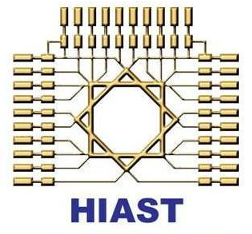
**الجمهورية العربية السورية**

**المعهد العالي للعلوم التطبيقية والتكنولوجيا**

**قسم المعلوميات**

**العام الدراسي 2024/2025**

**مشروع سنة رابعة**

تطبيق دردشة الند للند آمن

**إعداد**

**تمّام علي السليمان**

**إشراف**

**د.سميح جمول**

**م.مرح حسن**

**الملخص**

مع تزايد الاعتماد على تطبيقات التواصل الفوري في الحياة اليومية، برزت الحاجة إلى منصات تقدم تجربة مستخدم غنية ومستقرة. يقدم هذا المشروع تطبيق محادثة متكامل تم تصميمه ليكون فعّال وذا أداء عالٍ، حيث يتيح للمستخدمين التواصل الفعّال عبر الرسائل النصية الفورية عن طريق مكتبة Socket.io، بالإضافة إلى إجراء مكالمات الفيديو والصوت بجودة عالية اعتماداً على بروتوكول WebRTC الذي يؤسس اتصالاً مباشراً بين المستخدمين. تم بناء هذا المشروع بالاعتماد على منهجيات تصميم حديثة تضمن فصل الاهتمامات وتعزز قابلية الصيانة. حيث تم اعتماد نمط تصميم معماري يفصل منطق العرض عن منطق الأعمال في الواجهة الأمامية، بينما تم بناء الواجهة الخلفية وفق هيكلية معيارية نمطية.

**Abstract**

With the increasing reliance on instant communication applications in daily life, the need for platforms that offer a rich and stable user experience has become paramount. This project introduces an integrated chat application engineered for efficiency and high performance. It enables users to communicate effectively through instant text messages by leveraging the Socket.io library, in addition to conducting high-quality video and audio calls based on the WebRTC protocol, which establishes a direct peer-to-peer connection between users. The project was built using modern design methodologies that ensure separation of concerns and enhance maintainability. An architectural pattern was adopted to separate the presentation logic from the business logic on the front-end, while the back-end was constructed following a modular architecture.

# **الفصل الأول**

# **التعريف بالمشروع**

يتضمن هذا الفصل التعريف بالمشروع ومتطلباته.

## **مقدمة :**

في عصر الاتصالات الرقمية، أصبحت تطبيقات المحادثة الفورية جزءاً لا يتجزأ من حياتنا اليومية، مما خلق طلباً متزايداً على منصات لا تقدم فقط رسائل نصية، بل تجربة تواصل متكاملة وغنية. يهدف هذا المشروع إلى تطوير تطبيق محادثة فوري حديث للأجهزة المحمولة، مصمم ليكون فعال وذا أداء عالٍ، مع التركيز على تقديم تجربة مستخدم سلسة ومستقرة.

يتميز التطبيق بدعمه لمجموعة واسعة من وظائف التواصل، فبالإضافة إلى المحادثات النصية الفورية، يدعم التطبيق مكالمات الفيديو والصوت عالية الجودة في الزمن الحقيقي . (Real-time) تم تحقيق ذلك بالاعتماد على بروتوكول WebRTC الذي يؤسس اتصالاً مباشراً بين المستخدمين (Peer-to-Peer)، مما يقلل من زمن الاستجابة (Latency) ويضمن جودة مكالمات ممتازة حتى في ظروف الشبكة المتغيرة.

لضمان أداء عالٍ وقابلية للتوسع، تم بناء التطبيق بهيكلية تقنية حديثة. تم تطوير الواجهة الأمامية (Frontend) باستخدام إطار العمل Flutter، مما يتيح تقديم تجربة مستخدم موحدة وجذابة على كل من أنظمة Android و iOS . أما الواجهة الخلفية (Backend)، فقد تم بناؤها باستخدام Node.js لتميزها في إدارة الاتصالات المتزامنة بكفاءة. تم استخدام مكتبة Socket.io لإنشاء قناة اتصال ثابتة وموثوقة لإدارة الرسائل النصية، ونقل الملفات، وتحديثات الحضور الفورية، بينما تم وضع الأمان والخصوصية كأحد الركائز الأساسية في تصميم النظام.

## **هدف المشروع :**

يهدف المشروع إلى بناء تطبيق محادثة فوري ومتكامل للأجهزة المحمولة، يدعم الرسائل النصية ومكالمات الفيديو والصوت عالية الجودة عبر بروتوكول WebRTC. تم تطوير التطبيق كحل برمجي موحد يعمل على نظامي Android و iOS باستخدام Flutter، مع التركيز على تطبيق معماريات حديثة وقابلة للتطوير والتوسع مثل نمط MVVM في الواجهة الأمامية وهيكلية معيارية في الواجهة الخلفية المبنية بـ Node.js. كما ويوفر نظاماً لإدارة الحضور الفوري باستخدام Socket.io.

## **المتطلبات الوظيفية:**

1. إدارة المستخدمين :

* يجب أن يتيح النظام لكل مستخدم إنشاء حساب مستخدم جديد.
* يجب أن يتيح النظام لكل مستخدم الدخول بحسابه الذي قام بإنشائه مسبقاً.
* يجب أن يقوم النظام بتوثيق المستخدمين بأمان عند تسجيل الدخول.
* يجب أن يتمكن المستخدمون الموثقون من تعيين اسم عرض.

1. الدردشة النصية :

* يجب أن يتمكن المستخدمون من إرسال رسائل نصية .
* يجب أن يحافظ النظام على سجل تاريخي للرسائل لكل محادثة في قاعدة البيانات حتى يتمكن المستخدمون من استرجاع الرسائل السابقة.
* يجب أن يتيح النظام للمستخدم إنشاء مجموعة ودعوة جهات اتصال متعددة.
* يجب ان يتيح النظام للمستخدم إضافة مستخدم آخر الى مجموعة موجودة مسبقاً.
* يجب ان يتيح النظام عملية بحث حسب اسم العرض لإيجاد جهة اتصال معينة.

1. مكالمات الصوت والفيديو :

* يجب أن يتمكن المستخدم من بدء مكالمة صوتية/مرئية مع مستخدم آخر محدد من قائمة المستخدمين المتصلين.
* يجب أن يتمكن المستخدم من قبول أو رفض مكالمة واردة.
* يجب ان يتمكن المستخدم من انهاء المكالمة في أي وقت.
* القدرة على كتم/إلغاء كتم الميكروفون أثناء المكالمة.
* القدرة على تشغيل/إيقاف تشغيل الكاميرا (الفيديو) أثناء المكالمة.
* القدرة على التبديل بين الكاميرا الأمامية والخلفية.

## **المتطلبات غير الوظيفية:**

1. الأداء:

* يجب ان يؤمّن النظام وثوقية عالية في تلبية طلبات المستخدمين.
* يجب ان يكون التأخير في نقل الصوت والفيديو في المكالمات صغير نسبياً.

1. الموثوقية والاستقرار:

* يجب ألا يتعطل التطبيق أو يتوقف عن الاستجابة بشكل غير متوقع أثناء الاستخدام العادي أو أثناء المكالمات.

1. قابلية الاستخدام:

* يجب أن يكون تعلم استخدام الوظائف الأساسية سهلاً وبديهياً للمستخدم الجديد.
* يجب أن تكون جميع عناصر التحكم واضحة ومفهومة.

1. الأمان والمصادقة:

* يتم التحقق من هوية المستخدم عند الدخول الى التطبيق.
* يتم تشفير جميع البيانات التي يتم نقلها من جانب المرسل ويتم فك تشفيرها في جانب المستلم.

1. قابلية الصيانة والتطوير:

* يتم تنظيم كود التطبيق وفق بنية معمارية مناسبة لأعمال الصيانة والتطوير وتسهل عملية الفهم والتصحيح.

1. واجهة التطبيق :

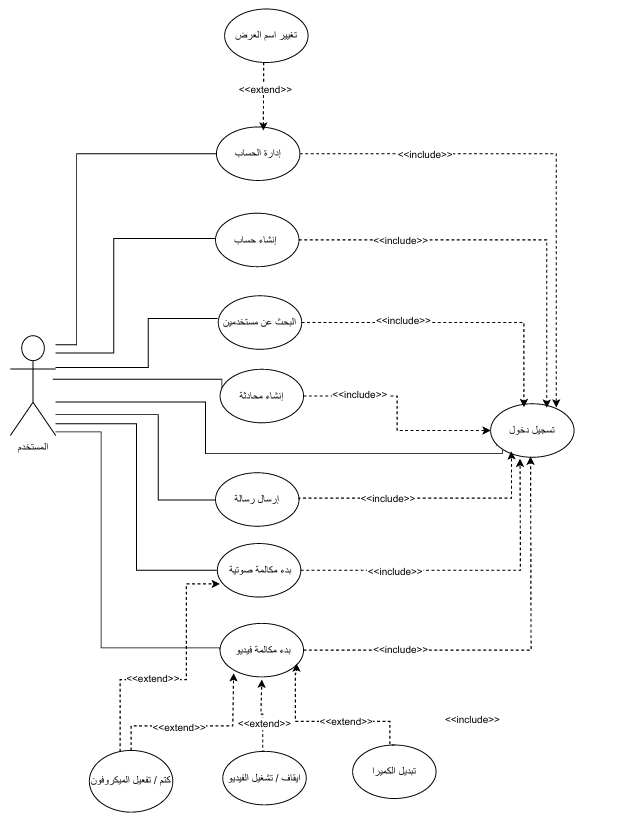
* يُبنى التطبيق كتطبيق موبايل.
* يجب أن يوفر التطبيق واجهة مستخدم واضحة وسهلة التصفح.
* يجب أن يعرض التطبيق رسائل مناسبة للمستخدم عند حدوث أخطاء أو حالات خاصة (مثل: كلمة المرور غير صحيحة ، تحقق من اتصالك بالإنترنت).

# **الفصل الثاني**

# **الدراسة التحليلية**

يوضح هذا الفصل عملية تحليل النظام ودراسة متطلباته وصولاً لتصميم النظام.

## مخططات حالات الاستخدام

* **حالات استخدام المستخدم :**

## السرد النصي لحالات استخدام المستخدم

### **إنشاء حساب**

|  |  |  |
| --- | --- | --- |
| اسم الحالة: إنشاء حساب | | |
| الوصف Description | يقوم المستخدم بإنشاء حساب جديد خاص به. |
| الفاعلين Actors | المستخدم |
| الشروط السابقة Precondition | لا يوجد. |
| الشروط اللاحقة Postcondition | تم إنشاء حساب جديد خاص بالمستخدم ضمن النظام والمستخدم في حالة تسجيل دخول. |

* **سير الاحداث**
* **السيناريو الأساسي الناجح**

|  |  |
| --- | --- |
| النظام | المستخدم |
|  | 1. يطلب إنشاء حساب جديد. |
| 1. يطلب النظام المعلومات التالية:  * اسم الحساب (Username). * اسم العرض (Displyname). * كلمة المرور (Password) |  |
|  | 1. يحدد المستخدم المعلومات المطلوبة ويطلب تأكيد العملية. |
| 1. يتحقق النظام من المعلومات المدخلة ويوجه المستخدم الى الصفحة الرئيسية. |  |

* **المسارات البديلة**

لا يوجد.

* **مسارات الأخطاء**

**E1:** في المرحلة رقم (4) في حال تحققت إحدى الحالات التالية :

* اسم المستخدم يطابق اسم مستخدم موجود سابقاً.
* كلمة المرور اقل من 6 محارف.
* اسم المستخدم او اسم العرض اقل من 3 محارف .

يتم استبدال الخطوة الرابعة في السيناريو الأساسي بالخطوة التالية:

يعيد النظام رسالة توضّح سبب الخطأ ويَطلب تحديد المعلومات مِن جديد.

### **تسجيل دخول**

|  |  |  |
| --- | --- | --- |
| اسم الحالة: تسجيل دخول | | |
| الوصف Description | يقوم المستخدم بتسجيل الدخول بحساب يمتلكه مسبقاً. |
| الفاعلين Actors | المستخدم |
| الشروط السابقة Precondition | ان يكون لديه حساب مُنشأ مسبقاً. |
| الشروط اللاحقة Postcondition | تم الدخول الى الحساب وتم اظهار الصفحة الرئيسية. |

* **سير الاحداث**
* **السيناريو الأساسي الناجح**

|  |  |
| --- | --- |
| النظام | المستخدم |
|  | 1. يدخل الى التطبيق ليقوم بتسجيل دخول. |
| 1. يطلب النظام المعلومات التالية:  * اسم الحساب (Username). * كلمة المرور (Password) |  |
|  | 1. يحدد المستخدم المعلومات المطلوبة ويضغط على زر تسجيل الدخول. |
| 1. يتحقق النظام من المعلومات المدخلة ويوجه المستخدم الى الصفحة الرئيسية. |  |

* **المسارات البديلة**

لا يوجد.

* **مسارات الأخطاء**

**E1:** في المرحلة رقم (4) في حال تحققت إحدى الحالات التالية :

* اسم المستخدم لا يطابق اسم مستخدم موجود سابقاً.
* كلمة المرور غير صحيحة.
* فقد الاتصال بالإنترنت.

يتم استبدال الخطوة الرابعة في السيناريو الأساسي بالخطوة التالية:

يعيد النظام رسالة توضّح سبب الخطأ ويطلب تحديد المعلومات مِن جديد.

### **إنشاء محادثة**

|  |  |  |
| --- | --- | --- |
| اسم الحالة: إنشاء محادثة | | |
| الوصف Description | يقوم المستخدم بإنشاء محادثة مع الاخرين |
| الفاعلين Actors | المستخدم |
| الشروط السابقة Precondition | المستخدم مسجل ضمن التطبيق . |
| الشروط اللاحقة Postcondition | تم إنشاء محادثة جديدة تحتوي على الأعضاء الذين تم تحديدهم، وتم عرض واجهة المحادثة. |

* **سير الاحداث**
* **السيناريو الأساسي الناجح**

|  |  |
| --- | --- |
| النظام | المستخدم |
|  | 1. يطلب إنشاء محادثة جديدة. |
| 1. يطلب تحديد أعضاء المحادثة. |  |
|  | 1. يقوم بالبحث عن الأعضاء المحددين عن طريق مربع بحث ، ويقوم بتحديدهم. |
|  | 1. يقوم بالضغط على زر تأكيد عملية الانشاء. |
| 1. ينشئ المجموعة ويوجه المستخدم الى واجهة المحادثة الجديدة. |  |

* **المسارات البديلة**

P1: في المرحلة رقم (4) في حال تراجع المستخدم عن انشاء المحادثة ، يتم استبدال المرحلة رقم (5) بالخطوة التالية :

يتجاهل النظام أي تحديدات قام بها المستخدم ، ويغلق الشاشة دون ان يتغير شيء.

* **مسارات الأخطاء**

E1: في المرحلة رقم (3) في حال قام بالبحث عم اسم مستخدم غير موجود يتم استبدال المرحلة ال(4) و ال(5) بالتالي:

يقوم النظام بإظهار عبارة لا يوجد أعضاء مطابقين ويتيح إعادة العملية من المرحلة (3).

### **البحث عن مستخدمين**

|  |  |  |
| --- | --- | --- |
| اسم الحالة: إنشاء محادثة | | |
| الوصف Description | يقوم المستخدم بالبحث عن المستخدمين ضمن التطبيق حسب اسم العرض. |
| الفاعلين Actors | المستخدم |
| الشروط السابقة Precondition | المستخدم مسجل ضمن التطبيق . |
| الشروط اللاحقة Postcondition | اظهار قائمة من الأشخاص التي تطابق أسماؤهم مع الاسم في خانة البحث. |

* **سير الاحداث**
* **السيناريو الأساسي الناجح**

|  |  |
| --- | --- |
| النظام | المستخدم |
|  | 1. يضغط على زر البحث . |
| 1. يطلب ادخال اسم عرض للبحث. |  |
|  | 1. يقوم بإدخال اسم عرض معين للبحث عنه. |
| 1. يقوم بإعادة قائمة من الأشخاص التي تطابق أسماؤهم مع الاسم في خانة البحث. |  |

* **المسارات البديلة**

لا يوجد.

* **مسارات الأخطاء**

E1: في المرحلة رقم (3) في حال قام بالبحث عم اسم مستخدم غير موجود يتم استبدال المرحلة ال(4) بالتالي:

يقوم النظام بإظهار عبارة لا يوجد أعضاء مطابقين ويتيح إعادة العملية من المرحلة (3).

### **إرسال رسالة**

|  |  |  |
| --- | --- | --- |
| اسم الحالة: إرسال رسالة | | |
| الوصف Description | يقوم المستخدم بإرسال رسالة ضمن محادثة معينة. |
| الفاعلين Actors | المستخدم. |
| الشروط السابقة Precondition | المستخدم مسجل ضمن التطبيق، والمستخدم ضمن المحادثة الموجودة بشكل مسبق. |
| الشروط اللاحقة Postcondition | تم ارسال الرسالة واظهارها في المحادثة. |

* **سير الاحداث**
* **السيناريو الأساسي الناجح**

|  |  |
| --- | --- |
| النظام | المستخدم |
|  | 1. يقوم بالنقر على خانة الكتابة ضمن المحادثة. |
|  | 1. يقوم بكتابة الرسالة المراد إرسالها. |
|  | 1. يقوم بالضغط على زر الارسال. |
| 1. يضيف الرسالة الى المحادثة مع علامة انتظار اتمام الارسال. |  |
| 1. يتحقق من اتصال المستخدم بالإنترنت. |  |
| 1. يرسل الرسالة الى الأعضاء ضمن المحادثة ويقوم بتخزين الرسالة في قاعدة المعيطات. |  |
| 1. يقوم بتغيير حالة الرسالة من انتظار الى تم الارسال. |  |

* **المسارات البديلة**

لا يوجد.

* **مسارات الأخطاء**

E1: في المرحلة رقم (5) في حال كان المستخدم غير متصل بالإنترنت يتم استبدال المرحلة رقم (6) بالتالي:

يتم اظهار رسالة تبين ان على المستخدم ان يتحقق من الاتصال بالإنترنت ، ويقوم بإعادة المحاولة عند الاتصال ويتابع من المرحلة (6).

### **بدء مكالمة فيديو**

|  |  |  |
| --- | --- | --- |
| اسم الحالة: إرسال رسالة | | |
| الوصف Description | يقوم المستخدم بطلب بدء مكالمة فيديو مع مستخدم اخر. |
| الفاعلين Actors | المستخدم. |
| الشروط السابقة Precondition | المستخدم مسجل ضمن التطبيق، والمستخدم الاخر متصل على التطبيق. |
| الشروط اللاحقة Postcondition | تم بدء المكالمة المطلوبة ، وتم عرض الفيديو الخاص بالطرفين. |

* **سير الاحداث**
* **السيناريو الأساسي الناجح**

|  |  |
| --- | --- |
| النظام | المستخدم |
|  | 1. يقوم بالنقر على ايقونة المستخدم الاخر التي تظهر ضمن قائمة الأشخاص المتصلين. |
| 1. يقوم بإظهار المستخدم الذي تم النقر عليه ضمن قائمة الاتصال. |  |
|  | 1. يقوم بالنقر على زر اتصال الفيديو. |
| 1. يتحقق من اتصال المستخدم بالإنترنت. |  |
| 1. يظهر شاشة تعبر عن انتظار المستخدم الاخر قبول المكالمة. |  |
| 1. بعد قبول المكالمة من الطرف الاخر يتم عرض الفيديو الخاص بالطرفين. |  |

* **المسارات البديلة**

في المرحلة رقم (6) في حال رفض الطرف الاخر المكالمة يتم استبدال المرحلة (6) بالتالي :

يخرج من حالة الانتظار ويعود الى صفحة الاتصال.

* **مسارات الأخطاء**

E1: في المرحلة رقم (4) في حال كان المستخدم غير متصل بالإنترنت يتم استبدال المرحلة رقم (5) بالتالي:

يتم اظهار رسالة تبين ان على المستخدم ان يتحقق من الاتصال بالإنترنت، ويقوم بإعادة المحاولة عند الاتصال من المرحلة رقم (3).

### **بدء مكالمة صوتية**

|  |  |  |
| --- | --- | --- |
| اسم الحالة: إرسال رسالة | | |
| الوصف Description | يقوم المستخدم بطلب بدء مكالمة صوتية مع مستخدم اخر. |
| الفاعلين Actors | المستخدم. |
| الشروط السابقة Precondition | المستخدم مسجل ضمن التطبيق، والمستخدم الاخر متصل على التطبيق. |
| الشروط اللاحقة Postcondition | تم بدء المكالمة المطلوبة، ويتم تبادل الصوت بين الطرفين. |

* **سير الاحداث**
* **السيناريو الأساسي الناجح**

|  |  |
| --- | --- |
| النظام | المستخدم |
|  | 1. يقوم بالنقر على ايقونة المستخدم الاخر التي تظهر ضمن قائمة الأشخاص المتصلين. |
| 1. يقوم بإظهار المستخدم الذي تم النقر عليه ضمن قائمة الاتصال . |  |
|  | 1. يقوم بالنقر على زر اتصال الصوت. |
| 1. يتحقق من اتصال المستخدم بالإنترنت. |  |
| 1. يظهر شاشة تعبر عن انتظار المستخدم الاخر قبول المكالمة. |  |
| 1. بعد قبول المكالمة من الطرف الاخر يتم تبادل الصوت بين الطرفين. |  |

* **المسارات البديلة**

في المرحلة رقم (6) في حال رفض الطرف الاخر المكالمة يتم استبدال المرحلة (6) بالتالي:

يخرج من حالة الانتظار ويعود الى صفحة الاتصال.

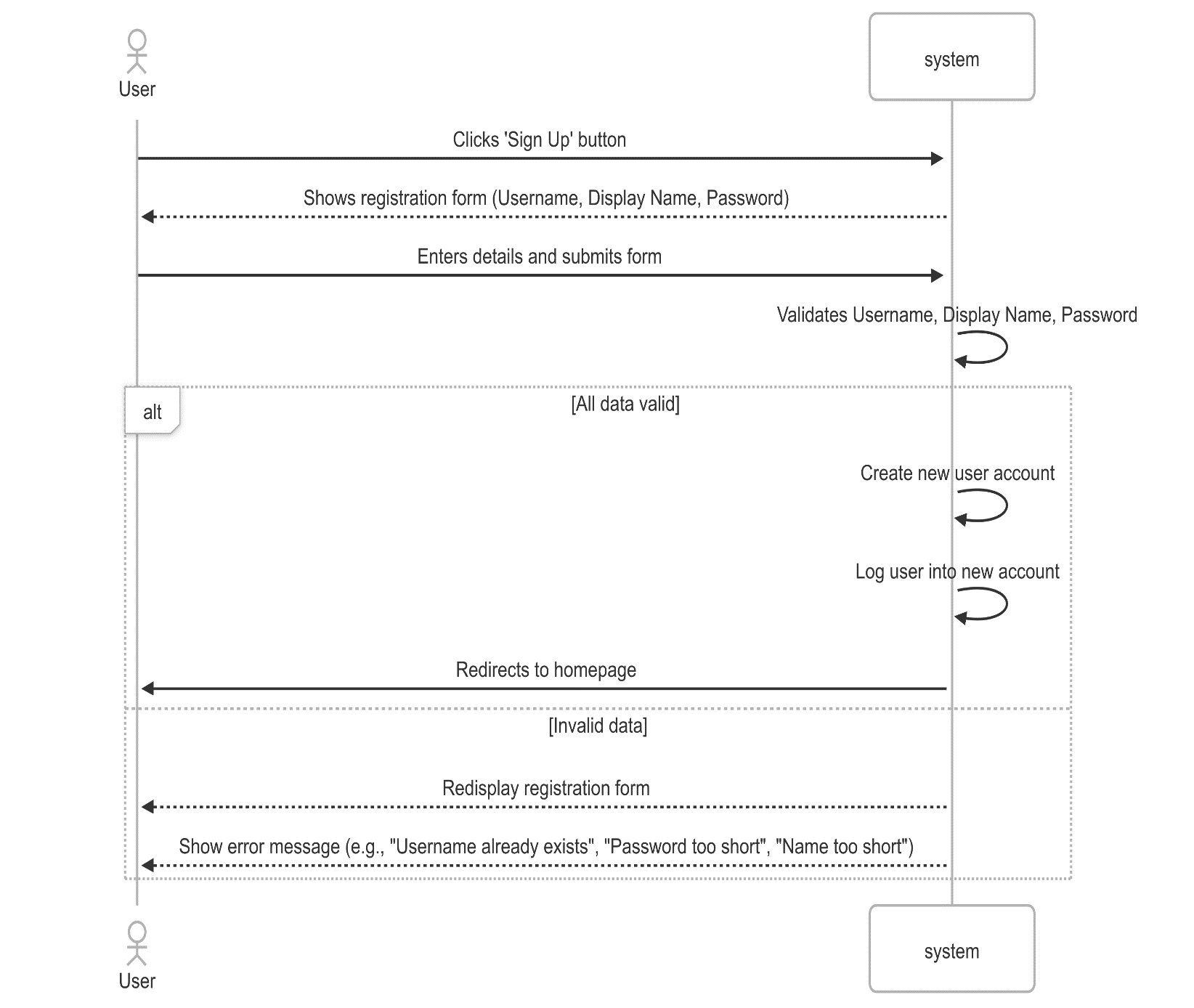
* **مسارات الأخطاء**

E1: في المرحلة رقم (4) في حال كان المستخدم غير متصل بالإنترنت يتم استبدال المرحلة رقم (5) بالتالي:

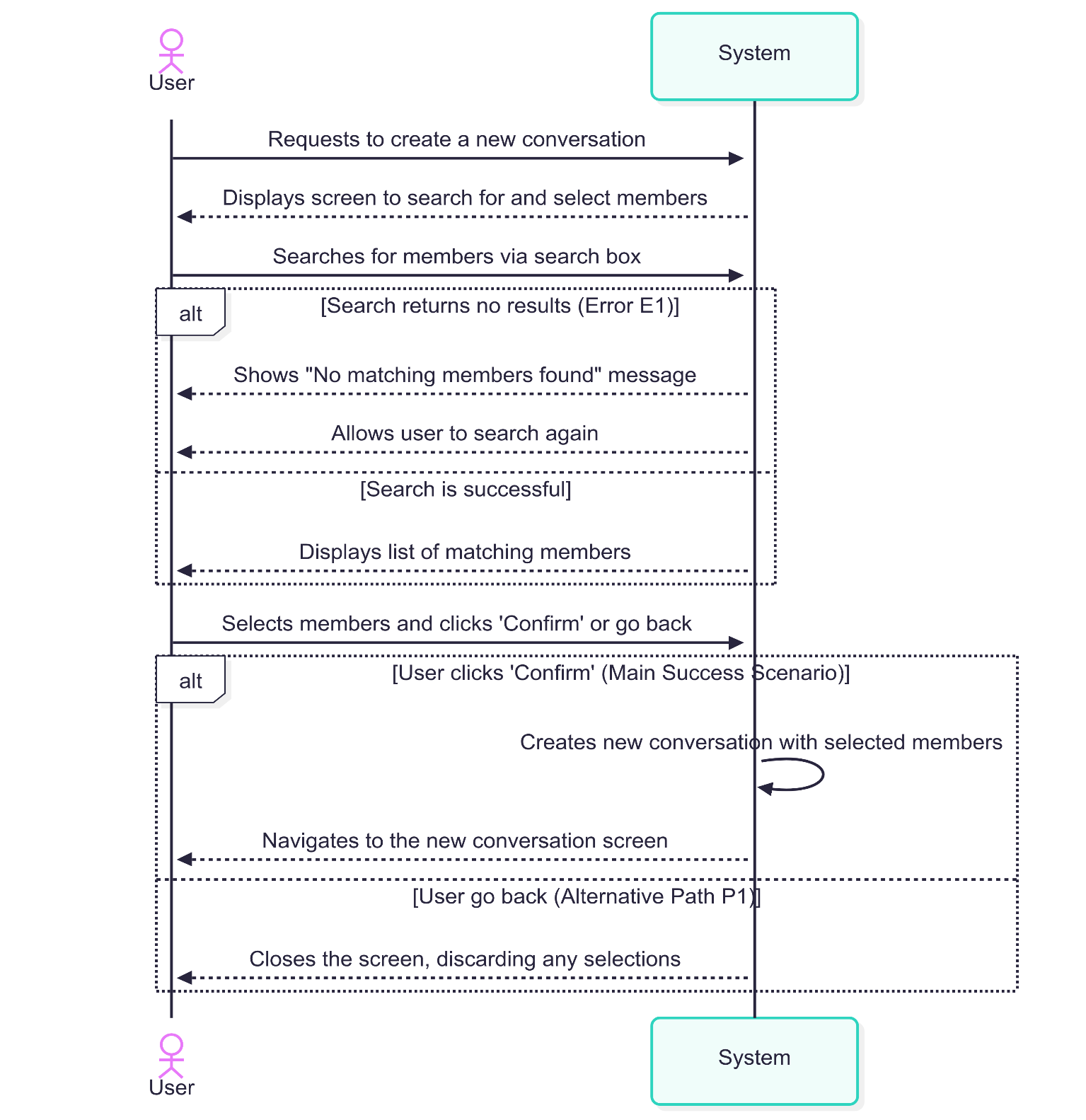
يتم اظهار رسالة تبين ان على المستخدم ان يتحقق من الاتصال بالإنترنت، ويقوم بإعادة المحاولة عند الاتصال من المرحلة رقم (3).

## مخططات تسلسل النظام SSD

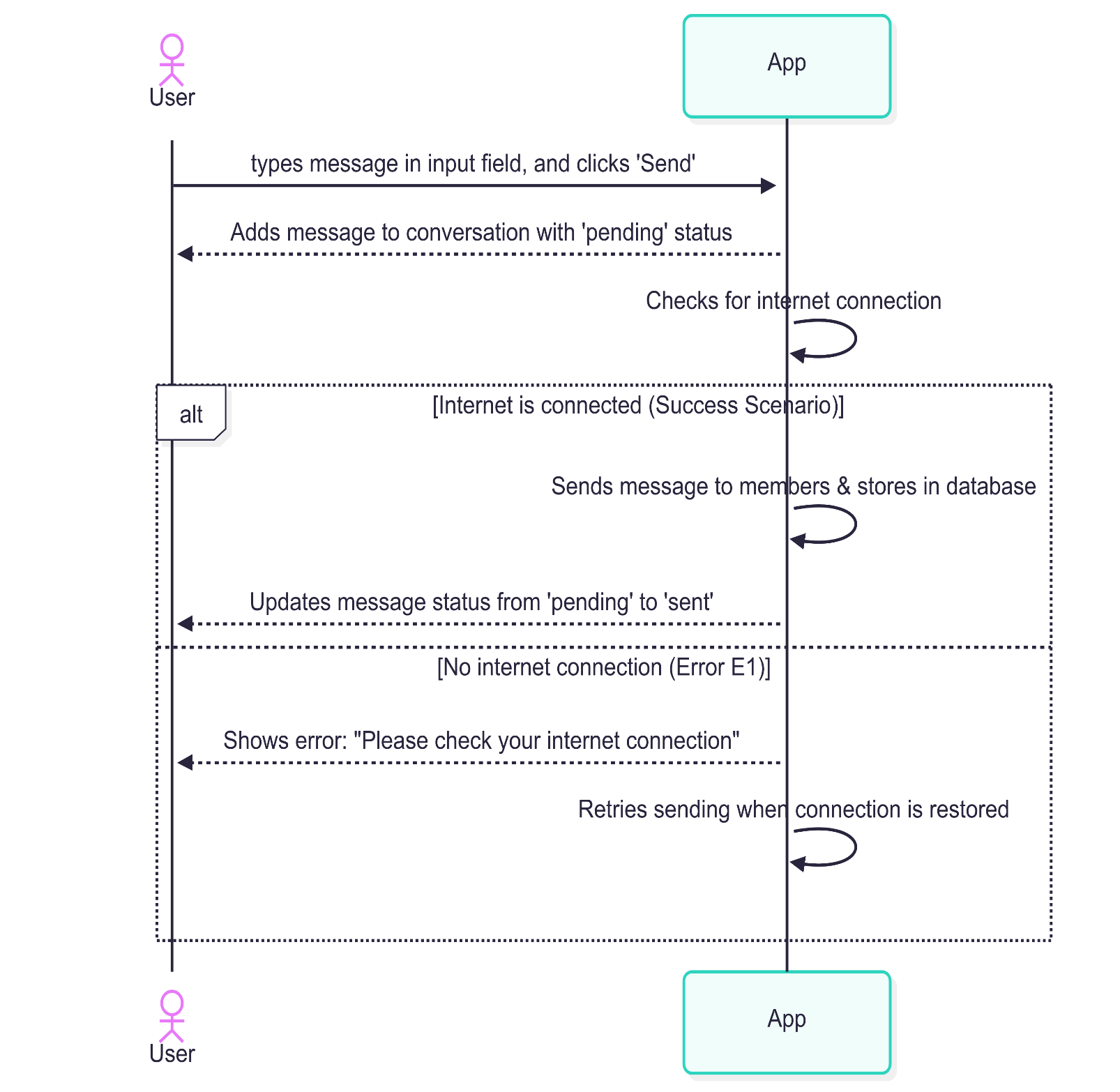
### **إنشاء حساب**

****

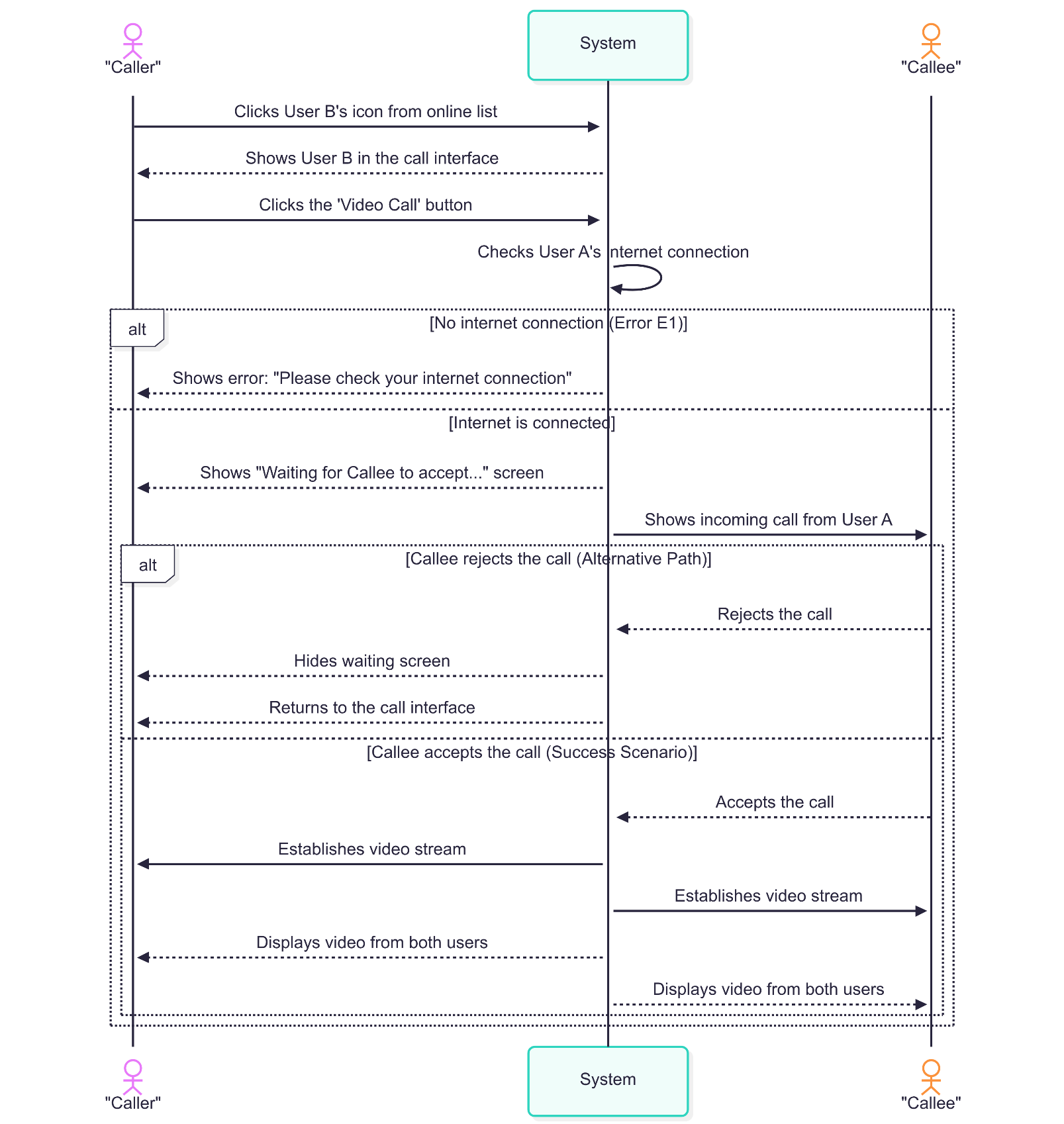
### **إنشاء محادثة**



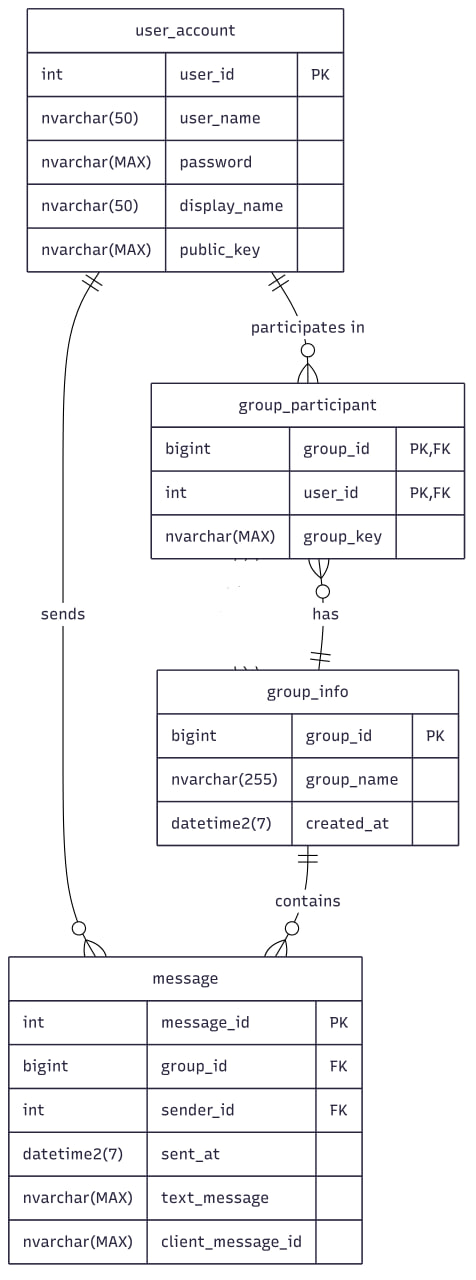
### **إرسال رسالة**



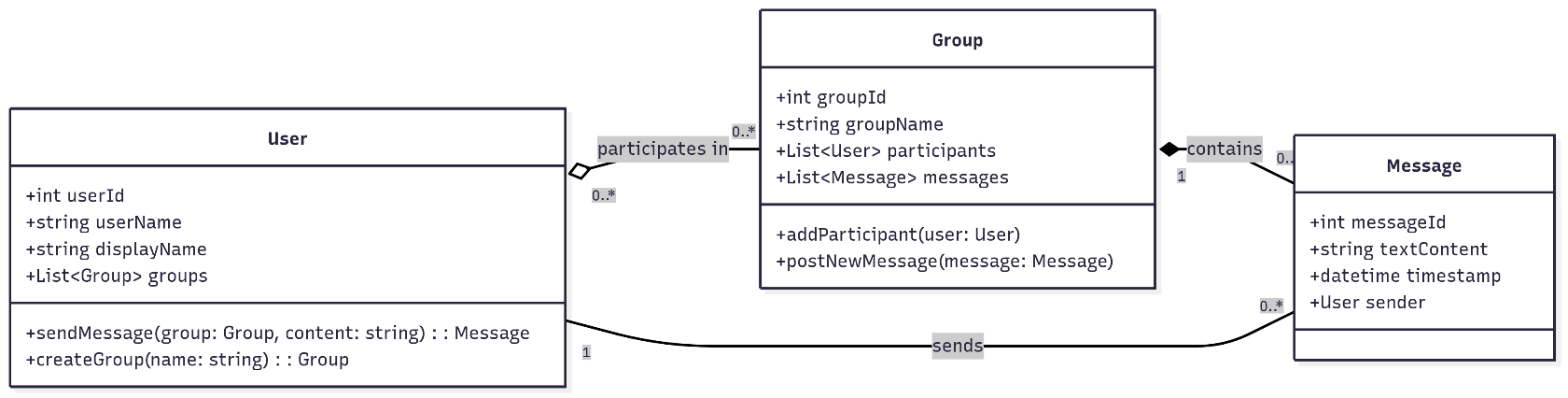
### **بدء مكالمة فيديو**



## مخطط علاقات الكيانات ERD لقاعدة بيانات السيرفر:



## مخطط الصفوف **(Class diagram) :**

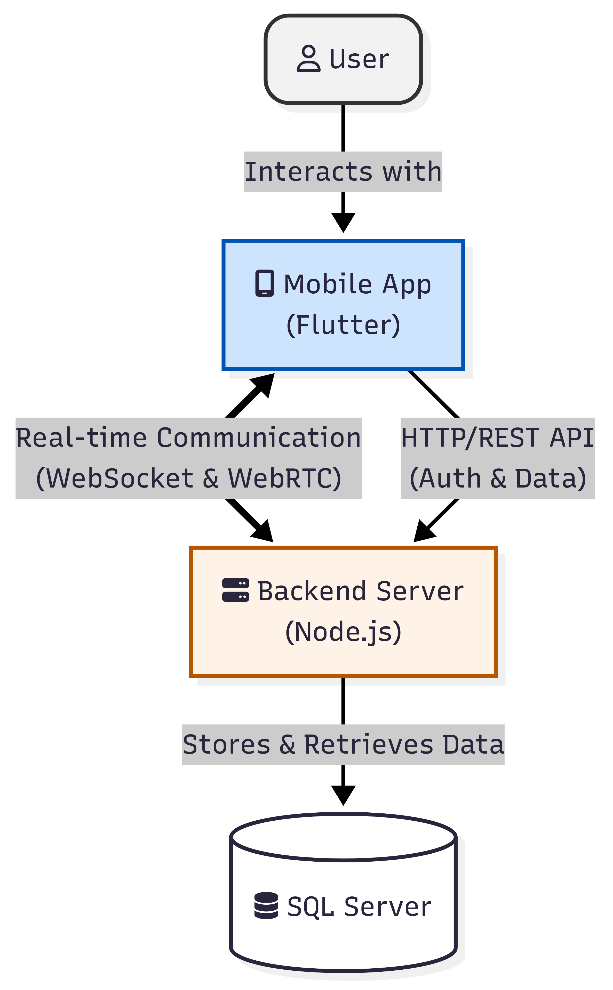


# **الفصل الثالث**

# **الدراسة التصميمة**

يهدف هذا الفصل إلى عرض التصميم المعماري والهيكلي لنظام تطبيق المحادثة. سيتم تقديم التصميم بدءاً من النظرة الشاملة للمعمارية الكلية، ثم التعمق في تصميم المكونات الرئيسية للنظام، بما في ذلك الواجهة الخلفية، الواجهة الأمامية، وقواعد البيانات.

## نظرة عامة لبنية النظام

تم تصميم النظام وفق معمارية حديثة متعددة الطبقات تهدف إلى تحقيق المبدأ الهندسي الأساسي "فصل الاهتمامات" (Separation of Concerns). هذا المبدأ يضمن أن كل جزء من النظام له مسؤولية واضحة ومحددة، مما يعزز من متانة النظام، ويسهل صيانته وتطويره مستقبلاً. تتكون البنية العامة، كما هو موضح في المخطط ()، من ثلاث مكونات رئيسية ومنفصلة منطقياً: العميل (Client)، الخادم (Backend Server)، وقاعدة البيانات (Database).

لنوضح بشكل عام الأساليب التي تم الاعتماد عليها في كل مكون:

* **العميل (Client)**

يمثل العميل تطبيق الهاتف الذي يتفاعل معه المستخدم مباشرة. تم تصميمه ليكون فعالاً فيما يخص واجهة المستخدم ومنطق العرض، ولكنه يعتمد بشكل كامل على الخادم في منطق الأعمال الأساسي. تم بناء هذا الجزء باستخدام نمط تصميم نموذج-عرض-نموذج العرض (Model-View-View Model - MVVM). يهدف هذا النمط إلى فصل واجهة المستخدم (View) عن حالتها ومنطقها (View Model)، مما يجعل الكود أكثر تنظيماً، ويسهل بشكل كبير عملية اختبار الوحدات (Unit Testing) للواجهة الأمامية وتطويرها بشكل مستقل عن الواجهة الخلفية.

* **الخادم (Backend Server)**

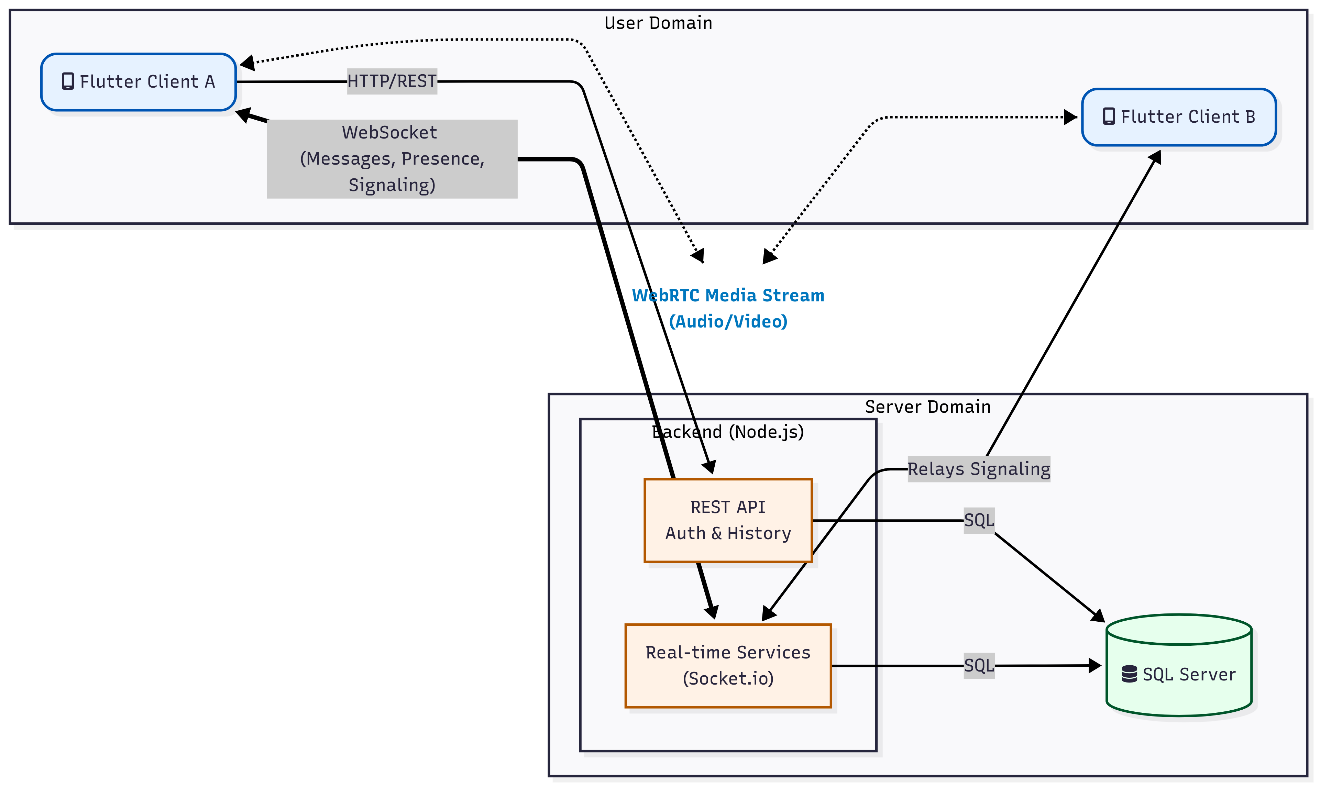
يعمل الخادم كمنسّق للنظام، حيث يقوم بتنسيق جميع الاتصالات، يدير منطق الأعمال، ويتعامل مع البيانات. تم تصميمه وفق هيكلية معيارية (Modular Monolith)، حيث تم تقسيم النظام إلى مجموعة من الخدمات المتخصصة والمستقلة التي تتواصل مع بعضها. تشمل هذه الوحدات خدمة المصادقة (Authentication)، وخدمة إدارة الحضور (Presence)، وخدمة المراسلة (Messaging). الفائدة الرئيسية لهذه المعمارية هي قابلية التوسع (Scalability)؛ حيث يمكن تحسين أداء كل خدمة على حدى، وسهولة الصيانة (Maintainability)، حيث يمكن تحديث أو استبدال أي وحدة دون التسبب في مشاكل لبقية النظام.

* **قاعدة البيانات (Database)**

هي الطبقة المسؤولة عن التخزين الدائم والمنظم لجميع بيانات التطبيق، مثل معلومات حسابات المستخدمين، تفاصيل المجموعات، وسجل الرسائل. تم اختيار قاعدة بيانات علائقية (SQL Server) لضمان تكامل البيانات واتساقها.

* يتم التفاعل بين هذه المكونات الرئيسية عبر بروتوكولات اتصال محددة تخدم أغراضاً مختلفة. يتم استخدام بروتوكول HTTP للعمليات القياسية التي لا تتطلب استجابة فورية، مثل تسجيل الدخول وإنشاء الحساب. في المقابل، يتم استخدام اتصال WebSocket المستمر للاتصالات في الزمن الحقيقي التي تتطلب تفاعلاً فورياً، مثل إرسال الرسائل وتحديثات حالة الحضور. وأخيراً، ولتحقيق أعلى كفاءة في مكالمات الفيديو والصوت، يتم الاعتماد على بروتوكول WebRTC لتأسيس اتصال مباشر وآمن بين العملاء (Peer-to-Peer).
* سيتم تفصيل هذه المعماريات وأنماط التفاعل بشكل أعمق في الأقسام اللاحقة من هذا الفصل.

## تدفق البيانات ضمن النظام

يستعرض المخطط () المعمارية الهجينة (Hybrid Architecture) المعتمدة للنظام، والتي تدمج بفعالية بين نموذجي العميل-الخادم (Client-Server) ونظير-إلى-نظير (Peer-to-Peer). تم تصميم هذه البنية لفصل المكونات منطقياً ضمن نطاقين أساسيين: نطاق المستخدم (User Domain) ونطاق الخادم (Server Domain). يهدف هذا الدمج إلى الاستفادة من موثوقية النموذج المركزي لإدارة البيانات والمراسلة، مع تحقيق الكفاءة والسرعة التي يوفرها الاتصال المباشر لمكالمات الفيديو والصوت، مما يضمن استقلالية المكونات ويسهل فهم تدفق البيانات.

يمكن تحليل تدفق البيانات والتفاعلات في النظام إلى ثلاثة أنماط متميزة، كل منها يخدم غرضاً محدداً:

* **التفاعلات غير المتزامنة عبر HTTP**

هذا هو نمط الاتصال التقليدي المستخدم للعمليات التي لا تتطلب استجابة فورية أو اتصالاً مستمراً. عندما يقوم المستخدم بعمليات مثل إنشاء حساب جديد، أو تسجيل الدخول، أو طلب سجل المحادثات القديمة، يقوم العميل بإرسال طلب HTTP إلى نقطة النهاية المخصصة في REST API داخل الواجهة الخلفية. تقوم هذه الخدمة بمعالجة الطلب، والتحقق من صحته، والتواصل مع قاعدة البيانات إذا لزم الأمر، ثم تعيد استجابة واحدة (one-time response) إلى العميل، وبعدها يتم إغلاق الاتصال.

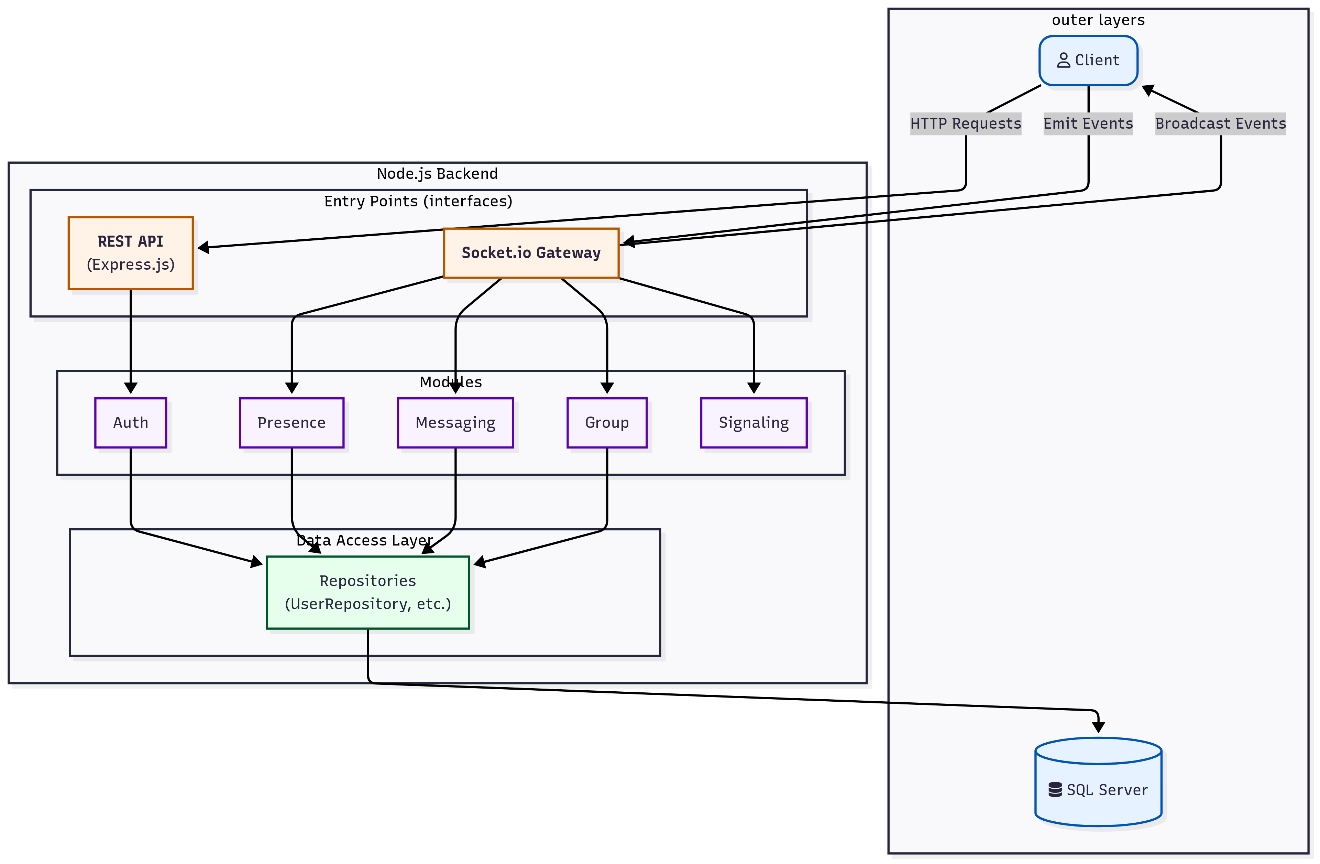
* **التفاعلات في الزمن الحقيقي عبر مكتبة Socket.io**

لتحقيق الفورية المطلوبة في تطبيقات المحادثة، يتم إنشاء قناة اتصال مستمرة وثنائية الاتجاه بين كل عميل والخادم باستخدام بروتوكول WebSocket، والذي تتم إدارته عبر مكتبة Socket.io. هذه القناة هي المسؤولة عن جميع الأحداث التي تتطلب تحديثاً فورياً، وأهمها:

* المراسلة الفورية: يتم إرسال واستقبال الرسائل النصية عبر هذه القناة لضمان وصولها في نفس اللحظة.
* إدارة الحضور: يتم بث تحديثات حالة المستخدمين (متصل/غير متصل) بشكل فوري لجميع جهات الاتصال المعنية.
* إشارات WebRTC: تعمل هذه القناة كوسيط لنقل رسائل الإشارة (Signaling Messages) بين العملاء. حيث يقوم الخادم بدور "الموجه" لهذه الرسائل، مما يساعد الطرفين على تأسيس الاتصال المباشر بينهما.
* **الاتصال المباشر من نظير إلى نظير عبر WebRTC**

هذا هو النمط الأكثر كفاءة وأداءً لنقل بيانات الوسائط الحية (الصوت والفيديو). بعد أن تنجح "Real-time Services" في مساعدة العميلين على تبادل معلومات الاتصال اللازمة عبر عملية الإشارة، يتم تأسيس اتصال WebRTC مباشر وآمن بينهما (Peer-to-Peer). جميع بيانات الصوت والفيديو الخاصة بالمكالمة تتدفق عبر هذا المسار المباشر (الموضح بالخط المتقطع في المخطط)، دون الحاجة للمرور عبر الخادم. هذا التصميم يوفر فائدتين رئيسيتين: أولاً، يقلل بشكل كبير من زمن الاستجابة (Latency)، مما ينتج عنه مكالمات أكثر سلاسة. ثانياً، يعزز الخصوصية لأن محتوى المكالمة الفعلي لا يمر عبر أي بنية تحتية مركزية، إضافةً الى طبقة الأمان التي يضيفها البروتوكول اثناء النقل.

## تصميم الواجهة الخلفية **(Backend)**

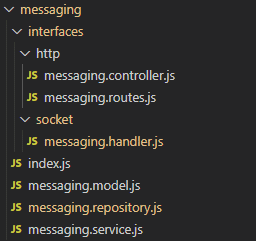
تم تصميم الواجهة الخلفية المبنية باستخدام Node.js وفق معمارية المونوليث المعياري (Modular Monolith) وهي بنية تهدف إلى الجمع بين بساطة النشر والصيانة الخاصة بالتطبيقات المترابطة (Monolithic) مع التنظيم والفصل المنطقي الذي توفره الوحدات المستقلة. داخلياً، تتبع هذه المعمارية نمط البنية متعددة الطبقات (Layered Architecture) لضمان فصل واضح بين المسؤوليات داخل كل وحدة والنظام ككل. يوضح المخطط () هذه الطبقات وتدفق الطلبات بينها.

* **تطبيق معمارية المونوليث المعياري**

في سياق هذا المشروع، تم تطبيق هذه المعمارية من خلال تقسيم قاعدة الكود البرمجية بأكملها إلى مجموعة من الوحدات (Modules) المستقلة منطقياً ولكنها تُنشر معاً كتطبيق واحد. كل وحدة مسؤولة عن مجال عمل (Business Domain) محدد ومتكامل، مما يمنع التداخل غير المنضبط بين أجزاء النظام المختلفة. الوحدات الرئيسية التي تم إنشاؤها هي:

* وحدة المستخدمين (Users Module): مسؤولة عن كل ما يتعلق بالمستخدم، من إنشاء حساب، وتسجيل الدخول، وإصدار التوكن (JWT)، وإدارة الملف الشخصي.
* وحدة الحضور (Presence Module): مسؤولة حصرياً عن تتبع وإدارة حالة اتصال المستخدمين (متصل/غير متصل) وبث هذه التحديثات.
* وحدة المجموعات (Group Module): تدير المنطق المعقد لإنشاء المجموعات وبروتوكول تبادل المفاتيح الآمن بين المشاركين.
* وحدة المراسلة (Messaging Module): تعالج منطق إرسال وتمرير الرسائل النصية والملفات.
* وحدة الإشارة (Signaling Module): تعمل كوسيط لتمرير حزم WebRTC اللازمة لتهيئة عملية الاتصال.
* **تطبيق البنية متعددة الطبقات داخل كل وحدة (Layered Architecture)**

لتحقيق تنظيم أعمق وضمان فصل المسؤوليات داخل كل وحدة من الوحدات المذكورة أعلاه، تم اعتماد البنية متعددة الطبقات (Layered Architecture). هذا يعني أن كل وحدة مقسمة داخلياً إلى طبقات أفقية، حيث تتواصل كل طبقة فقط مع الطبقة التي تليها مباشرة، مما يخلق تدفقاً للبيانات والتحكم باتجاه واحد. كما هو موضح:



تتكون كل وحدة (Module) من الطبقات التالية:

* **طبقة الواجهات ونقاط الدخول (Interfaces / Entry Points Layer):**

هي الطبقة العليا المسؤولة عن استقبال الطلبات من العملاء. في وحدة المراسلة، تم تقسيم هذه الطبقة بشكل واضح إلى واجهتين منفصلتين بناءً على بروتوكول الاتصال، كما يظهر في مجلد interfaces:

* **واجهة http**

  تحتوي على messaging.routes.js الذي يحدد نقاط النهاية مثل /api/chats/:chatId/messages)) ويقوم بعدها بربطها بالمتحكم المناسب. بالإضافة الى دور المتحكم messaging.controller.js الذي يستقبل طلبات HTTP لمعالجة العمليات مثل جلب سجل المحادثات.

* **واجهة socket**

تحتوي على messaging.handler.js، الذي يعمل كمعالج لأحداث WebSocket مثل (sendMessage) ويتعامل مع الاتصالات الفورية.

* **طبقة الخدمات (Service Layer):**

تحتوي هذه الطبقة على منطق الأعمال الفعلي (Business Logic) الخاص بالمراسلة. يتم تمثيلها بملف messaging.service.js. داخل هذا الملف، يتم تنفيذ العمليات المعقدة مثل التحقق من أن المستخدم عضو في المجموعة قبل السماح له بإرسال رسالة، وتنسيق عملية حفظ الرسالة في قاعدة البيانات ثم بثها إلى المشاركين الآخرين. هذه الطبقة هي التي تستدعي المستودعات (Repositories) لإنجاز مهامها.

* **طبقة الوصول إلى البيانات (Data Access Layer - DAL):**

تمثل هذه الطبقة صلة الوصل الوحيدة بين منطق الأعمال وقاعدة البيانات. يتم تمثيلها بملف messaging.repository.js. هذا الملف يحتوي حصرياً على الجزء المسؤول عن تنفيذ استعلامات SQL المتعلقة بالرسائل والمجموعات، مثل createMessage أو findMessagesByGroupId. هذا يعزل طبقة الخدمات تماماً عن تفاصيل كيفية تخزين واسترجاع البيانات.

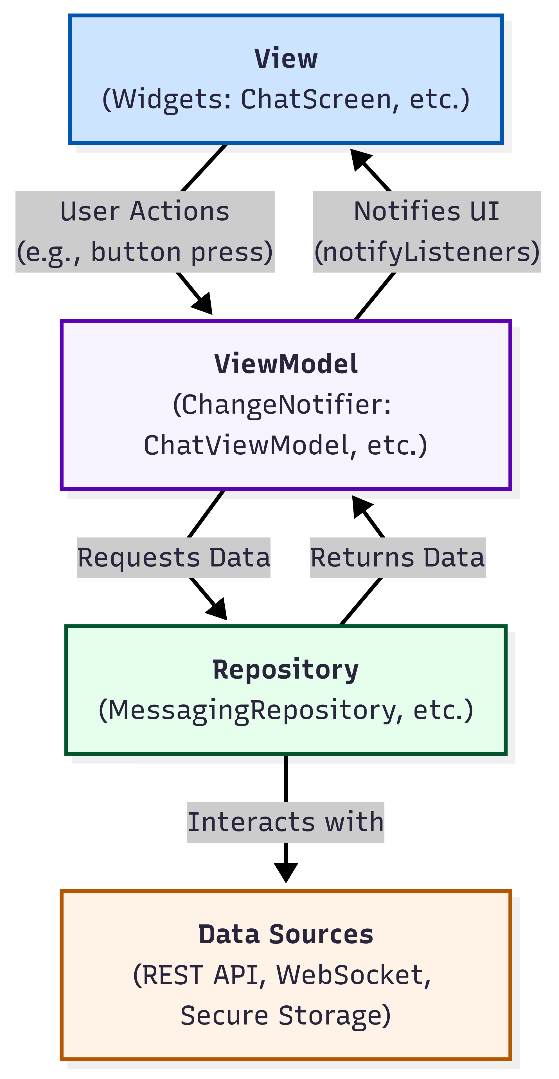
* **ملفات إضافية داعمة:**
* messaging.model.js:

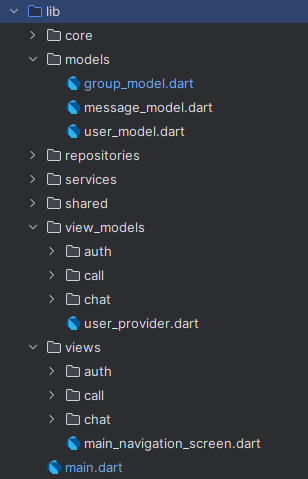
يحتوي هذا الملف على مخططات التحقق من صحة البيانات (Data Validation Schemas) باستخدام مكتبة Joi، مما يضمن أن البيانات الواردة (مثل محتوى الرسالة) تتبع التنسيق المطلوب.

* index.js: (نقطة التصدير الموحدة)

يلعب هذا الملف دوراً تنظيمياً هامّاً. هو يعمل كواجهة عامة (Public Interface) للوحدة. بدلاً من أن تقوم الوحدات الأخرى باستيراد الملفات بشكل فردي (مثل require ('../ messaging / messaging.service'))، تقوم فقط بالاستيراد من المجلد الرئيسي للوحدة. يقوم ملف index.js بتجميع وتصدير الأجزاء التي يجب أن تكون متاحة للوحدات الأخرى (مثلregisterMessagingHandlers و messagingRoutes)، بينما يخفي تفاصيل التنفيذ الداخلية. هذا المبدأ يسمى التغليف (Encapsulation) على مستوى الوحدة، وهو يعزز من استقلاليتها ويجعل إعادة هيكلة الملفات الداخلية أسهل.

## تصميم الواجهة الأمامية **(Frontend):**

تم تصميم الواجهة الأمامية لتطبيق الهاتف بهدف أساسي وهو إنشاء قاعدة منظمة، وقابلة للاختبار، وسهلة الصيانة. لتحقيق ذلك، تم اعتماد نمط التصميم المعماري (Model-View-View Model - MVVM). يوفر هذا النمط فصلاً واضحاً بين واجهة المستخدم (UI) ومنطق الأعمال الخاص بها، مما يسمح بتطوير واختبار كل جزء بشكل مستقل. كما يوضحه الشكل ().



يتكون تطبيق نمط MVVM في هذا المشروع من ثلاث مكونات رئيسية:

* **النموذج (Model):**

يمثل هذا المكون بنية البيانات (Data Structure). يتمثل الـ Model في فئات تحدد شكل الكيانات المختلفة، مثل UserModel، MessageModel، وGroupModel. هذه الفئات هي المسؤولة عن تحديد السمات (Attributes) لكل كيان وتوفير طرق لتحويل البيانات من وإلى تنسيق JSON للتواصل مع الواجهة الخلفية. يتم التعامل مع هذه النماذج بشكل أساسي داخل طبقة المستودعات (Repositories).

* **العرض (View):**

يمثل هذا المكون واجهة المستخدم (UI) التي يراها المستخدم ويتفاعل معها. في المشروع، تم تنظيم جميع الشاشات وعناصر الواجهة الخاصة بالعرض داخل مجلد views. تم تقسيم هذا المجلد أيضاً ، مثل auth (لشاشات تسجيل الدخول)، call (لشاشات الاتصال)، و chat (لشاشات المحادثة). ملفات مثل login\_screen.dart و chat\_screen.dart هي أمثلة على الـ View. وظيفتها الأساسية هي عرض الحالة التي يوفرها الـ ViewModel، وإعلامه عند حدوث أي تفاعل من المستخدم.

* **نموذج العرض (ViewModel):**

يعمل الـ ViewModel كوسيط بين الـ View والـ Model ويحتوي على كل منطق العرض وإدارة الحالة. تم تنظيم هذه المكونات داخل مجلد view\_models، وهو أيضاً مقسم حسب الميزة. على سبيل المثال، LoginViewModel (داخل مجلد auth) يدير حالة شاشة تسجيل الدخول، وChatViewModel (داخل مجلد chat) يدير حالة شاشة محادثة معينة. الـ ViewModel هو المسؤول عن استدعاء المستودعات (Repositories) لجلب البيانات، معالجة تفاعلات المستخدم، وتحديث حالته لإعلام الـ View بوجود تغييرات.

* **تدفق التحكم والبيانات في التطبيق:**
* من العرض إلى نموذج العرض (View → ViewModel): تبدأ الدورة عندما يقوم المستخدم بإجراء تفاعل على الواجهة، مثل الضغط على زر "إرسال". تقوم الـ View بإعلام الـ ViewModel بهذا الحدث عن طريق استدعاء دالة مقابلة، مثل viewModel.sendMessage("Hello").
* من نموذج العرض إلى المستودع (ViewModel → Repository): يقوم الـ ViewModel، الذي يحتوي على منطق الأعمال، بمعالجة هذا الطلب. يستدعي الدالة المناسبة في طبقة المستودعات (Repository) للتعامل مع البيانات، مثل (...)messagingRepository.sendEncryptedMessage.
* من المستودع إلى نموذج العرض (Repository → ViewModel): يقوم المستودع بالتفاعل مع مصادر البيانات (مثل REST API أو WebSocket)، وعندما تعود النتيجة (سواء كانت بيانات جديدة أو تأكيداً)، يتم إرجاعها إلى الـ ViewModel.
* من نموذج العرض إلى العرض (ViewModel → View): بناءً على النتيجة التي عادت من المستودع، يقوم الـ ViewModel بتحديث حالته الداخلية (على سبيل المثال، إضافة رسالة جديدة إلى القائمة أو تغيير حالة التحميل). بعد ذلك، يستدعي دالة ()notifyListeners، التي تقوم بإعلام الـ View بأن حالتها قد تغيرت. تستجيب الـ View لهذا الإشعار عن طريق إعادة بناء نفسها لتعرض البيانات المحدثة للمستخدم.