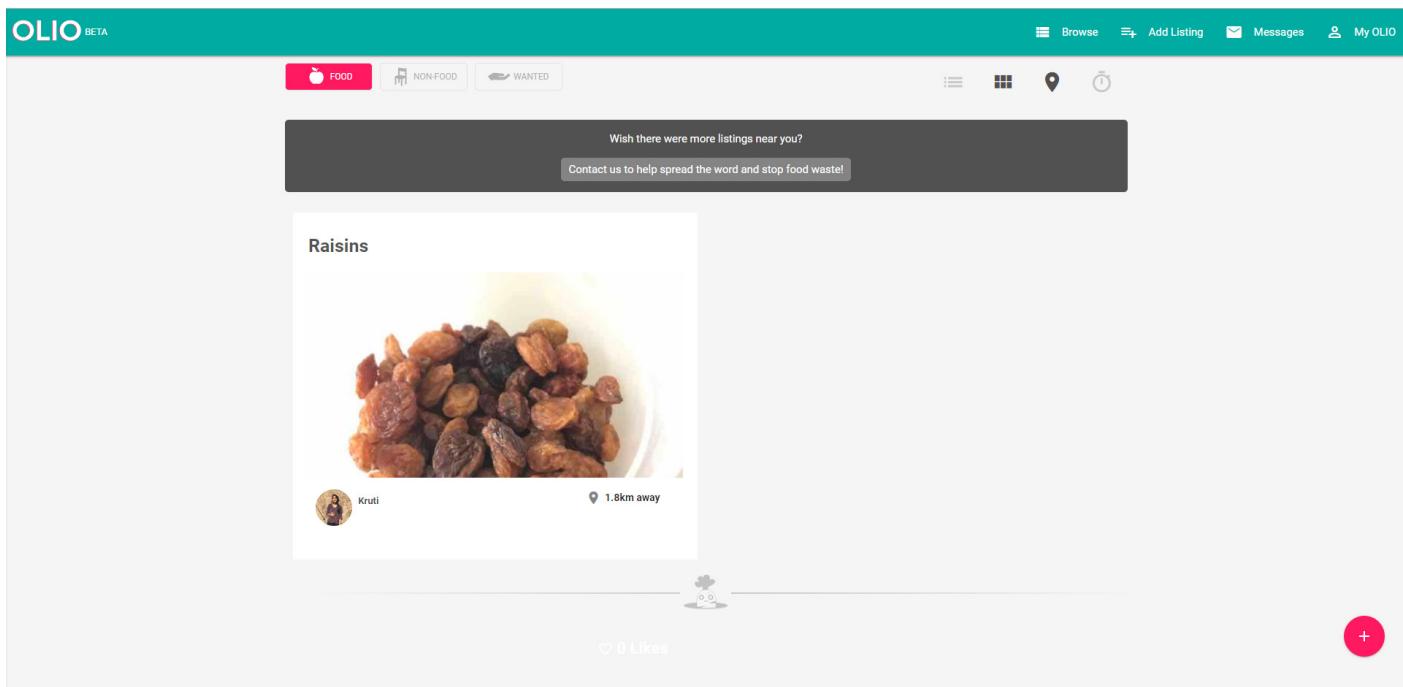
Planning

Task name	Progress	14 Jun				17 Jun
		Fri 6/ 14	Sat 6/ 15	Sun 6/ 16	Mon 6/ 17	
Productiedossier maken	63%	Productiedossier maken				
Discovery	Complete	Discovery				
Briefing	Complete	Briefing				
Define	78%	Define				
Concurrentie analyse	Complete	Concurrentie analyse				
Planning	Complete	Planning				
Scenarios	50%	Scenarios				
New task	Complete				New task	
Journey map	10%	Journey map				
Ideaboard	Complete	Ideaboard				
Persona sheet	Complete	Persona sheet				
Mood board	Complete	Mood board				
Design	60%	Design				
Sitemap	Complete	Sitemap				
Wireframe	Complete	Wireframe				
Wireflow	95%	Wireflow				
Style tile	25%	Style tile				
Style guide	10%	Style guide				
Visual designs	Complete				Visual designs	
Mockup	50%	Mockup				
Prototype	0%	Prototype				
Develop	70%	Develop				
Coding screenshots	70%	Coding screenshots				
Deliver	10%	Deliver				
Handleiding	10%	Handleiding				
App	42%				App	
DB	Complete				DB	
Define layouts	0%				Define layouts	
Frontend	0%				Frontend	
Backend	68%				Backend	
Location implementatie	90%				Location implementatie	
Chat	75%				Chat	
Routes	75%				Routes	
Comments	Complete				Comments	
Filtering	0%				Filtering	

Competitieve Analyse

<https://olioex.com/>

Het concept 'food sharing' is iets unieks dat je niet vaak terugvindt. OLIO is één van de enigste vlaamse bedrijven die streeft naar een groenere wereld zonder voedselverspilling. Ze hebben hiervoor ook een app zowel voor mobile als desktop.

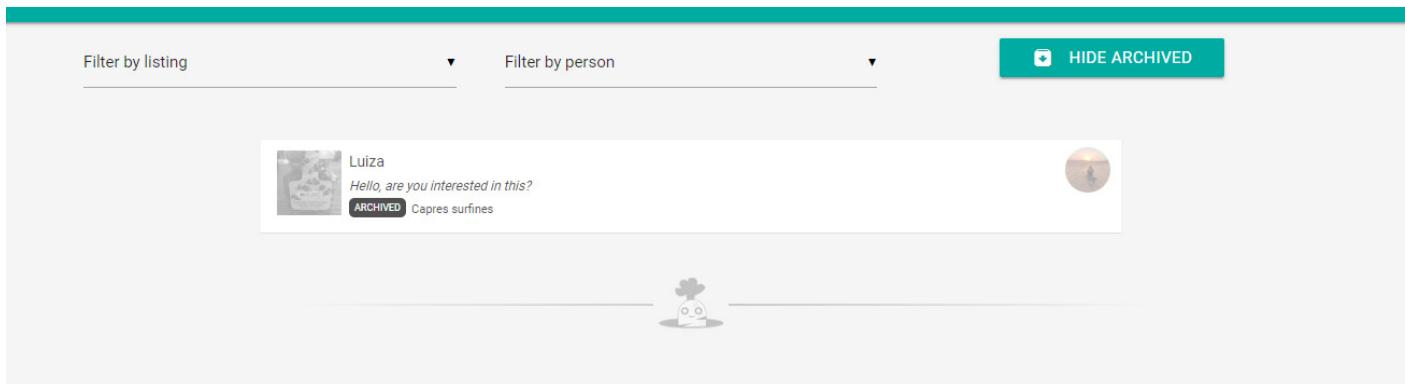


Layout

De layout van de app kan duidelijk beter. Het is zeer simplistisch en straalt geen originaliteit en charme uit. Dit is iets wat wij met Limento beter kunnen doen om gebruikers aan te trekken naar onze app.

Filter

Er zijn verschillende filter mogelijkheden aanwezig in de app. Dit is iets wat we zeker ook aanwezig willen in onze applicatie.



Chat

Je kan gebruikers een bericht sturen wanneer je geïnteresseerd bent. Ook hier zijn verschillende filter opties beschikbaar. In onze app laten we de regio waar de verkoper zich bevindt weg. We laten gebruikers via de chat elkaar ontmoeten en afspreken

Omschrijving

Functionele specificaties

De functionele eisen zijn volledig afhankelijk van de noden van de frontoffice, en indien nodig ook de frontend van de backoffice. Wordt besproken in samenwerking met de docenten en opdrachtgever.

Technische specificaties

Server

De server wordt geschreven met JavaScript (esnext) in Node.js (Links to an external site.)Links to an external site. met het Express (Links to an external site.)Links to an external site. framework. Voor authenticatie en autorisatie gebruiken we het Passport (Links to an external site.)Links to an external site. framework. Binnen Passport zullen we minimaal een local-strategy, jwt-strategy (Links to an external site.)Links to an external site. en een extra strategy (Links to an external site.)Links to an external site. (bv: facebook-strategy, google-strategy, twitchtv-strategy, ...) aanbieden.

De server is voorzien van een API (version controlled). Voor ieder collectie uit de database maken we een corresponderende controller, schema, validatie en routes. De volgende actiemethoden zijn minimaal in de controller aanwezig: index (alle documenten uit een collectie tonen, inclusief pagination), show (GET), create, store (POST), edit, update (PUT), delete (DELETE), softdelete (PATCH) en softundetele (PATCH).

Alle API-endpoints worden gedocumenteerd via de OpenAPI-specificatie (Links to an external site.)Links to an external site.m.b.v swagger-jsdoc en swagger-ui-express (of ReDoc). Beveilig de swagger-ui-express en dus ook de API-endpoints met JWT. De API is afgeschermd voor niet-bevoegden en voldoet aan de normale eisen voor beveiliging.

Zorg voor een globale foutafhandeling, logging en custom 404 en Error pages (m.b.v. ejs (Links to an external site.)Links to an external site.template engine).

De client wordt geïnjecteerd in de root route: `http://{your domain}:{your port}/`. De client wordt niet gerenderd op de server, maar wel de informatie in de <head>. Op deze manier wordt de app SEO-friendly en ook om te bookmarken via OpenGraph e.d.. SSR (Links to an external site.)Links to an external site. is ook mogelijk, maar combineren met PWA (Links to an external site.)Links to an external site. is niet eenvoudig.

Database

We gebruiken MongoDB als databasesysteem. Deze databank moet gehost worden op mLab (Links to an external site.)Links to an external site. of MongoDB (Links to an external site.)Links to an external site.. Gebruik op de campus of binnen het netwerk van de Arteveldehogeschool de wifi "ArteveldeHS Open".

Voor een team dat bestaat uit:

- 1 lid, moeten minimaal 7 collecties (tabellen) aanwezig zijn, exclusief users en roles.
- 2 leden, moeten minimaal 9 collecties (tabellen) aanwezig zijn, exclusief users en roles..
- 3 leden, moeten minimaal 11 collecties (tabellen) aanwezig zijn, exclusief users en roles.

De volgende relaties tussen documenten (en dit zonder embedded documents) moeten minstens eenmaal voorkomen:

1..1 (één op één)

1..N (één op veel)

N..N (veel op veel)

De volgende Unit Tests (Mocha, Chai, Sinon) moeten minimaal aanwezig zijn voor een team dat bestaat uit:

1 lid: tests voor een minimaal 1 controller, tests voor de auth controller, tests voor minimaal 1 mongoose model

2 leden: tests voor een minimaal 2 controllers, tests voor de auth controller, tests voor minimaal 2 mongoose models

3 leden: tests voor een minimaal 3 controllers, tests voor de auth controller, tests voor minimaal 3 mongoose models

Client

De cliënt die geïnjecteerd wordt in de server is een PWA React-client met gedeeltelijke SSR (volledig kan ook). De cliënt bevat ook state management via Redux (Links to an external site.)[Links to an external site.](#)met Redux middleware (Thunk, Sagas of Epics) . Het router-systeem dat we zullen gebruiken binnen React is react-router (Links to an external site.)[Links to an external site..](#)

Styling wordt binnen React toegevoegd via scss en / of via styled components (Links to an external site.)[Links to an external site..](#)

De volgende Unit Tests (Jest, fetch-mock, expect, Sinon) moeten minimaal aanwezig zijn voor een team dat bestaat uit:

1 lid: tests voor een minimaal 1 component, tests voor 1 redux store en middelware

2 leden: tests voor een minimaal 2 componenten, tests voor 2 redux stores en middelware

3 leden: tests voor een minimaal 3 componenten, tests voor 3 redux stores en middelware

Persona sheet

NANCY

Leeftijd 30
Beroep Lerares
Status Single

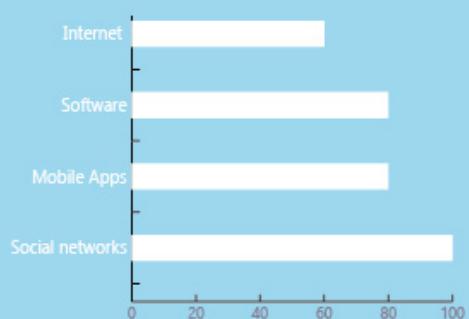


Biografie

Nancy werkt als lerares in Gent. Ze eet graag met andere mensen samen, maar al haar vrienden wonen niet meer in Gent.

Doelen

In de week minstens een keer samen eten met iemand.



PEDRO

Leeftijd 33
Beroep Projectmanager
Status Single

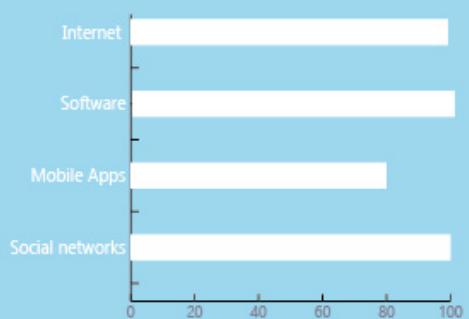


Biografie

Pedro is gestopt met studeren als hij 23 was en hij heeft zijn eigen bedrijf opgericht. Hij organiseert alle soorten evenementen. Ook al is hij heel social, heeft hij niet veel contact met de mensen van zijn buurt

Doelen

Pedro wil de mensen van zijn buurt ontmoeten om af en toe eens op café gaan.



Scenarios

Create a Post

Als geregistreerde gebruiker wil ik een post aanmaken.

Stel ik een aanbieding wil posten

En ik klik op de knop 'Create'

En ik vul alle velden van de formulier

Als ik op de knop 'Post' druk

Dan zou ik de mijn post willen zien in de feed van de applicatie

Filtering op locatie

Als geregistreerde gebruiker wil ik zien of er aanbiedingen zijn in mijn buurt

Stel ik een zoekertje doe gebaseerd op locatie

En ik klik op de knop 'Filter'

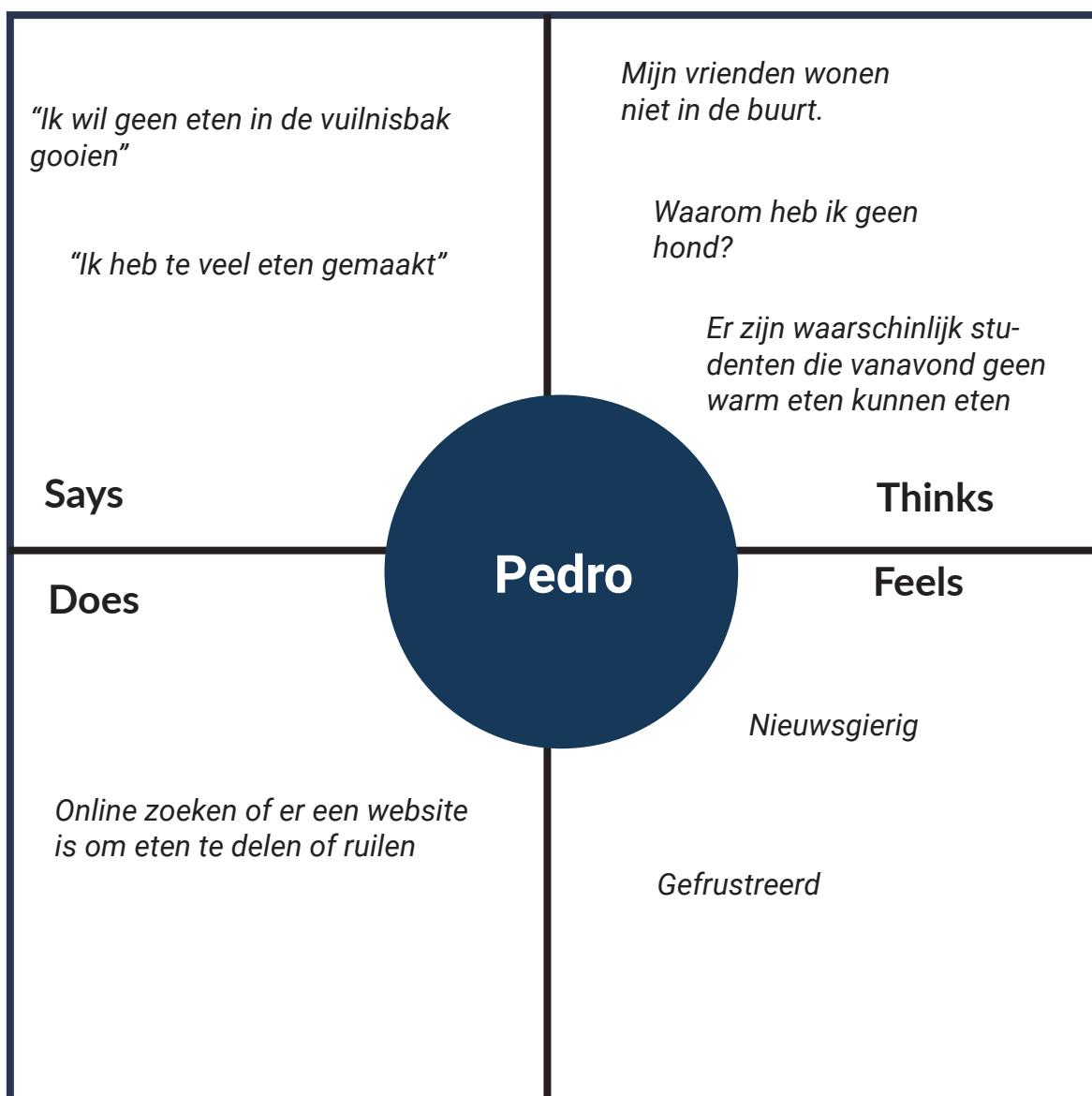
En ik selecteer afstand

Als ik op de knop 'Search' druk

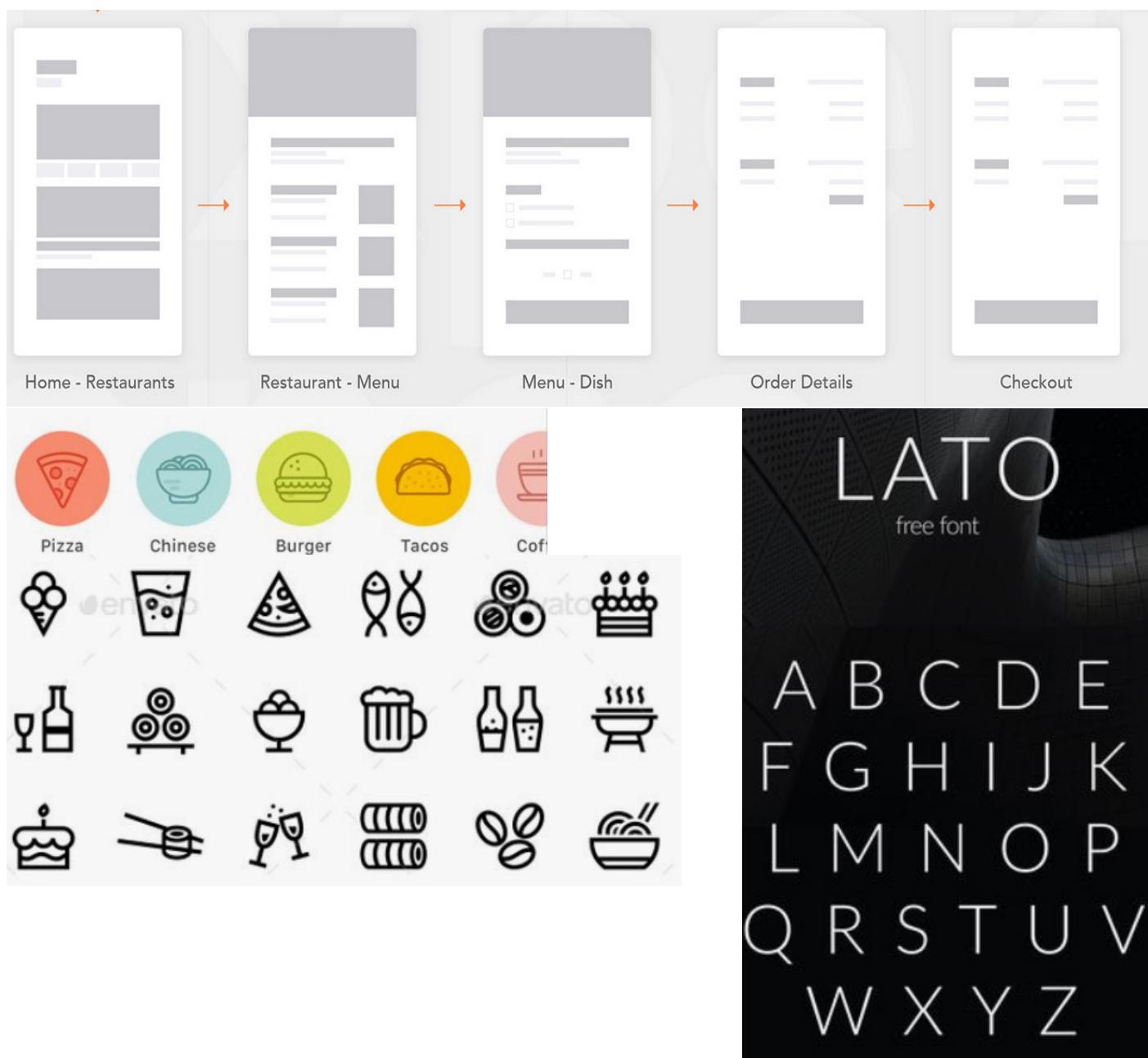
Dan zou ik willen zien hoeveel aanbiedingen zijn in de buurt.

Empathy map

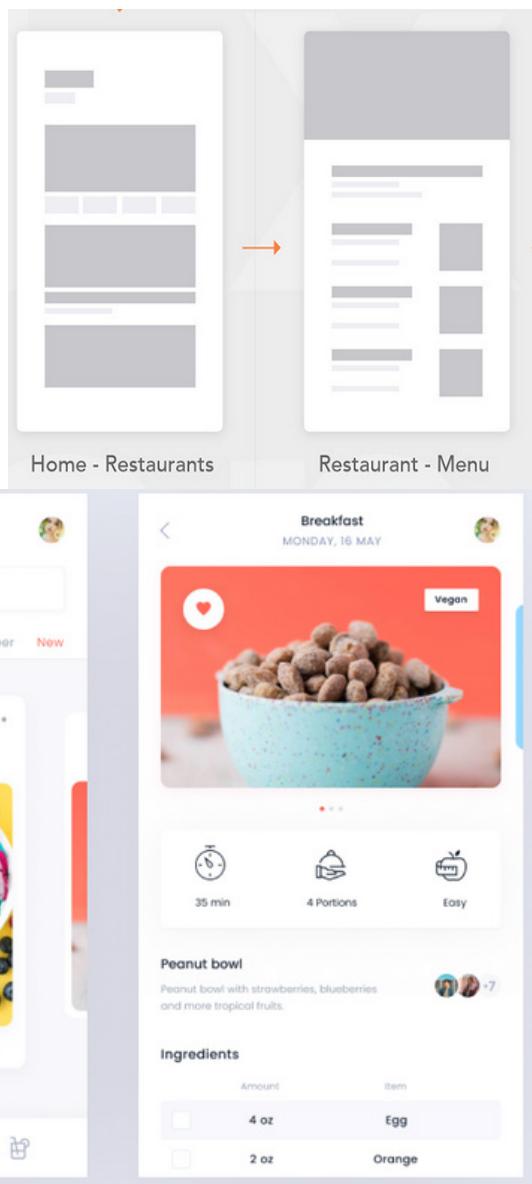
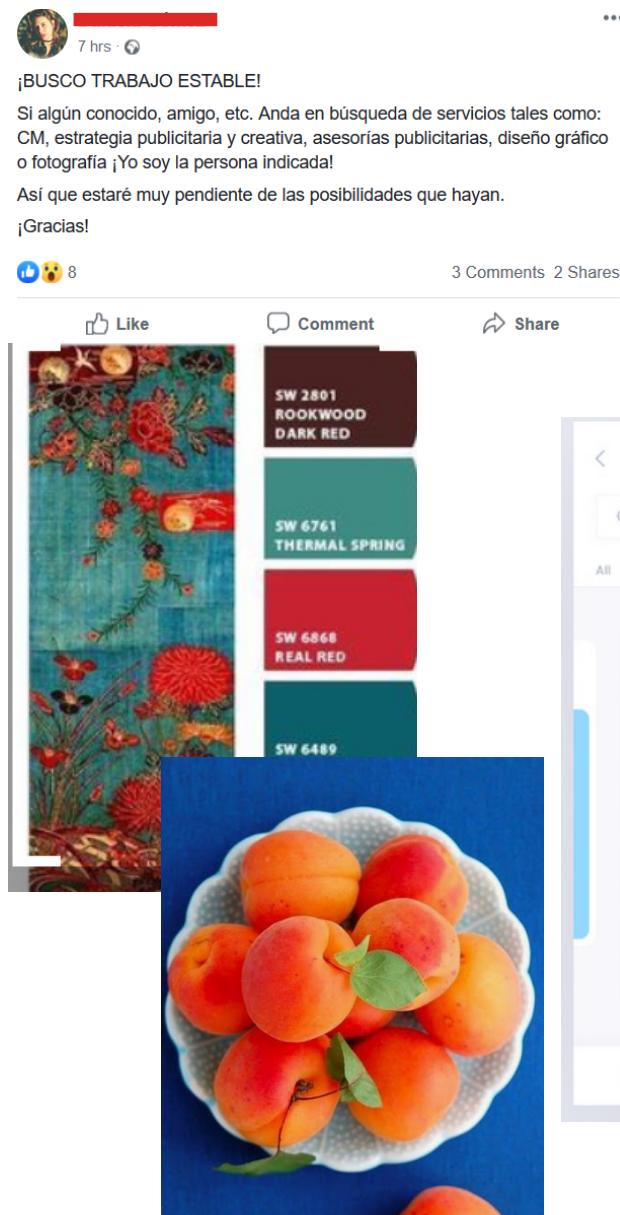
Voorbeeld : Voedsel ruilen en delen



Ideaboard

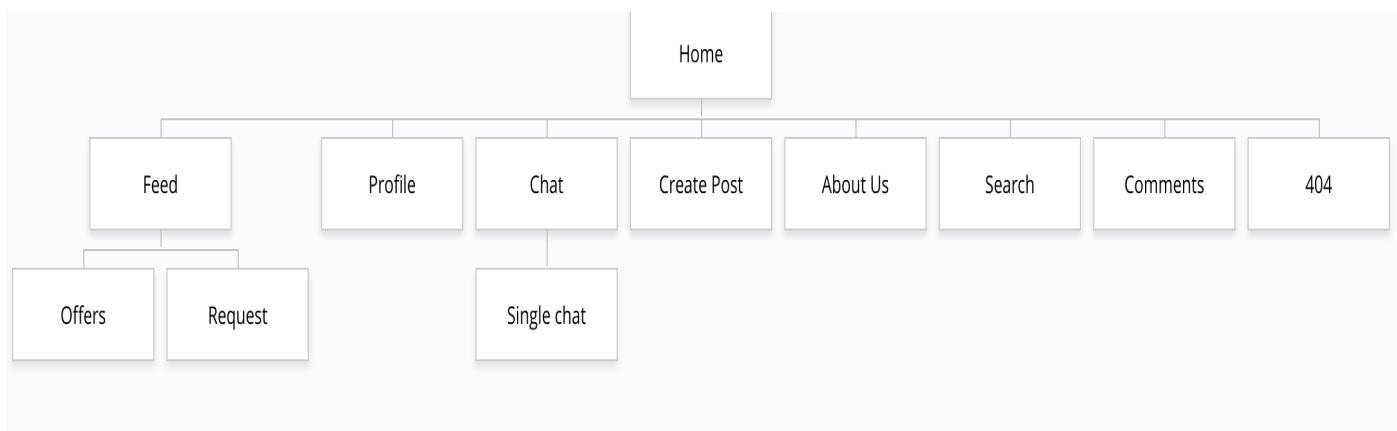


Moodboard

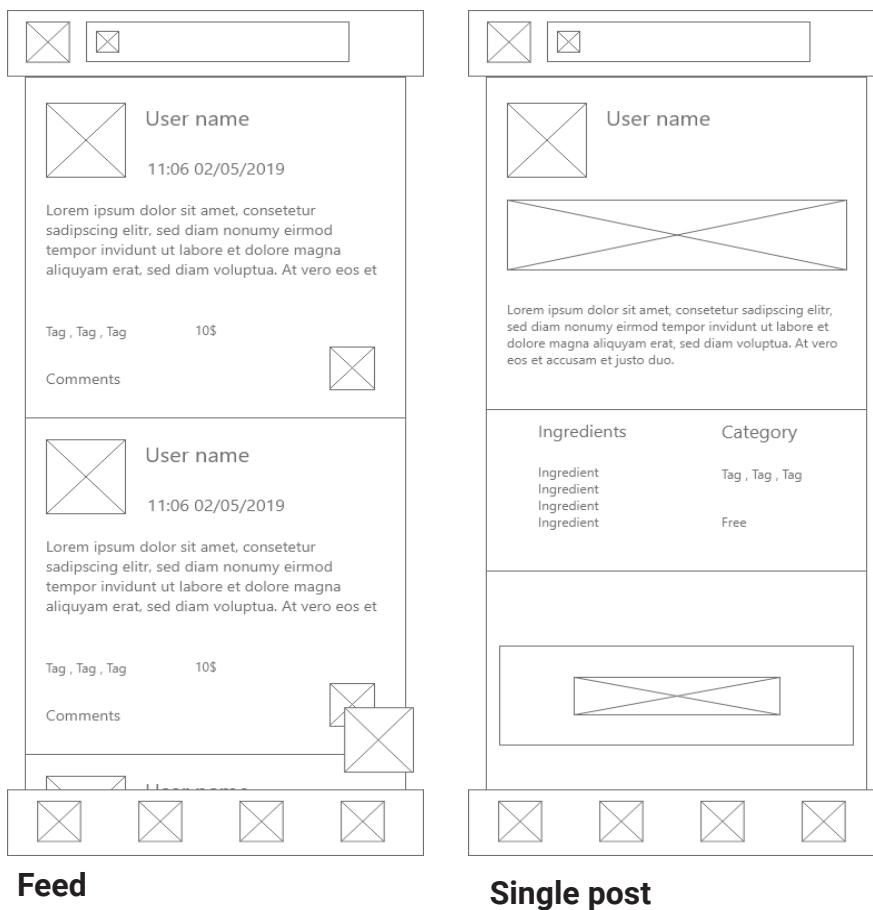


Design

Sitemap



Wireframes



The image displays two wireframe mobile application screens side-by-side. Both screens feature a header bar at the top with a small square icon containing an 'X' in the top-left corner.

Log in screen:

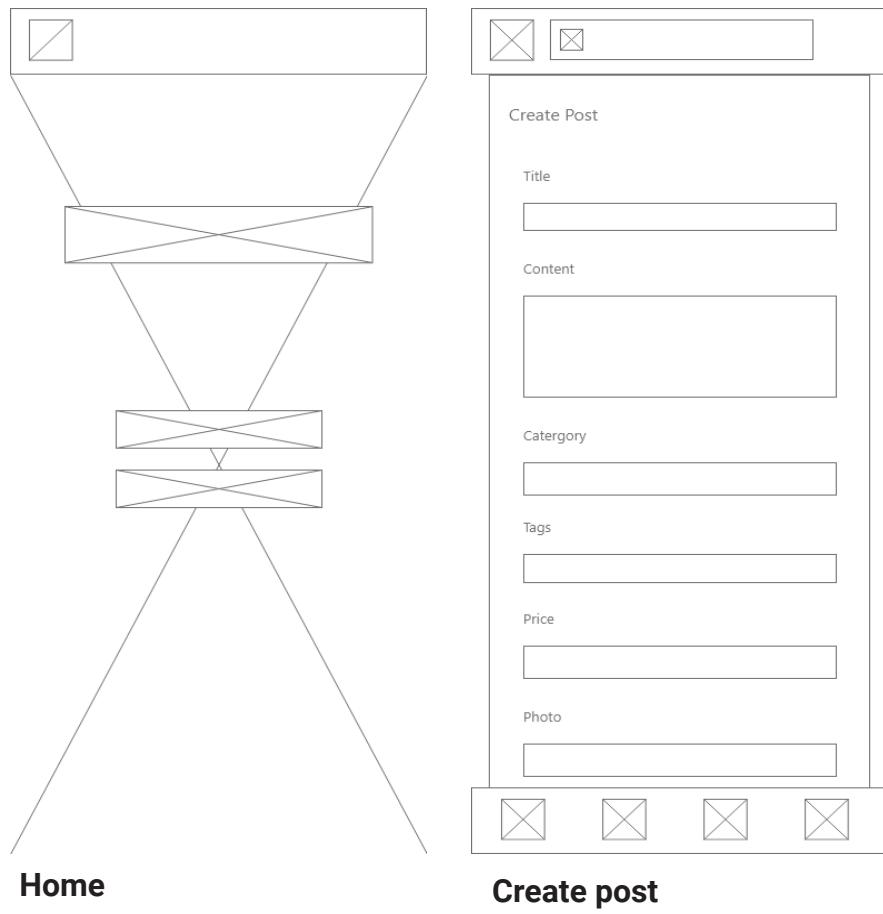
- Header bar with 'X' icon.
- Title: "Log in".
- Text input field labeled "Email".
- Text input field labeled "Password".
- Text input field labeled "Sign up" (with a large 'X' icon).

Register screen:

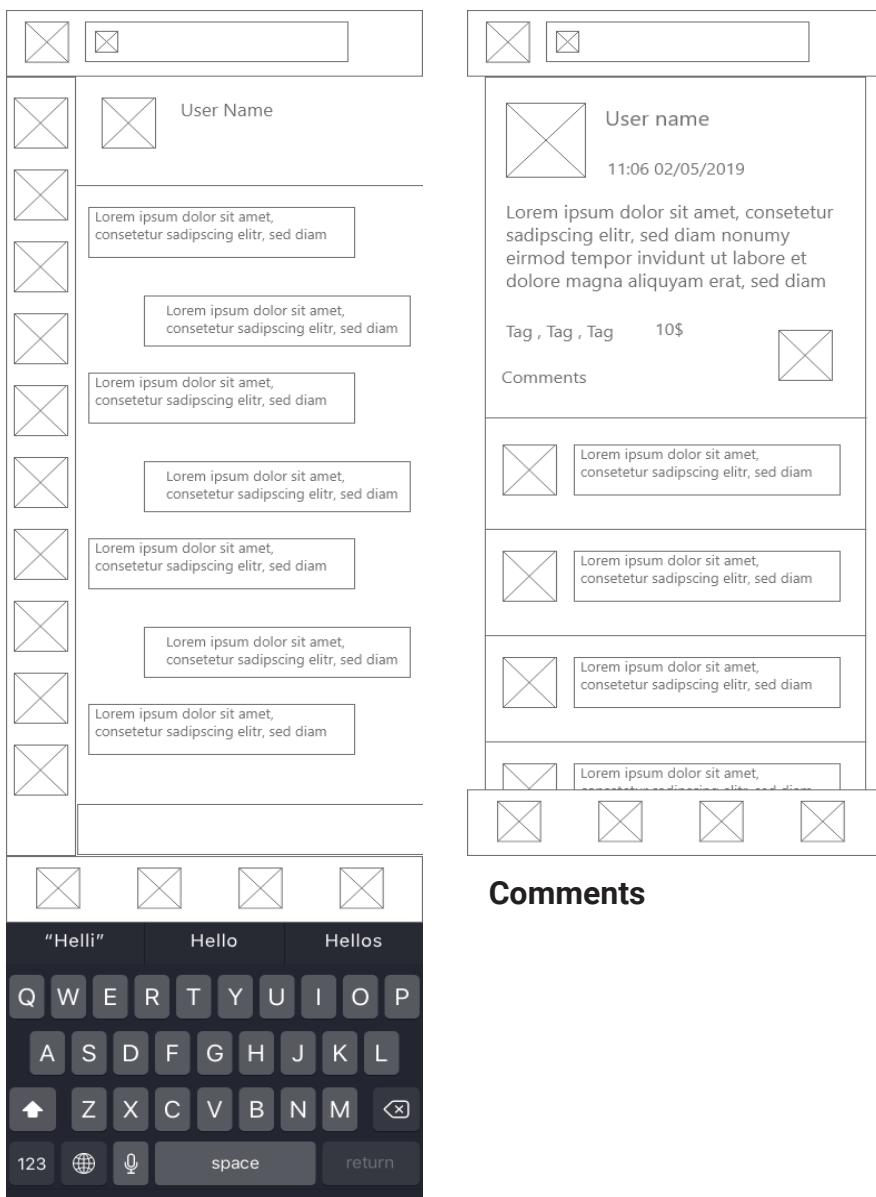
- Header bar with 'X' icon.
- Title: "Sign up".
- Text input field labeled "Name".
- Text input field labeled "Email".
- Text input field labeled "Password".
- Text input field labeled "Confirm Password".
- Text input field labeled "Upload Photo" (with a checked checkbox icon).
- Text input field labeled "Log in" (with a large 'X' icon).

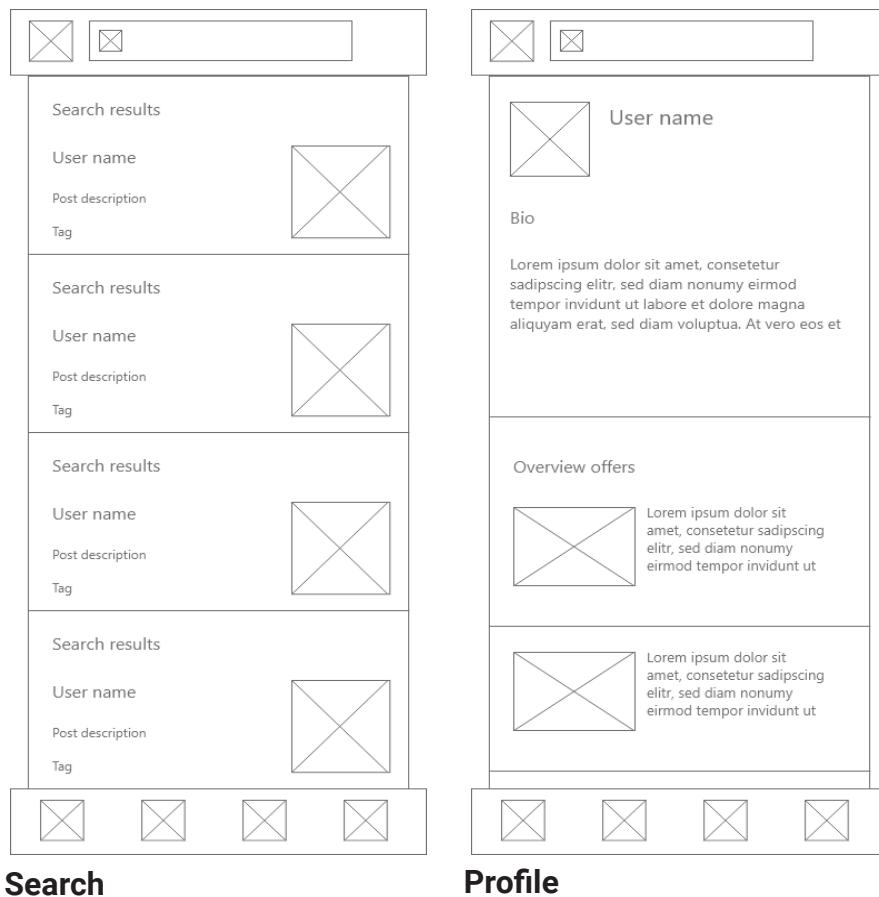
Log in

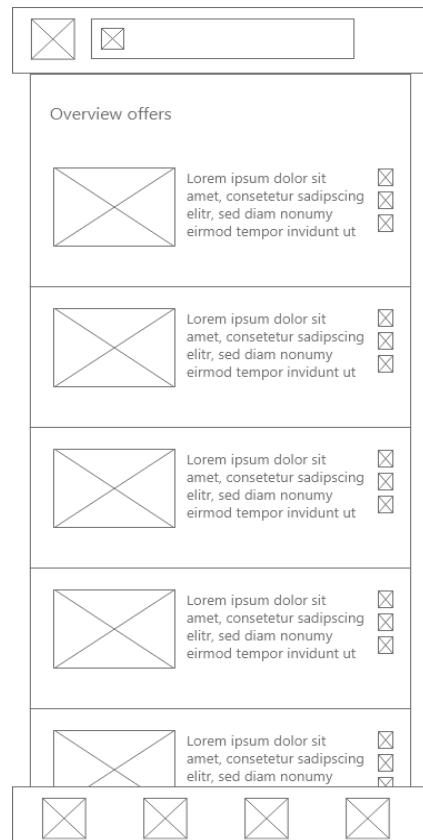
Register

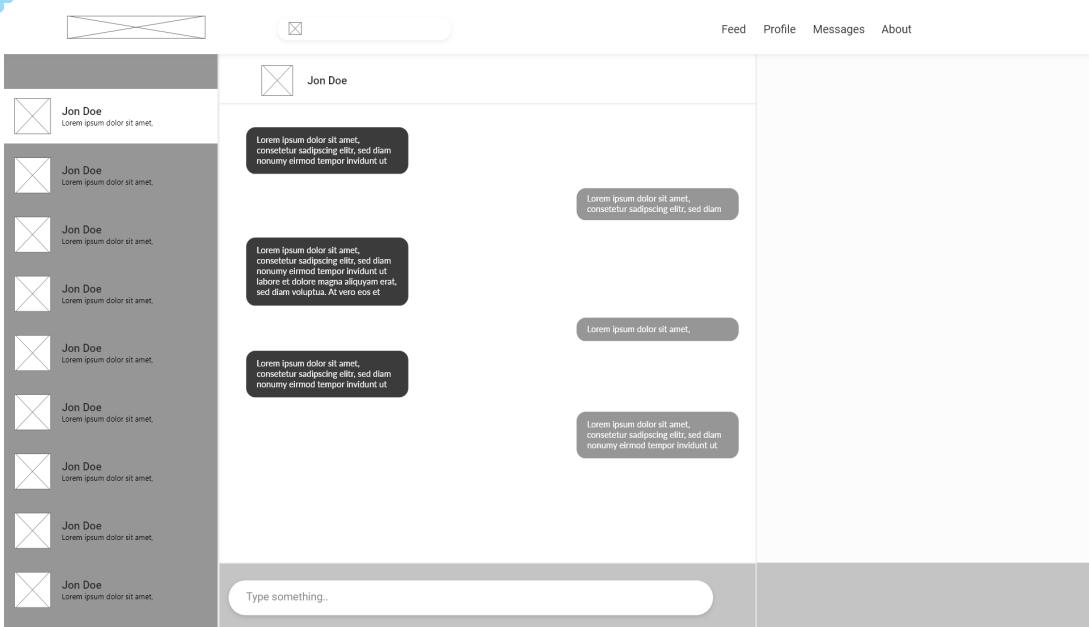


Chat









Chat

This wireframe shows a form for creating a new listing. At the top right are navigation links: Feed, Profile, Messages, and About. A prominent header bar reads 'Create a new listing'. Below it, the form begins with a question: 'I want to create a new' followed by two radio button options: 'Offer' (selected) and 'Request'. The next section, 'Give your post a title', contains a 'Title' input field. To its right, under 'Price', is a 'Price' input field. The following sections are 'Give us more information' (with a 'Description' input field) and 'Tags' (with a 'Tag' input field). At the bottom, there is a section for 'Add a picture to your post' with a '+ Add Photo' button.

Create post

The Offers screen displays three offers from a user named Alexa Adams. Each offer card includes a placeholder image, the user's name, the creation date, a short description, the category, a price, and interaction buttons.

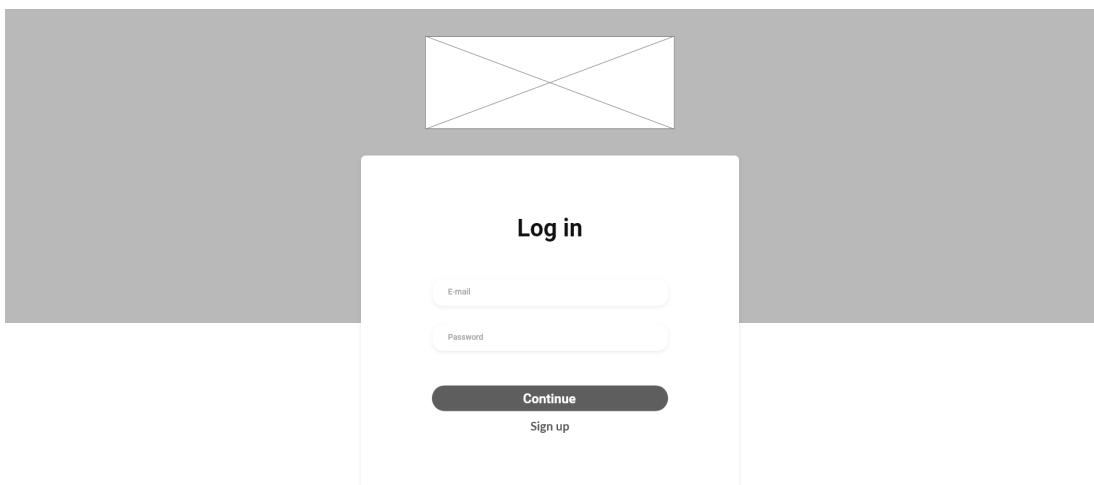
Offer Details	Category	Price	Action Buttons
Alexa Adams 11:06 02/05/2019 Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo. Italian - Fresh	Italian - Fresh	10\$	Send message View Comments
Alexa Adams 11:06 02/05/2019 Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo. Italian - Fresh	Italian - Fresh	10\$	Send message View Comments
Alexa Adams 11:06 02/05/2019 Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo.	Italian - Fresh	10\$	Send message View Comments

Feed

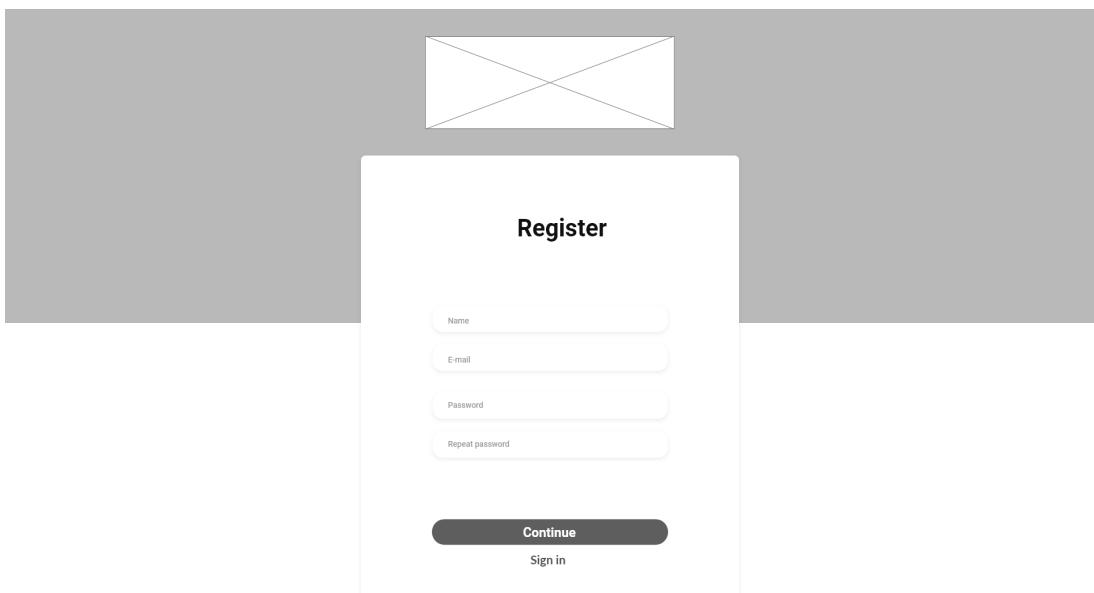
The Feed screen shows a summary of a user's profile (Jon Doe) on the left and a grid of offers on the right. Each offer card follows a similar structure to the Offers screen.

Profile Summary	Offer Grid
Jon Doe Bio Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo.	<div style="display: flex; justify-content: space-around;"> <div style="text-align: center;"> <p>Overview offers</p> <p>Taco Take Away</p> <p>Italian - Fresh</p> <p>FREE</p> </div> <div style="text-align: center;"> <p>Overview offers</p> <p>Taco Take Away</p> <p>Italian - Fresh</p> <p>FREE</p> </div> <div style="text-align: center;"> <p>Overview offers</p> <p>Taco Take Away</p> <p>Italian - Fresh</p> <p>FREE</p> </div> <div style="text-align: center;"> <p>Overview offers</p> <p>Taco Take Away</p> <p>Italian - Fresh</p> <p>FREE</p> </div> <div style="text-align: center;"> <p>Overview offers</p> <p>Taco Take Away</p> <p>Italian - Fresh</p> <p>FREE</p> </div> </div>

Profile



Log in



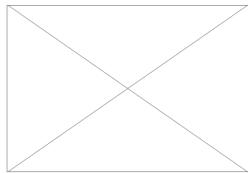
Register



Feed Profile Messages About



Jon Doe



Free Taco Giveaway

Taco is a traditional Mexican dish consisting of a corn or wheat tortilla folded or rolled around a filling. A taco can be made with a variety of fillings, including beef, pork, chicken, seafood, vegetables, and cheese, allowing great versatility and variety.

Send Message

Ingredients

Lettuce • Chicken • Cheese • Pork

Mexican

Taco • Meat

Similar offers



Taco Take Away

Loren ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore

Taco Take Away

Loren ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore

Taco Take Away

Loren ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore

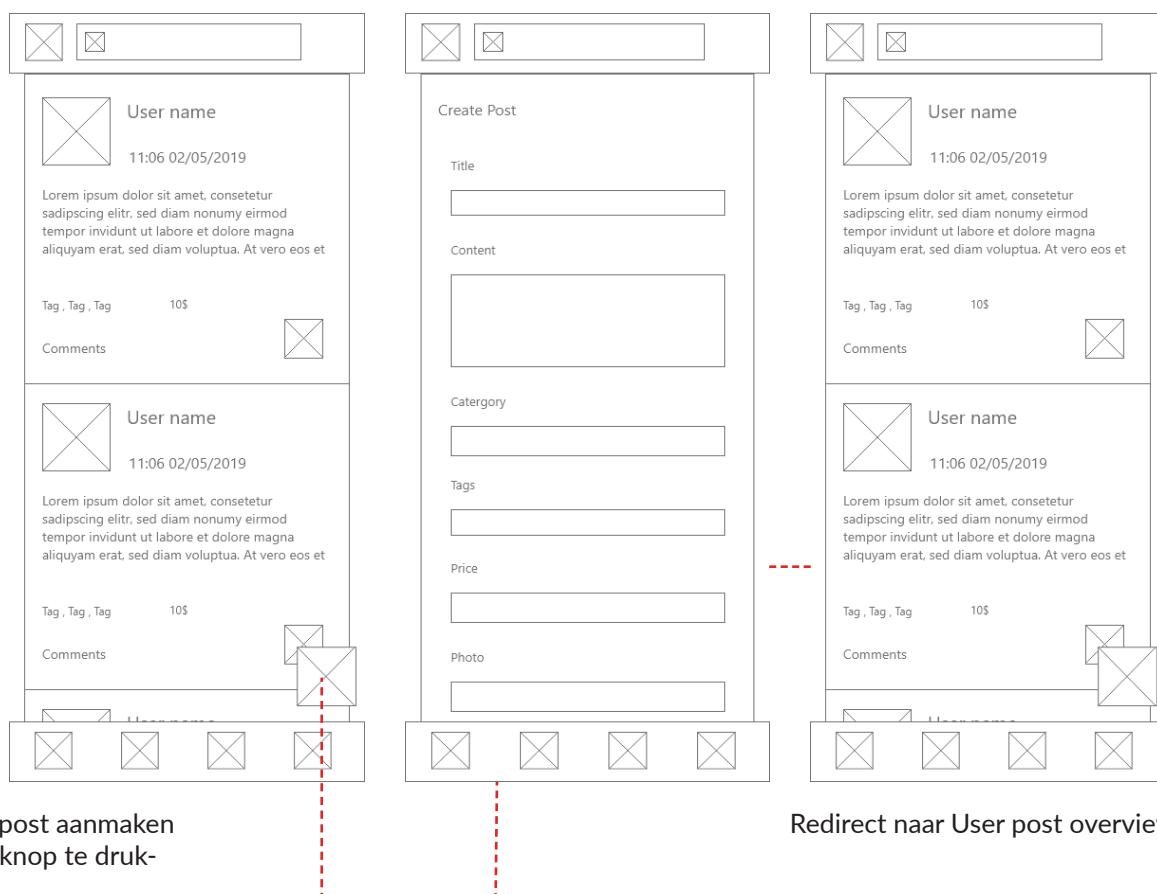
Taco Take Away

Loren ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore

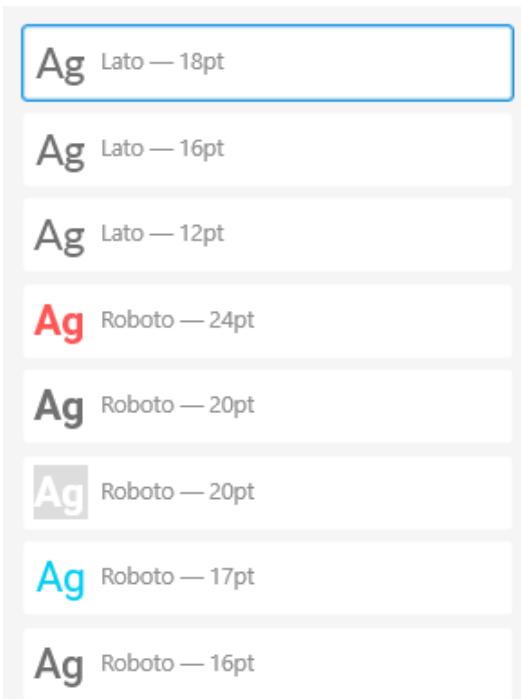
Single post

Wireflow

Alle velden invullen en op “create” drukken



Style tile



limento



Title



Create Post

Style Guide

Icons



20 x 20 px

40 x 40 px

Fonts

#Header 1 Roboto bold 24 #FF5555

#Header 2 Roboto regular 18 #00CEF7

#BasicText Lato 14 #707070

#ingredients lato 12 #707070

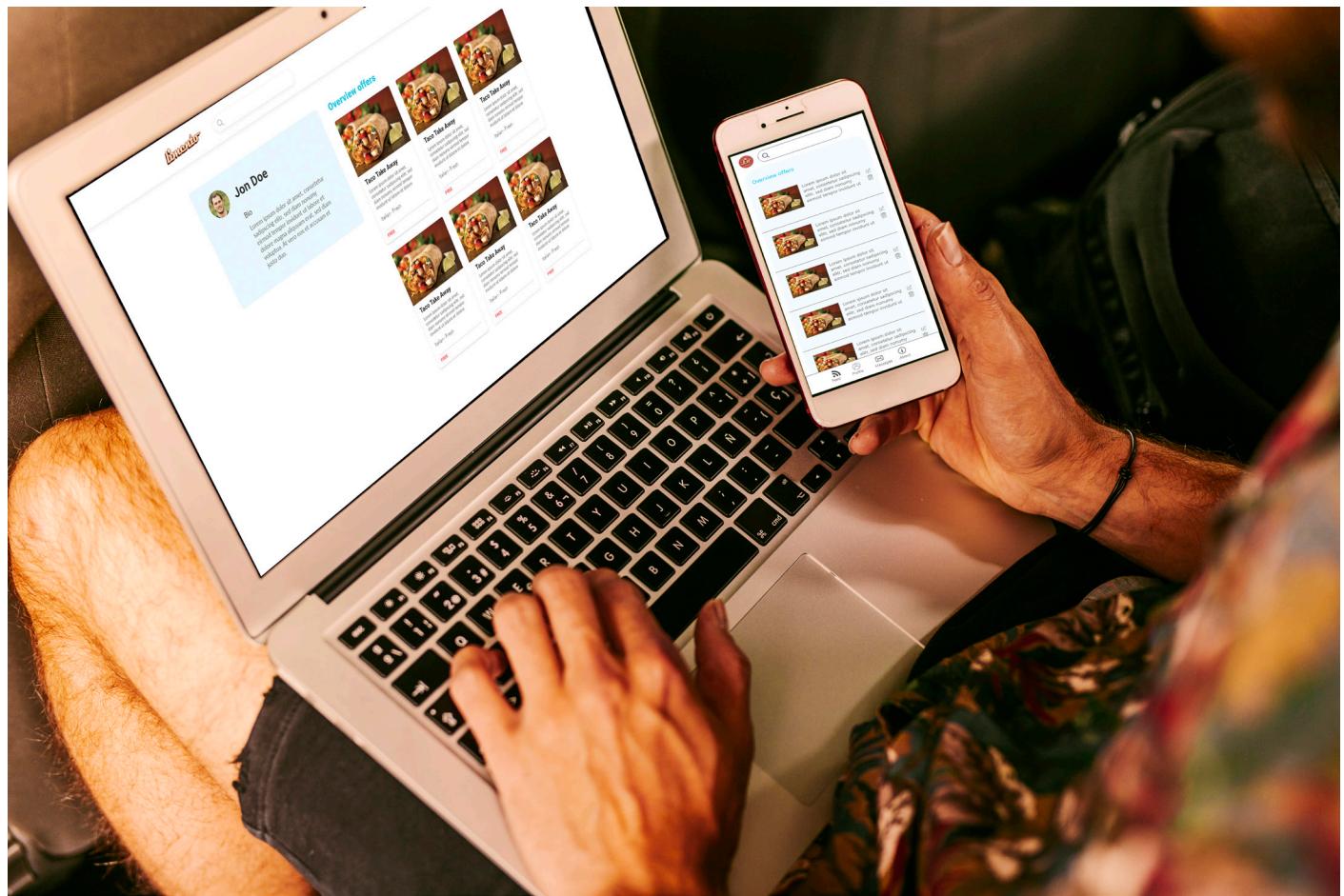
#Button roboto bold 20 #FFFFFF

Button

258 x 58 px

#FF5555

Mockup



Visual designs



The image shows the login screen of the Limento app. The background features a photograph of several hands holding glasses filled with a golden liquid, likely beer, in a toast. Overlaid on the image is the text "Welcome to Limento". Below this, there is a question: "Did you make too much food and feel like sharing or selling your specialty?". A descriptive text follows: "Limento is a platform made to boost up social interaction and help the local economy". At the bottom, there are two red buttons with white text: "Login" and "Register".

Home

Login

Log in

Email

Password

Continue

Sign up

The image consists of two main sections. On the left, there is a 'Sign up' form with fields for Name, Email, Bio, and Password, followed by a red 'Continue' button. Below the form is a 'Log in' link. On the right, there is a feed interface featuring a search bar and navigation icons. It displays two posts from users Alexa Adams and Dries Tijbergijn, each with a profile picture, name, timestamp, a preview of the post content, and interaction buttons for comments and messages.

Sign up

Name

Email

Tell others about yourself...

Password

Continue

[Log in](#)

f

≡

Offers
Requests

Alexa Adams
11:06 02/05/2019

Lore ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et

Tag , Tag , Tag
10\$

Comments
 Send message

Dries Tijbergijn
11:06 02/05/2019

Lore ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et

Tag , Tag , Tag
10\$

Comments
 Send message

Sign up

Feed



Jon Doe

Free Taco Take Away



Taco is a traditional Mexican dish consisting of a corn or wheat tortilla folded or rolled around a filling. A taco can be made with a variety of fillings, including beef, pork, chicken, seafood, vegetables, and cheese, allowing great versatility and variety.

Ingredients	Mexican
-Lettuce -Chicken -Cheese -Pork	Taco , Meat Free

Send message



Jon Doe

Bio

Lore ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo.

Overview offers



Lore ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut



Lore ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut



Lore ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut

Single post

Profile overview



Search results

Title

Jon Doe

Lore ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et

Title

Jon Doe

Lore ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et

Title

Jon Doe

Lore ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et



Overview offers


Lore ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut




Lore ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut




Lore ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut




Lore ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut



Search results

-  Feed
-  Profile
-  Messages
-  About

Overview posts

-  Feed
-  Profile
-  Messages
-  About

Chat

The image displays two side-by-side screenshots of a mobile application interface, likely a messaging or social media platform.

Screenshot 1 (Left): This screenshot shows a conversation with "Jon Doe". The message history is as follows:

- Jon Doe: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo.
- Tag , Tag , Tag 10\$
- Comments 5 Send message
- Comment 1: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam
- Comment 2: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam
- Comment 3: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam
- Comment 4: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam
- Comment 5: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam

Screenshot 2 (Right): This screenshot shows a list of messages from "Jon Doe". The messages are:

- Jon Doe: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam
- Jon Doe: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam
- Jon Doe: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam
- Jon Doe: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam
- Jon Doe: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam
- Jon Doe: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam
- Jon Doe: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam
- Jon Doe: Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam

Bottom Navigation Bar:

- Feed
- Profile
- Messages
- About

Keyboard View:

"Hello" Hello Hellos

Q	W	E	R	T	Y	U	I	O	P
A	S	D	F	G	H	J	K	L	
123	Z	X	C	V	B	N	M	return	

404

PAGE NOT FOUND

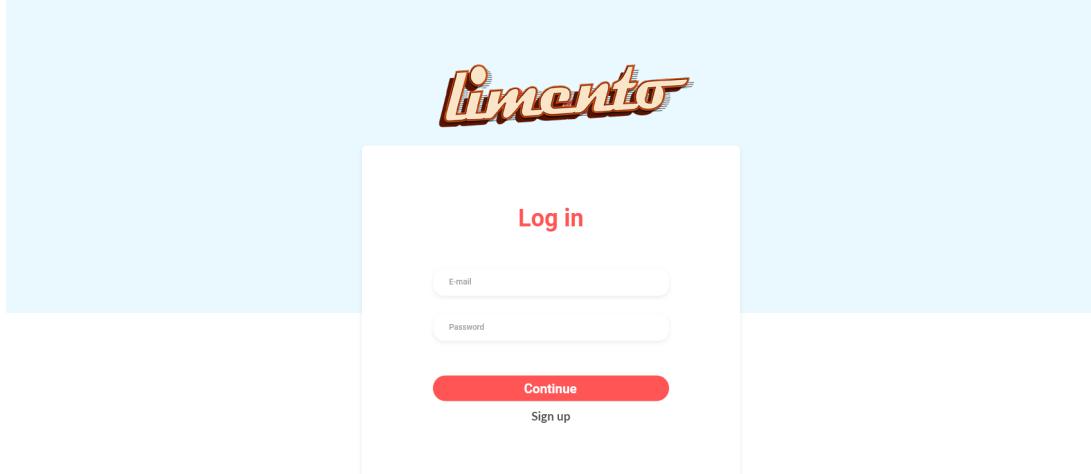
Feed Profile Messages About

About us

Mission

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo. eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo.

Feed Profile Messages About



Log in

The image shows the Limento profile page for a user named "Jon Doe". At the top is the Limento logo and a search bar. Below is a navigation bar with links: Feed, Profile, Messages, and About. The main content area starts with a profile picture and the name "Jon Doe". Below this is a "Bio" section with placeholder text. To the right is a "Overview offers" section showing six food items in a grid. Each item has a thumbnail image, the name "Taco Take Away", a short description, the category "Italian - Fresh", and a "FREE" label.

Offer	Description	Category	Status
Taco Take Away	Placeholder text	Italian - Fresh	FREE
Taco Take Away	Placeholder text	Italian - Fresh	FREE
Taco Take Away	Placeholder text	Italian - Fresh	FREE
Taco Take Away	Placeholder text	Italian - Fresh	FREE
Taco Take Away	Placeholder text	Italian - Fresh	FREE
Taco Take Away	Placeholder text	Italian - Fresh	FREE

Profile

Feed

The screenshot shows a social media feed with three posts from user Alexa Adams. Each post includes a profile picture, the user's name, the date (11.06.02.05.2019), a placeholder text message, the category (Italian - Fresh), the price (10\$), a 'Send message' button, and a 'View Comments' link.

Post 1:
Alexa Adams
11.06.02.05.2019
Lorem ipsum dolor sit amet, consetetur sadipscing elitr,
sed diam nonumy eirmod tempor invidunt ut labore et
dolore magna aliquyam erat, sed diam voluptua. At vero
eos et accusam et justo duo.
Italian - Fresh 10\$
[Send message](#) View Comments

Post 2:
Alexa Adams
11.06.02.05.2019
Lorem ipsum dolor sit amet, consetetur sadipscing elitr,
sed diam nonumy eirmod tempor invidunt ut labore et
dolore magna aliquyam erat, sed diam voluptua. At vero
eos et accusam et justo duo.
Italian - Fresh 10\$
[Send message](#) View Comments

Post 3:
Alexa Adams
11.06.02.05.2019
Lorem ipsum dolor sit amet, consetetur sadipscing elitr,
sed diam nonumy eirmod tempor invidunt ut labore et
dolore magna aliquyam erat, sed diam voluptua. At vero
eos et accusam et justo duo.

Single post

The screenshot shows a single post by user Jon Doe. It features a profile picture, the user's name (Jon Doe), a large image of a filled tortilla wrap, and a title 'Free Taco Giveaway'. Below the image is a descriptive text about what a taco is, followed by a 'Send Message' button. At the bottom, there are sections for 'Ingredients' (Lettuce, Chicken, Cheese, Pork) and 'Mexican' (Taco, Meat).

Post by Jon Doe:
Free Taco Giveaway
Taco is a traditional Mexican dish consisting of a corn or wheat tortilla folded or rolled around a filling. A taco can be made with a variety of fillings, including beef, pork, chicken, seafood, vegetables, and cheese, allowing great versatility and variety.
[Send Message](#)

Ingredients:
Lettuce • Chicken • Cheese • Pork

Mexican:
Taco • Meat

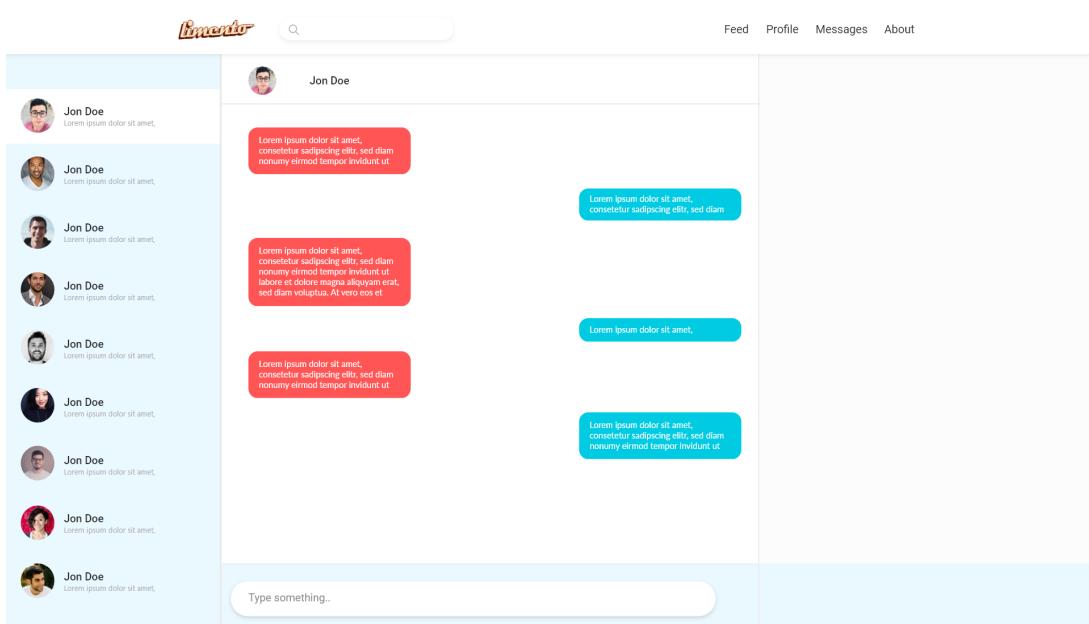
Similar offers



Create post

The screenshot shows the 'Create post' section of the Limento app. At the top, there's a navigation bar with the Limento logo, a search icon, and links for Feed, Profile, Messages, and About. Below the navigation is a light blue header bar with the text 'Create a new listing'. The main form area has several input fields: 'I want to create a new' (Offer or Request), 'Give your post a title' (Title and Price fields), 'Give us more information' (Description and Tags fields), and a photo upload section ('Add a picture to your post' with '+ Add Photo').

Chat



Develop

Backend

Controller for user posts overview

```
class UserController {
  // List all the models
  index = async (req, res, next) => {
    try {
      const { limit, skip } = req.query;
      let posts = null;
      if (limit && skip) {
        const options = {
          page: parseInt(skip, 10) || 1,
          limit: parseInt(limit, 10) || 10,
          populate: 'posts',
          sort: { created_at: -1 },
        };
        posts = await User.paginate({}, options);
      } else {
        posts = await User.find().populate('posts').sort({ created_at: -1 }).exec();
      }
      if (posts === undefined || posts === null) {
        throw new APIError(404, 'Collection for posts not found!');
      }
      return res.status(200).json(posts);
    } catch (err) {
      return handleAPIError(500, err.message || 'Some error occurred while retrieving posts', next);
    }
  };
}
```

Post controller, function to filter one specific post by ID

```
// Show specific model by id
show = async (req, res, next) => {
  try {
    const { id } = req.params;
    const item = await Post.findById(id)
      .populate('category')
      .populate('user')
      .populate('type')
      .populate('tags')
      .populate('media').exec();
    if (item === undefined || item === null) {
      throw new APIError(404, `Post with id: ${id} not found!`);
    }
    return res.status(200).json(item);
  } catch (err) {
    return handleAPIError(err.status || 500, err.message || 'Some error occurred while retrieving posts', next);
  }
};
```

Routes

Post

```
JS post.routes.js ✘
1  /*
2  Import the internal libraries:
3  - PostController
4  */
5  import { PostController } from '../controller';
6
7 // Create instance of PostController otherwise you can't use it
8 const postController = new PostController();
9
10 const initializeEndpoints = (parentRouter, authService) => {
11   /**
12    * @swagger
13    * /api/v1/posts:
14    *   get:
15    *     tags:
16    *       - Posts
17    *     description: Returns all posts
18    *     produces:
19    *       - application/json
20    *     responses:
21    *       200:
22    *         description: An array of posts
23    */
24   parentRouter.get('/posts', postController.index);
25   /**
26    * @swagger
27    * /api/v1/posts/create:
28    *   get:
29    *     tags:
30    *       - Post
31    *     description: Returns specific viewmodel such as categories
32    *     produces:
33    *       - application/json
34    *     responses:
35    *       200:
36    *         description: Create post
37    */

```

Database

```
const BlogSchema = new Schema(
{
  title: { type: String, required: true, max: 128 },
  description: { type: String, required: true, max: 512 },
  slug: {
    type: String, lowercase: true, unique: true, required: true,
  },
  published_at: { type: Date, required: false },
  deleted_at: { type: Date, required: false },
  categoryId: { type: Schema.Types.ObjectId, ref: 'Category', required: false },
  posts: [{ type: Schema.Types.ObjectId, ref: 'Post', required: false }],
},
{
  toJSON: { virtuals: true },
  toObject: { virtuals: true },
},
{
  timestamps: { createdAt: 'created_at', updatedAt: 'updated_at' },
},
);
}

const CategorySchema = new Schema(
{
  name: { type: String, required: true, max: 128 },
  description: { type: String, required: true, max: 512 },
  slug: {
    type: String, lowercase: true, unique: true, required: true,
  },
  published_at: { type: Date, required: false },
  deleted_at: { type: Date, required: false },
},
{
  toJSON: { virtuals: true },
  toObject: { virtuals: true },
},
{
  timestamps: { createdAt: 'created_at', updatedAt: 'updated_at' },
},
);
);
```

```

const CommentSchema = new Schema(
{
  author: {
    type: Schema.Types.ObjectId,
    ref: 'User',
    required: true
  },
  rating: {
    type: Number,
    min: 0,
    max: 5,
    required: true,
  },
  body: { type: String, required: true, max: 512 },
  published_at: { type: Date, required: false },
  deleted_at: { type: Date, required: false },
},
{
  toJSON: { virtuals: true },
  toObject: { virtuals: true },
},
{
  timestamps: { createdAt: 'created_at', updatedAt: 'updated_at' },
},
);

```



```

const ConversationSchema = new Schema(
{
  userOne: {
    type: Schema.Types.ObjectId,
    ref: 'User'
  },
  userTwo: {
    type: Schema.Types.ObjectId,
    ref: 'User'
  },
  messages: [
    {
      type: Schema.Types.ObjectId,
      ref: 'Message'
    }
  ],
  published_at: { type: Date, required: false },
  deleted_at: { type: Date, required: false },
},
{
  toJSON: { virtuals: true },
  toObject: { virtuals: true },
},
{
  timestamps: { createdAt: 'created_at', updatedAt: 'updated_at' },
},
);

```

```

const PostSchema = new Schema(
{
  title: { type: String, required: true, max: 128 },
  synopsis: { type: String, required: true, max: 1024 },
  body: { type: String, required: false },
  slug: {
    type: String, lowercase: true, unique: true, required: true,
  },
  price: { type: Number, required: false },
  published_at: { type: Date, required: false },
  deleted_at: { type: Date, required: false },
  media: { type: Schema.Types.ObjectId, ref: 'Media', required: false },
  user: { type: Schema.Types.ObjectId, ref: 'User', required: true },
  type: { type: Schema.Types.ObjectId, ref: 'PostType', required: true },
  category: { type: Schema.Types.ObjectId, ref: 'Category', required: false },
  tags: [{ type: Schema.Types.ObjectId, ref: 'Tag', required: false }]
},
{
  toJSON: { virtuals: true },
  toObject: { virtuals: true },
},
{
  timestamps: { createdAt: 'created_at', updatedAt: 'updated_at' },
},
);
}

const PostTypeSchema = new Schema(
{
  name: { type: String, required: true, max: 128 },
  description: { type: String, required: true, max: 1024 },
  slug: {
    type: String, lowercase: true, unique: true, required: true,
  },
  published_at: { type: Date, required: false },
  deleted_at: { type: Date, required: false },
},
{
  toJSON: { virtuals: true },
  toObject: { virtuals: true },
},
{
  timestamps: { createdAt: 'created_at', updatedAt: 'updated_at' },
},
);

```

```

const MediaSchema = new Schema(
{
  path: { type: String, required: true },
  published_at: { type: Date, required: false },
  deleted_at: { type: Date, required: false },
},
{
  toJSON: { virtuals: true },
  toObject: { virtuals: true },
},
{
  timestamps: { createdAt: 'created_at', updatedAt: 'updated_at' },
},
);

```



```

const MessageSchema = new Schema(
{
  conversation: {
    type: Schema.Types.ObjectId,
    ref: 'Conversation',
    required: true
  },
  sender: {
    type: Schema.Types.ObjectId,
    ref: 'User',
    required: true
  },
  body: { type: String, required: true, max: 512 },
  published_at: { type: Date, required: false },
  deleted_at: { type: Date, required: false },
},
{
  toJSON: { virtuals: true },
  toObject: { virtuals: true },
},
{
  timestamps: { createdAt: 'created_at', updatedAt: 'updated_at' },
},
);

```

```
const UserSchema = new Schema(
{
    avatar: { type: String, required: false },
    name: { type: String, required: true },
    email: {
        type: String,
        required: true,
        trim: true,
        unique: true,
        match: /^\\w+([\\.-]?\\w+)*@\\w+([\\.-]?\\w+)*(\\.\\w{2,3})+$/,
    },
    dateOfBirth: { type: Date, required: false },
    synopsis: { type: String, required: false },
    address: {
        city: {
            type: String,
            required: false,
        },
        street: {
            type: String,
            required: false,
        }
    },
    comments: [{ type: Schema.Types.ObjectId, ref: 'Comment', required: false }],
    localProvider: {
        password: {
            type: String,
            required: false,
            select: false
        },
    },
    facebookProvider: {
        id: { type: String, required: false },
        token: { type: String, required: false },
    },
    published_at: { type: Date, required: false },
    deleted_at: { type: Date, required: false },
},
{
    toJSON: { virtuals: true },
    toObject: { virtuals: true },
},
{
    timestamps: { createdAt: 'created_at', updatedAt: 'updated_at' },
},
```

```
| const TagSchema = new Schema(
| {
|   name: { type: String, required: true, max: 128 },
|   slug: {
|     type: String, lowercase: true, unique: true, required: true,
|   },
|   published_at: { type: Date, required: false },
|   deleted_at: { type: Date, required: false },
| },
| {
|   toJSON: { virtuals: true },
|   toObject: { virtuals: true },
| },
| {
|   timestamps: { createdAt: 'created_at', updatedAt: 'updated_at' },
| },
| );
| );
```

Frontend

Filtering

```
import React, { Component } from 'react';
import { withStyles } from '@material-ui/core/styles';

import Drawer from '@material-ui/core/Drawer';
import Button from '@material-ui/core/Button';
import List from '@material-ui/core/List';
import Divider from '@material-ui/core/Divider';
import ListItem from '@material-ui/core/ListItem';
import ListItemIcon from '@material-ui/core/ListItemIcon';
importListItemText from '@material-ui/core/ListItemText';
import ChevronLeftIcon from '@material-ui/icons/ChevronLeft';
import IconButton from '@material-ui/core/IconButton';
import InboxIcon from '@material-ui/icons/MoveToInbox';
import MailIcon from '@material-ui/icons/Mail';
import Title from './base/title';
import SelectInput from './base/select-input';
import { Api } from '../../services';

const styles = theme => ({
  drawerPaper: {
    width: '75%',
  },
  drawerHeader: {
    display: 'flex',
    alignItems: 'center',
    padding: '0 8px',
    justifyContent: 'flex-start',
  },
});

const filters = [
  { name: 'Date', value: 'date', },
  { name: 'Price', value: 'price', },
  { name: 'Distance', value: 'distance', },
]

class FilterPanel extends Component {
  state = {
    categories: [],
    tags: []
  }

  async componentDidMount() {
    await this.loadCategories();
    await this.loadTags();
  }

  loadCategories = () => {
    Api.findAllCategories()
      .then(res => {
        this.setState({
          categories: res,
        });
      });
  }

  loadTags = () => {
    Api.findAllTags()
      .then(res => {
        this.setState({
          tags: res,
        })
      })
  }
}
```

Thumbnails for posts

```
② PostCardProfile.jsx ✘
1  /*
2  Import external libraries
3  */
4  import React, { Component } from 'react';
5  import classNames from 'classnames';
6
7  import Title from '../../../../../base/title';
8
9  import UserInfo from '../../../../../user-info';
10 import EditIcon from '@material-ui/icons/Edit';
11 import IconButton from '@material-ui/core/IconButton';
12 import Tooltip from '@material-ui/core/Tooltip';
13 import DeleteIcon from '@material-ui/icons/Delete';
14 import { Divider } from '@material-ui/core';
15
16 class PostCardProfile extends Component {
17
18     onEditHandler = (id) => {
19         this.props.onEditHandler(id);
20     }
21
22     onDeleteHandler = (id) => {
23         this.props.onDeleteHandler(id);
24     }
25
26     render() {
27         const { post } = this.props;
28
29         return (
30             <React.Fragment>
31                 <article className="card--profile">
32                     { post.media && <img className="card--profile__thumbnail" src={ post.media.path } /> }
33                     <section className="card--profile__content">
34                         <Title type={5}>{ post.title }</Title>
35                     </section>
36                     <section className="card--profile__meta">
37                         <Tooltip title="Edit">
38                             <IconButton aria-label="Edit">
39                                 <EditIcon onClick={() => this.onEditHandler(post.id)} />
40                             </IconButton>
41                         </Tooltip>
42                         <Tooltip title="Delete">
43                             <IconButton aria-label="Delete">
44                                 <DeleteIcon onClick={() => this.onDeleteHandler(post.id)} />
45                             </IconButton>
46                         </Tooltip>
47                     </section>
48                     </article>
49                     <Divider></Divider>
50                 </React.Fragment>
51             );
52         }
53     }
54
55     export default (PostCardProfile);
```

Deliver

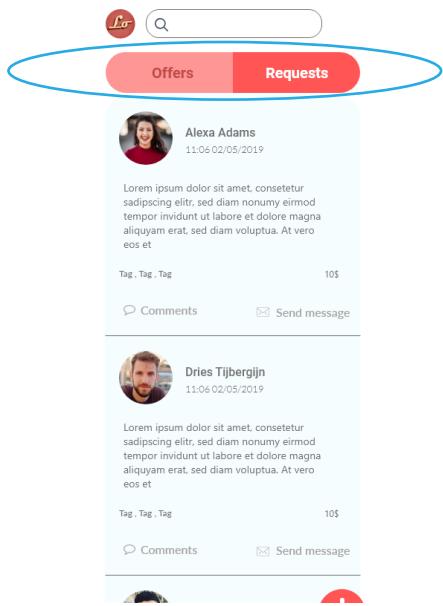
User handleiding



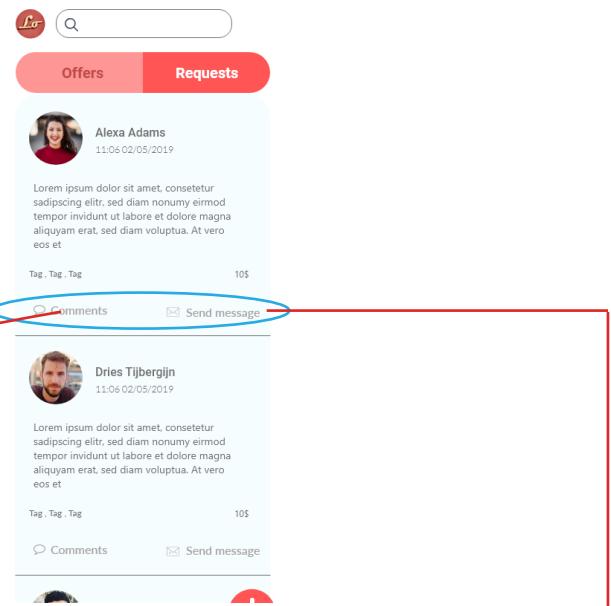
Als niet ingelogde gebruiker heb je de mogelijkheid om een nieuwe account aan te maken. Men kan ook bestande account gebruiken om in te loggen. Daarvoor zijn er twee knoppen voorzien.

A screenshot of the Limento app's sign-up form. It features a red header with the word "Sign up". Below are four input fields: "Name", "Email", "Tell others about yourself...", and "Password". A large red "Continue" button is at the bottom. Below the button, there is a "Log in" link.

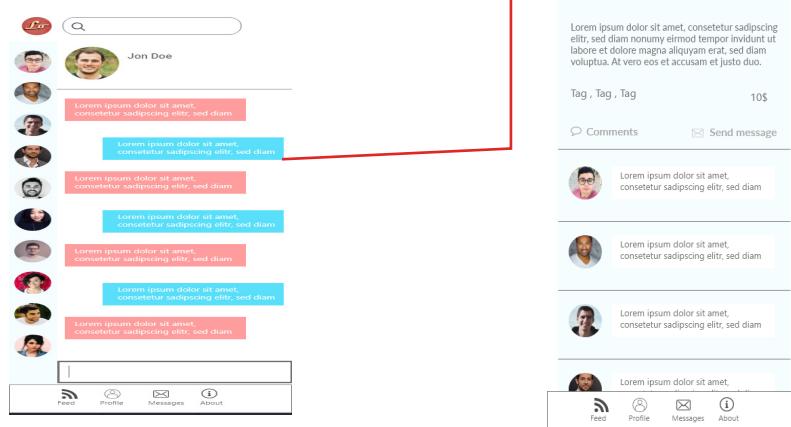
De informatie die de gebruiker ingeeft wordt opgeslagen in onze database. Als alle velden correct ingevuld zijn, mag de gebruiker op "Continue" klikken om verder te gaan naar de feed.

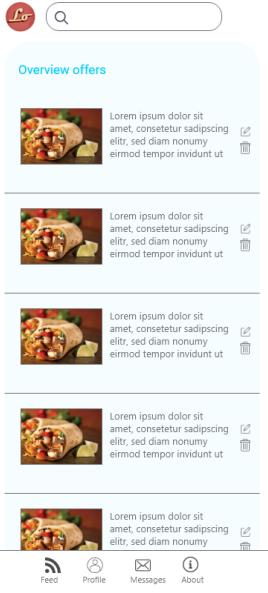


Op het moment dat de gebruiker in de feed zich bevindt, kan hij alle posts bekijken van andere gebruikers. Hij kan kiezen tussen "offers" en "requests". Afhankelijk van zijn keuze, krijgt hij andere posts te zien.

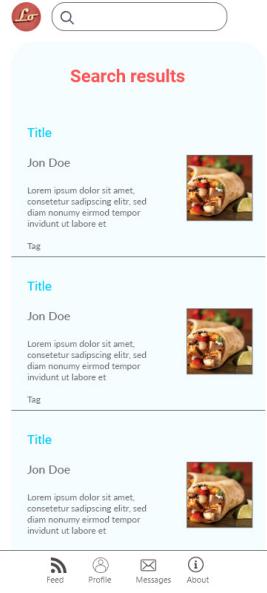


De gebruiker kan ook reacties bekijken en zelf plaatsen, of een bericht sturen naar de eigenaar van de post.





De user kan zijn eigen posts bewerken als hij op “profile” klikt. Dan krijgt hij een overzicht van zijn posts. Deze kunnen bewerkt of verwijderd worden.



De gebruiker kan, op basis van verschillende criteria, posts sorteren. z