الجمهورية العربية السورية
المعهد العالي للعلوم التطبيقية والتكنولوجيا
قسم النظم المعلوماتية – اختصاص شبكات ونظم تشغيل

مشروع السنة الرابعة

تصميم بوابة دفع آمنة

تقديم

اسماعيل عباس

إشراف

د.حسين خيو **م.مفيد دوزكنجي**

كلمة شكر

أتقدّم بأسمى عبارات الشكر والتقدير لكل من ساهم في إنجاز هذا المشروع، وكان له دور في دعمي ومساندتي خلال مراحل العمل المختلفة.

وفي مقدمتهم، أخصّ بالشكر والعرفان الدكتور حسين خيو، الذي كان حاضراً بتوجيهاته السديدة، ومتابعاً لكل خطوة بخبرته الواسعة ودعمه المتواصل. لقد كانت ثقته بي حافزاً كبيراً للاستمرار، ونصائحه القيمة دليلاً لي في تجاوز التحديات.

كما أتوجّه بجزيل الشكر والامتنان إلى المهندس مفيد دوزكنجي، المشرف على المشروع، لما قدّمه لي من توجيهات تقنية وملاحظات بنّاءة أثرت من جودة العمل، وأسهمت في تطوير رؤيتي لهذا المشروع بشكل كبير.

الخلاصة

مع التطور المتسارع للتجارة الإلكترونية والخدمات المصرفية الرقمية، أصبحت الحاجة إلى أنظمة دفع آمنة وسلسة من أولويات المستخدمين والمؤسسات على حد سواء. ومن هذا المنطلق، جاء مشروعنا لتصميم وتنفيذ نظام دفع إلكتروني متكامل يربط بين موقع تجاري، و بوابة دفع مصرفية، وتطبيق جوال مبني باستخدام تقنية Flutter، مع دعم لتقنيات المصادقة الحديثة مثل passkey، بهدف رفع مستوى الأمان وتحسين تجربة المستخدم.

قمنا في هذا المشروع بدراسة الجوانب النظرية والتقنية المتعلقة بأنظمة الدفع، المصادقة متعددة العوامل، آلية إشعارات الدفع، وواجهات برمجة التطبيقات (APIs) المستخدمة في التواصل مع الأنظمة البنكية. كما تناولنا مقارنة بين طرق المصادقة التقليدية (OTP، كلمات المرور) والتقنيات الحديثة مثل WebAuthn/Passkey، مع تحليل مزاياها وسلبياتها.

على الجانب العملي، قمنا بتنفيذ النموذج الأولي للنظام حيث يبدأ المستخدم بتحديد طلبه من الموقع التجاري، ثم يدخل بيانات الدفع التي تُرسل إلى النظام البنكي. بدوره، يقوم البنك بإرسال إشعار إلى التطبيق المحمول لتأكيد العملية، حيث يُمنح المستخدم خيار الموافقة أو الرفض عبر مصادقة passkey. كما تم استخدام خدمة Firebase Cloud Messaging لإرسال الإشعارات الآنية، وتقنيات التشفير لحماية بيانات المستخدم.

Abstract

With the rapid development of e-commerce and digital banking services, the need for secure and seamless payment systems has become a top priority for both users and institutions. In this context, our project was initiated to design and implement an integrated digital payment system that connects a commercial website, a banking payment gateway, and a mobile application built using Flutter technology. The system also supports modern authentication technologies such as passkeys, aiming to enhance security and improve user experience.

In this project, we studied the theoretical and technical aspects related to payment systems, multi-factor authentication, payment notification mechanisms, and the APIs used for communication with banking systems. We also provided a comparison between traditional authentication methods (such as OTPs and passwords) and modern technologies like WebAuthn/Passkeys, analyzing their advantages and disadvantages.

On the practical side, we implemented a prototype of the system, where the user starts by selecting their order from the commercial website and then enters the payment details, which are sent to the banking system. The bank, in turn, sends a notification to the mobile application to confirm the transaction, where the user is given the option to approve or reject the payment through passkey authentication. We also used Firebase Cloud Messaging for real-time notifications and encryption technologies to protect user data.

المحتويات

Contents

قائمة الأشكال

الاختصارات

عربي	English	الاختصار
المصادقة متعددة العوامل	Multi-Factor Authentication	**MFA**
واجهة برمجة التطبيقات	Application Programming Interface	**API**
بروتوكول نقل النص التشعبي الآمن	Hypertext Transfer Protocol Secure	**HTTPS**
نمط العرض-النموذج-المتحكم	Model-View-Controller	**MVC**
خدمة إشعارات فايربيس السحابية	Firebase Cloud Messaging	**FCM**
بروتوكول أمان الطبقة الآمنة	Secure Sockets Layer/Transport Layer Security	**SSL/TLS**
معيار أمان بيانات صناعة بطاقات الدفع	Payment Card Industry Data Security Standard	**PCI DSS**
وحدة النظام الموثوق	Trusted Platform Module	**TPM**
رمز التحقق لمرة واحدة	One-Time Password	**OTP**
لتحقق الثنائي	Two-Factor Authentication	**2FA**
خوارزمية رمز التحقق المعتمد على الوقت	Time-based One-Time Password	**TOTP**
مكتبة إدارة العلاقات مع الكائنات	Object-Relational Mapping	**ORM**
عمليات القراءة والإضافة والتعديل والحذف	Create, Read, Update, Delete	**CRUD**
مكتبة التوثيق المحلي	Local Authentication	**local_auth**
معيار المصادقة على الويب	Web Authentication	**WebAuthn**

مقدمة عامة

في عصرنا الحالي، أصبحت التجارة الإلكترونية والخدمات المصرفية الرقمية من الركائز الأساسية في حياة الأفراد والشركات، ومع هذا التوسع الكبير في المعاملات الرقمية، ازدادت بشكل ملحوظ التهديدات الأمنية والهجمات الإلكترونية التي تستهدف بيانات المستخدمين والأنظمة المالية. وقد تؤدي هذه الهجمات، إن لم يتم التصدي لها، إلى خسائر مادية فادحة وتعطيل كامل للبنية التحتية الرقمية.

من هذا المنطلق، برزت أهمية أمن المعلومات كعلم متخصص يهدف إلى حماية الأنظمة والبيانات من أي تهديد خارجي أو داخلي، من خلال تطبيق مبادئه الثلاثة الأساسية: السرية (Confidentiality)، السلامة (Integrity)، والتوافر (Availability). ولم يعد يكفي استخدام كلمات المرور التقليدية أو أنظمة التحقق البسيطة، بل أصبح من الضروري اعتماد تقنيات أكثر تطوراً تواكب حجم التهديدات وتضمن تجربة آمنة وسلسة للمستخدم.

في هذا السياق، تم تنفيذ هذا المشروع الذي يهدف إلى بناء نظام دفع إلكتروني متكامل يجمع بين موقع تجاري لطلب المنتجات، بوابة دفع مصرفية آمنة، وتطبيق جوال مبني باستخدام Flutter، يكون دوره استقبال الإشعارات وتنفيذ عمليات المصادقة باستخدام تقنية حديثة تعرف باسم passkey، والتي تُعتبر بديلاً آمناً وفعالاً لكلمات المرور والرموز المؤقتة (OTP).

الهدف من المشروع

هدف هذا المشروع بشكل أساسي إلى تصميم وتنفيذ نظام دفع إلكتروني آمن ومتكامل، يربط بين موقع إلكتروني تجاري، بوابة دفع مصرفية، وتطبيق جوال مطوّر باستخدام Flutter، ويعتمد في عملية المصادقة على تقنية حديثة تُعرف باسم Passkey، وذلك بهدف رفع مستوى الأمان وسهولة الاستخدام في المعاملات الرقمية.

كما يهدف المشروع إلى فهم وتحليل النماذج المختلفة لأنظمة الدفع الإلكتروني ومصادقة المستخدمين، من خلال دراسة معمقة للتقنيات الحالية، مثل المصادقة متعددة العوامل (MFA)، بروتوكولات WebAuthn ، وتقنيات إرسال الإشعارات مثل Firebase Cloud Messaging.

ومن الجوانب التطبيقية، يسعى المشروع إلى بناء بيئة اختبار تجريبية كاملة يمكن من خلالها محاكاة عمليات الشراء والدفع، وتنفيذ سيناريوهات متعددة تشمل حالات قبول العملية، رفضها، وفشل المصادقة، وذلك بهدف قياس فعالية النظام من حيث الأمان، سرعة الاستجابة، وتجربة المستخدم.

إضافة إلى ذلك، يُعتبر المشروع بمثابة قاعدة تقنية أولية يمكن البناء عليها مستقبلاً في مجال تطوير أنظمة الدفع الذكية، وربطها مع الخدمات المصرفية الحديثة، مما يفتح آفاقاً واسعة للبحث العلمي والتجاري على حد سواء

.

النية العامة للتقرير

يتكوّن هذا التقرير بشكل أساسي من أربعة فصول رئيسية، تغطّي جميع مراحل تطوير النظام المدروس بشكل شامل ومترابط، بدءًا من الجانب النظري والتحليلي، وصولاً إلى التنفيذ العملي واختبارات الأداء. فيما يلي عرض لمحتويات الفصول الرئيسية:

- الفصل الأول: الدراسة المرجعية

يتناول هذا الفصل عرضًا نظريًا مفصلًا لموضوع أنظمة الدفع الإلكتروني، والمصادقة متعددة العوامل، مع التركيز على تقنية Passkey ودورها في تعزيز أمن المعاملات. كما يتضمن شرحًا لآلية إشعارات الدفع، وبروتوكولات الحماية المعتمدة في التطبيقات المصرفية.

- الفصل الثاني: الأدوات المستخدمة

يستعرض هذا الفصل أهم الأدوات البرمجية والتقنية التي تم استخدامها في تطوير وتنفيذ النظام، مثل: Flutter، Firebase، ، وتقنيات WebAuthn، WebSocket إضافة إلى أدوات التجريب والمحاكاة التي ساعدت في بناء واختبار النظام.

- الفصل الثالث: التنفيذ العملي والاختبارات

يعتبر هذا الفصل من أكثر الفصول أهمية، حيث يوضّح خطوات بناء النظام من الناحية العملية بدءًا من الموقع التجاري مرورًا بالتكامل مع البنك ووصولاً إلى تطبيق الجوال. كما يتضمّن عرضًا مفصلًا للسيناريوهات المجربة، وتحليلًا لنتائج الاختبار من حيث الأداء والأمان. وتم أيضًا تقديم تطويرات برمجية لتحسين التكامل بين المكوّنات وتعزيز تجربة المستخدم.

- الفصل الرابع: خاتمة المشروع

يقدّم هذا الفصل خلاصة شاملة لما تم إنجازه خلال المشروع، إضافة إلى عرض لأبرز التحديات والصعوبات التي تمت مواجهتها أثناء التنفيذ، والمهارات التقنية التي تم اكتسابها، إلى جانب آفاق التوسّع والتطوير المستقبلي للنظام.

تجدر الإشارة إلى أننا قمنا بإضافة مؤشرات للمراجع والمصادر النظرية المستخدمة في معظم الفقرات، مما يتيح للقارئ العودة إلى المرجع الأصلي لكل فكرة أو مصطلح تقني في حال رغب بالتوسع والاطلاع على التفاصيل.

الفصل الأول الدراسة المرجعية

1. مقدمة عن أمن المعلومات والدفع الإلكتروني:

في عصر التحول الرقمي المتسارع، أصبحت الأنظمة الرقمية ومنصات التجارة الإلكترونية جزءًا لا يتجزأ من الحياة اليومية، حيث أسهمت في تبسيط العمليات التجارية وتسهيل الوصول إلى الخدمات المالية. ومع هذا التوسع، برزت الحاجة الملحة لحماية البيانات الحساسة والمعلومات الشخصية التي تُنقل عبر هذه الأنظمة. يواجه القطاع المالي تحديات أمنية متزايدة، تشمل التسريب غير المصرح به للبيانات، الاحتيال الإلكتروني، الهجمات السيبرانية المتطورة، وسرقة الهوية الرقمية. هذه التحديات تهدد استقرار الأنظمة المالية وتُعرّض ثقة المستخدمين للخطر، مما يجعل أنظمة الأمان ضرورة استراتيجية لضمان استمرارية الأعمال وحماية المستخدمين.

تعتمد أنظمة الأمان في هذا السياق على ثلاث ركائز أساسية:

- السرية (Confidentiality): وهي ضمان حماية البيانات من الوصول غير المصرح به، سواء أثناء التخزين أو النقل.

- السلامة (Integrity): وهي الحفاظ على دقة البيانات وسلامتها، بحيث لا يتم التلاعب بها أو تعديلها بطريقة غير مشروعة.

- التوافر (Availability): وهو ضمان إتاحة الأنظمة والبيانات للمستخدمين المصرح لهم في أي وقت، مع تقليل الانقطاعات الناتجة عن الهجمات أو الأعطال.

أنظمة الدفع الإلكتروني، التي حلت تدريجيًا محل الدفع النقدي ، أصبحت العمود الفقري للتجارة الرقمية. تعتمد هذه الأنظمة على بوابات دفع مصرفية وتطبيقات متقدمة تدمج تقنيات أمان حديثة، مثل المصادقة متعددة العوامل (MFA) وPasskey، لضمان حماية المعاملات المالية. تهدف هذه التقنيات إلى تقليل المخاطر المرتبطة بالدفع الإلكتروني، مع تعزيز تجربة المستخدم من خلال توفير عمليات دفع سريعة وآمنة. يسعى هذا الفصل إلى تقديم دراسة نظرية شاملة تُسلط

الضوء على هذه التقنيات، مع التركيز على دورها في بناء أنظمة دفع إلكترونية موثوقة وقادرة على مواجهة التحديات الأمنية المعاصرة [1] .

2. مفهوم المصادقة متعددة العوامل (MFA) :

تُعد المصادقة متعددة العوامل (Multi-Factor Authentication - MFA) واحدة من أهم الركائز الأمنية في العصر الرقمي، حيث تهدف إلى تعزيز حماية الحسابات والمعاملات من خلال التحقق من هوية المستخدم باستخدام أكثر من وسيلة مستقلة. على عكس الأنظمة التقليدية التي تعتمد على كلمة مرور واحدة، تجمع MFA بين ثلاث فئات رئيسية للتحقق:

- عامل المعرفة (Knowledge Factor): وهو شيء يعرفه المستخدم، مثل كلمة المرور أو رمز PIN.

- عامل الامتلاك (Possession Factor): وهو شيء يمتلكه المستخدم، مثل رمز تحقق يُرسل إلى هاتفه المحمول، البريد الإلكتروني، أو جهاز أمان مادي (Hardware Token).

- عامل الهوية (Inherence Factor): وهو شيء يمثل المستخدم، مثل بصمة الإصبع، التعرف على الوجه، أو تحليل نمط الصوت.

تعمل MFA على تقليل مخاطر الاختراق بشكل كبير، حيث يحتاج المهاجم إلى الوصول إلى جميع عوامل المصادقة في وقت واحد، وهو أمر صعب للغاية. على سبيل المثال، حتى لو تمكن المهاجم من سرقة كلمة المرور، فإنه سيحتاج أيضًا إلى الوصول إلى جهاز المستخدم أو بياناته البيومترية لإكمال عملية تسجيل الدخول.

مع ذلك، ظهرت تقنية Passkey كتطور ثوري في مجال المصادقة، متجاوزةً القيود المرتبطة بـ MFA التقليدية. تعتمد Passkey على نظام التشفير القائم على المفاتيح العامة والخاصة (Public/Private Key Cryptography)، مما يوفر مستوى أمان أعلى بكثير. يتم تخزين المفتاح الخاص بشكل آمن على جهاز المستخدم، بينما يُرسل المفتاح العام إلى الخادم للتحقق. هذا النهج يلغي الحاجة إلى كلمات المرور التقليدية، ويقلل من مخاطر هجمات التصيّد والاحتيال، مما يجعل Passkey أداة مثالية لتأمين المدفوعات الإلكترونية وغيرها من التطبيقات الحساسة [2].

3. آلية عمل Passkey :

تعتمد تقنية Passkey على معيار WebAuthn، وهو بروتوكول موحد طورته منظمة FIDO Alliance بالتعاون مع World Wide Web Consortium (W3C)يهدف هذا المعيار إلى توفير آلية مصادقة آمنة وسهلة الاستخدام تعتمد على التشفير غير المتماثل .(Asymmetric Cryptography) عند إعدادPasskey ، يتم إنشاء زوج من المفاتيح:

- **المفتاح الخاص (Private Key) :** يُخزَّن بشكل آمن داخل جهاز المستخدم، باستخدام تقنيات مثل Trusted Platform Module (TPM) أو Secure Storage في أجهزة.Apple
- **المفتاح العام :(Public Key)** يُرسل إلى الخادم أو الموقع المعني ويُستخدم للتحقق من هوية المستخدم.

عند محاولة تسجيل الدخول، يرسل الموقع أو التطبيق طلب تحقق (Challenge) إلى جهاز المستخدم. يقوم الجهاز بتوقيع هذا الطلب باستخدام المفتاح الخاص، ثم يُرسل التوقيع الرقمي إلى الخادم، الذي يتحقق منه باستخدام المفتاح العام المرتبط. هذه العملية تضمن أن الجهاز المستخدم هو الجهاز المصرح به، دون الحاجة إلى إدخال كلمة مرور أو مشاركة أي بيانات حساسة.

مزايا Passkey تشمل:

- **منع إعادة استخدام بيانات الدخول** :كل Passkey فريدة لكل موقع أو تطبيق، مما يمنع هجمات إعادة الاستخدام.(Credential Stuffing)
- **مقاومة هجمات التصيّد** :بما أن العملية تعتمد على الجهاز نفسه، فلا يمكن خداع المستخدم لإدخال بياناته في مواقع مزيفة.
- **سهولة الاستخدام** :تتيح Passkey تسجيل الدخول باستخدام تقنيات بيومترية مثل بصمة الإصبع أو التعرف على الوجه، مما يجعل العملية سلسة وسريعة.
- **التكامل عبر الأجهزة** :تدعم Passkey المزامنة عبر السحابة من خلال خدمات مثل iCloud Keychainأو Google Password Manager، مما يتيح استخدامها عبر أجهزة متعددة دون التضحية بالأمان.
- **التوافق مع المعايير العالمية** :بفضل اعتمادها علىWebAuthn ، تتوافق Passkey مع معظم المتصفحات والأنظمة الحديثة، مما يجعلها قابلة للتطبيق على نطاق واسع.

تُعتبر Passkey تقنية مثالية لتطبيقات الدفع الإلكتروني، حيث تجمع بين الأمان العالي وتجربة المستخدم المبسطة، مما يجعلها مناسبة لتأمين المعاملات المالية الحساسة [3] .

4. أنظمة الدفع الإلكترونية:

تُشكل أنظمة الدفع الإلكتروني العمود الفقري للتجارة الإلكترونية، حيث تتيح تحويل الأموال بين المشتري والبائع بسرعة وأمان عبر الإنترنت. تعمل هذه الأنظمة من خلال بوابات دفع مثل Stripe، PayPal، Razorpay، وSquare، والتي تؤدي دور الوسيط بين العميل والبنك. تقوم هذه البوابات باستلام بيانات الدفع، مثل معلومات البطاقة الائتمانية أو الحساب البنكي، وتشفيرها باستخدام بروتوكولات أمان متقدمة مثل SSL/TLS، ثم إرسالها إلى البنك أو المؤسسة المالية للتحقق والمعالجة.

وظائف بوابات الدفع تشمل:

- **التحقق من صحة البيانات** :التأكد من أن معلومات الدفع المقدمة (مثل رقم البطاقة أو تاريخ الانتهاء) صحيحة ومطابقة للسجلات المصرفية.
- **تشفير البيانات الحساسة** :استخدام تقنيات تشفير قوية مثل AES-256وRSA لحماية البيانات أثناء النقل عبر الشبكات.
- **إشعارات فورية** :إرسال تأكيدات أو إشعارات برفض المعاملات في الوقت الفعلي لضمان تجربة مستخدم سلسة.
- **إدارة المخاطر** :استخدام أنظمة ذكية للكشف عن الاحتيال، مثل تحليل السلوك، اكتشاف الأنماط غير الطبيعية، وتطبيق قواعد الأمان الديناميكية.

تُدمج بوابات الدفع عادةً مع واجهات برمجة التطبيقات (APIs) لتوفير تجربة دفع متكاملة في المتاجر الإلكترونية وتطبيقات الهواتف الذكية. هذه الواجهات تتيح للمطورين تخصيص عمليات الدفع وفقًا لاحتياجات العملاء، مع ضمان التوافق مع المعايير الأمنية مثلPCI DSS معيار أمان بيانات صناعة بطاقات الدفع.

5. التحديات الأمنية في أنظمة الدفع

على الرغم من التطورات الكبيرة في تقنيات الدفع الإلكتروني، إلا أن هذه الأنظمة لا تزال تواجه تحديات أمنية معقدة تهدد سلامتها. من أبرز هذه التحديات:

هجمات التصيّد (Phishing) : حيث يتم خداع المستخدمين لإدخال بياناتهم الحساسة في مواقع أو صفحات مزيفة مصممة لتبدو مشابهة للمواقع الرسمية.

التلاعب بالبيانات أثناء النقل :يحدث هذا عند استخدام بروتوكولات تشفير ضعيفة أو عند استغلال ثغرات أمنية في البرمجيات أو الشبكات.

سرقة بيانات البطاقة :تحدث غالبًا عند استخدام مواقع غير موثوقة أو شبكات Wi-Fi عامة غير آمنة، مما يتيح للمهاجمين اعتراض البيانات.

الاحتيال في المصادقة :مثل كسر رموز التحقق لمرة واحدة (OTP) أو استخدام كلمات مرور مسروقة من خلال هجمات القوة الغاشمة (Brute Force) أو تسرب قواعد البيانات.

الحلول المقترحة لمواجهة هذه التحديات:

المصادقة المتقدمة :اعتماد تقنيات مثل Passkey و MFAلتعزيز الحماية ضد الاختراق.

التشفير المتقدم :استخدام بروتوكولات تشفير قوية مثل AES-256و RSAلحماية البيانات أثناء النقل والتخزين.

إشعارات الدفع اللحظية :تمكين المستخدمين من مراقبة المعاملات في الوقت الفعلي والتفاعل معها فورًا.

مبدأ الصلاحيات الدنيا: تقييد الوصول إلى البيانات الحساسة للحد الأدنى الضروري، مما يقلل من مخاطر التسريب [4]

6. نظام الإشعارات اللحظية ودوره في الأمان

تُعد الإشعارات اللحظية ركيزة أساسية في تعزيز أمان أنظمة الدفع الإلكتروني. تعتمد هذه الأنظمة على خدمات مثل Firebase Cloud Messaging (FCM)، Apple Push Notification Service (APNs)، أو خدمات مشابهة لإرسال تنبيهات فورية إلى المستخدم عند إجراء أي معاملة مالية. تشمل هذه الإشعارات تفاصيل دقيقة مثل قيمة المعاملة، تاريخ ووقت العملية، والجهة المستلمة، مما يتيح للمستخدم التحقق من صحة المعاملة على الفور.

دور الإشعارات اللحظية في تعزيز الأمان:

- **الكشف السريع عن الاحتيال** :تمكن الإشعارات المستخدم من اكتشاف أي معاملة غير مصرح بها فور حدوثها، مما يقلل من الوقت المتاح للمهاجم.
- **التفاعل الفوري** :تتيح للمستخدم القدرة على رفض المعاملة، الإبلاغ عنها، أو اتخاذ إجراءات فورية مثل حظر الحساب.
- **تعزيز الثقة** :توفر الإشعارات شعورًا بالتحكم والشفافية، مما يعزز ثقة المستخدم في النظام المالي.
- **تقليل الأخطاء البشرية** :من خلال إبلاغ المستخدم بكل معاملة، يمكن تجنب الأخطاء الناتجة عن عدم الانتباه أو الإهمال.

تُعتبر الإشعارات اللحظية خط الدفاع الأول في مواجهة الاحتيال، حيث تمنح المستخدم القدرة على التدخل السريع، مما يحد من الخسائر المحتملة ويعزز كفاءة النظام. [5]

7. تحليل علاقة النظام بإجراءات الحماية الثلاثة

وفقًا لتصنيف عالم الأمن السيبراني Bruce Schneier، يمكن تقسيم تدابير الحماية الأمنية إلى ثلاث فئات رئيسية:

- **المنع (Prevention) :**يهدف إلى منع وقوع الهجمات من خلال تطبيق تقنيات مثلPasskey ، التشفير القوي مثل AES وRSA، جدران الحماية(Firewalls) ، وسياسات الوصول المقيد .(Access Control Lists) هذه الإجراءات تُقلل من احتمالية اختراق النظام منذ البداية.
- **الكشف (Detection):** يركز على رصد الأنشطة المشبوهة أو غير المصرح بها من خلال أنظمة تسجيل الأحداث(Logging Systems) ، المراقبة في الوقت الفعلي، وأنظمة الإنذار المبكر التي تستخدم تحليل البيانات والذكاء الاصطناعي.
- **الرد (Response) :** يشمل اتخاذ إجراءات فورية عند اكتشاف خرق أمني، مثل حظر الحسابات المشتبه بها، تحليل سجلات الأحداث، وتفعيل خطط الطوارئ لاحتواء الضرر واستعادة النظام.

في سياق هذا المشروع، يلعب Passkey دورًا محوريًا في مرحلة المنع من خلال توفير مصادقة آمنة مقاومة للاختراق، مما يقلل من مخاطر الوصول غير المصرح به. أما الإشعارات اللحظية فتُعزز مرحلة الكشف من خلال تنبيه المستخدم فور حدوث أي نشاط مشبوه، مما يتيح التدخل السريع. فيما يتعلق بالرد، يوفر النظام أدوات مثل الحظر التلقائي للحسابات أو طلب تأكيد إضافي عند اكتشاف أنماط غير طبيعية، مما يضمن استجابة فعالة وفورية.

8. التحديات المرتبطة بتقنية Passkey

رغم المزايا المتعددة لتقنية Passkey، إلا أنها تواجه تحديات تقنية وتطبيقية قد تعيق تبنيها على نطاق واسع:

عدم دعم الأجهزة القديمة :تتطلب Passkey أجهزة تدعم تقنيات أمان متقدمة مثل TPMأو Secure Enclave، وهي غير متوفرة في الأجهزة القديمة أو منخفضة التكلفة.

الاعتماد على أنظمة تشغيل محدثة :تعتمد Passkey على أنظمة تشغيل حديثة مثل iOS 16 أو Android 12 وما فوق، مما يحد من استخدامها في الأجهزة التي لا تتلقى تحديثات.

التكامل مع الأنظمة المصرفية التقليدية :العديد من البنوك لا تزال تعتمد على أنظمة قديمة غير متوافقة مع معيار WebAuthn، مما يتطلب استثمارات كبيرة لتحديث البنية التحتية.

مخاطر فقدان الجهاز :بما أن المفتاح الخاص مخزن على الجهاز، فإن فقدان الجهاز أو تعطله قد يؤدي إلى صعوبة الوصول إلى الحسابات المرتبطة.

التحديات الثقافية والتعليمية :قد يواجه بعض المستخدمين صعوبة في فهم واستخدام Passkey بسبب قلة الوعي أو مقاومة التغيير إلى تقنيات جديدة.

الحلول المقترحة:

النسخ الاحتياطي السحابي :استخدام خدمات مثل iCloudأو Google Cloudلتخزين نسخ احتياطية مشفرة من المفاتيح، مما يتيح استعادتها في حالة فقدان الجهاز.

دعم الأجهزة المتعددة :تمكين ربط الحساب بأكثر من جهاز لضمان استمرارية الوصول.

تحديث البنية التحتية :تشجيع المؤسسات المالية على تبني معايير حديثة مثل WebAuthn من خلال تقديم حوافز أو دعم فني.

توعية المستخدمين :إطلاق حملات توعية لتثقيف المستخدمين حول فوائد Passkey وكيفية إعدادها واستخدامها بسهولة.

تطوير حلول هجينة :توفير خيارات مصادقة بديلة(مثل MFA التقليديةكحل مؤقت للأجهزة غير المدعومة)[6].

9. مقارنة بين طرق المصادقة التقليدية والحديثة:

- **كلمات المرور** :تُعتبر الخيار الأضعف بسبب قابليتها للاختراق من خلال هجمات القوة الغاشمة، إعادة الاستخدام، أو التسريب عبر قواعد بيانات غير آمنة.
- **رموز التحقق لمرة واحدة (OTP):** توفر أمانًا أفضل من كلمات المرور، لكنها عرضة لهجمات التصيّد أو اعتراض الرسائلمثل هجماتSIM Swapping
- **المصادقة البيومترية** :تقدم مستوى أمان عالٍ بفضل الاعتماد على خصائص فريدة للمستخدم، لكنها تعتمد على جودة الأجهزة وقد تُعرض للاختراق في حالات نادرة.
- **Passkey:** تجمع بين الأمان الفائق وسهولة الاستخدام، مع مقاومة شبه كاملة للاختراق بفضل اعتمادها على التشفير غير المتماثل والتكامل مع الأجهزة الآمنة.

10. تطور تقنيات الدفع الإلكتروني:

شهدت أنظمة الدفع الإلكتروني تطورًا كبيرًا على مدى العقود الماضية، بدءًا من استخدام بطاقات الائتمان التقليدية إلى ظهور المحافظ الرقمية والبوابات الإلكترونية. في البداية، كانت أنظمة الدفع تعتمد على بروتوكولات بسيطة مثل SET

(Secure Electronic Transaction)الذي طورته Visa و Mastercard في التسعينيات. مع تطور التكنولوجيا، ظهرت بوابات دفع حديثة مثل PayPal في أواخر التسعينيات، والتي قدمت مفهوم الدفع الإلكتروني السريع عبر الإنترنت.

في العقد الأخير، أدت التطورات في مجال الهواتف الذكية إلى ظهور حلول دفع مبتكرة مثل Apple Payو Google Pay، التي تعتمد على تقنيات الاتصال القريب (NFC) والمصادقة البيومترية. كما ساهمت تقنيات الذكاء الاصطناعي والبلوك تشين في تعزيز أمان وكفاءة المدفوعات، مما يعكس التطور المستمر في هذا المجال.

تأثير الأمان على تجربة المستخدم

يُعد تحقيق التوازن بين الأمان وتجربة المستخدم أحد التحديات الرئيسية في تصميم أنظمة الدفع الإلكتروني. الأنظمة التي تفرض إجراءات أمنية معقدة (مثل طلب إدخال عدة رموز تحقق أو كلمات مرور طويلة) قد تؤدي إلى إحباط المستخدمين وتقليل معدلات إتمام المعاملات. في المقابل، الأنظمة التي تركز على السهولة على حساب الأمان قد تُعرّض المستخدمين لمخاطر الاختراق.

تقنيات مثل Passkeyو MFA تقدم حلاً مثاليًا لهذا التحدي، حيث توفر أمانًا عاليًا مع تقليل التعقيد. على سبيل المثال، تتيح Passkey إتمام عملية تسجيل الدخول بنقرة واحدة باستخدام البصمة أو التعرف على الوجه، مما يعزز تجربة المستخدم دون التضحية بالأمان. بالإضافة إلى ذلك، تساهم الإشعارات اللحظية في تعزيز الشفافية وإشراك المستخدم في عملية المراقبة، مما يعزز شعوره بالتحكم والأمان.

11. مستقبل أنظمة الدفع الإلكتروني:

مع استمرار التطور التكنولوجي، من المتوقع أن تشهد أنظمة الدفع الإلكتروني تغيرات جذرية في المستقبل. تشمل الاتجاهات المستقبلية:

التوسع في استخدام البلوك تشين :تقنيات مثل العملات الرقمية (Cryptocurrencies) والعقود الذكية Smart Contractsقد توفر مستويات أعلى من الأمان والشفافية.

تكامل الذكاء الاصطناعي :استخدام الذكاء الاصطناعي في تحليل السلوك واكتشاف الاحتيال بشكل أكثر دقة وسرعة.

توسيع استخدام Passkey :مع زيادة دعم WebAuthn في الأجهزة والمتصفحات، من المتوقع أن تصبح Passkey المعيار الافتراضي للمصادقة.

المدفوعات عبر الإنترنت من الأشياء (IoT) :مع انتشار الأجهزة الذكية، قد تصبح الأجهزة مثل الساعات الذكية والسيارات الذكية منصات لإجراء المدفوعات.

هذه الاتجاهات تشير إلى مستقبل يركز على الأمان، السرعة، والتكامل السلس، مما يتطلب مواصلة البحث والتطوير لضمان مواكبة التحديات الأمنية الجديدة.

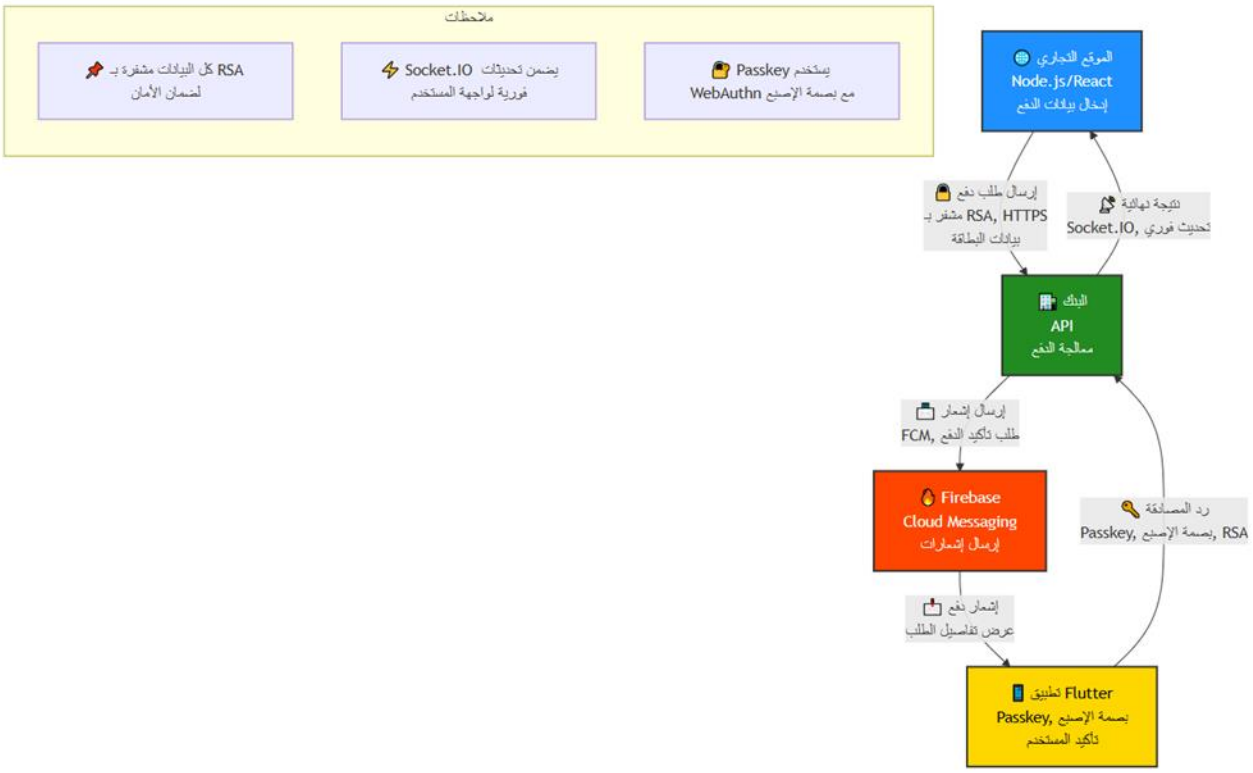
12. نظرة عامة على النظام:

يتكون النظام من ثلاثة مكونات رئيسية:

- **الموقع التجاري** :واجهة ويب تم تطويرها باستخدام Node.jsمع مكتبة Express.jsلمعالجة طلباتHTTP ، و htm,css, javascript واجهة المستخدم الأمامية. يتيح الموقع للمستخدمين التسجيل، تسجيل الدخول، اختيار المنتجات، وإتمام عمليات الدفع. يتم استخدام Socket.IO لتحديثات فورية لحالة الطلب
- **تطبيق الهاتف** :تم تطويره باستخدام Flutter بلغة .Dart يستقبل التطبيق إشعارات الدفع عبر Firebase Cloud Messaging (FCM)، ويدعم المصادقة عبر Passkeyوبصمة الإصبع باستخدام مكتبة .local_authيتيح للمستخدم قبول أو رفض طلبات الدفع.
- **الخدمات البنكية** :تعتمد على واجهات برمجية (APIs) لمعالجة المدفوعات. يتم إرسال طلبات الدفع من الموقع التجاري إلى الخادم البنكي، ويتم إرسال إشعارات التأكيد أو الرفض إلى تطبيق الهاتف عبر FCM .

13. ألية عمل المشروع:

المستخدم يدخل إلى الموقع التجاري من حسابه ثم يختار المنتجات التي يريد شراؤها وبعد انتهائه من ذلك يتأكد من مشترياته من صفحة ال order ثم يذهب إلى صفحة الشراء يقوم بإدخال معلوماته البنكية ويفتح اتصال websocket مع سرفر البنك ليبقى على اتصال لمعرفة حالة الرد بعد ذلك يقوم البنك بإرسال إشعار إلى التطبيق البنكي الخاص بالزبون الذي يفتح التطبيق عن طريق ال passkey ويوافق او يرفض الطلب بعد ذلك يعيد الرد الى البنك الذي بدورة يعيد الرد الى الموقع التجاري عن طريق الاتصال المفتوح سابقاً الآلية موضحة بالشكل التالي :



الشكل 1مخطط توضيحي لآلية عمل المشروع

الفصل الثاني الأدوات والتقنيات المستخدمة

1. لغة **Flutter** :

- **الوصف** Flutter :هو إطار عمل مفتوح المصدر تم تطويره من قبل شركةGoogle ، يُستخدم في بناء تطبيقات الهواتف المحمولة التي تعمل على أنظمة Android و iOSباستخدام قاعدة كود موحدة. يعتمد Flutter على لغةDart ، وهي لغة برمجة كائنية التوجه مصممة لتقديم أداء عالٍ من خلال الترجمة المسبقة (AOT) وكذلك الترجمة الفورية(JIT) ، مما يُمكّن المطورين من تجربة التطبيق بشكل مباشر أثناء التطوير باستخدام ميزة Hot Reload.يتميز Flutter بواجهاته الرسومية المبنية على مكونات تُسمّىWidgets ، والتي توفر تجربة مستخدم سلسة وقابلة للتخصيص بشكل كامل.
- **الاستخدام** :يُستخدم Flutter في تطوير تطبيق الهاتف الذكي الذي يتضمن عدّة خصائص متقدمة، منها: استقبال إشعارات Firebase Cloud Messaging (FCM) لإرسال التنبيهات إلى المستخدمين، وتوفير آليات تسجيل دخول آمنة مثل المصادقة باستخدام بصمة الإصبع من خلال مكتبة local_auth، إضافة إلى دعم

المصادقة بدون كلمات مرور باستخدام تقنية Passkey وفقًا لمعايير WebAuthn الحديثة، التي تعتمد على مفاتيح أمان عامة لتعزيز الأمان وسهولة الاستخدام.

2. لغة **Node.js** :

- **الوصف** Node.js :هي بيئة تشغيل مفتوحة المصدر مبنية على محرك V8 التابع لGoogle ، تُستخدم لتشغيل JavaScriptمن جهة الخادم .(Backend) تتيح Node.js بناء تطبيقات خفيفة وعالية الأداء تعتمد على نموذج غير متزامن (Asynchronous) ومبني على الأحداث(Event-driven) ، مما يجعلها مناسبة لتطبيقات الويب التي تحتاج إلى معالجة عدد كبير من الطلبات في الوقت الفعلي دون حظر .(Non-blocking I/O) تعتمد على مفهوم "single-threaded event loop" الذي يسمح بالتعامل مع المهام المتعددة بكفاءة.[7]
- **الاستخدام** :تُستخدم Node.js في تطوير الخادم الخاص بالموقع التجاري، مستفيدة من إطار العمل الشهير Express.js، الذي يُبسط إنشاء واجهات برمجية (APIs) ومعالجة طلبات HTTP بشكل مرن وسريع. كما تم دمج مكتبة Socket.IO لتوفير ميزة الاتصال في الوقت الحقيقي بين العميل والخادم، مما يتيح وظائف مثل التحديث الفوري للبيانات، الدردشة، أو تنبيهات النظام دون الحاجة لإعادة تحميل الصفحة أو إرسال طلبات متكررة.

3. **Socket.IO**:

- **الوصف** Socket.IO :هي مكتبة JavaScript تُستخدم لإنشاء قنوات اتصال في الوقت الفعلي (Real-time) بين العميل (Client) والخادم (Server) باستخدام بروتوكولWebSockets ، كما توفر دعمًا تلقائيًا لتقنيات نقل احتياطية (fallbacks) مثل HTTP long polling لضمان التوافق مع جميع البيئات. تُبنى هذه المكتبة على نموذج أحداث ثنائي الاتجاه(Bidirectional Event-Based Communication) ، ما يتيح إرسال واستقبال البيانات بشكل مباشر وسريع بين الطرفين دون الحاجة إلى إعادة تحميل الصفحة أو إنشاء طلبات HTTPمتكررة[8].
- **الاستخدام** :تُستخدم Socket.IO في النظام لتحديث حالة الطلبات بشكل لحظي، بحيث يتمكن المستخدم من معرفة نتيجة الطلب (الموافقة أو الرفض) فور حدوثها، دون الحاجة لإرسال طلب دوري للسيرفر .(Polling) هذا يسهم في تحسين تجربة المستخدم من خلال توفير استجابة أسرع وتقليل الضغط على الخادم.

4. **Firebase** :

- **الوصف** Firebase :هي منصة متكاملة لتطوير التطبيقات أطلقتهاGoogle ، توفّر العديد من الأدوات والخدمات التي تساعد المطورين في بناء تطبيقات عالية الجودة بسرعة وكفاءة. من أبرز ميزاتها Firebase Cloud Messagingلإرسال الإشعارات الفورية إلى أجهزة المستخدمين، وخدمة Realtime Database التي تسمح بتحديث البيانات بشكل لحظي بين العميل والخادم، إضافة إلى نظام مصادقة متكامل يدعم تسجيل الدخول باستخدام البريد الإلكتروني أو حسابات خارجية أو رقم الهاتف، مما يسهل إدارة المستخدمين ويوفر أماناً عالياً.
- **الاستخدام** :

تُستخدم Firebase في المشروع لإرسال الإشعارات الفورية إلى المستخدمين عبر خدمة FCM لإعلامهم بأي تحديث جديد. كما يتم الاستفادة من خدمات المصادقة لتسجيل دخول المستخدمين والتحقق من هويتهم بسهولة، مع إمكانية استخدام قاعدة البيانات الفورية لتخزين أو تحديث البيانات بشكل مباشر في الوقت الحقيقي دون الحاجة لإعادة تحميل التطبيق أو إرسال طلبات متكررة إلى الخادم.

5. **RSA** :

- الوصف RSA :هي خوارزمية تشفير غير متماثل تُستخدم على نطاق واسع لتأمين تبادل البيانات عبر الشبكات. تعتمد على زوج من المفاتيح، أحدهما عام يُستخدم لتشفير البيانات، والآخر خاص يُستخدم لفك التشفير. تعتمد RSA على صعوبة تحليل الأعداد الأولية الكبيرة، مما يجعلها آمنة وقوية في حماية المعلومات الحساسة مثل كلمات المرور والمفاتيح السرية وتفاصيل الهوية.[9]
- الاستخدام : في المشروع، تُستخدم خوارزمية RSA لتأمين تبادل المعلومات الحساسة بين العميل والخادم، مثل تبادل التحديات الأمنية أثناء المصادقة الثنائية. من خلال تشفير البيانات بالمفتاح العام، يمكن ضمان أن الشخص الوحيد القادر على فك تشفيرها هو من يملك المفتاح الخاص، مما يضيف طبقة حماية قوية ضد التلاعب أو التنصت.

6. **Passkey (WebAuthn)** :

- **الوصف** Passkey :هي تقنية مصادقة حديثة مبنية على معيارWebAuthn ، تتيح للمستخدمين تسجيل الدخول إلى التطبيقات والمواقع بدون الحاجة لاستخدام كلمات مرور. تعتمد على مفاتيح تشفير تُخزن بشكل آمن على جهاز المستخدم وتُستخدم للتوثيق بشكل مشفر، مما يعزز الأمان ويقلل من مخاطر السرقة أو التسريب مقارنة بكلمات المرور التقليدية.

- **الاستخدام** :في المشروع، يتم استخدام Passkey داخل تطبيق Flutter لتمكين مصادقة آمنة أثناء عمليات الدفع. من خلال هذه التقنية، يتم التأكد من هوية المستخدم باستخدام بيانات بيومترية (مثل بصمة الإصبع أو التعرف على الوجه) المرتبطة بمفتاح التشفير، مما يضمن تجربة استخدام سهلة وآمنة.

7. **VS Code** :

- **الوصف** Visual Studio Code :هو محرر نصوص وبيئة تطوير مدمجة (IDE) مفتوحة المصدر من تطوير Microsoft، يدعم لغات برمجة متعددة مثلJavaScript ، Dart، HTMLو .CSSيتميز بواجهة مرنة وخفيفة، ويوفر ميزات متقدمة مثل الإكمال التلقائي، إدارة المشاريع، والاندماج مع نظام التحكم بالإصدارات.Git
- **الاستخدام** Visual Studio Code :هو محرر نصوص وبيئة تطوير مدمجة (IDE) مفتوحة المصدر من تطوير Microsoft، يدعم لغات برمجة متعددة مثلJavaScript ، Dart، HTMLو .CSSيتميز بواجهة مرنة وخفيفة، ويوفر ميزات متقدمة مثل الإكمال التلقائي، إدارة المشاريع، والاندماج مع نظام التحكم بالإصدارات.Git

8. **Google Authenticator** :

الوصف: Google Authenticator هو تطبيق يُستخدم لتوليد رموز تحقق لمرة واحدة (OTP) تعتمد على الوقت باستخدام خوارزمية .(Time-based One-Time Password) TOTP يقوم بإنشاء رموز تحقق جديدة كل 30 ثانية، مما يوفر طبقة أمان إضافية عند تسجيل الدخول أو تنفيذ العمليات الحساسة مثل الدفع أو تعديل الحسابات.

الاستخدام : في هذا المشروع، تم دمج Google Authenticator لتفعيل ميزة التحقق الثنائي(2FA) ، حيث يتم إنشاء مفتاح سري (Secret Key) عند إعداد المصادقة، ويُستخدم لتوليد الرموز على التطبيق. ولأن المستخدم قد يفقد الوصول إلى جهازه أو يحذف التطبيق، تم التفكير في آلية استرداد الحساب عبر التحقق من بيانات إضافية مثل البريد الإلكتروني ورقم الهاتف، مما يتيح استعادة الوصول دون فقدان الأمان أو الحاجة لإعادة تهيئة المفتاح.

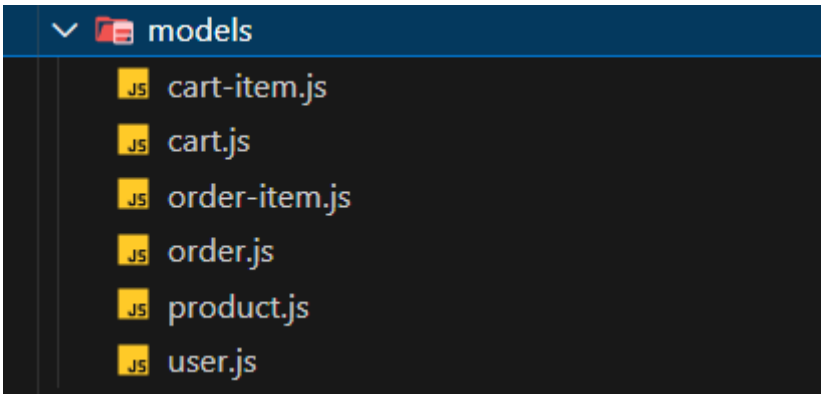
الفصل الثالث: التنفيذ العملي والاختبارات

1. بناء الموقع التجاري:

- تم بناء الموقع التجاري باستخدام نمط التصميم البرمجي **MVC (Model–View–Controller)**، وهو أحد أكثر الأنماط شيوعًا في هندسة البرمجيات، ويُستخدم على نطاق واسع في تطوير تطبيقات الويب والهواتف لقدرته العالية على تنظيم الكود، وفصل المهام بين أجزاء التطبيق المختلفة. يتميز هذا النمط بتقسيم التطبيق إلى ثلاث طبقات أساسية :النموذج (Models) و (Views) و(Controleer).

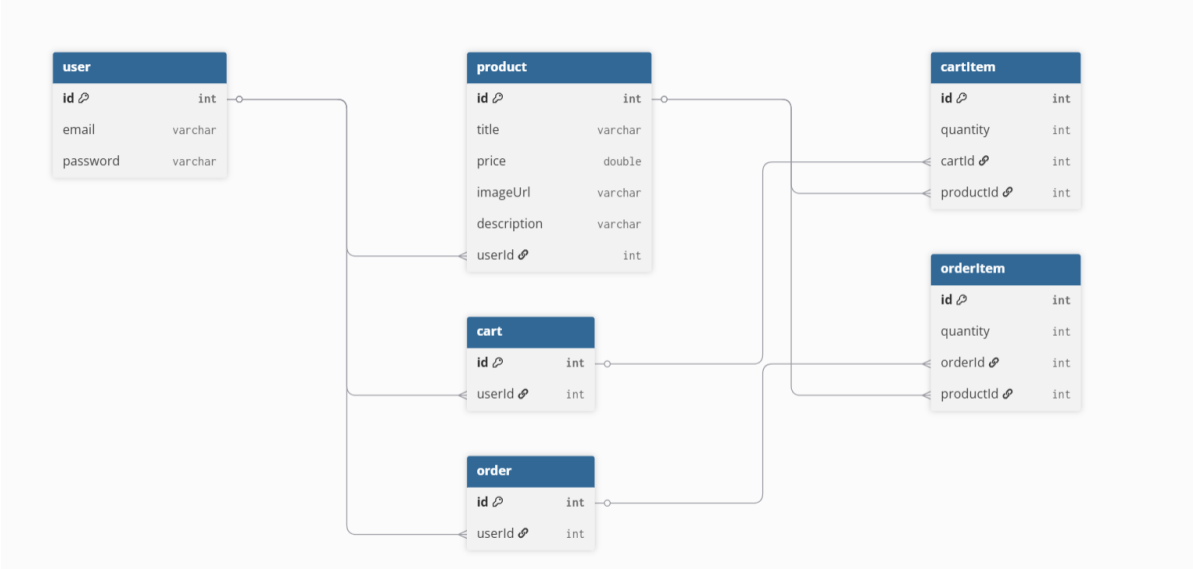
Model : الجزء المسؤول عن التعامل المباشر مع قاعدة البيانات. حيث تم إنشاء نماذج باستخدام تقنية ORM (Object-Relational Mapping)لتجسيد جداول النظام مثل جداول المنتجات، الطلبات، والمستخدمين، ما يتيح إمكانية إجراء عمليات القراءة، الإضافة، التعديل، والحذف (CRUD) بطريقة منظمة دون الحاجة لكتابة استعلامات SQL يدوية معقدة. هذه الطبقة تعمل كوسيط ذكي بين منطق التطبيق

وبياناته المخزنة، ما يضمن سلامة البيانات وتناسقها



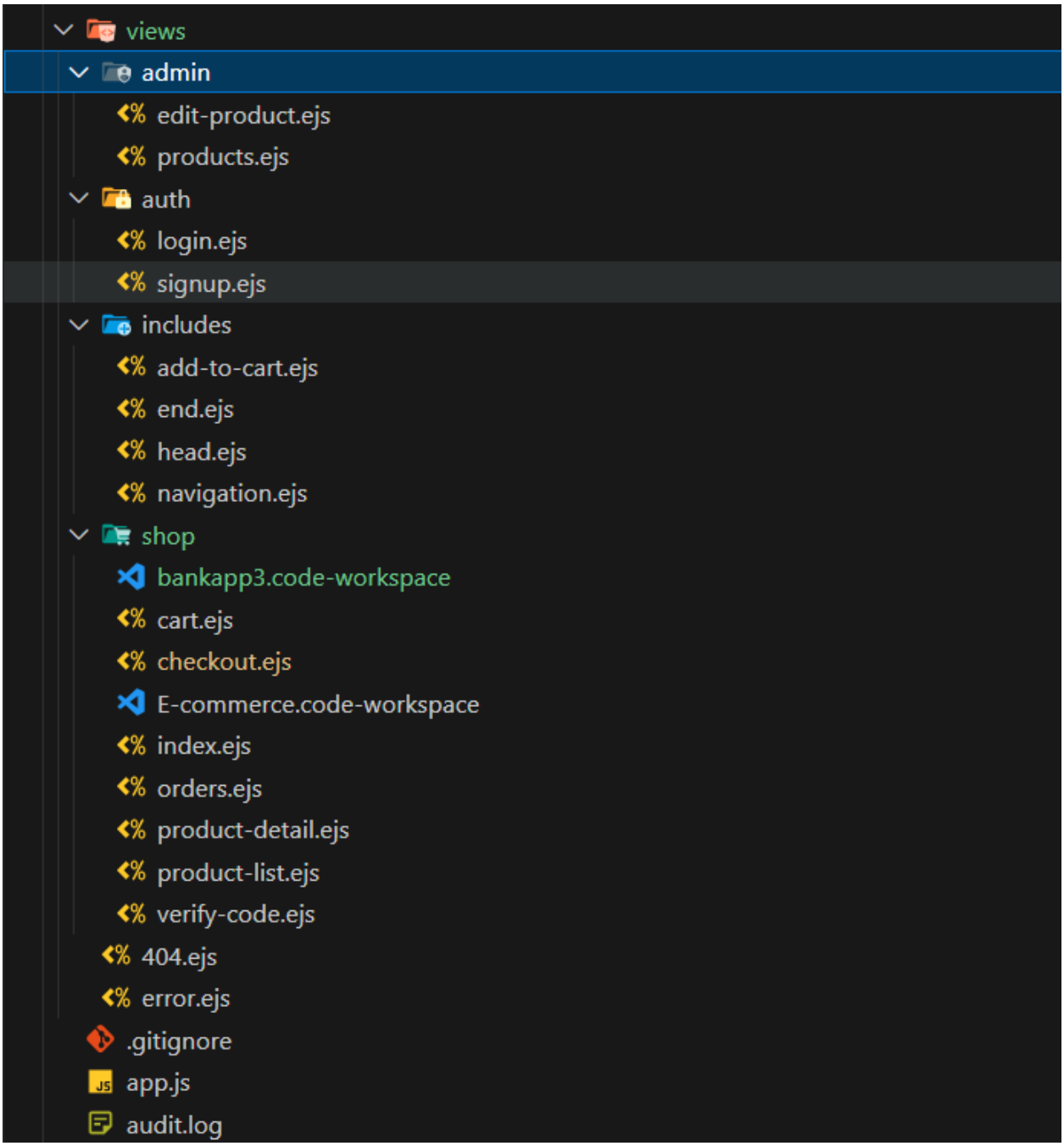
الشكل 2 يوضح بنية ال **model** في الموقع التجاري

العلاقات بين الجداول ممثلة كالتالي :



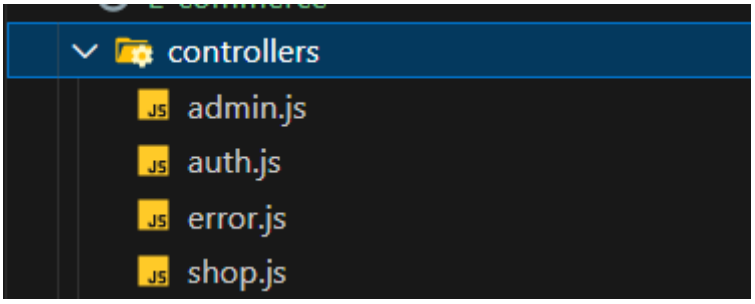
الشكل 3 بنية توضيحية لجداول البيانات ولعلاقات بينها

- **View:**تفهي مسؤولة عن عرض البيانات وتوفير واجهة تفاعل سهلة وسلسة للمستخدم. في هذا النظام، تشمل الواجهات صفحات مثل تسجيل الدخول، عرض قائمة المنتجات، تفاصيل المنتج، سلة الشراء، وواجهة الدفع .تم تصميم هذه الصفحات باستخدام تقنيات الويب القياسية مثل HTML، CSS، و JavaScriptلتقديم تجربة استخدام بصرية جذابة واستجابة عالية للأوامر، ما يضمن تفاعل المستخدم مع النظام بطريقة سلسة وواضحة.



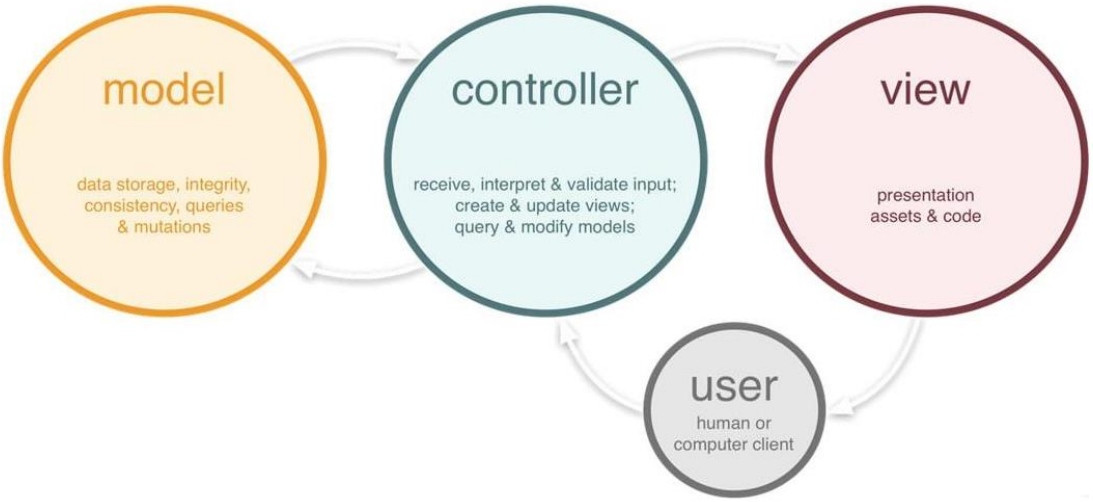
الشكل 4 بنية الواجهات للموقع التجاري

- **Controller** : فهي العقل الذي يربط بين الـ View والـ Model. حيث تستقبل هذه الطبقة الطلبات من المستخدم (مثل تسجيل الدخول أو تأكيد الطلب)، وتعالجها باستخدام المنطق البرمجي المطلوب، ومن ثم تطلب البيانات المناسبة من الـ Model أو تُرسل له التحديثات اللازمة، وتعيد عرض النتائج على الواجهة المناسبة عبر الـ View. هذا الفصل يتيح سهولة في اختبار الكود وتحديد موقع الأخطاء أو التعديلات



الشكل 5 يوضح بنية الـ **controller** في الموقع التجاري

يساعد هذا الهيكل التنظيمي على **فصل الاهتمامات(Separation of Concerns)** ، مما يسمح لفريق التطوير بالعمل على كل طبقة بشكل مستقل دون التأثير على الطبقات الأخرى. على سبيل المثال، يمكن لفريق التصميم تعديل واجهات المستخدم دون المساس بمنطق البيانات، كما يمكن لفريق قاعدة البيانات تحسين الأداء أو تغيير الهيكل دون التأثير على طريقة عرضها. بالإضافة إلى ذلك، يُسهل هذا النمط من التصميم توسيع المشروع مستقبلًا سواء بإضافة ميزات جديدة، أو دعم المزيد من الواجهات، أو دمج تقنيات حديثة، دون الحاجة لإعادة بناء النظام من الصفر.



الشكل 6 يوضح بنية النمط المعماري **mvc**

2. بناء التطبيق

- تم اعتماد نمط Clean Architecture في بناء تطبيق الهاتف باستخدام Flutter، والذي يُعد من أقوى الأنماط المعمارية الحديثة في تطوير البرمجيات، وخصوصًا التطبيقات الكبيرة والمعقدة التي تتطلب تنظيمًا عاليًا وقابلية للصيانة على المدى الطويل. يوفر هذا النمط فصلًا صارمًا بين منطق العمل، عرض البيانات، ومصادر البيانات، مما يسهل اختبار كل طبقة بشكل مستقل، ويمنح المطورين مرونة كبيرة عند تطوير أو توسيع المشروع في المستقبل.

تتكون البنية المعمارية للتطبيق من ثلاث طبقات رئيسية، تم تنظيمها وفقًا لمبدأ الاعتماد العكسي (Dependency Rule)، بحيث تعتمد الطبقات الخارجية على الطبقات الداخلية، وليس العكس. التفاصيل كما يلي :

- **Domain Layer**

تُعد القلب الحقيقي للتطبيق، حيث تحتوي على الكيانات (Entities) التي تعبّر عن المفاهيم الأساسية في المشروع مثل User، PaymentRequest، TransactionResult.كما تتضمن واجهات مجردة (Abstract Repositories) تمثل العقود التي تضمن تفاعل منطق العمل مع مصادر البيانات دون التقيد بتفاصيل التنفيذ، مما يتيح استبدال أو تعديل مصدر البيانات بسهولة (مثل التبديل من Firebase إلى REST API دون التأثير على منطق العمل).كما تحتوي هذه الطبقة على Use Cases التي تمثل منطق الأعمال (Business Logic) مثل تسجيل المستخدم، إرسال الطلبات، وتأكيد الدفع، مما يجعل هذه الطبقة قابلة للاختبار والوحدة (Unit Test) بشكل مباشر.

- **Data Layer**

تُعنى هذه الطبقة بتنفيذ الواجهات المعرّفة في طبقة Domain، حيث تحتوي على الـ Repositories الفعلية التي تتولى التعامل المباشر مع مصادر البيانات المختلفة. من أبرز هذه المصادر: Firebase، والذي يُستخدم في إرسال الإشعارات الفورية عبر خدمة FCM؛ وWeb API، الذي يُستخدم للتكامل مع خوادم البنك من أجل تنفيذ عمليات التحقق والدفع؛ بالإضافة إلى Local Storage لتخزين بيانات المستخدم محليًا بشكل مؤقت وتحسين أداء التطبيق في حال عدم الاتصال بالإنترنت.كما تتضمن هذه الطبقة آليات تحويل البيانات (Data Mappers)، والتي تقوم بدور حيوي في تحويل الكائنات القادمة من الـ API (التي تكون عادةً بصيغة

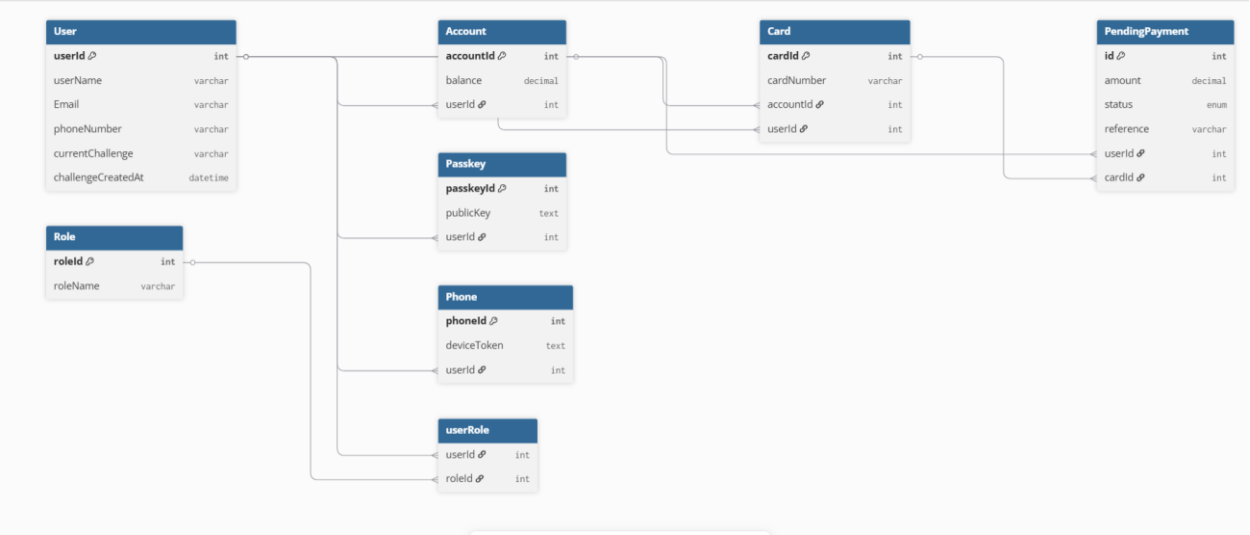
DTOs) إلى كيانات مفهومة تتماشى مع منطق العمل في طبقة Domain، والعكس كذلك. هذا الفصل بين هيكل البيانات الخارجي والداخلي يسهم في الحفاظ على بنية مرنة ومستقلة بين الطبقات المختلفة.

- **Presentation Layer**

تم بناء الواجهات الرسومية باستخدام Flutter، مع الاعتماد على نمط Bloc و Providerلإدارة الحالة بكفاءة، مع الحفاظ على انفصال تام بين منطق التطبيق وواجهة المستخدم.

تتفاعل هذه الطبقة مع Use Casesفي طبقة Domain فقط، مما يمنع أي ارتباط مباشر بمصادر البيانات أو التنفيذات الداخلية، وبالتالي تُعد هذه الطبقة سهلة التبديل والتحديث دون التأثير على بقية النظام.

تمثيل ال Database بهذا الشكل مع العلاقات فيما ببينها:



الشكل 7 يوضح جدول بيانات البنك والعلاقات فيما بينها

3. دورة حياة إرسال طلب الشراء :

بعد إدخال بيانات الدفع، يتم إرسال الطلب إلى الخادم البنكي عبر واجهة برمجية (API) مشفرة و .HTTPS يقوم البنك باستقبال الطلب ومعالجته باستخدام واستخراج device Token المقابل للبطاقة البنكية وارسال اشعار لصاحب البطاقة وفي الوقت ذاته يتم فتح اتصال webSocket مع سرفر البنك لمعرفة نتيجة الطلب في الصور المرفقة عملية ارسال المعطيات واستقبالها ومعالجتها من قبل البنك.

1.3 بدء عملية الشراء وإرسال معلومات الدفع والبطاقة البنكية:

بعد إدخال المعلومات الشخصية ورقم البطاقة يتم إرسال معلومات من الموقع التجاري إلى البنك من خلال التابع التالي:

```
const response = await fetch("http://localhost:3001/api/process-payment", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
    Accept: "application/json",
  },
  credentials: "include",
  body: JSON.stringify(paymentData),
});

console.log(
  response.status,
  response.statusText
);

if (!response.ok) {
  const errorData = await response.json().catch(() => ({}));
  const errorMsg = errorData.message || `خطأ في الخادم: ${response.status}`;
  throw new Error(errorMsg);
}

const result = await response.json();

if (!result.success) {
  throw new Error(result.message || "فشل إرسال طلب الدفع");
}

alert(
  "تم إرسال طلب الدفع بنجاح، بانتظار موافقة صاحب البطاقة من التطبيق."
);
```

الشكل 8 تابع يوضح طريق إرسال البيانات من الموقع التجاري إلى البنك

2.3 استقبال المعلومات من طرف البنك:

بعد استقبال المعلومات يتم التأكد من وجود البطاقة واستخراج ال device Token المقابل لصاحبها ويتم ارسال إشعار إلى هاتف العميل وإرسال رقم إلى المتصفح الخاص بالموقع التجاري حتى يتمكن من الانضام إلى الغرفة المحددة والاستماع إلى التحديثات من خلال اتصل websocket حتى يعلم اذا تم قبول طلب الشراء من خلال الاتصال المفتوح سابقا مع البنك الآلية موضحة من خلال الكود التالي:

```
exports.processPayment = async (req, res) => {
  try {
    const { cardNumber, amount, username, email } = req.body;
    const paymentAmount = parseFloat(amount);

    if (!cardNumber || !amount) {
      return res.status(400).json({ success: false, message: 'البيانات ناقصة' });
    }

    const card = await Card.findOne({
      where: { cardNumber },
      include: [{ model: Account }],
    });

    if (!card || !card.account) {
      throw new Error('بطاقة أو حساب غير صالح');
    }

    const userId = card.account.userId;

    const deviceTokenEntry = await Phone.findOne({ where: { userId } });

    if (!deviceTokenEntry || !deviceTokenEntry.deviceToken) {
      throw new Error('لا يوجد توكن للمستخدم');
    }

    const userDeviceToken = deviceTokenEntry.deviceToken;
    const transactionId = "1";

    await admin.messaging().send({
      token: userDeviceToken,
      notification: {
        title: 'طلب دفع جديد',
        body: `هل توافق على دفع ${paymentAmount}؟`,
      },
```

الشكل 9 تابع لاستقبال البيانات في البنك

```
      data: {
        userId: userId.toString(),
        amount: paymentAmount.toString(),
        cardNumber,
        transactionId,
      },
    });

    await PendingPayment.create({
      transactionId,
      userId,
      cardNumber,
      amount: paymentAmount,
      status: 'pending',
    });

    res.json({
      success: true,
      message: 'تم إرسال الإشعار، بانتظار موافقة المستخدم.',
      transactionId,
    });

  } catch (err) {
    res.status(500).json({ success: false, message: err.message });
  }
};
```

الشكل 10 تتمة التابع المسؤول عن استقبال البيانات

3.3 استقبال إشعار الدفع

يجب تهيئة تطبيق العميل لإستلام الإشعارات عن طريق ال Firebase من أجل الإستماع إلى للإشعارات في حال كان التطبيق مغلق أو مفتوح أو يعمل في الخلفية تتم الآلية من خلال الكود التالي :

```
Run | Debug | Profile
Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  await init();

  FirebaseMessaging.onMessage.listen(_handleNotification);
  FirebaseMessaging.onMessageOpenedApp.listen(_handleNotification);
  FirebaseMessaging.instance.getInitialMessage().then((message) {
    if (message != null) {
      _handleNotification(message);
    }
  });

  runApp(
    MultiBlocProvider(
      providers: [
        BlocProvider<NotificationBloc>(
          create: (_) => sl<NotificationBloc>()..add(LoadNotifications()),
        ), // BlocProvider
        BlocProvider<AlertBloc>(create: (_) => sl<AlertBloc>()),
      ],
      child: const MyApp(),
    ), // MultiBlocProvider
  );
}
```

الشكل 11 يوضح آلية استقبال الإشعارات في التطبيق

4.3 طلب الإنضمام إلى غرفة معينة

بعد فتح الاتصال وتهيئة ال websocket يجب طلب الإنضمام إلى غرفة معينه من خلال ال treanscatioId المرسل سابقا يطلب سرفر العميل العميل النضمام إلى الغرفة لمتالعة حالتها وذلك موضح من خلال الكود التالي :

```
function listenToPaymentResult(transactionId) {
  const socket = io("http://localhost:3001");

  socket.on("connect", () => {
    socket.emit("join-transaction", transactionId);
  });

  socket.on("payment-result", (data) => {

    if (data.status === "approved") {
      alert("تمت الموافقة على الدفع!");
    } else if (data.status === "rejected") {
      alert("تم رفض الدفع.");
    }

    socket.disconnect();
  });

  socket.on("disconnect", () => {
  });
}
```

الشكل 12 يوضح آلية فتح اتصال **websocket** من طرف الموقع التجاري

وبالمقابل على سرفر البنك يجب اضافة الاعدادت الخاصة بال webSocket

```
const activeSockets = {};

function setupSocket(io) {
  io.on('connection', (socket) => {
    console.log('🔌 New socket connected:', socket.id);

    socket.on('join-transaction', (transactionId) => {
      console.log(`📦 Joining transaction room: ${transactionId}`);
      socket.join(transactionId);
      activeSockets[transactionId] = socket;
    });

    socket.on('disconnect', () => {
      console.log(`🔌 Socket disconnected: ${socket.id}`);
    });
  });
}

module.exports = { setupSocket, activeSockets };
```

Figure 13 تهيئة اتصال واعداد **web socket** من طرف خادم البنك

5.3 الموافقة على الإشعار أو الرفض بعد توقيع التحدي

عند استقبال الإشعار وفتح التطبيق يتوجب علينا أن نقوم بادخال البصمة من أجل جلب المفتاح الخاص المخزن بطريقة آمنة الذي جرى توليدة عند عملية انشاء الحساب وتخزينه باستخدام Flutter secure storage

آلية تولبد المفتاح العام والخاص تتم عند عملية إنشاء الحساب يتم استخدام خوارزمية RSA وتحويلهم إلى الضيغة القياسية

```dart
final secureRandom = FortunaRandom();
final seedSource = Random.secure();
secureRandom.seed(
  KeyParameter(
    Uint8List.fromList(List.generate(32, (_) => seedSource.nextInt(256)))
  ), // KeyParameter
);

final keyGen = RSAKeyGenerator();
keyGen.init(
  ParametersWithRandom(
    RSAKeyGeneratorParameters(BigInt.parse('65537'), 2048, 64),
    secureRandom,
  ), // ParametersWithRandom
);

final keyPair = keyGen.generateKeyPair();
final publicKey = keyPair.publicKey as RSAPublicKey;
final privateKey = keyPair.privateKey as RSAPrivateKey;

final publicKeyPem = encodePublicKeyToPemPKCS1(publicKey);

final privateKeyPem = encodePrivateKeyToPemPKCS1(privateKey);

try {
  final asn1Parser = ASN1Parser(
    base64Decode(
      privateKeyPem
          .replaceAll('-----BEGIN RSA PRIVATE KEY-----', '')
          .replaceAll('-----END RSA PRIVATE KEY-----', '')
          .replaceAll('\n', ''),
    ),
  );
```

الشكل 14 يوضح كيفية توليد المفتاح العام والخاص باستخدام خوارزمية **RSA**

عندما يريد المستخدم ان يدخل للتطبيق يرسل سيرفر البنك تحدي له من خلال التابع التالي

```
exports.loginStart = async (req, res) => {
    try {
        const { userId } = req.body;
        if (!userId) {
            return res.status(400).json({ error: 'User ID required' });
        }

        const user = await User.findOne({
            where: { userId },
            include: [Passkey]
        });

        if (!user) {
            return res.status(404).json({ error: 'User not found' });
        }

        const passkey = await Passkey.findOne({ where: { userId: user.userId } });
        if (!passkey) {
            return res.status(400).json({ error: 'No passkey registered for this user' });
        }

        const challenge = crypto.randomBytes(32).toString('base64url');
        const challengeCreatedAt = new Date();

        await user.update({
            currentChallenge: challenge,
            challengeCreatedAt: challengeCreatedAt
        });

        res.json({ challenge });
    } catch (error) {
        console.error('Login start error:', error);
        res.status(500).json({ error: 'Internal server error' });
    }
};
```

الشكل 15 يوضح طريق إرسال التحدي من البنك لتطبيق الهاتف

يقوم المستخدم بإرسال التوقيع من خلال التابع التالي بعد التحقق من بصمة الإصبع

```dart
class LoginBloc extends Bloc<LoginEvent, LoginState> {
  Future<void> _onLoginCompleted(
    LoginCompleted event,
    Emitter<LoginState> emit,
  ) async {
    emit(LoginLoading());
    try {
      final bool authenticated = await localAuth.authenticate(
        localizedReason: 'قم بالمصادقة لتسجيل الدخول',
        options: const AuthenticationOptions(biometricOnly: true),
      );

      if (!authenticated) {
        emit(LoginError('فشلت المصادقة البيومترية'));
        return;
      }

      final privateKeyString = await localDataSource.getPrivateKey(
        event.userId,
      );
      if (privateKeyString == null) {
        emit(LoginError('لم يتم العثور على المفتاح الخاص'));
        return;
      }

      final privateKey = _parsePrivateKeyFromPem(privateKeyString);

      String challengeString = event.challenge;
      challengeString = challengeString.replaceAll('=', '');

      debugPrint('Challenge to sign (without padding): $challengeString');

      final challengeBytes = utf8.encode(challengeString);

      debugPrint(
        'Encoded Challenge Bytes (UTF-8): ${challengeBytes.toString()}',
      );
```

الشكل 16 آلية توقيع التحدي بهستخدام المفتاح الخاص

```
    // Sign challenge
    final signer = RSASigner(SHA256Digest(), '0609608648016503040201');
    signer.init(true, PrivateKeyParameter<RSAPrivateKey>(privateKey));
    final signature = signer.generateSignature(challengeBytes).bytes;
    final signatureBase64 = base64Encode(signature);

    debugPrint('Generated Signature: $signatureBase64');

    final result = await loginFinishUseCase.execute(
      userId: event.userId,
      signature: signatureBase64,
      deviceToken: event.deviceToken ?? '',
    );

    emit(
      result.fold((error) => LoginError(error), (authResponse) {
        localDataSource.cacheToken(authResponse.token);
        return LoginSuccess(authResponse: authResponse);
      }),
    );
  } catch (e, stacktrace) {
    emit(LoginError('حدث خطأ أثناء تسجيل الدخول: $e'));
  }
}
```

الشكل 17تتمة آلية توقيع التحدي بستخدام المفتاح الخاص

بعد الموافقة على الطلب يرسل الرد إلى سرفر البنك الذي بدوره يعمل على رد الإجابة إلى الموقع التجاري من خلال اتصال webSocket المفتوح سابقاً.

```
exports.approvePayment = async (req, res) => {
  try {
    const { userId, amount, cardNumber } = req.body;
    const paymentAmount = parseFloat(parseFloat(amount).toFixed(2));
     transactionId="1";


    if (!userId || !amount || !cardNumber || !transactionId) {
      return res.status(400).json({ success: false, message: 'بيانات ناقصة' });
    }

    const card = await Card.findOne({
      where: { cardNumber },
      include: [{ model: Account, required: true }],
    });
    if (!card) {
      throw new Error('تفاصيل البطاقة غير صحيحة');
    }

    console.log(JSON.stringify(card, null, 2));

    if (!card.account) {
      throw new Error('البطاقة غير مرتبطة بحساب صالح');
    }

    if (!card) throw new Error('تفاصيل البطاقة غير صحيحة');

    if (card.account.balance < paymentAmount) {
      throw new Error('الرصيد غير كافٍ');
    }

  await sequelize.query(
  'UPDATE accounts SET balance = balance - :amount WHERE accountId = :accountId',
  {
    replacements: {
      amount: paymentAmount,
      accountId: card.account.accountId,
```

الشكل 18 تابع يوضح طريقة خصم النقود بعد الموافقة على الطلب

```
  await sequelize.query(
  'UPDATE accounts SET balance = balance - :amount WHERE accountId = :accountId',
  {
    replacements: {
      amount: paymentAmount,
      accountId: card.account.accountId,
    },
    type: sequelize.QueryTypes.UPDATE,
  }
);
    await card.account.reload();

    const io = req.app.get('io');
    io.to(transactionId).emit('payment-result', {
      status: 'approved',
      transactionId,
      amount: paymentAmount,
    });

    await PendingPayment.update(
      { status: 'approved' },
      { where: { transactionId } }
    );

    res.json({ success: true, message: 'تمت الموافقة على العملية' });

  } catch (err) {
    res.status(500).json({ success: false, message: err.message });
  }
};
```

Figure 19 تتمة تابع يوضح طريقة خصم النقود بعد الموافقة على الطلب مع ارسال النتيجة غلى الموقع التجاري

6.3 عملية إسترداد الحساب

إحدى أبرز سلبيات استخدام ال Passkey هي أن عند حذف التطبيق من الهاتف يتم فقدان المفتاح الخاص ولايمكن توقيع التحديات المرسلة من السرفر فكان لا بد من طريق آمنة لاسترداد الحساب من خلال استخدام تطبيق google authenticator الذي يعمل على تغيير الرمز كل 30 ثانية ويجب على المستخدم أن يقوم بتفعيل خيارة المصادقة الثنائية داخل التطبيق يتم توليد مفتاح مرتبط بتطبيق المصادقة من خلال مكتبة speakeasy وذلك يتم من خلال الكود التالي :

```
exports.secret = async (req, res) => {
  const { email } = req.body;

  if (!email) return res.status(400).json({ error: 'Missing email' });

  const user = await User.findOne({ where: { email } });
  if (!user) return res.status(404).json({ error: 'User not found' });

  const secret = speakeasy.generateSecret({ length: 20 });

  console.log('Generated TOTP Secret:', secret.base32);
  console.log('OTPAuth URL:', secret.otpauth_url);

  await user.update({ totpSecret: secret.base32 });
  res.json({
    secret: secret.base32,
    otpauthUrl: secret.otpauth_url,
  });
};
```

الشكل 20 آلية توليد المقتاح السري للربط مع تطبيق المصادقة

عند عملية الإسترداد يجب أن يرسل المستخدم الرمز الموجود حاليا داخل تطبيق المصادقة ويتم استقبالة ومعالجة الطب من خلال :

```
exports.verify = async (req, res) => {
  const { email, phoneNumber, otp, publicKey } = req.body;
  if (!email || !phoneNumber || !otp || !publicKey) {
    return res.status(400).json({ error: 'Missing required fields' });
  }
  try {
    const user = await User.findOne({
      where: { email: email, phoneNumber: phoneNumber }
    });

    if (!user || !user.totpSecret) {
      return res.status(400).json({ error: '2FA not set up for user' });
    }
    const expectedToken = speakeasy.totp({
      secret: user.totpSecret,
      encoding: 'base32'
    });

    const isValid = speakeasy.totp.verify({
      secret: user.totpSecret,
      encoding: 'base32',
      token: otp,
      window: 1,
    });

    if (!isValid) {
      return res.status(401).json({ success: false, error: 'Invalid OTP' });
    }
    const transaction = await sequelize.transaction();

    try {
      if (!publicKey.includes('-----BEGIN PUBLIC KEY-----')) {
        await transaction.rollback();
        return res.status(400).json({ error: 'Invalid public key format' });
      }
      const existingUser = await User.findOne({
        where: { Email: email },
```

الشكل 21 يوضح آلية التحقق من الرمز من أجل استرداد الحساب

```
    if (!existingUser) {
      await transaction.rollback();
      return res.status(404).json({ error: 'User not found' });
    }
    let passkey = await Passkey.findOne({
      where: { userId: existingUser.userId },
      transaction,
    });

    if (passkey) {
      await passkey.update({ publicKey }, { transaction });
    } else {
      await Passkey.create({
        userId: existingUser.userId,
        publicKey: publicKey,
      }, { transaction });
    }
    await transaction.commit();
    return res.status(200).json({
      success: true,
      user: {
        id: existingUser.userId,
        username: existingUser.userName,
        email: existingUser.Email
      },
      phoneNumber: phoneNumber
    });
  } catch (dbError) {
    await transaction.rollback();
    console.error('Database error:', dbError);
    return res.status(500).json({ error: 'Failed to update public key' });
  }
} catch (error) {
  console.error('Error verifying OTP:', error);
  return res.status(500).json({ error: 'Internal server error' });
}
```

الشكل 22 تتمة يوضح آلية التحقق من الرمز من أجل استرداد الحساب

الفصل الرابع خاتمة المشروع

في عالمٍ تتسارع فيه الابتكارات التقنية، ويزداد فيه الاعتماد على الخدمات الرقمية يوماً بعد يوم، لا يعود تأمين المعاملات المالية ترفاً، بل ضرورة وجودية لأي منظومة مصرفية أو تجارية تطمح للاستمرارية والمنافسة. من هنا، شكّل مشروعنا استجابة حقيقية لهذا التحدي، من خلال تقديم نظام دفع إلكتروني متكامل يجمع بين الأمان، السرعة، وسلاسة الاستخدام.

لقد مزج هذا المشروع بين الجوانب النظرية العميقة والتطبيق العملي الدقيق، مع اعتماد أحدث ما توصلت إليه تقنيات المصادقة مثل Passkey، والتي أعادت تعريف مفاهيم الأمان والهوية الرقمية. وتمكّنا من بناء بنية تحتية فعالة تتكوّن من موقع تجاري، بوابة دفع مصرفية، وتطبيق جوّال مبني على Flutter، يتفاعل مع المستخدم لحظةً بلحظة من خلال إشعارات آنية تعتمد على .Firebase Cloud Messaging

وخلال مراحل التطوير والاختبار، أثبت النظام كفاءته في محاكاة سيناريوهات حقيقية شملت الموافقة على الطلب، الرفض، وفشل المصادقة، مع تقديم أداء مستقر وتجربة استخدام راقية تعكس النضج التقني والتكامل بين المكونات.

لكن هذه ليست النهاية، بل مجرد البداية... إذ يُعدّ هذا المشروع نواة صلبة يمكن البناء عليها لتطوير أنظمة دفع أكثر ذكاءً، وأكثر تكاملاً مع الذكاء الاصطناعي، تحليل السلوك، وسلاسل الكتل .(Blockchain) نطمح من خلال هذا العمل إلى أن نكون قد وضعنا لبنة جديدة في صرح التحول الرقمي الآمن، وأثبتنا أن الجمع بين الإبداع والوعي الأمني ليس فقط ممكناً، بل ضرورة في زمن تتسارع فيه المخاطر مثلما تتسارع فيه التقنية.

4. المراجع

[1] A. Alzahrani & R. Kumar, 2019. Security and Privacy Issues in E-Commerce: A Survey. International Journal of Computer Applications, Vol. 178, No. 7, pp.1-6

[2] F. Hao & P. Ryan, 2016. J-PAKE: Authenticated Key Exchange without PKI. ACM TISSEC, Vol.14, No.1, Article 4, pp.1–25.

[3] L. T. Nguyen & S. Lee, 2023. Passkeys and the Future of Passwordless Authentication: A WebAuthn-Based Approach. Proc. of the 2023 Int. Conf. on Cybersecurity and Trust, pp.101–109.

[4] M. Conti, N. Dragoni, & V. Lesyk, 2016. A Survey of Man In The Middle Attacks. IEEE Comm. Surveys & Tutorials, Vol.18, No.3, pp.2027–2051.

[5] M. G. Jaatun, S. Pearson, & C. Bernsmed, 2018. Security and Privacy in Mobile Push Notification Services. Proc. of the 2018 Int. Conf. on Cloud Computing and Services Science (CLOSER), pp.73–80.

[6] T. Metzger, A. Krombholz, & M. Smith, 2023. Adoption Barriers of Passwordless Authentication: A Study on the Perception of Passkeys and FIDO2. Proc. of the 32nd USENIX Security Symposium, pp.1047–1064.

[7] Tilkov, S., & Vinoski, S. (2010). *Node.js: Using JavaScript to build high-performance network programs*. IEEE Internet Computing, 14(6), 80–83.

[8] Karan, C. (2017). *Real-time Web Application Development using Socket.IO*. In *Building Real-Time Web Applications with ASP.NET Core SignalR* (pp. 135–150). Apress.

[9]Tilkov, S., & Vinoski, S. (2010). *Node.js: Using JavaScript to build high-performance network programs*. IEEE Internet Computing, 14(6), 80–83.