ENSAM
ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS
UNIVERSITÉ HASSAN II DE CASABLANCA

**Filière : Génie Informatique et Intelligence artificielle**

# MARATHON PHOTO FINDER


**Sous la supervision de :**

Pr. ZAKRANI Abdelali

**Elaboré par :**

- RITOUB Ayoub
- MASSIR Hamza
- EL JAMIY Ismail
- AMZYL Mohammed Ali
- LAHROUR Abdessamad
- BOUACHRINE Ahmed Reda


Rapport — Mai 2024

ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS
UNIVERSITÉ HASSAN II DE CASABLANCA

# SOMMAIRE :

# INTRODUCTION GENERALE :

Notre initiative, le projet de gestion des photos « **Marathon Photo Finder** », s'inscrit dans une démarche visant à dynamiser et à optimiser l'expérience des coureurs de marathons et des organisateurs d'événements. Cette application se positionne comme un outil essentiel, simplifiant la recherche et l'accès aux photos de course en fournissant une vue complète et intuitive sur les photos attribuées à chaque numéro de dossard.

**Ce rapport est structuré en cinq chapitres :**

- Le premier chapitre présente le cahier des charges, en détaillant le contexte du projet, les objectifs, les cibles, ainsi que les besoins fonctionnels et non fonctionnels de la plateforme.
- Le deuxième chapitre se concentre sur la collecte et l'annotation des données, en expliquant les sources de données, le processus d'annotation avec Roboflow, et les étapes de préparation des données.
- Le troisième chapitre est dédié au développement du modèle, avec une explication de l'utilisation de YOLO pour la détection des dossards, l'entraînement et la validation du modèle, ainsi que la reconnaissance des numéros de dossard avec PaddleOCR.
- Le quatrième chapitre traite de la conception, en présentant le diagramme de cas d'utilisation.
- Le cinquième chapitre couvre la réalisation, incluant les outils et technologies utilisés, le guide de configuration et d'exécution de l'application, les fonctionnalités principales, le travail réalisé, et le code.

Enfin, une conclusion générale récapitule les principaux résultats et les perspectives futures du projet.

# I. CAHIER DE CHARGES :

## A. Contexte du projet :

### 1. Objectifs du site :

L'objectif principal de « Marathon Photo Finder » est de simplifier et d'améliorer l'expérience des coureurs en leur offrant une solution rapide et efficace pour retrouver leurs photos de course. En utilisant la technologie de reconnaissance d'images et d'autres techniques de machine learning, notre site permet :

1. **Identification Rapide** : Permettre aux coureurs de retrouver leurs photos rapidement en entrant simplement leur numéro de dossard.
2. **Expérience Utilisateur Optimale** : Fournir une interface conviviale qui rend la recherche de photos simple et agréable.
3. **Gestion Efficace des Photos** : Les photographes peuvent facilement télécharger en masse les photos prises pendant l'événement. Ces photos sont ensuite automatiquement classées en fonction du numéro de dossard, facilitant ainsi l'organisation et la recherche.
4. **Accessibilité et Facilité d'Utilisation** : Tant les photographes que les participants bénéficient d'une interface intuitive. Les photographes peuvent gérer le téléchargement et la classification des photos, tandis que les coureurs peuvent rechercher et accéder facilement à leurs photos personnelles.

### 2. Cible adressée par le site :

L'application « Marathon Photo Finder » s'adresse aux photographes couvrant les marathons, aux coureurs amateurs et professionnels, ainsi qu'aux organisateurs de courses.

### 3. Périmètre du projet :

L'application « Marathon Photo Finder » est conçue pour être flexible et accessible, s'adaptant à diverses plateformes, y compris les appareils mobiles.

## B. Besoins :

### 1. Organisation de la plateforme :

- **Page Home (contient la Liste de Marathons Passés)** : À l'accès au site, une liste de marathons déjà passés s'affiche, incluant le nom, le pays, la description, la distance et le genre des participants, avec un bouton "Upload".
- **Page Upload de Photos** : Les photographes peuvent utiliser le bouton "Upload" pour télécharger les photos d'un nouveau marathon. Ces photos sont ensuite ajoutées dynamiquement à la liste des marathons.
- **Processus de Téléchargement** : En cliquant sur "Submit" dans le formulaire d'upload, les photos sont téléchargées dans une base de données et traitées par un modèle de détection des dossards.
- **Page de Recherche de Photos par les Participants** : Les participants peuvent rechercher le nom du marathon, cliquer sur "View" pour afficher le marathon, et utiliser une barre de recherche pour saisir leur numéro de dossard. Toutes les images correspondantes apparaissent, et ils peuvent les télécharger localement.

### 2. Navigation de la plateforme :

- **Accueil → Téléchargement des Photos** : Pour les photographes souhaitant uploader de nouvelles photos.
- **Accueil → Détails du Marathon → Recherche par Numéro de Dossard** : Pour les coureurs cherchant leurs photos.

### 3. Besoins non fonctionnels :

Les besoins non fonctionnels de l'application « Marathon Photo Finder » sont les suivants :

- **Sécurité et Confidentialité** : Assurer la protection des données des utilisateurs et des photos téléchargées, avec des droits d'accès strictement contrôlés.

- **Performance et Rapidité** : Garantir des temps de réponse rapides pour une expérience utilisateur fluide, même lors du téléchargement et de la recherche de grandes quantités de photos.
- **Ergonomie et Convivialité** : Proposer des interfaces claires, intuitives et faciles à utiliser, offrant une expérience utilisateur optimale pour les photographes et les participants.
- **Maintenance** : Faciliter la compréhension et la modification du code source pour une maintenance aisée et efficace de l'application.
- **Utilisabilité** : Assurer que l'application est facile à apprendre et à utiliser, avec une courbe d'apprentissage minimale pour les nouveaux utilisateurs.
- **Fiabilité** : Garantir le bon fonctionnement de l'application en toutes circonstances, sans dysfonctionnements majeurs, et avec une haute disponibilité.
- **Réutilisabilité :** Concevoir des composants de l'application qui peuvent être réutilisés pour développer d'autres applications similaires ou pour étendre les fonctionnalités existantes.

## 4. Besoins fonctionnels :

Les besoins fonctionnels de l'application « Marathon Photo Finder » sont :

1. **Gestion des Photos** :

- **Téléchargement de Photos** : Les photographes doivent pouvoir télécharger en masse les photos prises lors de l'événement via l'interface de téléchargement.
- **Classification Automatique** : Les photos téléchargées doivent être classées automatiquement en fonction des numéros de dossard des participants en utilisant un modèle de machine learning.
- **Recherche par Numéro de Dossard** : Les participants doivent pouvoir saisir leur numéro de dossard pour accéder rapidement à toutes les photos les concernant.
- **Visualisation et Téléchargement des Photos** : Les participants doivent pouvoir visualiser les photos associées à leur numéro de dossard et les télécharger localement.

2. **Interface Utilisateur :**

- **Interface pour Photographes** : Fournir une interface conviviale permettant aux photographes de télécharger les photos d'un nouveau marathon, déclenchant ainsi leur ajout dynamique à la liste des marathons et leur traitement par le modèle de détection des dossards.
- **Interface pour Participants** : Fournir une interface intuitive permettant aux participants de rechercher le nom du marathon, de cliquer sur "View" pour accéder à une barre de recherche où ils peuvent saisir leur numéro de dossard et visualiser toutes leurs photos.

### C. Conclusion :

Dans ce chapitre, nous avons examiné le cahier des charges de l'application pour comprendre ses besoins fonctionnels et non fonctionnels, ce qui nous permettra de procéder ensuite à la phase de conception.

## II. COLLECTE ET ANNOTATION DES DONNEES :

### A. Source de Données :

Pour ce projet, nous avons utilisé un dataset initial fourni par notre professeur, comprenant des photos de participants lors d'un événement sportif (un marathon), où chaque photo contient un ou plusieurs participants, chacun avec un numéro de dossard différent. Nous avons complété ce dataset avec un dataset additionnel provenant de Roboflow pour enrichir notre collection de photos. Un nettoyage rigoureux des données a été effectué, ne conservant que les photos où le dossard du participant est clairement visible.

### B. Annotation des données avec Roboflow :

Les annotations des photos ont été effectuées à la main à l'aide de l'outil Roboflow afin de repérer et de marquer les numéros de dossard.

### C. Préparation des Données :

Pour préparer notre dataset en vue de l'entraînement du modèle de machine learning, nous avons utilisé l'outil Roboflow.

**Voici les étapes détaillées de notre processus de préparation des données :**

- **Prétraitement des Données :**

**- Auto-Orientation**: Les images ont été automatiquement orientées pour s'assurer qu'elles sont toutes alignées de la même manière.
- **Redimensionnement:** Toutes les images ont été redimensionnées à une résolution de 640x640 pixels pour garantir une uniformité et améliorer l'efficacité de l'entraînement.
- **Filtrage des Images Nulles:** Nous avons configuré le système pour exiger que toutes les images contiennent des annotations.

- **Répartition du Dataset :**

Pour optimiser l'entraînement et l'évaluation de notre modèle, nous avons divisé notre dataset en trois ensembles distincts :
- **Ensemble d'entraînement**: Contient 3000 images, soit 90% du dataset total. Cet ensemble est utilisé pour l'entraînement du modèle.
- **Ensemble de validation**: Comprend 220 images, soit 7% du dataset total. Cet ensemble est utilisé pour évaluer les performances du modèle pendant l'entraînement.
- **Ensemble de test**: Contient 128 images, soit 4% du dataset total. Cet ensemble est utilisé pour évaluer la performance finale du modèle après l'entraînement.

- **Augmentation des Données :**

Pour enrichir notre dataset et améliorer la robustesse de notre modèle, nous avons appliqué diverses techniques d'augmentation des données :

- **Flip** : Les images ont été retournées horizontalement.
- **Crop** : Un zoom minimum de 0% et maximum de 5% a été appliqué.
- **Rotation** : Les images ont été pivotées aléatoirement entre -15° et +15°.
- **Grayscale** : 15% des images ont été converties en niveaux de gris.
- **Teinte (Hue)** : Les teintes des images ont été ajustées entre -15° et +15°.
- **Saturation** : La saturation des images a été modifiée entre -25% et +25%.
- **Flou (Blur)** : Un flou jusqu'à 1.9 pixels a été appliqué.
- **Bruit (Noise)** : Du bruit a été ajouté à hauteur de 0.54% des pixels.

- **Rotation de la Boîte de Délimitation** : Les boîtes de délimitation ont été pivotées entre -15° et +15°.
- **Exposition de la Boîte de Délimitation** : L'exposition des boîtes de délimitation a été modifiée entre -10% et +10%.

→ Ces étapes de préparation et d'augmentation des données nous ont permis d'obtenir un dataset diversifié et de haute qualité, prêt pour l'entraînement du modèle de machine learning. La répartition judicieuse du dataset en ensembles d'entraînement, de validation et de test assure une évaluation rigoureuse des performances du modèle. Grâce aux diverses techniques d'augmentation des données telles que le flip horizontal, la rotation, le recadrage, l'ajustement de la teinte et de la saturation, ainsi que l'ajout de flou et de bruit, notre modèle a pu apprendre de manière plus efficace et généralisée. Ces efforts ont contribué à atteindre une précision de **90.1%** lors de la détection des dossards.

# III. DEVELOPPEMENT DU MODELE :

## A. Utilisation de YOLO pour la détection des dossards :

Nous avons choisi YOLOv8 pour notre modèle de machine learning, en raison de sa réputation pour des performances élevées en détection d'objets.

## B. Entraînement et validation du modèle :

L'entraînement du modèle a été réalisé sur Google Colab, utilisant ses puissantes ressources de calcul pour accélérer le processus. Après l'entraînement, notre modèle a atteint une précision de 90,1%, attestant de son efficacité pour la détection des dossards.

### C. Reconnaissance des Numéros de Dossard avec PaddleOCR :

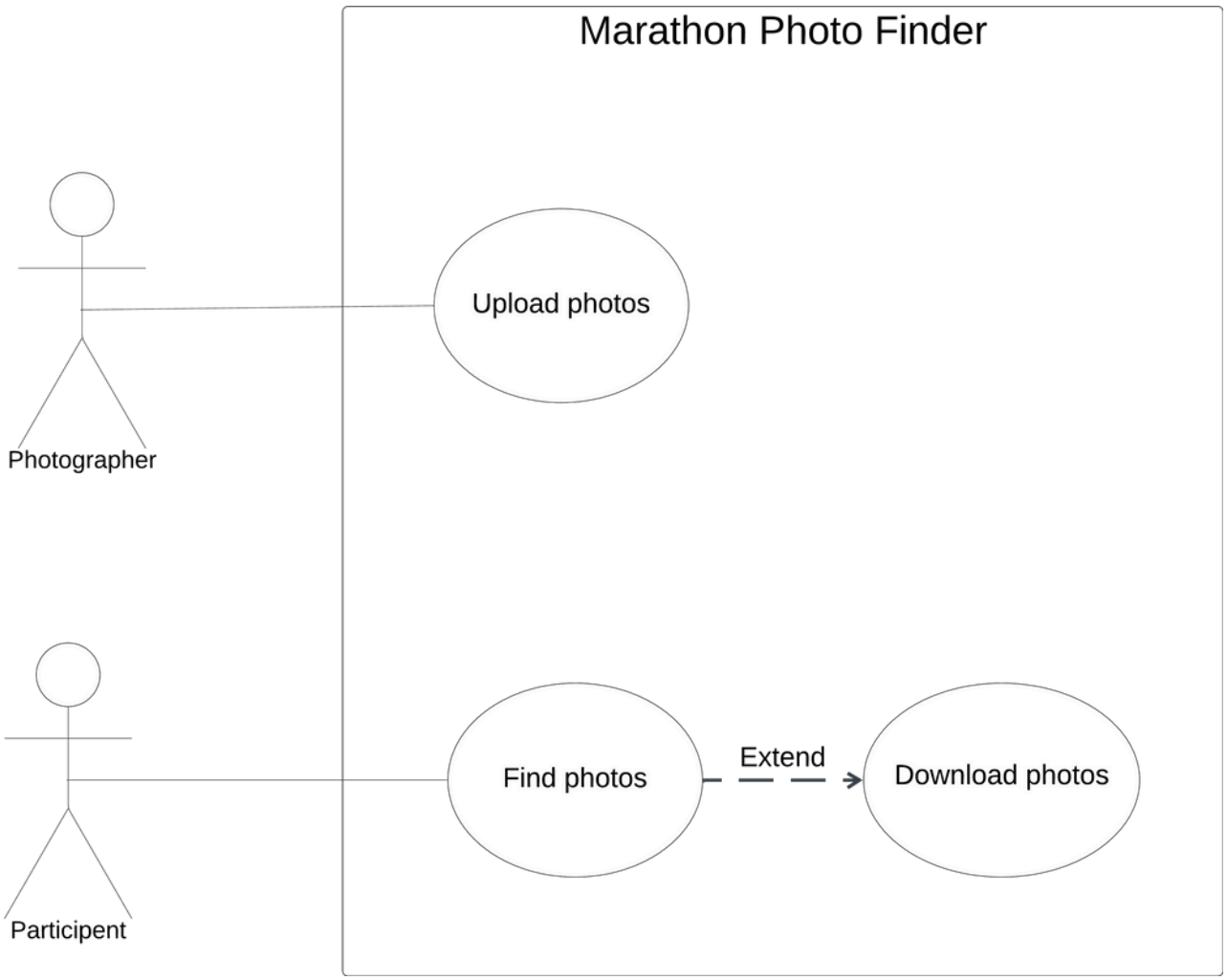
Pour extraire les numéros de dossard à partir des régions détectées par YOLOv8, nous avons intégré PaddleOCR, un outil performant de reconnaissance optique de caractères. PaddleOCR a été utilisé pour identifier et lire les numéros de dossard à partir des images segmentées.

→Ces étapes de collecte, d'analyse, d'annotation et de préparation des données ont abouti à la création d'un **dataset** robuste et de haute qualité, idéal pour l'entraînement du modèle de **machine learning**. Grâce à ce processus méticuleux, nous avons développé un modèle performant, capable de détecter les dossards avec une précision remarquable et de lire les numéros de dossard avec une grande fiabilité dans les photos d'événements sportifs. Ces résultats démontrent l'efficacité de notre approche et la solidité de notre pipeline de traitement des données, contribuant ainsi à l'amélioration des systèmes de reconnaissance dans des contextes réels .

## IV. Conception :

### A. Diagramme de cas d'utilisation :

Le diagramme de cas d'utilisation (DCU) est un diagramme UML utilisé pour une représentation du comportement fonctionnel d'un système logiciel. En effet, un cas d'utilisation (use cases) représente une unité discrète d'interaction entre un utilisateur (humain ou machine) et un système. Ainsi, dans un diagramme de cas d'utilisation, les utilisateurs sont appelés acteurs (actors), et ils apparaissent dans les cas d'utilisation. Ils permettent de décrire l'interaction entre l'acteur et le système. Le cas d'utilisation est une description des interactions qui vont permettre à l'acteur d'atteindre son objectif en utilisant le système.




→ Ce diagramme de cas d'utilisation illustre les interactions principales entre les photographes, les participants, et le système. Les relations d'extension garantissent que les utilisateurs passent par les étapes nécessaires pour accéder aux fonctionnalités appropriées du système.

# III. REALISATION :

## A. Outils et technologies utilisées :

**Front-End :**

- **Tailwind CSS :** Un framework CSS utilitaire qui permet de construire des interfaces utilisateur rapidement et efficacement. Plutôt que de se baser sur des composants prédéfinis, Tailwind CSS fournit des classes utilitaires à faible niveau qui peuvent être combinées pour créer des designs personnalisés. Il favorise une approche de conception modulaire et réutilisable, ce qui accélère le développement et facilite la maintenance du code CSS.
- **Next.js :** Un framework de développement pour React, conçu pour la création d'applications web performantes et optimisées. Il permet le rendu côté serveur (SSR), le rendu statique (SSG), et offre des fonctionnalités avancées comme le routage automatique, la génération de pages dynamiques et la gestion des API intégrée. Next.js est reconnu pour améliorer les performances et l'expérience utilisateur, en facilitant le développement de sites web réactifs et rapides.

**Back-End :**

- **Express.js :** Un framework minimaliste et flexible pour Node.js, utilisé pour construire des applications web et des API. Il simplifie la gestion des requêtes HTTP et offre une architecture robuste pour le développement de serveurs web. Express.js est apprécié pour sa simplicité, ses performances et son vaste écosystème de middleware.
- **Flask :** Un micro-framework web en Python, utilisé pour certaines tâches spécifiques du projet. Il permet de créer des applications web rapidement, offrant des outils pour le routage des URLs et la gestion des requêtes et des réponses.
- **Flask-RESTful :** Une extension pour Flask qui simplifie la création d'API RESTful. Elle fournit des classes et des méthodes pour gérer les endpoints de l'API de manière structurée.
- **PaddleOCR :** Une bibliothèque pour la reconnaissance optique de caractères (OCR), utilisée pour extraire les numéros de dossard des images de manière précise.

- **YOLO (You Only Look Once) :** Un modèle de détection d'objets en temps réel, développé par Ultralytics. Il permet de détecter et de localiser des objets dans les images avec une grande rapidité et précision.
- **os :** Un module de la bibliothèque standard Python pour interagir avec le système d'exploitation. Utilisé pour la manipulation des fichiers et des répertoires, l'exécution de commandes système, et la gestion des variables d'environnement.
- **cv2 (OpenCV) :** Une bibliothèque de vision par ordinateur utilisée pour le traitement des images, incluant des fonctions pour la lecture, l'écriture, la transformation, la détection et la reconnaissance d'objets.
- **uuid :** Un module de la bibliothèque standard Python pour générer des identifiants uniques, utilisé pour nommer de manière unique les fichiers téléchargés.
- **shutil :** Un module de la bibliothèque standard Python pour effectuer des opérations de haut niveau sur les fichiers, telles que la copie et le déplacement.

## B. Guide de Configuration et d'Exécution pour l'Application :

**Prérequis :**

- Node.js et npm installés sur votre machine.
- Votre application Next.js et votre serveur Express configurés dans votre répertoire de projet.

**Étapes pour exécuter votre application Next.js et le serveur Express :**

1. Naviguer vers le répertoire de votre projet :
   Ouvrez votre terminal et naviguez vers le répertoire racine de votre projet.
   → cd /chemin/vers/votre/projet
2. Installer les dépendances :
   Installez les packages npm nécessaires pour votre application Next.js et le serveur Express.
   → npm install

3. Exécuter directement le serveur :
→ node server.js
4. Démarrer le serveur de développement Next.js :
Next.js fonctionne généralement sur le port 3000. Démarrez-le avec la commande suivante :
→ npm run dev
5. Accéder à votre site web :
- Ouvrez votre navigateur web.
- Pour l'application Next.js, allez à `http://localhost:3000`.
- Si vous devez interagir directement avec le serveur Express, allez à `http://localhost:3001`.

**Prérequis :**
- Python installé sur votre machine.
- Votre application Next.js configurée comme décrit précédemment.

**Étapes pour configurer le modèle backend :**
1. Créer un nouveau répertoire pour votre API Flask :
→ mkdir flask_api
cd flask_api
2. Créer un environnement virtuel (optionnel mais recommandé) :
→ python -m venv venv
source venv/bin/activate # Sur Windows, utilisez `venv\Scripts\activate`
3. Créer un fichier `requirements.txt` :
Listez toutes les bibliothèques nécessaires :
- Flask
- flask_restful
- paddleocr
- opencv-python
- flask_cors
- ultralytics

4. Installer les dépendances :
→ pip install -r requirements.txt
6. Exécuter le serveur API Flask :
→ python app.py

## C. Fonctionnalités principales :

- **Gestion des marathons passés :**

**- Fonctionnalité :** Affichage d'une liste dynamique des marathons passés.
- **Détail :** Chaque marathon est listé avec son nom, pays, description, distance et genre des participants.

- **Téléchargement en masse des photos :**

**- Fonctionnalité :** Les photographes peuvent télécharger un grand nombre de photos d'un événement.
**- Détail :** Un bouton "upload" permet aux photographes de télécharger les photos d'un nouveau marathon. Les photos sont ensuite ajoutées dynamiquement à la base de données et associées au marathon correspondant.

- **Classification automatique des photos :**

**- Fonctionnalité :** Les photos téléchargées sont automatiquement classées par numéro de dossard.
**- Détail :** Utilisation de modèles de machine learning pour détecter les numéros de dossard sur les photos et les associer aux participants.

- **Recherche par numéro de dossard :**

**- Fonctionnalité :** Les participants peuvent rechercher leurs photos en entrant leur numéro de dossard.
**- Détail :** Les participants recherchent le nom du marathon, cliquent sur "view" pour afficher les photos du marathon, et saisissent leur numéro de dossard dans la barre de recherche pour voir leurs photos.

- **Interface conviviale pour les photographes :**

**- Fonctionnalité :** Une interface dédiée pour les photographes professionnels.
**- Détail :** Permet de gérer facilement le téléchargement et la classification des photos.

- **Interface conviviale pour les participants :**

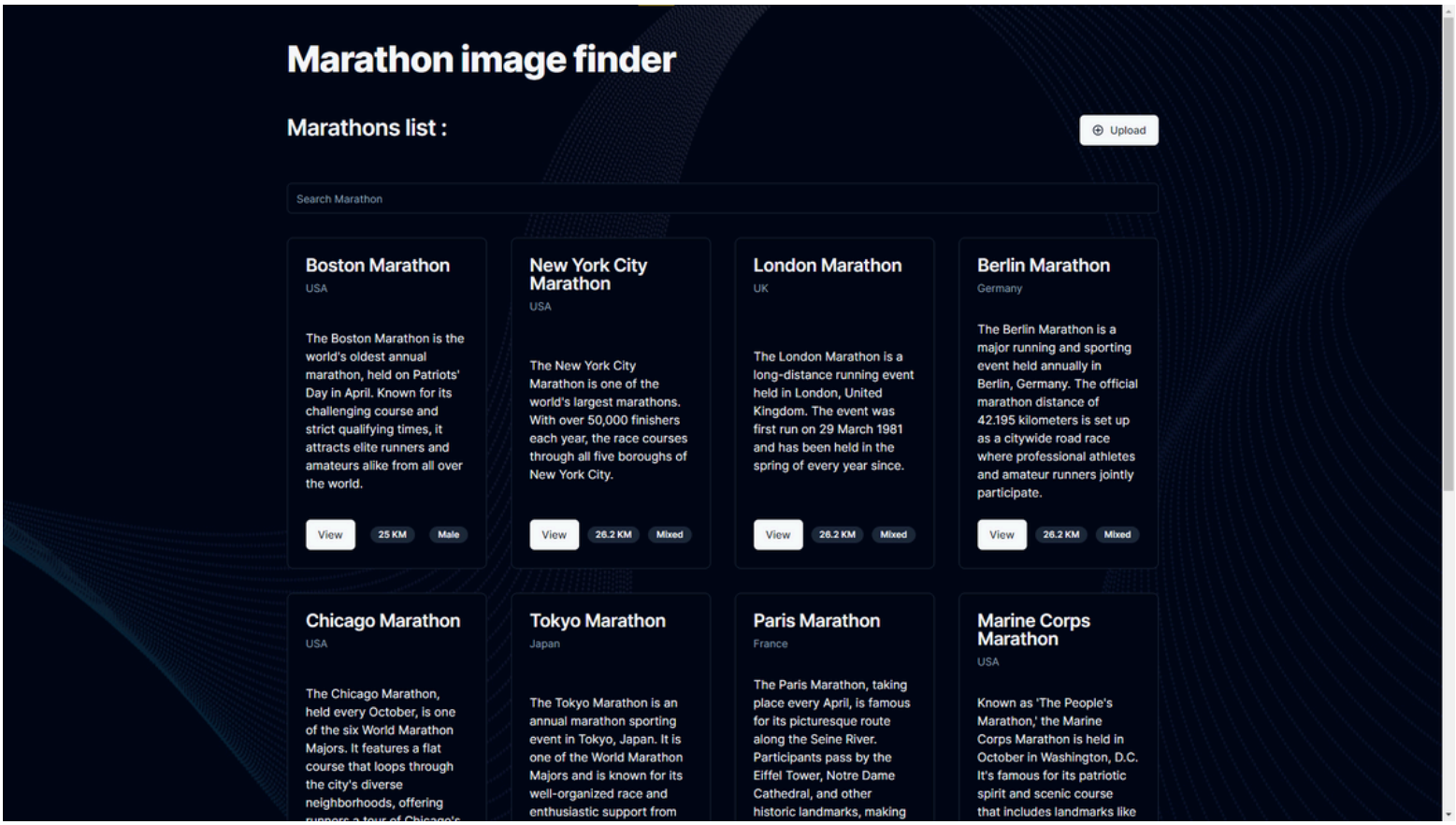
**- Fonctionnalité :** Une interface dédiée pour les participants.
**- Détail :** Permet de rechercher et accéder facilement à leurs photos personnelles.

- **Téléchargement des photos pour les participants** :**

**- Fonctionnalité :** Les participants peuvent télécharger leurs photos localement.
**- Détail :** Après avoir trouvé leurs photos en utilisant le numéro de dossard, les participants peuvent les télécharger sur leurs appareils.

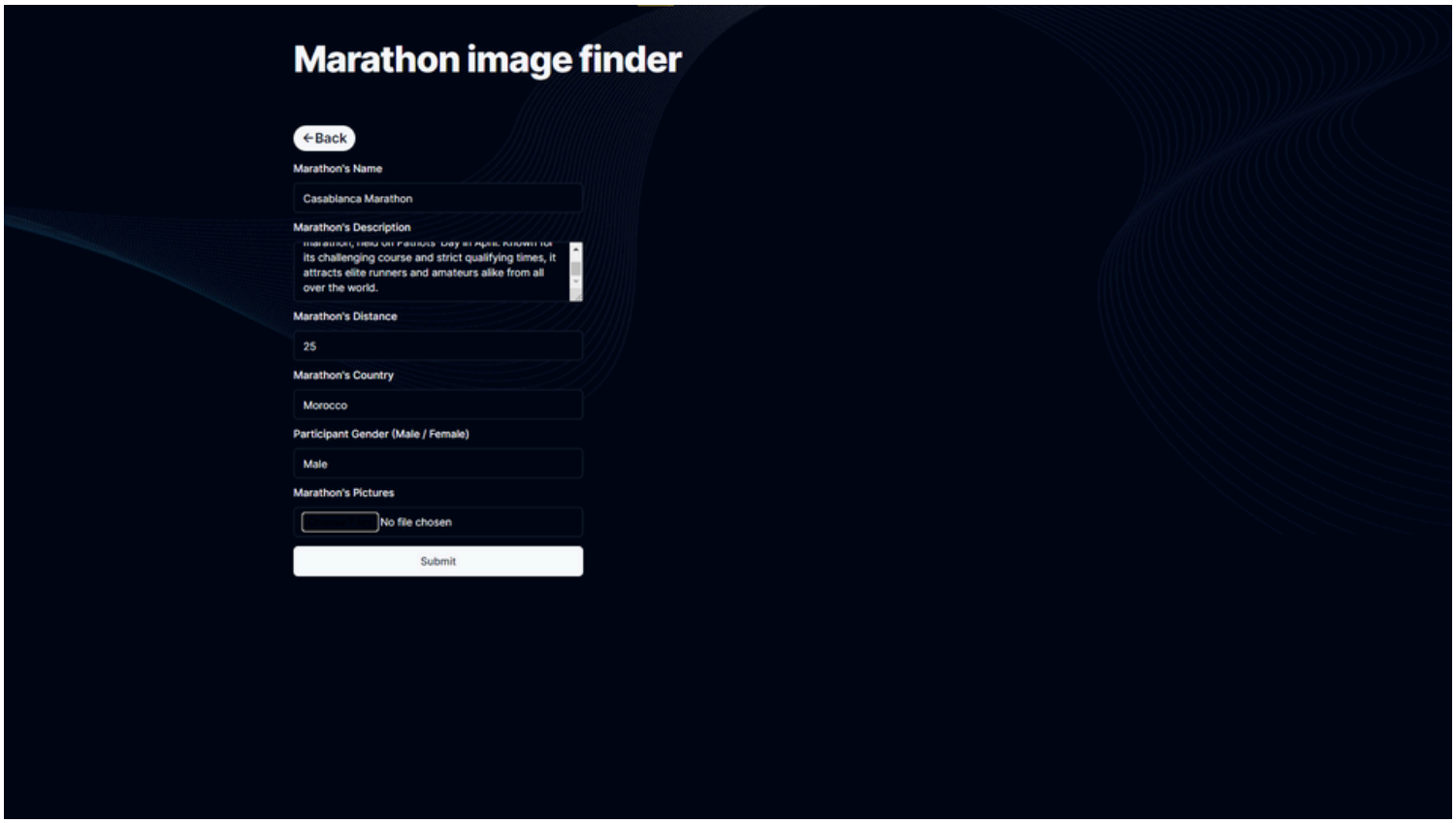
## D. Travail réalisé :

- **Page d'accueil (Home) :**

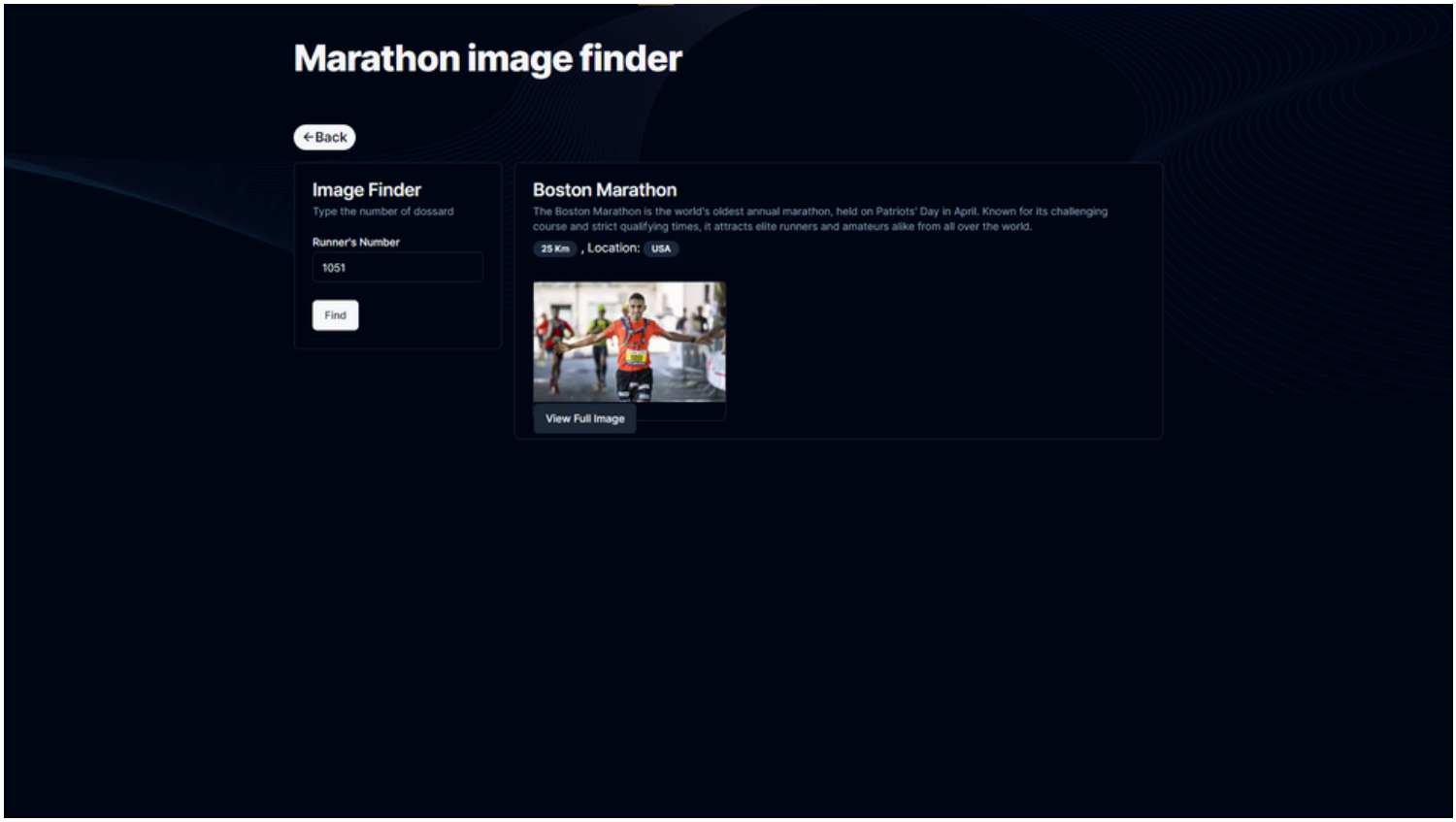


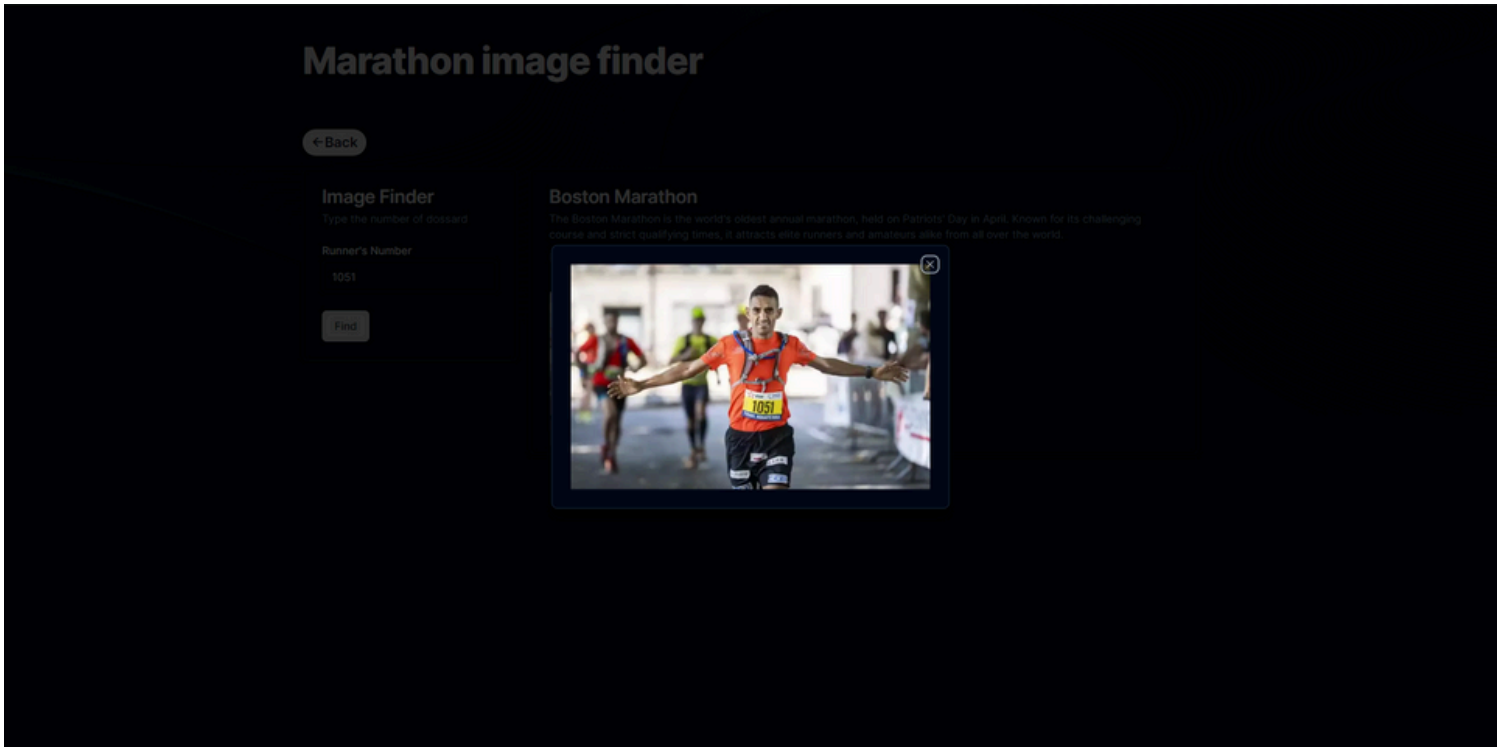

→ Cette page montre l'affichage d'une liste dynamique des marathons passés avec :

- un bouton "View" pour afficher les détails et la bare de recherche des photos par numéro de dossard,
- et un bouton "Upload" qui permet aux photographes de télécharger les photos d'un nouveau marathon.




→ Cette page apparaît lorqu'on clique sur le bouton "Upload", le photographe doit saisir les informations du nouveau marathon et uploader les photos.
Les photos téléchargées sont automatiquement classées par numéro de dossard grâce au modèle du machine learning.




→ Les participants peuvent chercher leurs photos en saisissant le numéro de dossard et les télécharger par la suite.

## E. Code Source :

```python
from flask import Flask, request, jsonify
from flask_restful import Api, Resource, reqparse
from paddleocr import PaddleOCR
from ultralytics import YOLO
import os
import cv2
import uuid
import shutil


os.environ["KMP_DUPLICATE_LIB_OK"] = "TRUE"

# Initialize the OCR and YOLO models
ocr = PaddleOCR(use_angle_cls=True, lang='en')
model = YOLO("best.pt")
# model.to('cuda')

app = Flask(__name__)
api = Api(app)

parser = reqparse.RequestParser()
# Add the arguments that you expect in the dictionary
parser.add_argument('images', type=list, location='json')
parser.add_argument('marathon_name', type=str, location='json')

# Directory to save uploaded images
UPLOAD_FOLDER = 'uploaded_images'
if not os.path.exists(UPLOAD_FOLDER):
    os.makedirs(UPLOAD_FOLDER)

response_json = []
```

```python
def detect_bib_number(image):
    results = model.predict(image, imgsz=2016)
    return results

def extract_text(result):
    first_elements = [text[1] for sublist in result for text in sublist]
    return first_elements

def remove_non_alphanumeric(text):
    return ''.join(char for char in text if char.isalnum())

def filter_wanted_text(extracted_text, text_confidence_score):
    # Remove text with low confidence score
    text = [x for x in extracted_text if x[1] > text_confidence_score]
    # Get only the text
    text = [x[0] for x in text]
    # Remove non-alphanumeric characters
    text = [remove_non_alphanumeric(x) for x in text]
    # Remove text that are not digit
    text = [x for x in text if x.isdigit()]
    return text[0] if text else None
```

```python
@app.route('/upload', methods=['POST'])
def upload_images():
    # Parse the arguments from the incoming HTTP request
    args = parser.parse_args()

    # Access the values of the arguments
    filepaths = args['images']
    marathon_name = args['marathon_name']
    text_confidence_score = 0.3  # Default confidence score

    for filepath in filepaths:
        filename = os.path.basename(filepath)
        new_filepath = os.path.join(UPLOAD_FOLDER, filename)
        shutil.copy(filepath, new_filepath)
        image = cv2.imread(new_filepath)
        print(f'Processing image: {new_filepath}...')
        results = detect_bib_number(new_filepath)
        print(f'Number of detected bounding boxes: {len(results)}')

        bib_numbers = []
        for result in results:
            for box in result.boxes:
                x1, y1, x2, y2 = map(int, box.xyxy[0])
                roi = image[y1:y2, x1:x2, :]
                bib_number = ocr.ocr(roi, cls=True)

                try:
                    extracted_text = extract_text(bib_number)
                    filtered_text = filter_wanted_text(extracted_text, text_confidence_score)
                    if filtered_text:
                        bib_numbers.append(filtered_text)
                        bib_numbers = list(set(bib_numbers))
                except:
                    pass

        response_json.append({'marathon_name': marathon_name, 'image_path': new_filepath, 'detected_numbers': bib_numbers})

    return jsonify({'message': 'Images uploaded and processed successfully'})

@app.route('/search', methods=['GET'])
def search_bib_number():
    desired_bib_number = request.args.get('desired_bib_number')
    marathon_name = request.args.get('marathon_name')
    results = []

    for record in response_json:
        if desired_bib_number in record['detected_numbers'] and record['marathon_name'] == marathon_name:
            results.append(record['image_path'])

    return jsonify({'desired_bib_number': desired_bib_number, 'marathon_name': marathon_name, 'image_paths': results})

if __name__ == '__main__':
    app.run(debug=True)
```

# CONCLUSION GENERALE :

La réalisation du projet "Marathon Photo Finder" a été une expérience enrichissante et stimulante. Nous tenons à exprimer notre profonde gratitude envers notre professeur pour ses conseils avisés, son encouragement constant et sa précieuse guidance tout au long du développement de ce projet. Son expertise a été une source d'inspiration et a contribué de manière significative à la qualité et au succès de notre travail.

Nous souhaitons également remercier l'ensemble de l'équipe pour son dévouement et sa collaboration exemplaire. Chaque membre a apporté sa pierre à l'édifice, et ce projet n'aurait pas pu aboutir sans l'effort collectif et la détermination de chacun.

Enfin, nous sommes reconnaissants envers toutes les parties prenantes qui ont contribué de près ou de loin à la réalisation de ce projet. Grâce à cette collaboration, "Marathon Photo Finder" est devenu une application efficace pour la gestion des photos de marathons, offrant une solution pratique et innovante pour les photographes et les participants.

Avec "Marathon Photo Finder", nous avons réussi à simplifier la recherche de photos de course pour les participants et à offrir une interface conviviale aux photographes, facilitant ainsi la gestion des événements sportifs. Ce projet illustre le pouvoir des technologies de machine learning et de reconnaissance d'images pour améliorer les expériences utilisateur dans des contextes réels.