



Université Libanaise  
Faculté de Technologie

## LICENCE II

Année universitaire 2024 - 2025

Génie des réseaux informatiques et de  
télécommunications

# TFC Mangemnet System

Préparé par RAMMAL Lyn  
du 17/3/2025 au 17/5/2025  
À



## **Remerciements**

Je tiens à exprimer ma sincère reconnaissance envers toutes les personnes qui ont joué un rôle clé dans la réalisation de ce rapport de stage. Leur soutien indéfectible et leur précieuse contribution ont été pour moi une source inestimable d'encouragement et d'inspiration.

Tout d'abord, Je tiens à exprimer ma sincère gratitude à toute l'équipe d'ENSEMBLE POUR CHEHIM pour l'opportunité de stage que j'ai pu réaliser, axée sur le développement d'application web TFC management system .

Je remercie chaleureusement Dr Nassib ABDALLAH pour son soutien précieux, ses encouragements constants et ses orientations avisées tout au long de l'élaboration de ce rapport.

Je souhaite également exprimer ma reconnaissance et ma sincère gratitude à toute l'équipe d'ENSEMBLE POUR CHEHIM pour l'opportunité de stage que j'ai pu réaliser, axée sur le développement d'application web TFC management system .

Un merci particulier au Rana SAAB et au Dr. Aya HAGE CHEHADE, dont l'aide et le soutien constants m'ont permis d'acquérir de nouvelles compétences tout au long de mon stage. Cette expérience m'a non seulement permis de développer mes compétences pratiques, mais aussi d'enrichir mon expérience de travail virtuel et responsabilité du travail

Je tiens également à remercier profondément le Professeur Abd El Salam Al Hajjar ainsi que tous mes enseignants à l'Université Libanaise Faculté de Technologie. Leur enseignement enrichissant a joué un rôle crucial en me motivant à améliorer constamment mes compétences

Enfin, je tiens à exprimer ma gratitude à chacun d'entre vous pour avoir contribué à cette étape importante de ma vie académique et professionnelle. Votre soutien et vos encouragements ont été essentiels à ma progression et à la réussite de cette expérience.

Merci à tous pour cette belle expérience et pour avoir enrichi mon parcours académique

# Plan

<b>Résumé.....</b>	<b>6</b>
<b>Mots clés :.....</b>	<b>6</b>
<b>Introduction.....</b>	<b>1</b>
• Contexte.....	1
• Problématiques et motivation.....	1
• Objectifs à atteindre.....	1
• Plan du Rapport.....	1
<b>Presentation of Collaborative Research and Projects Committee (CRPC): Vision, Expertise, and Impact.....</b>	<b>2</b>
<b>1. Bilan du Stage.....</b>	<b>3</b>
1.1 Analyse Fonctionnelle.....	3
1.1.1 Détermination de l'objectif : méthode QQOQCP.....	3
1.1.2 Définition du besoin.....	4
1.1.2.1 Contexte.....	4
1.1.2.2 Besoins identifiés.....	4
❖ Diagramme bête à cornes.....	5
❖ Diagramme de pieuvre.....	5
1.1.3 Cahier des charges.....	5
1. Contexte du projet.....	5
❖ Diagramme de Gantt.....	6
Tâches et Livrables.....	7
<b>2 . Etat de l'art.....</b>	<b>7</b>
2.1 Logiciels Utilisés.....	7
Figma – Outil de design UI/UX.....	7
Draw.io (ou diagrams.net) – Création de diagrammes.....	7
Visual Studio Code – Éditeur de code.....	8
2.2 Langage de programmation.....	8
1-Frontend : React, Axios, Bootstrap.....	8
2-Backend : Django et Django REST Framework.....	8
3-base de Donnée : PostgreSQL.....	10
<b>3. Travaux Réalisés.....</b>	<b>10</b>
3.1 Introduction.....	10
3.2 Interface utilisateur, Les pages du system.....	10
3.3 Conception UI avec Figma.....	11
3.4 Structure de la Base de Données.....	12
3.5.Structure Du projet.....	14
A-Page : Authentification.....	14
B-Barre de Navigation (Navbar).....	15
C. Page : Accueil (Home).....	15
E-Page Projects.....	19
→ Formulaire d'ajout d'un projet.....	22
→ Fonctionnalité "Ajouter un nouveau superviseur".....	26

F- PAGE : Modification d'un Projet.....	29
→ FORMULAIRE : Ajout d'un nouveau stagiaire au projet.....	31
→ FORMULAIRE Modification des informations d'un stagiaire affecté à un projet : .....	34
→ FORMULAIRE Suppression d'un stagiaire affecté à un projet.....	35
F-Page Détails du Projet.....	37
G- Page Stagiaire.....	38
→ FORMULAIRE : Ajout d'un nouveau stagiaire au projet.....	39
→ FORMULAIRE : Modification d'un stagiaire.....	40
H- Page Details.....	42
→ FORMULAIRE : suprématie d'un stagiaire d'un stage.....	44
I-Page Supervisors.....	44
→ FORMULAIRE : Ajout d'un nouveau superviseur.....	46
→ FORMULAIRE : Modification d'un superviseur.....	48
H- Page Details -Superviser.....	49
I- Page Members.....	50
→ FORMULAIRE : Ajout d'un nouveau superviseur.....	51
→ FORMULAIRE : Modification d'un membre.....	54
J- Page details Members.....	54

## **Liste des figures**

(Cette partie est obligatoire)

## **Liste des tableaux**

(Cette partie est obligatoire)

## Résumé

Dans le cadre de mon stage, j'ai participé au développement d'un système de gestion numérique intitulé **TFC Management System**, visant à moderniser et centraliser la gestion des activités d'une association. Ce projet m'a permis de travailler à la fois sur le **front-end** et le **back-end** d'un site web, consolidant ainsi mes compétences en **programmation web**, mais aussi en **sécurité, performance et gestion de bases de données**.

L'objectif principal de ce stage était de concevoir et de développer une **plateforme numérique complète** permettant de gérer de manière centralisée toutes les facettes de l'association : gestion des membres, des projets , des interactions..

## Mots clés :

Développement web Sécurité, informatique, Base de données, Centralisation de l'information, Interaction, utilisateur ,Système d'information, Fiabilité

# Introduction

- Contexte

Le projet **TFC Management System** s'inscrit dans le cadre de la digitalisation des processus de gestion d'une association. Une première version de la plateforme avait déjà été réalisée par un développeur précédent. Mon stage a consisté à reprendre ce projet existant afin de le corriger, l'améliorer et le rendre pleinement fonctionnel. Cette plateforme vise à centraliser la gestion des membres, des superviseurs, des stagiaires et des projets, et des interactions internes de l'association via une application web complète et sécurisée.

- Problématiques et motivation

L'un des défis majeurs rencontrés dès le début du stage a été de **comprendre un code existant**, d'en identifier les **bugs** et de les **corriger efficacement**. Cette étape est cruciale dans la réalité professionnelle, car un bon développeur ne crée pas uniquement des applications : il doit aussi être capable d'**analyser, maintenir et améliorer des systèmes existants**.

Par ailleurs, la plateforme devait répondre à plusieurs contraintes techniques et fonctionnelles, notamment :

- La gestion d'une base de données complexe avec des **relations hiérarchiques** (multi-héritage).
- La mise en place d'un système d'**authentification sécurisé**.
- La **validation rigoureuse des données** saisies par les utilisateurs.

Ces problématiques ont constitué une réelle **source de motivation** pour moi, en me permettant de me confronter à des exigences professionnelles concrètes, tout en approfondissant mes compétences en développement full-stack.

- Objectifs à atteindre

Les objectifs principaux de ce stage étaient les suivants :

- **Corriger les bugs** dans la version initiale du système et stabiliser l'ensemble du code.
- **Renforcer la sécurité** du site web à travers la mise en place d'un système d'authentification robuste (gestion des rôles, validation des entrées).
- **Optimiser la base de données**, en utilisant des modèles avancés avec **multi-héritage** et des relations bien définies entre les entités.
- **Assurer la cohérence fonctionnelle** entre le front-end et le back-end pour garantir une expérience utilisateur fluide et fiable.
- Déployer une **architecture web full-stack sécurisée**, capable de gérer efficacement les données sensibles et les différents profils utilisateurs de l'association.

- Plan du Rapport

Le rapport s'articule autour de plusieurs grandes sections. Il débute par une présentation de l'association, qui met en avant son engagement dans la recherche et les initiatives culturelles. Ensuite, le bilan du stage est détaillé, comprenant une analyse fonctionnelle, la définition des besoins et un cahier des charges qui précise les tâches à réaliser. Suit une section sur l'état de l'art, qui explore les méthodes et technologies pertinentes pour le développement d'une application web. Ensuite, les travaux réalisés sont présentés, mettant en lumière les résultats obtenus et les défis rencontrés. Enfin, le rapport se conclut par une conclusion qui résume les enseignements tirés et propose des recommandations.

# Presentation of Collaborative Research and Projects Committee (CRPC): Vision, Expertise, and Impact

Collaborative Research and Projects Committee (CRPC), established in October 2023 under the non-profit association Ensemble pour Chehim/ Together For Chehim (founded in 2015), is a multidisciplinary academic and research platform committed to fostering scientific excellence, collaborative innovation, and societal advancement. Based in Chehim, Lebanon, the CRPC provides structured academic programs, promotes applied research, and builds strategic international partnerships to support students, early-career researchers, and professionals.

The CRPC is led by Dr. Nassib Abdallah (President of the Committee) and Dr. Aya Hage Chehade (Vice-President of the Committee), and brings together a multidisciplinary network of PhD holders, engineers, and other qualified professionals affiliated with research laboratories, industry, and academic institutions across Lebanon and France. This diverse expertise strengthens the committee's ability to achieve its strategic objectives. Together, they oversee the development and implementation of the CRPC's academic programs, research initiatives, and community outreach activities.

## Strategic Vision and Mission

The CRPC's activities are guided by four fundamental pillars:

- Academic training and supervision: More than 370 hours of free, in-person training have been delivered locally, tailored to the latest scientific and technological needs.
- International collaboration: Strategic agreements with universities, industries, and research institutions support knowledge exchange and joint project development.
- Widening access to higher education: Special focus is given to underprivileged and geographically remote populations to enhance educational equity.
- Continuous scientific innovation: CRPC actively addresses emerging technological and societal challenges through interdisciplinary research and agile innovation strategies.

This comprehensive mission includes a strong commitment to mentorship, capacity building, community outreach, and research leadership.

## Domains of Expertise

CRPC engages in high-impact scientific and technical fields through both education and applied research:

1. Artificial Intelligence and Data Science
  - Machine learning, deep learning, and intelligent systems
  - Applications in healthcare, cybersecurity, and meteorological modeling
2. Computer Science and Software Development
  - Algorithm design, programming foundations, and software engineering
  - Emphasis on project-based learning and collaborative development environments
3. Electronics and Robotics
  - Embedded systems, Arduino-based robotics, automation
  - Prototyping, electronics integration, and practical system development
4. Industrialization and Systems Engineering
  - Bridging innovation and real-world implementation
  - Emphasis on smart systems, industrial automation, and applied technologies
5. Foreign Language Training and Soft Skills Development
  - English language training, including SAT preparation
  - Communication skills, teamwork, and leadership development

These domains are addressed through structured academic curricula, hands-on workshops, and collaborative research activities.

# 1. Bilan du Stage

## 1.1 Analyse Fonctionnelle

### 1.1.1 Détermination de l'objectif : méthode QQOQCP

a) Quoi ?

a.1) Qu'est-ce que c'est ?

Le projet consiste à développer une plateforme numérique sécurisée destinée à la gestion des membres, des projets, des stagiaires, des superviseurs, ainsi que du paiement annuel au sein d'une association.

a.2) Objectif du travail :

L'objectif principal est de concevoir un système centralisé permettant de regrouper, organiser et conserver toutes les informations essentielles de manière fiable, rapide et sécurisée. Il vise à éviter la perte de données, à faciliter leur consultation et à améliorer l'efficacité de la gestion administrative de l'association.

b) Qui ?

b.1) Qui est impliqué ?

Les utilisateurs de la plateforme sont principalement :

- Le président ou l'administrateur de l'association, chargé de la gestion globale.
- Les superviseurs qui sont des membres, qui peuvent consulter les projets et stagiaires dont ils sont responsables.

c) Où ?

c.1) Où est-ce utilisé ?

La plateforme est utilisée au sein de l'association. L'accès est prévu pour les administrateurs, les superviseurs, les membres via une interface web. Chaque utilisateur accède aux fonctionnalités selon son rôle et ses autorisations.

d) Quand ?

d.1) Quand est-ce utilisé ?

Le système est utilisé lors de :

- L'ajout de nouveaux projets, stagiaires ou superviseurs, stages.
- La mise à jour ou la consultation des informations administratives.
- Les échanges de données ou la génération de résumés
- La gestion annuelle des paiements des membres.

e) Comment ?

e.1) Comment ça fonctionne ?

Le projet est développé en architecture full stack, avec une interface utilisateur intuitive et accessible à tous les utilisateurs. Il comprend un backend et un frontend ergonomique qui permettent d'effectuer les opérations de gestion de manière fluide. Des systèmes de filtrage, de recherche pour faciliter la navigation et la prise de décision.

f) Pourquoi ?

f.1) Pourquoi est-ce important ?

La mise en place de ce système est essentielle pour plusieurs raisons :

- Centralisation des données et meilleure organisation interne.
- Sécurisation et pérennité des informations sensibles.
- Gain de temps considérable dans la gestion administrative.
- Amélioration de la communication entre les membres de l'association.
- Professionnalisation du fonctionnement de l'association, notamment en matière de suivi des projets et des stagiaires

### 1.1.2 Définition du besoin

#### 1.1.2.1 Contexte

Dans une association composée d'un président, de membres et de superviseurs, il devient difficile de gérer toutes les informations de manière manuelle. Chaque projet est encadré par un ou plusieurs superviseurs (qui peuvent aussi être membres) et réalisé par un ou plusieurs stagiaires. Un stagiaire peut faire plusieurs stages, et chaque stage doit contenir des fichiers importants comme la convention (PDF), le rapport, la présentation, le certificat et un code de validation.

En plus, les membres doivent payer une cotisation chaque année. Ils ont 10 jours pour faire le paiement, et l'association doit donner un reçu automatiquement. Pour éviter les erreurs, les pertes d'information ou les retards, il est important de créer un système numérique simple et sécurisé.

#### 1.1.2.2 Besoins identifiés

##### 1 Centralisation et structuration des données des membres, projets et stages

- *Problème* : Informations éparpillées ou perdues dans des fichiers manuels ou non interconnectés.
- *Besoin* : Concevoir une base de données unique et relationnelle pour gérer efficacement les membres, les projets, les stagiaires et leurs interactions

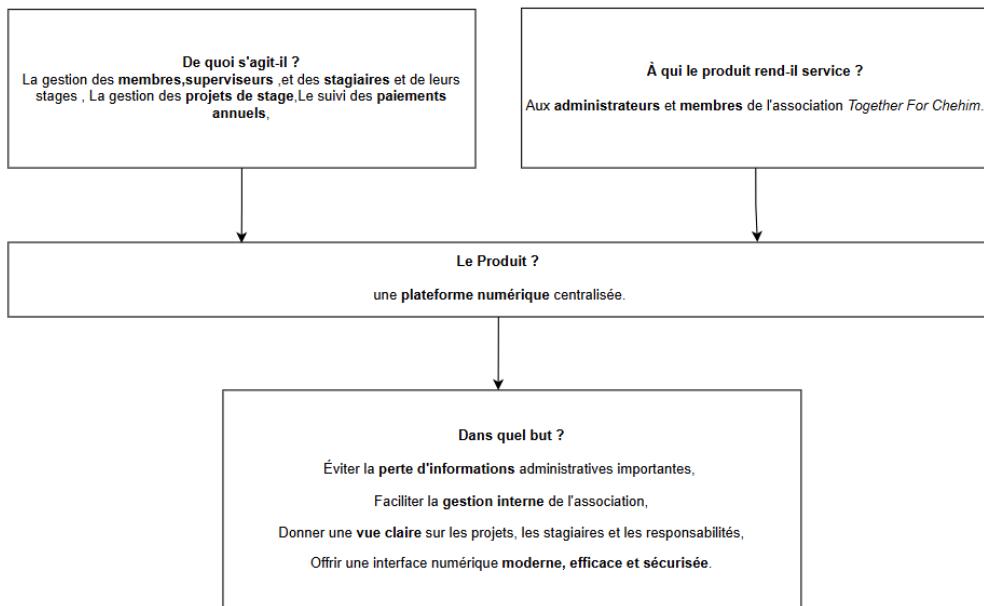
##### 2 Gestion dynamique et sécurisée des stages

- *Problème* : Suivi difficile des affectations stagiaire-projet, gestion manuelle des documents, et validation peu structurée.
  - *Besoin* : Mettre en place un système numérique permettant l'upload, la visualisation et la certification des documents de stage (convention, rapport, certificat, etc.).
- Automatisation de la gestion des paiements annuels**
- *Problème* : Suivi manuel des paiements, non-respect des délais, aucune preuve automatisée de paiement.
  - *Besoin* : Système de rappels pour les paiements avec suivi des échéances et génération automatique de reçus PDF.

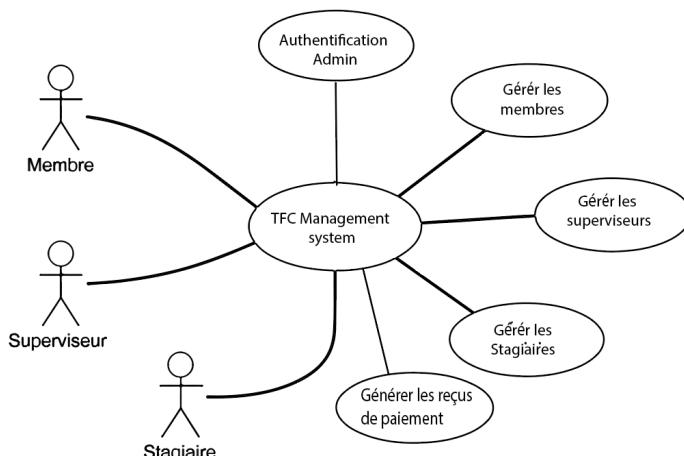
##### 3 Interface utilisateur intuitive et accessible

- *Problème* : Complexité d'utilisation des systèmes existants ou manque de compatibilité avec les rôles des utilisateurs.
- *Besoin* : Développer une interface web claire, responsive et adaptée à chaque profil d'utilisateur (superviseur, stagiaire).

## ❖ Diagramme bête à cornes



## ❖ Diagramme de pieuvre



F2

### 1.1.3 Cahier des charges

#### 1. Contexte du projet

L'association compte plusieurs types d'acteurs : des membres, des superviseurs, des stagiaires, et un président qui assure la gestion globale. Dans la pratique, **une même personne peut occuper plusieurs rôles** à la fois. Par exemple, un **membre** peut également être **superviseur** ou même **stagiaire**. Cette superposition des rôles rend la gestion manuelle longue, confuse et sujette aux erreurs.

De plus, la gestion actuelle des projets de stage se fait de manière fragmentée : les informations sont dispersées, les documents sont échangés par e-mail ou en version papier, et le suivi

administratif (notamment des paiements annuels et des documents de stage) est difficile à automatiser.

Face à cette situation, il est devenu nécessaire de mettre en place un **système numérique centralisé et intelligent**, capable de gérer ces différents rôles de manière flexible tout en assurant la sécurité, la fiabilité et l'accessibilité des données.

## 2. Objectifs détaillés du projet

Le projet a pour but de développer une plateforme web complète permettant :

### a) La gestion unifiée des utilisateurs

- Un même individu peut être **membre, superviseur et/ou stagiaire**, selon le contexte.
- La plateforme doit reconnaître les **rôles multiples** sans créer de doublons, grâce à une structure de données centralisée.

### b) La gestion des projets et des stages

- Création et suivi des projets avec enregistrement de la date, description, et assignation d'un ou plusieurs superviseurs
- Attribution d'un ou plusieurs stagiaires par projet, avec la possibilité qu'un stagiaire participe à plusieurs stages à des périodes différentes.

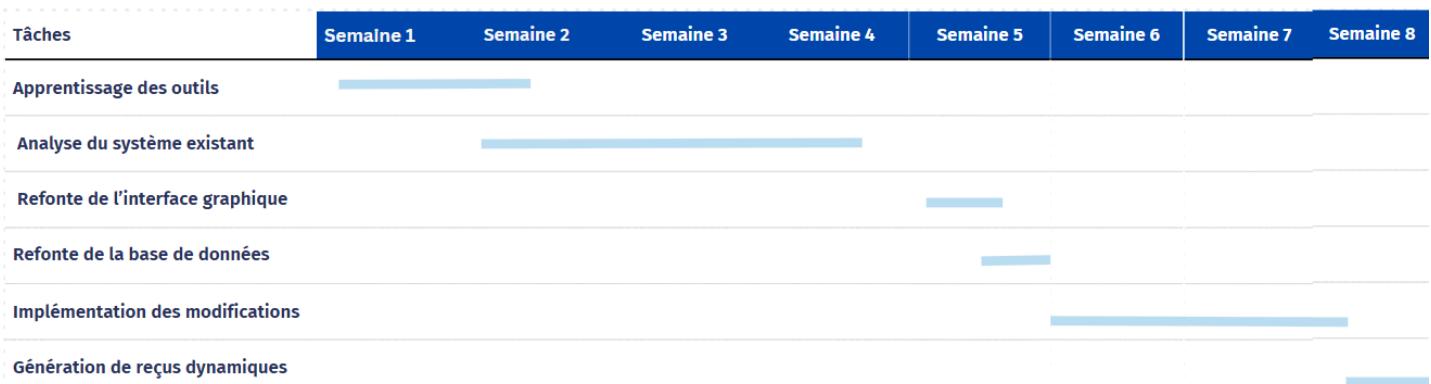
### c) Le suivi administratif des stages

- Téléversement des documents liés aux stages : convention, rapport, code source, présentation, certificat.

### d) La gestion des paiements annuels

- Suivi des membres ayant payé leur cotisation annuelle.
- Notification automatique des membres ayant moins de 10 jours pour payer.
- Génération dynamique d'un reçu personnalisé après paiement.

## ❖ Diagramme de Gantt



## Tâches et Livrables

Au cours du stage, plusieurs tâches ont été planifiées et réalisées dans le but d'améliorer le système existant et de proposer de nouvelles fonctionnalités adaptées aux besoins de l'association. Voici les principales étapes :

### Tâche 1 : Apprentissage des outils nécessaires

- Revoir les bases de **Django** pour le développement backend.
- Révision de **Bootstrap** et **React** pour améliorer l'interface utilisateur.

### Tâche 2 : Analyse du système existant

- Comprendre la logique du système déjà développé par un ancien stagiaire.
- Étudier la structure de la base de données et le fonctionnement des modules.
- Identifier les bugs ou failles et proposer des améliorations.
- Apprentissage du **Django REST Framework** pour la création d'API.

### Tâche 3 : Refonte de l'interface graphique

- Création de maquettes sur **Figma** pour proposer un nouveau style plus moderne et ergonomique.
- Intégration du design avec React/Bootstrap.

### Tâche 4 : Refonte de la base de données

- Mise en place d'une nouvelle structure de base de données basée sur le principe de l'**héritage multiple** (multi-inheritance).
- Optimisation des modèles pour mieux représenter les relations entre membres, superviseurs, stagiaires, etc.

### Tâche 5 : Implémentation des modifications

- Application des changements dans le système existant.
- Création ou mise à jour des **API** nécessaires.
- Mise à jour des **modèles Django** et synchronisation avec la nouvelle base de données.

### Tâche 6 : Génération de reçus dynamiques

- Apprentissage du système de **templates Django**.
- Développement d'un module de génération de **reçus PDF** dynamiques pour les paiements des membres

## 2 . Etat de l'art

### 2.1 Logiciels Utilisés

Pendant la réalisation de ce projet, plusieurs logiciels ont été utilisés pour concevoir, développer et documenter le système. Voici les principaux :

#### Figma – Outil de design UI/UX

- Utilisé pour créer des maquettes modernes
- Permet de visualiser l'apparence des pages avant leur développement.
- Site officiel : <https://www.figma.com/>



#### Draw.io (ou diagrams.net) – Crédit de diagrammes

- Utilisé pour dessiner la structure de la base de données (diagrammes Entité-Association) ainsi que d'autres schémas fonctionnels comme le **diagramme bête à cornes**
- Accessible en ligne et gratuit.
- Site officiel : <https://www.draw.io/>



## Visual Studio Code – Éditeur de code

- Environnement de développement léger mais puissant pour écrire le code en **Django, Python, HTML, CSS, JavaScript**, etc.
- Possède de nombreuses extensions pour faciliter le développement web.
- Téléchargement : <https://code.visualstudio.com/>



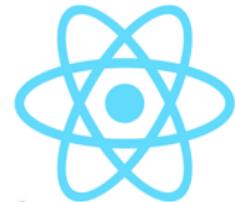
## 2.2 Langage de programmation

Le projet utilise plusieurs langages et technologies modernes côté frontend (React) et backend (Django), en s'appuyant sur des bibliothèques et frameworks pour faciliter le développement web complet.

### 1-Frontend : React, Axios, Bootstrap

#### React.js

Le frontend est développé avec **React**, une bibliothèque JavaScript utilisée pour construire des interfaces utilisateur dynamiques et modulaires. Elle permet de diviser l'interface en **composants réutilisables** et de gérer efficacement l'état de l'application.



#### Axios

Pour communiquer avec le backend (API Django), on utilise **Axios**, une bibliothèque JavaScript qui permet d'envoyer des requêtes HTTP (GET, POST, PUT, DELETE) aux endpoints définis dans Django REST Framework.



#### Bootstrap & Font Libraries

Le design est amélioré avec **Bootstrap React** ainsi que des **librairies et d'icônes** pour une meilleure présentation.

## 2-Backend : Django et Django REST Framework

Le backend de ce projet est construit avec **Django**, un framework web basé sur Python, et **Django REST Framework (DRF)**, un outil puissant pour créer des API RESTful. Cette combinaison permet de gérer efficacement la logique métier, les échanges de données, la sécurité, et la communication avec le frontend (React).



#### Django (Python)

Django est un framework **rapide, sécurisé et robuste**, utilisé pour construire des applications web complètes. Voici ses principales fonctions :

- **Configuration des URLs** : les routes sont définies pour faire correspondre les URLs aux fonctions ou classes qui les traitent.
- **Applications modulaires** : Django permet de structurer le projet en plusieurs "apps" indépendantes (ex : app stage, app myapi, ).
- **Modèles et base de données** : chaque table de la base est représentée par une **classe Python** (modèle). Cela permet une **abstraction via l'ORM** (Object-Relational Mapping), évitant l'écriture de requêtes SQL manuelles.
- **Sécurité contre les injections SQL** : grâce à l'ORM, Django protège automatiquement les requêtes des attaques par injection SQL.
- **Interface d'administration** : Django génère automatiquement une interface web d'admin pour gérer les utilisateurs et les données du projet sans coder d'interface personnalisée.

## Django REST Framework (DRF)

DRF transforme Django en **API web RESTful**, accessible par des clients comme React ou Postman. Il permet de structurer les données et les interactions via des endpoints clairs et sécurisés.

### ViewSets

Les **ViewSets** sont des classes qui regroupent les opérations de base :

- GET → Récupérer la liste (list) ou un seul objet (retrieve)
- POST → Créer un nouvel objet (create)
- PUT/PATCH → Modifier un objet (update, partial\_update)
- DELETE → Supprimer un objet (destroy)

Ces vues sont déjà créées automatiquement en utilisant ModelViewSet, ce qui évite d'écrire du code répétitif.

### Serializers

Les **serializers** font le lien entre :

- les **objets Python/Django** (venant de la base de données)
- et le **format JSON** (utilisé côté frontend ou API)

Ils fonctionnent dans les **deux sens** :

- Python → JSON (pour l'envoi de données vers le frontend)
- JSON → Python (pour la réception et validation des données envoyées par l'utilisateur)

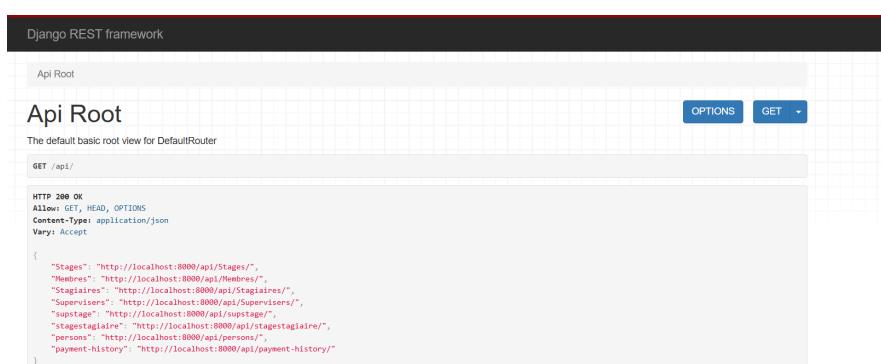
### Filtres, Recherche et Pagination

DRF propose :

- des **filtres** personnalisés (ex : ?nom=Ali)
- une **pagination automatique** pour ne pas surcharger les réponses (par exemple, afficher 5 projets par page)

### Interface Web DRF

DRF offre une **interface graphique intégrée**, pratique pour **tester rapidement** les endpoints (GET, POST, etc.) sans avoir besoin d'un outil externe.



## Authentification et Sécurité

Le système inclut un mécanisme d'authentification basé sur les **JSON Web Tokens (JWT)** :

- Lorsqu'un utilisateur se connecte, il reçoit un **token d'accès** et un **token de rafraîchissement**.
- Ces tokens sont utilisés pour sécuriser l'accès aux routes protégées de l'API (selon le type d'utilisateur).

## 1. Démarrer le serveur de développement Django

`python manage.py runserver`

- Cette commande lance le serveur local à l'adresse :  
<http://127.0.0.1:8000> par défaut.

## 2. Créer un superutilisateur (admin)

`python manage.py createsuperuser`

- Cette commande permet de **créer un compte administrateur** avec accès à l'interface d'administration Django.
- Elle te demandera :
  - un nom d'utilisateur
  - une adresse e-mail
  - un mot de passe (deux fois)

## 1. Appliquer les changements de la base de données

a) Créer une migration (après avoir modifié un modèle)

`python manage.py makemigrations`

b) Appliquer les migrations à la base de données

`python manage.py migrate`

Ces deux commandes permettent de **mettre à jour la base de données** après toute modification dans les modèles (models.py).

## 2. Accéder au shell Django

`python manage.py shell`

Ce shell permet de tester **des requêtes Python avec l'ORM Django**.

Exemple :

```
from myapp.models import Membre  
Membre.objects.all()
```

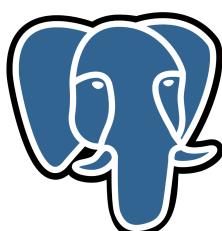
## 3-base de Donnée : PostgreSQL

Commande SQL TRUNCATE

La commande SQL TRUNCATE permet de **supprimer toutes les données d'une table sans supprimer la table elle-même**. C'est plus rapide que DELETE FROM  
`TRUNCATE TABLE nom_de_la_table RESTART IDENTITY CASCADE;`

Explication des options :

- RESTART IDENTITY : remet les identifiants auto-incrémentés à zéro (id = 1, etc.).
- CASCADE : supprime aussi les données des tables liées par des clés étrangères.



## 3. Travaux Réalisés

### 3.1 Introduction

Dans le cadre de ce projet, plusieurs tâches ont été réalisées pour améliorer ce système. Le travail a porté à la fois sur le frontend (React/Bootstrap) et le backend (Django/DRF), en plus de la conception de l'interface utilisateur, la modélisation de la base de données, et la démonstration fonctionnelle du système.

### 3.2 Interface utilisateur, Les pages du système

L'interface a été développée avec **React** et stylisée avec **Bootstrap** pour assurer une navigation fluide. Voici les principales pages mises en place :

- **Page d'authentification** : Permet la connexion sécurisée selon le type d'utilisateur (admin ou membre).
- **Navbar statique** : Composant de navigation affiché sur toutes les pages après authentification.
- **Page d'accueil (Home)** : Affiche les projets non attribués, les membres ayant payé, ceux qui ont 10 jours pour payer, et les stages proches de la fin.
- **Page des projets** : Liste tous les projets enregistrés. Chaque projet doit avoir au moins un superviseur principal.
- **Page des stagiaires** : Affiche les stagiaires associés à chaque projet.
- **Page des superviseurs** : Gestion des superviseurs membres ou non-membres.
- **Page des membres** : Liste des membres avec leurs informations et leur statut de paiement

### 3.3 Conception UI avec Figma

Avant le développement, une maquette des pages principales a été conçue avec **Figma** :

- Page de login:



- Page d'accueil:

- Page projets et les autres pages : suivent le même style visuel

- Les formulaire et les PopUp

**Add Project Details**

Enter Title

Enter Domain

Enter Speciality

Choose File image\_file.jpg

Select date

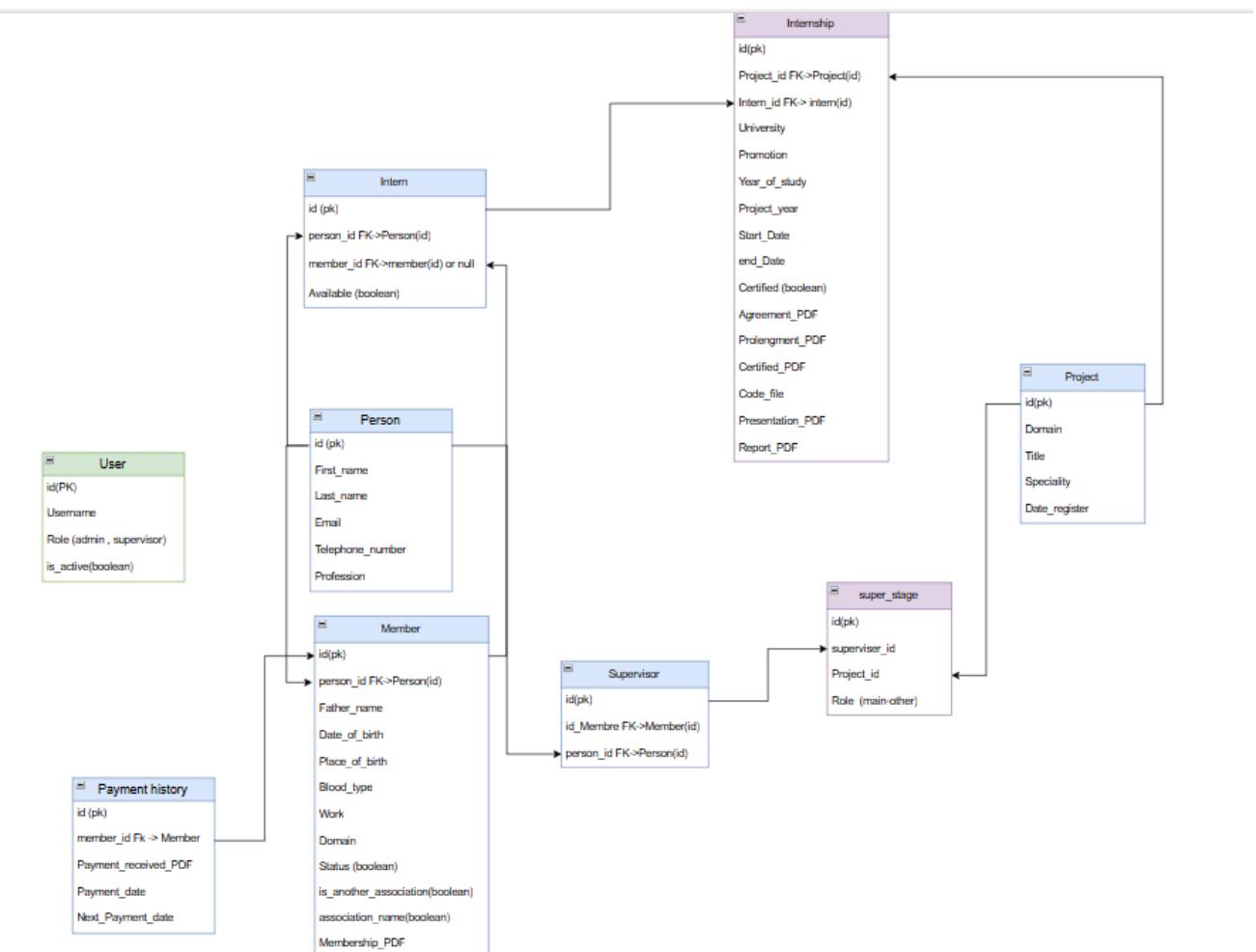
Add Project

Delete Project X

Are you sure you want to permanently delete this Project?

Delete Cancel

### 3.4 Structure de la Base de Données



## 1. Le modèle de base : Person

Le modèle Person représente une entité de base commune à plusieurs autres modèles (comme Member, Supervisor, Intern). Il contient des champs génériques tels que :

- first\_name, last\_name, email, phone\_number, etc.

Cela permet de centraliser les données personnelles communes et de les réutiliser par héritage.

## 2. Intern, Supervisor et Member : rôles dynamiques

Rôle	Peut aussi être...
Intern	Member Supervisor
Supervisor	Intern Member
Member	Intern, Supervisor

## 3. Member et l'historique des paiements (Payment History)

Chaque Membre peut avoir plusieurs paiements enregistrés. Cette relation est de type one-to-many via une clé étrangère

Cela permet de :

- Suivre les cotisations d'un membre.
- Générer des rapports d'adhésion.

## 4. Relation ManyToMany entre Intern et Project

Un stagiaire peut travailler sur plusieurs projets, et un projet peut impliquer plusieurs stagiaires

## 5. Relation ManyToMany entre Supervisor et Project

De la même manière, un projet peut être supervisé par plusieurs personnes, et un superviseur peut suivre plusieurs projets

Cela permet de :

- Désigner un superviseur principal et des superviseurs secondaires.
- Gérer dynamiquement la composition de l'encadrement de chaque projet.

Explication technique : Héritage multiple en Django

### 1. Qu'est-ce que l'héritage multiple ?

En Python , l'héritage multiple permet à une classe d' hériter de plusieurs classes parentes. Cela permet de réutiliser du code et d'assembler des comportements depuis plusieurs sources.

### 2. Comment Django gère-t-il l'héritage multiple ?

#### b. Multi-table inheritance

Chaque classe hérite de models.Model, crée sa propre table, et Django crée automatiquement des clés étrangères internes entre elles.

```
class Person(models.Model):
    first_name = models.CharField(max_length=100)
    last_name = models.CharField(max_length=100)...
class Member(Person):
    Father_name = models.CharField(max_length=100)...
```

## 3.5. Structure Du projet

### A-Page : Authentification

#### 1. Capture d'écran



The screenshot shows a 'Log In' form with the following fields:

- User Name: lyn
- Password: lyn12345
- Remember me:
- Forgot Password?
- Log In button

The background features the CHEHIM logo on the left.



The screenshot shows a 'Log In' form with the following fields:

- User Name: lyn
- Password: lyn1234
- Remember me:
- Forgot Password?
- Log In button

A red error message at the top states: 'Login failed. Please check your Password Or Username.' The background features the CHEHIM logo on the left.

#### 2. Description fonctionnelle

Cette page permet à l'utilisateur de se connecter au système à l'aide de son **nom d'utilisateur et mot de passe**.

L'accès au reste du système dépend du **type d'utilisateur** :

- As\_admin → accès complet (gestion des membres, stagiaires, projets, etc.)
- As\_member → accès limité selon les autorisations

Le système gère également une option Remember Me et une déconnexion automatique après 2 heures d'inactivité (sécurité).

Si l'option Remember Me est décochée → le token est conservé uniquement pour 5 minutes.

Si cochée → le token refresh permet de maintenir la session ouverte jusqu'à 2 heures.

#### 3. Frontend

Formulaire avec deux champs : **username** et **password**, + case à cocher "Remember Me".

Envoi des données via **Axios** à l'API JWT :

```
axios.post("/api/token/", { username, password })
```

Récupération des **tokens JWT** (access + refresh) et stockage dans le **localStorage**.

```
axios.get("/api/get_me/", {headers:{Authorization:bearer ${acces}}})
```

Récupération du type de l'utilisateur (admin ou membre) est également stocké dans le localStorage pour gérer le **routage conditionnel** selon le rôle.

#### 4. Backend

Authentification avec JWT :

**URL d'obtention du token :**

```
path('api/token/', TokenObtainPairView.as_view(), name='token_obtain_pair')
```

- Méthode : POST
- Requête : { "username": "user", "password": "pass" }

## B-Barre de Navigation (Navbar)

La **navbar** représente un élément central de l'interface utilisateur. Elle permet une **navigation intuitive et dynamique** entre les différentes pages du système.

Comportement et apparence

- Lors du **chargement initial**, la **page d'accueil (Home)** est **active par défaut**, avec un style distinctif (couleur bleue) pour indiquer qu'elle est sélectionnée.
- À chaque navigation vers une autre page, la navbar **met à jour dynamiquement la section active** selon l'URL, en appliquant un style bleu uniquement sur la page courante.

Informations utilisateur

La navbar affiche **les informations de l'utilisateur connecté**, obtenues via l'API :

[/api/get\\_me/](#)

- Ces données permettent d'afficher dynamiquement le nom, le rôle ou d'autres détails utiles sur le profil de l'utilisateur.

Déconnexion

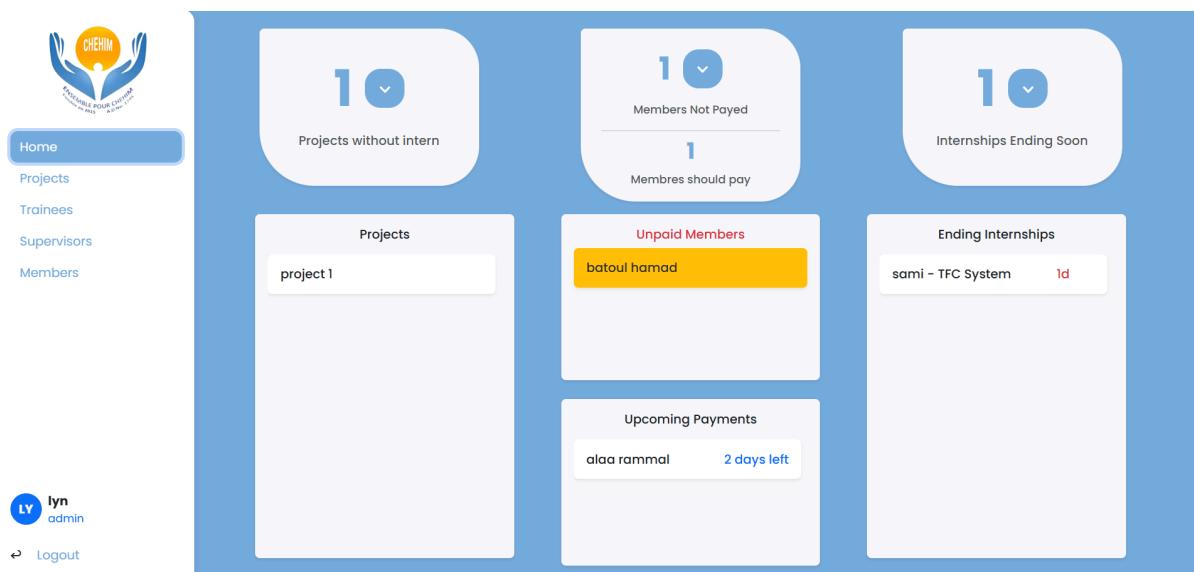
- Un **bouton "Logout"** est présent dans la barre de navigation.
- Lors du clic, ce bouton :
  - Supprime le **token d'accès** et le **token de rafraîchissement** du **localStorage**.
  - Supprime également le type d'utilisateur (**type\_of\_user**) utilisé pour la gestion des routes.
  - Redirige immédiatement l'utilisateur vers la page de connexion.

## C. Page : Accueil (Home)

Résumé du contenu

La page d'accueil constitue un tableau de bord dynamique qui résume l'état général du système de gestion **TFC Management System**. Elle est conçue pour afficher rapidement les informations essentielles, avec une interface fluide et organisée, notamment grâce à l'utilisation de **cards** et d'une **scroll bar interne** pour éviter le défilement vertical de toute la page.

1. Capture d'écran



## 2-Contenu principal de la page

### 1. Cards statistiques

En haut de la page, trois **cards statistiques** affichent les indicateurs suivants :

- **Nombre total de projets sans stagiaires assignés**
- **Nombre de membres ayant un paiement à venir (upcoming payment)**
- **Nombre de membres n'ayant pas encore payé**
- **Nombre des Stagiaires sans projet ou conventions expirant bientôt**

Chaque **card** inclut un bouton "**Voir plus**" permettant de naviguer vers une page dédiée avec plus de détails (→ voir captures dans les pages suivantes du rapport).

### 2. Liste des projets sans stagiaire

Une **card latérale** affiche la liste des projets encore **sans stagiaire assigné**.

- Chaque ligne contient le **nom du projet**

### 3. Liste des membres à surveiller

Deux sous-sections dans une card :

- **Membres avec paiements à venir**
- **Membres en retard de paiement**

### 4. Stagiaires sans projet ou conventions expirant bientôt

Cette section affiche :

- Les **stagiaires associés à un projet**
- Ceux dont la **convention de stage expire dans moins de 14 jours**

Chaque ligne contient des **liens vers les détails du stagiaire** pour permettre un suivi rapide.

## 3-Partie Frontend

La page d'accueil est conçue comme un tableau de bord interactif, utilisant ReactJS et Bootstrap.

Elle s'articule autour de composants modulaires, de liens dynamiques, et d'une gestion efficace des appels API.

Composant principal : StatCard

La page utilise un composant personnalisé appelé **StatCard**, qui reçoit plusieurs props pour afficher dynamiquement les blocs d'information.

**Props utilisées :**

- count : nombre ou statistique principale.
- message : message associé au count.
- href : lien pour le bouton "**Voir plus**".
- count2, message2 : valeurs supplémentaires facultatives.

Exemple de navigation par bouton :

- /admin-dashboard/Stage?is\_taken=false → affiche les stages sans stagiaire.
- /admin-dashboard/Member?A\_paye=false → liste des membres non payés.
- /admin-dashboard/Stagiaire?with\_condition=true → stagiaires avec convention expirant bientôt.

## Hyperliens dynamiques

Chaque nom (de projet, membre ou stagiaire) dans les listes est un **lien cliquable** menant à sa page de détail :

- /admin-dashboard/DetailsStage?stage=ID
- /admin-dashboard/DetailsMember/?member=ID
- /admin-dashboard/Detailsintern?id=ID

#### 4-Appels API côté frontend

La page effectue plusieurs appels API via Axios pour charger dynamiquement les données :

Fonction	API appelée	Description
fetchProjects()	/api/Stages/without_interns/	Récupère les projets sans stagiaires.
fetchUnpaidMembers()	/api/payment-history/unpayed/	Liste des membres non payés.
getUpcomingPayments()	/api/payment-history/upcoming/	Paiements à venir des membres.
getEndingInternships()	/api/stagestagiaire/	Stagiaires avec conventions expirant bientôt.(filtre niveau front)
updatePayedStatus()	/api/payment-history/id/ (PATCH)	Met à jour le statut de paiement des membres.

#### Sécurité et rafraîchissement des données

- La page d'accueil est **la première page accessible après l'authentification**.
- Un mécanisme de **rafraîchissement automatique** vérifie toutes les **2 heures** l'état des paiements via la fonction `updatePayedStatus()`.
- Si `today > nextPaymentDate` et que `record.payed != false`, une requête **PATCH** est envoyée pour mettre à jour le statut.

#### 5- Partie Backend : API pour la page d'accueil

##### 1. Liste des projets sans stagiaires /api/Stages/without\_interns/

Cette méthode est définie dans le `ProjectViewSet`. Elle utilise le décorateur `@action` pour créer une route personnalisée.

Description :

- Récupère tous les projets qui ne sont pas associés à un stagiaire.
- Utilise une requête ORM pour obtenir les `Project_id` déjà assignés dans le modèle `Internship` (association entre stage et stagiaire).
- Exclut ces projets de la requête.

Sérialise les projets restants pour les renvoyer dans la réponse.

##### 2. Liste des membres non payés /api/payment-history/unpayed/

Définie dans le `PaymentHistoryViewSet`, cette action permet d'extraire les membres dont le statut de paiement est encore à `False`.

Description :

- Utilise un simple filtre ORM sur la table des historiques de paiement (`Payment_history`) où `payed=False`.
- Cette API est utilisée pour surveiller la situation financière des membres.

### 3. Paiements à venir /api/payment-history/upcoming/

Toujours dans PaymentHistoryViewSet, cette méthode permet d'identifier les membres qui devront payer bientôt.

Description :

- Calcule la date actuelle (today) et une date 10 jours plus tard (ten\_days\_later) via la bibliothèque datetime.
- Filtre les enregistrements dont la date de prochain paiement (next\_payment\_date) tombe entre aujourd'hui et les 10 prochains jours.
- Cette logique aide à anticiper les paiements à venir pour prévenir les retards.

6-Presentation Resultat du navigation du button voir plus

- **Page Projects du page Home avec /admin-dashboard/Stage?is\_taken=false**

Avec filtre

Domain	Specialty	Title	Date Register	Project Taken	Main Supervisor	Actions
AI	informatique	project I	2025-05-15	<input checked="" type="checkbox"/>	olaar rammal	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

sans filtre

Domain	Specialty	Title	Date Register	Project Taken	Main Supervisor	Actions
web	full stock	TFC System	2025-05-15	<input checked="" type="checkbox"/>	olaar rammal	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
AI	informatique	project I	2025-05-15	<input checked="" type="checkbox"/>	olaar rammal	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

- **Page Membres du page Home avec /admin-dashboard/Member?A\_paye=false avec filtre**

ID	First Name	Last Name	Email	Phone	Address	Has Payed
3	batoul	harrod	batoul@gmail.com	78 625 167	doraya	<input checked="" type="checkbox"/>

sans filtre

ID	First Name	Last Name	Email	Phone	Address	Has Payed
3	batoul	harrod	batoul@gmail.com	78 625 167	doraya	<input checked="" type="checkbox"/>
2	ola	rammal	ola@nextmail.com	00 000 000	nabatiyah	<input type="checkbox"/>

- **Page Trainees du page Home avec /admin-dashboard/Stagiaire?with\_condition=true avec filtre**

ID	Name	Promotion	Annee	Current Internship	Start Date	End Date	Convention PDF	Certified	Certified PDF	Actions
1	sami	L3	2025	TFC System	2025-05-15	2025-06-16	diaf_.pdf	<input checked="" type="checkbox"/>	Not certified	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

sans filtre

ID	Name	Promotion	Annee	Current Internship	Start Date	End Date	Convention PDF	Certified	Certified PDF	Actions
2	jamil	L3	2022	TFC System	2025-05-15	2025-07-18	diaf_.pdf	<input checked="" type="checkbox"/>	Not certified	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
1	sami	L3	2025	TFC System	2025-05-15	2025-06-16	diaf_.pdf	<input checked="" type="checkbox"/>	Not certified	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

## E-Page Projects

### 1. Capture d'écran

The screenshot shows the 'Projects' section of the E-Page Projects application. At the top, there is a search bar with fields for 'Supervisor First Name', 'Supervisor Last Name', 'Domain', 'Title', 'Speciality', 'Project is Taken' (with radio buttons for checked, unchecked, and pending), and a 'Date Register'. Below the search bar is a yellow button labeled '+ ADD New'. The main area displays a table of registered projects with columns: Domain, Speciality, Title, Date Register, Project Taken, Main Supervisor, and Actions. Two projects are listed: 'TFC System' (Domain: web, Speciality: full stack) and 'project 1' (Domain: Ai, Speciality: informatique). Both projects are marked as taken by 'alaa rammal'. At the bottom, it says 'Showing 1 to 2 of 2 entries' and has navigation arrows. On the left sidebar, there are links for Home, Projects (which is selected), Trainees, Supervisors, and Members. On the right sidebar, there is a user profile for 'lyn admin' and a 'Logout' link.

### 2. Contenu de la page

La page affiche un **tableau d'informations** sur les projets enregistrés dans le système. Voici les colonnes du tableau :

Colonne	Description
Domain	Domaine du projet
Speciality	Spécialité du projet
Title	Titre du projet
Date Register	Date d'enregistrement
Project Taken	Si le projet est déjà assigné à un stagiaire
Main Supervisor	Superviseur principal du projet
Actions	Boutons pour <i>Modifier</i> , <i>Supprimer</i> et <i>Afficher les détails</i>

Le tableau est paginé, avec **5 projets par page**.

### 2. Système de filtre

Un système de **filtres dynamiques** permet de rechercher parmi les projets :

- Par **prénom** ou **nom de famille** du superviseur principal
- Par **domaine**, **titre** ou **spécialité** du projet
- Par **statut de prise du projet**
- Par **date d'enregistrement**

Après avoir saisi les critères, l'utilisateur peut cliquer sur **Search** pour appliquer les filtres.

### 3. Ajout de projet

Un bouton “**Add**” redirige vers un formulaire pour créer un nouveau projet avec superviseur principal.

#### 3-Partie Frontend

##### Chargement des projets

###### 1. Détection du filtre `is_taken=false` dans l’URL

Au moment du chargement de la page, le composant React récupère les **paramètres de l’URL** via `URLSearchParams`.

S’il détecte que `is_taken=false`, cela signifie qu’on veut afficher **uniquement les projets qui n’ont pas encore été pris par un stagiaire** mais qui ont **déjà un superviseur principal**.

Dans ce cas :

- Le composant exécute la fonction `fetchSupStagesFromHome()`.
- Cette fonction envoie une requête GET à l’API en ajoutant `project_taken=false` dans les paramètres.
- API `/api/supstage/?page=${currentPage}&Role=Admin&project_taken=${false}`

Sinon :

- Si aucun filtre spécifique n’est détecté, la fonction `fetchSupStages()` est appelée pour charger **tous les projets supervisés**.
- API `/api/supstage/?page=${currentPage}&Role=Admin`

**Avantage** : Cette logique permet d’utiliser la **même interface** pour plusieurs cas d’usage, en adaptant dynamiquement les données affichées selon les paramètres de l’URL.

###### 2. Gestion des filtres dynamiques

Le formulaire de filtrage comprend plusieurs champs : prénom/nom du superviseur, domaine, titre, spécialité, statut du projet, date.

Lorsqu’un utilisateur clique sur **Search** :

- Une fonction crée dynamiquement une URL contenant **uniquement les champs remplis**.
- Exemple : `...?supervisor_first_name=Ali&project_domain=IA`
- Une requête est alors envoyée à l’API avec cette URL construite.

Cela permet :

- Une recherche **précise et personnalisée**.
- D’éviter les requêtes inutiles avec des champs vides.

**Avantage**: L’API ne reçoit que les paramètres nécessaires, et la réponse est optimisée.

###### 3. Pagination

Pour ne pas surcharger la page, l'affichage est **paginé** à raison de **5 projets par page**.

Voici comment c'est géré côté React :

- Le système récupère la valeur `totalCount` renournée par l’API.
- Il calcule le **nombre total de pages** en divisant ce total par 5.

Lorsque l’utilisateur clique sur **page suivante / précédente** :

- Le `currentPage` est mis à jour.
- Une nouvelle requête est envoyée avec le numéro de page inclus dans l’URL (`?page=2` par exemple).
- Le tableau est mis à jour automatiquement avec les nouvelles données.

**Avantage**: Une navigation fluide, contrôlée par l’état React, avec des appels API propres et ciblés.

#### 4-Partie Backend :

##### 1. Concept des filtres dynamiques avec Django Filter

Dans le backend, les filtres sont gérés grâce à **django\_filters**, un module puissant qui permet de créer des filtres dynamiques sur les modèles Django.

- Un filtre spécifique a été créé pour le modèle **supervisor\_internship**, qui relie un projet à un superviseur.
- Ce filtre permet à l'API d'accepter des **paramètres dans l'URL** pour filtrer les résultats selon différents critères.

Les champs filtrables :

- **ID du projet et ID du superviseur** (filtrage exact).pour autre utilisation
- **Prénom / nom du superviseur** (filtrage partiel avec icontains pour la recherche approximative).
- **Titre / domaine / spécialité du projet.**
- **Statut du projet (is\_taken).**
- **Date d'enregistrement du projet.**
- **Rôle du superviseur** (ex. : Admin).

**Objectif** : Cela permet d'interroger la base de données de manière flexible, directement depuis les paramètres d'une URL.

Exemple d'URL de requête :

/api/supstage/?supervisor\_first\_name=Ali&project\_domain=IA&project\_taken=false

##### 3. Pagination côté backend

La pagination est définie à l'aide d'une classe personnalisée **StandardResultsSetPagination**, qui utilise :

- **5 éléments par page** (page\_size = 5).
- Un paramètre optionnel dans l'URL (page\_size=) pour personnaliser cette taille.

Exemple :

- /api/supstage/?page=2 retournera la **deuxième page** des projets.
- Le système renvoie en plus dans la réponse :
  - **Le nombre total** d'éléments.
  - **Le nombre de pages**.
  - Et les liens vers la **page suivante ou précédente**.

**Avantage** : Cela rend l'API plus performante et adaptée aux grandes bases de données, tout en facilitant la navigation côté frontend.

#### 5-Capture d'écran

test du filtre project taken

pagination :Bouton Next enable

la bouton << indique la premier page et >> la dernière page, puisque pas des pages encors bouton next disable et partie previous enable

#### → Formulaire d'ajout d'un projet

##### 1. Contenu de la page

Ce formulaire permet à un administrateur d'ajouter un nouveau projet dans le système. Il est composé de plusieurs parties essentielles :

##### 1-Champs du projet à remplir :

- **Domaine** : champ avec **validation** (lettres uniquement). Il est lié à une datalist, qui affiche tous les domaines existants (ex. : IA, Réseaux...). L'utilisateur peut :
  - Soit **choisir dans la liste**.
  - Soit **taper manuellement** un nouveau domaine (toujours en lettres uniquement).
- **Spécialité** : même validation, uniquement des lettres.
- **Titre du projet** : champ texte libre.
- **Fichier PDF du projet** : fichier obligatoire.  
**Validation** : seuls les fichiers **.pdf** sont acceptés.  
En cas d'extension incorrecte, une **alerte** s'affiche.

##### 2- Ajout du superviseur principal – Obligatoire

Règle :

- **Le projet doit obligatoirement avoir un superviseur principal.**
  - Si l'utilisateur clique sur "Finish Submission" sans sélectionner un superviseur principal, une **popup React**
- Sélection dynamique :
- Le menu déroulant affiche :
    - Tous les **membres existants**.
    - Tous les **membres déjà superviseurs**.
  - Quand on sélectionne un **membre non-superviseur**, il devient automatiquement **superviseur** dans le backend.

##### 3. Ajout de superviseurs secondaires (optionnel)

- Les superviseurs secondaires peuvent être
  - Des **membres superviseurs existants**.
  - Les **superviseurs qui ne sont pas membre**

- Des **membres simples** (et ils seront promus superviseurs si sélectionnés).
- Le système applique la **même logique dynamique** : tout membre choisi devient superviseur automatiquement.

#### 4. Ajout d'un nouveau superviseur (bouton)

- Un bouton spécial permet d'**ouvrir un formulaire pour créer un nouveau superviseur**.
  - Ce formulaire peut être :
    - Soit pour un **membre** (avec case “est aussi superviseur”).
    - Soit pour un **superviseur non membre**.
  - Une fois ajouté, ce superviseur est **ajouté automatiquement dans la liste déroulante** pour pouvoir l'assigner directement au projet.

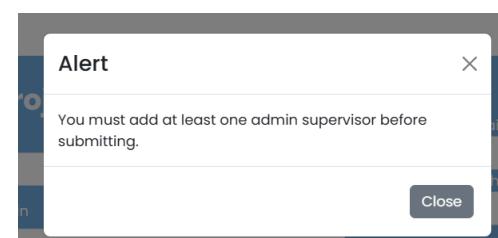
#### 5. Bouton final Finish Submission

Quand on clique sur ce bouton :

1. Le projet est d'abord créé (POST vers l'API des projets).
2. Ensuite :
  - Le **superviseur principal** est lié au projet via une autre API (supstage).
  - Les **superviseurs secondaires** sont aussi associés.

#### 2-Capture d'écran

validation avec messages d'erreurs



alert pour mettre un superviseur Admin

Name	X	Headers	Preview	Response	Initiator	Timing
get_me/	1			[		
members_as_supervisor/	-			{		
Stages/	-			"id": 6,		
members_not_supervisor/	-			"first_name": "fatima",		
Supervisors/?no_member=true	-			"last_name": "hamad",		
	-			"email": "fatima@gmail.com",		
	-			"phone_number": "79 625 167",		
	-			"profession": "no need",		
	-			"Father_name": "hasan",		
	-			"Date_of_birth": "2025-05-15",		
	-			"Place_of_birth": "hilaliya",		
	-			"Adresse": "nabatiyeh",		
	-			"Blood_type": "O+",		
	-			"Work": "ai",		
	-			"Domain": "ccne",		
	-			"is_another_association": false,		
	-			"association_name": "",		
	-			"Application_PDF": "http://localhost:8000/medi",		
	-			"is_supervisor": false,		
	-			"member_payed": false		
	-			}		
	-			]		

apres creation d'un instance superviseur

#### Superviser Instance

GET /api/Supervisors/6/

HTTP 200 OK

Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS  
Content-Type: application/json  
Vary: Accept

```
{
    "id": 6,
    "first_name": "fatima",
    "last_name": "hamad",
    "email": "fatima@gmail.com",
    "phone_number": "79 625 167",
    "profession": "no need",
    "Id_Membre": 6
}
```

avant de select un membre

après l'ajout

### 3- Partie Frontend

#### 1. Validation des données utilisateur (validate)

Avant l'envoi du formulaire, une fonction de validation est utilisée pour vérifier les données saisies. Cette validation repose principalement sur des **expressions régulières (regex)** pour s'assurer que les champs respectent un format attendu (ex. : email, domaine, etc.). Cela permet :

- d'éviter l'envoi de données incomplètes ou mal formatées ;
- d'améliorer l'expérience utilisateur avec des messages d'erreur en temps réel ;
- de renforcer la sécurité du système en limitant les erreurs côté serveur.

#### 2. Soumission du formulaire principal (handleSubmit)

Une fois les données validées, elles sont envoyées à l'API de création des stages :

`/api/Stages/`

Cette soumission déclenche l'enregistrement du stage dans la base de données. Le retour de cette API fournit l'**ID du stage nouvellement créé**, qui sera utilisé ensuite pour lier les superviseurs à ce stage.

#### 3. Gestion dynamique des superviseurs (handleChangeMulti + ensureMemberIsSupervisor)

Lorsqu'un utilisateur sélectionne un membre qui **n'est pas encore superviseur**, une fonction est déclenchée pour :

- vérifier son statut ;
  - si nécessaire, envoyer une requête à l'API :
- `/api/Supervisors/create_supervisor_from_member/`  
afin de le transformer en superviseur.

Cette automatisation évite à l'utilisateur de devoir créer manuellement un superviseur avant de le sélectionner.

#### 4. Mise à jour dynamique des listes déroulantes (updateDropdownOptions)

Lorsque l'utilisateur sélectionne un superviseur principal ou secondaire, les listes déroulantes sont mises à jour automatiquement pour :

- retirer le superviseur principal de la liste des superviseurs secondaires ;
- retirer les superviseurs secondaires de la liste du superviseur principal.

Cela permet d'éviter les doublons dans la sélection et assure l'unicité du rôle principal.

## 5. Chargement des superviseurs disponibles (fillSupervisors)

Trois appels API permettent de charger toutes les catégories de superviseurs et membres :

- /api/Membres/members\_as\_supervisor/ → membres déjà superviseurs.
- /api/Membres/members\_not\_supervisor/ → membres pas encore superviseurs.
- /api/Supervisors/?no\_member=true → superviseurs extérieurs, non associés à un membre.

Ces données sont ensuite structurées avec des attributs spécifiques (ismember, onlymember, etc.) pour permettre une distinction claire dans les listes.

### Préparation des options des dropdowns

Deux listes sont générées :

- mainDropdownOptions pour la **sélection unique** du superviseur principal ;
- multiDropdownOptions pour la **sélection multiple** des superviseurs secondaires.

Ces listes sont composées des données récupérées depuis les APIs, en combinant :

- les membres superviseurs ;
- les membres non superviseurs ;
- les superviseurs externes.

Des drapeaux (ismember, onlymember, etc.) sont utilisés pour ajuster l'affichage ou la logique conditionnelle côté frontend.

## 7. Soumission des superviseurs (submitSupervisors)

Une fois le stage enregistré, une fonction de soumission secondaire est appelée pour **lier les superviseurs au stage** via l'API : /api/supstage/.

Cette fonction transmet les superviseurs sélectionnés (principal et secondaires) en les associant à l'ID du stage retourné précédemment.

## 8. Extraction des domaines existants (fetchDomains)

Pour proposer des suggestions dynamiques ou pour éviter les doublons, une fonction interroge l'API /api/Stages/ pour extraire tous les **domaines existants** dans les stages déjà créés. Ensuite :

- les doublons sont éliminés ;
- une liste unique de domaines est générée ;
- cette liste est affichée comme suggestions dans un champ ou un menu.

Cela améliore l'ergonomie du formulaire et garantit la cohérence des données.

4-Appels API côté frontend

Nom de la Fonction	API Appelée
handleSubmit	POST /api/Stages/
submitSupervisors	POST /api/supstage/
ensureMemberIsSupervisor	POST /api/Supervisors/create_supervisor_from_member/
fillSupervisors	- GET /api/Membres/members_as_supervisor/ - GET /api/Membres/members_not_supervisor/ - GET /api/Supervisors/?no_member=true

## 5-Partie Backend

### 1. Création d'un stage

- **API** : POST /api/Stages/
- **ViewSet** : ProjectViewSet
- **Fonction** : create() (*déjà intégrée par Django via ModelViewSet*)

### 2. Association des superviseurs à un stage

- **API** : POST /api/supstage/
- **ViewSet** : supervisor\_internshipViewSet
- **Fonction** : create() (*déjà intégrée par Django via ModelViewSet*)

### 3. Crée un superviseur à partir d'un membre

- **API** : POST /api/Supervisors/create\_supervisor\_from\_member/
- **ViewSet** : SupervisorViewSet
- **Concept** : route personnalisée (@action) permet de convertir automatiquement un **membre en superviseur**.

Lorsqu'un utilisateur sélectionne un membre qui n'est pas encore superviseur, cette API est appelée avec l'ID du membre.

Elle crée un objet Supervisor lié à ce Member.

Cela évite d'obliger les utilisateurs à passer manuellement par un formulaire séparé.

### 4. Récupérer les membres superviseurs

- **API** : GET /api/Membres/members\_as\_supervisor/
- **ViewSet** : MemberViewSet
- **Concept** : Cette action retourne la liste de tous les membres qui sont **déjà liés à un superviseur**. Elle est utilisée pour alimenter la liste déroulante du frontend qui affiche uniquement les membres superviseurs.

### 5. Récupérer les membres non superviseurs

- **API** : GET /api/Membres/members\_not\_supervisor/
- **ViewSet** : MemberViewSet
- **Concept** : Cette action retourne la liste des membres qui **ne sont pas encore superviseurs**. Elle est utilisée dans le formulaire pour permettre à l'utilisateur de sélectionner un membre à convertir en superviseur si nécessaire.

### 6. Lister les superviseurs externes (non membres)

- **API** : GET /api/Supervisors/?no\_member=true
- **ViewSet** : SupervisorViewSet avec un filterset\_class personnalisé
- **Concept** : Cette API retourne uniquement les superviseurs qui **ne sont pas liés à un membre**

Grâce au filtre no\_member=true, on peut facilement distinguer les superviseurs externes des superviseurs internes dans le frontend

→ Fonctionnalité "Ajouter un nouveau superviseur"

## 1-Partie Frontend

### Composant Intégré : AddSupervisorFromAddProject

Ce composant est affiché dans une **modale Bootstrap** déclenchée par un bouton. Il permet l'ajout d'un nouveau superviseur, qu'il soit **membre** ou **non membre**. Le formulaire est dynamique, avec deux options via **radio buttons** :

- **Membre** : un formulaire permet de saisir les informations d'un membre interne (enseignant, encadrant, etc.).
- **Non membre** : un formulaire distinct est présenté pour les superviseurs externes.

## Logique de Soumission (handleSubmit)

### Cas 1 : Superviseur Non Membre

- **API utilisée :** POST /api/Supervisors/
- Après validation du formulaire, le superviseur est directement créé en base.
- Il est **automatiquement placé dans la liste des superviseurs secondaires (others)**.
- Ce placement est automatique car un non-membre ne peut être principal par défaut.

### Cas 2 : Superviseur Membre ET Superviseur Principal déjà sélectionné

- **API utilisée :**
  - POST /api/Membres/ (si création du membre)
  - puis POST /api/Supervisors/create\_supervisor\_from\_member/ (conversion en superviseur)
- Le superviseur est créé et placé automatiquement dans la liste **des superviseurs secondaires**, car la place principale est déjà occupée.

### Cas 3 : Superviseur Membre ET Aucun superviseur principal sélectionné

- Dans ce cas, un **popup de confirmation** est déclenché pour demander à l'utilisateur :
  - **Souhaitez-vous le placer comme superviseur principal ?**
  - Ou bien l'ajouter aux superviseurs secondaires ?

## Validation des Champs

Chaque formulaire (membre ou non membre) dispose de sa propre fonction de validation :

- Vérification des formats (email, téléphone, etc.)
- Champs obligatoires selon le contexte
- Alerte si des données manquent ou sont incorrectes avant l'envoi

## Mise à jour de l'interface après soumission

Une fois le superviseur créé :

- La modale est fermée automatiquement si l'opération est réussie.
- Le superviseur est **auto-sélectionné** dans le menu déroulant approprié via la fonction onSupervisorAdded :
  - Cela déclenche une mise à jour des champs visibles dans le formulaire principal (sélection dynamique dans la dropdown).
  - Le système différencie les superviseurs membres et non membres via un **flag isMember**.

## 2-Appels API côté frontend

Statut	Étapes d'appel API	Objectif
Non membre	POST /api/Supervisors/	Création directe et placement en secondaire
Membre	POST /api/Membres/ → POST /api/Supervisors/create_supervisor_from_member/	Création du membre, puis transformation en superviseur

### 3-Partie Backend

API /api/Supervisors/create\_supervisor\_from\_member/

Localisation dans le Backend

- **ViewSet concerné** : SupervisorViewSet
- **Type de méthode** : Action personnalisée (custom action)
- **Décorateur Django REST** : @action(detail=False, methods=['post'])

Objectif de l'API

Cette API a pour but de créer un superviseur à partir d'un membre existant.

4-Capture d'écran

addition avec validation

Auto select dans other

addition superviseur comme membre

choix de placement  
apres choisir main  
apres submit

## F- PAGE : Modification d'un Projet

### 1. Capture d'écran

The screenshot shows two overlapping web pages. The top page is titled 'Modify Project' and contains fields for 'Title' (project 7), 'Domain' (web), 'Speciality' (fullstack), 'PDF of Project' (choose file: No\_en, llyrhammal\_qsbr), 'Project is taken' (checkbox checked), 'Date\_register' (05/16/2025), and a 'Save' button. The bottom page is titled 'Update Project Supervisors' and lists 'Main Supervisor' (walid Rammal) and 'Other Supervisors' (dana rammal). It has a 'Save' button and three buttons for managing interns: 'Add New Intern', 'Modify existing Intern', and 'Delete existing Intern'. On the left side of the main page, there is a sidebar with links: Home, Projects, Trainees, Supervisors, Members, and a user profile for 'lyn admin'.

Permettre à un utilisateur (admin ou encadrant) de modifier :

- Les **informations principales du projet**  
Le ou les **superviseurs**

### La liste des stagiaires affectés au projet

2-Partie Frontend

1. Récupération des informations du projet sélectionné

Fonction : `fillProjectData()`

Envoie une requête GET à l'API `/api/Stages/?id__icontains=${stageid}`

Cette API retourne les **détails du projet** :

Ces données sont **pré-remplies dans le formulaire**.

2. Récupération des superviseurs existants

Fonction : `fetchData()`

Permet de :

Récupérer tous les membres qui peuvent être superviseurs depuis `/api/Supervisors/`

Distinguer les superviseurs par :

`id_member`: superviseur lié à un membre

les autres : superviseurs externes

API appelée :

`get api/supstage/?project_id=${stageid}`;`

Résultat : une liste des superviseurs affectés à ce projet

Classification :

- main : celui avec Role: "Admin"
- others : ceux avec Role: "Other"

3. Soumission des modifications

Fonction : handleSubmit(e)

Quand l'utilisateur clique sur "Modifier le projet", les actions suivantes se produisent :

a. Mise à jour du projet lui-même :

Envoie une requête PATCH : /api/Stages/\${formData.id}/

Met à jour les champs modifiés du projet.

Fonction : Submit(e)

b. Superviseur principal :

Si un superviseur principal est sélectionné :

Si une relation existe déjà (adminEntryId), on fait un PUT

Sinon, un POST est envoyé pour la créer

c. Superviseurs secondaires :

Pour chaque superviseur sélectionné :

    Si nouveau → POST (création)

Pour les superviseurs supprimés → DELETE pour chaque id concerné

#### 4. Actions sur les stagiaires

Interface utilisateur :

Chaque bouton (**Ajouter**, **Modifier**, **Supprimer un stagiaire**) ouvre un **modal** (fenêtre contextuelle avec formulaire).

Si sujet\_pris est **false** :

    Les boutons **Modifier** et **Supprimer** sont désactivés

Dès qu'un stagiaire est ajouté :

    La valeur sujet\_pris est mise à **true**

    Les boutons **Modifier** et **Supprimer** deviennent **actifs**

Capture d'écran

The screenshot shows the application's navigation bar on the left with links for Home, Projects, Trainees, Supervisors, and Members. A user profile icon for 'lyn admin' is also present. Two modals are displayed over the main content area:

- Modify Project** modal:
  - Title: project 1
  - Domain: AI
  - Specialty: informatique
  - PDF of Project: Choose File No... (with file name 'lynrammal\_11rlvW')
  - Project is taken:
  - Date\_register: 05/15/2025
  - Buttons: Save, Cancel, Modify Project
- Update Project Supervisors** modal:
  - Main Supervisor: alaa rammal
  - Other Supervisors: Select additional supervisors
  - Buttons: Save, Add New Intern, Modify existing Intern, Delete existing Intern

Formulaire avec les données pré-remplies

## Modification avec validation

→ FORMULAIRE : Ajout d'un nouveau stagiaire au projet

### Objectif :

Permettre à l'utilisateur d'ajouter un stagiaire :

- Qui n'a jamais fait de stage
- Ou qui a déjà fait un stage (et est certifié)
- Avec remplissage automatique si certaines données existent déjà
- Avec validation logique des champs
- Et deux options d'ajout : **Ajouter un autre ou Terminer**

### Partie Frontend

#### 1. Récupération des stagiaires éligibles

Lors de l'ouverture du formulaire select intern contient :

- Tous les stagiaires **jamais assignés à un stage**
- Ou ceux qui ont **déjà eu un stage et l'ont terminé avec un certificat**

#### 2. Remplissage automatique si le stagiaire a déjà un historique

Une fois un stagiaire sélectionné : On vérifie s'il existe un lien dans stage\_stagiaire :

get /api/stagestagiaire/?stagiaire\_id=\${selectedInternId}

- Si l'intern a un stage précédent :
  - Les champs comme **année, université, promotion** peuvent être remplis automatiquement
  - Mais les champs **start date, end date**, et **fichier de convention** restent vides et obligatoires

#### 3. Calcul automatique de l'année universitaire

calcul se base sur le mois courant :

- Juillet (mois 7) ou plus → année actuelle + 1
- Sinon → année précédente + actuelle

#### 4. Validation du formulaire

Avant soumission :

- Tous les champs obligatoires doivent être remplis  
**La date de début** doit être **inférieure** à la **date de fin**
  - Si ce n'est pas le cas → une erreur est affichée

#### 5. Soumission du formulaire

Le formulaire propose **deux boutons** :

Bouton "Add More"

- Valide le formulaire
- Fait un POST vers /api/stagestagiaire/ pour créer le lien entre stagiaire et projet
- Réinitialise le formulaire (sans fermer le modal)

Bouton "Finish"

- Ferme ensuite le modal

La bouton Add intern ouvre un Modal pour ajouter un nouveau intern et auto sélectionné après l'ajout

### 3-Partie backend

#### 1. Vérification de la non-redondance

Avant d'ajouter un stagiaire à un projet, une requête est envoyée à l'endpoint /api/stagestagiaire/ avec le paramètre Project\_id. Cela permet de récupérer tous les stagiaires déjà associés à ce projet et ainsi d'éviter d'enregistrer deux fois le même stagiaire pour un même projet.

#### 2. Filtrage des stagiaires disponibles

Pour garantir que seuls les stagiaires non affectés à un projet soient affichés à l'utilisateur, une requête est envoyée à l'endpoint /api/Stagiaires/ avec le filtre available=true. Ce champ (booléen ou calculé) permet d'exclure les stagiaires déjà affectés à un stage en cours.

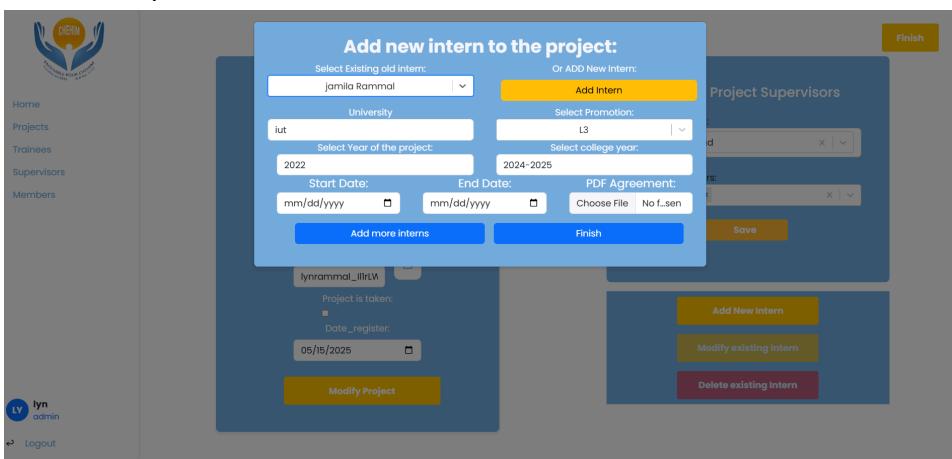
#### 3. Mise à jour de l'état du projet

Lorsque l'ajout d'un stagiaire à un projet est confirmé, une requête PATCH est envoyée à l'endpoint /api/Stages/<id>/ afin de mettre à jour le champ is\_taken à true.

#### 4. Crédit de la relation stage-stagiaire

Enfin, une requête POST est effectuée vers l'endpoint /api/stagestagiaire/ pour créer la liaison entre le stagiaire et le projet

Test avec captures d'écran



sélection d'un stagiaire et recuperation des données

## Add new intern to the project:

Select Existing old intern: jamila Rammal

Or ADD New Intern: Add Intern

University: iut

Select Promotion: L3

Select Year of the project: 2022

Select college year: 2024-2025

Start Date: 05/16/2025 End Date: 05/15/2025 PDF Agreement: Choose File lynn...l.pdf

**End date must be after start date.**

Add more interns Finish

validation sur le date

### Add new Intern

Last Name: rama

First Name: rammal

Email: rama@nextmail.com

Phone number: 00 000 000

ex: 03 123 456

Profession: ccne

Add Intern Cancel

## Add new intern to the project:

Select Existing old intern: rama rammal

Or ADD New Intern: Add Intern

University: Select...

Select Promotion: Select...

Select Year of the project: Enter academic year

Select college year: Enter academic year

Start Date: mm/dd/yyyy End Date: mm/dd/yyyy PDF Agreement: Choose File No f.sen

Add more interns Finish

addition d'un nouveau intern

auto selecte

Home  
Projects  
Trainees  
Supervisors  
Members

lyn admin  
Logout

### Modify Project

Title: project 1

Domain: Ai

Speciality: medical with ai

PDF of Project: Choose File No.en lynnrammal\_llrltW

Project is taken:

Date\_register: 05/15/2025

Modify Project

### Update Project Supervisors

Main Supervisor: batoul hamad

Other Supervisors: dana rammal

Save

apres l'addition

→ FORMULAIRE Modification des informations d'un stagiaire affecté à un projet :

The screenshot shows a modal window titled "Modify Stage Intern Info". The form contains the following fields:

- Intern:** rama rammal
- University:** iut
- Promotion:** L3
- Year of Study:** 2024-2025
- Project Year:** 2023
- Certified:**
- Agreement (PDF):** Choose File | No file chosen  
lynrammal\_VxitSwy....
- Certificate (PDF):** Choose File | No file chosen
- Report (PDF):** Choose File | No file chosen
- Presentation (PDF):** Choose File | No file chosen
- Code File (.zip or .rar):** Choose File | No file chosen

At the bottom of the form are two buttons: "Modify" (highlighted in yellow) and "Finish".

### 1. Récupération des stagiaires déjà affectés au projet

Lors du chargement du formulaire de modification, une requête est envoyée au backend via l'endpoint `/api/stagestagiaire/?Project_id=<id>`, ce qui permet d'obtenir la liste des stagiaires déjà associés au stage correspondant. Cela garantit que seules les données pertinentes sont affichées pour modification.

### 2. Pré-remplissage du formulaire

Une fois les stagiaires récupérés, les données du stagiaire sélectionné sont automatiquement insérées dans le formulaire. Cela permet à l'utilisateur de voir les informations actuelles (nom, université, promotion, fichiers joints, etc.) et de les modifier si nécessaire.

### 3. Validation des données

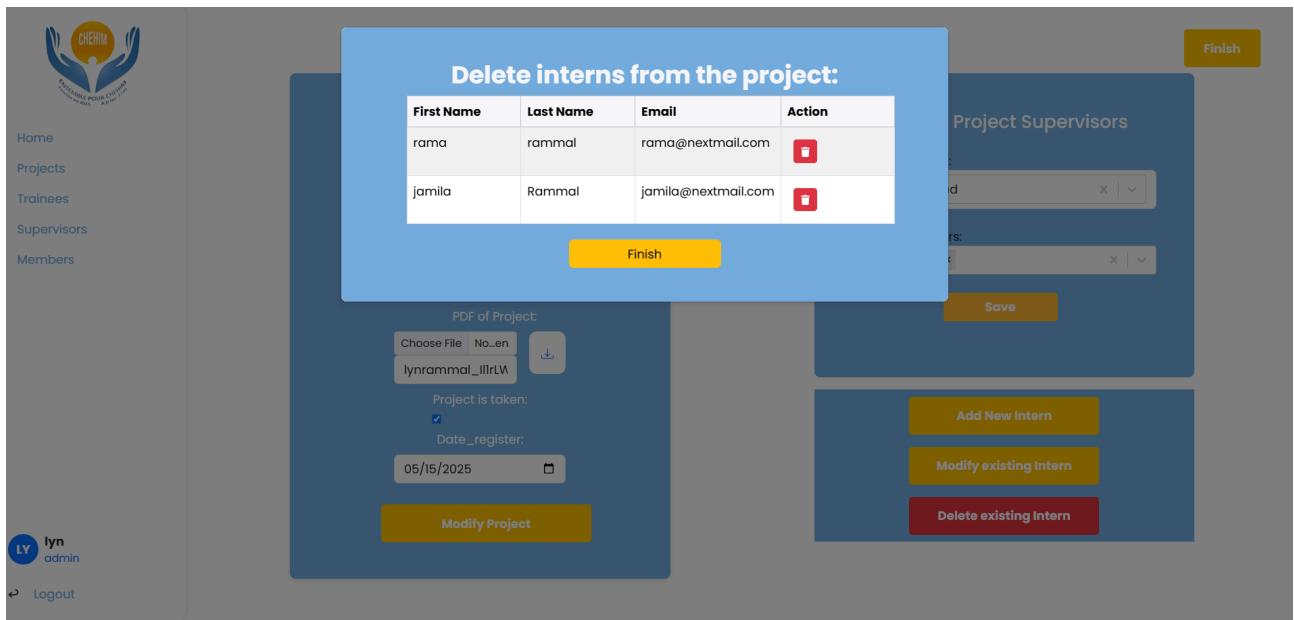
Avant toute soumission, des validations sont appliquées sur le formulaire (obligation de remplir certains champs, vérification du type de fichier, format des dates, etc.) afin d'assurer la cohérence et la fiabilité des informations saisies.

### 4. Actions disponibles

Deux boutons sont proposés à l'utilisateur :

- **Modify** : Permet d'enregistrer les modifications sans quitter la page. Une requête PATCH ou PUT est envoyée pour mettre à jour les données existantes dans la base.
- **Finish** : Enregistre les données et réinitialise le formulaire. Cela permet à l'utilisateur d'enchaîner facilement avec une autre opération sans interférence.

## → FORMULAIRE Suppression d'un stagiaire affecté à un projet

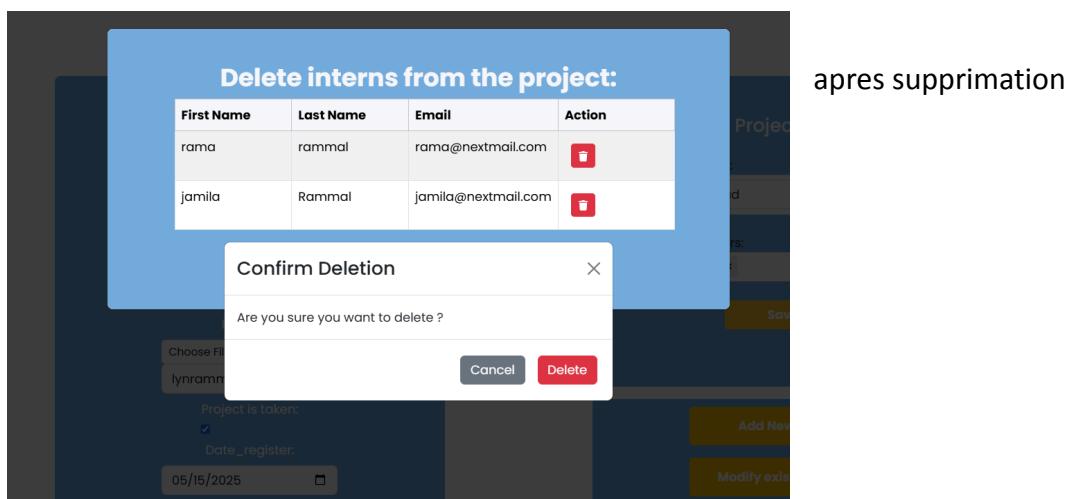


### 1. Affichage des stagiaires affectés

Lorsqu'un utilisateur souhaite supprimer un stagiaire d'un projet, un **popup (fenêtre modale)** s'ouvre. Ce popup contient un **tableau récapitulatif** affichant la liste des stagiaires actuellement affectés au projet. Chaque ligne contient les informations suivantes :Nom et prénom du stagiaire et email

### 2. Sélection et confirmation

À côté de chaque stagiaire dans le tableau, un **bouton de suppression** est affiché. Lorsqu'un utilisateur clique dessus, un **message de confirmation** apparaît pour éviter toute suppression accidentelle. L'utilisateur doit confirmer explicitement son choix.



si tous les stagiaire son suprime , le projet renvoi a project\_taken = false

Appels API côté frontend

Méthode	Endpoint	Détail
GET	/api/stagestagiaire/?stage_id={stageId}	Récupère les stagiaires assignés à un stage spécifique
DELETE	/api/stagestagiaire/{id}	Supprime un stagiaire d'un projet
PATCH	/api/Stages/{stageId}	Met à jour le champ is_taken du projet (false s'il n'y a plus de stagiaire assigné)

## F-Page Détails du Projet

La page "Détails du Projet" affiche des informations détaillées sur un stage sélectionné, notamment :

- **Les détails du projet** : titre, domaine, spécialité, date d'enregistrement, état (pris ou non), lien vers le fichier PDF.
- **Les superviseurs associés au projet** : noms, rôles, avec un lien vers leurs profils.
- **Les stagiaires affectés** : noms, périodes de stage (dates de début et de fin), avec un lien vers leur profil individuel.

## Appels API côté frontend

Méthode	Endpoint	Détail
GET	/api/Stages/{id}/	Récupère les détails complets d'un projet/stage donné ( <b>id</b> est fourni via <code>searchParams</code> )
GET	/api/supstage/?supervisor_id={id}&project_id={id}	Récupère les informations détaillées d'un superviseur spécifique lié à un projet donné
GET	/api/stagestagiaire/?stage__id={id}	Récupère la liste des stagiaires affectés à un projet donné

The screenshot shows a web application interface for managing projects. On the left, there is a sidebar with a logo for 'CIEHM' and links to 'Home', 'Projects', 'Trainees', 'Supervisors', and 'Members'. At the bottom of the sidebar, there are 'lyn' and 'admin' status indicators, along with a 'Logout' link. The main content area is titled 'project 1'. It contains three main sections: 'Project Information' (Title: project 1, Domain: AI, Speciality: medical with ai, Taken: 2025-05-15, PDF: Open PDF), 'Supervisors' (batoul hamad, dana rammal), and 'Interns' (rama - rammal, Start: 2025-05-16, End: 2025-05-24).

page details

## G- Page Stagiaire

### 1-Contenu de la page

Cette page permet à l'administrateur de visualiser, rechercher, filtrer, ajouter ou supprimer les stagiaires associés aux stages, tout en affichant certaines informations clés (nom, promotion, année, certification, etc.). Elle inclut également une pagination, des filtres dynamiques et une option de suppression sécurisée via une modale de confirmation.

### 2-Partie frontend

#### 1. Filtres dynamiques (formulaire) :

Permet de filtrer les stagiaires selon :

**Prénom / Nom du stagiaire** **Promotion** **Année de projet** **Titre du stage** **Certification flag** oui ou non

Les champs sont liés à l'état searchValues, qui est appliqué aux filtres réels via un bouton Search.

#### 3. Boutons principaux :

- **Search** : applique les filtres saisis.
- **Add New** : redirige vers la page de création d'un nouveau stagiaire.

#### 4. Affichage des stagiaires (Tableau) :

- Un tableau affiche les stagiaires paginés avec leurs informations principales.
- Chaque ligne comporte des actions :
  - Modifier un stagiaire
  - Supprimer un stagiaire (confirmation via ConfirmModal)
  - Détails supplémentaires

#### 5. Pagination :

- Permet de naviguer entre les pages de stagiaires via des boutons :

#### 6. Logique conditionnelle (URL) :

- Si l'URL contient ?with\_condition=true, alors seuls les stages **bientôt terminés** (dans les 14 jours) sont affichés.
- Sinon, on affiche tous les stagiaires selon les filtres.

#### 7. Suppression sécurisée :

- Lors du clic sur , une fenêtre ConfirmModal s'affiche.
- Si confirmé, l'appel axios.delete est déclenché pour supprimer le stagiaire.

## 3-Appels API côté frontend

Méthode	Route	Description
GET	/api/stagestagiaire/	Récupère la liste paginée des stagiaires liés aux stages. Prend en charge les filtres et pagination
DELETE	/api/stagestagiaire/{id}/	Supprime un enregistrement stagiaire-stage par ID
GET	/api/stagestagiaire/{id}/	(utilisé ailleurs) Récupère les détails d'un stagiaire-stage spécifique

## 4-Capture d'écran

The screenshot shows a web-based application for managing interns. On the left, there's a sidebar with links: Home, Projects, Trainees (which is selected and highlighted in blue), Supervisors, and Members. The main content area has a blue header bar with search fields for Intern First Name, Intern Last Name, Promotion, and Project Year, along with a 'Search' button. Below this is a section titled 'Certified' with radio buttons (● ✓ ● ✗). A yellow button labeled '+ ADD New' is visible. The main table lists one intern entry:

Id	Name	Promotion	Année	Current Internship	Start Date	End Date	Convention		Certified PDF	Certified PDF	Actions	
							PDF	Certified				
5	rama rammal	L2	2022	project1	2025-05-16	2025-05-24	dia/...pdf		Not certified			

Below the table, it says 'Showing 1 to 1 of 1 entries'. At the bottom right of the main area are navigation icons: double arrows, a left arrow, a right arrow, and a double right arrow.

In the bottom-left corner of the main area, there's a small circular icon with 'LY' and the name 'lyn' next to it, followed by 'Logout'.

## → FORMULAIRE : Ajout d'un nouveau stagiaire au projet

### 1. Contenu de la page

La page de gestion des stagiaires est divisée en deux étapes principales :

- Étape 1 : Ajout d'un nouveau stagiaire via un formulaire de saisie de données personnelles.
- Étape 2 : Assignation d'un projet/stage au stagiaire, accompagnée de l'envoi de documents justificatifs.

### 2. Partie frontend

#### 2.1. Ajout d'un stagiaire ([AddStagierVersion1](#))

- Le formulaire contient des champs :

Prénom Nom Email Numéro de téléphone Profession

- La validation est faite côté client avec des expressions régulières
- En cas de validation réussie, les données sont envoyées à l'API pour création.

#### 2.2. Assignation de stage ([AddStageToInternForm](#))

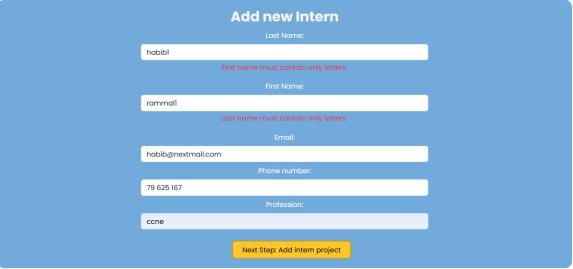
- Cette étape permet de :
  - Sélectionner un projet existant
  - Saisir l'université, promotion, année d'étude
  - Télécharger plusieurs fichiers (PDF de convention, certificat, rapport, etc.)
  - Saisir les dates de début et de fin de stage
- Une validation est également faite côté client (champ requis, format des dates, etc.)

### 3. Appels API côté frontend

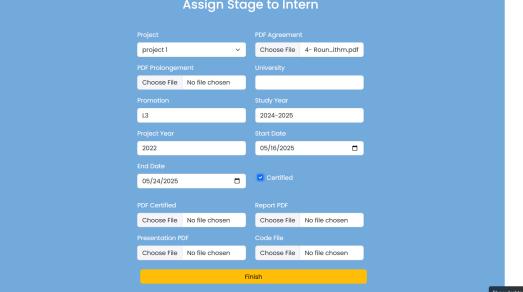
Étape	Méthode	URL	Description
Ajout d'un stagiaire	POST	http://localhost:8000/api/Stagiaires/	Crée un nouveau stagiaire
Récupération des stages	GET	http://localhost:8000/api/Stages/	Récupère la liste des projets disponibles
Assignation de stage	POST	http://localhost:8000/api/stagestagiaire/	Crée une assignation stage-stagiaire
Mise à jour du sujet	PATCH	http://localhost:8000/api/Stages/{id}/	Marque le projet comme pris

### 4. Captures d'écran

**Formulaire d'ajout de stagiaire avec validation**



**Formulaire d'assignation de stage**



#### → FORMULAIRE : Modification d'un stagiaire

##### 1. Contenu de la page

La page "Modifier un stagiaire" permet de mettre à jour les informations personnelles d'un stagiaire déjà existant dans le système.

Les champs disponibles pour la modification sont :

Prénom Nom Adresse email Numéro de téléphone Profession

## 2. partie frontend

La page est développée avec React et utilise plusieurs hooks :

- useEffect : pour récupérer les données actuelles du stagiaire depuis l'API dès le chargement de la page.
- useState : pour stocker à la fois les données du formulaire et les éventuelles erreurs de validation.
- useSearchParams : pour gérer la navigation et extraire l'identifiant du stagiaire depuis l'URL.

## 3. Requêtes envoyées à l'API

Deux requêtes principales sont faites :

1. Lecture des données du stagiaire : une requête GET pour récupérer les informations existantes et les afficher dans le formulaire. api/Stagiaires/\${internId}/

2. Mise à jour des données : une requête PATCH est utilisée pour envoyer les données modifiées au backend. api/Stagiaires/\${internId}

## 4. Validation

- Tous les champs obligatoires doivent être remplis.
- Les noms/prénoms doivent contenir uniquement des lettres.
- L'email doit avoir un format valide.
- Le numéro de téléphone doit respecter un format spécifique

## 3. Captures d'écran

The screenshot shows a 'Modify intern' form with the following fields:

- Last Name: rammal
- First Name: rama
- Email: rama@nextmail.com
- Phone Number: 00 000 000
- Profession: ccne

A yellow 'Modify Intern' button is at the bottom.

avant modification

The screenshot shows the same 'Modify intern' form after validation errors have been displayed:

- Last Name: rammal (error: Only letters are allowed)
- First Name: rama
- Email: rama@nextmail.com
- Phone Number: 123 (error: Format must be 00 000 000)
- Profession: ccne

A yellow 'Modify Intern' button is at the bottom.

validation

## H- Page Details

### 1-Captures d'écran

Home  
Projects  
Trainees  
Supervisors  
Members

CHEHM  
CENTRE D'ETUDES ET DE RECHERCHES EN HÉMATOLOGIE MÉDICALE

Intern Information

Name: rama rammal  
Email: rama@nextmail.com  
Promotion: L2  
Telephone: 22 345 678  
Certified:   
Available:   
Is member:

List of projects by rama rammal

project 1  
Start: 2025-05-16 End: 2025-05-24

Project Information

Title: project 1  
Domain: AI  
Speciality: medical with ai  
Date Registered: 2025-05-15  
Internship Duration  
Start Date: 2025-05-16  
End Date: 2025-05-24  
Year: 2024-2025

Supervisors

batoul hamad  
dana rammal

Files

Convention PDF   
Rapport PDF - No PDF  
Présentation PDF - No PDF  
Prolongement PDF - No PDF

LY lyn admin

Logout

### 1. Contenu de la page

La page affiche les **détails complets d'un stagiaire** inscrit à un stage. Elle comprend :

- Les **informations personnelles du stagiaire** : nom, email, téléphone, promotion, disponibilité, etc.
- La **liste de tous les projets** auxquels ce stagiaire a participé.
- Les **détails du projet actuel** : titre, domaine, spécialité, dates, etc.
- La **liste des superviseurs** associés à ce projet.
- Les **fichiers liés au stage** : convention, rapport, présentation, prolongement.

### 2. Partie frontend

L'interface est organisée en deux grandes sections :

- **En haut de la page :**
  - À gauche : une **carte avec les infos du stagiaire**
  - À droite : une **liste des projets** effectués par ce stagiaire, avec titre et dates
- **En bas de la page :**

Une seule grande section divisée en trois colonnes :
  - **Colonne 1** : informations détaillées du **projet actuel**
  - **Colonne 2** : noms des **superviseurs**, avec indication du superviseur principal
  - **Colonne 3** : liens de **téléchargement des fichiers PDF** (si disponibles)

Hyperliens vers les projets

- **But** : accéder à la page de détails d'un projet/stage spécifique.

Hyperliens vers les superviseurs

- **But** : afficher les informations détaillées d'un superviseur

### 3. Appels API côté frontend

Méthode	URL appelée	Paramètres / Identifiants	But / Description
GET	/api/stagestagiaire/{id}/	id = ID du stage_stagiaire (via URL)	Récupère toutes les informations du stage affecté à un stagiaire donné.
GET	/api/Supervisers/{supId}/	supId = ID d'un superviseur	Récupère les détails (nom, prénom) d'un superviseur.
GET	/api/stagestagiaire/?intern_id={data.intern_id}	intern_id = ID du stagiaire	Récupère la liste de tous les stages associés à un même stagiaire.

### 4. Partie Backend

/api/stagestagiaire/{id}/

Permet de récupérer les détails complets d'un stage spécifique lié à un stagiaire.

Utilise retrieve pour obtenir une seule instance via son identifiant (pk).

/api/Supervisers/{supId}/

Retourne les informations d'un superviseur en fonction de son ID.

Utilise également retrieve pour cibler une ressource unique.

/api/stagestagiaire/?intern\_id={data.intern\_id}

Filtre et retourne tous les stages associés à un stagiaire donné.

Utilise filter dans la vue pour interroger selon un paramètre intern\_id

Donc :

On récupère un stage → on accède au stagiaire et au projet.

Depuis le projet, on récupère les superviseurs (via leurs IDs).

Avec l'ID du stagiaire, on liste tous ses autres stages passés.

## → FORMULAIRE : suppression d'un stagiaire d'un stage

The screenshot shows a list of trainees with one entry for 'rama rammal'. A modal window titled 'Delete intern' is open, asking 'Are you sure you want to permanently delete this intern from this stage?'. The modal has 'Delete' and 'Cancel' buttons.

ID	Name	Promotion	Année	Current Internship	Start Date	End Date	Convention PDF	Certified PDF	Actions
5	rama rammal	L2	2020	stage 1	2020-05-01	2020-05-15	dia/...pdf		Not certified

Il supprime uniquement la relation entre un stagiaire et un projet (dans la table intermédiaire stagestagiaire).

Ce que cela signifie :

Le stagiaire n'est pas supprimé de la base.

Le projet n'est pas supprimé non plus.

Seule l'association entre eux est retirée

API :

/api/stagestagiaire/\${deleteld}/ supprime la **relation stagiaire-projet** identifiée par deleteld, via la vue destroy() de Django REST framework.

## I-Page Supervisors

### 1-Captures d'écran

The screenshot shows a list of supervisors with columns for 'Name', 'Email', 'Profession', and 'Member ?'. The 'Member ?' column contains icons for edit, info, and trash.

ID	Name	Email	Profession	Member ?
8	walid Rammal	zein@gmail.com	ccne	
7	dana rammal	dana@nextmail.com	ccne	
6	fatima hamad	fatima@gmail.com	no need	
3	batoul hamad	batoul@gmail.com	no need	
2	alaq rammal	alaq@nextmail.com	ccne	

## 1. Contenu de la page

La page des superviseurs au sein du système. Elle permet de consulter, rechercher, ajouter, modifier, supprimer et consulter les détails des superviseurs enregistrés. Elle affiche les superviseurs sous forme de tableau avec pagination, chaque ligne présentant les informations clés telles que l'identifiant, le nom complet, l'email, la profession, et une indication de leur statut de membre.

## 2. Partie Frontend

Le frontend est développé avec React.js et utilise Bootstrap pour le design et le responsive. Il intègre plusieurs composants comme :

- Formulaires de recherche avec filtres dynamiques (prénom, nom, email, profession)
- Boutons d'action pour l'ajout, la modification, la suppression et la consultation des superviseurs
- Tableau dynamique paginé pour l'affichage des superviseurs

L'interface offre une navigation fluide grâce à l'utilisation de react-router-dom pour les redirections internes, et react-paginate pour la gestion des pages.

## 3. Appels API côté frontend

Méthode HTTP	Endpoint	Fonctionnalité
GET	/api/Supervisors/	Récupération de la liste des superviseurs avec possibilité de filtrage.
DELETE	/api/Supervisors/<id>/delete-supervisor-role/	Suppression du rôle de superviseur pour un utilisateur existant.

## 4. Partie Backend

GET /api/Supervisors/

Cette méthode permet de récupérer la liste des superviseurs existants dans le système. Elle prend en charge la **pagination** ainsi que des **filtres dynamiques** sur plusieurs champs : first\_name, last\_name, email et profession.

Sur le plan technique, cette fonctionnalité est implémentée à l'aide de la méthode list du ModelViewSet de Django REST Framework, qui est personnalisée pour intégrer les filtres selon les paramètres transmis dans l'URL (via request.query\_params).

DELETE /api/Supervisors/<id>/delete-supervisor-role/

cette méthode permet de **supprimer uniquement le rôle de superviseur** d'un utilisateur spécifique, tout en conservant ses données de membre (car un superviseur est également un membre dans notre modèle).

Contexte technique :

Étant donné que le modèle Supervisor hérite du modèle Member via une **clé étrangère person\_ptr** (héritage multi-table), la suppression classique via la méthode `destroy()` est bloquée, car elle tenterait de supprimer toute la hiérarchie liée.

Solution mise en œuvre :

Nous avons contourné cette limitation en créant une méthode personnalisée à l'aide du décorateur `@action` :

code effectue deux actions principales :

1. Elle supprime manuellement les enregistrements associés dans la table d'association `supervisor_internship` (liée aux stages encadrés)
2. Elle supprime ensuite directement l'entrée correspondante dans la table `Supervisor`, en utilisant des requêtes SQL brutes (raw SQL) pour un contrôle précis et éviter les conflits liés aux relations d'héritage.

`DELETE FROM myapi_supervisor_internship WHERE supervisor_id_id = %s"`

Ce mécanisme garantit une **intégrité des données** tout en permettant de gérer les rôles de manière flexibl

**Captures d'écran:**

ID	Name	Email	Profession	Member ?
1	ahmed rammal	ahmed@nextmail.com	ccone	<input checked="" type="checkbox"/>

ID	Name	Email	Profession	Member ?
8	wold Rammal	zein@gmail.com	ccone	<input checked="" type="checkbox"/>
7	dana rammal	dana@nextmail.com	ccone	<input checked="" type="checkbox"/>
2	aloa rammal	aloa@nextmail.com	ccone	<input checked="" type="checkbox"/>
1	ahmed rammal	ahmed@nextmail.com	ccone	<input checked="" type="checkbox"/>

montrer la pagination

filtrage selon last\_name : rammal

→ FORMULAIRE : Ajout d'un nouveau superviseur

1. Contenu de la page

Cas 1 : Ajout d'un superviseur parmi les membres existants

- Un **dropdown** permet de sélectionner un membre présent dans le système.
- Seuls les membres qui ne sont pas déjà superviseurs peuvent être sélectionnés

Cas 2 : Ajout manuel d'un superviseur externe

- Permet d'ajouter un superviseur qui n'a pas de compte dans le système (pas un membre).

2. Partie Frontend

- Un choix entre **“Membre”** et **“Non membre”** (radio buttons ou onglets).
- Si “Membre” est sélectionné :
  - Un **dropdown** récupère et affiche la liste des membres non superviseurs.
- Si “Non membre” est sélectionné :
  - Un formulaire classique pour entrer manuellement les données du nouveau superviseur (nom, email, etc.).
- Un bouton **“Ajouter”** déclenche la validation du formulaire et l'envoi des données.

### 3. Appels API côté Frontend

Méthode HTTP	URL API	Description
GET	/api/members/non-supervisors	Récupérer la liste des membres disponibles qui ne sont pas encore superviseurs. Permet d'alimenter le dropdown du formulaire.
POST	/api/Supervisors/create_supervisor_from_member/	Enregistre un nouveau superviseur en associant un membre existant sélectionné dans la liste.
POST	/api/supervisors/	Enregistre un nouveau superviseur externe ajouté manuellement via le formulaire.

#### 4-Partie Backend

1-api/Membres/members\_not\_supervisor/ fournit la liste de tous les membres qui ne sont pas encore liés à un rôle de superviseur.

Cette API est une action personnalisée d'un ViewSet qui interroge la base de données pour trouver uniquement les membres qui ne supervisent aucun projet.

Elle retourne ensuite ces membres sous forme de données formatées (JSON) prêtes à être consommées par le frontend.

2-/api/Supervisors/create\_supervisor\_from\_member/ permet de créer un superviseur à partir d'un membre déjà enregistré dans le système.

**Add new Supervisor**

Supervisor Type:

Member  Not a Member

Select Member:

**Add new Supervisor**

Supervisor Type:

Member  Not a Member

First Name:

Last Name:

Profession:

Email:

Phone Number:

Superviser comme membre

superviseur normal

Apres d'ajout

ID	Name	Email	Profession	Member?
15	elie rammal	elie@nextmail.com	ccne	<input checked="" type="radio"/>
14	yara hasan	yara@gmail.com	no need	<input checked="" type="radio"/>
10	tala saab	tala@gmail.com	no need	<input checked="" type="radio"/>
8	walid Rammal	zein@gmail.com	ccne	<input checked="" type="radio"/>
7	dana rammal	dana@nextmail.com	ccne	<input checked="" type="radio"/>

Showing 1 to 5 of 9 entries

## → FORMULAIRE : Modification d'un superviseur

### 1-Captures d'écran:

The screenshot shows a user interface for modifying a supervisor. On the left, there's a sidebar with a logo for 'CHEM' and links to Home, Projects, Trainees, Supervisors, and Members. The main area is titled 'Modify Supervisor' and contains fields for Last Name (filled with 'rammal'), First Name (filled with 'elie'), Profession (filled with 'ccne'), Email (filled with 'elie@nextmail.com'), and Phone (filled with '79 625 167'). A 'Save Changes' button is at the bottom. On the far left, there's a vertical bar with a 'lyn admin' icon and a 'Logout' link.

### 1.partie frontend

- Le formulaire est pré-rempli avec les données actuelles du superviseur, récupérées via une API dédiée.
- Il intègre des validations côté client pour garantir la qualité des données saisies.
- Parmi ces validations, des expressions régulières (regex) sont utilisées pour vérifier certains champs, comme :
  - le format de l'adresse email, le numéro de téléphone, ou d'autres champs spécifiques nécessitant un format précis.

### 2. Appels API côté frontend

Méthode HTTP	URL API	Description
GET	<a href="http://localhost:8000/api/Supervisors/\${id}/">http://localhost:8000/api/Supervisors/\${id}/</a>	Récupère les informations actuelles du superviseur identifié par son ID, pour pré-remplir le formulaire.
PATCH	<a href="http://localhost:8000/api/Supervisors/\${id}/">http://localhost:8000/api/Supervisors/\${id}/</a>	Envoie les données modifiées du superviseur pour mettre à jour son profil côté serveur.

### 3. Fonctionnement côté backend

- La méthode **retrieve** (GET) retourne les données détaillées d'un superviseur spécifique en fonction de son identifiant.
- La méthode **partial\_update** (PATCH) permet de modifier partiellement les informations du superviseur, en ne mettant à jour que les champs fournis dans la requête.

## H- Page Details -Superviser

### 1-Captures d'écran:

The screenshot shows a web application interface for managing users. On the left, there's a sidebar with links: Home, Projects, Trainees, Supervisors, and Members. The main content area has two yellow-highlighted sections. The left section is titled 'Personal Information' and contains fields: Name (alaa rammal), Email (alaa@nextmaiol.com), Telephone (00 000 000), Profession (ccne), and Member? (with a checked checkbox). The right section is titled 'Projects where alaa is Supervisor' and lists several projects with small star icons: TFC System, project 2, project4, project5, and project test.

## 2. Contenu de la page

La page **DetailsSupervisor** affiche les informations détaillées d'un superviseur sélectionné ainsi que la liste des projets dont il est responsable.

- Elle récupère l'identifiant du superviseur depuis les paramètres de l'URL.
- Elle affiche les données personnelles du superviseur : nom, prénom, email, téléphone, profession, et indique s'il est membre ou non du système.
- En dessous des informations personnelles, la page affiche la liste des projets supervisés par cette personne, chaque projet étant cliquable et redirigeant vers une page détaillée du projet.
- Si le superviseur a un rôle particulier (ex. "Admin"), une icône en forme d'étoile apparaît à côté du projet.

## 3. Partie frontend

- **React & Hooks :**
  - useState pour stocker les détails du superviseur et la liste des projets.
  - useEffect pour déclencher la récupération des données au chargement du composant.
  - useSearchParams pour extraire l'ID du superviseur depuis l'URL.
- **Bootstrap & React-Bootstrap :**  
Utilisation des composants Container, Card, Button pour la mise en page et le style.
- **Navigation :**  
Link de react-router-dom permet de naviguer vers la page de détails d'un projet sans recharger la page.
- **Icônes :**  
FaStar pour marquer certains projets avec un statut spécial.

### 3. Appels API côté frontend (Méthodes et usage)

Méthode HTTP	URL API	But
GET	/api/Supervisors/\${id}/	Récupérer les détails personnels du superviseur par son ID.
GET	/api/supstage/?supervisor_id=\${id}	Récupérer la liste paginée des projets où le superviseur est impliqué.

### I- Page Members

#### 1-Captures d'écran:

The screenshot shows a web application interface for managing members. At the top, there are search fields for 'Member Last Name', 'Member First Name', and 'Address', along with a 'Search' button. Below the search bar is a filter section labeled 'Had Payed' with radio buttons for 'Yes', 'No', and 'All'. A 'Add New' button is located in the top right corner. On the left side, there is a sidebar with navigation links: Home, Projects, Trainees, Supervisors, and Members (which is currently selected). The main content area displays a table of member data with columns: Id, First Name, Last Name, Email, Phone, Address, and Has Payed. Each row contains five entries. The 'Has Payed' column includes icons for edit, delete, and other actions. At the bottom of the table, it says 'Showing 1 to 5 of 5 entries' and has navigation arrows. On the far left, there is a user profile icon with 'lyn admin' and a 'Logout' link.

Id	First Name	Last Name	Email	Phone	Address	Has Payed
14	yara	hasan	yara@gmail.com	79 625 167	nabatiyeh	
10	tala	saab	tala@gmail.com	07 625 167	nabatiyeh	
6	fatima	hamad	fatima@gmail.com	79 625 167	nabatiyeh	
3	batoul	hamad	batoul@gmail.com	79 625 167	daraya	
2	alaa	rammal	alaa@nextmaiol.com	00 000 000	nabatiyeh	

#### 2. Contenu de la page

Affiche une liste paginée des membres avec filtres sur :  
 Prénom, Nom de famille, Adresse, Statut "Has Payed" (payé ou non)  
 Recherche dynamique basée sur les filtres.  
 Pagination (5 résultats par page).  
 Bouton d'ajout d'un nouveau membre (redirige vers un formulaire).  
 Actions possibles sur chaque membre :  
 Modifier, Voir détails, Supprimer (avec confirmation modale).

### 3. Appels API côté frontend

Méthode	Endpoint	Description
GET	/api/Membres/	Récupérer la liste paginée des membres avec filtres (nom, adresse, payé)
DELETE	/api/Membres/{id}/delete-member-role/	Supprimer définitivement un membre par ID

→ FORMULAIRE : Ajout d'un nouveau superviseur

1-Captures d'écran:



Home  
Projects  
Trainees  
Supervisors  
Members

lyn  
admin  
Logout

Nouveau membre

Depuis un superviseur existant

Depuis un stagiaire

2. Contenu de la page

- Composant React AddMember avec useState pour gérer :

userType : type de membre à ajouter (newMember, supervisor, intern)

Sélection du superviseur (selectedSupervisorId) et du stagiaire (selectedInternId) selon le cas

useEffect pour charger la liste des superviseurs ou stagiaires selon userType

- **Gestion des formulaires :**

Champs conditionnels affichés selon userType

Validation simple (ex: contrôle du type de fichier, présence des champs obligatoires)

Soumission via une fonction handleSubmit qui appelle la fonction adaptée selon le type choisi

### 3-. Appels API côté frontend

Méthode	Endpoint	Description
GET	/api/Supervisors/?no_member=true	Récupère la liste des superviseurs qui ne sont pas encore membres (pour sélection)
GET	api/Stagiaires/?id_membre_isnull=true	Récupère la liste des stagiaires qui ne sont pas encore membres (pour sélection)
POST	/api/Membres/	Crée un nouveau membre à partir du formulaire classique (nouveau membre)
POST	api/Membres/create_member_from_supervisor/	Crée un nouveau membre à partir d'un superviseur existant, avec données additionnelles
POST	/api/Stagiaires/create_member_from_intern/	Convertit un stagiaire en membre avec données additionnelles

### 3-Partie backend

/api/Supervisers/?no\_member=true

La méthode personnalisée filter\_no\_member

- Cette méthode est déclenchée uniquement quand on utilise le filtre no\_member dans l'URL.

- Si la valeur de no\_member est vraie (true), on applique un filtre SQL

Id\_Membre\_\_isnull=True pour ne garder que les superviseurs qui n'ont pas de membre lié.

api/Stagiaires/create\_member\_from\_intern/

Cette méthode personnalisée dans une ViewSet (avec le décorateur @action)

- Elle récupère les données reçues en POST (via request.data).
- Elle vérifie que les champs obligatoires sont présents.
- Elle récupère un objet Intern à partir de son ID.
- En transaction atomique, elle crée un nouveau Member avec les infos fournies, en utilisant la même ID que l'intern (probablement à cause du multi-héritage ou relation 1-to-1). Elle lie ensuite l'intern avec ce nouveau membre (intern.Id\_Membre = member).
- Elle met à jour un booléen available à False (le stagiaire n'est plus disponible, probablement car il est devenu membre).

api/Membres/create\_member\_from\_supervisor/

### Récupération des données

- L'ID du superviseur (supervisor\_id) est fourni dans les données POST.

D'autres champs obligatoires sont aussi récupérés : nom du père, date/lieu de naissance, adresse, groupe sanguin, profession, domaine, etc.

### Vérification des champs

- Si un champ essentiel est manquant, la fonction retourne une erreur (400 BAD REQUEST).

### Recherche du superviseur

- On cherche un Supervisor avec l'ID donné.
- S'il n'existe pas, on retourne une erreur (404 NOT FOUND).

### Création du membre (Member)

Si le superviseur existe et qu'il n'est pas déjà lié à un Member, on crée un Member :

- En utilisant **exactement le même ID que le superviseur** → cela permet de garder la même personne de base.
- Ce comportement repose sur une structure où Member, Supervisor, etc., héritent probablement d'une classe Person.

### Liaison entre le superviseur et le membre

- Le superviseur est mis à jour avec un lien (Id\_Membre = member) pour indiquer qu'il est maintenant aussi un membre.

- FORMULAIRE : Modification d'un membre

### 1-Captures d'écran:



Modify member

First name	Last name
yara	hosson
Father name	Date of birth
iyad	2025/05/17
Place of birth	Phone number
saida	79 626 167
Address	Blood Type
nabatiyeh	O+
john	Profession

## 2-Contenu de la page

1. Chargement automatique des données du membre à partir de l'ID présent dans l'URL.
2. Affichage d'un formulaire pré rempli avec les données existantes et on peut les modifier
3. La soumission du formulaire modifie les données via un appel PATCH.
4. Affiche le fichier PDF existant et permet d'en télécharger un nouveau.

## 3-Partie frontend

useSearchParams() est utilisé pour récupérer l'id du membre depuis l'URL.

### 3-Appels API côté frontend

Méthode	Endpoint	Description
GET	/api/Membres/{id}/	Récupérer les données du membre avec son ID pour pré remplir le formulaire.
PATCH	/api/Membres/{id}/	Met à jour les informations du membre (formulaire envoyé en multipart).

## J- Page details Members

### 1-Contenu de la page

- affiche les **informations personnelles** d'un membre
- **historique des paiements**, avec la possibilité de :
  1. Générer dynamiquement un **reçu PDF** via une **template Django**.
  2. Sauvegarder ce paiement dans la base.
  3. Remplacer ce PDF par une version signée (upload).

### 2-Partie frontend

Récupération des données du membre

```
.get(`http://localhost:8000/api/Membres/${id}`)
```

- Récupère les **données personnelles** d'un membre spécifique en utilisant son ID.
- Le useEffect assure que la récupération s'exécute **à chaque changement de id dans l'URL**.

- Ces données alimentent le **bloc gauche** "Personal Information".
2. Récupération de l'historique des paiements  
`.get('http://localhost:8000/api/payment-history/?Id_Membre=${id}')`
- Résultat affiché dans un tableau sous la section "**Payment History**" (bloc droit).
3. Génération dynamique du reçu PDF  
`.get('http://localhost:8000/api/generate-receipt/${id}/')`
- Envoie une requête GET au backend pour **générer un reçu PDF dynamiquement**.
  - Le backend utilise probablement une **template HTML Django convertie en PDF** (`xhtml2pdf`).

#### 4. Sauvegarde du paiement dans la base de données

- `.post('http://localhost:8000/api/add-payment/'...)`
- Appel à une route POST pour **enregistrer un nouveau paiement**

#### 5. Remplacement du reçu PDF par version signée

Lorsque le reçu initial n'a **pas encore de signature**, un bouton "Replace PDF" apparaît.

##### a. Sélection du fichier

- Permet à l'utilisateur d'uploader un fichier (PDF signé).

##### b. Envoi du fichier au backend

`axios.patch('http://localhost:8000/api/payment-history/${selectedPaymentId}/...');`

- Envoie le fichier au backend via une requête PATCH.
- Le champ `Payment_received_PDF_with_signature` est mis à jour.
- Ce fichier remplace donc le reçu initial non signé.

### 3-Appels API côté frontend

Méthode HTTP	Endpoint API	Description
GET	<code>/api/Membres/{id}/</code>	Récupère les informations personnelles d'un membre à partir de son id.
GET	<code>/api/payment-history/?Id_Membre={id}</code>	Récupère l'historique des paiements du membre identifié.
GET	<code>/api/generate-receipt/{id}/</code>	Génère dynamiquement un reçu PDF pour le membre (via un template Django).
POST	<code>/api/add-payment/</code>	Enregistre un nouveau paiement dans l'historique, avec lien vers le reçu.
PATCH	<code>/api/payment-history/{payment_id}/</code>	Remplace le PDF d'un paiement existant par une version signée manuellement.

### 4-partie Backend

`generate_receipt(request, member_id)`

**Objectif** : Générer un fichier PDF de reçu pour un membre donné, à partir d'un template HTML.

Étapes :

1. **Vérification de méthode HTTP** : Accepte uniquement GET.
2. **Récupération du membre** via son id.
3. **Calcul des dates** :
  - payment\_date : date actuelle.
  - next\_payment : 1 an plus.
4. **Génération du PDF** :
  - Utiliser un template HTML (receipt.html) rempli avec les infos du membre.
  - Convertir le HTML en PDF avec xhtml2pdf (pisa.CreatePDF).
5. **Sauvegarde du fichier** :  
Génère un nom de fichier (receipt\_<id>\_<date>.pdf).
  - Sauvegarde dans MEDIA\_ROOT/PDF/Payment/.
6. **Réponse JSON** :
  - Retourne l'URL du fichier PDF généré.

Exemple de réponse :

```
{  
    "pdf_url": "/media/PDF/Payment/receipt_3_2025-05-17.pdf"  
}
```

add\_payment(request)

**Objectif** : Enregistrer un nouveau paiement pour un membre, avec le chemin du reçu PDF.

Étapes :

1. **Vérification méthode HTTP** : Accepte uniquement POST.
2. **Lecture des données JSON** :
  - member\_id : identifiant du membre.
  - payment\_pdf\_path : chemin du reçu PDF.
3. **Récupération du membre** et calcul des dates.
4. **Enregistrement en base de données** :
  - Crée un objet Payment\_history avec :
    - membre lié (Id\_Membre)
    - chemin relatif du PDF (Payment\_received\_PDF)
    - dates (paiement et prochaine échéance)
    - payed=True

5-Captures d'écran:



**Member Details**

[Generate Receipt & Save Payment](#)

**Personal Information**

Name:	yara hasan	Date of Birth:	2025-05-17
Place of Birth:	saida	Telephone:	79 625 167
Address:	nabatiyeh	Blood Group:	O+
Profession:	no need	Domain:	ccone
Email:	yara@gmail.com	Supervisor or no:	<input checked="" type="checkbox"/>
Other Association:	<input checked="" type="checkbox"/>		

**Payment History**

No payment history found for this member.

**lyn** admin

[Logout](#)

Avant d'ajout d'un paiement

**Member Details**

[Generate Receipt & Save Payment](#)

**Personal Information**

Name:	yara hasan	Date of Birth:	2025-05-17
Place of Birth:	saida	Telephone:	79 625 167
Address:	nabatiyeh	Blood Group:	O+
Profession:	no need	Domain:	ccone
Email:	yara@gmail.com	Supervisor or no:	<input checked="" type="checkbox"/>
Other Association:	<input checked="" type="checkbox"/>		

**Payment History**

Payment Date	Next Payment Date	Payed	PDF	Replace PDF
2025-05-17	2026-05-17	<input checked="" type="checkbox"/>	<a href="#">View PDF</a>	<a href="#">Replace PDF</a>

Apres clique sur bouton Generate receipt & save payment

## Payment Receipt

**Name:** yara hasan  
**Amount Paid:**  
**Payment Date:** May 17, 2025 **Next Payment Due:** May 17, 2026

Member's Signature

Admin's Signature

TOGETHER FOR CHEHIM

[view PDF](#)

**Member Details**

[Generate Receipt & Save Payment](#)

**Personal Information**

Name:	yara hasan
Place of Birth:	saida
Address:	nabatiyeh
Profession:	no need
Email:	yara@gmail.com

**Payment History**

Date	Next Payment Date	Payed	PDF	Replace PDF
2025-05-17	2026-05-17	<input checked="" type="checkbox"/>	<a href="#">View PDF</a>	<a href="#">Replace PDF</a>

**Upload New PDF**

Select New Signed PDF

Choose File No file chosen

[Cancel](#) [Upload & Replace](#)

57

remplacer PDF

The screenshot shows the 'Member Details' page of the TFC Management System. On the left, there's a sidebar with links for Home, Projects, Trainees, Supervisors, and Members. The main content area has two yellow-highlighted sections: 'Personal Information' and 'Payment History'. The 'Personal Information' section contains fields like Name (yara hasan), Date of Birth (2025-05-17), Place of Birth (sodda), Telephone (79 625 167), Address (nobatiyah), Blood Group (O+), Profession (no need), Email (yara@gmail.com), Supervisor or no: (with a blue button), and Other Association (with a blue button). The 'Payment History' section shows a single entry: Payment Date 2025-05-17, Next Payment Date 2026-05-17, Payed (with a blue button), and PDF (with a blue button). At the top right is a 'Generate Receipt & Save Payment' button.

après le remplacement

## Conclusion

Dans le cadre de ce projet de développement du **TFC Management System**, une plateforme web destinée à la gestion des stages de fin d'études, j'ai pu acquérir et renforcer de nombreuses compétences techniques, en particulier dans le développement **full stack**. Le backend a été développé avec **Django** et le frontend avec **React.js**, deux technologies modernes que j'ai appris à maîtriser de manière approfondie.

Dans un premier temps, j'ai été confronté à un système existant complexe, ce qui m'a permis d'améliorer ma capacité à analyser des problèmes, à identifier les bugs, et à les résoudre de manière efficace. Cela m'a également amené à améliorer la structure de la base de données, notamment en mettant en œuvre **l'héritage multiple (multi-inheritance)** pour éviter la redondance des informations et réduire les erreurs logiques.

Par ailleurs, j'ai utilisé **Figma** pour concevoir un design plus professionnel de l'interface utilisateur. Cet outil m'a permis de créer des maquettes claires et fonctionnelles, facilitant la validation des écrans par le client avant l'étape de développement, et évitant ainsi de réécrire inutilement du code.

J'ai aujourd'hui une **compréhension solide de Django**, de son ORM et du **Django REST Framework** pour la création d'API sécurisées. J'ai également intégré des mécanismes d'**authentification, de validation et de gestion des autorisations** afin d'assurer la sécurité et la fiabilité du système. Django offre en effet une base sécurisée, notamment contre les injections SQL grâce à son ORM robuste.

Fonctionnalités principales du système :

- Une **page d'accueil dynamique** présentant les informations essentielles : projets sans stagiaire, membres inactifs ou en attente de paiement, stages arrivant à échéance.
- Une interface claire avec des **liens hypertextes** et des **boutons "voir plus"** pour accéder rapidement aux détails.
- Des **tableaux filtrables et dynamiques** dans toutes les pages (stagiaires, membres, projets, etc.), avec des actions disponibles : modifier, voir les détails, ou supprimer.
- Des **opérations avancées de gestion** :
  - Création d'un **superviseur**, ou d'un **superviseur à partir d'un membre** ;
  - Ajout d'un **membre**, y compris à partir d'un stagiaire ou d'un superviseur ;
  - Dans la page projet, possibilité de **modifier les superviseurs, d'ajouter ou supprimer un stagiaire** associé.
  - **Génération dynamique des paiements** : création automatique d'un **reçu PDF (receipt)** lié au membre, sauvegarde du fichier dans le système, et enregistrement sécurisé de l'historique de paiement dans la base de données.

## Références bibliographiques

### 1. Documentation officielle :

- Django Project. *Authentication system — Django documentation*. Disponible sur :  
<https://docs.djangoproject.com/en/5.2/ref/contrib/auth/>
- Django Project. *Django Tutorial — W3Schools*. Disponible sur :  
<https://www.w3schools.com/django/>

### 2. Articles et tutoriels en ligne :

- Mahmatali Mas. *Remember me login feature Django*. Medium. Disponible sur :  
<https://medium.com/@mahmutali.mas/remember-me-login-feature-django-b58558b8d56d>
- GeeksforGeeks. *Django Tutorial*. Disponible sur :  
<https://www.geeksforgeeks.org/django-tutorial/>
- Django Forum. *Remember me checkbox - Django*. Disponible sur :  
<https://forum.djangoproject.com/t/remember-me-checkbox/19381>

### 3. Packages et bibliothèques :

- PyPI. *django-auth-remember*. Disponible sur :  
<https://pypi.org/project/django-auth-remember/>

### 4. Ressources vidéo :

- Corey Schafer. *Django Tutorials for Beginners [YouTube Playlist]*. Disponible sur :  
<https://www.youtube.com/watch?v=UmljXZIypDc&list=PL-osiE80TeTtoQCKZ03TU5fNfx2UY6U4p>