تقييم المحاضر

تم تقييم المشروع بشكل إيجابي، حيث حصل على إشادة من المحاضر من حيث جودة التنفيذ، تجربة المستخدم، والالتزام بالمواصفات المطلوبة. كان الأداء العام ممتازًا، ولم تُسجل ملاحظات سوى على صفحات تسجيل الدخول (Sign In) والتسجيل (Sign Up) التي تحتاج إلى بعض التعديلات.

التحسينات المقترحة لصفحات Sign In وSign Up

التأكد من توافق التصميم مع باقي أجزاء المشروع.

تحسين تجربة المستخدم من خلال إضافة تلميحات أو تحسين واجهة الإدخال.

التحقق من سهولة استخدام النموذج عبر الأجهزة المختلفة.

التأكد من أن التحقق من صحة البيانات (Validation) يعمل بشكل مثالي.

تحسين مظهر الحقول وأزرار التسجيل لزيادة التفاعل وسهولة الاستخدام.

---

شرح طريقة عمل تطبيق الدردشة (Chati App) باستخدام React وNode.js

Chati App هو تطبيق دردشة حديث ومتقدم مبني باستخدام أحدث التقنيات، ويدعم جميع الميزات التي يحتاجها المستخدمون لتجربة تواصل سلسة وآمنة. دعونا نتعمق في طريقة عمله.

1. التقنيات المستخدمة في التطبيق

الواجهة الأمامية React.js (Frontend)

React.js: لإنشاء واجهة مستخدم ديناميكية وسريعة.

Redux: لإدارة حالة التطبيق بطريقة سلسة ومنظمة.

Supabase: لتخزين الصور والصوتيات وإدارة ملفات المستخدمين.

CSS / Tailwind: لتنسيق وتصميم الواجهة بشكل متجاوب وجذاب.

الواجهة الخلفية Node.js (Backend) وSocket.IO

Node.js: لبناء الخادم الخلفي القوي.

Express.js: لإنشاء API والتعامل مع الطلبات بسهولة.

Socket.IO: لدعم المراسلة الفورية في الوقت الحقيقي.

MongoDB: لتخزين بيانات المستخدمين والدردشات بشكل آمن وفعال.

JWT (JSON Web Token): لإدارة تسجيل الدخول وتأمين الجلسات.

---

2. كيف يعمل التطبيق؟

نظام المصادقة (Authentication)

عند تسجيل المستخدم، يتم تشفير كلمة المرور باستخدام bcrypt.

عند تسجيل الدخول، يتم التحقق من صحة بيانات المستخدم وإرسال JWT Token إليه لتأكيد الجلسة.

يدعم التطبيق التحقق عبر البريد الإلكتروني (OTP) لضمان أمان الحسابات.

نظام الدردشة الفورية باستخدام Socket.IO

1. إنشاء الاتصال:

عندما يسجل المستخدم الدخول، يتم إنشاء اتصال WebSocket بينه وبين الخادم عبر Socket.IO.

يتم تخزين معرف الـ Socket لكل مستخدم لمتابعة حالته (متصل / غير متصل).

2. إرسال واستقبال الرسائل:

عندما يرسل المستخدم رسالة، يتم تمريرها عبر Socket.IO إلى الخادم.

يقوم الخادم بحفظ الرسالة في MongoDB ثم يعيد إرسالها إلى المستخدمين الآخرين عبر الـ WebSocket.

يدعم التطبيق إرسال النصوص، الصور، الفيديوهات، الملفات، والرسائل الصوتية.

3. إشعارات الكتابة (Typing Indicator):

عند بدء الكتابة، يرسل المتصفح حدث "userTyping" إلى الخادم.

يقوم الخادم بإرسال هذا الحدث إلى الطرف الآخر لعرض مؤشر "يكتب الآن...".

4. عرض حالة الاتصال (Online / Offline):

عند تسجيل الدخول، يتم تحديث الحالة إلى متصل.

عند تسجيل الخروج أو فقدان الاتصال، يتم تحديثها إلى غير متصل.

دعم الوسائط والملفات:

مشاركة الصور والفيديوهات باستخدام Supabase Storage.

مشاركة الملفات والمستندات داخل الدردشة.

إرسال واستقبال رسائل صوتية يتم تسجيلها مباشرة من التطبيق.

تحسين تجربة المستخدم (UX):

✅ دعم الوضع الليلي والفاتح (Light & Dark Mode).

✅ تصميم متجاوب يعمل على الجوال، التابلت، والكمبيوتر.

✅ تحديث ملف المستخدم بسهولة، مع إمكانية تغيير الصورة الشخصية عبر Supabase.

✅ إثراء الروابط داخل المحادثة، بحيث يتم عرض معاينة للروابط المشتركة تلقائيًا.

✅ تكامل مع Giphy لإرسال صور متحركة وإيموجي بسهولة.

---

3. كيف يتم نشر التطبيق؟

الخطوات التقنية لنشر المشروع

1. نشر الواجهة الخلفية (Backend)

   يتم نشر Node.js وMongoDB على VPS أو Firebase أو Render.

   استخدام Nginx أو PM2 لإدارة الخادم وضمان تشغيله المستمر.

2. نشر الواجهة الأمامية (Frontend)

   يتم نشر React.js عبر Vercel أو Netlify أو Firebase Hosting.

   يتم ضبط الاتصال بين الواجهة والخادم باستخدام Axios للتعامل مع الـ APIs.

---

ملخص:

يعتمد التطبيق على React.js، Node.js، MongoDB، وSocket.IO.

يستخدم JWT Authentication لتأمين الحسابات.

يدعم المراسلة الفورية وإرسال الصور، الفيديوهات، الملفات، والرسائل الصوتية.

يتم تخزين الوسائط عبر Supabase لضمان سرعة وأمان التخزين.

يدعم التحديث المباشر لحالة الاتصال وإشعارات الكتابة باستخدام WebSockets.

يمكن نشره بسهولة على أي سيرفر سحابي مثل Firebase أو Vercel.