limento

Cedric Vanhaegenberg
Juan José Muñoz
Arteveldehogeschool-Grafische & Digitale Media
New Media Development II
2018-2019

Inhoudstafel

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

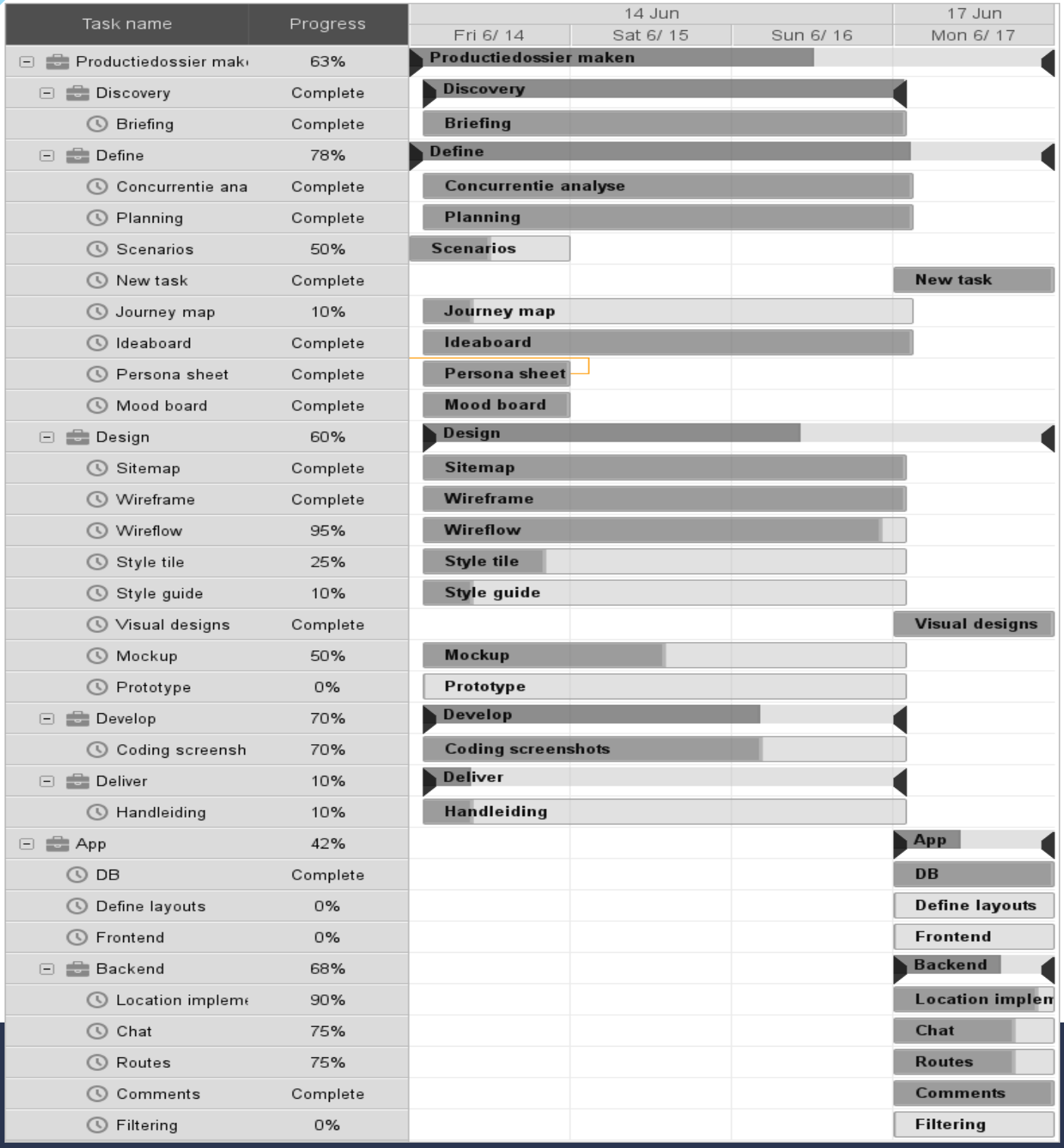
talenkennis : Engels

Inkomen: niet van toepassing

Geboorteplaats : Gent-België

Define

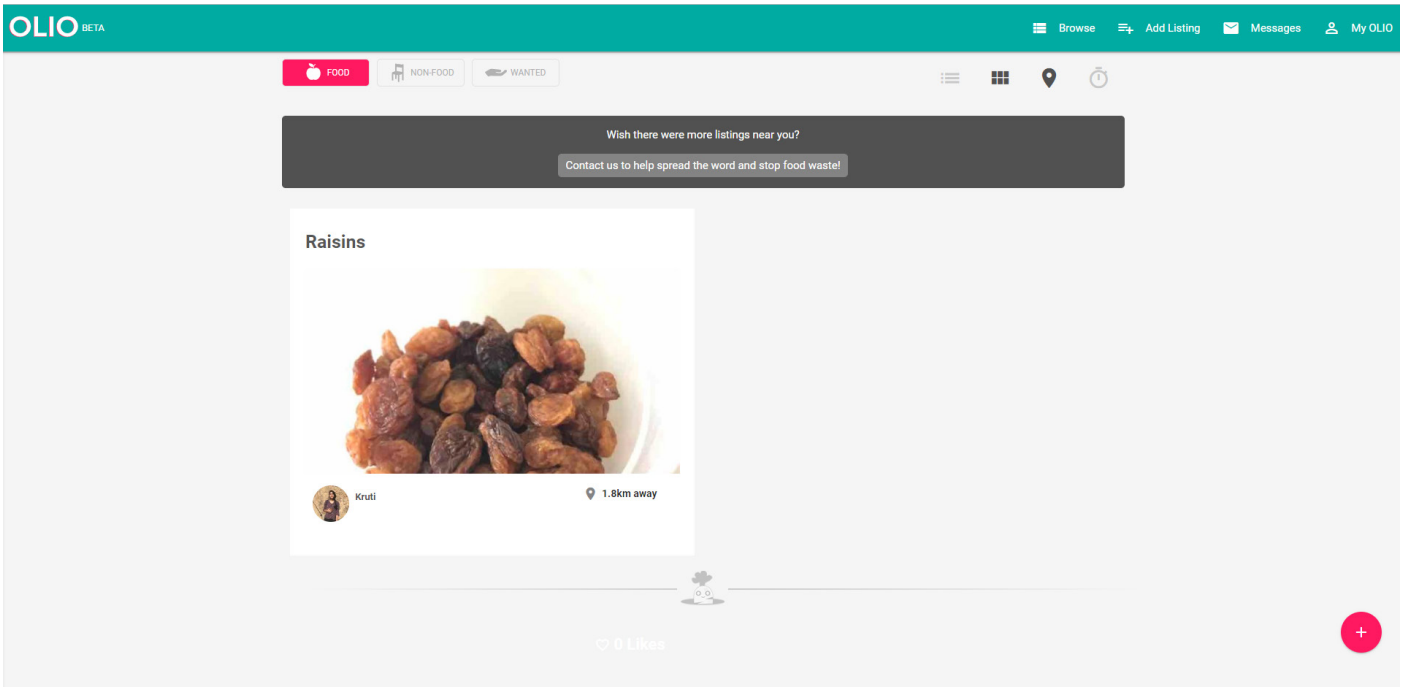
Plannning



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.

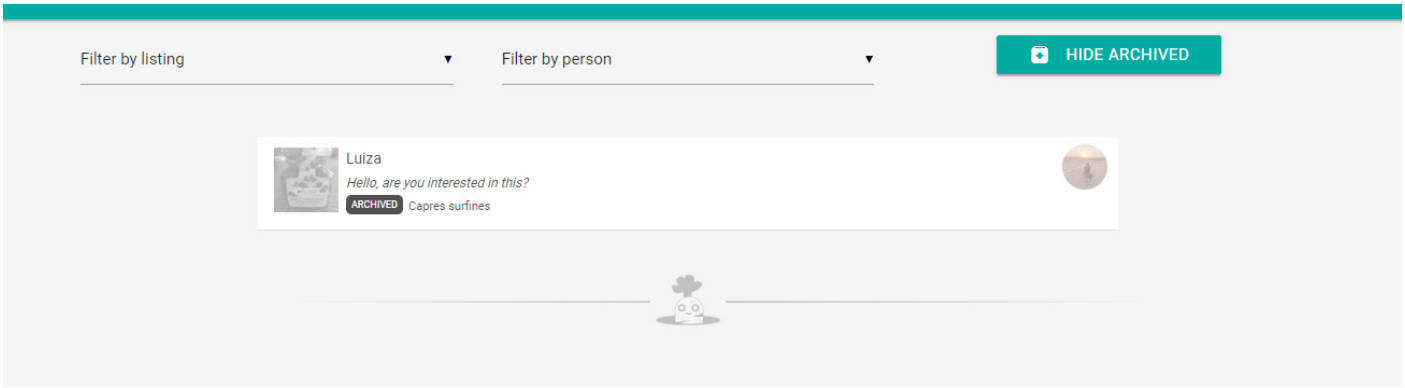


Layout

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 softundelete (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 beveliging.

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

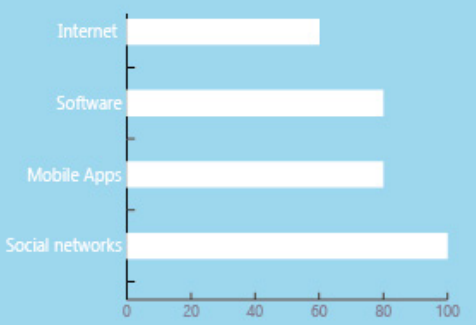
Leeftijd	30
Beroep	Lerares
Status	Single

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

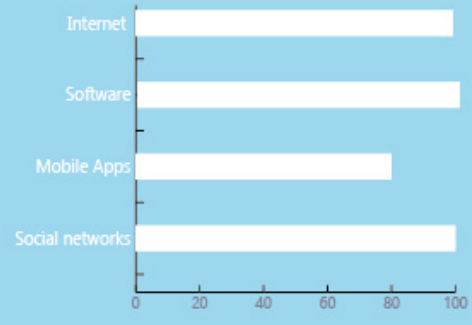
Leeftijd	33
Beroep	Projectmanager
Status	Single

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 gebasseerd op locatie

En ik klik op de knop ‘Filter‘

En ik selecteer afstand

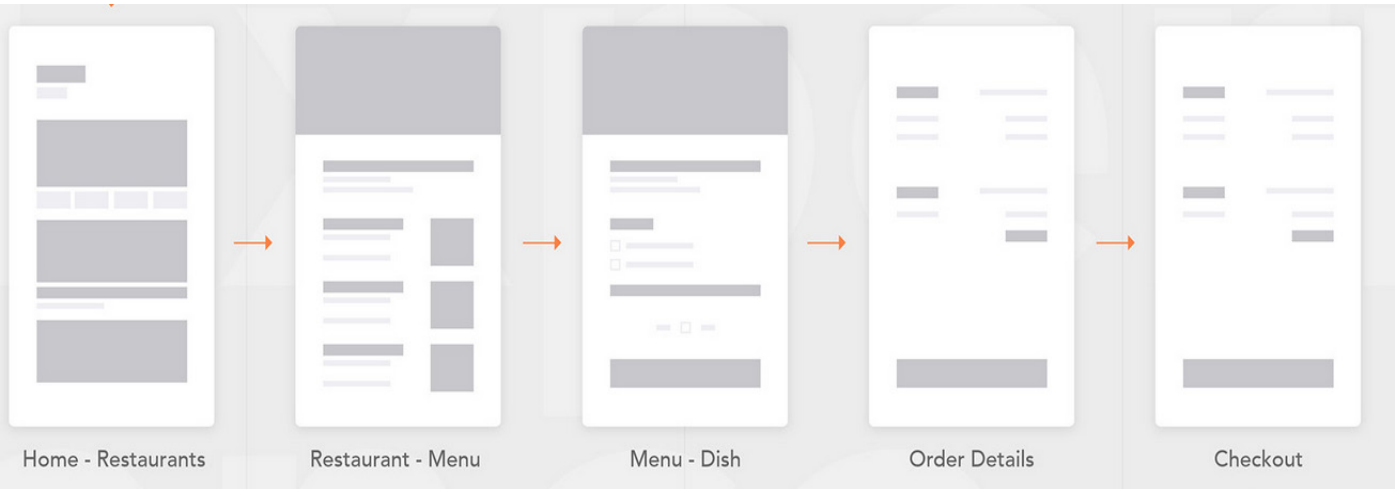
Als ik op de knop ‘Search’ druk

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

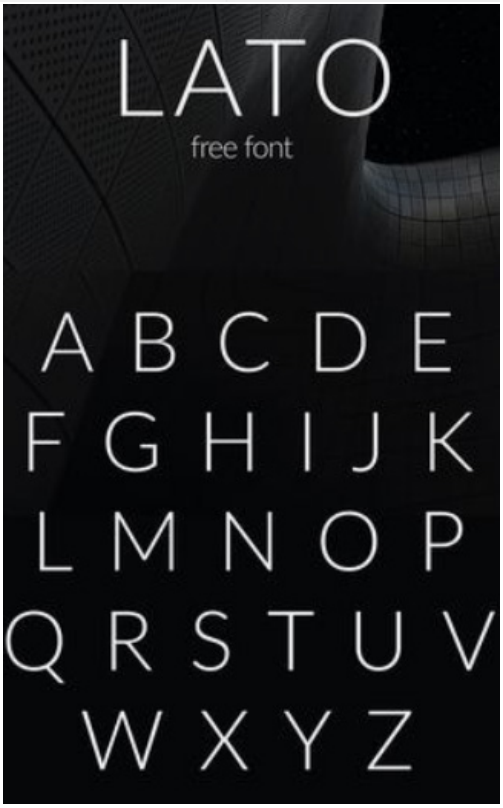
Empathy map

Voorbeeld : Voedsel ruilen en delen

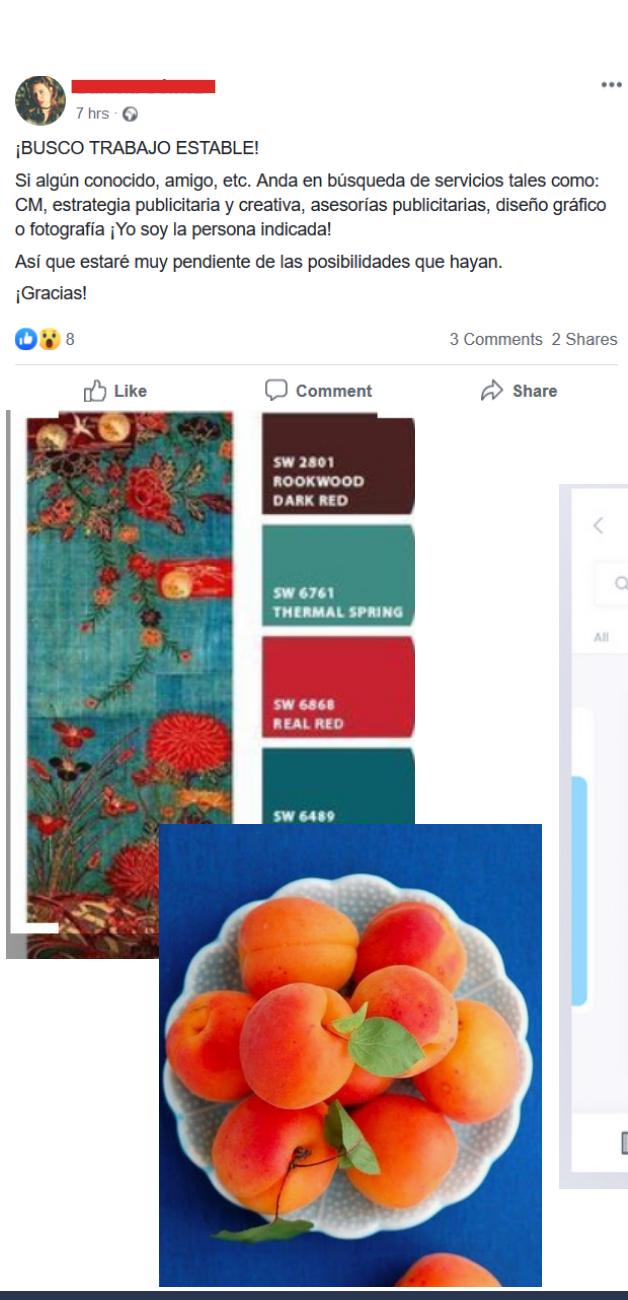
Ideaboard

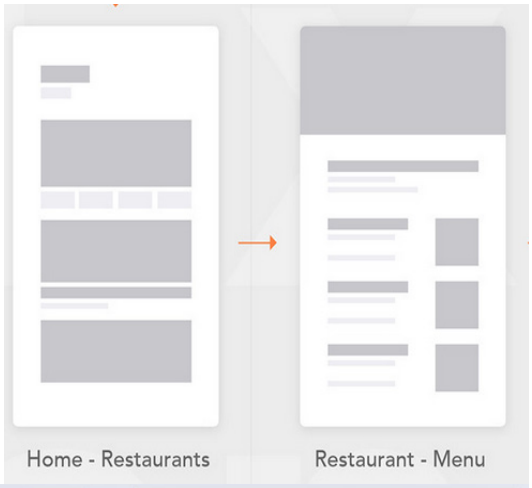

Home - Restaurants
Restaurant - Menu
Menu - Dish
Order Details
Checkout

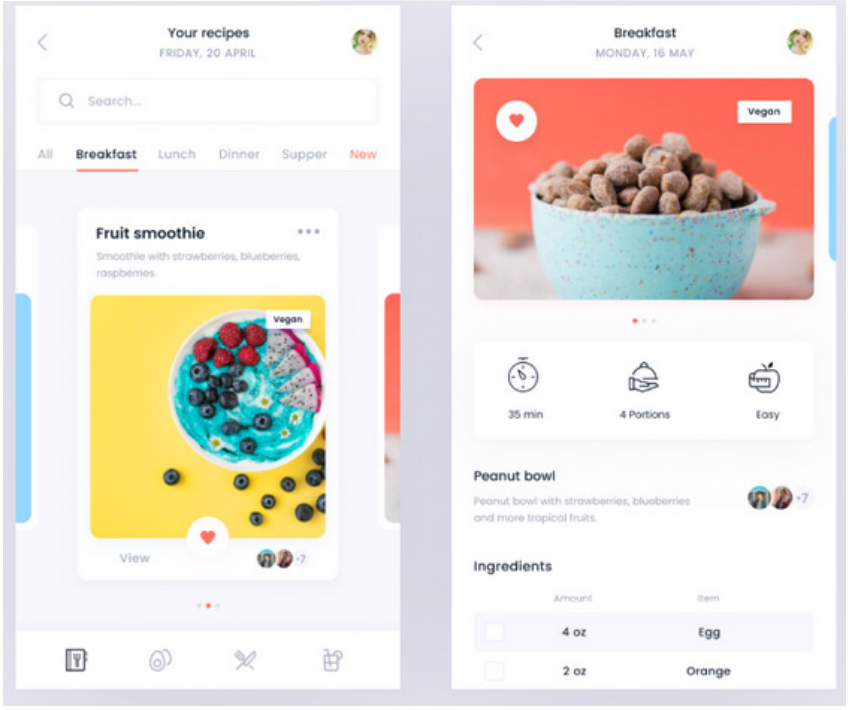

Pizza
Chinese
Burger
Tacos


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

Moodboard

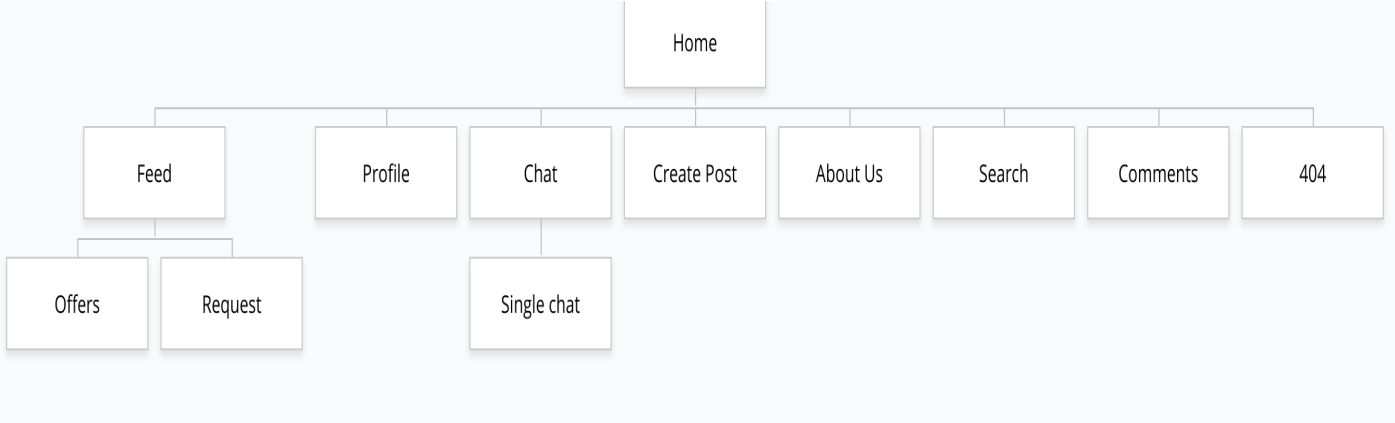

7 hrs
¡BUSCO TRABAJO ESTABLE!
Si algún conocido, amigo, etc. Anda en búsqueda de servicios tales como: CM, estrategia publicitaria y creativa, asesorías publicitarias, diseño gráfico o fotografía ¡Yo soy la persona indicada!
Así que estaré muy pendiente de las posibilidades que hayan.
¡Gracias!
8
3 Comments 2 Shares
Like
Comment
Share
SW 2801 ROOKWOOD DARK RED
SW 6761 THERMAL SPRING
SW 6868 REAL RED
SW 6489


Home - Restaurants
Restaurant - Menu


Your recipes
FRIDAY, 20 APRIL
Search...
All
Breakfast
Lunch
Dinner
Supper
New
Fruit smoothie
Smoothie with strawberries, blueberries, raspberries.
Vegan
View
Breakfast
MONDAY, 16 MAY
Vegan
35 min
4 Portions
Easy
Peanut bowl
Peanut bowl with strawberries, blueberries and more tropical fruits.
Ingredients
Amount
Item
4 oz
Egg
2 oz
Orange

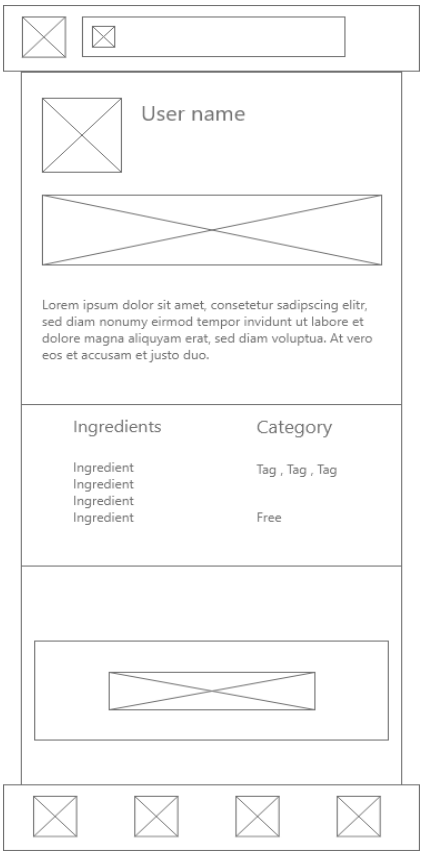
Design

Sitemap

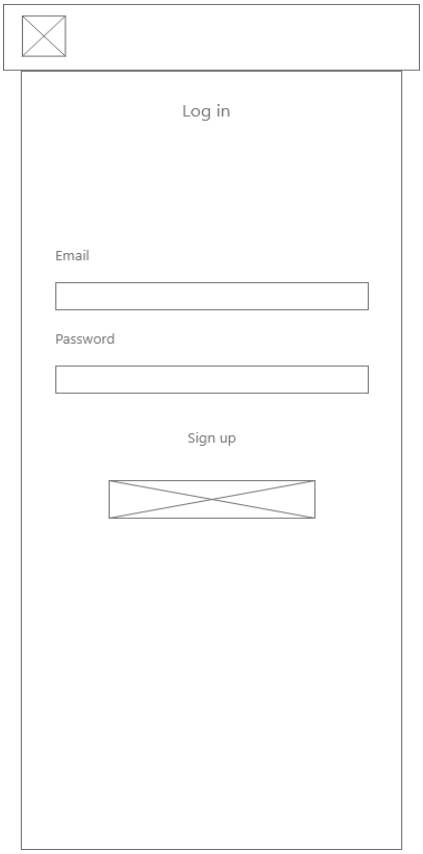


Wireframes

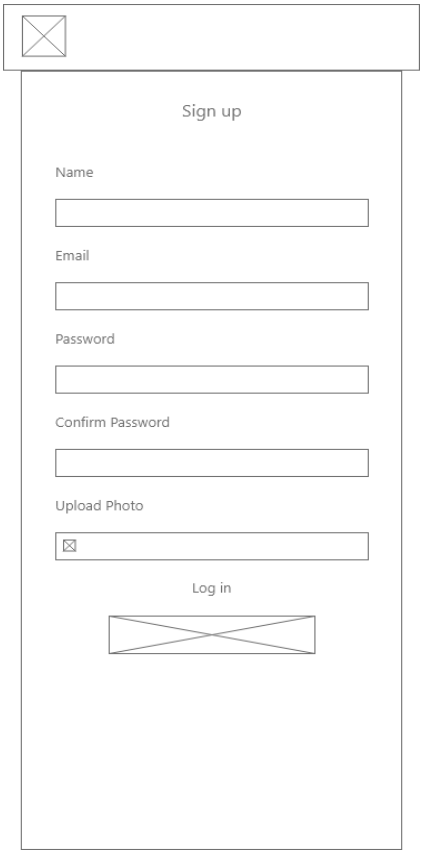
Feed



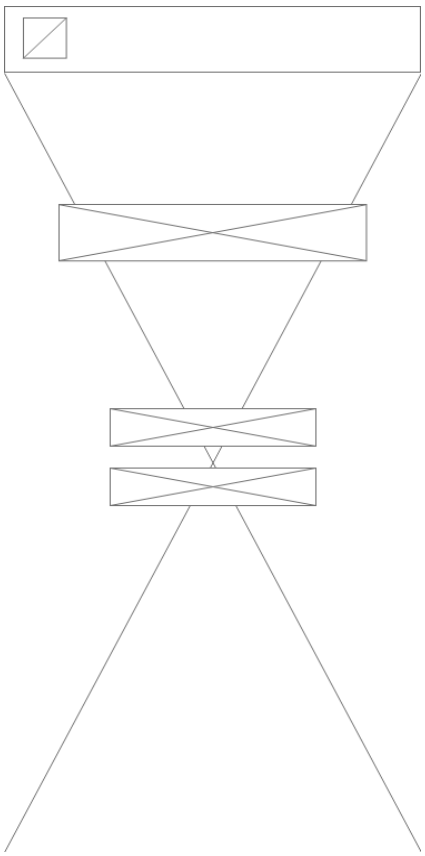
Single post



Log in



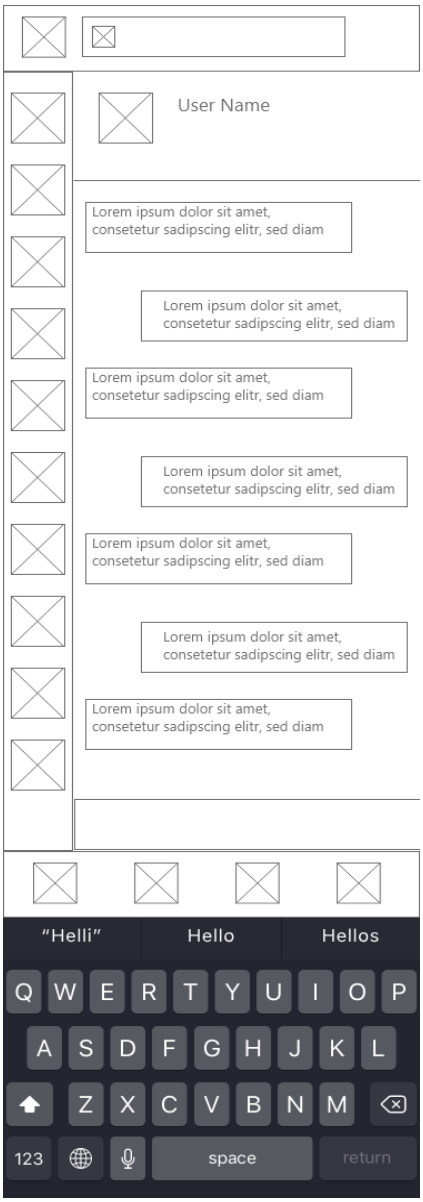
Register



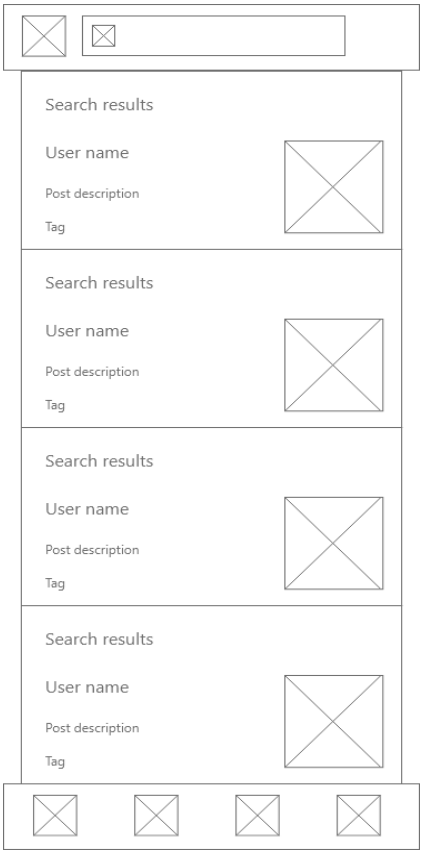
Home

Create post

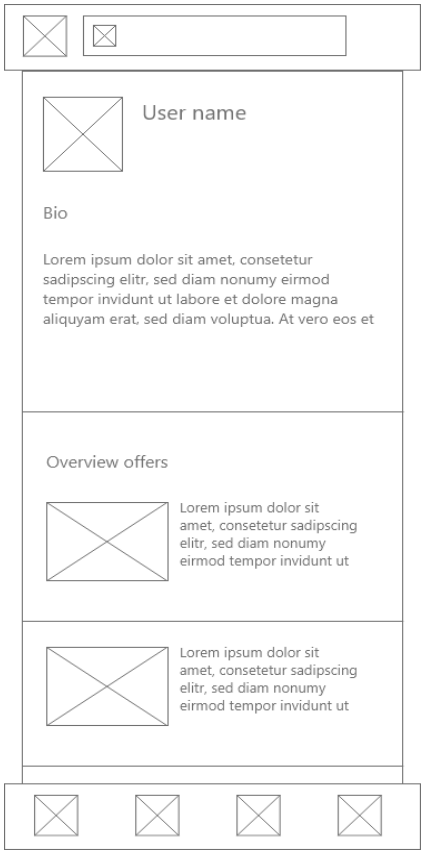
Chat



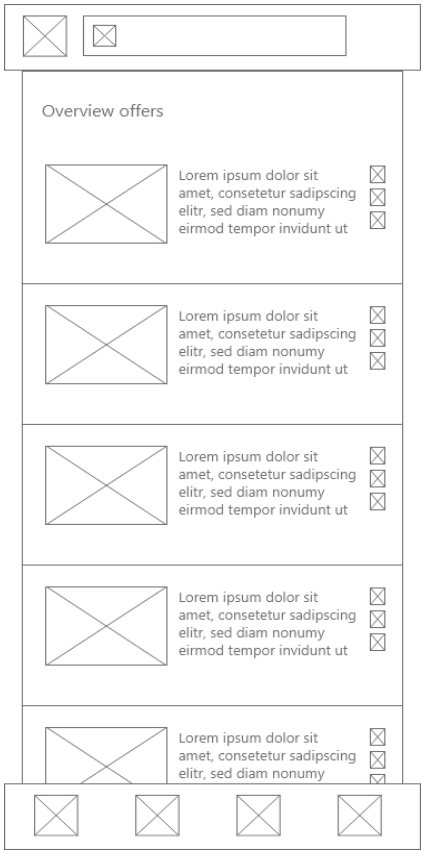
Comments



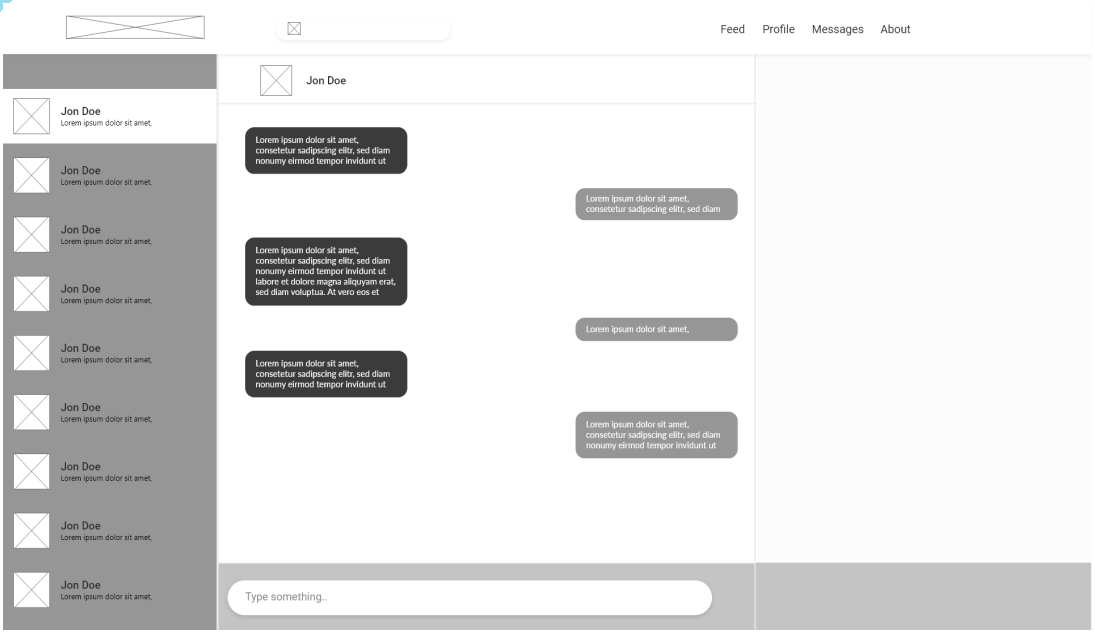
Search



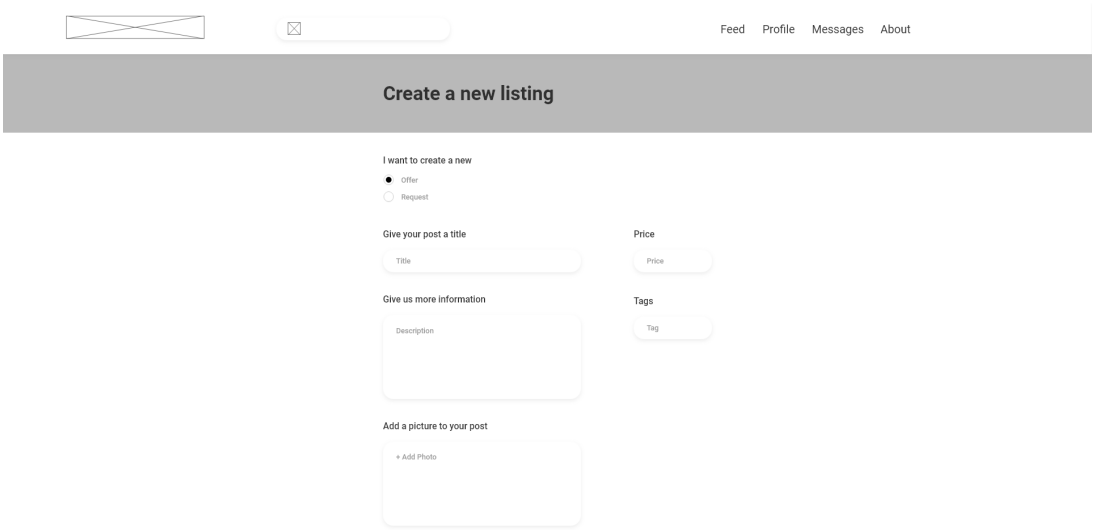
Profile



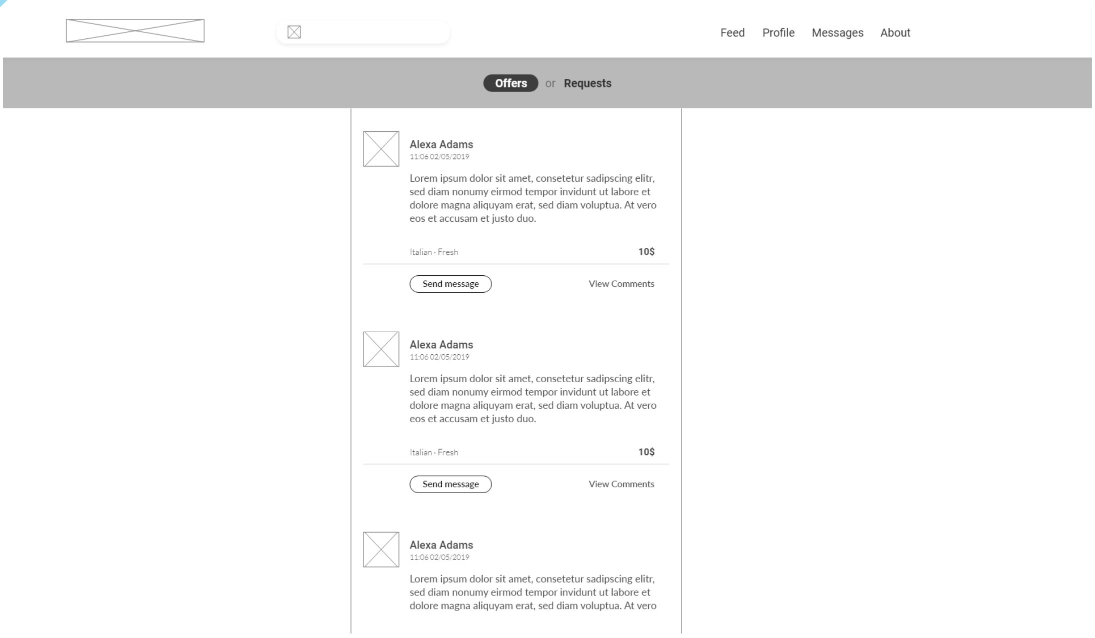
Overview user posts



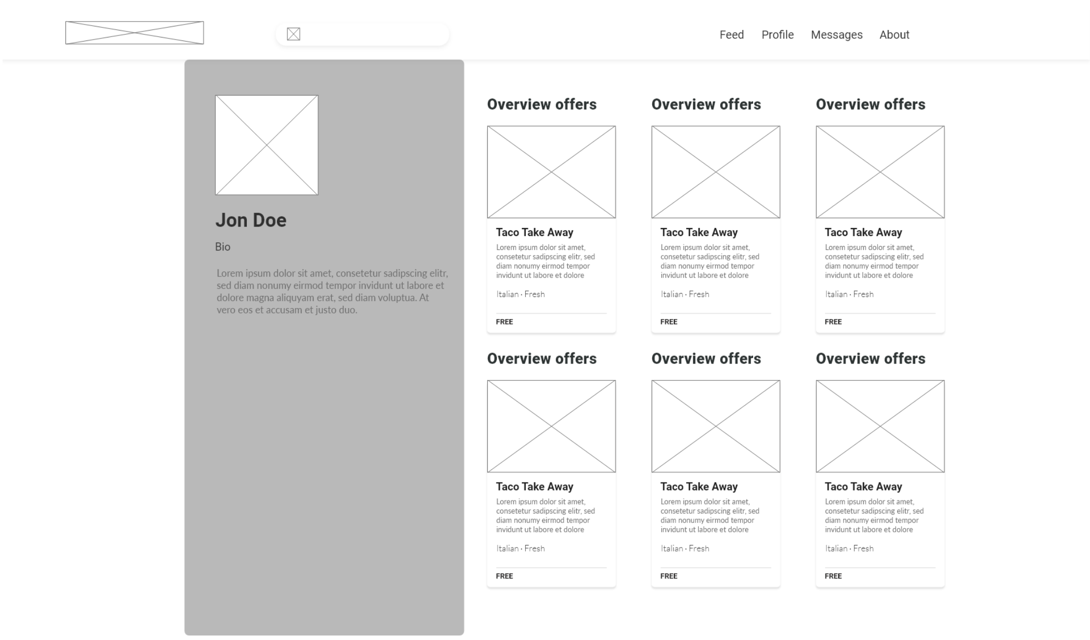
Chat



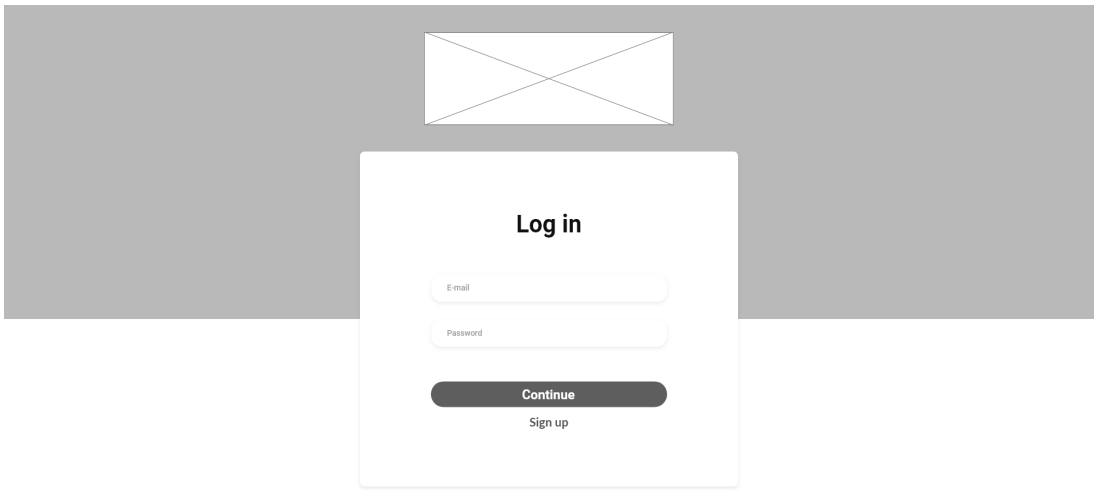
Create post



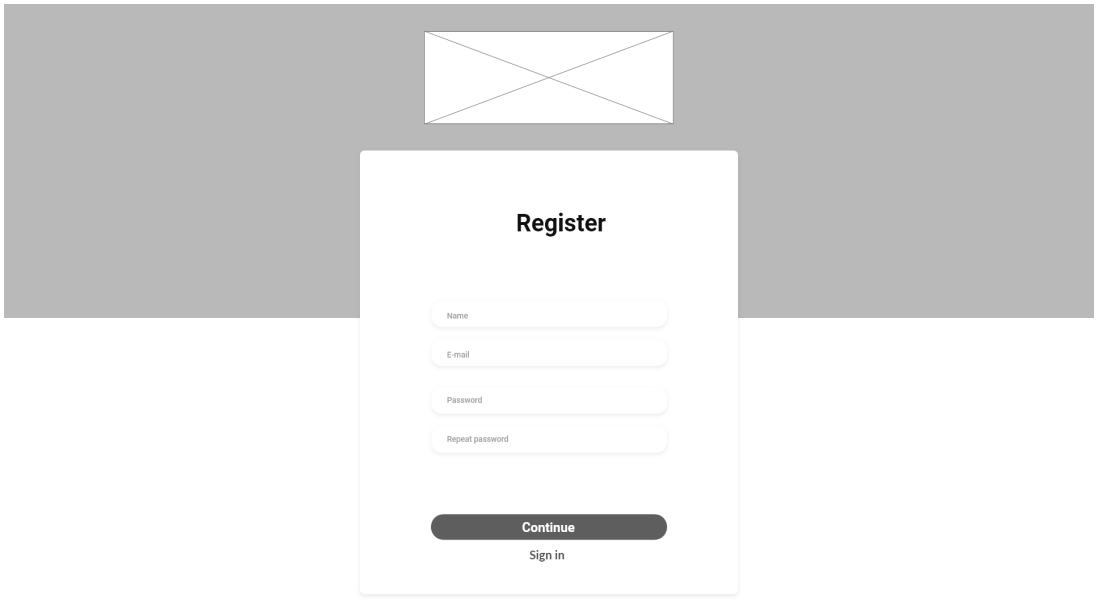
Feed



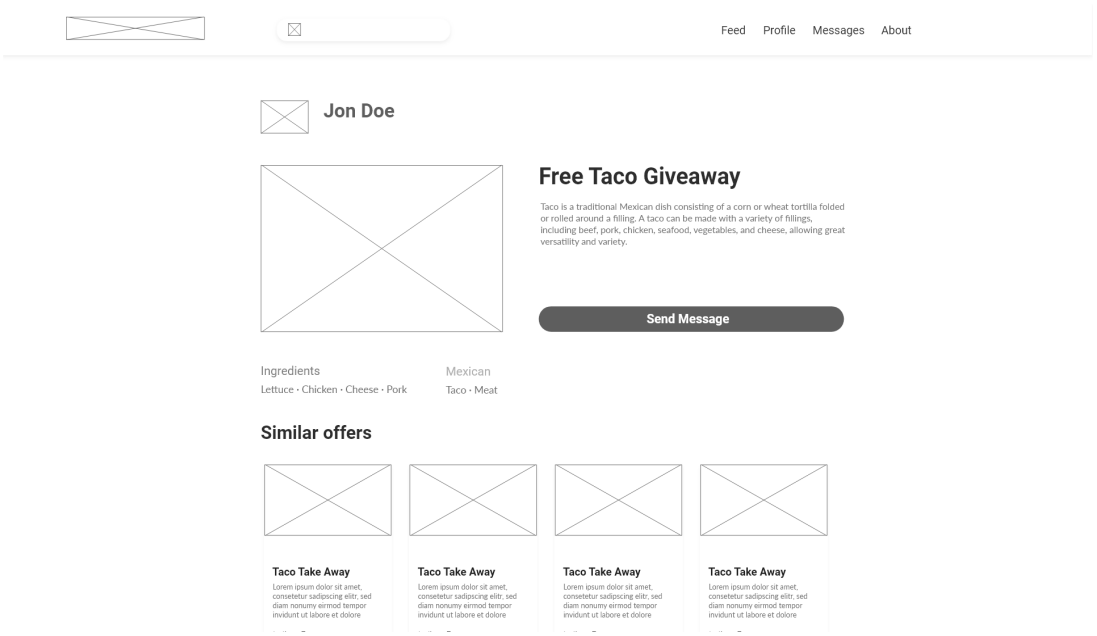
Profile



Log in



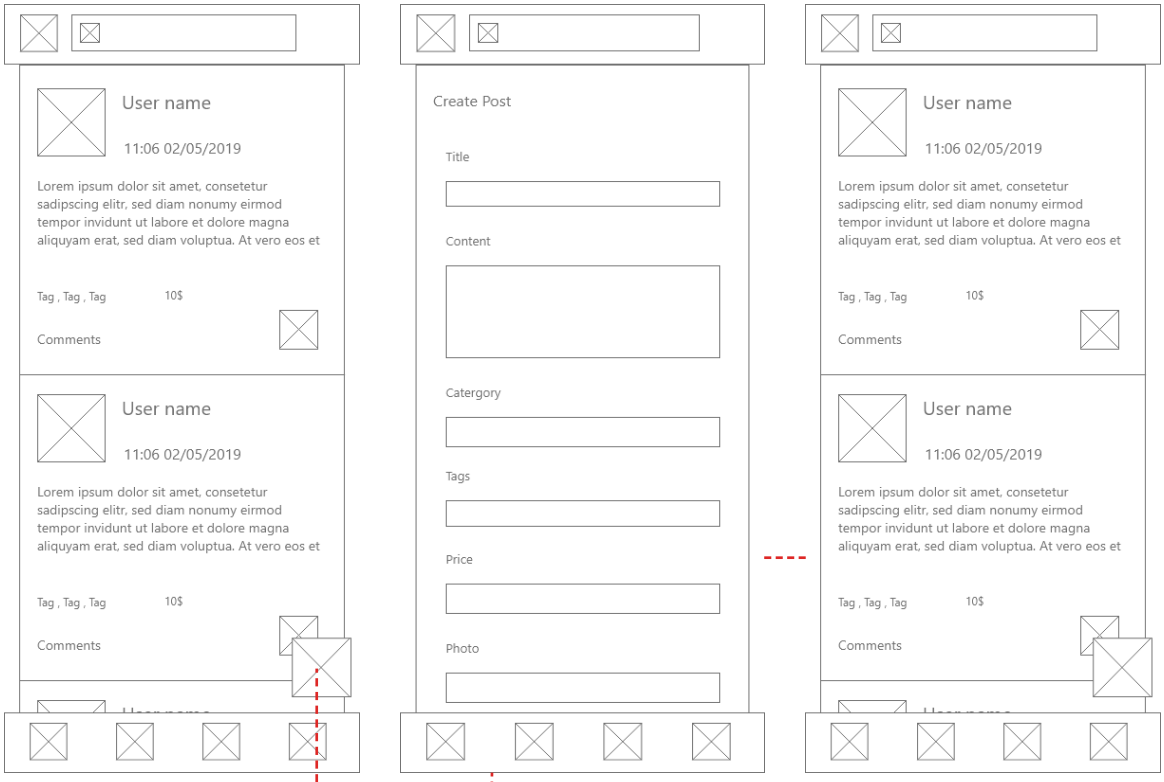
Register


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
Taco Take Away
Taco Take Away
Taco Take Away

Single post

Wireflow

Alle velden invullen en op “create” drukken



Nieuwe post aanmaken door op knop te drukken

Redirect naar User post overview

Style tile

Ag Lato — 18pt

Ag Lato — 16pt

Ag Lato — 12pt

Ag Roboto — 24pt

Ag Roboto — 20pt

Ag Roboto — 20pt

Ag Roboto — 17pt

Ag Roboto — 16pt

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

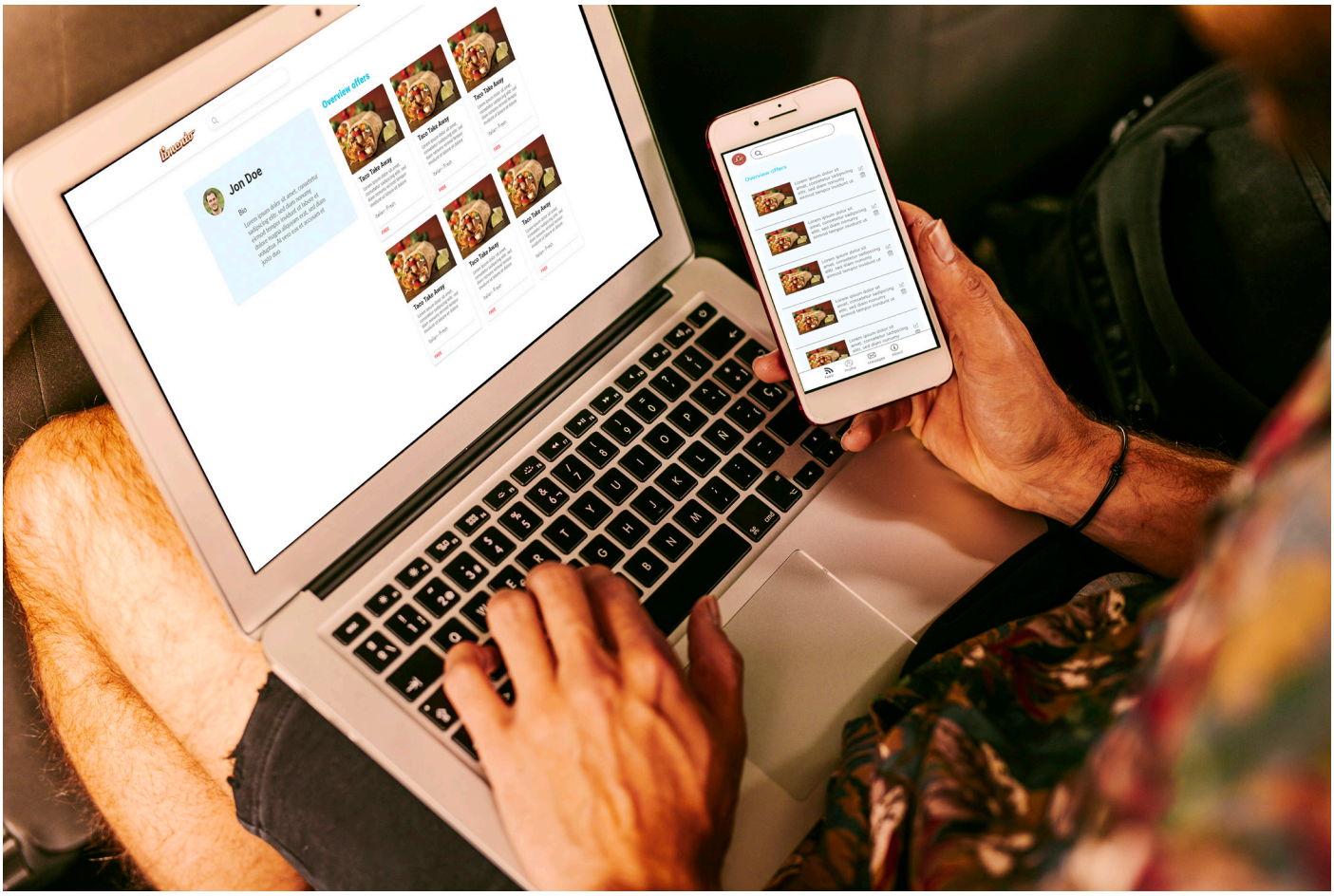
#Button roboto bold 20 #FFFFFF

Button

258 x 58 px

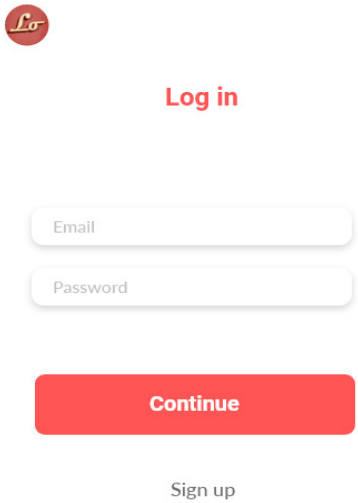
#FF5555

Mockup

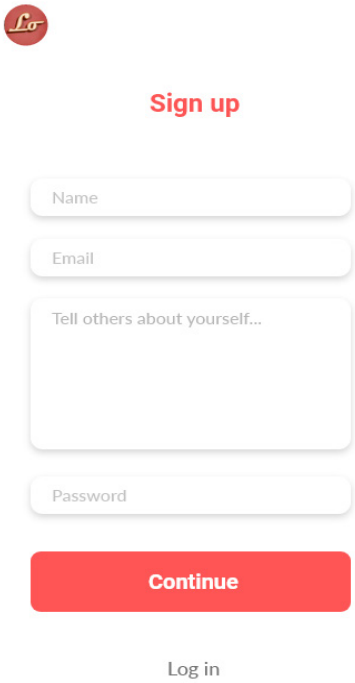


Visual designs

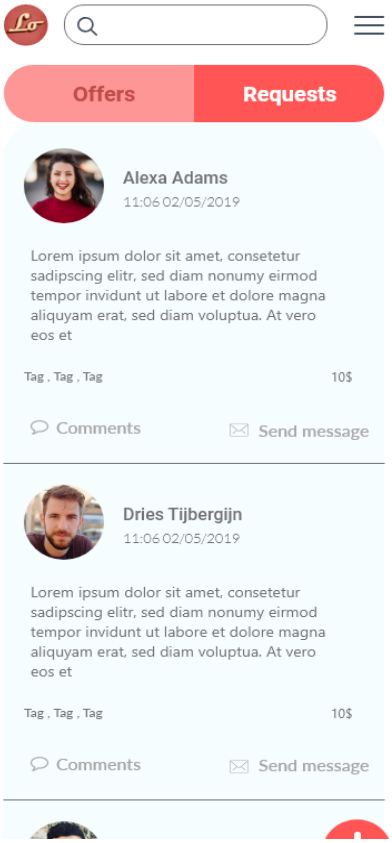
Home



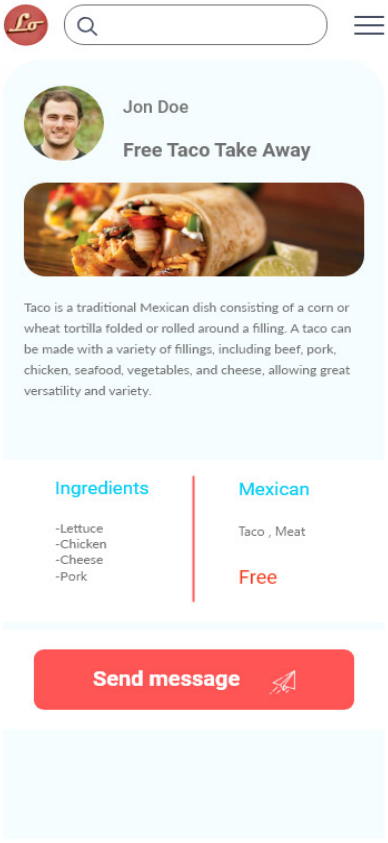
Login


Sign up
Name
Email
Tell others about yourself...
Password
Continue
Log in

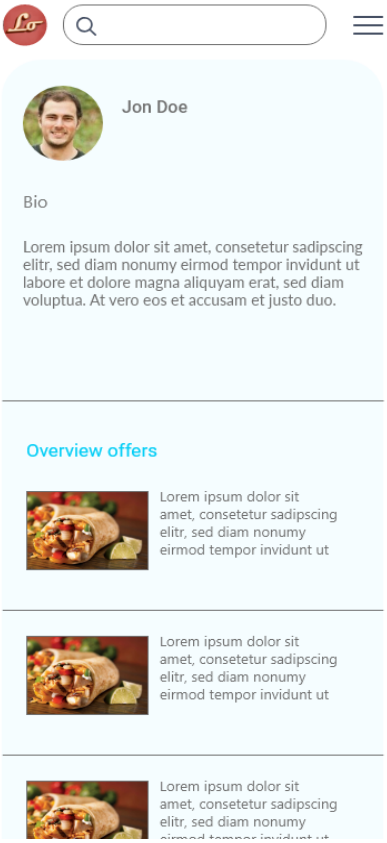
Sign up


Offers
Requests
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
Tag , Tag , Tag
10$
Comments
Send message
Dries Tijbergijn
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
Tag , Tag , Tag
10$
Comments
Send message

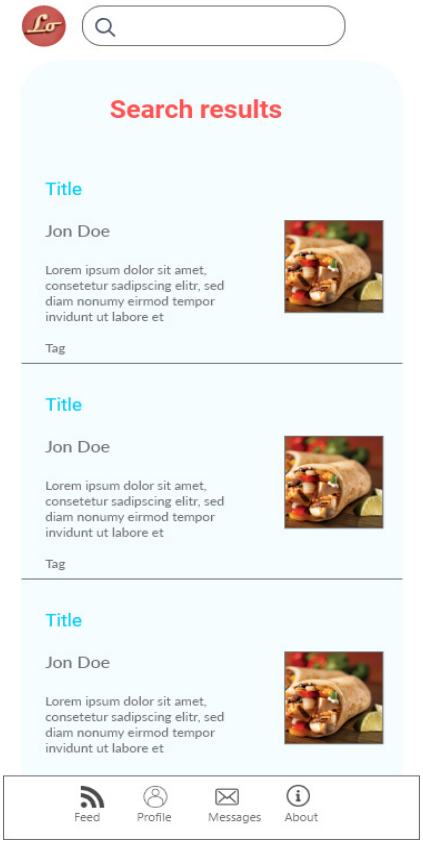
Feed



Single post



Profile overview

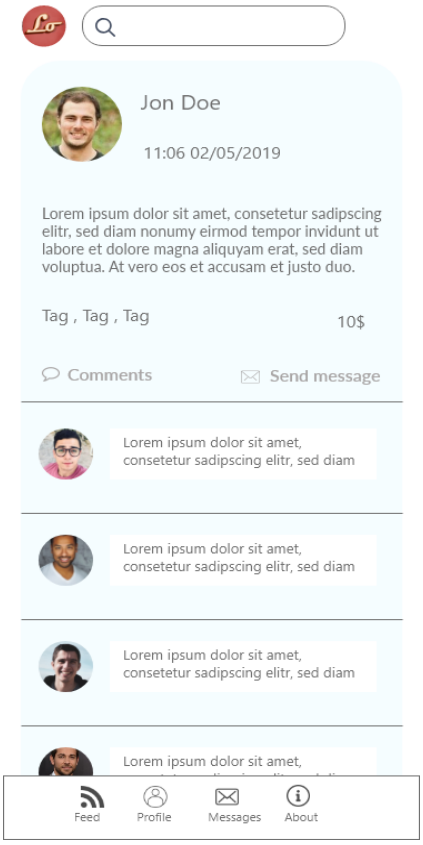


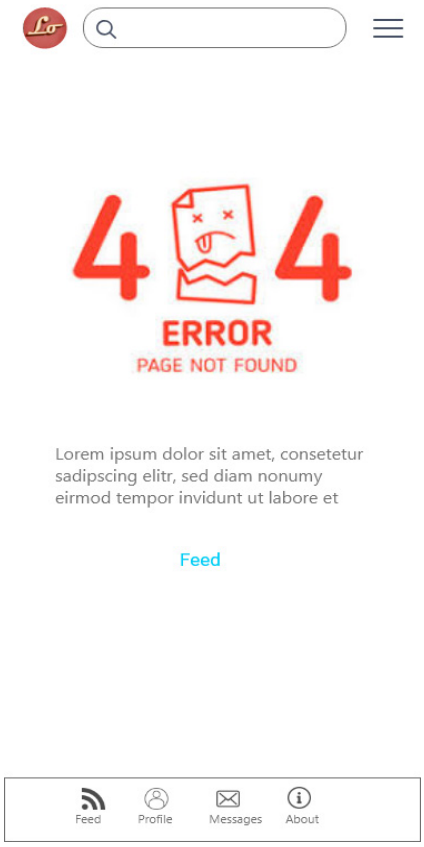
Search results

Overview posts

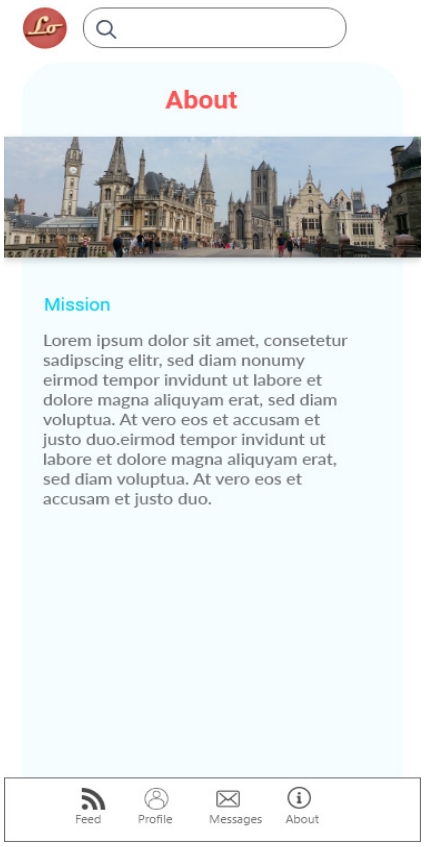
Chat

Comments

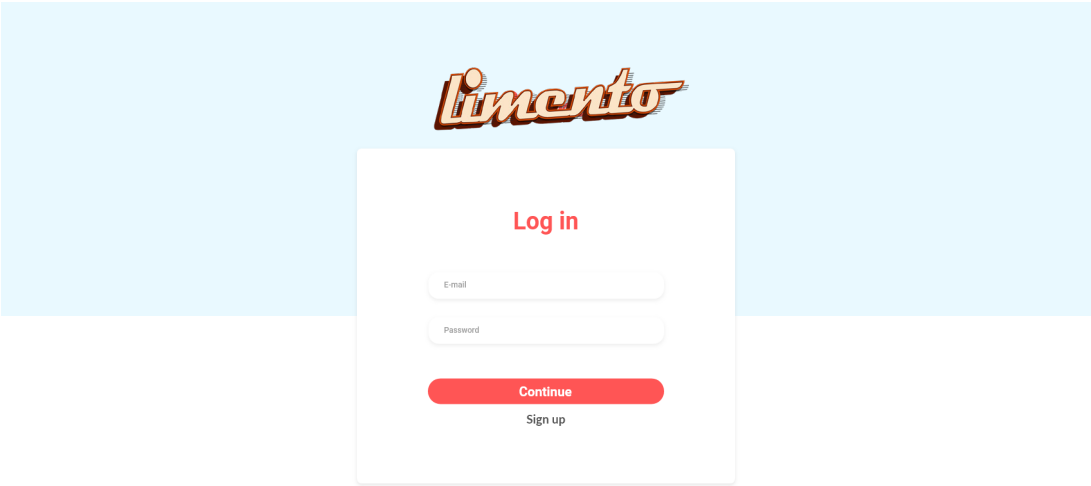



404
ERROR
PAGE NOT FOUND
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et
Feed
Feed
Profile
Messages
About

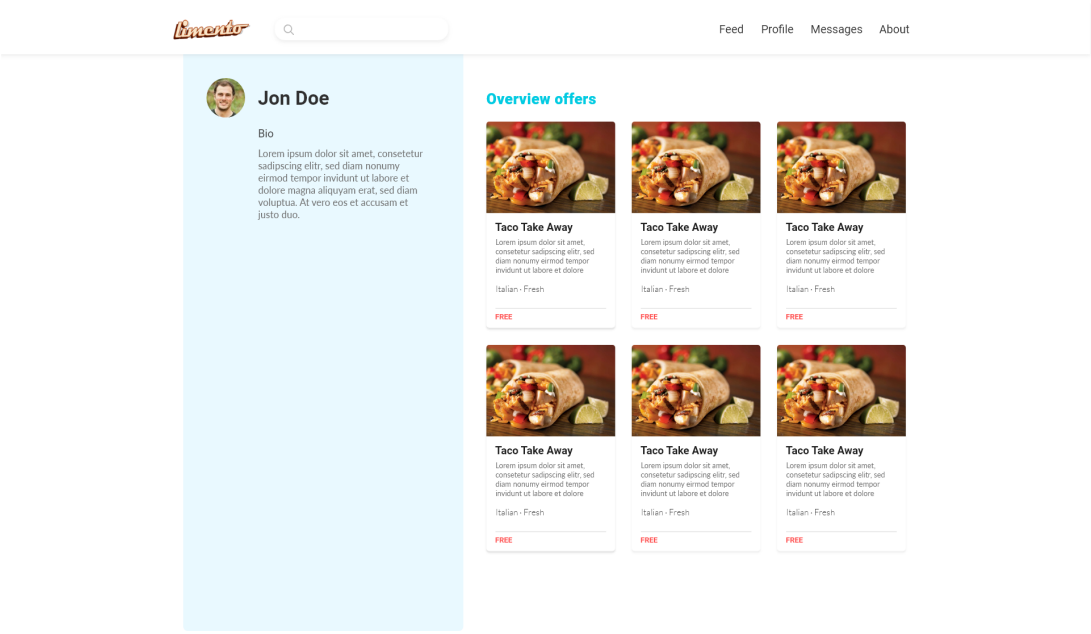
404


About
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

About us

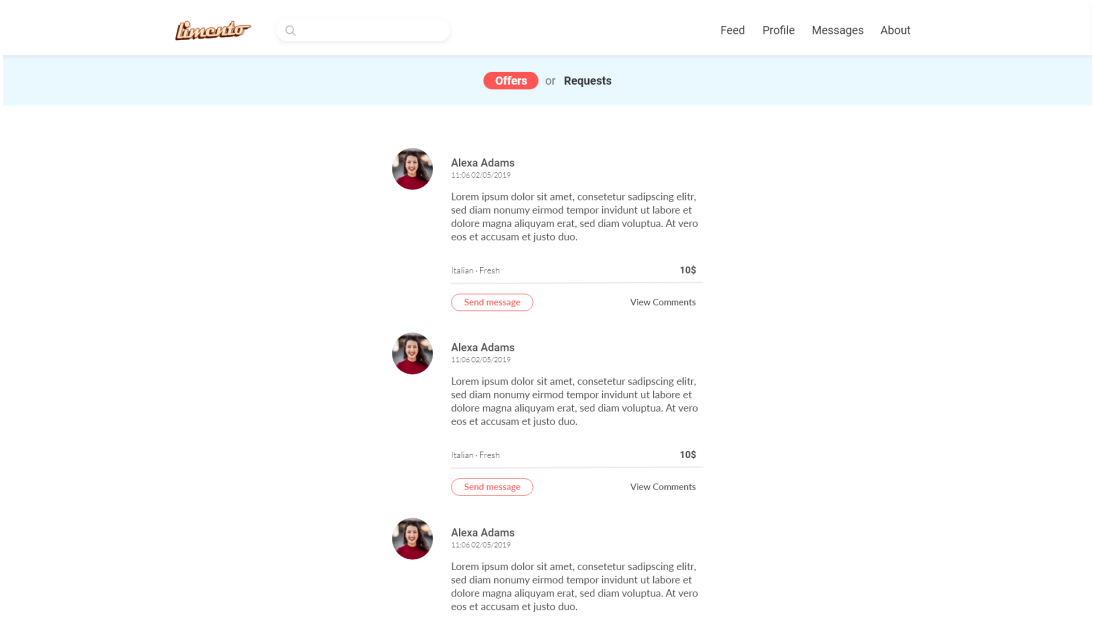


Log in

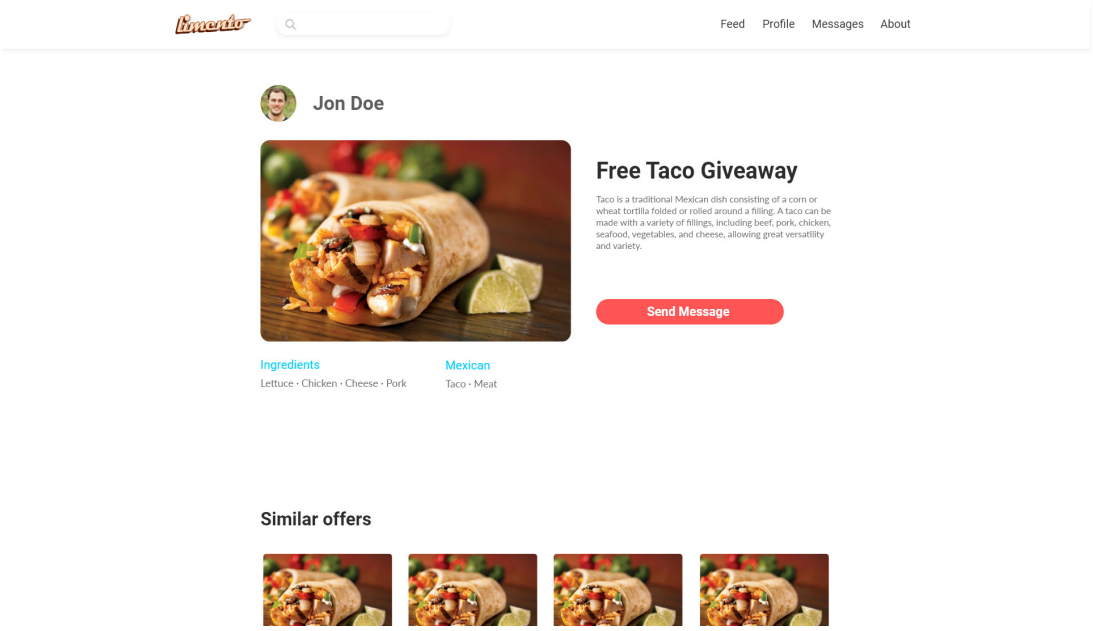


Profile

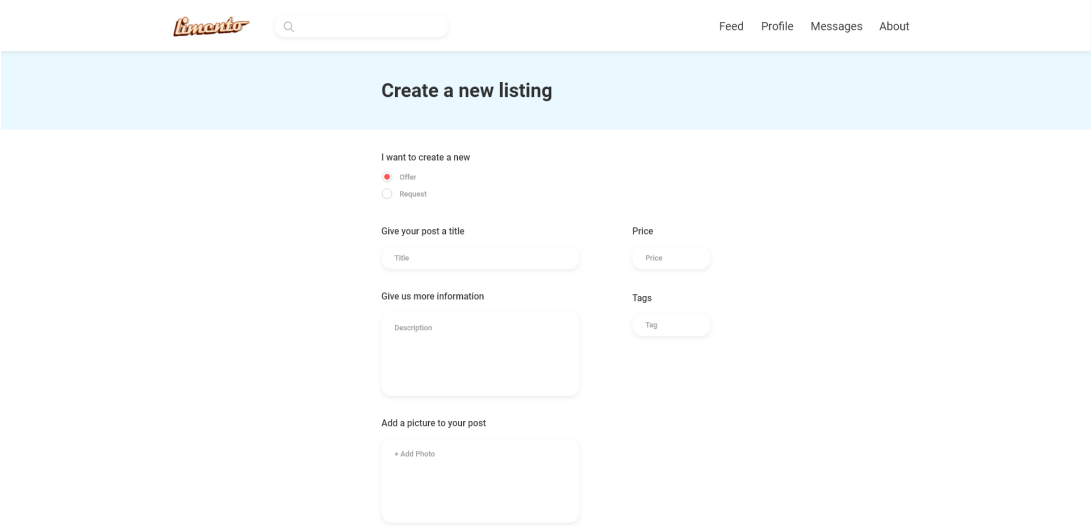
Feed



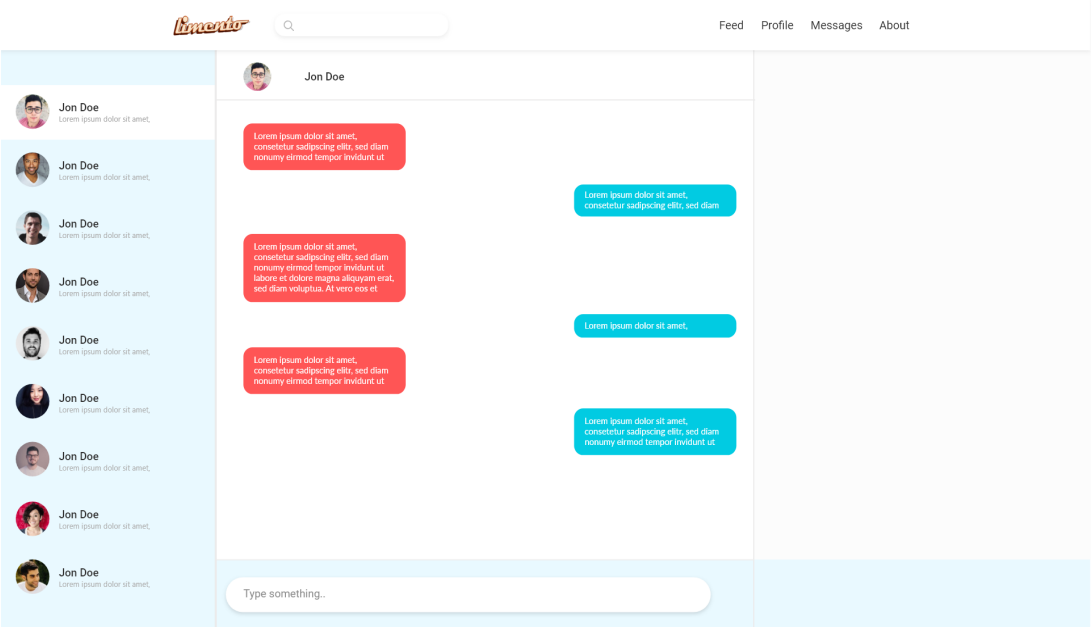
Single post



Create post



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

post.routes.js

```
/*
Import the internal libraries:
- PostController
*/
import { PostController } from '../controller';

// Create instance of PostController otherwise you can't use it
const postController = new PostController();

const initializeEndpoints = (parentRouter, authService) => {
    /**
     * @swagger
     * /api/v1/posts:
     *   get:
     *     tags:
     *       - Posts
     *     description: Returns all posts
     *     produces:
     *       - application/json
     *     responses:
     *       200:
     *         description: An array of posts
     */
    parentRouter.get('/posts', postController.index);
    /**
     * @swagger
     * /api/v1/posts/create:
     *   get:
     *     tags:
     *       - Post
     *     description: Returns specific viewmodel such as categories
     *     produces:
     *       - application/json
     *     responses:
     *       200:
     *         description: Create post
     */
```

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';
import ListItemText 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,
                })
            })
    }
```

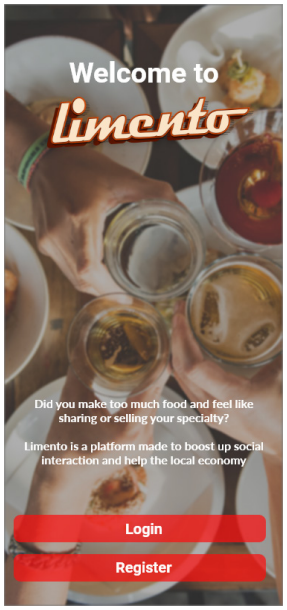
Thumbnails for posts

PostCardProfile.jsx

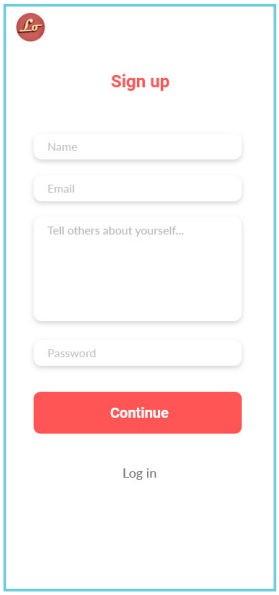
```jsx
/*
Import external libraries
*/
import React, { Component } from 'react';
import classNames from 'classnames';

import Title from '../../base/title';

import UserInfo from '../../user-info';
import EditIcon from '@material-ui/icons/Edit';
import IconButton from '@material-ui/core/IconButton';
import Tooltip from '@material-ui/core/Tooltip';
import DeleteIcon from '@material-ui/icons/Delete';
import { Divider } from '@material-ui/core';

class PostCardProfile extends Component {

    onEditHandler = (id) => {
        this.props.onEditHandler(id);
    }

    onDeleteHandler = (id) => {
        this.props.onDeleteHandler(id);
    }

    render() {
        const { post } = this.props;

        return (
            <React.Fragment>
                <article className="card--profile">
                    { post.media && <img className="card--profile__thumbnail" src={ post.media.path } /> }
                    <section className="card--profile__content">
                        <Title type={5}>{ post.title }</Title>
                    </section>
                    <section className="card--profile__meta">
                        <Tooltip title="Edit">
                            <IconButton aria-label="Edit">
                                <EditIcon onClick={() => this.onEditHandler(post.id)} />
                            </IconButton>
                        </Tooltip>
                        <Tooltip title="Delete">
                            <IconButton aria-label="Delete">
                                <DeleteIcon onClick={() => this.onDeleteHandler(post.id)} />
                            </IconButton>
                        </Tooltip>
                    </section>
                </article>
                <Divider></Divider>
            </React.Fragment>
        );
    }
}

export default (PostCardProfile);
```

Deliver

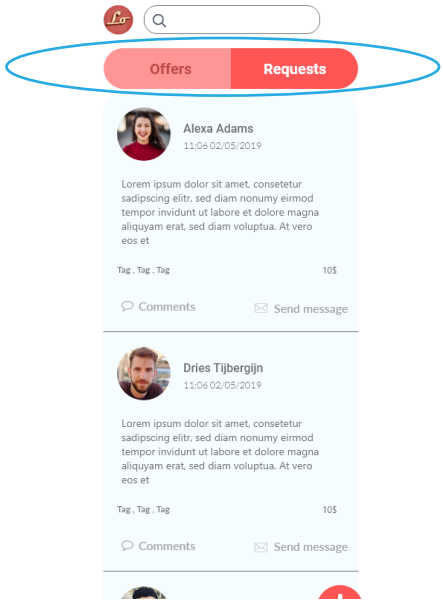
User handleiding



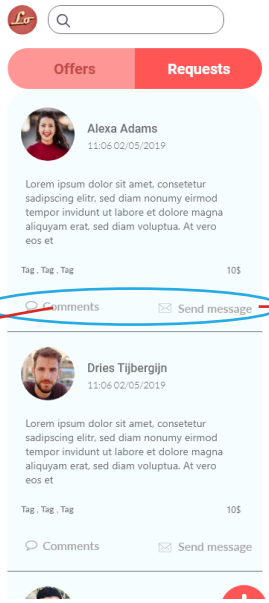
Als niet ingelogde gebruiker heb je de mogelijkheid om een nieuwe account aan te maken. Men kan ook bestande account gebruiken om in te loggen. Daarvoor zijn er twee knoppen voorzien.



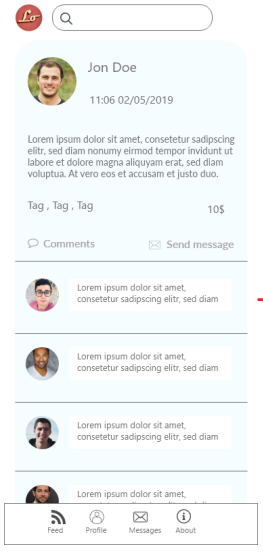
De informatie die de gebruiker ingeeft wordt opgeslagen in onze database. Als alle velden correct ingevuld zijn, mag de gebruiken op “Continue” klikken om verder te gaan naar de feed.

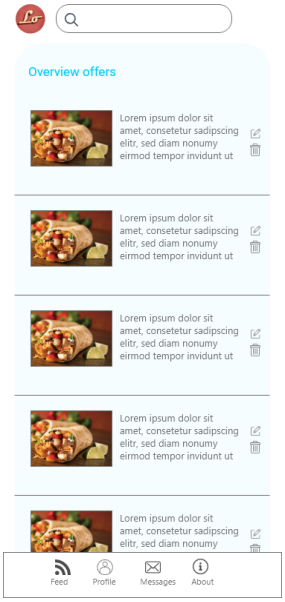


Op het moment dat de gebruiker in de feed zich bevindt, kan hij alle posts bekijken van andere gebruikers. Hij kan kiezen tussen “offers” en “requests” Afhankelijk van zijn keuze, krijgt hij andere posts te zien.

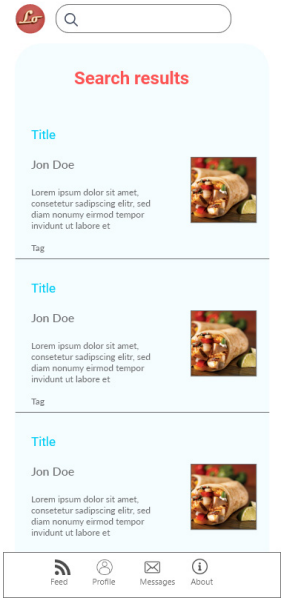


De gebruiker kan ook reacties bekijken en zelf plaatsen, of een bericht sturen naar de eigenaar van de post.





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