Séance 8 : Projets Finaux Détaillés (16 Sujets pour 8 Binômes)

Chaque projet doit être développé en **Angular ou React (au choix du binôme)** et inclure obligatoirement : **TypeScript**, **Gestion d'État (Service/Context)**, **Routing** et **Interaction avec une API (CRUD)**.

Thème I : Outils et Utilitaires

#	Titre du Projet	Énoncé Détaillé	Binomes
1	**Gestionnaire de Budgets**	Créer une application pour suivre les revenus et les dépenses. Les utilisateurs doivent pouvoir **ajouter**, **modifier** et **supprimer** des transactions. L'application doit afficher un **tableau de bord** avec la **balance totale** et un **historique** des transactions filtrable par catégorie. *(Intégration API CRUD sur les transactions.)*	408 - 338
2	**Convertisseur d'Unités Avancé**	Développer un outil capable de convertir des unités dans plusieurs catégories (poids, distance, température, données numériques). Le binôme devra utiliser une **API externe** (ou simuler des données) pour les taux de conversion ou les formules. Le **routing** doit permettre des vues séparées par catégorie d'unité.	353 - Rado
3	**Planificateur d'Études (Pomodoro)**	Construire une application avec un **minuteur Pomodoro** intégré (25 min de travail, 5 min de pause). Les utilisateurs peuvent lister les tâches à accomplir et associer chaque session Pomodoro à une tâche. Le tableau de bord doit afficher le **nombre total d'heures** de travail effectuées par tâche. *(Utilisation avancée d'état et d'effets secondaires (`useEffect`/`ngOnDestroy`) pour gérer le timer.)*	334 – 391 (A)
4	**Générateur de Mots de Passe Séc. / Clés API**	Application pour générer des mots de passe personnalisables (longueur, inclusion de majuscules, nombres, symboles). Ajouter une fonctionnalité de "Coffre-fort local" (simulé par un service) pour **enregistrer** les mots de passe générés de manière sécurisée (simuler le chiffrement).	389 – 327 (B)

Thème II : Applications de Gestion de Contenu et Médias

#	Titre du Projet	Énoncé Détaillé	Binôme
5	**Catalogue de Films / Séries**	Créer une application pour rechercher des films/séries. L'application doit interroger une **API publique** (comme The Movie Database - TMDB, si autorisée) ou une API simulée. L'utilisateur doit pouvoir **sauvegarder** ses favoris dans une liste locale et afficher les **détails** de chaque film via un **routing dynamique** (`/films/:id`).	377 – 018 ©
6	**Système de Gestion de Bibliothèque**	Développer une interface pour gérer un inventaire de livres. Les fonctionnalités **CRUD** doivent permettre d'ajouter, modifier et supprimer des livres (titre, auteur, ISBN, quantité). Ajouter un **formulaire avancé** pour la modification et un **tableau filtrable** pour la liste.	363 - 376
7	**Blog Personnel Simplifié**	Créer une interface permettant de lister des articles de blog. Les utilisateurs peuvent **créer** et **éditer** leurs propres articles (titre, contenu, date). La page d'accueil liste les articles. Chaque article doit être accessible par son propre chemin d'URL via **routing**. *(Implémenter une validation de formulaire stricte.)*	007-008
8	**Album Photo / Galerie**	Application pour simuler le stockage et l'affichage d'images. Le *frontend* doit interagir avec une API pour obtenir les URL d'images (utiliser des images d'une API gratuite comme Unsplash/Pexels, ou simuler). La fonctionnalité clé est un **système de modale** ou de **vue détaillée** affichant les métadonnées de l'image.	330-433

Thème III : Commerce et Réservation

#	Titre du Projet	Énoncé Détaillé	Binôme

9	**Mini Panier d'Achat**	Simuler une petite boutique en ligne. L'application doit afficher une liste de produits. Le binôme doit implémenter un **Panier d'Achat** géré par le **Service/Context**. L'utilisateur doit pouvoir **ajouter/retirer** des produits et modifier les quantités. La page de validation doit afficher le **total TTC**.	007-008
10	**Système de Réservation de Salles**	Créer une interface permettant de visualiser et de réserver des salles. L'utilisateur sélectionne une date et une heure, et le système affiche les salles disponibles (via API). Implémenter le **CRUD** sur les réservations. La vue principale doit être un **calendrier** ou une liste d'horaires.	334 - 391
11	**Interface de Notation de Produits**	Afficher une liste de produits. Les utilisateurs (simulés) peuvent **noter** les produits (étoiles) et **ajouter des commentaires** (CRUD sur les commentaires). L'application doit calculer et afficher la **note moyenne** pour chaque produit.	353 - Rado
12	**Annuaire de Restaurants**	Développer un annuaire où les restaurants sont listés. Les utilisateurs peuvent **filtrer** par type de cuisine et rechercher par nom. Le **routing dynamique** doit afficher une page de détails pour chaque restaurant avec son adresse et sa carte (données simulées).	377 - 018

---

416 – 410 : Interface de communication avancé (appel, message, pj) / Application de signalement d’incident HSE

Thème IV : Jeux et Compétences Techniques

#	Titre du Projet	Énoncé Détaillé	Binôme
13	**Quiz Interactif**	Construire un quiz sur un thème au choix (JS, Histoire, etc.). Les questions et réponses sont gérées par une **API CRUD** (simulée). L'application doit afficher les questions une par une, enregistrer les réponses de l'utilisateur et calculer un **score final**.	330-433
14	**Jeu de Mémoire (Paires)**	Développer un jeu où l'utilisateur doit trouver des paires d'images/cartes. L'état du jeu (cartes retournées, cartes trouvées) doit être géré par un **Service/Context**. L'interface doit afficher un **compteur de coups** et le temps écoulé. *(Focus sur la manipulation d'état complexe et les effets.)*	363-376
15	**Suivi d'Objectifs (Habit Tracker)**	Application pour suivre les habitudes quotidiennes/hebdomadaires (ex: "Boire 2L d'eau", "Faire du sport"). Les utilisateurs peuvent **créer** leurs objectifs (CRUD). La vue principale doit permettre de **cocher/décocher** l'habitude pour le jour en cours et afficher une **statistique de complétion** (pourcentage).	389 – 317
16	**Tableau de Bord Météo**	Afficher la météo pour 5 villes prédéfinies en utilisant une **API Météo publique et gratuite** (ex: OpenWeatherMap, avec clé API gratuite). L'application doit afficher les prévisions sur 5 jours et permettre à l'utilisateur de **passer d'une ville à l'autre** via le routing ou une liste déroulante. *(Focus sur l'intégration API et la gestion des différents états de chargement.)*	408 - 338

Critères d'Évaluation Communs

Chaque binôme sera évalué sur :

1. **Respect des Exigences Techniques (50%) :** Utilisation correcte de TypeScript, implémentation du CRUD via API, gestion de l'état (Context/Service), et routing.
2. **Qualité du Code (25%) :** Clarté, modularité des composants, utilisation des bonnes pratiques (Hooks/Life Cycle/Directives), lisibilité.
3. **UI/UX et Déploiement (25%) :** Design cohérent (utilisation d'une librairie UI/Tailwind), accessibilité de base, et succès du déploiement (Vercel/Netlify/GitHub Pages).