École Polytechnique Sousse

Année Universitaire 2025-2026

Student Arena

Plateforme de Compétition de Projets pour Étudiants

Rapport Technique

Projet Full-Stack MERN

MongoDB • Express.js • React • Node.js

Réalisé par :

Maroua Hattab

Janvier 2026

Table des matières

1 Introduction

1.1 Contexte du Projet

Ce rapport présente le projet **Student Arena**, une plateforme web moderne dédiée aux compétitions de projets pour étudiants. Cette application a été développée dans le cadre du projet final du cours Stack MERN à l'École Polytechnique de Sousse.

1.2 Objectifs

Les objectifs principaux de ce projet sont :

— Démontrer la maîtrise du développement full-stack moderne avec le stack MERN
— Mettre en pratique les architectures REST API et SPA (Single Page Application)
— Appliquer les bonnes pratiques de sécurité et de structuration du code
— Intégrer des fonctionnalités avancées d'Intelligence Artificielle (Gemini API)
— Produire un code propre, maintenable et documenté

1.3 Description de l'Application

Student Arena est une plateforme web permettant aux étudiants de :

— Participer à des compétitions de projets (en solo ou en équipe)
— Créer et gérer des équipes collaboratives
— Soumettre leurs travaux via des liens GitHub
— Grimper dans le classement mondial grâce à un système de points innovant
— Bénéficier de fonctionnalités IA (génération de bio, recommandations, chatbot)

2 Système de Rôles

2.1 Vue d'Ensemble

L'application implémente un système de rôles avec **deux niveaux d'accès** :

Rôle	Code	Description
Utilisateur	`user`	Étudiant standard avec accès aux fonctionnalités de participation
Administrateur	`admin`	Super-utilisateur avec accès complet à la gestion de la plateforme

Table 1 – Les deux rôles du système

2.2 Attribution Automatique du Rôle Admin

Règle d'Administration

Important : Le **premier utilisateur** à s'inscrire sur la plateforme devient automatiquement administrateur. Tous les utilisateurs suivants sont des utilisateurs standards par défaut.

```
// authController.js - Inscription
const register = async (req, res) => {
  // ...

  // Le premier utilisateur devient automatiquement admin
  let role = 'user';
  const userCount = await User.countDocuments();
  if (userCount === 0) {
    role = 'admin';  // Premier utilisateur = Admin
  }

  // Creer l'utilisateur avec le role determine
  const user = await User.create({
    firstName, lastName, email, password: hashedPassword,
    userName, role, createdAt: new Date()
  });
  // ...
};
```

Listing 1 – Logique d'attribution du rôle admin

2.3 L'Administrateur est aussi un Utilisateur

L'administrateur hérite de **toutes les fonctionnalités d'un utilisateur standard**, en plus de ses privilèges d'administration :

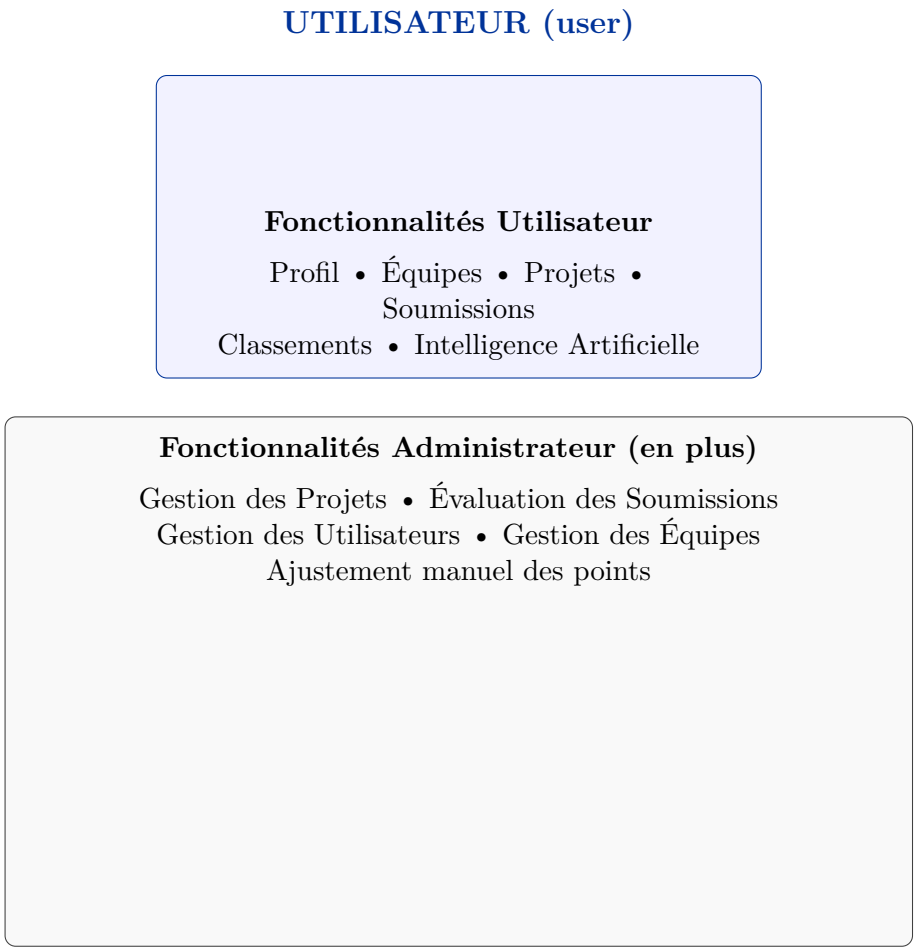

FIGURE 1 – Hiérarchie des rôles

3 Architecture Technique

3.1 Stack Technologique

Le projet utilise le stack MERN, composé des technologies suivantes :

Composant	Technologies
Frontend	React 19, Vite, React Router 7, Axios, Vanilla CSS
Backend	Node.js, Express.js, REST API
Base de données	MongoDB avec Mongoose ODM
Authentification	JWT (JSON Web Tokens), Bcrypt.js
Intelligence Artificielle	Google Gemini 1.5 Flash API
Sécurité	Helmet, CORS, Express-Validator

TABLE 2 – Stack technologique du projet

3.2 Architecture Globale

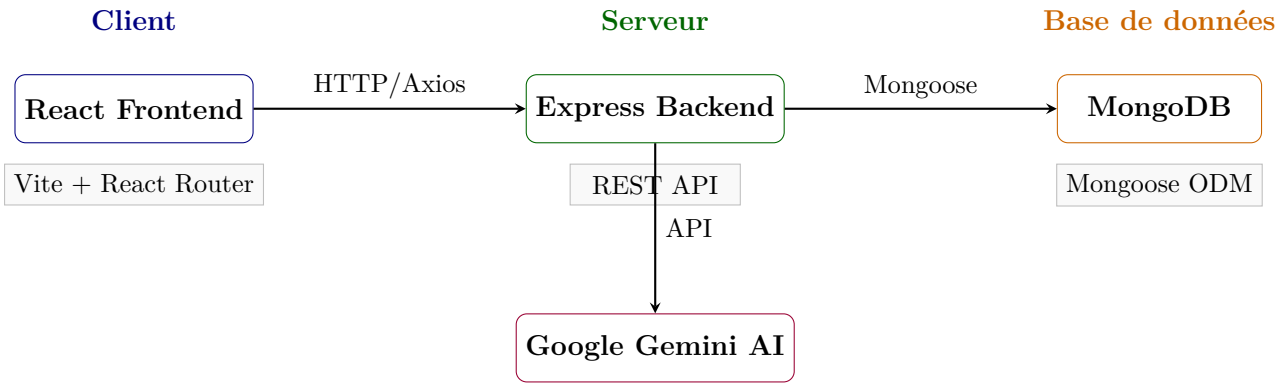

FIGURE 2 – Architecture globale de l'application

3.3 Structure du Projet

```
projet-mern/
|-- Backend/
|   |-- config/          # Configuration DB & AI
|   |-- controllers/     # Logique metier (7 fichiers)
|   |-- middleware/      # Auth JWT & Validation
|   |-- models/          # Schemas Mongoose (5 entites)
|   |-- routes/          # Routes API (7 fichiers)
|   |-- server.js        # Point d'entree
|   +-- package.json
|-- Frontend/
|   +-- student-arena/
|       |-- src/
|       |   |-- api/         # Configuration Axios
|       |   |-- components/  # Composants reutilisables
|       |   |-- context/     # AuthContext
|       |   |-- pages/       # 15 pages
|       |   +-- utils/       # Validation
|       +-- package.json
|-- img/
|   +-- logo.jpg         # Logo de la plateforme
+-- README.md
```

Listing 2 – Structure des dossiers

4 Modèle de Données

4.1 Vue d'Ensemble des Entités

Le projet contient **5 entités principales** conformément aux exigences du cahier des charges :

Entité	Description	Rôle
User	Représente un utilisateur (étudiant ou admin)	Authentification, Participation
Team	Représente une équipe collaborative	Groupement, Compétition
Project	Représente un projet/compétition	Gestion des défis
Submission	Représente une soumission de projet	Évaluation, Classement
Leaderboard	Représente les classements	Gamification, Rankings

TABLE 3 – Les 5 entités du modèle de données

4.2 Diagramme Entité-Relation

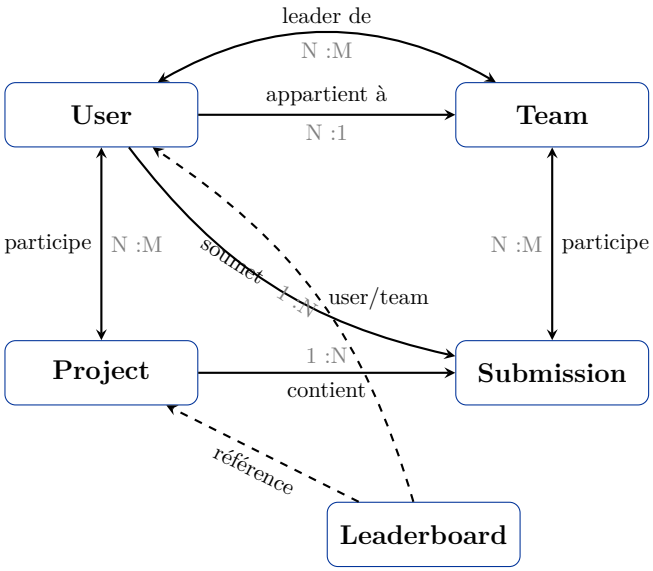

FIGURE 3 – Diagramme Entité-Relation simplifié

4.3 Détail des Relations

Conformément au cahier des charges, le projet implémente les types de relations suivants :

4.3.1 Relations 1-to-1 (Un à Un)

— **User** → **Team** (appartenance unique) : Un utilisateur ne peut appartenir qu'à une seule équipe à la fois.

4.3.2 Relations 1-to-Many (Un à Plusieurs)

— **User** → **Submission** : Un utilisateur peut avoir plusieurs soumissions.
— **Team** → **Submission** : Une équipe peut avoir plusieurs soumissions.
— **Project** → **Submission** : Un projet peut recevoir plusieurs soumissions.
— **User (Admin)** → **Project** : Un admin peut créer plusieurs projets.
— **Team** → **User (members)** : Une équipe contient plusieurs membres.

4.3.3 Relations Many-to-Many (Plusieurs à Plusieurs)

— **User** ↔ **Project** (registeredProjects) : Participation multiple.
— **Team** ↔ **Project** (registeredProjects) : Participation d'équipes.
— **User** ↔ **Team** (leaders) : Max 2 leaders par équipe.

4.4 Schémas Mongoose Détaillés

4.4.1 User Schema

```
const userSchema = new mongoose.Schema({
  firstName: String,
  lastName: String,
  email: String,
  password: String,           // Hashe avec bcrypt
  userName: String,
  bio: String,
  team: { type: ObjectId, ref: 'Team' },        // Relation 1:1
  points: { type: Number, default: 0 },
  submissions: [{ type: ObjectId, ref: 'Submission' }],  // 1:N
  registeredProjects: [{ type: ObjectId, ref: 'Project' }], // N:
   M
  role: { type: String, enum: ['user', 'admin'] },
  isTeamLeader: { type: Boolean, default: false }
});
```

Listing 3 – Modèle User

4.4.2 Team Schema

```
const teamSchema = new mongoose.Schema({
  name: String,
  description: String,
  slogan: String,
  leaders: [{                 // N:M (max 2 leaders)
    type: ObjectId,
    ref: 'User'
  }],
  members: [{                 // 1:N
    type: ObjectId,
    ref: 'User'
  }],
  invitationCode: String,     // Code unique d'invitation
  points: { type: Number, default: 0 },
  submissions: [{ type: ObjectId, ref: 'Submission' }],
  registeredProjects: [{ type: ObjectId, ref: 'Project' }],
  status: { enum: ['active', 'inactive', 'archived'] }
});
```

Listing 4 – Modèle Team

4.4.3 Project Schema

```
const projectSchema = new mongoose.Schema({
  title: String,
  description: String,
  successCriteria: String,
  tags: [String],            // Max 10 tags
  type: { enum: ['individual', 'team'] },
  startDate: Date,
  endDate: Date,
  createdBy: { type: ObjectId, ref: 'User' },  // Admin
  participants: [{ type: ObjectId, refPath: 'participantType' }],
  submissions: [{ type: ObjectId, ref: 'Submission' }],
  // Systeme de points
  firstPlacePoints: { type: Number, default: 100 },
  secondPlacePoints: { type: Number, default: 75 },
  thirdPlacePoints: { type: Number, default: 50 },
  otherParticipantsPoints: { type: Number, default: 25 },
  status: { enum: ['draft', 'active', 'completed', 'archived'] }
});
```

Listing 5 – Modèle Project

4.4.4 Submission Schema

```
const submissionSchema = new mongoose.Schema({
  project: { type: ObjectId, ref: 'Project', required: true },
  submittedByUser: { type: ObjectId, ref: 'User' },
  submittedByTeam: { type: ObjectId, ref: 'Team' },
  githubLink: String,        // Format: https://github.com/user/
   repo
  status: { enum: ['pending', 'approved', 'rejected'] },
  score: { type: Number, min: 0, max: 100 },
  ranking: { type: Number, min: 1 },
  feedback: String,
  reviewedBy: { type: ObjectId, ref: 'User' },  // Admin
   evaluateur
  reviewedAt: Date
});
```

Listing 6 – Modèle Submission

5 Cas d'Utilisation

Cette section présente les fonctionnalités disponibles selon le rôle de l'utilisateur, sous forme de **user stories**.

5.1 En tant qu'Utilisateur (Étudiant)

5.1.1 Authentification

- ☐ **En tant qu'utilisateur**, je peux **créer un compte** avec mon email, nom d'utilisateur et mot de passe.
- ☐ **En tant qu'utilisateur**, je peux **me connecter** avec mon email ou nom d'utilisateur.
- ☐ **En tant qu'utilisateur**, je peux **me déconnecter** de la plateforme.

5.1.2 Gestion du Profil

- ☐ **En tant qu'utilisateur**, je peux **consulter mon profil** (points, équipe, projets).
- ☐ **En tant qu'utilisateur**, je peux **modifier mes informations** (prénom, nom, bio).
- ☐ **En tant qu'utilisateur**, je peux **changer mon mot de passe**.
- ☐ **En tant qu'utilisateur**, je peux **générer ma bio avec l'IA**.

5.1.3 Gestion d'Équipe

- ☐ **En tant qu'utilisateur**, je peux **créer une nouvelle équipe** et devenir leader.
- ☐ **En tant qu'utilisateur**, je peux **rejoindre une équipe** via un code d'invitation.
- ☐ **En tant que leader**, je peux **ajouter des membres** à mon équipe.
- ☐ **En tant que leader**, je peux **retirer des membres** de mon équipe.
- ☐ **En tant que leader**, je peux **promouvoir un membre** au rang de co-leader (max 2).
- ☐ **En tant qu'utilisateur**, je peux **quitter mon équipe**.

5.1.4 Participation aux Projets

- ☐ **En tant qu'utilisateur**, je peux **explorer tous les projets** disponibles.
- ☐ **En tant qu'utilisateur**, je peux **m'inscrire à un projet individuel**.
- ☐ **En tant que leader**, je peux **inscrire mon équipe** à un projet d'équipe.
- ☐ **En tant qu'utilisateur**, je peux **voir mes inscriptions** en cours.

5.1.5 Soumissions

- ☐ **En tant qu'utilisateur**, je peux **soumettre un lien GitHub** pour un projet.
- ☐ **En tant qu'utilisateur**, je peux **suivre l'état** de mes soumissions (pending, approved, rejected).
- ☐ **En tant qu'utilisateur**, je peux **consulter le feedback** de l'évaluateur.

5.1.6 Classements

- ☐ **En tant qu'utilisateur**, je peux **consulter le classement global** des utilisateurs et équipes.
- ☐ **En tant qu'utilisateur**, je peux **consulter le classement par projet**.

5.1.7 Intelligence Artificielle

- ☐ **En tant qu'utilisateur**, je peux **chatter avec l'assistant IA** pour obtenir de l'aide.
- ☐ **En tant qu'utilisateur**, je peux **obtenir des recommandations de projets** personnalisées.
- ☐ **En tant qu'utilisateur**, je peux **générer des idées de projets** innovantes.

5.2 En tant qu'Administrateur

Note : L'administrateur a accès à toutes les fonctionnalités utilisateur listées ci-dessus, **plus** les fonctionnalités suivantes :

5.2.1 Gestion des Projets (AdminProjects)

- ☐ **En tant qu'admin**, je peux **créer un nouveau projet** avec titre, description, dates, type et points.
- ☐ **En tant qu'admin**, je peux **modifier les détails** d'un projet existant.
- ☐ **En tant qu'admin**, je peux **changer le statut** d'un projet :
  - Draft → Active (visible et ouvert aux inscriptions)
  - Active → Completed (terminé, plus d'inscriptions)
  - Completed → Archived (archivé dans l'historique)
- ☐ **En tant qu'admin**, je peux **supprimer un projet**.
- ☐ **En tant qu'admin**, je peux **utiliser l'IA pour générer** :
  - Des tags automatiques
  - Une description améliorée
  - Des critères de succès

5.2.2 Évaluation des Soumissions (AdminSubmissions)

- ☐ **En tant qu'admin**, je peux **voir toutes les soumissions** de tous les projets.
- ☐ **En tant qu'admin**, je peux **consulter le lien GitHub** d'une soumission.
- ☐ **En tant qu'admin**, je peux **approuver une soumission**.
- ☐ **En tant qu'admin**, je peux **rejeter une soumission** avec un feedback.
- ☐ **En tant qu'admin**, je peux **attribuer un score** (0-100).
- ☐ **En tant qu'admin**, je peux **attribuer un rang** (1er, 2e, 3e, Top 10).
- ☐ **En tant qu'admin**, je peux **écrire un feedback** détaillé.

5.2.3 Gestion des Utilisateurs (AdminUsers)

- ☐ **En tant qu'admin**, je peux **voir la liste** de tous les utilisateurs.
- ☐ **En tant qu'admin**, je peux **créer manuellement** un nouvel utilisateur.
- ☐ **En tant qu'admin**, je peux **modifier les informations** d'un utilisateur.
- ☐ **En tant qu'admin**, je peux **changer le rôle** d'un utilisateur (user ↔ admin).
- ☐ **En tant qu'admin**, je peux **supprimer** un compte utilisateur.
- ☐ **En tant qu'admin**, je peux **ajouter des points bonus** à un utilisateur.
- ☐ **En tant qu'admin**, je peux **retirer des points (malus)** à un utilisateur.

5.2.4 Gestion des Équipes (AdminTeams)

- ☐ **En tant qu'admin**, je peux **voir toutes les équipes** avec leurs statistiques.
- ☐ **En tant qu'admin**, je peux **modifier** les informations d'une équipe (nom, slogan, description).
- ☐ **En tant qu'admin**, je peux **voir les membres** d'une équipe.
- ☐ **En tant qu'admin**, je peux **ajouter un membre** à une équipe par email/username.
- ☐ **En tant qu'admin**, je peux **retirer un membre** d'une équipe.
- ☐ **En tant qu'admin**, je peux **supprimer une équipe**.
- ☐ **En tant qu'admin**, je peux **ajouter des points bonus** à une équipe.
- ☐ **En tant qu'admin**, je peux **retirer des points (malus)** à une équipe.

5.3 En tant que Visiteur (Non Connecté)

- ☐ **En tant que visiteur**, je peux **explorer les projets publics**.
- ☐ **En tant que visiteur**, je peux **consulter le classement global**.
- ☐ **En tant que visiteur**, je peux **voir les équipes publiques**.
- ☐ **En tant que visiteur**, je peux **créer un compte** pour accéder aux fonctionnalités.
- ☐ **En tant que visiteur**, je peux **me connecter** avec un compte existant.

5.4 Tableau Récapitulatif des Permissions

Fonctionnalité	Visiteur	User	Admin
Authentification			
Inscription / Connexion	✓	✓	✓
Profil			
Consulter/Modifier son profil		✓	✓
Générer bio IA		✓	✓
Équipes			
Créer/Rejoindre une équipe		✓	✓
Gérer les membres (si leader)		✓	✓
Gérer TOUTES les équipes			✓
Projets			
Explorer les projets	✓	✓	✓
S'inscrire à un projet		✓	✓
Créer/Modifier/Supprimer un projet			✓
Changer le statut d'un projet			✓
Soumissions			
Soumettre un projet		✓	✓
Évaluer les soumissions			✓
Attribuer des rangs			✓
Classements			
Consulter le classement	✓	✓	✓
Utilisateurs			
Gérer tous les utilisateurs			✓
Ajuster les points manuellement			✓
Intelligence Artificielle			
Chatbot / Recommandations / Idées		✓	✓
Générer tags/description/critères			✓

Table 4 – Matrice des permissions par rôle

6 Documentation API REST

Toutes les routes sont préfixées par `/api`. Les routes marquées [**AUTH**] nécessitent un token JWT. Les routes marquées [**ADMIN**] nécessitent des privilèges administrateur.

6.1 Authentification (/api/auth)

Méthode	Route	Accès	Description
POST	/register	Public	Inscription d'un nouvel utilisateur
POST	/login	Public	Connexion et réception du token JWT
GET	/me	[AUTH]	Récupérer les infos de l'utilisateur connecté

6.2 Utilisateurs (/api/users)

Méthode	Route	Accès	Description
GET	/profile	[AUTH]	Profil complet de l'utilisateur
GET	/leaderboard	Public	Top 10 des utilisateurs
GET	/	[ADMIN]	Liste de tous les utilisateurs
POST	/	[ADMIN]	Création manuelle d'un utilisateur
PUT	/ :id	[AUTH]	Mise à jour du profil
PUT	/ :id/password	[AUTH]	Changement de mot de passe
DELETE	/ :id	[ADMIN]	Suppression d'un compte

6.3 Équipes (/api/teams)

Méthode	Route	Accès	Description
POST	/	[AUTH]	Créer une nouvelle équipe
GET	/	Public	Lister toutes les équipes
GET	/ :id	Public	Détails d'une équipe
POST	/join	[AUTH]	Rejoindre via code d'invitation
POST	/ :id/leave	[AUTH]	Quitter l'équipe actuelle
POST	/ :id/add-member	Leader/[ADMIN]	Ajouter un membre
DELETE	/ :id/members/ :mId	Leader/[ADMIN]	Expulser un membre

6.4 Projets (/api/projects)

Méthode	Route	Accès	Description
GET	/	Public	Liste de tous les projets
GET	/active	Public	Projets ouverts aux inscriptions
GET	/my-projects	[AUTH]	Projets auxquels je participe
GET	/team-projects	[AUTH]	Projets de mon équipe
POST	/	[ADMIN]	Créer un nouveau projet
POST	/ :id/register	[AUTH]	S'inscrire à un projet
PUT	/ :id/status	[ADMIN]	Changer le statut

6.5 Soumissions (/api/submissions)

Méthode	Route	Accès	Description
POST	/	[AUTH]	Soumettre un lien GitHub
GET	/my-submissions	[AUTH]	Voir mes propres soumissions
GET	/	[ADMIN]	Voir toutes les soumissions
PUT	/ :id/review	[ADMIN]	Noter et donner un feedback
PUT	/ :id/rank	[ADMIN]	Assigner un classement
POST	/add-points	[ADMIN]	Ajouter/retirer des points

6.6 Intelligence Artificielle (/api/ai)

Méthode	Route	Accès	Description
POST	/generate-bio	[AUTH]	Générer une bio via Gemini
GET	/recommend-projects	[AUTH]	Projets suggérés par l'IA
POST	/chat	[AUTH]	Interagir avec l'assistant IA
POST	/generate-project-idea	[AUTH]	Générer une idée de projet
POST	/generate-tags	[ADMIN]	Générer des tags pour un projet
POST	/improve-description	[ADMIN]	Améliorer une description
POST	/generate-criteria	[ADMIN]	Générer des critères de succès

7 Sécurité

7.1 Mécanismes Implémentés

Mécanisme	Description
JWT	Authentification stateless avec tokens signés. Expiration configurable.
Bcrypt.js	Hashage des mots de passe avec 10 rounds de salage.
Helmet	Protection contre les vulnérabilités HTTP (XSS, MIME sniffing, etc.)
CORS	Configuration stricte des origines autorisées.
Express-Validator	Validation stricte de toutes les données entrantes.
Variables .env	Secrets stockés dans des variables d'environnement.

TABLE 5 – Mécanismes de sécurité

7.2 Middlewares d'Authentification

```
// authMiddleware.js
const protect = async (req, res, next) => {
  const token = req.headers.authorization?.split(' ')[1];
  if (!token) {
    return res.status(401).json({ message: 'Non autorise' });
  }

  const decoded = jwt.verify(token, process.env.JWT_SECRET);
  req.user = await User.findById(decoded.id);
  next();
};

const admin = (req, res, next) => {
  if (req.user.role !== 'admin') {
    return res.status(403).json({ message: 'Acces refuse - Admin
   requis' });
  }
  next();
```

```
};
```

Listing 7 – Middleware d'authentification

8 Système de Points et Gamification

8.1 Distribution des Points

Position	1ère	2ème	3ème	Autres	
Points par défaut	100	75	50	25	

TABLE 6 – Points attribués par défaut

8.2 Logique de Bonus

8.2.1 Projet Individuel

— L'étudiant reçoit **100%** des points du rang
— Si l'étudiant appartient à une équipe, son équipe reçoit un **bonus de 50%**

8.2.2 Projet en Équipe

— L'équipe reçoit **100%** des points du rang
— Chaque membre reçoit un **bonus individuel de 50%**

9 Intégration de l'Intelligence Artificielle

9.1 Google Gemini 1.5 Flash

Fonctionnalité	Accès	Description
Générateur de Bio	User/Admin	Bio professionnelle automatique
Recommandations	User/Admin	Projets suggérés selon l'historique
Assistant Chatbot	User/Admin	Aide interactive sur la plateforme
Générateur d'Idées	User/Admin	Idées de projets innovantes
Générateur de Tags	Admin	Tags automatiques pour projets
Amélioration Description	Admin	Description professionnelle
Générateur de Critères	Admin	Critères de succès automatiques

TABLE 7 – Fonctionnalités IA par rôle

10 Conclusion

10.1 Récapitulatif

Le projet **Student Arena** répond à l'ensemble des exigences du cahier des charges :

✓ **5 entités minimum** avec relations variées (1 :1, 1 :N, N :M)
✓ **API REST complète** avec opérations CRUD
✓ **Authentification JWT** sécurisée
✓ **Hashage bcrypt** des mots de passe
✓ **Validation des données** avec express-validator
✓ **Protection CORS** et Helmet
✓ **Intégration IA** avec Google Gemini (bonus)
✓ **Interface React** moderne et responsive
✓ **Système de rôles** (user/admin) avec permissions appropriées

10.2 Compétences Démontrées

Ce projet démontre la maîtrise des compétences suivantes :

— Développement full-stack avec le stack MERN
— Conception de bases de données NoSQL avec Mongoose
— Implémentation d'API REST sécurisées
— Gestion de l'authentification et des autorisations
— Intégration d'APIs tierces (Google Gemini)
— Design d'interfaces utilisateur modernes
— Bonnes pratiques de développement et structuration du code