

# ReadMe tranquili'score

## Projet "Tranquili'score" - Équipe 7

Ce projet propose un estimateur du sentiment d'insécurité dans les transports. L'objectif est d'attirer un nouveau public dans les transports en communs en proposant des itinéraires tranquilles. Pour chaque gare et chaque tronçon de ligne, nous estimons un indice reflétant le sentiment d'insécurité. A partir de ces indices, nous évaluons la tranquillité de chaque trajet demandé par l'utilisateur. En parallèle, une interface opérateur permet de visualiser sur une carte les zones où nous estimons un sentiment de sécurité fort.

### Présentation du projet

Ce projet a été développé dans le cadre du [Hackathon IA et Mobilités](#), organisé par Île-de-France Mobilités les 21 et 22 novembre 2024. Pour en savoir plus, voici le [Guide des participants et participantes](#).

#### Le problème et la proposition de valeur

##### A quel problème le projet répond-t-il ?

Le projet vise à réconcilier les personnes hésitantes à prendre les transports en commun. Les transports en commun peuvent en effet être considérés comme un milieu hostile (inconfort, promiscuité, VSS, sentiment d'insécurité, ...). On va donc chercher à identifier des itinéraires "tranquilles" pour encourager ces personnes à prendre les transports.

##### Quels sont les usagers cibles ?

En particulier, on s'adresse à un public fragile (personnes âgées, enfants) et les femmes.

Personae :

Je suis Louise, j'ai 27 ans, « je veux qu'il y ait des agents en gare ça me rassure ». Je suis Nathalie, j'ai 60 ans, « je ne prends pas les transports en commun le soir car ça fait peur ». Je suis Antoine, j'ai 45 ans, « je veux avoir de la place assises ». Je suis Robert, j'ai 80 ans, « j'ai peur de tomber quand je suis dans la foule ».

#### La solution

##### Notre solution et son fonctionnement général

On a créé une brique logiciel sous forme de page web permettant la visualisation des "tranquili'scores". La page web puise dans une base de données contenant les données géographiques et temporelles ainsi que les indices prédictifs. Pour l'instant, la base de données est chargée à la main via un bouton mais on l'imagine être actualisé en temps réel en back-end.

##### Les données mobilisées

Techniquement, l'indice est calculé à partir de données telles que :

- taux de fraude dans la gare
- temps d'attente à quai
- date, jour de la semaine, heure, minute
- taux d'occupation à bord
- taux d'occupation à quai

et d'autres données non implémentées dans le calcul à date (taux de criminalité INSEE, nombre de bornes dans la gare, temps de trajet, ...).

##### Comment elle répond au problème

Quand un utilisateur souhaite voyager sur un itinéraire "tranquille", il peut utiliser notre brique logiciel qui lui affiche un indice de tranquillité (le "tranquili'score"), par gare et par tronçon de son itinéraire.

En plus, un indice global est calculé pour l'itinéraire (non implémenté à date).

##### La solution sous forme de web app

Nous avons créé une web app en nous inspirant du site d'Île-de-France mobilités. Nous avons repris le système de recherche d'un itinéraire, et nous y avons ajouté des informations pour renseigner le niveau d'insécurité calculé par notre algorithme

Nous avons développé 3 vues : - Vue itinéraire - Vue sécurité en gare - Vue liste des trajets

Les vues itinéraire et sécurité en gare contiennent une carte Openstreetmap du module Leaflet

La vue itinéraire permet de visualiser le sentiment d'insécurité prédict par tronçon entre chaque gare.

La vue sécurité en gare permet de visualiser le sentiment d'insécurité prédict dans chaque gare.

La vue liste des trajets permet à l'utilisateur de consulter le niveau d'insécurité prévu pour chaque trajet sur une ligne, ce qui lui permet de choisir parmi les prochains trajets prévus.

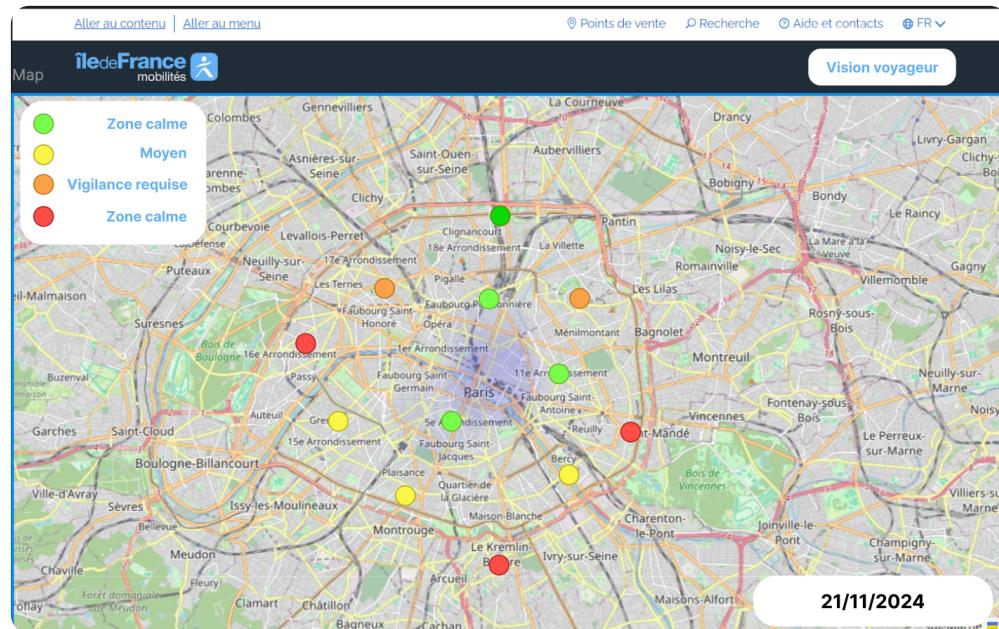
## Nos Maquettes

### Sur Figma

On a commencé par définir sur figma les fonctionnalités que nous voulions implémenter, notre site devrait se composer de deux interface:

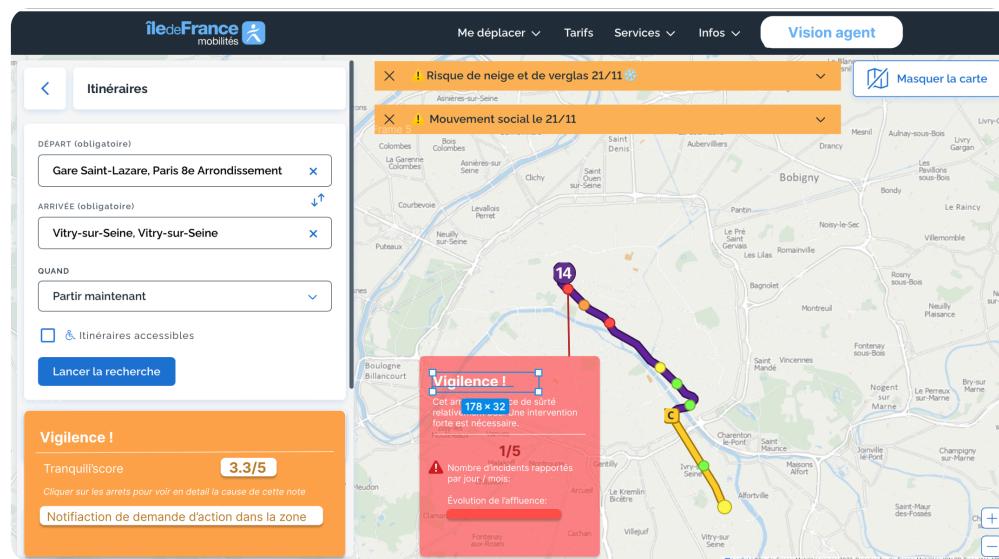
#### L'interface voyageurs

Moins détaillée pour l'utilisateur afin qu'il se sente informé



#### L'interface agents

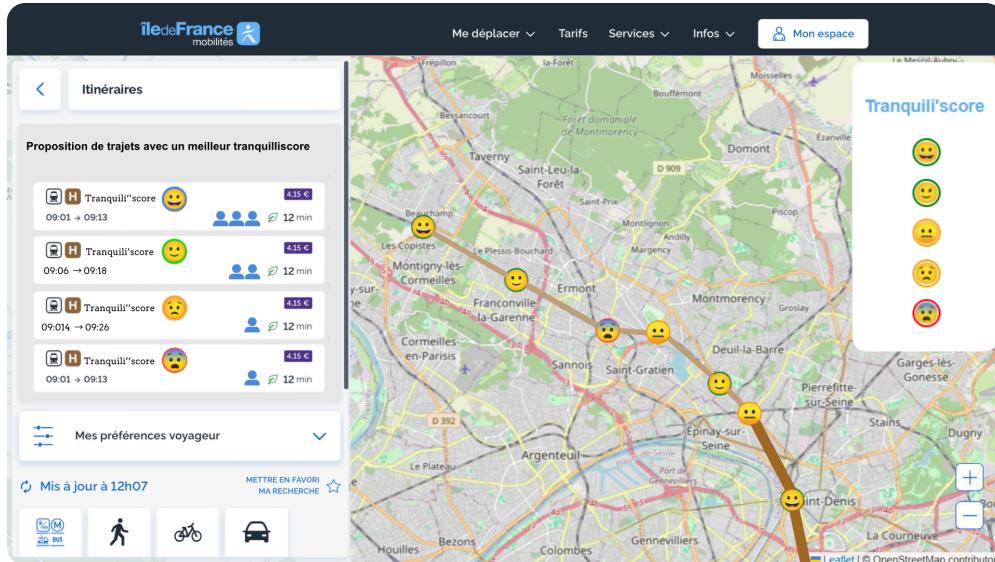
Beaucoup plus détaillé permettant aux agent d'agir en conséquence et assurer la sécurité des voyageurs



### Sur Canva

Sur Canva nous avons fait la partie utilisateur de notre projet :

[https://www.canva.com/design/DAGXM2\\_5VQo/RYEpk0izK\\_tdrGY9DZXP3Q/edit](https://www.canva.com/design/DAGXM2_5VQo/RYEpk0izK_tdrGY9DZXP3Q/edit)



En python

Determination des paramètres optimale et entraînement et évaluation du model

```

for indice in ["Indice_troncon_AB","Indice_gare_B","Indice_gare_A"]:
    df_encoded_train = df_encoded[df_encoded.numero_jour < 15]
    df_encoded_test = df_encoded[df_encoded.numero_jour == 15]

    X_train = df_encoded_train.drop(["Indice_troncon_AB","Indice_gare_A","Indice_gare_B"], axis=1)
    X_test = df_encoded_test.drop(["Indice_troncon_AB","Indice_gare_A","Indice_gare_B"], axis=1)

    Y_train = df_encoded_train[[indice]]
    Y_test = df_encoded_test[[indice]]

    clf = RandomForestClassifier(max_depth=30)

    grid_search = GridSearchCV(estimator=clf, param_grid=param_grid, cv=3, n_jobs=-1, verbose=2)

    start_time = time.time()

    # entraîne le classifier avec gridsearch pour optimiser les paramètres
    grid_search.fit(X_train, Y_train.values.ravel())

    # Meilleurs paramètres trouvés par GridSearchCV
    best_params = grid_search.best_params_
    print("Meilleurs paramètres trouvés : ", best_params)

    # Utiliser le meilleur modèle pour prédire
    best_clf = grid_search.best_estimator_

    end_time = time.time()

    training_time = end_time - start_time

    print(f"Temps d'entraînement : {training_time:.2f} secondes")

    start_time = time.time()

    # predict using classifier
    Y_pred = best_clf.predict(X_test)
    end_time = time.time()
    prediction_time = end_time - start_time

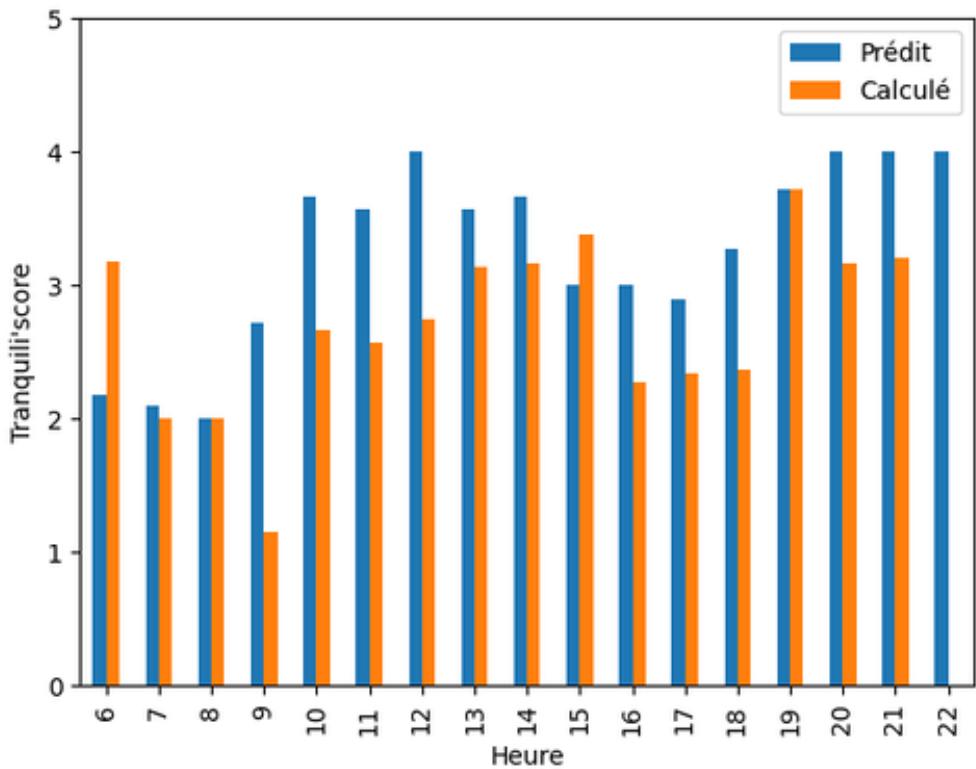
    print(f"Temps de prédiction : {prediction_time:.2f} secondes")

    # calculate accuracy
    accuracy = accuracy_score(Y_test, Y_pred)
    print('Accuracy pour la prédiction de ' + indice + f' : {accuracy}')

    print("Paramètres du modèle :")
    print(best_clf.get_params())

```

Tranquili'score prédit comparé au tranquilli'score sur la journée du 15 décembre (échantillon test )



Création de la base de donnée de trained test

```
Entrée [52]: path = r"C:\Users\{CP}\SNCF\Pôle data - Lab' MTA Grp0365 - Documents\Projets\Hackathon\Données\Données brutes\Base_de_donnees\train.csv"
df_final = pd.read_csv(path,encoding='latin-1',sep=',')
df_final['Jour-heure'] = df_final['Jour-heure'].astype(str)

#Merge avec les comptages
display(data_pdt_comptage[(data_pdt_comptage['Jour-heure']=="2023-11-01 04:10:00")&(data_pdt_comptage.Gare_A=="SDE")])
display(df_final[(df_final['Jour-heure']=="2023-11-01 06:20:00")&(df_final.Gare_A=="SDE")])
df_final = df_final.merge(data_pdt_comptage,on=['Gare_A','Jour-heure'],how='left').rename(columns={'Montees':'Montees_A',"Taux_occassionnel_AB": "Taux_occassionnel_AB_A", "Taux_occassionnel_BA": "Taux_occassionnel_AB_B", "Montees_B": "Montees_B_A", "Montees_A": "Montees_B_B"},errors='ignore')
df_final = df_final.merge(data_pdt_comptage.rename(columns={'Gare_A':'Gare_B'}),on=['Gare_B','Jour-heure'],how='left').rename(columns={'Montees':'Montees_B',"Taux_occassionnel_AB": "Taux_occassionnel_AB_B", "Taux_occassionnel_BA": "Taux_occassionnel_AB_A", "Montees_B": "Montees_A_B", "Montees_A": "Montees_A_A"},errors='ignore')

#Merge les temps d'attente
path_t = r"C:\Users\{CP}\SNCF\Pôle data - Lab' MTA Grp0365 - Documents\Projets\Hackathon\Données\Données pré-traitées\database\Temps_attente.xlsx"
data_tattente = pd.read_excel(path_t)
data_tattente['Date'] = data_tattente['Date'].astype(str)
df_final = df_final.merge(data_tattente.rename(columns={'Gare':'Gare_A','Date':'Jour-heure'},on=['Gare_A','Jour-heure'],how='left').rename(columns={'Taux_attente': 'Taux_attente_A', 'Temps_attente': 'Temps_attente_A', 'Taux_occassionnel_AB': 'Taux_occassionnel_AB_A', 'Taux_occassionnel_BA': 'Taux_occassionnel_AB_B', 'Montees_A': 'Montees_A_A', 'Montees_B': 'Montees_B_A'},errors='ignore'))
df_final = df_final.merge(data_tattente.rename(columns={'Gare':'Gare_B','Date':'Jour-heure'},on=['Gare_B','Jour-heure'],how='left').rename(columns={'Taux_attente': 'Taux_attente_B', 'Temps_attente': 'Temps_attente_B', 'Taux_occassionnel_AB': 'Taux_occassionnel_AB_B', 'Taux_occassionnel_BA': 'Taux_occassionnel_AB_A', 'Montees_A': 'Montees_A_B', 'Montees_B': 'Montees_B_B'},errors='ignore'))

#Merge les validations
df_final['tranche_horaire'] = df_final['Jour-heure'].apply(lambda x: int(str(x)[11:13])).replace({0:24,1:25,2:26})
df_final['demi_heure'] = df_final['Jour-heure'].apply(lambda x: "0-30" if int(str(x)[14:16])<30 else "30-60")
df_final['Date'] = df_final['Jour-heure'].apply(lambda x: str(x)[:10])
data_val_final['date'] = data_val_final['Date'].astype(str)

df_final = df_final[[['Gare_A', 'Gare_B', 'Ordre', 'Sens', 'Jour-heure', 'Trigramme_X', 'Lat_A', 'Long_A', 'Trigramme_Y', 'Lat_B', 'Long_B', 'Presence_gare_A', 'Presence_gare_B', 'Taux_criminalite_A', 'Taux_criminalite_B', 'Ligne_y', 'Gare_X', 'tranche_horaire_x', 'demi_heure_x', 'Montees_A', 'Taux_occ_tot_AB', 'Taux_occ_assis_AB', 'Montees_B', 'Temps_attente_A', 'Temps_attente_B', 'tranche_horaire', 'demi_heure', 'Date', 'Temps_A', 'Temps_B]]]

df_final = df_final.merge(data_val_final,on=['Gare_A','tranche_horaire','demi_heure','Date'],how='left').rename(columns={'Validation_A': 'Validation_A_A', 'Validation_B': 'Validation_B_A', 'Fraude_A': 'Fraude_A_A', 'Fraude_B': 'Fraude_B_A', 'Taux_occ_A': 'Taux_occ_A_A', 'Taux_occ_B': 'Taux_occ_B_A'},errors='ignore')
df_final['Validations_A'] = df_final['Validations_A']/6
df_final['Validations_B'] = df_final['Validations_B']/6
df_final['Fraude_A'] = 1-df_final['Validations_A']/df_final['Montees_A']
df_final['Fraude_B'] = 1-df_final['Validations_B']/df_final['Montees_B']
df_final['Taux_occ_A'] = df_final['Validations_A']/600
df_final['Taux_occ_B'] = df_final['Validations_B']/600

df_final = df_final[[['Gare_A', 'Gare_B', 'Ordre', 'Sens', 'Jour-heure', 'Trigramme_X', 'Lat_A', 'Long_A', 'Trigramme_Y', 'Lat_B', 'Long_B', 'Presence_gare_A', 'Presence_gare_B', 'Taux_criminalite_A', 'Taux_criminalite_B', 'Montees_A', 'Taux_occ_tot_AB', 'Taux_occ_assis_AB', 'Montees_B', 'Temps_attente_A', 'Temps_attente_B', 'tranche_horaire', 'demi_heure', 'Date', 'Validations_A', 'Validations_B', 'Fraude_A', 'Fraude_B', 'Taux_occ_A', 'Taux_occ_B', 'Temps_A', 'Temps_B', 'gare_X', 'gare_Y']]]
display(df_final)
#df_final.to_csv(r"C:\Users\{CP}\SNCF\Pôle data - Lab' MTA Grp0365 - Documents\Projets\Hackathon\Données\Données pré-traité\train.csv")
```

Date	Ligne	Gare	Gare_A	tranche_horaire	demi_heure	Jour-heure	Montees	Taux_occupation_totale	Taux_o_occupation_assis	
1072155	2023-11-01	H	St-Denis Bâtiment	SDE	4	0-30	2023-11-01 04:10:00	160.0	0.232646	0.464449

Calcul de l'indicateur Tranquilli'score

```

def score_temps_trajet(temps_trajet):
    temps_trajet = 1
    return(temps_trajet)

def score_temps_attente(temps_attente):
    temps_attente[temps_attente < 2] = 1
    temps_attente[(temps_attente >= 2) & (temps_attente < 5)] = 2
    temps_attente[(temps_attente >= 5)] = 3
    return(temps_attente)

def score_presence_en_gare(presence_en_gare):
    presence_en_gare = 1
    return(presence_en_gare)

def score_fraude(fraude):
    fraude[fraude >= 0.5] = 4
    fraude[(fraude >= 0.2) & (fraude < 0.5)] = 3
    fraude[(fraude >= 0.1) & (fraude < 0.2)] = 2
    fraude[(fraude < 0.1)] = 1
    return(fraude)

def score_taux_criminalite(taux_criminalite):
    taux_criminalite = 1
    return(taux_criminalite)

def feel_safe_score(data):
    # Taux d'occupation
    score_occupation_A = score_occupation_quai(data["Taux_occ_A"])
    score_occupation_B = score_occupation_quai(data["Taux_occ_B"])
    score_occupation_AB = score_occupation_bord(data["Taux_occ_assis_AB"])

    # Temps de contact avec le service
    # Temps d'attente
    score_attente_A = score_temps_attente(data["Temps_A"])
    score_attente_B = score_temps_attente(data["Temps_B"])

    # score_attente_AB = ...

    # Fraude
    score_fraude_A = score_fraude(data["Fraude_A"])
    score_fraude_B = score_fraude(data["Fraude_B"])

    # Présence en gare
    score_presence_gare_A = score_presence_en_gare(data["Presence_gare_A"])
    score_presence_gare_B = score_presence_en_gare(data["Presence_gare_B"])

    # Taux de criminalité
    score_criminalite_gare_A = score_presence_en_gare(data["Taux_criminalite_A"])
    score_criminalite_gare_B = score_presence_en_gare(data["Taux_criminalite_B"])

    max_score_A = 4 * 3 + 3 * 1 + 3
    max_score_B = 4 * 3 + 3 * 1 + 3
    max_score_AB = 4

    data["Indice_troncon_AB"] = score_occupation_AB / max_score_AB
    data["Indice_gare_A"] = (score_occupation_A + score_attente_A + score_fraude_A +
                            score_presence_gare_A + score_criminalite_gare_A) / max_score_A
    data["Indice_gare_B"] = (score_occupation_B + score_attente_B + score_fraude_B +
                            score_presence_gare_B + score_criminalite_gare_B) / max_score_B

    return data

```

## Les problèmes surmontés

- Difficultés dans le merge entre les bases de données, manque de référentiels
- Manque de données fiables et pertinentes pour le calcul de notre indice
- Gérer la "sensibilité" du sujet (lié aux VSS, à la sécurité)
- Trouver le bonne équilibre entre solution technique viable et idée

## Et la suite ?

- Si on avait eu plus de temps, on aurait voulu fiabiliser l'indice en introduisant d'autres sources de données (notamment les autres sources de données évoquées plus haut, mais aussi d'autres sur la fiabilité des lignes, les signalements d'incidents, des données issues de plateformes VSS comme THE SORORITY, ...)
- On imagine une validation de notre indice via des données de crowdsourcing (sondage sur le sentiment de sécurité ou la tranquillité liée à un itinéraire)
- On intégrerait notre brique logiciel dans l'application Ile de France Mobilité et on sugererait l'utilisation de Tranquil'iti pour les personnes appartenant au public cible

## Intallation et utilisation

### Utiliser la webapp :

1. Cliquer sur la page index.html
2. Pour afficher les trajets dans la vue utilisateur, il faut importer le fichier Test Sécurité Trajet Utilisateur.csv qui contient des informations créées par le programme de prévision 3) De même pour la vue des gares, il faut importer le fichier Test Sécurité Gare Agents.csv

## La licence

Ici, il faut décrire la ou les licences du projet. Vous pouvez utiliser la licence [MIT](#), qui est très permissive. Si on souhaite s'assurer que les dérivés du projet restent Open-Source, vous pouvez utiliser la licence [GPLv3](#).

Le code et la documentation de ce projet sont sous licence [MIT](#).