Spécifications Techniques pour FlashMcCards

1. Contexte et Objectifs

1.1 Contexte

L'application web permettra aux utilisateurs de créer, organiser et réviser des flashcards de manière personnalisée.

1.2 Objectifs

- Permettre aux utilisateurs de créer un compte et de se connecter.
- Créer, modifier, supprimer et organiser des flashcards dans des catégories.
- Fournir un mode de révision avec des options personnalisables (mode évaluation, méthode Leitner).
- Offrir une interface utilisateur ludique, intuitive et responsive.

---

2. Architecture Technique

2.1 Frontend

- **Framework** : React.js
- **Gestion d'état** : Redux.
- **Style** : CSS.
- **Outils de Build** : Vite.
- **Librairies supplémentaires** :
    - Axios (pour les appels API) ?
    - React Router (pour la navigation)
    - React Hook Form (pour les formulaires)

2.2 Backend

- **Framework** : Express.js
- **Architecture** : RESTful API
- **Sécurité** : Middleware pour authentification (JWT).
- **Fichier** : Multer

2.3 Base de Données

- **SGBD** : PostgreSQL
- **ORM** : Sequelize ?

---

3. Fonctionnalités

3.1 Fonctionnalités Utilisateur

- **Authentification** :
  - Inscription : Email, mot de passe, pseudo, nom, prénom.
  - Connexion : JWT pour gérer les sessions.
- **Gestion des Flashcards** :
  - Créer, lire, modifier et supprimer (CRUD) sur carte/paquet.
  - Organiser les flashcards par catégories.
  - Fonction de tri par catégories
- **Révision** :
  - Mode évaluation, méthode Leitner
  - Option de suivi des performances (statistiques...).

3.2 Fonctionnalités Administrateur

- Gestion des utilisateurs (blocage, suppression).
- Gestion des cartes et paquets (blocage, suppression).

---

4. API Endpoints

Authentification

Méthode HTTP	Route	Action	Description
POST	/auth/login	loginUser	Connecter un utilisateur
POST	/auth/logout	logoutUser	Déconnecter un utilisateur
POST	/auth/token/refresh	refreshToken	Générer un nouveau token d'accès

Utilisateur

Méthode HTTP	Route	Action	Description
GET	/users	getAllUsers	Récupérer tous les utilisateurs

Méthode HTTP	Route	Action	Description
GET	/users/:userId	getUserById	Récupérer un utilisateur spécifique
POST	/users	createUser	Créer un nouvel utilisateur
PUT	/users/:userId	updateUser	Mettre à jour un utilisateur
PATCH	/users/:userId	patchUser	Mettre à jour partiellement un utilisateur
DELETE	/users/:userId	deleteUser	Supprimer un utilisateur

Deck

Méthode HTTP	Route	Action	Description
GET	/decks	getAllDecks	Récupérer tous les paquets
GET/POST	/decks/search	searchDecks	Rechercher dans les paquets
GET	/decks/:deckId	getDeckById	Récupérer un paquet spécifique
POST	/decks	createDeck	Créer un nouveau paquet
PUT	/decks/:deckId	updateDeck	Mettre à jour un paquet
PATCH	/decks/:deckId	patchDeck	Mettre à jour partiellement un paquet
DELETE	/decks/:deckId	deleteDeck	Supprimer un paquet
GET	/decks/:deckId/evaluate	evaluateDeck	Jouer en mode évaluation
GET	/decks/:deckId/revise	reviseDeck	Jouer en mode Leitner
GET	/decks/:deckId/stats	getStatistics	Obtenir des statistiques (mode/type en params)
POST	/decks/:deckId/export	exportDeck	Exporter un paquet

Store

Méthode HTTP	Route	Action	Description
GET/POST	/store/decks/search	searchStoreDecks	Rechercher des paquets du store
GET	/store/decks/:deckId	getStoreDeckDetails	Obtenir les détails d'un paquet du store

Méthode HTTP	Route	Action	Description
POST	/store/decks/:deckId/import	importDeck	Importer un paquet
POST	/store/decks/:deckId/rate	rate	Noter un paquet du store

Card

Méthode HTTP	Route	Action	Description
GET	/decks/:deckId/cards	getCardsByDeck	Récupérer toutes les cartes d'un paquet (avec pagination (limit/numéro de page en params))
GET	/cards/:id	getCardById	Récupérer une carte spécifique
POST	/cards	createCard	Créer une nouvelle carte
PUT	/cards/:id	updateCard	Mettre à jour une carte
PATCH	/cards/:id	patchCard	Mettre à jour partiellement une carte
DELETE	/cards/:id	deleteCard	Supprimer une carte

---

5. Sécurité

5.1 Frontend

- Utilisation de HTTPS pour tous les échanges.
- Stockage des tokens JWT dans un cookie httpOnly, Secure et Same-Site strict (?).

5.2 Backend

- Hash des mots de passe avec argon2.

5.3 Base de Données

- Paramétrer les rôles et permissions PostgreSQL.

---

6. Outils et Déploiement

6.1 Environnement de Développement

- **Frontend** : React DevTools, ESLint, Prettier.
- **Backend** : Prettier.
- **Base de Données** : pgAdmin4, scripts SQL pour migrations.

6.2 Déploiement

- Docker compose.

6.3 CI/CD

- Azure DevOps **(Git)**

---

7. Tests

Tests Unitaires

- **Backend** : Jest et Postman pour l'API.

---

8. Planning et Rendu

Milestone	Date
Rendu V1 (Dossier d'analyse)	12/01/2025
Audit de projet	14/01/2025
Rendu V2 (Dossier d'analyse corrigé)	26/01/2025
Front React statique	29/01/2025
Audit de projet	04/02/2025
Audit de projet	03/03/2025
Rendu du code / hébergement	21/03/2025
Démonstration finale	26/03/2025