الجمهورية العربية السورية

جامعة حمص

الكلية التطبيقة

قسم تقنيات حاسوب

اسم المشروع

نظام شطرنج تفاعلي على الويب مع دعم تسجيل

المستخدمين والمباريات

Interactive Web-Based Chess System with
User and Match Management

دراسة اعدت لنيل درجة الإجازة في العلوم التطبيقية

باختصاص تقنيات حاسوب

إشراف

الدكتور فدوى صافية

إعداد الطالب

أغيد يحيى

الملخص

تُعَدّ لعبة الشطرنج من أعرق وأشهر الألعاب الذهنية في العالم، حيث تتميز بقدرتها على تنمية التفكير المنطقي، وتعزيز مهارات التخطيط، واتخاذ القرار، وحل المشكلات. ومع الانتشار الواسع للإنترنت والتطور الكبير في تقنيات الويب، انتقلت هذه اللعبة من مجالها التقليدي إلى الفضاء الرقمي، مما أتاح لملايين اللاعبين حول العالم فرصة اللعب والتعلم والتفاعل عبر المنصات الإلكترونية المتنوعة.

وعلى الرغم من توفر العديد من المنصات العالمية للشطرنج، إلا أن المحتوى العربي في هذا المجال ما يزال محدودًا، مما يبرز الحاجة إلى تطوير منصة عربية شاملة تتيح للمستخدمين من مختلف المستويات اللعب الفوري عبر الإنترنت، إدارة بيانات اللاعبين، متابعة الأداء الشخصي، وتعلّم الافتتاحيات الأساسية. كما يهدف المشروع إلى سد هذا النقص من خلال توفير تجربة تعليمية وترفيهية متكاملة تتناسب مع احتياجات المستخدم العربي.

تم تصميم وتطوير المشروع بالاعتماد على تقنيات حديثة وموثوقة مثل Node.js, Express.js, MongoDB, Mongoose, Socket.io, JSON Web Token, وBcrypt، بالإضافة إلى استخدام مكتبة chess.jsلتطبيق قواعد اللعبة برمجيًا. وقد جرى بناء قاعدة بيانات متكاملة تشمل اللاعبين، المسؤولين، الافتتاحيات، وسجلات المباريات، بما يضمن مرونة النظام وقابليته للتوسع مستقبلاً.

ومن خلال هذا المشروع ChessInArabic، أسعى إلى تقديم منصة عملية قابلة للتطوير مستقبلاً، بحيث يمكن إضافة خصائص جديدة مثل دمج محركات تحليل النقلات (Chess Engines) أو تطوير تطبيقات مخصصة للهواتف الذكية، مما يساهم في تعزيز الثقافة الرقمية للشطرنج في العالم العربي.

Abstract

Chess is one of the most ancient and renowned intellectual games in the world. It is distinguished by its ability to enhance logical thinking, strengthen planning skills, and improve decision-making and problem-solving abilities. With the widespread adoption of the Internet and the rapid development of web technologies, chess has moved from its traditional setting into the digital space, allowing millions of players worldwide to play, learn, and interact through various online platforms.

Despite the availability of several global chess platforms, Arabic content in this field remains limited. This highlights the necessity of developing a comprehensive Arabic platform that enables users of different levels to engage in real-time online gameplay, manage player data, track personal performance, and learn essential openings. The project also aims to fill this gap by providing an integrated educational and entertaining experience tailored to the needs of Arabic-speaking users.

The project is designed and implemented using modern and reliable technologies such as Node.js, Express.js, MongoDB, Mongoose, Socket.io, JSON Web Token, and Bcrypt, in addition to employing the chess.js library to implement game rules programmatically. A comprehensive database has been developed to manage players, administrators, openings, and match records, ensuring system flexibility and scalability.

Through this project, the team aims to deliver a practical platform with the potential for future expansion, such as integrating chess engines for move analysis or developing dedicated mobile applications. This contributes to enhancing the digital chess culture within the Arab world.

Table of Contents

Table Of Figures

Table Of Tables

الفصل الأول
لمحة عامة عن المشروع

1.1. المقدمة

في ظل التطور المتسارع للتقنيات الرقمية وتوسع نطاق استخدام الإنترنت، أصبحت المنصات التفاعلية أداة محورية لتوفير تجارب تعليمية وترفيهية جديدة. تُعد لعبة الشطرنج، بتاريخها الغني وعمقها الاستراتيجي، مثالاً بارزاً على الأنشطة التي تستفيد من هذا التطور، حيث أصبحت منصات الشطرنج عبر الإنترنت ساحة عالمية تجمع الملايين من اللاعبين. رغم هذا النمو، لا يزال هناك فراغ ملحوظ في المحتوى والمنصات المتخصصة باللغة العربية، مما يحد من وصول اللاعبين العرب إلى بيئة تفاعلية شاملة تعزز مهاراتهم وتشارك في إثراء مجتمع الشطرنج المحلي.

من هذا المنطلق، يأتي هذا المشروع لتطوير **منصة متكاملة للعب وتعليم الشطرنج عبر الإنترنت** .يهدف المشروع إلى توفير بيئة آمنة، سلسة، وجذابة للاعبين من جميع المستويات، مع التركيز على تلبية احتياجات المجتمع العربي. يرتكز تصميم النظام على منهجية تطوير برمجيات منظمة، بدأت بمرحلة تحليل المتطلبات وتصميم المخططات المعمارية) مثل ERD و (Use Case Diagram لضمان بناء هندسي متين.

تم بناء الواجهة الخلفية للمنصة باستخدام **Node.js** و**Express.js** ، مع الاستفادة من مكتبات متقدمة مثل **Socket.io** للتواصل الفوري و **bcrypt** و**jsonwebtoken** لتوفير أعلى مستويات الأمان. هذا الدمج للتقنيات الحديثة لا يضمن فقط أداءً عاليًا، بل يوضح أيضًا تطبيقًا عمليًا للمفاهيم الهندسية المكتسبة. ستوفر المنصة وظائف رئيسية تشمل اللعب في الوقت الفعلي، ومصادقة آمنة للمستخدمين، وتتبع الأداء الشخصي، مما يجعلها مساهمة قيمة في إثراء المحتوى التقني والثقافي العربي.

1.2. أهمية المشروع

تكمن أهمية هذا المشروع في ثلاثة أبعاد أساسية :**علمي–تقني، اجتماعي–ثقافي، وشخصي**.

1.2.1. الأهمية العلمية والتقنية

تكامل التقنيات الحديثة: يمثل المشروع تطبيقًا عمليًا لمنهجيات تطوير الأنظمة المتكاملة، حيث يدمج بفعالية بين الواجهة الخلفية (backend) القائمة على **Node.js** و **Express.js**، وقاعدة بيانات **MongoDB**، بالإضافة إلى استخدام تقنيات الاتصال الفوري مثل **Socket.io**. هذا التكامل يعكس فهمًا عميقًا لكيفية بناء تطبيقات ويب حديثة وذات أداء عالٍ.

التصميم الهندسي المنهجي: يبرز المشروع التزامًا بالمنهجيات الهندسية المتبعة في تطوير البرمجيات. فبدءًا من مرحلة تحليل المتطلبات ووصولًا إلى تصميم المخططات المعمارية (مثل ERD و Use CaseDiagram)، يظهر المشروع قدرة على التخطيط المسبق وتنفيذ الحلول البرمجية بشكل منظم ومنهجي.

حلول الأمان: يركز المشروع على تطبيق أفضل ممارسات الأمان من خلال استخدام تقنيات التشفير المتقدمة مثل **bcrypt** و**jsonwebtoken**، مما يضمن حماية بيانات المستخدمين ويعكس وعيًا بالجوانب الحاسمة لأمان المعلومات.

1.2.2. الأهمية الاجتماعية والثقافية

سد الفجوة في المحتوى العربي: في ظل التطور العالمي المتسارع للعبة الشطرنج وزيادة أعداد ممارسيها، يساهم المشروع في إثراء المحتوى العربي المتخصص. فهو يوفر منصة متكاملة تُلبي احتياجات اللاعبين العرب، مما يعزز حضورهم في المجتمع العالمي للشطرنج.

تنمية المهارات المعرفية: يُنظر إلى الشطرنج أكاديميًا كأداة فعّالة لتنمية التفكير الاستراتيجي، حل المشكلات، وصناعة القرار. وبالتالي، فإن توفير هذه المنصة يُعد مساهمة مباشرة في تعزيز هذه المهارات لدى المستخدمين العرب بشكل تفاعلي وممتع.

تعزيز التواصل: تتيح المنصة للاعبين فرصة التفاعل والمنافسة، مما يساهم في بناء مجتمع عربي مهتم بالشطرنج، ويشجع على التبادل المعرفي بين مختلف المستويات.

1.2.3. الأهمية الشخصية

تأتي الأهمية الشخصية لهذا المشروع من الاهتمام الفردي باللعبة، حيث شكلت الممارسة المنتظمة للشطرنج دافعًا رئيسيًا لتطوير هذه المنصة. وكما هو موضح في **(الشكل 1.1)**، فقد حققت المركز الثالث بالتساوي في النقاط في بطولة جامعة حمص لعام 2023.

1 الشكل 1.1. مركز الثالث من بطولة جامعة حمص

1.3. أهداف المشروع

يهدف مشروع **ChessInArabic** إلى تطوير منصة إلكترونية حديثة للعب الشطرنج عبر الإنترنت، موجّهة بالدرجة الأولى إلى الجمهور العربي، مع التركيز على الجوانب التقنية والأمنية لضمان تجربة استخدام سلسة وموثوقة. يمكن تلخيص الأهداف الرئيسية للمشروع كما يلي:

1.3.1. الهدف الثقافي والمعرفي:

تعزيز حضور لعبة الشطرنج في المجتمعات العربية من خلال توفير منصة تدريبية وتنافسية تساعد اللاعبين على صقل مهاراتهم وزيادة فرص مشاركتهم في البطولات الدولية.

1.3.2. الهدف التقني:

تصميم نظام لعب تفاعلي يعتمد على بروتوكول WebSocket لضمان استجابة منخفضة وتوفير تجربة لعب آنية وسلسة بين اللاعبين.

1.3.3. الهدف الأمني:

- تطبيق آلية مصادقة آمنة ومستقرة باستخدام JSON Web Token (JWT) لضمان صلاحيات الوصول وحماية الجلسات.
- استخدام خوارزمية bcrypt لتشفير كلمات المرور وتخزينها بأمان في قاعدة البيانات.
- تأمين قواعد بيانات المستخدمين ضد الاختراق أو سوء الاستخدام.

1.3.4. الهدف المستقبلي:

إتاحة المجال للتوسع عبر دمج أنظمة تحليل النقلات (Chess Engines) لتطوير الجانب التعليمي، إضافة إلى إمكانية إنشاء تطبيقات مخصصة للهواتف الذكية تدعم المنصة وتزيد من انتشارها.

1.4. مشكلة المشروع

تواجه منصات الألعاب الإلكترونية عبر الإنترنت مجموعة من التحديات التقنية المرتبطة بتوفير تجربة لعب لحظية وموثوقة للمستخدمين. فمن جهة، يتطلب تصميم أنظمة لعب تفاعلية دعم الاتصال ثنائي الاتجاه (Full-Duplex) لتحقيق تبادل مستمر وسريع للبيانات بين اللاعبين، مع الحفاظ على زمن استجابة منخفض (Low Latency) يضمن سلاسة التفاعل وعدم حدوث تأخير يؤثر سلبًا على مجريات اللعبة.

ومن جهة أخرى، فإن تأمين بيئة الاستخدام يمثل تحديًا لا يقل أهمية، إذ يجب حماية بيانات المستخدمين وخصوصًا كلمات المرور، وضمان إدارة الجلسات والمصادقة بشكل آمن لتجنب الاختراقات أو التلاعب بنتائج المباريات.

إضافة إلى ذلك، ينبغي أن تراعي المنصة عامل سهولة الاستخدام عبر توفير واجهة واضحة وبسيطة تتيح للمستخدمين من مختلف المستويات التفاعل مع النظام دون تعقيد، مع إمكانية التوسع مستقبلًا لدعم وظائف أكثر مثل دمج محركات تحليل النقلات أو إنشاء تطبيقات للهواتف الذكية.

1.5. المستفيدون

يمكن تلخيص المستهدفون كما يلي:

1.5.1. المجتمع التقني والمطورون:

الأفراد والمؤسسات المهتمون بتصميم وبناء أنظمة تواصل بالوقت الفعلي (Real-Time Systems)، والراغبون في استكشاف وتطبيق تقنيات حديثة مثل WebSocket و Full-Duplex Communication لتطوير بيئات تفاعلية لحظية.

1.5.2. مهتمون بالأمن وحماية البيانات:

المستخدمون والمطورون الراغبون في تطبيق أساليب المصادقة وتأمين قواعد البيانات، بما في ذلك استخدام JSON Web Token (JWT) و bcrypt لضمان حماية المعلومات الشخصية وكلمات المرور، وتحقيق بيئة استخدام موثوقة وآمنة.

1.5.3 اللاعبون والمستخدمون الراغبون في التعلم:

الأفراد المهتمون بتنمية مهاراتهم في التفكير المنطقي والتحليل الاستراتيجي، والاستفادة من لعبة الشطرنج كأداة تعليمية وتنافسية لتطوير قدراتهم الذهنية، وتحسين الأداء في بيئة تدريبية وتفاعلية.

بهذا الشكل، يوفر المشروع قيمة مضافة لكل فئة من هذه الفئات من خلال تلبية احتياجاتها التقنية، الأمنية والتعليمية ضمن منصة واحدة متكاملة وسهلة الاستخدام.

الفصل الثاني

الدراسة النظرية

2.1. الأدوات واللغات البرمجية

تعتمد البنية التقنية للمشروع على ثلاث مكونات أساسيين يشكلون الأساس في عملية التطوير:

- عمل الواجهة الأمامية (Frontend)
- عمل الواجهة الخلفية (Backend)
- قاعدة البيانات (DataBase)

بالإضافة إلى ذلك، تم استخدام مجموعة من أدوات التطوير (Development Tools) التي ساعدت في تسهيل عمليات البرمجة، وإدارة المشروع، واختبار النظام بشكل متكامل وفعّال.

2.1.1. الواجهة الأمامية (Frontend)

تمثل الواجهة الأمامية الجانب المرئي من التطبيق، وهو كل ما يتفاعل معه المستخدم مباشرةً. لقد اعتمد المشروع على مكتبات وأطر عمل حديثة للواجهة الأمامية تفاعلية، وممتعة. يضمن هذا الجانب من التطبيق سلاسة التنقل، وسرعة الاستجابة، وتصميمًا جذابًا يسهل على اللاعبين من جميع المستويات التفاعل مع المنصة.

الأدوات والمكتبات المستخدمة في تطوير الواجهة الأمامية **ما يلي:**

2.1.1.1 HTML5:

HTML5 هي النسخة الحديثة من لغة توصيف النص التشعبي (HyperText Markup Language)، وتعتبر المعيار الأساسي لبناء هيكل صفحات الويب. تسمح بتحديد العناصر المختلفة على الصفحة مثل العناوين، الفقرات، الجداول، النماذج، والروابط، كما توفر دعماً للعناصر التفاعلية والوسائط المتعددة [1].

استُخدمت HTML5 لبناء الهيكل العام لمنصة ChessInArabic بطريقة معيارية، مما يسهل دمج المكتبات الأخرى مثل chessboard.js وchess.js لتوفير تجربة مستخدم تفاعلية [1].

2.1.1.2 CSS3:

CSS3 هي النسخة الحديثة من لغة تنسيق صفحات الويب (Cascading Style Sheets)، وتتيح التحكم بمظهر العناصر على الصفحة مثل الألوان، الخطوط، الهوامش، التخطيطات، والحركات. كما تدعم تصميم الواجهات المتجاوبة (Responsive Design) لتتناسب مع مختلف أجهزة العرض [2].

استُخدمت CSS3 لتنسيق واجهات المستخدم لمنصة ChessInArabic، مع تحسين قابلية القراءة والجماليات البصرية، خاصة عند دمج إطار Tailwind CSS لتصميم سريع ومرن [2].

JavaScript (ES6+) 2.1.1.3:

JavaScript هي لغة برمجة عالية المستوى تعمل على المتصفح وعلى الخادم عبر بيئة Node.js، وتُستخدم لإضافة التفاعل والتحكم الديناميكي على صفحات الويب. النسخ الحديثة (ES6+) تقدم ميزات متقدمة مثل الـ Modules، Arrow Functions، Promises، Classes [3] [4].

استُخدمت JavaScript لإضافة المنطق التفاعلي في الواجهة، مثل تحريك القطع على الرقعة، التحقق من الحركات باستخدام chess.js، وربط الواجهة بالخادم عبر Socket.IO للعب اللحظي، إرسال طلبات للمخدم عند القيام بحدث كلضغط على زر [3] [4].

Tailwind 2.1.1.4:

مكتبة Tailwind CSS إطار عمل (Framework) مخصص لتصميم واجهات المستخدم باستخدام أسلوب Utility-First CSS، حيث يوفّر مجموعة كبيرة من الأصناف الجاهزة (Utility Classes) التي يمكن دمجها مباشرة داخل عناصر HTML لبناء تصاميم متناسقة ومرنة [5].

الميزة الأساسية في Tailwind أنها لا تفرض تصميماً محدداً على المطور، بل تمنحه أدوات منخفضة المستوى (Low-level utilities) يمكن من خلالها تخصيص الواجهة بدقة عالية. هذا الأسلوب يساعد على تقليل الاعتماد على ملفات CSS منفصلة وكبيرة، ويجعل عملية التطوير أسرع وأكثر تنظيمًا [6].

من أبرز المميزات:

- إعادة الاستخدام (Reusability): يمكن استخدام نفس الأصناف في أكثر من عنصر دون الحاجة إلى كتابة أكواد جديدة.
- التخصيص (Customization): توفر ملف إعدادات (Configuration File) يتيح للمطور ضبط الألوان، الخطوط، والمسافات بما يتناسب مع هوية المشروع.
- الأداء (Performance): يدعم Tailwind خاصية إزالة الأصناف غير المستخدمة (PurgeCSS) مما يقلل حجم الملفات النهائية ويحسّن سرعة تحميل الصفحات [5].
- التكامل السلس: يمكن دمجه بسهولة مع لغات البرمجة الحديثة وأطر العمل مثل React و Vue و Next.js.

تم بناء Tailwind أساسًا بلغة CSS ولكنه يقدم طبقة تنظيمية أعلى من خلال PostCSS التي تسمح بإنشاء هذه الأدوات الديناميكية [6].

في هذا المشروع، ساعدت مكتبة Tailwind CSS الاصدار 3.7.2 على بناء واجهة تفاعلية حديثة وسهلة الاستخدام لمنصة ChessInArabic، حيث مكّنت من تطوير تصميم متجاوب (Responsive Design) بسرعة، مع توفير تجربة مستخدم سلسة وبصرية جذابة.

2.1.1.5 Chessboard.js:

chessboard.js مكتبة جافاسكربت مفتوحة المصدر برخصة استخدام MIT license مخصّصة لعرض لوحة الشطرنج (Chessboard UI) على صفحات الويب بطريقة تفاعلية وبسيطة. تم تطويرها لتكون مكمّلة لمكتبة chess.js التي تتعامل مع منطق اللعبة وقوانينها.

توفر عرض رسومي تفاعلي، يتيح إنشاء واجهة رسومية لعرض رقعة الشطرنج بشكل مرن وسهل التخصيص، مما يمكّن المستخدم من رؤية القطع وتحريكها عبر السحب والإفلات (Drag & Drop) [7].

التكامل مع مكتبات أخرى، غالبًا ما تُستخدم جنبًا إلى جنب مع مكتبة chess.js، حيث تتكفل chessboard.js بالواجهة الرسومية بينما تتولى chess.js التحقق من صحة النقلات وتطبيق قوانين اللعبة [7].

سهلة الاستخدام تعتمد المكتبة على JavaScript و HTML/CSS بشكل مباشر، ما يجعلها مناسبة للدمج في أي مشروع ويب دون تعقيد إضافي.

تدعم خيارات تخصيص عديدة مثل تحديد وضع العرض (للاعب الأبيض أو الأسود)، إضافة/إزالة أزرار التحكم، أو برمجة ردود أفعال عند تحريك القطع.

مكتبة Cheesboard.js بالأساس على jQuery وتحتاج إلى الاصدار 3.4.1 ومافوق كشرط أساسي للتشغيل [7].

أُستخدمت مكتبة chessboard.js في هذا المشروع لتوفير واجهة رسومية ديناميكية تمثل رقعة الشطرنج بشكل واضح وسهل الاستخدام للمستخدمين. هذه الواجهة تعتبر عنصرًا جوهريًا لتحقيق تجربة لعب واقعية، حيث تمكّن اللاعبين من تحريك القطع مباشرة عبر المتصفح ومشاهدة التغيرات في الزمن الحقيقي، مع الحفاظ على بساطة التصميم وسرعة الأداء.

Jquery 2.1.1.6:

مكتبة jQuery إطارًا مبنيًا على JavaScript يوفّر واجهة برمجية مبسطة للتعامل مع عناصر صفحات الويب. وقد تم اعتمادها كأساس ضروري لعمل مكتبة chessboard.js، حيث تعتمد هذه الأخيرة على وجود jQuery لتنفيذ عمليات التفاعل مع الرقعة، مثل معالجة أحداث السحب والإفلات وتحديث الحركات بشكل ديناميكي. ومن الناحية البنائية، فإن jQuery مكتوبة بلغة JavaScript، وتعمل كطبقة وسيطة بين متصفح المستخدم والوظائف المدمجة في مكتبة chessboard.js، مما يتيح دمج التمثيل البصري للرقعة مع الأحداث التفاعلية بشكل سلس وفعّال [7] [8].

تُعد مكتبة chess.js واحدة من أبرز المكتبات مفتوحة المصدر المُستخدمة في برمجة تطبيقات الشطرنج عبر الويب. تم تطويرها بلغة JavaScript، الأمر الذي يجعلها متوافقة مع معظم بيئات التطوير الحديثة سواء في جانب العميل (Client-Side) أو الخادم (Server-Side) عند دمجها مع Node.js.

Chess.js 2.1.1.7:

الغرض الأساسي من المكتبة هو معالجة منطق لعبة الشطرنج (Chess Logic)، فهي لا تتعامل مع الرسوميات أو عرض الرقعة، بل تركز على الجوانب الحسابية والقانونية للعبة. على سبيل المثال، تتيح التحقق من صحة النقلات، تتبع حالة اللعبة (كش ملك، تعادل، أو استسلام)، وتوليد النقلات القانونية الممكنة في أي وضعية معينة.

ميزاتها:

- التحقق من قواعد الشطرنج القياسية بشكل كامل
- إمكانية توليد النقلات القانونية ومتابعة التسلسل الزمني للعبة.
- قابليتها للتكامل بسهولة مع مكتبات أخرى مثل chessboard.js لعرض الرقعة، مما يتيح الفصل بين منطق اللعبة والواجهة البصرية.
- دعم تمثيل اللعبة باستخدام صيغة FEN (Forsyth–Edwards Notation) و PGN ***هما طريقتا تسجيل النقلات*** [31] [30].

المكتبة مكتوبة بلغة JavaScript بالكامل، وهو ما يجعلها مرنة وسهلة الدمج مع مختلف الأطر (Frameworks) في تطوير تطبيقات الويب [9] .

الانتشار والشهرة:

حظيت المكتبة بانتشار واسع بين المطورين نظرًا لكونها خفيفة الوزن، مفتوحة المصدر، وسهلة الدمج. تُستخدم في مشاريع تعليمية، منصات لعب تفاعلية، وكذلك في بناء أدوات تحليلية تعتمد على محركات الشطرنج (Chess Engines). ويعود جانب من شهرتها أيضًا إلى تكاملها الشائع مع chessboard.js لتوفير بيئة متكاملة تجمع بين المنطق (Logic) والواجهة الرسومية (UI).

الجدول 2.1 يوضح العلاقة بين مكتبة chess.js و Chessboard.js في المشروع

البند	chess.js	chessboard.js
الغرض الأساسي	معالجة منطق اللعبة (Game Logic) والتحقق من قواعد الشطرنج.	عرض واجهة رسومية (Graphical Interface) لرقعة الشطرنج.
المهام الرئيسية	توليد النقلات القانونية، التحقق من صحة النقلات، تتبع حالة اللعبة.	عرض الرقعة والقطع، تحريك القطع بصريًا، دعم التفاعل مع اللاعب.(Drag & Drop)
صيغة البيانات	يدعم FEN و PGN لتمثيل الحالة والتاريخ.	يعتمد على تمثيل الحالة القادمة من مكتبة منطق مثل.chess.js
لغة التطوير	مكتوبة بالكامل بلغة.JavaScript	مكتوبة بلغة JavaScript وتعتمد على مكتبة jQuery.
الوظيفة في المشروع	إدارة المنطق الداخلي للعبة وضمان صحة النقلات.	توفير تجربة مرئية وتفاعلية لواجهة الشطرنج.
علاقة التكامل	تُستخدم عادةً مع chessboard.js لعرض نتائج المنطق بشكل رسومي.	تعتمد غالبًا على chess.js لتلقي المنطق وقواعد اللعبة.

2.1.1. مقارنة بين مكتبة Chess.js وChessBoard.js

:Socket.io-Client 2.1.1.8

socket.io-client المكوّن الأساسي في جهة العميل (Client Side) لمكتبة Socket.IO، حيث تتيح إنشاء اتصال ثنائي الاتجاه (Full Duplex) بين المتصفح والخادم باستخدام بروتوكول WebSocket، مع دعم بروتوكولات بديلة لضمان التوافق [10].

الغرض من استخدامها هو تمكين التطبيقات من تبادل البيانات بالزمن الحقيقي (Real-Time) مثل إرسال واستقبال النقلات في لعبة الشطرنج دون الحاجة إلى إعادة تحميل الصفحة [11].

2.1.2. الواجهة الخلفية (Backend)

يُعد الجانب الخلفي للمشروع بمثابة المحرك الرئيسي للنظام. يهدف هذا الجزء إلى توفير بيئة خادم مستقرة وآمنة لإدارة كافة عمليات التطبيق، بما في ذلك:

- معالجة طلبات المستخدمين.
- إدارة جلسات اللعب في الوقت الفعلي.
- يؤدي دور الوسيط بين الواجهة الأمامية وقاعدة البيانات، لضمان نقل البيانات بشكل منظم وآمن.
- مصادقة عمليات تسجيل الدخول

تم تحقيق هذه الأهداف باستخدام مجموعة من الأدوات والمكتبات المتخصصة، التي تضمن كفاءة الأداء وقابلية التوسع.

:Node.js 2.1.2.1

Node.js هو بيئة تشغيل مفتوحة المصدر تعتمد على محرك V8 JavaScript الخاص بجوجل، وتتيح تنفيذ لغة JavaScript على جهة الخادم (Backend) لتطوير تطبيقات شبكية عالية الأداء وقابلة للتوسع [12]. يتميز Node.js بكونه غير متزامن (Asynchronous) ومعتمد على الأحداث (Event-Driven)، مما يجعله مناسبًا للتطبيقات التي تتطلب معالجة عدد كبير من الاتصالات في الزمن الحقيقي مثل الدردشة والألعاب.

على الرغم من أن Node.js يعمل كـ Single-Threaded Process، إلا أنه يستطيع إدارة العديد من العمليات في الوقت نفسه بفضل حلقة المعالجة (Event Loop). تقوم الحلقة بمراقبة قائمة الأحداث (Event Queue) وتنفيذ المهام غير المتزامنة عند اكتمالها، مما يسمح للنظام بالتعامل مع آلاف الاتصالات دون الحاجة إلى

إنشاء خيوط متعددة لكل عملية. بهذا الشكل، يحافظ Node.js على كفاءة استخدام الموارد ويقلل من الحمل على الخادم مقارنة بالأنظمة التقليدية متعددة الخيوط [13] .

2.1.2.2 Express.js:

Express.js هو إطار عمل مفتوح المصدر يعتمد على بيئة تشغيل Node.js، ويُستخدم لتطوير تطبيقات الويب وواجهات برمجة التطبيقات (APIs) بسرعة وكفاءة، مع توفير بنية منظمة للتعامل مع طلبات واستجابات HTTP [1].

تم اختيار Express.js في مشروع ChessInArabic بسبب بساطته ومرونته في إنشاء تطبيقات الويب المتقدمة، بالإضافة إلى تكامله السلس مع مكتبات Socket.IO وMongoose، ما يتيح إدارة الاتصالات اللحظية وقاعدة البيانات بشكل متكامل وفعّال [14].

أهم ميزاتها بالنسبة لطبيعة المشروع:

- التوجيه (Routing): القدرة على إدارة جميع طلبات HTTP للمباراة وحركات اللاعبين بشكل منظم.
- الوسائط (Middleware): إضافة وظائف مثل التحقق من هوية المستخدم، إدارة الجلسات، ومعالجة البيانات قبل إرسالها للواجهة الأمامية.
- التكامل مع قواعد البيانات: سهولة الربط مع MongoDB عبر Mongoose لتخزين واسترجاع بيانات اللاعبين والمباريات.
- الكفاءة والأداء: دعم تطبيقات الوقت الحقيقي دون الحاجة لبناء بنية معقدة، ما يجعلها مثالية لتطبيق ChessInArabic الذي يعتمد على تحديثات لحظية للحركات وحالة اللاعبين.

2.1.2.3 Scoket.io:

Socket.IO هو مكتبة JavaScript تُستخدم لإنشاء تطبيقات شبكية تفاعلية في الزمن الحقيقي (Real-Time Applications)، مثل الدردشة، الألعاب متعددة اللاعبين، والتحديثات اللحظية في الواجهات [11].

تعتمد Socket.IO على بروتوكول WebSocket ، ولكنها توفر طبقة إضافية من المرونة لدعم المتصفحات القديمة، حيث يمكن أن تتحول تلقائيًا إلى تقنيات بديلة) مثل (Long Polling إذا لم يكن WebSocket مدعومًا [10] [11].

- تعمل مكتبة Socket.IO على إطلاق واستقبال الأحداث بين العميل والخادم بشكل فوري.

- كل حدث يمكن أن ينقل بيانات أو تعليمات، وتستقبل الجهة الأخرى هذه الأحداث مباشرة [11].

يقارن الجدول 2.2. بين مكوّنات Socket.IO من جهة الخادم والعميل حسب جدول

البند	Socket.IO الخادم	Socket.IO-Client العميل
مكان التنفيذ	يعمل على جهة الخادم (Server) ضمن بيئةNode.js	يعمل على جهة العميل (Client) في المتصفح أو تطبيقات الموبايل
الغرض الأساسي	إدارة الاتصالات، بث الرسائل، معالجة الأحداث	إنشاء الاتصال بالخادم، إرسال واستقبال البيانات اللحظية
الدور	منظم الاتصال والتحكم بالجلسات	منفذ للاتصال وتفاعل مباشر مع المستخدم
البنية التقنية	يعتمد على بروتوكول WebSocket مع دعم بدائل عند الحاجة	يعتمد على بروتوكول WebSocket للاتصال بخادمSocket.IO
العلاقة	لا يعمل بدون وجود عملاء متصلين	لا يمكنه العمل دون خادم يستجيب

2 2.2. مقارنة بين مكوّنات Socket.IO من جهة الخادم والعميل

2.1.2.3.2 بروتوكول WebSocket:

هو بروتوكول شبكي يُتيح اتصالًا ثنائي الاتجاه (Full-Duplex) بين العميل والخادم عبر منفذ TCP واحد، ويتميز بما يلي [10]:

اتصال دائم: بعد إنشاء الاتصال، يظل مفتوحًا لتبادل البيانات دون الحاجة لإعادة إنشاء الطلبات.

تقليل الحمل على الشبكة: البيانات تُرسل بشكل مضغوط بدون رؤوس HTTP كبيرة.

التفاعل اللحظي: يتيح إرسال البيانات من العميل إلى الخادم أو العكس مباشرة فور حدوث الحدث.

جدول 2.3. يوضح الفرق بين WebSocket وHttp

المعيار	HTTP	WebSocket
نوع الاتصال	طلب/استجابة(Request/Response)	ثنائي الاتجاه دائم(Full-Duplex)
الحاجة لإعادة الاتصال	نعم، لكل طلب	لا، الاتصال مستمر
زمن الاستجابة	أطول بسبب إنشاء كل طلب	أسرع، لحظي تقريبًا
الاستخدام الأمثل	تصفح صفحات الويب	تطبيقات الزمن الحقيقي

3 2.3. الفرق بين HTTP وWebSocket

تم اختيار مكتبة Socket.io في هذا المشروع لعدة اعتبارات تقنية، من أبرزها:

- إتاحة اللعب في الزمن الحقيقي، حيث يوفّر البروتوكول آلية لنقل الحركات بين اللاعبين بشكل مباشر دون تأخير ملحوظ.
- إدارة فعّالة للجلسات، وذلك من خلال تخصيص قناة اتصال مستقلة لكل مباراة، بما يمنع حدوث أي تداخل في البيانات بين اللاعبين المختلفين.
- تعزيز التفاعل اللحظي، إذ يسمح البروتوكول بعرض الحركات على لوحة اللعب فور تنفيذها، مع ضمان تحديث حالة اللاعبين النشطين وإدارة التوقيت بشكل متزامن.

:Json Web Token 2.1.2.4

يُعتبر JSON Web Token (JWT) معيارًا مفتوحًا (RFC 7519) يُستخدم لتبادل المعلومات بين طرفين بشكل آمن على هيئة كائنات JSON. يتميز JWT بكونه مُدمجًا ومحمولًا (Compact & Self-contained)، مما يجعله مناسبًا لتطبيقات الويب الحديثة التي تحتاج إلى إدارة جلسات المستخدم والمصادقة [1].

يتكون الرمز من ثلاثة أجزاء رئيسية مفصولة بنقاط (.):

1. **Header (الرأس):** يحتوي على نوع الرمز (JWT) وخوارزمية التشفير المستخدمة (مثل HS256 أو RS256).

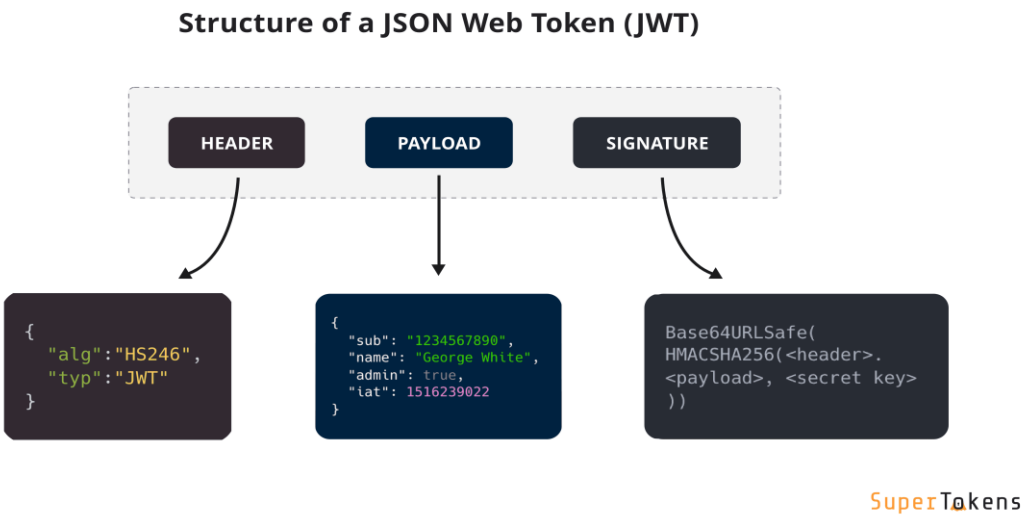
2. **Payload (الحمولة)**: يتضمن البيانات المراد نقلها مثل معرف المستخدم أو الصلاحيات (Claims).
3. **Signature (التوقيع)**: يُستخدم للتحقق من صحة الرمز ومنع التلاعب به، حيث يُنشأ عبر خوارزمية التشفير باستخدام المفتاح السري.

عادةً ما يُستخدم خوارزميات HMAC مع SHA-256 (مثل HS256) أو خوارزميات RSA (مثل RS256) لتوقيع الرموز. هذا التوقيع يضمن أن أي تعديل غير مصرح به في البيانات سيؤدي إلى بطلان الرمز.

تم اختيار JWT في المشروع لعدة أسباب، أهمها:

- إدارة جلسات المستخدم (User Sessions): بما أن اللعبة متعددة اللاعبين، فإن JWT يوفر آلية فعّالة لتوثيق كل لاعب والتحقق من هويته.
- الأمان: يمنع الوصول غير المصرح به من خلال توقيع مشفر يصعب التلاعب به.
- التكامل مع REST APIs: كونه معيارًا شائعًا، يسهل دمجه مع Express.js وSocket.IO لإدارة المصادقة في الوقت الحقيقي.
- خفيف وسهل النقل: يمكن تضمينه في ترويسات HTTP (HTTP Headers) أو في الكوكيز، مما يجعله مثاليًا لتطبيقات الويب التفاعلية مثل ChessInArabic.

الشكل 2.1. يوضح بنية JWT



الشكل 2.1. بنية JWT 2

Mongoose 2.1.2.5:

Mongoose مكتبة (Object Data Modeling – ODM) مبنية على Node.js تُستخدم للتعامل مع قاعدة البيانات MongoDB. تعمل كطبقة وسيطة (Abstraction Layer) بين التطبيق وقاعدة البيانات، حيث توفر واجهة برمجية منظمة لتعريف البيانات عبر Schemas والتعامل معها ككائنات (Objects) في JavaScript بدلاً من التعامل المباشر مع أوامر MongoDB [15].

آلية العمل:

- تسمح Mongoose بإنشاء مخططات (Schemas) تحدد بنية البيانات (الحقول، الأنواع، القواعد).
- تدعم التحقق من صحة البيانات (Validation) قبل تخزينها في قاعدة البيانات.
- توفر وظائف متقدمة مثل العلاقات بين الجداول (Population) والوسطاء (Middleware) لمعالجة البيانات قبل أو بعد عمليات الحفظ أو الاسترجاع.
- تُمكّن من كتابة استعلامات بطريقة كائنية (Object-Oriented) مما يُبسّط تطوير التطبيقات.

تم اختيار Mongoose لعدة مبررات:

- إدارة منظمة للبيانات: حيث يسمح بإنشاء مخططات محددة لبنية بيانات اللاعبين، المباريات، وحركات الشطرنج، مما يقلل من الأخطاء.
- التكامل مع Node.js وExpress: كونها مكتبة مبنية خصيصًا للعمل مع Node.js جعلها الخيار الأمثل لطبقة البيانات في المشروع.
- التحقق من صحة البيانات: يضمن إدخال بيانات صحيحة مثل صيغة البريد الإلكتروني أو التحقق من الحقول الإلزامية قبل تخزينها.
- المرونة في التعامل مع MongoDB: تُبسط كتابة الاستعلامات المعقدة وتزيد من سرعة التطوير مقارنة بالاعتماد على استعلامات MongoDB الأصلية.

Bcrypt 2.1.2.6:

تُعتبر bcrypt خوارزمية لتجزئة (Hashing) كلمات المرور طُوّرت عام 1999 بواسطة Niels Provos وDavid Mazières، وتعتمد على خوارزمية Blowfish كخوارزمية أساسية للتشفير. تُستخدم على نطاق واسع لتأمين كلمات المرور وتخزينها في قواعد البيانات بشكل مشفّر بدلًا من تخزينها كنصوص صريحة، مما يزيد من مستوى الأمان [16].

تقوم bcrypt بتحويل كلمة المرور إلى تجزئة مشفّرة عبر عدة جولات (Rounds) من الحسابات الرياضية، تستخدم ما يُعرف بـ Salt، وهو قيمة عشوائية تُضاف إلى كلمة المرور قبل تشفيرها، وذلك لمنع هجمات مثل Rainbow Table Attack، يمكن تعديل عدد الجولات (Cost Factor) لزيادة صعوبة عملية فك التجزئة مع تطور قدرات الحوسبة.

تم اعتماد مكتبة bcrypt في المشروع لعدة أسباب رئيسية:

حماية كلمات المرور: بحيث لا تُخزن بيانات الاعتماد بشكل مباشر، مما يمنع الوصول إليها حتى في حال اختراق قاعدة البيانات.

مقاومة الهجمات الشائعة: مثل Brute Force وRainbow Tables بفضل استخدام Salt وجولات متعددة.

التكامل مع Node.js: توفر مكتبة bcrypt.js واجهة برمجية سهلة الاستخدام لتشفير كلمات المرور والتحقق منها عند تسجيل الدخول.

الأمان المتكيف (Adaptive Security): إمكانية زيادة عدد الجولات مع مرور الوقت لزيادة صعوبة الاختراق مع تطور العتاد الحاسوبي.

:Dotenv 2.1.2.7

dotenv مكتبة مفتوحة المصدر في بيئة Node.js تُستخدم لإدارة المتغيرات البيئية (Environment Variables) من خلال ملف مخصص عادة باسم .env. تسمح هذه المكتبة بتحميل القيم الحساسة (مثل كلمات المرور، مفاتيح التشفير، عناوين قواعد البيانات) إلى التطبيق بشكل آمن ومنفصل عن الشيفرة المصدرية [17].

تقوم dotenv بقراءة ملف .env وتخزين القيم الموجودة فيه ضمن كائن process.env في Node.js.

يمكن استدعاء هذه القيم داخل التطبيق مثلًا: process.env.DB_URL.

هذا الفصل بين الإعدادات والشيفرة المصدرية يسهل نقل التطبيق بين بيئات التطوير والإنتاج دون تعديل الكود.

:Cookie-Parser 2.1.2.8

تُعتبر cookie-parser مكتبة (Middleware) في Node.js وExpress.js تُستخدم لتحليل (Parse) ملفات تعريف الارتباط (Cookies) المرسلة من العميل إلى الخادم. حيث تقوم بقراءة الكوكيز من ترويسات الطلبات (HTTP Headers) وتحويلها إلى كائن JavaScript يسهل التعامل معه داخل التطبيق[18] .

عند إرسال المستخدم طلب HTTP يحتوي على كوكيز، تقوم المكتبة بترجمتها إلى كائن يُضاف إلى الخاصية req.cookies، يمكنها أيضًا تحليل الكوكيز الموقعة (Signed Cookies) عند استخدام مفتاح سري، مما يوفر مستوى إضافيًا من الأمان، تُستخدم عادةً جنبًا إلى جنب مع مكتبات أخرى مثل express-session أو JWT لإدارة الجلسات والمصادقة.

سبب استخدامها في المشروع:

- إدارة جلسات اللاعبين: تساعد في تخزين معرفات الجلسات (Session IDs) بشكل آمن للتفريق بين اللاعبين.
- التكامل مع المصادقة: عند دمجها مع JWT، يمكن استخدامها لتخزين رموز المصادقة داخل الكوكيز مما يسهل عملية التحقق من هوية المستخدم.
- سهولة التعامل: توفر وسيلة مباشرة للوصول إلى الكوكيز دون الحاجة إلى تحليلها يدويًا.

تعزيز الأمان: من خلال دعم الكوكيز الموقعة (Signed Cookies) التي تمنع التلاعب من جهة العميل.

:CORS 2.1.2.9

يشير CORS إلى آلية أمان على مستوى المتصفح تُستخدم للتحكم في كيفية مشاركة الموارد بين أصول (Origins) مختلفة. يُقصد بالأصل (Origin) مزيج البروتوكول + النطاق (Domain) + المنفذ (Port). بشكل افتراضي، يقوم المتصفح بمنع الطلبات القادمة من أصول مختلفة عن الخادم المستضيف (Same-Origin Policy)، وهنا يأتي دور CORS للسماح أو منع هذه الطلبات عبر ترويسات HTTP مخصّصة .[19]

:Validator 2.1.2.10

تُعد مكتبة Validator مكتبة JavaScript شائعة تُستخدم للتحقق من صحة البيانات (Data Validation) في تطبيقات الويب. توفر المكتبة مجموعة كبيرة من الدوال الجاهزة للتحقق من أنواع البيانات المختلفة مثل البريد الإلكتروني، أرقام الهواتف، العناوين، النصوص، وحتى عناوين URL، مما يساهم في منع إدخال بيانات خاطئة إلى قاعدة البيانات [20].

:Morgan 2.1.2.11

تُعد مكتبة Morgan مكتبة Middleware شائعة الاستخدام مع Express.js في بيئة Node.js، وتُستخدم لتسجيل الطلبات HTTP الصادرة من العملاء إلى الخادم. توفر المكتبة آلية سهلة لتتبع النشاطات في التطبيق وتحليل الأداء ومراقبة الأخطاء، مما يُساهم في صيانة التطبيقات وتحسينها [1].

تعمل Morgan كوسيط (Middleware) بين الطلب والاستجابة في تطبيق Express، حيث تقوم بتسجيل معلومات الطلب مثل طريقة الطلب (GET, POST)، عنوان الـ URL، حالة الاستجابة (Status Code)، ومدة المعالجة، يمكن تخصيص صيغة السجلات (Logging Format) وفقًا لحاجة المطور، مثل صيغة 'combined' أو 'dev'، تساعد على تتبع الأخطاء والمشاكل في التطبيق أثناء التطوير أو في بيئة الإنتاج، وتدعم التصدير إلى ملفات للسجلات أو تكاملها مع أنظمة المراقبة.

سبب استخدامها في المشروع:

تسهيل اكتشاف الأخطاء: تساعد على معرفة مكان حدوث الأخطاء أو التأخير في الطلبات.

تحليل الأداء: يمكن استخدام السجلات لمعرفة عدد الطلبات، مدة المعالجة، والاستجابة لـ API.

التكامل السهل مع Express.js: حيث يمكن إدراج Morgan كـ middleware بسهولة دون تعديل هيكل التطبيق.

:NPM 2.1.2.12

يُعد npm اختصارًا لـ Node Package Manager، وهو مدير الحزم الرسمي لبيئة Node.js. يُستخدم لإدارة وتثبيت المكتبات (Packages) والأدوات (Modules) مفتوحة المصدر، مما يسهّل عملية بناء التطبيقات عبر إعادة استخدام الأكواد البرمجية بدلاً من كتابتها من الصفر[21] .

يحتوي على مستودع npm registry الذي يُعد من أكبر المستودعات البرمجية مفتوحة المصدر، يدعم تثبيت الحزم بشكل محلي (Local) داخل المشروع أو بشكل عام (Global) على الجهاز، يوفر ملف package.json لإدارة التبعيات (Dependencies) وتحديد إصدارات المكتبات المطلوبة، يتيح تنفيذ Scripts مخصصة لأتمتة عمليات مثل الاختبار (Testing) أو التشغيل (Start) أو البناء (Build).

أهميته في المشروع:

- تم الاعتماد على npm لتثبيت المكتبات الأساسية مثل: Express.js، Socket.IO، Mongoose، bcrypt وغيرها.
- يساعد في إدارة إصدارات المكتبات لتجنب مشاكل التوافق.
- يوفر مرونة في تشغيل أوامر التطوير (مثل استخدام nodemon عبر سكربت npm run dev).
- يختصر وقت وجهد التطوير عبر إتاحة آلاف المكتبات مفتوحة المصدر الجاهزة للاستخدام.

2.1.3 قاعدة البيانات DataBase:

تُعرّف قاعدة البيانات بأنها نظام منظّم لتخزين وإدارة البيانات بطريقة تسمح بالوصول إليها واسترجاعها ومعالجتها بكفاءة وأمان. تُستخدم قواعد البيانات في التطبيقات الحديثة لتخزين المعلومات الحيوية مثل بيانات المستخدمين، سجلات النشاط، وإعدادات التطبيق، مع الحفاظ على سلامة البيانات وتسهيل إدارتها.

أهمية قاعدة البيانات في المشروع:

- تخزين بيانات اللاعبين: مثل أسماء الحسابات، كلمات المرور المشفّرة، ومستويات اللعب.
- تخزين المباريات والنتائج: لضمان استرجاع حركات اللعب وعرض تاريخ المباريات.
- إدارة الحالة اللحظية للعبة: مثل اللاعبين النشطين والمباراة الجارية، مما يدعم التفاعل في الوقت الحقيقي.
- أمان البيانات: منع فقدان البيانات أو تعديلها بشكل غير مصرح به.

في هذا المشروع تم الاعتماد على MongoDB كقاعدة بيانات أساسية.

2.1.3.1 MongoDB:

MongoDB قاعدة بيانات NoSQL مفتوحة المصدر، تعتمد على تخزين البيانات في شكل مستندات (Documents) بصيغة BSON (Binary JSON)، بدلًا من الجداول والعلاقات التقليدية في قواعد البيانات

العلائقية (Relational Databases). تتيح هذه البنية مرونة عالية في تمثيل البيانات، حيث يمكن أن تحتوي المستندات على حقول متغيرة البنية، مما يجعلها مناسبة للتطبيقات الحديثة سريعة التطور[22] .

تُخزّن البيانات داخل مجموعات (Collections)، وكل مجموعة تتكون من مستندات Documents، تسمح MongoDB بتنفيذ العمليات الأساسية على البيانات (CRUD):

- Create (إضافة بيانات جديدة).
- Read (استرجاع البيانات).
- Update (تعديل البيانات).
- Delete (حذف البيانات).

تدعم الاستعلام باستخدام لغة خاصة بها تُشبه JSON، مما يُسهّل التعامل معها من قبل مطوري تطبيقات الويب.

أسباب اختيارها للمشروع:

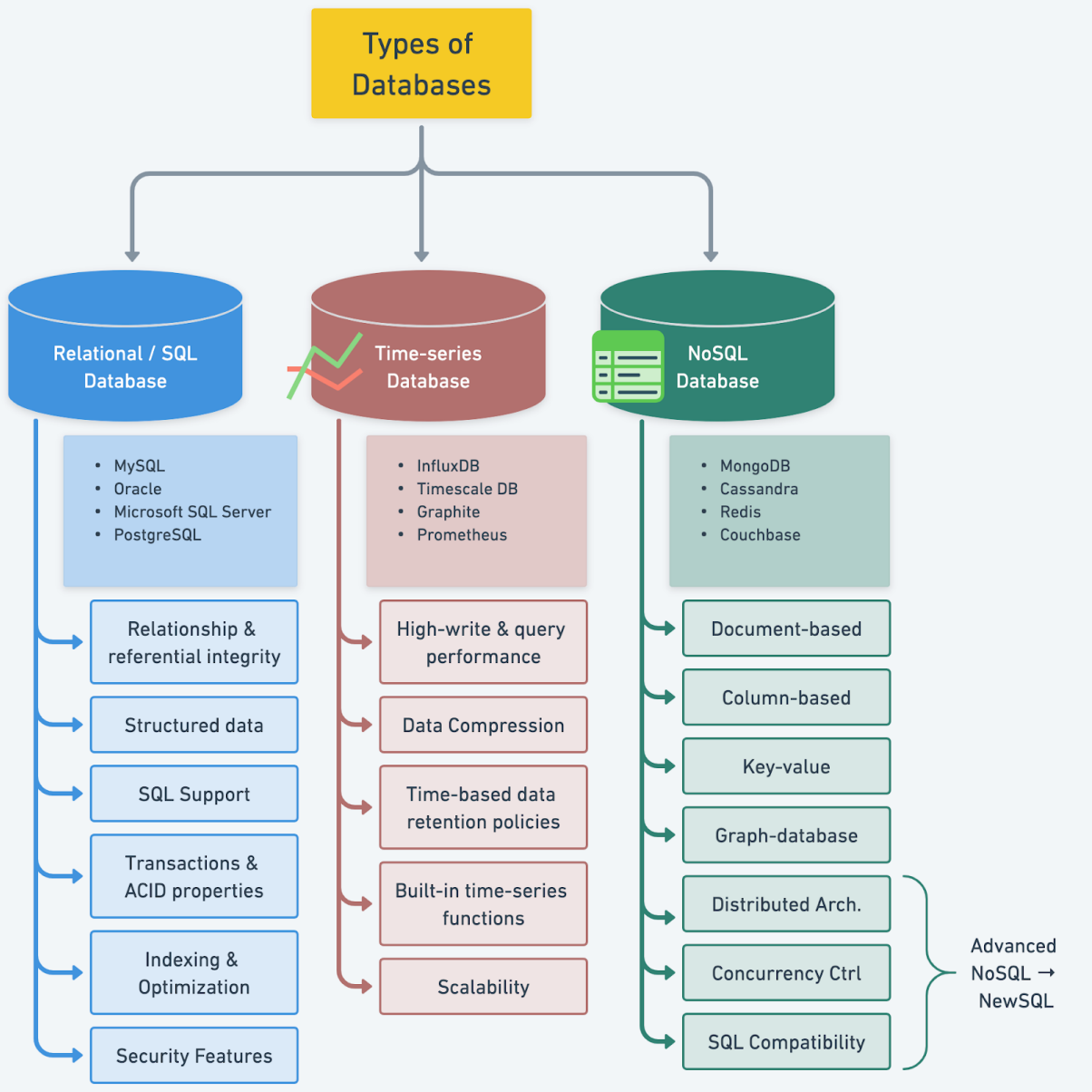
مرونة تخزين البيانات: سهولة تمثيل بيانات اللاعبين والمباريات وحركات الشطرنج بصيغة JSON.

الاستجابة للتغييرات: لا تحتاج إلى تعديل بنية قاعدة البيانات عند إضافة خصائص جديدة.

الدعم للتطبيقات اللحظية (Real-Time): مما يجعلها ملائمة مع WebSocket و Socket.IO لإدارة المباريات المباشرة.

القابلية للتوسع: ما يسمح للنظام باستيعاب عدد كبير من اللاعبين مع نمو قاعدة المستخدمين.

الشكل 2.2 بوضح الفرق بين قواعد البيانات



الشكل 2.2 الفرق بين قواعد البيانات 3

2.1.4 أدوات التطوير (Development Tools):

تشير أدوات التطوير البرمجية (Development Tools) إلى مجموعة البرمجيات والتطبيقات التي تُستخدم من قبل المطورين خلال دورة حياة تطوير البرمجيات (Software Development Life Cycle – SDLC). تساعد هذه الأدوات على تصميم، كتابة، اختبار، تتبع، وتصحيح (Debugging) الشيفرة البرمجية، إضافةً إلى تحسين جودة المنتج وتسريع عملية التطوير .

الأدوات المستخدمة بالمشروع كما يلي:

Vs code 2.1.4.1:

يُعد (Visual Studio Code (VS Code بيئة تطوير متكاملة خفيفة الوزن (Lightweight Integrated Development Environment – IDE) طورتها شركة Microsoft. يتميز بكونه محرر شيفرة (Source Code Editor) مفتوح المصدر ومجاني، مع دعم واسع لمختلف لغات البرمجة عبر إضافات (Extensions) يمكن تثبيتها حسب حاجة المطور، يحتوي على ميزات متقدمة مثل التكملة التلقائية (IntelliSense)، إدارة الشيفرة البرمجية (Code Navigation)، وإبراز الأخطاء البرمجية (Linting & Debugging)، يتكامل مع Git لتتبع الإصدارات والتحكم بالتعديلات بشكل مباشر من داخل بيئة التطوير،يتيح للمطورين تخصيص بيئتهم عبر الثيمات والإضافات، مما يجعله مرنًا وسهل التكيف مع متطلبات المشاريع [23].

Nodemon 2.1.4.2:

يُعد Nodemon أداة مفتوحة المصدر تُستخدم في بيئة Node.js لتسهيل عملية تطوير التطبيقات. تعمل الأداة كوسيط لتشغيل الخادم (Server) مع ميزة إعادة التشغيل التلقائي (Auto-Restart) عند اكتشاف أي تعديل في ملفات المشروع. وبذلك فهي تُسرّع من دورة التطوير (Development Cycle) عبر تقليل الحاجة إلى إعادة تشغيل الخادم يدويًا [24].

MongodbCompass 2.1.4.3:

MongoDB Compass الأداة الرسومية الرسمية (GUI – Graphical User Interface) التي طورتها شركة MongoDB Inc. لإدارة قواعد بيانات MongoDB. يتيح للمطورين والمستخدمين استكشاف البيانات وإدارتها بطريقة مرئية دون الحاجة إلى الاعتماد فقط على سطر الأوامر (CLI) [25].

أهم ميزاته:

- استعراض قواعد البيانات والمجموعات (Collections): تمكين المستخدم من تصفح البُنى المختلفة للبيانات بسهولة.
- إجراء الاستعلامات (Queries): كتابة وتنفيذ أوامر CRUD (إضافة، قراءة، تحديث، حذف) بشكل مرئي وسريع.
- تحليل البيانات (Data Analysis): عرض إحصائيات ورسوم بيانية حول البيانات المخزنة.
- إدارة الفهارس (Indexes): إنشاء وإدارة الفهارس لتحسين أداء الاستعلامات.
- التكامل مع أدوات المطور: دعم الاتصال مع قواعد البيانات المحلية أو البعيدة.

Postman 2.1.4.4:

Postman أداة تطوير واختبار لواجهات برمجة التطبيقات (APIs) تُستخدم على نطاق واسع من قبل المبرمجين والمختبرين. تُمكّن هذه الأداة المستخدمين من إرسال الطلبات (Requests) إلى الخادم واستقبال الاستجابات (Responses) بطريقة مرئية وسهلة، مما يساهم في تسهيل تطوير واختبار التكامل بين الأنظمة المختلفة [26].

الخصائص التقنية:

- إرسال الطلبات: دعم بروتوكولات متعددة مثل HTTP, HTTPS, GraphQL, WebSocket.
- إدارة مجموعات الاختبار (Collections): تنظيم الطلبات في مجموعات لتسهيل إعادة الاستخدام.
- التوثيق (Documentation): توليد وثائق تفاعلية لواجهات الـ API.

الأداة / المكتبة	الإصدار	الغرض من الاستخدام
Chess.js	0.13.4	معالجة منطق لعبة الشطرنج (القوانين، الحركات، الكش مات).
Chessboard.js	3.7.1	عرض رقعة الشطرنج بشكل رسومي وتفاعلي.
Tailwind CSS	3.7.2	تصميم واجهات المستخدم بأسلوب utility-first مع دعم التصميم المتجاوب.
Socket.IO	4.8.1	يستخدم على جهة الخادم لإنشاء وإدارة قنوات اتصال لحظية ثنائية الاتجاه بين المستخدمين والتطبيق، عبر WebSocket.
Socket.IO-Client	4.7.2	يستخدم على جهة العميل لفتح الاتصال مع الخادم والتفاعل مع الرسائل والأحداث بشكل فوري.
Express.js	4.21.2	إطار عمل لتطوير الخادم ومعالجة الطلبات والاستجابات.
Mongoose	5.13.23	إدارة قواعد بيانات MongoDB باستخدام نماذج (Schemas).
MongoDB	7.0	قاعدة بيانات NoSQL لتخزين بيانات اللاعبين والمباريات.

الأداة / المكتبة	الإصدار	الغرض من الاستخدام
bcrypt	6.0.0	تشفير كلمات المرور وتخزينها بشكل آمن.
JSON Web Token (JWT)	9.0.2	إدارة المصادقة وصلاحيات الوصول للمستخدمين.
dotenv	17.2.1	إدارة المتغيرات البيئية وحماية البيانات الحساسة.
cookie-parser	1.4.7	التعامل مع ملفات تعريف الارتباط.(Cookies)
CORS	2.8.5	السماح بالوصول من مصادر مختلفة (Cross-Origin Resource Sharing).
Morgan	1.10.1	تسجيل الطلبات الواردة للخادم بغرض التحليل والمراقبة.
Validator	13.15.15	التحقق من صحة مدخلات المستخدم (البريد الإلكتروني، كلمات المرور).
Node.js	22.11.0	بيئة تشغيل الخادم، تدعم معالجة الطلبات المتزامنة، بناء واجهات برمجة التطبيقات(APIs) ، وإدارة قواعد البيانات والجلسات.
HTML5	-	بناء الهيكل الأساسي لصفحات الويب.
CSS3	-	تنسيق وتصميم واجهات المستخدم، ودعم تصميم متجاوب.
JavaScript	ES6+ وما بعده	بيئة تشغيل الخادم، تدعم معالجة الطلبات المتزامنة، بناء واجهات برمجة التطبيقات(APIs) ، وإدارة قواعد البيانات والجلسات.
npm	11.5.2	دارة الحزم والتبعيات، وتسهيل تثبيت المكتبات اللازمة لتشغيل المشروع.
MongoDB Compass	1.46.8	أداة رسومية لإدارة واستعراض قاعدة البيانات MongoDB بسهولة وفعالية.

الأداة / المكتبة	الإصدار	الغرض من الاستخدام
(VS Code)		بيئة تطوير مُدمجة وخفيفة الوزن تدعم البرمجة، التصحيح، والتحكم بالإصدارات.
Postman	11.58.0	اختبار APIs ، تنظيم الطلبات، وإنشاء اختبارات تلقائية واستكشاف الأخطاء.
nodemon	3.1.10	إعادة تشغيل الخادم تلقائيًا عند تعديل الملفات، مما يسرّع عملية التطوير.

4 2.4. ملخص لغات، ومكتبات المستخدمة

ساهم التكامل بين الأدوات واللغات المستخدمة في المشروع في توفير بيئة تطوير متكاملة تجمع بين الكفاءة والمرونة. فقد مكّنت تقنيات الخادم وقاعدة البيانات من إدارة البيانات بكفاءة، في حين ساعدت أدوات الواجهة الأمامية على بناء تجربة مستخدم تفاعلية وسلسة. وبذلك، شكّل هذا المزيج التقني الأساس الذي اعتمد عليه مشروع (ChessInArabic) لتحقيق أهدافه.

2.2. الدراسات السابقة

2.2.1 تمهيد:

تُعد مراجعة الدراسات السابقة ركيزة أساسية في أي بحث علمي، حيث توفر فهمًا عميقًا للسياق الأكاديمي والتقني للمشروع. في مجال تطوير ألعاب الشطرنج الإلكترونية والأنظمة التفاعلية في الوقت الفعلي، تكتسب هذه المراجعة أهمية خاصة نظرًا للتطور المتسارع في تقنيات الذكاء الاصطناعي، والشبكات، وتجارب المستخدم. من خلال الاستعراض النقدي للأعمال المنجزة، يمكن تحديد الفجوات البحثية، وتحليل التحديات التقنية، واستكشاف الفرص المتاحة لابتكار حلول جديدة.

أهمية مراجعة الدراسات السابقة

2.2.1.1. تحديد الفجوة البحثية:

تُمكن مراجعة الدراسات السابقة الباحث من تحديد المجالات التي لم يتم تناولها بشكل كافٍ أو التي تحتاج إلى تطوير. على سبيل المثال، على الرغم من أن دراسات مثل "Digitization of Chess Game: A Survey on different techniques used for detection and analysis of Chessgame" قد أظهرت أهمية تقنيات الذكاء الاصطناعي في تحليل اللعب، فإنها لم تتطرق بشكل كافٍ إلى التفاعل اللحظي بين اللاعبين عبر الإنترنت، مما يحدد فجوة واضحة يمكن لهذا المشروع سدها.

2.2.1.2. تقييم التقنيات والمنهجيات

تتيح مراجعة الأعمال السابقة التعرف على الأدوات والتقنيات التي أثبتت فعاليتها في هذا المجال. من خلال تحليل الخيارات التقنية التي اتخذها باحثون آخرون، يمكن للمشروع تبني أفضل الممارسات وتجنب الأخطاء الشائعة. كما يمكنها أن توضح أهمية واجهات المستخدم التفاعلية في جذب اللاعبين وتحسين تجربتهم الرقمية.

2.2.1.3. التوجيه نحو الابتكار

تساعد دراسة الأبحاث السابقة في فهم الاتجاهات الحالية والمستقبلية في مجال تطوير ألعاب الشطرنج الإلكترونية. هذا الفهم يوجه المشروع نحو تقديم حلول مبتكرة لا تقتصر على تلبية المتطلبات الوظيفية الأساسية فحسب، بل تضيف قيمة حقيقية من خلال اعتماد منهجيات جديدة أو دمج تقنيات حديثة.

2.2.2 عرض الدراسات:

يهدف هذا القسم إلى تقديم عرض منظم ومفصل للدراسات والأبحاث السابقة المتعلقة بتطوير ألعاب الشطرنج الإلكترونية والألعاب اللحظية. يُعد عرض الدراسات خطوة محورية لفهم الإطار النظري والتقني الذي يعتمد عليه المشروع، حيث يتيح للباحث تحليل الأساليب، الأدوات، والتقنيات المستخدمة في مشاريع مماثلة، وتحديد نقاط القوة والضعف فيها.

من خلال هذا العرض، يمكن تحديد الفجوات البحثية والتقنية التي لم يتم تناولها بشكل كافٍ في المشاريع السابقة، كما يوفر هذا القسم أساسًا لمقارنة الخيارات التقنية، وتوضيح كيفية اختيار الأدوات والطرق الأنسب لتحقيق أهداف المشروع، سواء من حيث التفاعل اللحظي بين اللاعبين، إدارة البيانات، أو تحسين تجربة المستخدم.

2.2.2.1. دراسة: ChessMate – Real-Time Chess Application Using Node.js [27]

الهدف: تطوير لعبة شطرنج لحظية متعددة اللاعبين عبر الويب باستخدام Node.js وSocket.IO.

المنهجية: إنشاء خادم Node.js لإدارة الجلسات والاتصال اللحظي، مع واجهة مستخدم تفاعلية للعب ومراقبة المباريات.

النتائج: أثبت المشروع فعالية Socket.IO في إرسال واستقبال النقلات بين اللاعبين بشكل لحظي، لكنه لم يقدم قاعدة بيانات متقدمة لكل مباراة.

الجدول 2.5. مقارنة بين المشروعين

البند / الخاصية	ChessMate – Real-Time Chess Application	ChessInArabic
الهدف الرئيسي	تطوير لعبة شطرنج لحظية متعددة اللاعبين عبر الإنترنت	تطوير لعبة شطرنج لحظية متعددة اللاعبين، إدارة متقدمة للمستخدمين والمباريات، وتحسين تجربة المستخدم

البند / الخاصية	ChessMate – Real-Time Chess Application	ChessInArabic
التقنيات المستخدمة	Node.js, Socket.IO, HTML/CSS/JS	Node.js, Express.js, Socket.IO, MongoDB, Mongoose, Chessboard.js, Chess.js, Tailwind CSS
التفاعل اللحظي بين اللاعبين	نعم، باستخدامSocket.IO	نعم، باستخدامSocket.IO
واجهة المستخدم	واجهة تفاعلية أساسية	واجهة مستخدم مرنة مع تصميم حديث وسهل الاستخدام
إدارة البيانات	محدودة، لم يتم دمج قاعدة بيانات متقدمة	متكاملة، باستخدام MongoDB وMongoose مع النماذج التالية: • Users: معلومات اللاعبين • SuperAdmin: إدارة الحسابات والإشراف • Games: حفظ بيانات المباريات • Openings: حفظ نقلات البداية وفتوحات الشطرنج
الأمان والمصادقة	لم يتم دمج أمان متقدم	يدعم الأمان والمصادقة باستخدام JWT وbcrypt
التحليل الفني	يوفر لعبة لحظية بسيطة	الأمان، إدارة البيانات المتقدمة، دعمSuperAdmin ، وتحسين تجربة المستخدم بشكل شامل
القيمة المضافة	تجربة لعب لحظية أساسية	منصة متكاملة تشمل إدارة اللاعبين، المباريات، الافتتاحيات،

5 2.5. مقارنة بين مشروع ChessMate وChessInArabic

2.2.2.2 دراسة Chess-Game-Using-Socket.Io [28]:

1. الهدف:

الهدف الرئيسي للمشروع هو إنشاء تطبيق شطرنج تفاعلي في الزمن الحقيقي يسمح للاعبين بالتنافس ضد بعضهم البعض عبر الإنترنت، يركز المشروع على تجربة اللعب اللحظية وسرعة الاستجابة بين اللاعبين، مع إمكانية تبادل التحركات بين الأجهزة بشكل مباشر، لا يركز بشكل كبير على إدارة قواعد البيانات أو التحليلات المتقدمة للألعاب، بل الهدف هو تجربة لعب سلسة في الزمن الحقيقي.

2. المنهجية

يستخدم المشروع Node.js لإنشاء خادم (Server) قادر على التعامل مع الطلبات في الزمن الحقيقي.

يعتمد على Socket.IO لتبادل البيانات بين الخادم والعميل بشكل لحظي، مثل تحركات القطع، الدردشة، أو إشعارات اللعبة.

المنهجية تركز على الاستجابة اللحظية والتزامن بين اللاعبين، مع تركيز أقل على إدارة البيانات طويلة الأمد أو تحليلات الأداء.

لا يوجد دعم كبير لإدارة المستخدمين أو الإحصاءات أو الافتتاحيات بشكل مفصل.

3. النتائج

إنتاج تطبيق يتيح للاعبين اللعب ضد بعضهم البعض مباشرة.

توفير واجهة لعب سلسة وتحركات لحظية للقطع.

نتائج المشروع تركز على تجربة اللعب نفسها، دون إحصاءات تحليلية أو إدارة معقدة للألعاب والمستخدمين.

4. الربط بالمشروع

مشروع Kshiti-24 يمكن اعتباره نموذجًا أوليًا لتطبيقات الشطرنج الزمن الحقيقي باستخدام Socket.IO.

التركيز على اللعب اللحظي يوفر الأساس لتطوير ميزات إضافية، مثل إدارة قواعد البيانات والتحليلات، كما هو موجود في مشروعك.

الجدول 2.6. يقارن بين المشروعين

المحور	ChessInArabic	Kshiti-24 – Chess Game Using Socket.IO	الفرق الرئيسي / الربط بالمشروع
الهدف	دمج تجربة اللعب اللحظية مع إدارة المستخدمين، الألعاب، الافتتاحيات وتحليل الأداء	تجربة لعب شطرنج في الزمن الحقيقي بين لاعبين فقط	ChessInArabic أوسع ويشمل إدارة البيانات والتحليلات، بينما Kshiti-24 يركز على اللعب اللحظي فقط
المنهجية	Node.js + Socket.IO + MongoDB، نماذج Users, Games, Openings, Superadmin، دعم واجهة متعددة المستويات	Node.js + Socket.IO، التركيز على تبادل البيانات اللحظية بين اللاعبين	ChessInArabic أكثر شمولًا، ويعتمد على قاعدة بيانات منظمة وتحليلات للأداء
النتائج	منصة قابلة لإدارة وتحليل الألعاب، دعم الإحصاءات، قابلية توسعة الميزات مثل التقييم والتحديات	تجربة لعب سلسة في الزمن الحقيقي وسرعة استجابة اللعبة	ChessInArabic يقدم نتائج أكثر تنوعًا تشمل التحليل والإدارة، بينما Kshiti-24 يركز على تجربة اللعب فقط
الربط بالمشروع	تطوير متقدم للأفكار الأساسية في Kshiti-24، دمج اللعب اللحظي مع إدارة البيانات والتحليلات	نموذج أساسي للعب اللحظي باستخدام Socket.IO	ChessInArabic يمثل نسخة مطورة وقابلة للتوسع من Kshiti-24

6 2.6. مقارنة بين مشروع ChessInArabic وKshiti

2.2.3. خاتمة الفصل:

من خلال استعراض الدراسات السابقة في مجال تطوير ألعاب الشطرنج الإلكترونية، يتضح أن هناك تركيزًا كبيرًا على توفير تجربة لعب تفاعلية في الزمن الحقيقي باستخدام تقنيات مثل Node.js وSocket.IO، مع الاهتمام بتحسين أداء الاتصال بين اللاعبين. كما أفتقرت بعض الدراسات أهمية إدارة قواعد البيانات وتوثيق الحركات والافتتاحيات لدعم التحليل وتحسين تجربة المستخدم. وبناءً على هذه الدراسات، يظهر أن مشروع ChessInArabic يسعى إلى تطوير هذه التجربة بشكل متكامل من خلال دمج اللعب اللحظي مع نظام إدارة متكامل للمستخدمين والألعاب والافتتاحيات، مما يتيح إمكانية توسيع المشروع مستقبلًا لتشمل تحليلات متقدمة وتقييم أداء اللاعبين.

الفصل الثالث

الدراسة التحليلية والتصميمية

3.1. أهداف النظام

يهدف هذا المشروع إلى تطوير منصة إلكترونية للعب الشطرنج عبر الإنترنت تدعم التفاعل اللحظي وتوفر بيئة آمنة وسهلة الاستخدام للمستخدمين، وذلك من خلال تحقيق الأهداف الآتية:

3.1.1. توفير منصة عربية حديثة للعب الشطرنج عبر الإنترنت:

- دعم اللغة العربية وملاءمة الواجهة للمستخدم العربي.
- تمكين اللاعبين من خوض مباريات تفاعلية في أي وقت ومن أي مكان.
- إتاحة بيئة تدريبية تناسب المبتدئين والمحترفين على حد سواء.

3.1.2. ضمان تجربة لعب لحظية وسلسة:

- تقليل زمن الاستجابة (Low Latency) بما يضمن انسيابية التفاعل أثناء المباريات.
- إدارة الاتصالات المتزامنة دون التأثير على أداء الخادم.

3.1.3 تعزيز أمان المنصة وحماية بيانات المستخدمين:

- تطبيق نظام مصادقة آمن باستخدام .
- تأمين قاعدة البيانات ضد الاختراق أو التلاعب.

3.1.4 تصميم واجهة استخدام سهلة وفعّالة:

- توفير تصميم مبسط وواضح
- ضمان سهولة الوصول للاعبين من مختلف المستويات.
- دعم الاستخدام عبر مختلف الأجهزة.

3.1.5 إتاحة المجال للتوسع المستقبلي:

- تزويد المستخدمين بأدوات تحليلية متقدمة لفهم استراتيجيات الشطرنج.
- تصميم محتوى تعليمي لتلبية احتياجات المتعلمين على مختلف المستويات.
- توفير مسائل وأنشطة مناسبة لجميع مستويات اللاعبين، من المبتدئين إلى المتقدمين.
- إنشاء بيئة اجتماعية داخل المنصة تسمح بالتواصل وتبادل الخبرات بين اللاعبين والمتعلمين.
- توفير نسخة متوافقة مع أجهزة الهواتف الذكية لتعزيز إمكانية الوصول وتوسيع قاعدة المستخدمين.
- دعم خصائص إضافية مستقبلًا مثل البطولات الجماعية أو التصنيف العالمي للمستخدمين.

3.2. متطلبات النظام

ستقسم إلى قسمين متطلبات وظيفية ومتطلبات غير وظيفية

3.2.1 متطلبات وظيفية (Functional Requirements)

3.2.1.1 إدارة المستخدمين:

- يجب أن يتيح النظام للمستخدمين إنشاء حسابات (Sign-up) مع تشفير كلمات المرور لضمان أمان البيانات.
- يجب أن يسمح النظام تسجيل الدخول للمستخدمين والتحقق من صحة بياناتهم.
- يجب أن يتمكن النظام من التحقق من المستخدمين المسجلين قبل السماح لهم بالوصول إلى الميزات الأساسية.

3.2.1.2 إدارة المسؤولين (Admins):

- يجب أن يوفر النظام إمكانية تسجيل دخول المدراء.
- يجب أن يمكن المدراء من إجراء عمليات CRUD (إنشاء، قراءة، تعديل، حذف) للمستخدمين.
- يجب أن يكون للمدراء قاعدة بيانات خاصة لتخزين وإدارة بياناتهم بشكل منفصل عن المستخدمين العاديين.

3.2.1.3. ملفات المستخدمين والمعلومات الشخصية

- يجب أن يمكن النظام المستخدمين من عرض ملفهم الشخصي وتحديث بياناتهم الأساسية.
- يجب تخزين بيانات المستخدمين بشكل منظم وآمن.

3.2.1.4. تخزين وإدارة بيانات اللعبة

- يجب أن يوفر النظام تخزين معلومات المباراة لكل مباراة يتم لعبها.
- يجب أن يوفر النظام قاعدة بيانات للافتتاحيات لدعم التعلم والتحليل الاستراتيجي.
- يجب أن يسمح النظام بفتح اتصال مباشر (real-time) للعب المباشر بين اللاعبين.
- يجب أن يتم مصادقة المستخدمين قبل السماح باللعب لضمان أمان المنصة.

3.2.2. المتطلبات غير الوظيفية (Non-Functional Requirements)

3.2.2.1 قابلية التوسع (Scalability)

- يجب تصميم النظام بحيث يمكن إضافة ميزات جديدة مستقبلاً بسهولة، مثل البطولات الجماعية أو تصنيف اللاعبين العالمي.

3.2.2.2. خصوصية وأمان المستخدمين (Security & Privacy)

- يجب حماية بيانات المستخدمين والمعلومات الشخصية من الوصول غير المصرح به أو الاختراق.
- يجب تشفير كلمات المرور وتأمين الاتصال بين العميل والخادم.

3.2.2.3. الأداء (Performance)

- يجب أن يكون أداء النظام سلسًا، مع زمن استجابة منخفض أثناء اللعب المباشر والتحليل الفوري للنقلات.

3.2.2.4. موثوقية النظام (Reliability)

- يجب أن يضمن النظام استقرار العمليات الأساسية مثل تسجيل الدخول، إدارة الحسابات، وتخزين البيانات، دون أخطاء أو فقدان معلومات.

يظهر الجدول 3.1. متطلبات النظام

رقم	نوع المتطلب	المتطلب	الوصف / الهدف
1	وظيفي	تسجيل المستخدم(Sign-up)	يتيح للمستخدمين إنشاء حسابات جديدة مع تشفير كلمات المرور لضمان الأمان.
2	وظيفي	تسجيل الدخول(Login)	يسمح للمستخدمين بالوصول إلى حساباتهم بعد التحقق من بياناتهم.
3	وظيفي	التحقق من المستخدمين المسجلين	التأكد من صحة حساب المستخدم قبل السماح بالوصول للميزات الأساسية للمنصة.

رقم	نوع المتطلب	المتطلب	الوصف / الهدف
4	وظيفي	تسجيل دخول المدراء (Admin Login)	يتيح للمدراء الدخول إلى النظام للتحكم وإدارة المستخدمين.
5	وظيفي	إدارة المستخدمينCRUD	تمكين المدراء من إنشاء، قراءة، تعديل، وحذف حسابات المستخدمين.
6	وظيفي	عرض ملف المستخدم الشخصي	يتيح للمستخدمين استعراض وتحديث بياناتهم الشخصية.
7	وظيفي	تخزين بيانات المستخدمين	حفظ معلومات المستخدمين بشكل منظم وآمن في قاعدة البيانات.
8	وظيفي	تخزين معلومات المباراة	تسجيل تفاصيل المباريات لتسهيل والمتابعة.
9	وظيفي	قاعدة بيانات للافتتاحيات	توفير مكتبة افتتاحيات لدعم التعلم.
10	وظيفي	قاعدة بيانات المدراء	تخزين معلومات المدراء بشكل منفصل لضمان الخصوصية والأمان.
11	وظيفي	مصادقة المستخدمين قبل اللعب	التأكد من أن جميع اللاعبين مسجلون وصالحون قبل بدء أي مباراة.
12	وظيفي	فتح اتصال مباشر real-time للعب	تمكين اللعب المباشر بين اللاعبين بشكل فوري وسلس.
13	غير وظيفي	قابلية التوسع(Scalability)	تصميم النظام بحيث يمكن إضافة ميزات جديدة مستقبلاً بسهولة.

رقم	نوع المتطلب	المتطلب	الوصف / الهدف
14	غير وظيفي	خصوصية وأمان المستخدمين	حماية بيانات المستخدمين وتأمين كلمات المرور والاتصال بين العميل والخادم.
15	غير وظيفي	أداء سلس(Performance)	ضمان استجابة سريعة للنظام أثناء اللعب المباشر والتحليل الفوري.
16	غير وظيفي	موثوقية النظام(Reliability)	استقرار العمليات الأساسية مثل تسجيل الدخول وإدارة الحسابات وتخزين البيانات دون أخطاء.

7 3.1. متطلبات الوظيفية والغير وظيفية

3.2. مخططات النظام

مخططات النظام هي تمثيلات رسومية تُستخدم لتحليل وتصميم النظام قبل برمجته، وتساعد على فهم:

- مكونات النظام.
- طريقة تفاعل المستخدمين مع النظام.
- تدفق البيانات والعمليات داخله.

أهمية مخططات النظام:

- تحسين التواصل بين فريق المشروع والمشرفين.
- تسهيل اكتشاف الأخطاء في التصميم قبل التنفيذ.
- توفير وثائق مفصلة للنظام يمكن الرجوع إليها لاحقًا.

3.2.1. مخطط النشاط (Activity Diagram)

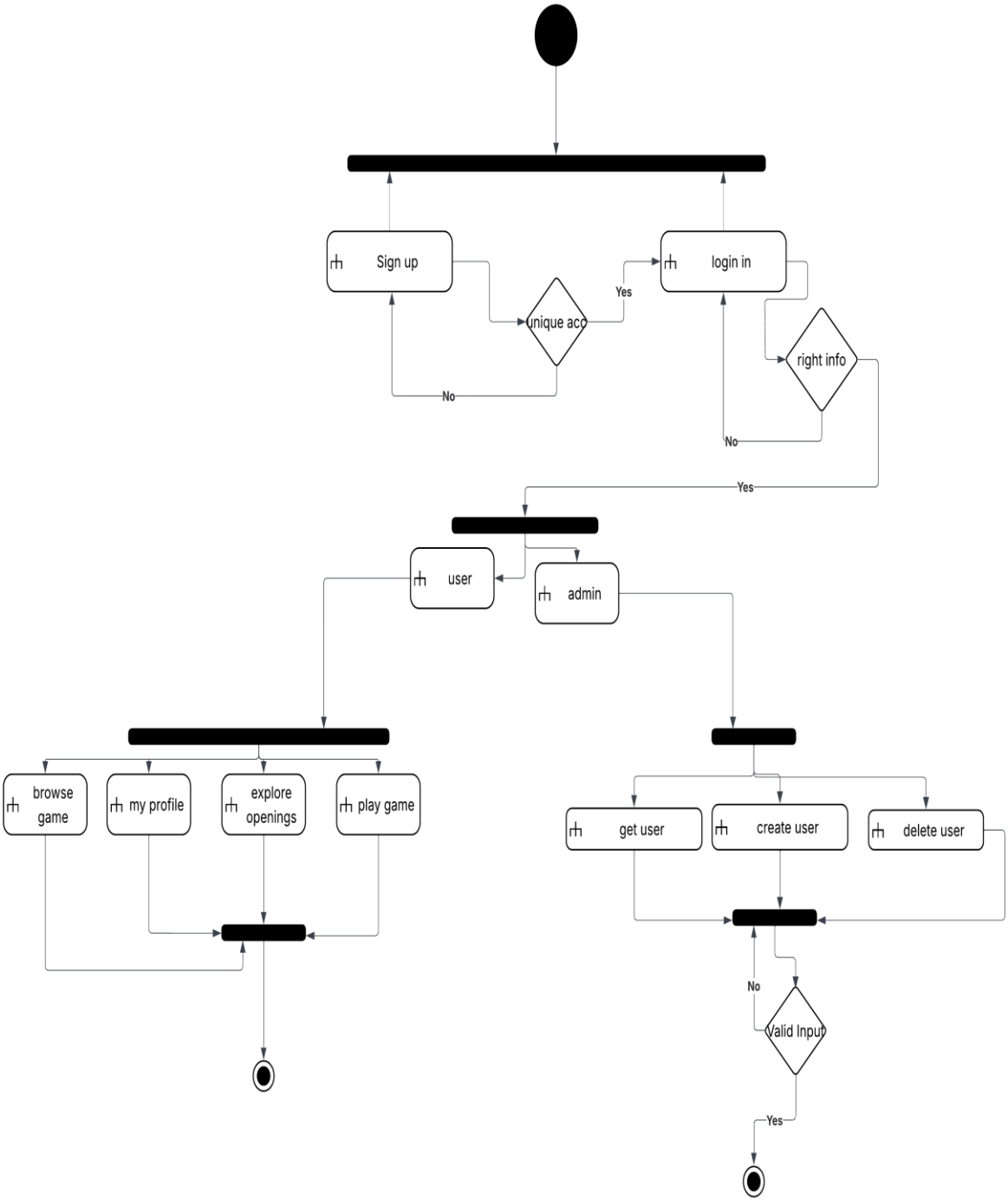
يُعد مخطط النشاط (Activity Diagram) أحد المخططات الأساسية في لغة UML (Unified Modeling Language)، ويُستخدم لتمثيل تدفق الأنشطة (Workflow) أو تتابع العمليات داخل النظام.

يُشبه هذا المخطط في بعض جوانبه مخططات التدفق (Flowcharts)، لكنه أكثر قوة من الناحية الوصفية لأنه يوضح حالات التفرع، التوازي، واتخاذ القرارات داخل النظام.

الهدف من مخطط النشاط:

- توضيح كيفية تنفيذ العمليات داخل النظام خطوة بخطوة.
- تمثيل سلوك النظام عند تنفيذ وظيفة معينة (مثل تسجيل الدخول، أو إجراء معاملة).
- إعطاء صورة أوضح عن سير العمليات للمستخدمين أو المبرمجين أو المصممين.
- دعم التوثيق الأكاديمي للمشروع من خلال عرض الجانب الديناميكي (Dynamic View) للنظام.

الشكل 3.1. يوضح مخطط النشاط للمشروع



4 الشكل 3.1. مخطط النشاط

3.2.2 مخطط الحالات (Use Cases)

يُعد مخطط الحالات (Use Case Diagram) أحد المخططات الرئيسية في لغة UML، ويُستخدم لتوضيح المتطلبات الوظيفية للنظام من وجهة نظر المستخدمين (Actors).

يُظهر المخطط التفاعل بين الممثلين (Actors) و حالات الاستخدام (Use Cases)، مما يساعد على فهم ما يقدمه النظام من خدمات.

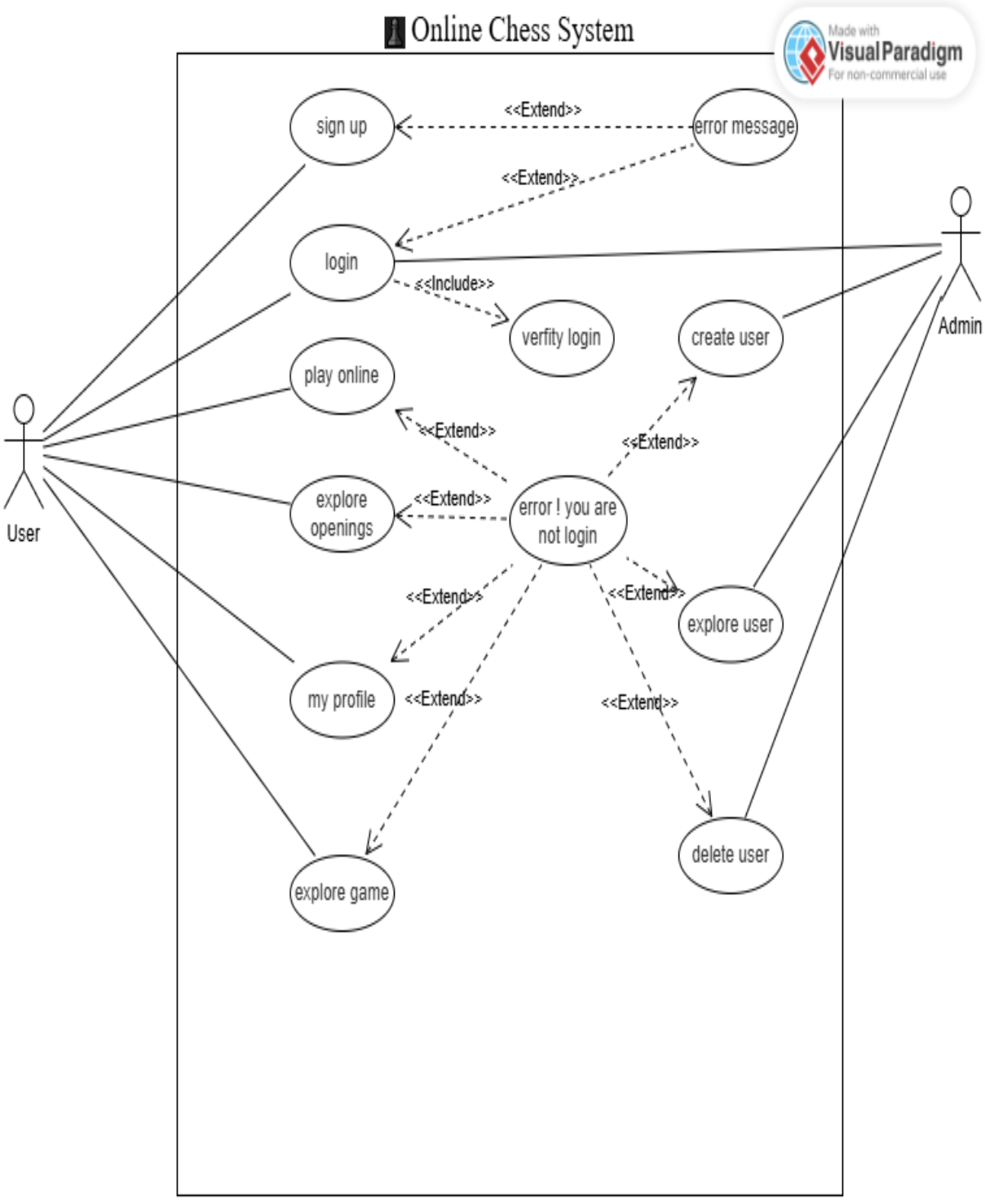
الهدف من مخطط الحالات:

- تحديد الوظائف الأساسية التي يجب أن يدعمها النظام.
- بيان العلاقة بين النظام والمستخدمين الخارجيين.
- توفير نظرة عامة عالية المستوى عن نطاق النظام (System Scope).
- تسهيل التواصل بين المبرمجين، المصممين، وأصحاب المشروع.

فوائد استخدام مخطط الحالات:

- يساعد على توضيح متطلبات النظام قبل البدء في التصميم البرمجي.
- يُعتبر أداة تواصل بصرية بين الفريق الأكاديمي (طلاب، مشرف، لجنة).
- يسهل تحديد أولويات التطوير ومعرفة السيناريوهات الرئيسية.

يظهر الشكل 3.2. مخطط الحالة للنظام



5 الشكل 3.2. مخطط الحالة

3.2.3 مخطط الكيانات (ER Digram)

ERD أو Entity-Relationship Diagram هو مخطط يُستخدم لتصميم قاعدة البيانات بصريًا، ويظهر الكيانات (Entities) وعلاقاتها (Relationships) والصفات (Attributes) الخاصة بكل كيان.

Entity (كيان): عنصر يمثل مجموعة من البيانات المتشابهة، مثل: Student أو Game.

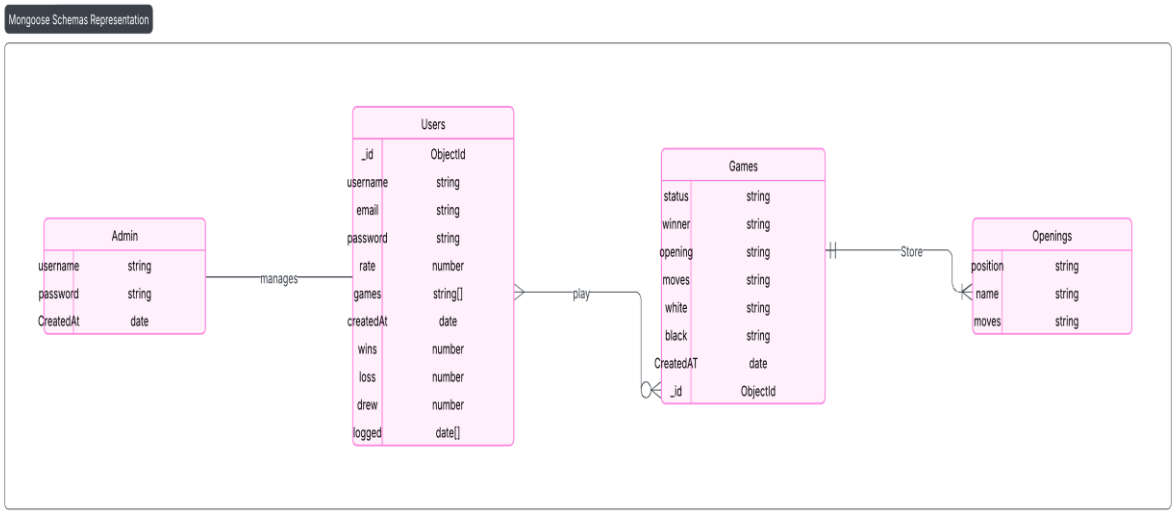
Attribute (خاصية): سمة تصف الكيان، مثل: name أو email.

Relationship (علاقة): يوضح كيف يرتبط كيان بآخر، مثل: Student يسجل في Course.

أهمية ERD:

- توضيح الهيكل العام للبيانات.
- تقليل الأخطاء عند التصميم.
- تسهيل فهم المشروع للمطورين والمشرفين.

يشير الشكل 3.3. إلى مخطط ER في المشروع



الشكل 3.3 مخطط ER 6

الفصل الرابع

التطبيق البرمجي

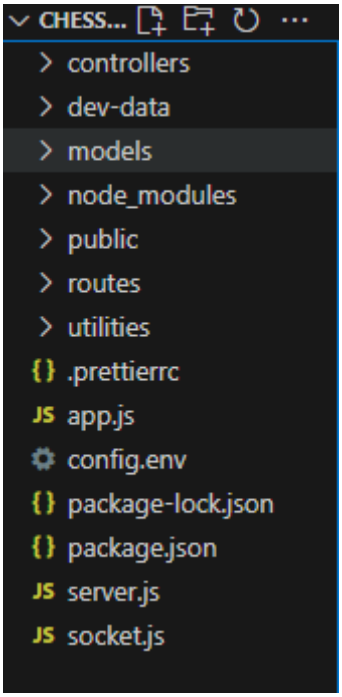
4.1. مقدمة

يهدف هذا الفصل إلى توضيح كيفية تحويل التحليل والتصميم الذي تم في الفصول السابقة إلى نظام برمجي يعمل بشكل فعلي. بالإضافة إلى شرح المكونات الأساسية مثل الواجهة الأمامية التي يتفاعل معها المستخدم، والواجهة الخلفية المسؤولة عن معالجة البيانات وتنفيذ العمليات

كما يتناول هذا الفصل عرض أهم المقاطع البرمجية مع شرح وظيفتها، مثل كود تسجيل الدخول، وكود إدارة الألعاب، وكود التعامل مع قاعدة البيانات..

4.1.1. بنية ملفات النظام

الشكل 4.1. يظهر بنية ملفات النظام



الشكل 4.1. بنية ملفات النظام 7

تم تنظيم النظام البرمجي ضمن مجموعة من المجلدات والملفات، بحيث يسهل ذلك عملية التطوير والصيانة. فيما يلي شرح هذه المكونات:

- مجلد Controllers: يضم الملفات التي تحتوي على الدوال (Functions) المسؤولة عن منطق المعالجة والتي يتم استدعاؤها في الأجزاء الأخرى من النظام.

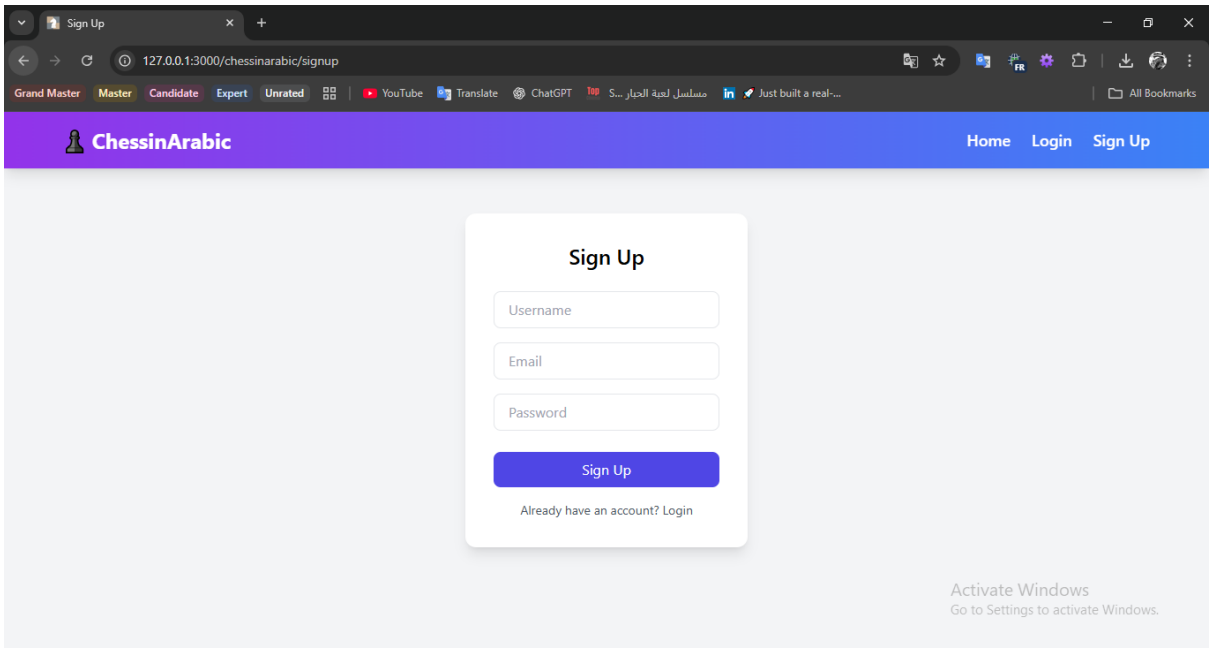
- مجلد dev-data: يحتوي على ملف بصيغة JSON يتضمن بيانات افتتاحيات الشطرنج (حوالي 11 ألف افتتاحية)، وقد تم استخدامه لإدخال هذه البيانات في قاعدة البيانات.
- مجلد Models: يشمل تعريفات قواعد البيانات من حيث إنشاء المخططات (Schemas) وطرق الاتصال بها.
- مجلد node_modules: يخزن مكتبات Node.js المثبتة عبر مدير الحزم (npm).
- مجلد Public: يضم الملفات المرئية للمستخدم النهائي، مثل ملفات HTML, CSS, JavaScript، بالإضافة إلى مكتبات خاصة بتصميم الواجهة الأمامية وبعض الصور.

- مجلد Routes: يحتوي على جميع المسارات (Routes) التي تحدد عناوين وواجهات النظام.
- مجلد Utilities: يضم ملفات تحتوي على دوال مساعدة للاستخدامات العامة غير المرتبطة مباشرةً بالـ Controllers.
- ملف .prettierrc: ملف خاص بأداة Prettier لتنسيق الكود وجعله أكثر وضوحًا وانسجامًا.
- ملف App.js: يمثل نقطة التطبيق الخاصة بـ Express.js حيث يتم استدعاء المسارات وإضافة الـ Middleware.
- ملف Config.env: يتضمن الإعدادات الأساسية للنظام والتي يمكن تخصيصها مثل بيانات الاتصال وقيم المتغيرات البيئية.
- ملف Package-lock.json: ملف تلقائي يتم إنشاؤه بواسطة npm يحدد الإصدارات الدقيقة لجميع المكتبات في مجلد node_modules.
- ملف Package.json: يحتوي على معلومات المشروع وقائمة بالمكتبات التي قام المطور بتثبيتها، بالإضافة إلى بعض التفاصيل الأخرى.
- ملف Server.js: يعد الملف الرئيسي الذي يقوم بتشغيل الخادم لخدمة المستخدمين (Clients).
- ملف Socket.js: مسؤول عن إدارة الاتصالات باستخدام بروتوكول WebSocket لتسهيل التفاعل اللحظي بين المستخدمين

4.2. واجهات النظام

4.2.1. إنشاء حساب Signup

الشكل 4.2. يظهر واجهة إنشاء حساب مستخدم



الشكل 4.2. واجهة إنشاء حساب مستخدم 8

يقوم المستخدم بإدخال اسم الحساب والايميل وكلمة السر لينشئ حساب مستخدم جديد

الكود البرمجي

```
const { username, password, email } = req.body;
const user = await new usersModels({ username, password, email }).save();
```

مبدأ العمل يقوم السيرفر بأخذ username، password، email المرسلة ضمن ال request من client

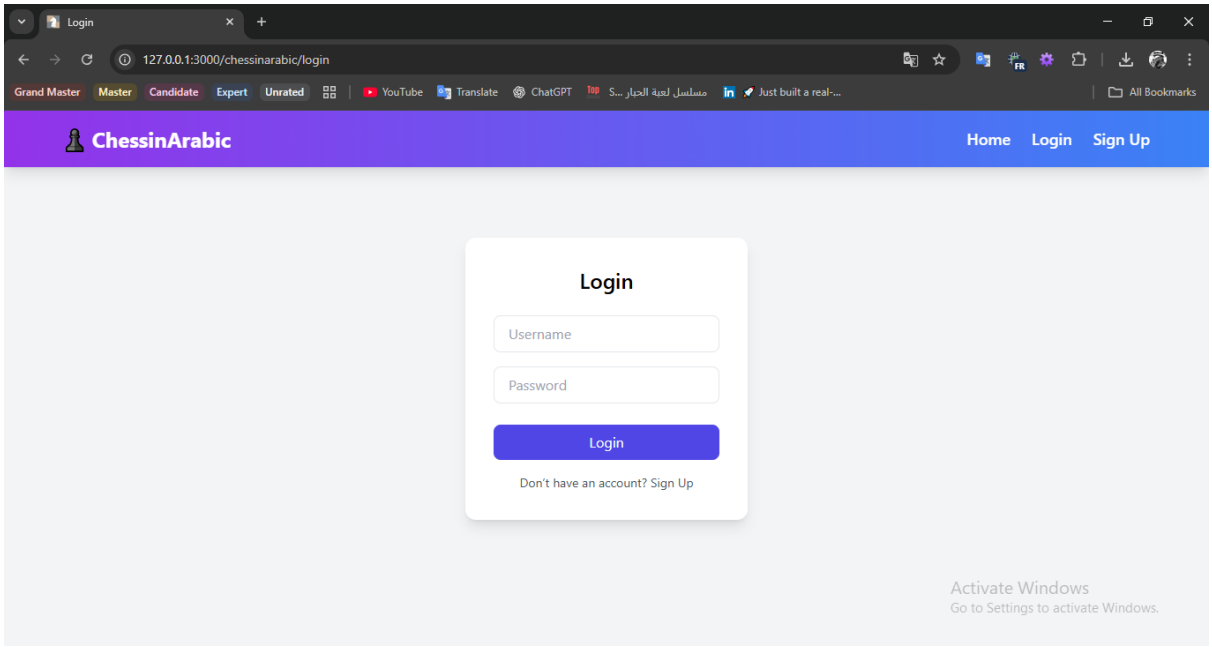
وينشئ حساب مستخدم جديد بحال وافق شروط قاعدة البيانات ويشفر كلمة السر تلقائياً عند أي حفظ

```
this.password = await bcrypt.hash(
  this.password,
  process.env.BCRYPT_SALT * 1
);
```

يتم إستدعاء هذه الدلة عند كل إنشاء لمستخدمين جدد ،تقوم الأداة bcrypt بتشفير كلمة السر وارجاع قيمة جديدة مشفرة

4.2.2 تسجيل دخول Login

الشكل4.3. يظهر واجهة تسجيل دخول مستخدم



الشكل 4.3. واجهة تسجيل دخول مستخدم 9

يقوم المستخدم بإدخال اسم الحساب وكلمة السر بحال كانت المعلومات صحيحة يسجل دخول

الكود البرمجي

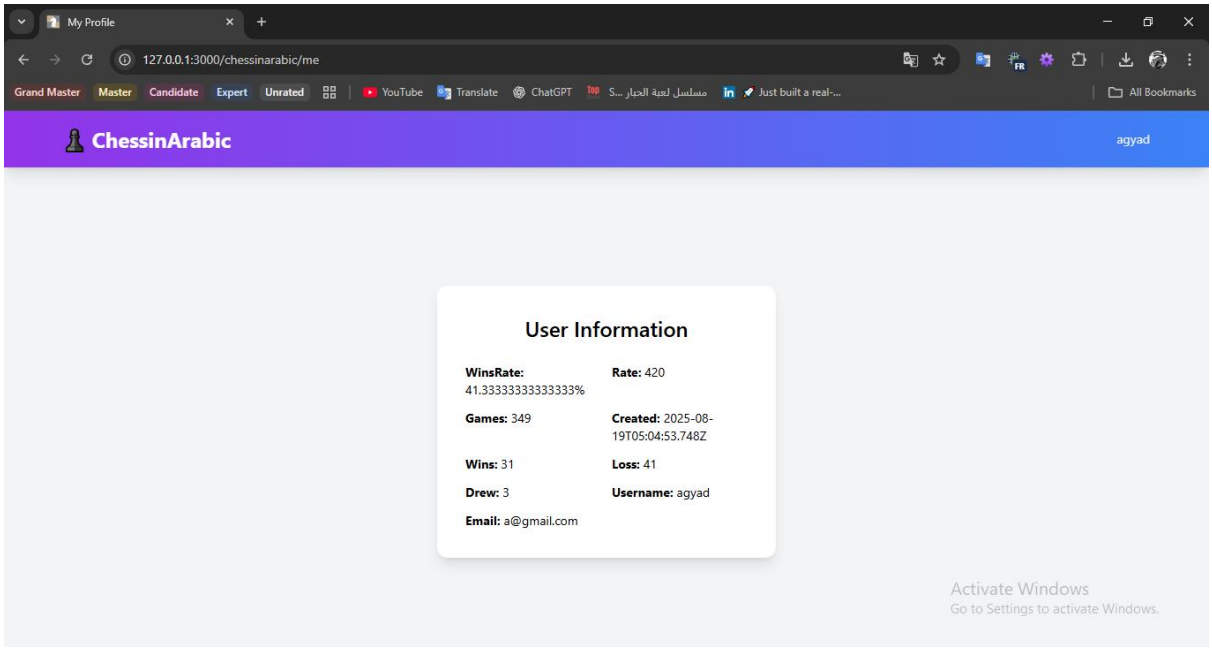
```
const user = await usersModels
  .findOne({ username: req.body.username })
  .select('+password');
if (user) {
  // 2 - compare password
  const password = await hashPassword.compare(
    req.body.password,
    user.password
  );
  if (password) {
    // 3 - create token
    const token = jwt.sign({ _id: user._id }, process.env.JWT_SECRET, {
      expiresIn: process.env.JWT_EXPIRES_IN * 1,
    });
```

يقوم السيرفر بالبحث بقاعدة البيانات user عن مستخدم بنفس الاسم بحال وجد يقارن كلمة السر المشفرة بالمدخلة عن طريق أداة التشفير bcrypt يستدعيها من ملف hasspassword، في حال تطابق كلمة المرور، يقوم الخادم بإنشاء رمز وصول JWT (JSON Web Token):

يتم تضمين معرف المستخدم (_id) كـ Payload، يستخدم مفتاح سري (JWT_SECRET) مخزن في Environment Variables لتشفير الرمز.، وثم ارسل ال token إلى client

4.2.3. ملفي My Profile

الشكل 4.4. يظهر واجهة معلومات ملف المستخدم



10 الشكل 4.4. My Profile

تظهر هذه الواجهة معلومات المستخدم بعد ان يقوم بتسجيل دخول

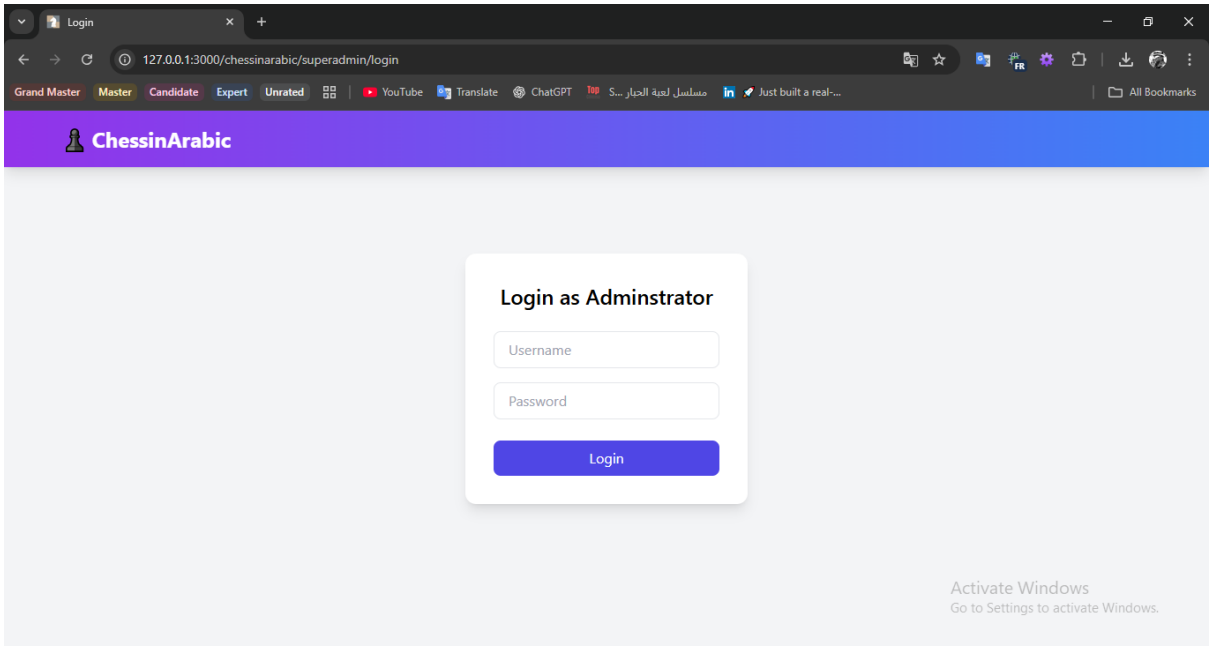
الكود البرمجي

```
    const token = req.cookies.jwt;
    const decod = jwt.verify(token, process.env.JWT_SECRET);
    const user = await usersModels.findById(decod._id);
```

للتحقق من صحة تسجيل الدخول، للتحقق من صحة تسجيل الدخول، استلام الرمز (JWT Token) المخزن في ملفات الكوكيز (Cookies) من العميل، التحقق من صحة الرمز باستخدام مفتاح سري (JWT_SECRET) لاسترجاع المعرف المخزن (_id) في البايلو، البحث في قاعدة البيانات عن المستخدم الذي يمتلك هذا المعرف، في حال وجود المستخدم، تتم المصادقة بنجاح ويتم إرسال بيانات الحساب إلى العميل لعرضها في واجهة ملف المستخدم

4.2.4. واجهة تسجيل دخول للمدراء

الشكل 4.5. يظهر واجهة تسجيل دخول مدراء



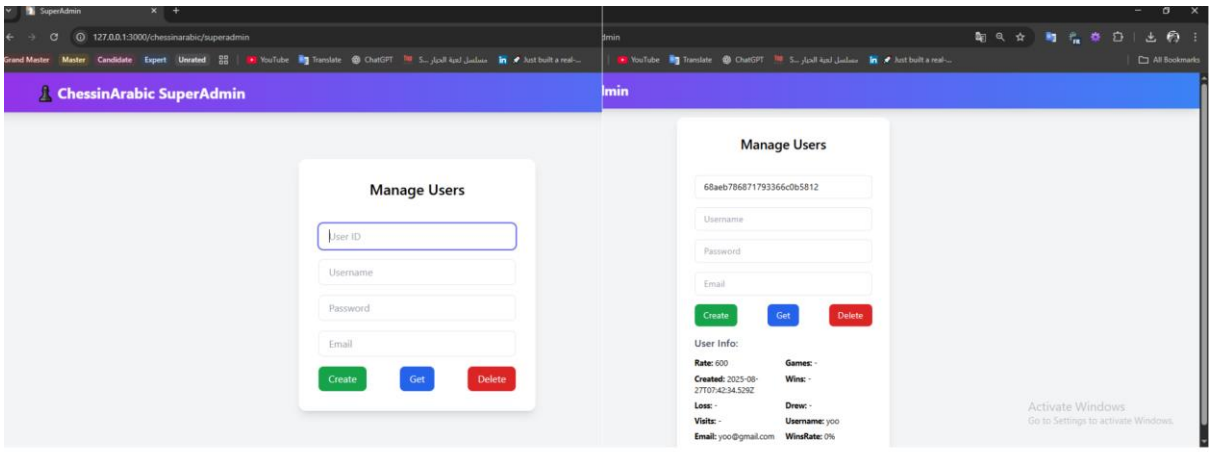
الشكل 4.5. واجهة تسجيل دخول مدير 11

يقوم المدير بإدخال اسم الحساب وكلمة السر بحال كانت المعلومات صحيحة يسجل دخول كمدير

الكود البرمجي نفس تسجيل دخول مستخدم ولكن الفرق عللى قاعدة بيانات المدراء

4.2.5. واجهة تحكم مدير

يظهر الشكل 4.6. واجهة تحكم مدير



الشكل 4.6. واجهة تحكم مدير 12

بعد تحقق من تسجيل الدخول يستطيع المدير القيام بثلاث مهام:

1. اظهار معلومات مستخدم عن طريق id_

الكود البرمجي

```
const ID = req.params.id;
const user = await usersModels.findById(ID);
```

يقوم الكود باالبحث عن مستخدم من خلال id_ الذي ادخله المستخدم ويرجع النتيجة بحال كانت صحيحة

2. حذف حساب مستخدم عن طريق id_

الكود البرمجي

```
    const ID = req.params.id;
    const user = await usersModels.findByIdAndDelete(ID);
```

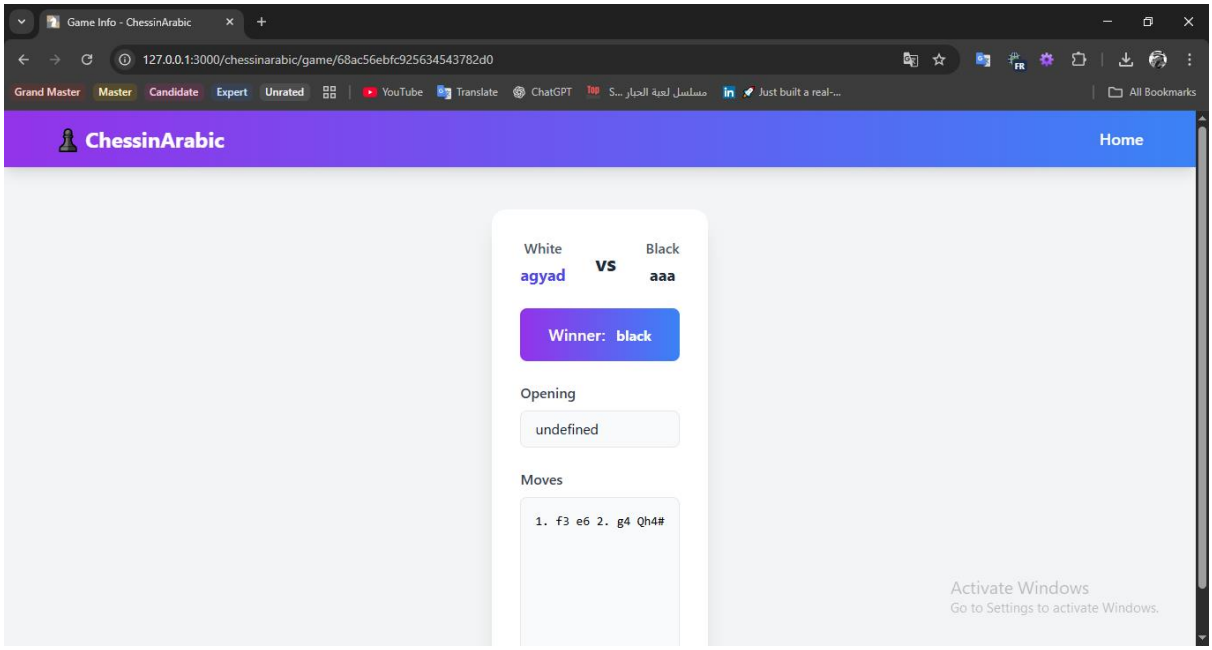
يقوم الكود باالبحث عن مستخدم من خلال id_ الذي ادخله المستخدم وحذف المستخدم بحال كانت صحيحة

3. إنشاء حساب مستخدم بعد ان يتم تعبئة الحقول email، password، username

الكود نفس صفحة signup للمستخدم

4.2.6. واجهة معلومات اللعب

يظهر الشكل 4.7. واجهة لعب



13 الشكل 4.7. واجهة تصفح مباريات

تقوم هذه الواجهة بعرض معلومات كل مباراة تم لعبها، مثل: اسم اللاعبين، الفائز، الافتتاحية، وحركات المباراة.

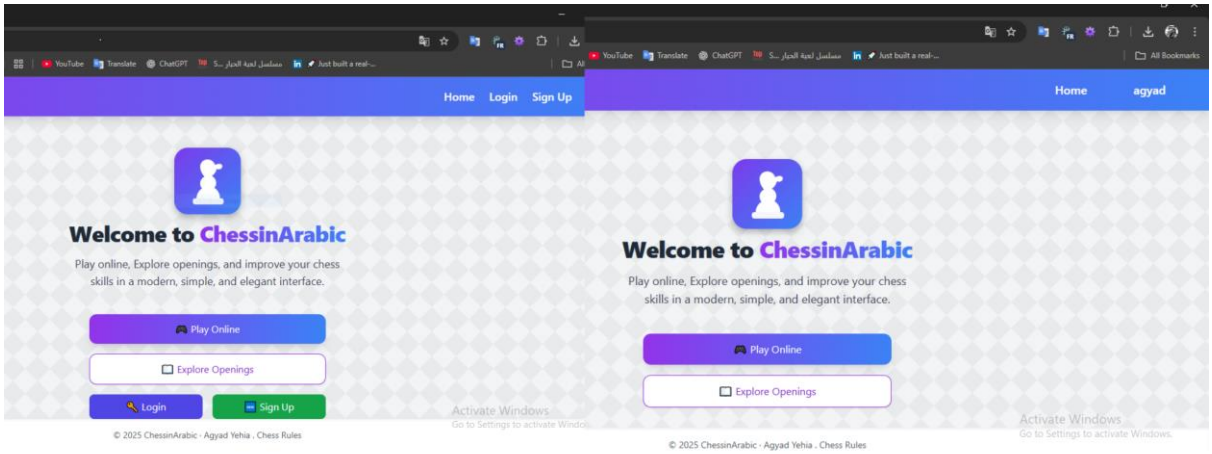
الكود البرمجي

```
const id = req.params.id;
if (id) {
  const game = await gamesModels.findOne({ _id: id });
  if (game) {
    let ResponeFile = GamePage.replace(/%{WHITE}%/g, game.white);
    ResponeFile = ResponeFile.replace(/%{BLACK}%/g, game.black);
    ResponeFile = ResponeFile.replace(/%{WINNER}%/g, game.winner);
    ResponeFile = ResponeFile.replace(/%{OPENING}%/g, game.opening);
    ResponeFile = ResponeFile.replace(/%{MOVES}%/g, game.moves);
```

استلام معرف المباراة (id) من طلب العميل، البحث في قاعدة البيانات عن المباراة المطابقة للمعرف، إذا تم العثور على المباراة، يتم استبدال العلامات النصية المخصصة في ملف HTML بالقيم الفعلية للمباراة

4.2.7. واجهة الرئيسية

يظهر الشكل 4.8. واجهة المشروع الرئيسية



14 الشكل 4.8. واجهة النظام الأساسية

عندما يقوم المستخدم بتسجيل الدخول يظهر اسمه في الأعلى وتخفئ خيارات التسجيل وتسجيل الدخول

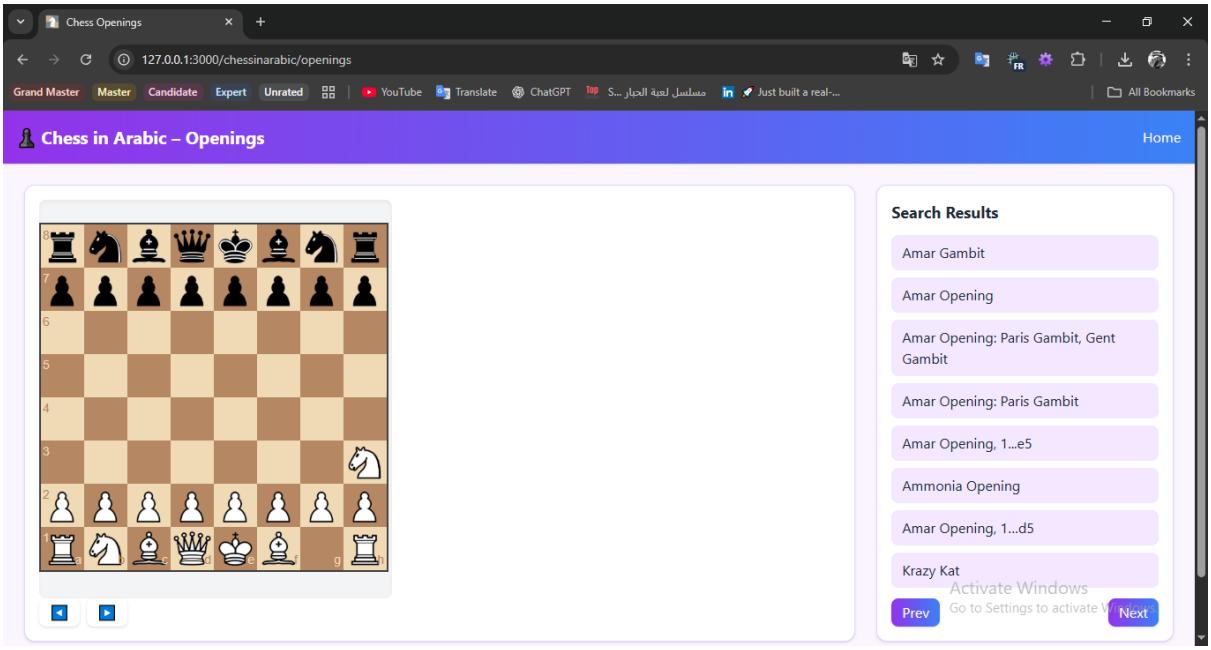
الكود

```
ResponeFile = ResponeFile.replace(/%hide-login%/g, 'hide-login');
ResponeFile = ResponeFile.replace(
  /{%username%}/g,
  req.CURRENTUSER.username
);
ResponeFile = ResponeFile.replace(/Display-User-Name/g, ' ');
```

بعد أن تأكد من تسجيل الدخول يستبدل كلاس %hide-login% بكلاس hide-login المعد لإخفاء خيارات التسجيل ويستبدل النص {%username} باسم المستخدم، مسح كلاس Display-user-Name الذي كان يخفي اسم المستخدم

4.2.8. واجهة تصفح الإفتتاحيات

يظهر الشكل 4.9. واجهة تفاعلية لتصفح الإفتتاحيات



15 الشكل 4.9. واجهة تصفح تفاعلية للإفتتاحيات

يقوم المستخدم عند الدخول إلى الصفحة بتحريك القطع وبعد كل حركة يقوم بعرض 20 إفتتاح كحد أقصى حسب النقلات ويمكن المستخدم طلب المزيد من زر next، ويمكنه التراجع عن النقلة أو العودة من خلال أزرار الأسهم بالكيبورد أو الأسهم بالواجهة

الكود ضخم في جانب المخدم والعميل لذلك سأكتفي بمناقشة أهم الأمور وكامل الكود سيرفع على Gtihub [29]

الكود البرمجي

```
const moves = `^${req.query.moves}`.trim().replace(/\s+/g, '\\s*');
const openings = await openingsModel
  .find({ _id: { $gt: new Types.ObjectId(req.query.curser) }, moves: { $regex:
moves, $options: 'i' } })
  .sort({ _id: 1 })
  .limit(20);

res.json({ status: true, data: { openings, nextcurser: openings.at(-1)._id }
});
```

يتم أولًا تجهيز نص الحركات المدخلة ليُستخدم في عملية البحث باستخدام التعبير المنتظم (Regex)، بعد ذلك يتم جلب 20 افتتاحية فقط من قاعدة البيانات بحيث تكون بعد آخر عنصر تم عرضه سابقًا (باستخدام الـ Cursor)، أخيرًا، يتم إرسال النتائج إلى العميل مع تحديد المؤشر التالي ليُستخدم في استدعاء الطلب القادم.

الكود البرمجي على جانب العميل

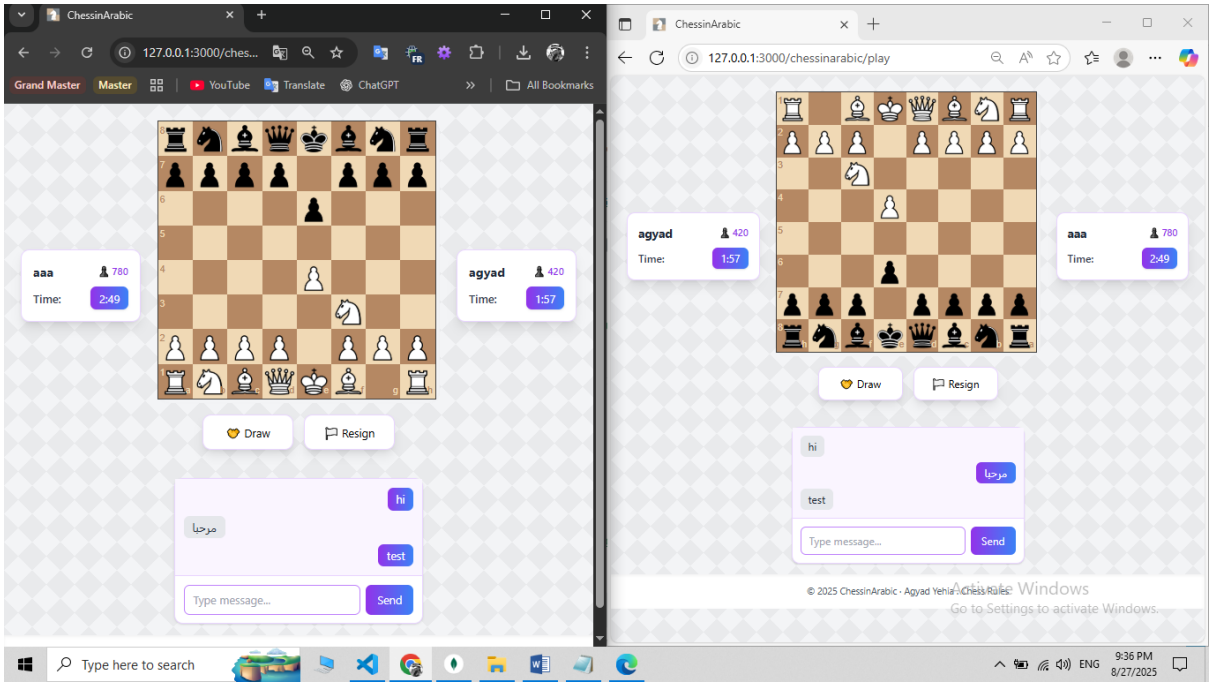
```
const res = await
fetch(`/chessinarabic/openings/data?moves=${game.pgn()}&curser=${curser}`);
const { status, data } = await res.json();

resultsID.innerHTML = '';
if (status && data.openings.length) {
  data.openings.forEach(o => {
    const child = Object.assign(document.createElement('div'), { textContent:
o.name });
    child.className = 'bg-purple-100 text-gray-800 px-3 py-2 rounded-lg
hover:opacity-90 active:scale-95 transition cursor-pointer';
    resultsID.appendChild(child);
  });
  nextcurser = data.nextcurser;
} else {
  resultsID.innerHTML = '<div>nothing found</div>';
}
```

يبدأ الكود بإرسال طلب HTTP GET إلى الخادم للحصول على بيانات الافتتاحيات بناءً على الحركات (moves) والمُؤشر (cursor)، إذا تم العثور على نتائج، يتم إنشاء عناصر لكل افتتاحية وإضافتها للواجهة مع تنسيقات جاهزة من مكتبة Talwind، في حال عدم وجود نتائج، يتم عرض رسالة بديلة تفيد بـ "nothing found".

4.2.9. واجهة لعب شطرنج

يظهر الشكل 4.10. واجهة لعب بين شخصين بنفس الوقت



الشكل 4.10. واجهة لعب بين شخصين بنفس الوقت 16

عندما يتوفر لاعبين اثنين يمكن لهما بدأ مباراة، ويمكنهم التحدث ضمن المباراة وعرض تعادل أو استسلام

الكود ضخم في جانب المخدم والعميل لذلك سأكتفي بمناقشة أهم الأمور وكامل الكود سيرفع على Gtihub [29]

الكود البرمجي

```
io = new Server(server);
let Players = [];

io.on('connection', async (socket) => {
  // 1- chat in room
  socket.on('chat-message', (msg, room) => socket.to(room).emit('chat-
message', msg));

  socket.on('offer-draw', (room) => socket.to(room).emit('offer-draw'));

  // 2- add players to lobby
  try {
    const Player = await socketControllers.addToLobby(socket);
    Players.push(Player);
  } catch {
```

```
    return socket.disconnect(true);
}

// 3- start game when players>=2
if (Players.length >= 2) {
    socketControllers.startGame(Players[0], Players[1]);
    Players.splice(0, 2);
}

// 4- move-pices
socket.on('make-move', (room, source, target, promotion) =>
    socket.to(room).emit('make-move', room, source, target, promotion)
);

// 5- checkmate or time
socket.on('checkmate-timeout', async (room, info, moves) => {
    await socketControllers.checkmate(room, info, moves);
    io.to(room).emit('game-over', false); // false = not a drew
    (await io.in(room).fetchSockets()).forEach((s) => s.leave(room));
});

// 6- drew
socket.on('draw', async (room, info, moves) => {
    await socketControllers.drew(room, info, moves);
    io.to(room).emit('game-over', true); // true = drew
    (await io.in(room).fetchSockets()).forEach((s) => s.leave(room));
});
```

يعتمد Socket.io على مبدأ الأحداث حيث يستقبل أحداث من العملاء ويولد أحداث أخرى لهم ويرسلها إلى غرف مخصصة أو إلى الجميع

الأحداث الرئيسية في الكود

- الاتصال: عند اتصال لاعب جديد، يتم إضافته إلى قائمة الانتظار (Players).
- الدردشة: يمكن للاعب إرسال رسالة داخل غرفة، وتصل لجميع الموجودين في الغرفة.
- طلب التعادل: لاعب يرسل طلب تعادل للخصم.
- بدء اللعبة: عندما يتوفر لاعبان أو أكثر في قائمة الانتظار، يبدأ النظام مباراة جديدة بين أول لاعبين.
- تحريك القطع: كل حركة يقوم بها لاعب تُرسل إلى الخصم في الغرفة.
- التعادل بسبب الوقت: إذا انتهى الوقت دون فوز، يتم إعلان التعادل.

- إنهاء اللعبة: سواء بكش مات أو اتفاق على التعادل، يرسل النظام رسالة إنهاء لجميع اللاعبين في الغرفة.

على جانب العميل تم تلخيص الكود إلى مجموعة من الأحداث الرئيسية

```
// 1. تهيئة لوحة الشطرنج
const board = Chessboard('myBoard', {
  draggable: true,
  position: 'start',
  onDragStart: handleDragStart,
  onDrop: handleDrop,
  onSnapEnd: handleSnapEnd
});

function onDragStart(source, piece, position, orientation) {}
function onDrop(source, target) {}
function onSnapEnd() {}
$resignBtn.on('click', () => {});
$drawBtn.on('click', () => {});
$ChatSend.on('click', () => {});
socket.on('chat-message', (msg) => {});
socket.on('offer-draw', () => {});
socket.on('get-information-game', (gameRoom, myname, opponentname,
orientation, myrate, opponentrate) => {});
socket.on('make-move', (room, source, target, promotion) => {});
socket.on('game-over', (draw) => {});
socket.on('disconnect', () => {});
```

1. تهيئة لوحة الشطرنج
   - إنشاء لوحة قابلة للسحب(`draggable`)
   - تعيين الأحداث`onDragStart`, `onDrop`, `onSnapEnd` :
2. سحب القطع(onDragStart)
   - منع الحركة إذا انتهت اللعبة
   - منع اللاعب من تحريك قطع الخصم أو خارج دوره
3. إسقاط القطع(onDrop)
   - التحقق من قانونية الحركة
   - إرسال الحركة للخصم عبر الخادم
   - تبديل اللاعب النشط
4. (onSnapEnd)

- تحديث موقع القطع حسب حالة اللعبة

5. أزرار التحكم

- الاستسلام: إعلان فوز الخصم
- عرض التعادل: إرسال طلب تعادل للخصم

6. الدردشة الحية

- إرسال الرسائل
- عرض الرسائل الواردة من الخصم

7. التعامل مع طلب التعادل

- إرسال القرار للخصم

8. استقبال معلومات اللعبة من الخادم

- ضبط اللوحة والاتجاه
- بدء اللعبة والموقت
- تهيئة حالة اللاعبين

9. استقبال حركة الخصم

- التحقق من انتهاء اللعبة (كش ملك أو تعادل)

10. نهاية اللعبة وفصل الاتصال

- إظهار رسالة الفائز أو التعادل
- إعادة توجيه المستخدم عند انتهاء اللعبة أو الانقطاع

الفصل الخامس

النتائج والآفاق المستقبلية

5.1. النتائج

قدم الموقع مجموعة من الوظائف الأساسية التي تم تنفيذها والتحقق من عملها بنجاح حتى المرحلة الحالية، وتشمل:

5.1.1 إنشاء حساب المستخدم (Signup)

الوصف: تم تطوير واجهة تسجيل حساب جديد للمستخدمين.

الوظيفة: يمكن للمستخدم إدخال اسم الحساب، البريد الإلكتروني، وكلمة المرور لإنشاء حساب.

التحقق: يتم التحقق من صحة البيانات، وتشفير كلمة المرور باستخدام مكتبة bcrypt قبل حفظها في قاعدة البيانات.

5.1.2. تسجيل دخول المستخدم (Login)

الوصف: واجهة تسجيل دخول تتيح للمستخدم الوصول إلى حسابه.

الوظيفة: يتم التحقق من اسم المستخدم وكلمة المرور، وإنشاء رمز JWT لتأمين الجلسة بعد تسجيل الدخول بنجاح.

5.1.3. تصفح الافتتاحيات (Openings)

الوصف: واجهة تفاعلية تتيح للمستخدم استعراض مختلف افتتاحيات الشطرنج المتاحة.

الوظيفة: يمكن عرض معلومات كل افتتاحية بما في ذلك الاسم والوضعية على الرقعة.

5.1.4. لعب الشطرنج والمحادثة داخل اللعبة

الوصف: واجهة اللعبة التفاعلية تمكن المستخدم من لعب مباريات الشطرنج ضد لاعبين آخرين.

الوظيفة:

- دعم تحريك القطع والتحقق من قانونية الحركات.
- إمكانية إرسال واستقبال الرسائل النصية أثناء المباراة.
- تحديث لوحة اللعبة في الوقت الحقيقي باستخدام WebSocket.

5.1.5. عرض ملف المستخدم (My Profile)

الوصف: بعد تسجيل الدخول، يمكن للمستخدم الاطلاع على معلومات حسابه الشخصي.

الوظيفة: يتم استرجاع البيانات من قاعدة البيانات بعد التحقق من صحة الجلسة باستخدام JWT.

5.1.6. عرض المباريات السابقة للمستخدم

الوصف: عرض تفاصيل كل مباراة تم لعبها من قبل المستخدم.

الوظيفة: يمكن عرض اسم اللاعبين، الفائز، افتتاحية المباراة، وحركات اللعب.

5.1.7. تسجيل دخول المشرف (Admin Login)

الوصف: واجهة تسجيل دخول خاصة بالمشرفين لإدارة الموقع.

الوظيفة: بعد التحقق من بيانات المشرف، وإنشاء رمز JWT لتأمين الجلسة بعد تسجيل الدخول بنجاح، يمكنه الوصول إلى لوحة التحكم الخاصة بالإدارة.

5.1.8. عمليات الإدارة (CRUD)

الوصف: لوحة تحكم المشرف تمكنه من إدارة المحتوى والمستخدمين والمباريات.

الوظيفة: تنفيذ العمليات الأساسية:

- إنشاء (Create)
- قراءة (Read)
- حذف (Delete)

التحقق: تم اختبار جميع العمليات للتأكد من تنفيذها بشكل صحيح وآمن.

الخلاصة

- جميع الوظائف الأساسية للموقع تعمل بنجاح وفق التصميم المخطط.
- تم التحقق من أمان البيانات من خلال تشفير كلمات المرور واستخدام JWT للجلسات.
- يوفر الموقع تجربة تفاعلية للمستخدمين، مع إمكانية متابعة المباريات والتواصل الفوري أثناء اللعب، بالإضافة إلى إمكانيات الإدارة الشاملة للمشرفين.

5.2. الآفاق المستقبلية

على الرغم من اكتمال العديد من الوظائف الأساسية للموقع، إلا أن هناك عدة تحسينات وتوسعات مستقبلية يمكن تنفيذها لتعزيز تجربة المستخدم وزيادة كفاءة النظام، وتشمل ما يلي:

5.2.1. تحسين تجربة المستخدم (UX/UI)

- تطوير واجهات أكثر تفاعلية وجاذبية بصريًا.
- إضافة مؤثرات ديناميكية لتحريك القطع وعرض الرسائل في الوقت الحقيقي بطريقة سلسة.
- تحسين استجابة الموقع على الأجهزة المختلفة (Responsive Design).

5.2.2. دعم ميزات إضافية للعبة الشطرنج

- إضافة مستويات صعوبة مختلفة للعب محركات.
- تمكين اللاعبين من تحدي أصدقائهم عبر روابط مباشرة للعبة.
- اضافة انواع شطرنج مختلفة مثل chess960.
- اقامة بطولات دورية
- دعم نظام لإكتشاف الغشاشين

5.2.3. دعم ميزات تعليمية للموقع

- اضافة ألغاز متنوعة مناسبة لكل المستخدمين
- اضافة دروس، واتاحة امكانية للأساتذة بإضافات دورات مدفوعة

5.2.4. توسيع نظام المحادثة

- إضافة المحادثة الصوتية أو الفيديو أثناء اللعب.
- توفير سجلات المحادثة لكل مباراة لزيادة التفاعل الاجتماعي بين اللاعبين.
- إنشاء مجمتمع، يمكن للإعضاء الإنضمام لفرق

5.2.5. توسيع وظائف الإدارة

- إضافة إمكانية تعديل وحذف المباريات والافتتاحيات بشكل أسرع وأكثر ديناميكية.
- توفير لوحة تحكم متقدمة لإدارة الإحصائيات العامة للموقع.
- دعم إنشاء تقارير مفصلة للمشرفين حول أداء اللاعبين ونشاط الموقع.

5.2.6. التكامل مع خدمات أخرى

- إمكانية نشر المباريات والنتائج على وسائل التواصل الاجتماعي مباشرة.
- المشاركة في البطولات المحلية كرعاية أو إستضافة

5.2.7. تحقيق دخل مستدام

- الاشتراكات المدفوعة
- الإعلانات المدفوعة
- تنظيم بطولات مدفوعة
- الشراكات والرعايات
- المتجر الرقمي داخل الموقع

References

[1] W3C, "HTML5: A vocabulary and associated APIs for HTML and XHTML," W3C, 2014.

[2] W3C, "Cascading Style Sheets (CSS) Snapshot," W3C, 2018.

[3] D. Flanagan, JavaScript: The Definitive Guide, 7th ed., O'Reilly Media, 2020.

[4] ECMA International, "ECMAScript 2022 Language Specification," ECMA International, 2022.

[5] Tailwind Labs, "Tailwind CSS documentation," Tailwind Labs, 2023. [Online]. Available: https://tailwindcss.com/docs.

[6] L. Moroney, Learning Tailwind CSS: Rapidly build modern websites without ever leaving your HTML, O'Reilly Media, 2022.

[7] Oakmac, "chessboard.js: A JavaScript chessboard," [Online]. Available: https://github.com/oakmac/chessboardjs.

[8] jQuery, "jQuery API documentation," jQuery Foundation, 2023. [Online]. Available: https://api.jquery.com.

[9] Oakmac, "chess.js: A JavaScript chess library for chess move generation, validation, and utilities," [Online]. Available: https://github.com/jhlywa/chess.js.

[10] Fette, I., & Melnikov, A., "The WebSocket Protocol (RFC 6455)," Internet Engineering Task Force (IETF), 2011.

[11] Rauch, G., & Contributors, "Socket.IO Documentation," 2023. [Online]. Available: https://socket.io/docs.

[12] S. &. V. S. Tilkov, Node.js: Using JavaScript to Build High-Performance Network Programs, IEEE Internet Computing, 2010.

[13] Node.js.org, "The Node.js Event Loop," Node.js.org, [Online]. Available: https://nodejs.org/en/learn/asynchronous-work/event-loop-timers-and-nexttick.

[14] Kinsta, "What Is Express.js? Everything You Should Know.," 2023. [Online]. Available: https://kinsta.com/knowledgebase/what-is-express-js/.

[15] Mongoose , "Mongoose: Elegant MongoDB object modeling for Node.js," [Online]. Available: https://mongoosejs.com.

[16] N. Provos and D. Mazieres, "A Future-Adaptable Password Scheme," in *USENIX Annual Technical Conference*, 1999.

[17] Motdotla, "dotenv - Loads environment variables from .env file," [Online]. Available: https://github.com/motdotla/dotenv.

[18] Express.js Documentation, "cookie-parser middleware," 2025. [Online]. Available: https://expressjs.com/en/resources/middleware/cookie-parser.html.

[19] Mozilla Developer Network, "Cross-Origin Resource Sharing," 2025. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS.

[20] Validator.js Documentation, "String validation and sanitization," 2025. [Online]. Available: https://github.com/validatorjs/validator.js.

[21] npm, Inc, "About npm," 2024. [Online]. Available: https://docs.npmjs.com/about-npm.

[22] MongoDB Inc, "Introduction to MongoDB," 2024. [Online]. Available: https://www.mongodb.com/docs.

[23] Microsoft, "Visual Studio Code Documentation," 2024. [Online]. Available: https://code.visualstudio.com/docs.

[24] Nodemon Documentation, "Nodemon Introduction," https://nodemon.io. [Online].

[25] MongoDB Inc, "MongoDB Compass Documentation," 2024. [Online]. Available: https://www.mongodb.com/products/compass.

[26] Postman Inc, "Postman Learning Center," 2024. [Online]. Available: https://learning.postman.com.

[27] aamirfarookh, "ChessMate," [Online]. Available: https://github.com/aamirfarookh/Chessmate.

[28] Kshiti, "Chess-Game-Using-Socket.IO," [Online]. Available: https://github.com/Kshiti-24/Chess-Game-using-Socket-io.

[29] A. Yehia, "online-chess," 2025. [Online]. Available: https://github.com/agyad/online-chess.

[30] Wikipedia, "Forsyth–Edwards Notation," 9 August 2025. [Online]. Available: https://en.wikipedia.org/wiki/Forsyth_Edwards_Notation.

[31] Wikipedia, "Portable Game Notation," 7 May 2025. [Online]. Available: http://en.wikipedia.org/wiki/Portable_Game_Notation.