Webgencia

Spécifications techniques

« Menu Maker » by **Qwenta**

Version	Auteur	Date	Approbation
1.0	Soufiane, Webgencia	28/08	John, Qwenta

I. Choix technologiques

- État des lieux des besoins fonctionnels et de leurs solutions techniques :

Besoin	**Contraintes**	**Solution**	**Description de la solution**	**Justification (2 arguments)**
Landing Page Non Connectée	L'utilisateur non connecté doit avoir accès à la page d'accueil et comprendre l'utilité du site grace au visuel.	**Frontend:** React, CSS pour animations **Backend:** Node.js, Express	**Frontend:** Utiliser React pour créer une landing page dynamique et interactive. Utiliser les keyframes CSS pour les animations sur les éléments de la page. **Backend:** Utiliser Node.js avec Express pour servir les fichiers statiques et gérer les requêtes HTTP.	**Frontend:** 1) React permet de construire des interfaces utilisateur réactives et dynamiques. 2) Les keyframes CSS ajoutent des animations qui accrochent et améliorent l'expérience utilisateur et qui attirent l'attention sur des éléments clés. **Backend:** 1) Node.js et Express fournissent un environnement performant pour servir les fichiers statiques. 2) Express permet une bonne gestion des requêtes

				HTTP.
Page login	Sécurité et accessibilité des données utilisateurs	**Frontend:** React, react-modal **Backend:** Node.js, Express, JWT, Nodemailer	**Frontend:** Utiliser React pour créer une interface utilisateur réactive et dynamique. Utiliser react-modal pour afficher la fenêtre de connexion sous forme de modale. **Backend:** Utiliser Node.js avec Express pour gérer les requêtes HTTP, JWT pour la gestion des sessions, et Nodemailer pour l'envoi d'e-mails d'authentification.	**Frontend:** 1) React permet de créer des interfaces dynamiques et interactives. 2) React-modal facilite la gestion des modales de connexion. **Backend:** 1) Node.js et Express est une plateforme robuste pour les opérations côté serveur. 2) JWT permet une gestion sécurisée des sessions et Nodemailer permet l'envoi d'e-mails sécurisés.
Création d'une catégorie de menu	L'ajout d'une catégorie doit pouvoir se faire directement sur l'écran de création de menu depuis une modale.	**Frontend:** React, react-modal **Backend:** Node.js, Express	**Frontend:** Utiliser React pour créer une interface utilisateur réactive et dynamique. Utiliser react-modal pour afficher la fenêtre d'ajout catégorie sous forme de modale.	**Frontend:** 1) Nous avons choisi de développer en React, la librairie est cohérente avec ce choix. 2) Il s'agit de la librairie la plus utilisée.

			Backend: Utiliser Node.js avec Express pour gérer les requêtes HTTP et sauvegarder les nouvelles catégories dans une base de données.	**Backend:** 1) Node.js et Express offrent une plateforme robuste pour les opérations côté serveur. 2) Express permet une gestion efficace des requêtes HTTP.
Création de plat	L'ajout d'un plat doit pouvoir se faire juste après l'ajout d'une catégorie avec possibilité d'insérer une image, un nom, un prix et une description. Plusieurs plats peuvent être insérer.	**Frontend:** React, Sharp et Muter **Backend:** Node.js, Express, Cloudinary	**Frontend:** Utiliser React avec FormData pour créer une interface utilisateur réactive et dynamique pour les formulaires. Utiliser Sharp et Multer pour la gestion des images des plats. **Backend:** Utiliser Node.js avec Express pour gérer les requêtes HTTP, et Cloudinary pour le stockage des images.	**Frontend:** 1) React avec FormData facilitent la création d'interfaces utilisateur pour les formulaires. 2) Sharp et Multer permet une gestion simple des images. **Backend:** 1) Node.js et Express offrent une plateforme éfficace pour les opérations côté serveur. 2) Cloudinary permet la gestion et le stockage des images.
Style de menu	La personnalisation du style doit pouvoir	**Frontend:** React, Styled-components,	**Frontend:** Utilisation de React pour dynamiser	**Frontend:** 1) Facilite la gestion des

	se faire avec le choix de la police, de la couleur et d'une mise en page du document.	Color PickerLibrary **Backend:** Node.js, Express	l'interface, Styled-components pour appliquer les styles de manière isolée, et une bibliothèque de sélection de couleurs pour permettre la personnalisation de la couleur du texte. **Backend:** Utiliser Node.js avec Express pour gérer les requêtes de personnalisation en temps réel.	styles dynamiques. 2) Interaction fluide et réactive. **Backend:** 1) Node.js et Express permettent une bonne gestion des requêtes de personnalisation. 2) Solution légère et rapide à implémenter.
Exportation PDF	La génération du PDF doit pouvoir se faire depuis le menu créé.	**Frontend:** jsPDF, React **Backend:** Node.js, Express	**Frontend:** Utiliser la bibliothèque jsPDF pour créer et télécharger des fichiers PDF directement depuis l'interface utilisateur React. **Backend:** Node.js avec Express pour gérer les requêtes si nécessaire pour générer des PDF dynamiquement.	**Frontend:** 1) jsPDF permet la génération de PDF côté client. 2) Solution légère et rapide à implémenter. **Backend:** 1) Node.js et Express permettent une bonne gestion des requêtes. 2) Flexibilité pour des personnalisations avancées.

Commander l'impression d'un menu	Lien direct vers le back-office de Qwenta	**Frontend:** React **Backend:** Node.js, Express	**Frontend:** Fournir un lien clairement visible sur la page d'accueil qui ouvre le système de commande d'impression dans un nouvel onglet. **Backend:** Utiliser Node.js avec Express pour gérer les requêtes et redirections vers le back-office de Qwenta.	**Frontend:** 1) Permet une transition simple et directe vers les services d'impression. 2) Facilité de commande pour les utilisateurs sans intégration complexe côté front-end. **Back-end :** 1) Node.js et Express offrent une plateforme éfficace pour les opérations côté serveur. 2) Solution légère et rapide à implémenter.
Gestion des menus précédents	Affichage efficace de multiples entrées, opérations CRUD sécurisées	**Frontend:** React **Backend:** Node.js, Express, MongoDB	**Frontend:** Utiliser React pour créer une interface utilisateur réactive permettant de visualiser, modifier et supprimer des menus. **Backend:** Utiliser Node.js avec Express pour gérer les requêtes HTTP et MongoDB pour stocker les menus précédents.	**Frontend:** 1) React permet une interaction utilisateur fluide et dynamique. 2) Facilite la gestion et la manipulation des menus. **Back-end :** 1) Node.js et Express offrent une bonne plateforme pour les opérations CRUD. 2) MongoDB permet une gestion flexible des

				données.
Informations légales	Contenu statique accessible depuis toutes les pages	**Frontend:** React, React-modal **Back-end :** Node.js, Express	**Frontend:** Utiliser une modale React pour afficher les mentions légales et l'information "Tous droits réservés" sur toutes les pages. **Backend:** Utiliser Node.js avec Express pour gérer les requêtes liées aux informations légales.	**Frontend:** 1) Une modale React permet d'accéder aux informations légales sans rechargement ou navigation supplémentaire. 2) Facilite la mise à jour des informations légales. **Backend:** 1) Node.js et Express permettent une bonne gestion des requêtes. 2) Solution légère et rapide à implémenter.
Tarifs	Ajouter un onglet "Tarifs" qui redirige les utilisateurs vers une page externe contenant les tarifs.	**Frontend:** React, Redirection URL **Backend:** Node.js, Express	**Frontend:** Créer un lien "Tarifs" visible sur la page d'accueil ou dans le menu de navigation qui redirige vers l'URL fournie par Qwenta pour les tarifs. Le lien doit ouvrir un nouvel onglet. **Backend:** Utiliser Node.js avec Express pour gérer les	**Frontend:** 1) React permet de gérer facilement les redirections et les liens dynamiques. 2) Fournir un accès facile et direct aux informations tarifaires améliore l'expérience utilisateur. **Back-end :** 1) Node.js et Express permettent une bonne

			redirections.	gestion des redirections. 2) Solution légère et rapide à implémenter.
Exportation Deliveroo	Interaction avec une API externe	**Frontend:** React, **Back-end :** Node.js, Express	**Front-end :** Utilisation de React pour l'interface utilisateur permettant de déclencher l'exportation **Back-end :** Intégration avec l'API Deliveroo pour envoyer les données du menu. Redirection de l'utilisateur vers l'application Deliveroo pour finaliser l'opération.	1) Utiliser l'API Deliveroo permet une intégration fluide et directe. 2) La redirection garantit que l'utilisateur peut immédiatement vérifier et modifier le menu directement dans Deliveroo.
Partage sur Instagram	Connexion avec API Instagram, traitement d'image	**Frontend:** React, **Back-end :** Node.js, Express, Sharp, Cloudinary	**Front-end :** Utilisation de React pour l'interface utilisateur permettant de déclencher le partage. **Back-end :** Utilisation de Sharp pour traiter les images en format carré, et Cloudinary pour le stockage et la gestion des images. Intégration	1) Sharp permet de manipuler des images côté serveur. Cloudinary offre une gestion simplifiée des ressources média. 2) L'API Instagram permet le partage direct sur les réseaux sociaux.

			avec l'API Instagram pour faciliter le partage.	
Déconnexion	Accessibilité sur toutes les pages connectées	**Frontend:** React **Back-end :** Node.js, JWT	**Frontend:** Implémenter un bouton de déconnexion qui, lorsqu'activé, efface le JWT (JSON Web Token) côté client et met à jour l'état de l'application pour refléter le statut non connecté de l'utilisateur. **Backend:** Utiliser Node.js avec JWT pour la gestion des sessions utilisateurs.	**Frontend:** 1) React permet une gestion d'état réactive, facilitant la mise à jour instantanée de l'interface utilisateur lors de la déconnexion. 2) Solution légère et rapide à implémenter. **Back-end :** 1) Utiliser JWT pour la gestion de session sécurise les sessions utilisateurs. 2) Node.js offre une plateforme robuste pour les opérations côté serveur.
Modification des infos utilisateur	Sécurité des données, gestion des identifiants multiples	**Frontend:** React **Back-end :** Node.js, Express, MongoDB	**Frontend:** Utiliser React pour permettre la modification des informations utilisateur. **Back-end :** Utiliser Node.js avec Express pour gérer les requêtes,	1) React facilite la création d'interfaces utilisateur dynamiques pour la modification des données. 2) MongoDB permet une gestion sécurisée et flexible des données utilisateur.

			et MongoDB pour stocker les données utilisateur de manière sécurisée.	
Dashboard	Accessibilité et intégration de contenus divers	**Frontend:** React **Back-end :** Node.js, Express, RSS/API	**Front-end :** Utiliser React pour développer le dashboard. **Back-end :** Intégrer des APIs pour les interactions (création, diffusion, impression de menus) et utiliser RSS ou une API pour intégrer les articles de blog.	1) React permet une intégration fluide des différentes fonctionnalités sur un seul dashboard. 2) Les APIs assurent que les actions comme créer, diffuser et imprimer sont gérées correctement.
Création de branding pour le restaurant	Facilité d'utilisation, accessibilité, stockage sécurisé des médias	**Frontend:** React, Styled-components **Back-end :** Node.js, Express, Cloudinary	**Front-end :** Utiliser React pour la gestion de l'interface utilisateur, et Styled-components pour appliquer les couleurs de base. **Back-end :** Utiliser Cloudinary pour le stockage et la manipulation des logos.	1) React et Styled-components offrent une expérience utilisateur fluide et dynamique pour la personnalisation. 2) Cloudinary gère les fichiers médias en toute sécurité.

II. Liens avec le back-end

Langage pour le serveur :

- **Node.js avec Express** : Node.js a été sélectionné pour ses performances et sa capacité à gérer un grand nombre de requêtes simultanément grâce à son architecture événementielle. Express, un framework léger pour Node.js, simplifie la création de routes API RESTful. Ce duo garantit une structure de code claire et une intégration fluide avec le front-end.

Base de données choisie :

- **MongoDB** : MongoDB, une base de données NoSQL, offre une gestion flexible et dynamique des données. Elle est idéale pour des informations, telles que les menus, qui peuvent évoluer en termes de structure et de volume. MongoDB stocke les données sous forme de documents JSON, ce qui est parfaitement compatible avec des applications JavaScript comme celles basées sur Node.js et React.

III. Préconisations concernant le domaine et l'hébergement

- **Nom de domaine** : qwenta.fr et qwenta.com
- **Hébergement** : Utiliser Amazon Web Services (AWS) est recommandé en raison de son infrastructure fiable, capable de gérer facilement des hausses de trafic et de volume de données. AWS propose également une large gamme de services pour la gestion des serveurs, la sécurité et les bases de données.

IV. Accessibilité

- **Compatibilité navigateur** : Support des versions récentes de Chrome, Safari et Firefox. Le projet étant focalisé sur les performances desktop, aucune version mobile n'est prévue pour le moment.

- **Accessibilité** : Conformité aux standards WCAG (Web Content Accessibility Guidelines) pour garantir l'accessibilité, notamment via la navigation au clavier et la compatibilité avec les lecteurs d'écran.

V. Recommandations en termes de sécurité

- **Sécurité des comptes utilisateurs** : Pour assurer la protection des comptes, utiliser des jetons JWT et des e-mails de vérification. Ces jetons authentifient de manière sécurisée les utilisateurs et garantissent que seuls les utilisateurs autorisés peuvent accéder aux sections protégées. Les e-mails de vérification valident l'authenticité des adresses e-mail.
- **Protection des API** : Les API qui relient le front-end et le back-end doivent être sécurisées pour éviter les attaques. Les jetons d'authentification sont utilisés pour restreindre l'accès aux utilisateurs autorisés et les données saisies doivent être validées afin de prévenir toute injection de code malveillant.
- **Utilisation des API externes** : Pour interagir avec des services tiers, des jetons API fournis par ces services sont utilisés pour authentifier les requêtes. Ces jetons garantissent que seules les entités autorisées peuvent accéder aux services externes. Il est essentiel de stocker ces jetons de manière sécurisée.

VI. Maintenance du site et futures mises à jour

- **Mises à jour régulières des dépendances (React, Node.js, MongoDB).**
- **Sauvegardes automatiques** de la base de données et des fichiers médias.
- **Surveillance des performances** via AWS CloudWatch.
- **Mises à jour de sécurité** pour les technologies critiques.
- **Suivi des erreurs** et des logs pour détecter et corriger les problèmes.