

Faculty of Engineering

Department of Informatics Engineering

Software Engineering and Information Systems



Integrated Social Media Platform (NOONIFY)

A junior project report - submitted to complete the requirements for obtaining a bachelor's degree in informatics engineering – Software Engineering and Information Systems

Prepared by

Sara Eyad Attah

Aya Al-salamat

Supervised by

Dr. Eng. Akram Massouh

منصة تواصل اجتماعي متكامل

تقرير مشروع فصلي قدم استكمالاً لمتطلبات الحصول على درجة
البكالوريوس في هندسة المعلوماتية - هندسة البرمجيات ونظم المعلومات

إعداد

ساره اياد عطا

آيه هيثم السلامات

إشراف

الدكتور. أكرم مسوح

SUPERVISION CERTIFICATION

I Certify that the preparation of this project entitled

[Integrated Social Media Platform (NOQNIFY)]

Prepared by

[Sara Eyad Atta - Aya Alsalamat]

was made under my supervision at Faculty of Informatics Engineering in partial Fulfillment of the
Requirements for the Degree of Bachelor of Software Engineering and Information System

Name:

Signature:

Date:

الملخص

يهدف هذا المشروع إلى تصميم منصة تواصل اجتماعي متكاملة تتاح للمستخدمين إنشاء حساباتهم، مشاركة المحتوى النصي والمرئي، والتفاعل مع الآخرين بطريقة آمنة وسلسة، يعتمد النظام على MERN Stack (MongoDB, Express.js, React.js, Node.js) لتغطية الجوانب الأمامية والخلفية، إضافة إلى أدوات Ingest Clerk لإدارة الحسابات وتوثيق المستخدمين، وImageKit لمعالجة وتحسين الصور، وIngest لـ إدارة المهام الخلفية المجدولة.

توفر المنصة مجموعة واسعة من الوظائف تشمل التسجيل والدخول (Sign Up / Sign In)، إدارة الملف الشخصي (Profile Management)، إنشاء المنشورات (Posts) والقصص (Stories)، التفاعل الاجتماعي عبر الإعجابات والتعليقات والمتابعة (Follow / Unfollow)، بالإضافة إلى نظام مراسلة فورية (Real-time Chat).

كما تتيح المنصة البحث والاكتشاف (Search & Discover)، الإشعارات اللحظية (Real-time Notifications)، وإدارة الوسائل باستخدام حلول تخزين سحابية فعالة.

من الناحية التقنية، يتميز النظام بالمرنة العالية (Scalability) بفضل بنية MongoDB وNode.js، وواجهة مستخدم تفاعلية وسهلة الاستخدام مبنية بـ React.js. كما تم الاهتمام بعوامل الأمان (Security) باستخدام تقنيات قوية مثل HTTPS، وضمان الاعتمادية (Reliability) وقابلية الصيانة (Maintainability) من خلال تصميم مكونات واضحة ومنفصلة.

بشكل عام، يقدم المشروع نموذجاً عملياً متكاملاً لتطبيق تواصل اجتماعي حديث، يربط بين تجربة المستخدم الممتعة والأداء القوي المدعوم بأحدث تقنيات Full Stack Development.

Abstract

This project aims to develop a comprehensive social media platform that enables users to create accounts, share text and media content, and interact with others in a secure and seamless environment. The system is built using the MERN Stack (MongoDB, Express.js, React.js, Node.js), along with supporting tools such as Clerk for authentication and account management, ImageKit for media optimization, and Inngest for background task scheduling.

The platform provides a wide range of functionalities including Sign Up / Sign In, Profile Management, creating and managing Posts and Stories, engaging with others through Likes, Comments, and Follow/Unfollow, and real-time messaging (Real-time Chat). It also includes Search & Discover, Real-time Notifications, and efficient Media Management using cloud storage services.

Technically, the system ensures Scalability with Node.js and MongoDB, offers a highly interactive user interface built with React.js, and maintains strong Security standards via HTTPS. It emphasizes Reliability and Maintainability through modular code structure and clear component separation.

Overall, the project presents a fully functional and modern social networking application that combines an enjoyable user experience with robust technical performance, showcasing practical Full Stack Development skills

Table of Contents

4.....	الملخص.....
5.....	Abstract
1.....	1. الفصل الأول: المقدمة
2	أهمية المشروع
3	الهدف من المشروع
4	الدراسات المرجعية لمنصة التواصل الاجتماعي
4	الدراسة الأولى
.5	الدراسة الثانية
6	جدول مقارنة بين الدراسات المرجعية
7	الجدول الزمني Gantt Chart
7	المنهجية المتبعة في تنفيذ المشروع (منهجية SCRUM)
8	تطبيق منهجية SCRUM في مشروع منصة التواصل الاجتماعي
9	المعمارية (System Architecture)
9	أولاً: معمارية Client–Server
10	مزايا معمارية Client–Server
10	سلبيات معمارية Client–Server
11	ثانياً: معمارية (Model – View – Controller)
133	إيجابيات معمارية MVC
133	سلبيات معمارية MVC
144	الربط بين MVC وClient–Server في المشروع
177	2. الفصل الثاني : الدراسة التحليلية
18	المتطلبات العامة (General Requirements)
199	المتطلبات الوظيفية (Functional Requirements)
19	نظام التسجيل والدخول (Authentication System)
199	إدارة الملف الشخصي (User Profile Management)
20	إدارة المنشورات (Posts Management)
20	إدارة القصص (Stories / Status System)
21	نظام المراسلة (Real-time Chat & Messages)

21	نظام العلاقات الاجتماعية (Connections System)	2.2.6
21	نظام البحث والاكتشاف (Search & Discover System)	2.2.7
22	نظام الإشعارات (Notifications System)	2.2.8
22	إدارة الوسائط (Media Management)	2.2.9
22	النظام الخلفي (Backend Operations)	2.2.10
23.....	المتطلبات غير الوظيفية(Non-Functional Requirements)	3.2
23.....	المتطلبات التي سوف تتفذ هذا الفصل	4.2
26	مخطط ال Use Case Diagram	5.2
267	تصنيف حالات الأستخدام Use Case Specifications	6.2
27	مخططات النشاط Activity Diagrams	7.2
26	مخططات التسلسل Sequence Diagrams	8.2
612.....	3. الفصل الثالث : الدراسة التصميمية	
63	مخطط قاعدة البيانات ERD	3.1
634	مخطط الصور Class Diagram	3.2
634	جدول Test Cases	3.3
65	جدول Requirements Traceability Matrix (RTM)	3.4
68	مخطط ال Site Map	3.5
689.....	4. الفصل الرابع : تنفيذ المشروع	
70	الأدوات المستخدمة	4.1
71	أدوات الواجهة الأمامية(Front-End Tools)	4.1.1
72	أدوات الواجهة الخلفية(Back-End Tools)	4.1.2
712	أدوات قاعدة البيانات(Database Tools)	4.1.3
73	ملخص الأدوات المستخدمة	4.1.4
73	الواجهات	4.2
789	الخاتمة	4.3
80	الآفاق المستقبلية(Future Work)	4.4

List of Tables

6	جدول 1 مقارنة بين الدراسات المرجعية
27	جدول 2 توصيف حالة تسجيل الحساب
28	جدول 3 توصيف حالة تسجيل الدخول
30	جدول 4 توصيف حالة الدخول الآمن
31	جدول 5 توصيف حالة تسجيل الخروج
33	جدول 6 توصيف حالة عرض الواجهة الأساسية
35	جدول 7 توصيف حالة التعديل على الحساب
378	جدول 8 توصيف حالة عرض معلومات الحسابات
40	جدول 9 توصيف حالة البحث عن الأصدقاء
42	جدول 10 توصيف حالة نشر منشور جديد
44	جدول 11 توصيف حالة عرض المنشورات
46	جدول 12 توصيف حالة الإعجاب بالمنشورات
48	جدول 13 توصيف حالة التعليق على المنشورات
49	جدول 14 توصيف حالة مشاركة المنشورات
51	جدول 15 توصيف حالة إضافة صديق
53	جدول 16 توصيف معرفة حالة طلب الصداقة
55	جدول 17 توصيف حالة إضافة قصة جديدة
57	جدول 18 توصيف عرض القصص المتاحة للمتابعين
59	جدول 19 توصيف عرض قائمة المشاهدين
654	جدول 20 الـ Test Cases
66	جدول 21 الـ RTM

List of Figures

16	1 مخطط grant char
8	2 مخطط scrum methodology
11	3 مخطط client_server
14	4 مخطط mvc
16	5 مخطط دمج المعماريات
25	6 مخطط Class Diagram
27	7 مخطط النشاط حالة تسجيل الحساب
27	8 مخطط التسلسل حالة تسجيل الحساب
30	9 مخطط النشاط حالة تسجيل الدخول

29	10 مخطط التسلسل حالة تسجيل الدخول
32	11 مخطط التسلسل حالة تسجيل الخروج
32	12 مخطط النشاط حالة تسجيل الخروج
33	13 مخطط التسلسل حالة عرض الواجهة الأساسية
34	14 مخطط النشاط حالة عرض الواجهة الأساسية
36	15 مخطط التسلسل حالة التعديل على الحساب
36	16 مخطط النشاط حالة التعديل على الحساب
38	17 مخطط النشاط حالة عرض معلومات الحسابات
38	18 مخطط النشاط حالة عرض معلومات الحسابات
40	19 مخطط النشاط حالة البحث عن الأصدقاء
41	20 مخطط التسلسل حالة البحث عن الأصدقاء
42	21 مخطط التسلسل حالة نشر منشور جديد
43	22 مخطط النشاط حالة نشر منشور جديد
44	23 مخطط النشاط حالة عرض المنشورات
45	24 مخطط التسلسل حالة عرض المنشورات
46	25 مخطط التسلسل حالة الإعجاب بالمنشورات
47	26 مخطط النشاط حالة الإعجاب بالمنشورات
48	27 مخطط النشاط حالة التعليق على المنشورات
48	28 مخطط التسلسل حالة التعليق على المنشورات
50	29 مخطط النشاط حالة مشاركة المنشورات
50	30 مخطط التسلسل حالة مشاركة المنشورات
51	31 مخطط التسلسل حالة إضافة صديق
53	32 مخطط النشاط حالة إضافة صديق
53	33 مخطط التسلسل معرفة حالة طلب الصداقة
54	34 مخطط النشاط معرفة حالة طلب الصداقة
55	35 مخطط التسلسل حالة إضافة قصة جديدة
56	36 مخطط النشاط حالة إضافة قصة جديدة
58	37 مخطط النشاط حالة عرض القصص المتاحة للمتابعين
58	38 مخطط التسلسل حالة عرض القصص المتاحة للمتابعين
61	39 مخطط النشاط عرض قائمة المشاهدين
60	40 مخطط النشاط عرض قائمة المشاهدين
62	41 مخطط ال ERD
63	42 مخطط ال Class Diagram
67	43 مخطط ال Site Map
72	44 الشكل (1.1) واجهة تسجيل حساب جديد
73	45 الشكل (1.2) واجهة تسجيل الدخول
75	46 الشكل (1.3) زر تسجيل الخروج
74	47 الشكل (1.4) عرض المعلومات الأساسية للمستخدم
76	48 الشكل (1.5) رفع قصة جديدة
75	49 الشكل (1.6) إنشاء منشورات جديد
77	50 الشكل (1.7) عرض ملفات المستخدمين الآخرين
76	51 الشكل (1.8) تعديل الملف الشخصي
78	52 الشكل (1.9) عرض القصص

78 الشكل (1.10) عرض منشورات الآخرين 53

1. الفصل الأول: المقدمة

1.1 أهمية المشروع

تبعد أهمية هذا المشروع من الحاجة المتزايدة إلى منصات تواصل اجتماعي حديثة وآمنة تتيح للمستخدمين التفاعل ومشاركة المحتوى بسهولة وكفاءة، ومع التطور السريع في تقنيات الويب وازدياد الاعتماد على التطبيقات الرقمية في التواصل اليومي، أصبح من الضروري تطوير نظام تواصل اجتماعي متكامل يجمع بين الأداء العالي وتجربة المستخدم المميزة.

يساهم المشروع في تعزيز مهارات المطوريين في مجال Full Stack Development من خلال تطبيق عملي يعتمد على تقنيات MERN Stack (MongoDB, Express.js, React.js, Node.js)، والتي تمكّن من بناء نظام متكامل يغطي جميع الجوانب: Front-End لتصميم واجهة تفاعلية، و Back-End لمعالجة الطلبات وإدارة البيانات، و Database لتخزين المعلومات بطريقة فعالة وآمنة.

تكمّن أهمية المشروع أيضًا في كونه يدمج مفاهيم وتقنيات حديثة مثل Real-time Communication، وإدارة الوسائط باستخدام ImageKit، ونظام توثيق المستخدمين الآمن بواسطة Clerk، بالإضافة إلى إدارة المهام الخلفية باستخدام Inngest.

وبذلك، لا يقتصر المشروع على كونه تجربة تعليمية فحسب، بل يشكل نموذجًا متكاملاً يمكن تطويره مستقبلاً ليصبح منصة تواصل اجتماعي حقيقية تتميز بالأمان، والاعتمادية، وسهولة الاستخدام، وتواكب التطورات التقنية الحديثة في عالم الويب.

1.2 الهدف من المشروع

يهدف هذا المشروع إلى تطوير منصة تواصل اجتماعي متكاملة تتيح للمستخدمين إنشاء حساباتهم، إدارة ملفاتهم الشخصية، ومشاركة المحتوى النصي والمرئي بطريقة آمنة وسلسة. كما يسعى إلى توفير بيئة تفاعلية تجمع بين التصميم الجذاب، الأداء العالي، وسهولة الاستخدام عبر مختلف الأجهزة والمنصات.

يتمثل الهدف الأساسي في بناء تطبيق ويب يعتمد على MERN Stack (MongoDB, Express.js,

React.js, Node.js) لتغطية جميع طبقات النظام من الواجهة الأمامية (Front-End) وحتى النظام

الخلفي (Back-End) وقاعدة البيانات (Database)، كما يهدف المشروع إلى تعزيز الجوانب التقنية من خلال دمج أدوات وخدمات حديثة مثل Clerk لإدارة المستخدمين وتوثيق الحسابات، وImageKit لمعالجة الصور وتحسين أدائها، وIngest لتنفيذ المهام الخلفية المجدولة مثل الإشعارات أو حذف البيانات المؤقتة.

ومن الأهداف المهمة كذلك، تطبيق مفاهيم Real-time Features مثل المراسلة الفورية (Real-time Notifications) وإشعارات اللحظية (Real-time Notifications) لتوفير تجربة استخدام ديناميكية وتفاعلية.

إضافة إلى ذلك، يسعى المشروع إلى بناء واجهة استخدام متجاذبة (UI Responsive) باستخدام

React.js و CSS تتيح سهولة التنقل وتفاعل المستخدمين بسلامة.

باختصار، يهدف المشروع إلى تقديم نموذج تطبيقي متكامل يربط بين الجانب الأكاديمي والعملي، ويمكن المطورين من تطبيق مهاراتهم في بناء Social Media Platform احترافية تُبرز المفاهيم الأساسية لتقنيات الويب الحديثة وأفضل ممارسات التطوير المتكامل.

1.3 الدراسات المرجعية لمنصة التواصل الاجتماعي

1.3.1 الدراسة الأولى

بنية بيانات وتخزين الشبكة الاجتماعية

"TAO: Facebook's Distributed Data Store for the Social Graph"

تتناول هذه الدراسة النظام TAO الذي طورته شركة Facebook لإدارة العلاقات بين المستخدمين "TAO: Facebook's Distributed Data Store for the Social Graph" بفاءة عالية، يهدف النظام إلى تحسين سرعة الاستعلام والوصول إلى البيانات من خلال Data Store مخصص ومحسن للقراءة، مع واجهة API موحدة تسهل عمليات الاستعلام دون الحاجة لفهم البنية الداخلية للبيانات ويعتبر TAO نموذجاً لتصميم قاعدة بيانات مخصصة للتطبيقات الاجتماعية ذات الحجم الكبير، حيث يجمع بين MySQL و Caching Layer لتقليل زمن الاستجابة وتحسين الأداء.

أبرز الدروس المستفادة:

- تصميم طبقة بيانات متخصصة لتناسب طبيعة التطبيقات الاجتماعية.
- استخدام API Layer و Caching Layer يقلل من الحمل على قواعد البيانات.
- التوزيع الجغرافي للبيانات يحسن من سرعة الوصول للمستخدمين حول العالم.

.Nathan Bronson et al., USENIX ATC, 2013 – Facebook Research : المصدر :

1.3.2 الدراسة الثانية

بنية الأنظمة وتحجيم منصات التراسل/التواصل

"Scaling big data mining infrastructure: the Twitter experience"

تستعرض هذه الدراسة تجربة Twitter في بناء بنية تحتية قادرة على التعامل مع ميلارات الرسائل يومياً مع الحفاظ على الأداء اللحظي (Real-time). استخدمت Twitter تقنيات مثل Kafka, HBase و Distributed Storage Systems لمعالجة التدفق المستمر من البيانات وتحليلها بسرعة.

ترکّز التجربة على أهمية بناء نظام مرن (Scalable) يسمح بتحديث المكونات دون توقف الخدمة، وتحقيق توازن بين Latency و Throughput لتقديم تجربة تفاعلية مستقرة.

أبرز الدروس المستفادة:

- اعتماد تقنيات Kafka و Message Queue Streaming ضروري للميزات اللحظية.
- بناء بنية مرن قابلة للتوسيع والصيانة دون انقطاع.
- أهمية أدوات المراقبة (Monitoring) والـ Logging في تحليل الأداء.

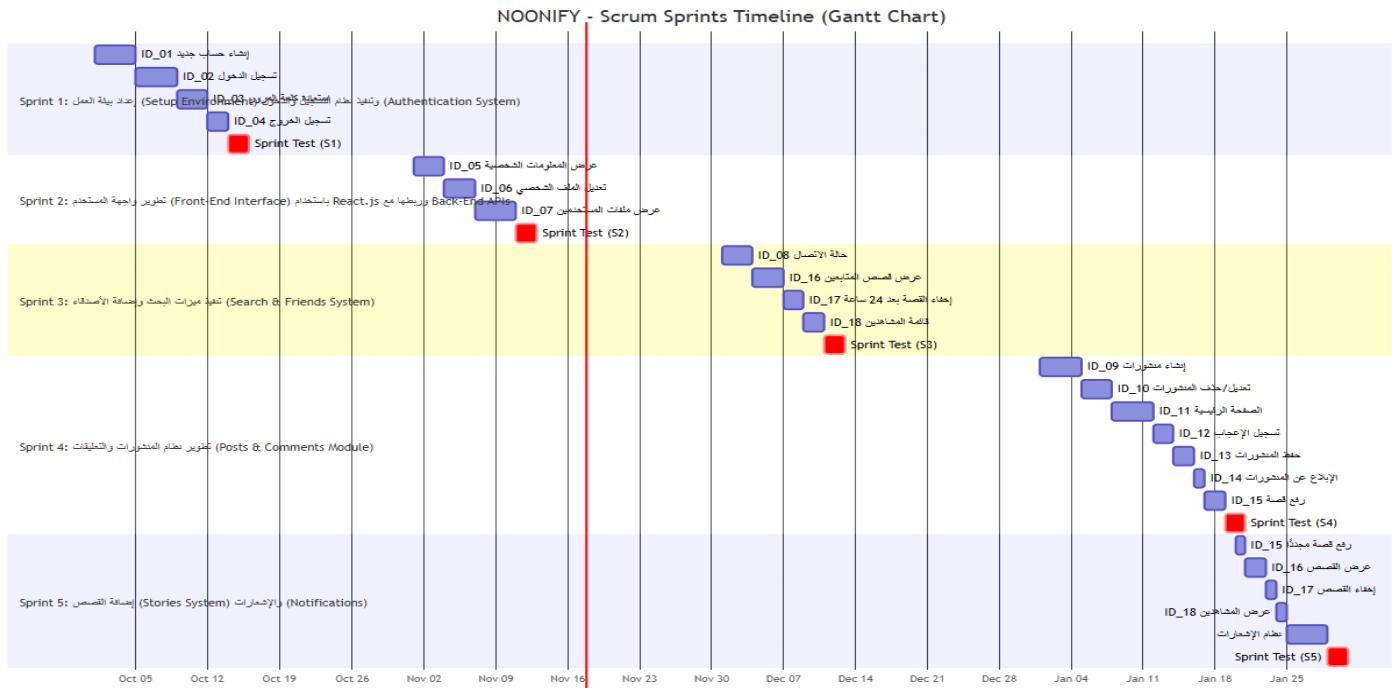
المصدر: .Twitter Engineering Reports, Scaling Big Data Systems

1.3.3 جدول مقارنة بين الدراسات المرجعية.

الدراسة	TAO-Facebook	Twitter	مشروعنا NOONIFY
التقنيات المستخدمة	Data Store Cashing Layer,API	Kafka,HBase,Distributed Storage.	MERN Stack
الإيجابيات	أداء عالي واستجابة سريعة للبيانات	معالجة بيانات ضخمة في الزمن الحقيقي	واجهات حديثة ، أداء فعال، تكامل مكونات جاهزة.
السلبيات	تعقيد البنية وصعوبة الصيانة	تكلفة تشغيل عالية وتعقيد إداري	اعتماد جزئي على خدمات خارجية.
التحديات	الحفاظ على اتساق البيانات الموزعة	التوازن بين السرعة والتوفير	ضمان الأمان والخصوصية مع الأداء العالي.
أوجه التميز	تخصيص النظام لمعالجة العلاقات الاجتماعية	بنية مرنة تدعم التحليل اللحظي	دمج تقنيات حديثة بمرونة عالية وسرعة تطوير .

جدول 1 مقارنة بين الدراسات المرجعية

1.4 الجدول الزمني Gantt Chart



1 مخطط الجدول الزمني Gantt Chart

1.5 المنهجية المتبعة في تنفيذ المشروع (منهجية SCRUM)

تم اعتماد منهجية SCRUM كإطار عمل (Agile Framework) لتنفيذ هذا المشروع، نظراً لمرونته العالية وقدرتها على تنظيم عملية التطوير بطريقة تفاعلية وتدريجية (Iterative and Incremental)

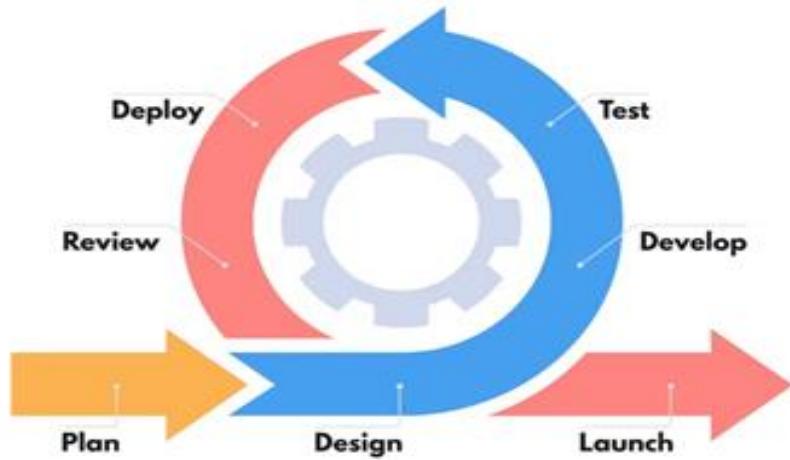
العالية وقدرتها على تنظيم عملية التطوير بطريقة تفاعلية وتدريجية (Iterative and Incremental) ، تعد منهجية SCRUM واحدة من أكثر منهجيات Agile Software Development Process

استخداماً في المشاريع البرمجية الحديثة، حيث ترتكز على تقسيم العمل إلى مراحل قصيرة تُعرف باسم

Sprints، يتم خلالها تنفيذ مجموعة محددة من المهام ذات الأولوية العالية، ومراجعةها بشكل مستمر

لتحسين جودة المنتج النهائي.

SCRUM METHODOLOGY



2 مخطط ال scrum methodology

1.5.1 تطبيق منهجية SCRUM في مشروع منصة التواصل الاجتماعي

تم تقسيم عملية التطوير إلى عدة Sprints، وكل Sprint يرتكز على مجموعة محددة من الوظائف الأساسية ضمن المنصة.

والوظائف هي:

Authentication: إعداد بيئة العمل (Setup Environment) وتنفيذ نظام التسجيل والدخول (Sprint 1 System).

Back-End APIs) وربطها مع React.js (Front-End Interface) باستخدام Sprint 2

.End APIs

.(Search & Friends System) تطوير واجهة المستخدم (Sprint 3

.(Posts & Comments Module) تطوير نظام المنشورات والتعليقات (Sprint 4

.(Notifications) Stories System) وإشعارات إضافة القصص (Sprint 5

بعد كل Sprint، يتم مراجعة Sprint Review Meeting (لعرض التقدم وتقييم ما تم إنجازه، يعقبه

اجتماع تخطيطي Sprint Planning Meeting (لتحديد مهام الدورة التالية).

1.6 المعمارية (System Architecture)

1.6.1 أولاً: معمارية Client-Server

تم اعتماد معمارية Client-Server كأساس لبناء المنصة، وهي من أكثر المعماريات استخداماً في تطبيقات الويب الحديثة، خصوصاً تلك التي تعتمد على الاتصال بين واجهة المستخدم (Client) والخادم (Server) عبر بروتوكول HTTP/HTTPS.

في هذا المشروع، يمثل الـ Client واجهة المستخدم المبنية باستخدام React.js، بينما يمثل الـ Server الخادم الخلفي المطور باستخدام Node.js Express.js و MongoDB. يتم تبادل البيانات بين الطرفين بصيغة JSON عبر واجهات RESTful APIs.

آلية العمل:

1. يقوم المستخدم (Client) بإرسال طلب (Request) عبر المتصفح مثل تسجيل الدخول أو عرض المنشورات.
2. يستقبل الخادم (Server) الطلب من خلال Express.js Controllers ويعالجه وفقاً للمنطق البرمجي.
3. يتصل الخادم بقاعدة البيانات MongoDB لاسترجاع أو تعديل البيانات المطلوبة.
4. يعيد الخادم استجابة (Response) إلى Client، تُعرض النتيجة عبر الواجهة باستخدام React.js.

:Client–Server مزايا معمارية

1.6.1.1

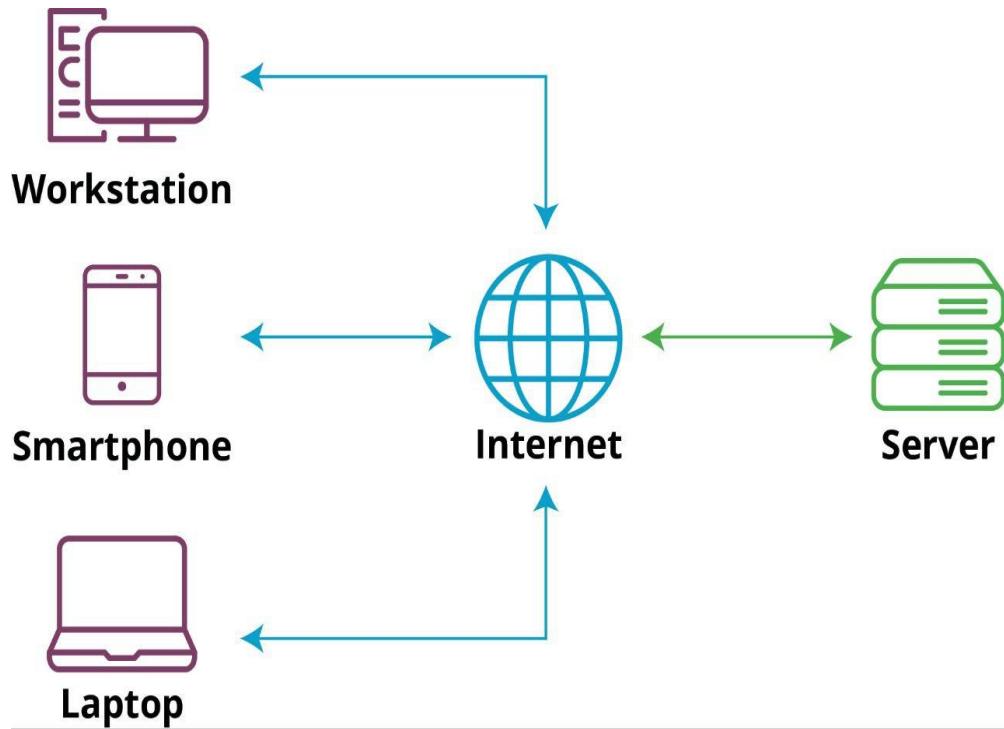
- فصل واضح بين الواجهة والتطبيق الخلفي (Separation of Concerns) مما يسهل التطوير والتحديث.
- أمان أعلى لأن معالجة البيانات الأساسية تتم في الخادم وليس في واجهة المستخدم.
- قابلية التوسيع (Scalability) بإمكانية توزيع الخوادم أو تحديث الواجهة بشكل مستقل.
- إعادة الاستخدام (Reusability) للخدمات الخلفية عبر واجهات API يمكن استدعاها من تطبيقات أخرى مستقبلاً (مثل تطبيق موبايل).

:Client–Server سلبيات معمارية

1.6.1.2

- اعتماد الواجهة على توفر الخادم بشكل دائم، مما يجعل انقطاعه يوقف النظام.

- زيادة الحمل على الخادم مع زيادة عدد المستخدمين، مما يتطلب تحسين الأداء أو توزيع الأحمال.
- الحاجة إلى إدارة أمان الاتصال بشكل دقيق لتجنب الثغرات.



client_server 2 مخطط

1.6.2 ثانياً: معمارية **MVC (Model – View – Controller)**

بالإضافة إلى معمارية Client–Server، سيتم تطبيق نمط MVC داخل الخادم (Server–Side) لتنظيم الكود وتحسين قابلية الصيانة والتوسع.

تقوم معمارية MVC على تقسيم النظام إلى ثلاث طبقات رئيسية:

1. Model (النموذج): تمثل البيانات والهيكل المرتبطة بها، وتشمل جميع العمليات الخاصة بالتخزين والاسترجاع من قاعدة البيانات.

2. View (العرض): تمثل واجهات المستخدم التي تُعرض عبر React.js في الواجهة الأمامية.

3. Controller (المتحكم): تدير التفاعل بين Model وView، وتتولى استقبال الطلبات من المستخدم وتنفيذ المطلوب اللازم ثم إرسال النتائج إلى الواجهة.

تطبيق MVC في هذا المشروع

:Controllers المستخدمة

- MessageControl: مسؤول عن إدارة الرسائل والمحادثات.
- PostControl: مسؤول عن إنشاء، تعديل، وحذف المنشورات.
- StoryControl: مسؤول عن رفع القصص (Stories) وإدارتها.
- UserControl: مسؤول عن تسجيل المستخدمين وتحديث معلوماتهم.

:Models المستخدمة

Email, Message, Post, Story, User, Connection •

- تمثل كل منها الجداول أو المجموعات (Collections) في قاعدة البيانات MongoDB، وتُعرف الخصائص (Attributes) وال العلاقات فيما بينها.

:View

• واجهات المستخدم المبنية باستخدام React.js و CSS، والتي تتلقى البيانات من الخادم عبر

RESTful APIs وتعرضها للمستخدم بطريقة تفاعلية وسهلة الاستخدام.

1.6.2.1 إيجابيات معمارية MVC

1. تنظيم الكود (Code Organization): تقسيم الكود إلى مكونات منفصلة يسهل قراءتها وصيانتها.

2. سهولة التطوير الجماعي: يمكن لفريق الواجهة الأمامية (Front-End) وفريق الخادم (Back-End) العمل بالتوافق دون تعارض.

3. إعادة الاستخدام (Reusability): يمكن إعادة استخدام الموديلات (Models) في خدمات أو مشاريع أخرى.

4. قابلية الاختبار (Testability): سهولة اختبار كل طبقة على حدة (Unit Testing).

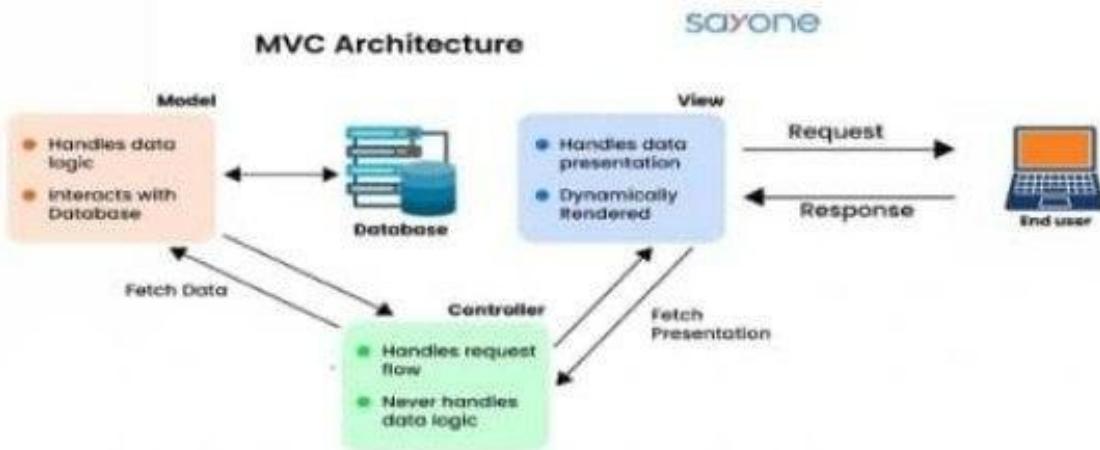
1. مرنة التطوير: يسهل تحديث أي جزء من النظام دون التأثير على الأجزاء الأخرى.

1.6.2.2 سلبيات معمارية MVC

1. زيادة التعقيد (Complexity): في المشاريع الصغيرة قد تبدو المعمارية معقدة أكثر من اللازم.

2. تعدد الطبقات (Multiple Layers): يسبب زيادة في عدد الملفات والاعتمادات البرمجية.

3. صعوبة تتبع الأخطاء أحياناً: لأن الطلب ينتقل بين أكثر من طبقة قبل الوصول للنتيجة النهائية.



3 مخطط mvc

الربط بين Client–Server في المشروع و MVC

1.6.2.3

يعتبر النظام في هذا المشروع مزيجاً من المعماريين:

من الخارج، النظام قائم على Client–Server Architecture، حيث يتفاعل المستخدم عبر الواجهة مع الخادم (Client) عبر الإنترنت. ومن الداخل، يدار الخادم وفق MVC Pattern (Server) الذي ينظم سير البيانات داخل النظام بين View – Controller – Model، مما يجعل النظام قابلاً للتوسيع وسهل الصيانة.

Controller Layer: تستقبل الطلبات وتتقىد المنطق المطلوب من خلال وحدات مثل MessageControl و StoryControl و PostControl و UserControl.

Model Layer: تتعامل مع البيانات الممثلة في نماذج مثل User و Post و Story و Message .MongoDB و Connection و Email، وتتصل بقاعدة البيانات .

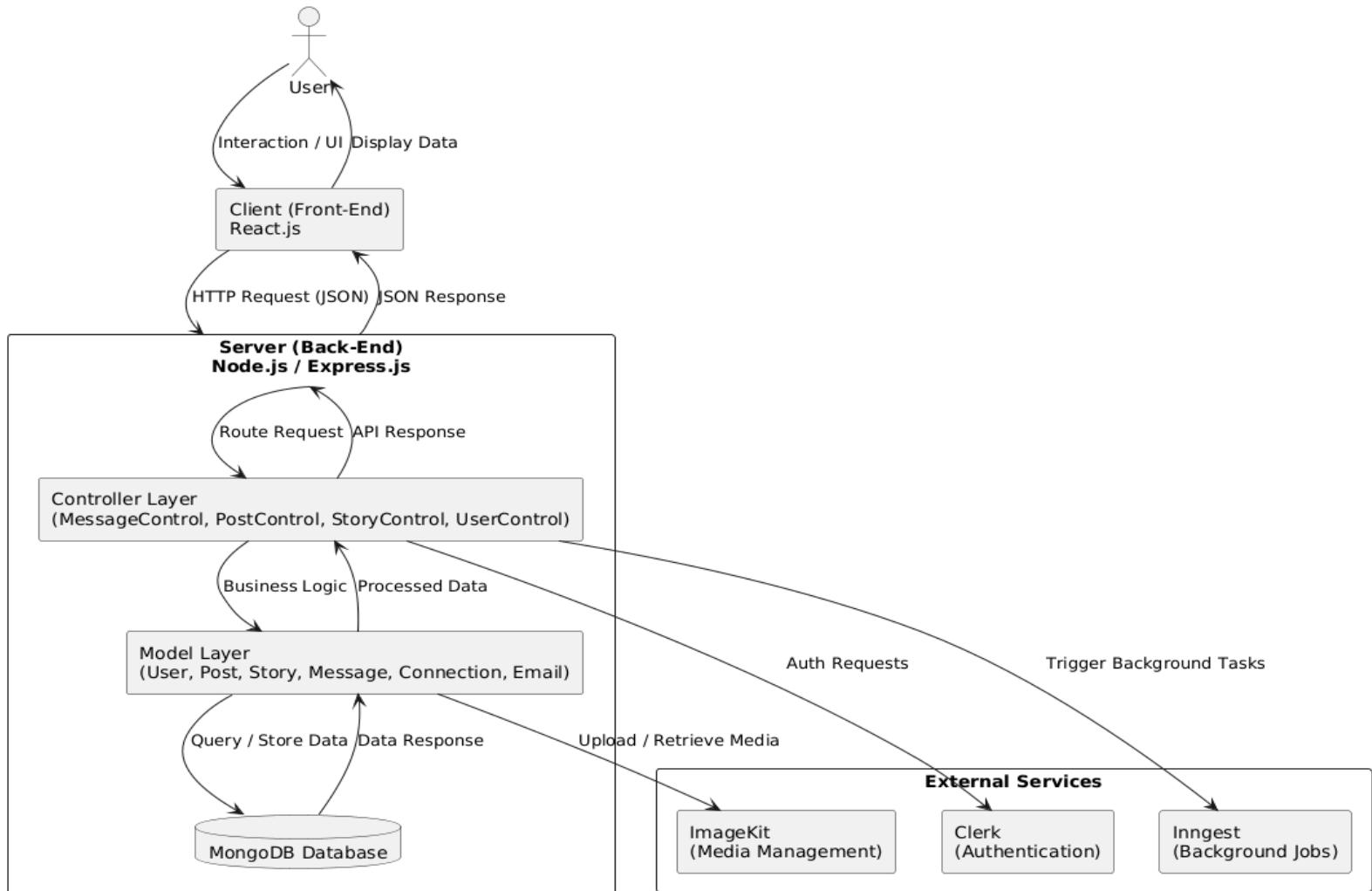
View Layer: تُرسل البيانات النهائية إلى الواجهة الأمامية ليتم عرضها للمستخدم.

يرتبط النظام أيضًا بخدمات خارجية مثل Clerk للتوثيق، و ImageKit لإدارة الوسائط، و Ingest للمهام الخلفية. تسير البيانات في تسلسل واضح:

المستخدم → الواجهة → الخادم →Controllers و Models () قاعدة البيانات → ثم تعود النتائج إلى المستخدم عبر الواجهة.

بذلك تحقق المنصة توازنًا مثالياً بين التوزيع الشبكي (Networked Architecture) وتنظيم المكونات الداخلية (Software Structure)، ما يؤدي إلى نظام قوي، آمن، وسهل التطوير المستقبلي.

Architecture Diagram - Client Server + MVC Integration



٤ مخطط دمج المعماريين

الفصل الثاني : الدراسة التحليلية

2.1 المتطلبات العامة (General Requirements)

تمثل المتطلبات العامة الإطار العام للمشروع، وتشمل الأهداف والقيود والعناصر الأساسية الواجب تحقيقها لضمان بناء منصة تواصل اجتماعي متكاملة. وتنتمي فيما يلي:

1. تطوير نظام تواصل اجتماعي متكامل (Integrated Social Media Platform) يسمح للمستخدمين بالتفاعل عبر المنشورات، الرسائل، والقصص.
2. استخدام تقنيات MERN Stack (MongoDB, Express.js, React.js, Node.js) لبناء واجهات أمامية وخلفية متزامنة.
3. ضمان تجربة مستخدم سلسة (User Experience – UX) وواجهة تفاعلية جذابة (User Interface – UI) باستخدام React.js و CSS.
4. تحقيق الأمان العالي (Security) في توثيق المستخدمين وإدارة الجلسات باستخدام Clerk، HTTPS، و API.
5. دعم الميزات اللحظية (Real-time Features) مثل الدردشة الفورية (Chat) والإشعارات (Notifications) باستخدام Socket.io.
6. تخزين الوسائط (Media Storage) ومعالجتها عبر ImageKit لضمان السرعة وجودة الصور.

7. إدارة المهام الخلفية (Background Jobs) عبر Ingest لتشغيل العمليات المجدولة مثل حذف القصص المنتهية أو إرسال التنبية.
8. نشر النظام عبر الإنترن特 (Deployment) ليكون متاحاً للمستخدمين باستخدام خوادم سحابية (Cloud Hosting). والأنظمة.

2.2 المتطلبات الوظيفية (Functional Requirements)

2.2.1 نظام التسجيل والدخول (Authentication System)

- إمكانية إنشاء حساب جديد (Sign Up) باستخدام البريد الإلكتروني أو رقم الهاتف.
- تسجيل الدخول (Sign In) باستخدام بيانات الاعتماد أو عبر Social Login (Google / GitHub).
- تفعيل خاصية إعادة تعيين كلمة المرور (Reset Password) عند نسيانها.
- تسجيل الخروج من الجلسة الحالية (Log Out).

2.2.2 إدارة الملف الشخصي (User Profile Management)

1. عرض المعلومات الأساسية للمستخدم (الاسم، الصورة، Bio، عدد المتابعين).
2. تعديل الملف الشخصي (Edit Profile) وتحديث البيانات.
3. عرض ملفات المستخدمين الآخرين (View Other Profiles).
4. عرض حالة الاتصال (Online / Offline Status).
5. حماية الدخول وإدارة الجلسات باستخدام Clerk Authentication Tokens.

2.2.3 إدارة المنشورات (Posts Management)

1. إنشاء منشورات جديدة (Create Post) نصية أو تحتوي على صور.
2. تعديل أو حذف المنشورات (Edit / Delete Post).
3. عرض المنشورات في الصفحة الرئيسية (Feed) بترتيب زمني.
4. التفاعل مع المنشورات عبر الإعجابات (Like) والتعليقات (Comment).
5. حفظ المنشورات المفضلة (Save Post) أو مشاركتها (Share Post).
6. الإبلاغ عن المنشورات المخالفة (Report Post).

2.2.4 إدارة القصص (Stories / Status System)

1. رفع قصة جديدة (Upload Story) بصيغة صورة أو فيديو قصير.
2. عرض القصص الخاصة بالمستخدمين الذين يتبعهم (View Stories).

3. حذف القصة يدوياً أو اختقاؤها تلقائياً بعد 24 ساعة.

4. عرض قائمة المشاهدين (Viewers List).

2.2.5 نظام المراسلة (Real-time Chat & Messages)

1. بدء محادثة جديدة (Start Conversation) بين المستخدمين.

2. إرسال واستقبال الرسائل النصية والصورية والرموز التعبيرية (Emojis).

3. دعم المراسلة اللحظية (Real-time Chat) باستخدام Socket.io.

4. عرض حالة الرسائل (مرسلة - مستلمة - مقرؤة).

5. حذف أو كتم المحادثة (Mute / Delete Chat).

2.2.6 نظام العلاقات الاجتماعية (Connections System)

1. متابعة أو إلغاء متابعة المستخدمين (Follow / Unfollow).

2. إرسال واستقبال طلبات الصداقة (Friend Requests).

3. عرض قوائم المتابعين والمتابعين (Followers / Following Lists).

4. اقتراح أصدقاء بناءً على الاهتمامات المشتركة (Friend Suggestions).

5. حظر أو إلغاء حظر المستخدمين (Block / Unblock).

2.2.7 نظام البحث والاكتشاف (Search & Discover System)

1. البحث عن المستخدمين بالاسم أو اسم المستخدم (Username Search).

2. البحث عن المستخدمين من خلال الموقع.

3. البحث عن المنشورات باستخدام الوسوم (Hashtags).
4. عرض صفحة اكتشاف (Discover Page) تحتوي على المنشورات الأكثر تفاعلاً والمستخدمين المقترنين.

2.2.8 نظام الإشعارات (Notifications System)

1. إرسال إشعارات فورية (Real-time Notifications) عند التفاعل مع المنشورات أو المتابعة الجديدة.
2. عرض جميع الإشعارات في صفحة مخصصة (Notifications Page).
3. إمكانية حذف الإشعارات أو وضع علامة "تمت القراءة" (Mark as Read).

2.2.9 إدارة الوسائط (Media Management)

2. رفع الصور عبر ImageKit وتحسينها تلقائياً.
3. دعم صيغ الصور المختلفة (JPEG, PNG, WEBP).
4. ضغط الصور لتقليل الحجم دون فقدان الجودة.
5. حماية روابط الوسائط من الوصول غير المصرح به (Access Control).

2.2.10 النظام الخلفي (Backend Operations)

1. إنشاء واجهات للتفاعل مع المستخدمين والمنشورات والرسائل.
2. تنفيذ المهام المجدولة عبر Ingest مثل حذف القصص المنتهية.
3. تسجيل الأخطاء والنشاطات (Error Logging & Monitoring).

2.3 المتطلبات غير الوظيفية (Non-Functional Requirements)

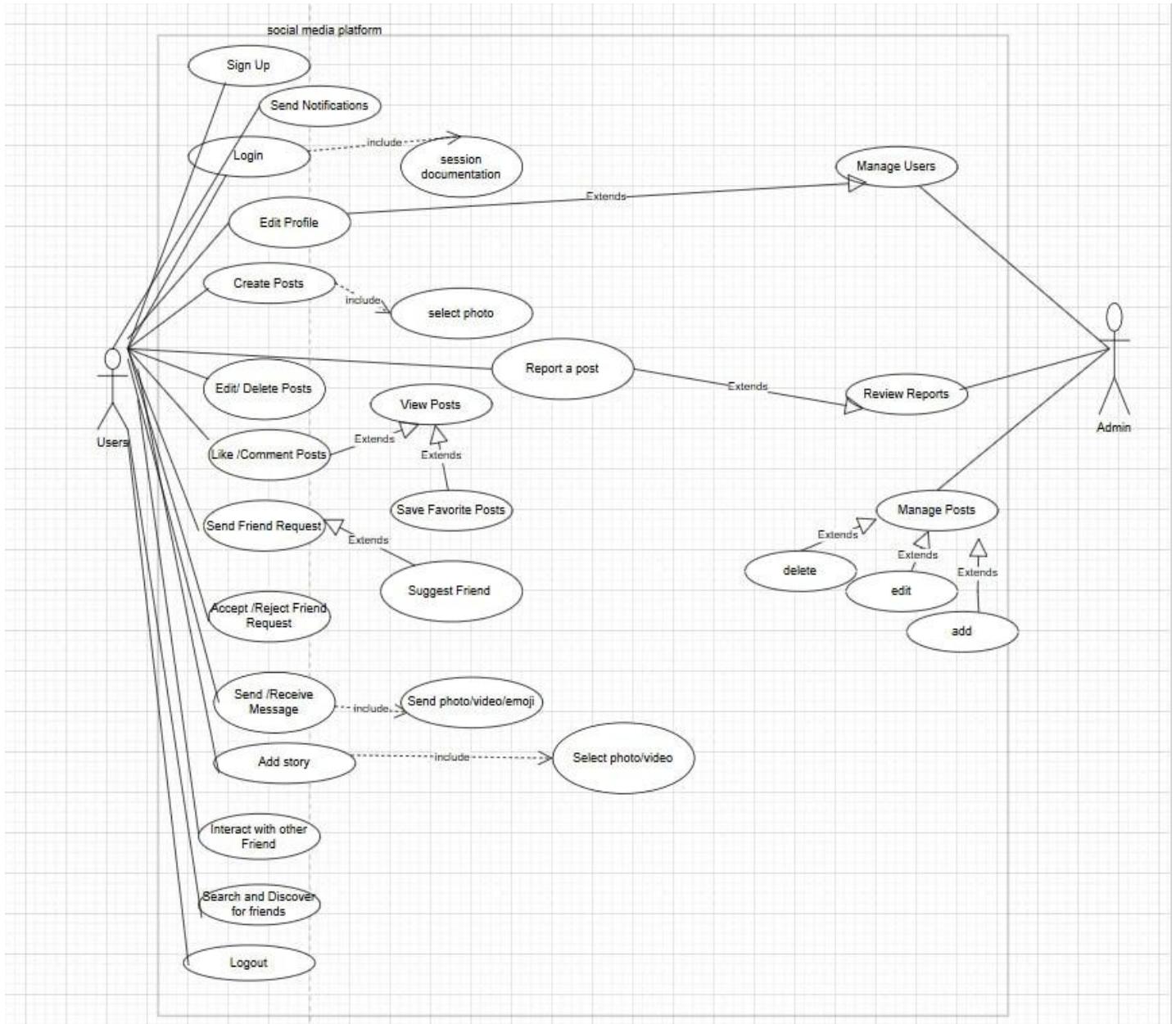
- .1: استخدام Clerk للتحقق، و HTTPS لتشифير البيانات.
- .2: سرعة تحميل عالية، استجابة فورية، Performance .Real-time Chat ، Caching
- .3: دعم التوسعة الأفقية باستخدام Scalability .MongoDB و Node.js
- .4: واجهة تفاعلية، تصميم متواكب (Responsive Design) Usability .
- .5: تشغيل مستقر، مراقبة النظام Reliability .(Monitoring & Logging)
- .6: كود منظم (Modular Code)، توثيق جيد. Maintainability .
- .7: دعم جميع المتصفحات والأجهزة (Cross-Platform) Compatibility .
- .8: حماية بيانات المستخدمين، إعدادات خصوصية مرنة.

2.4 المتطلبات التي سيتم تنفيذها في المشروع الفصلي

1. إمكانية إنشاء حساب جديد (Sign Up) باستخدام البريد الإلكتروني أو رقم الهاتف.
2. تسجيل الدخول (Sign In) باستخدام بيانات الاعتماد أو عبر Social Login (Google / GitHub).
3. التحقق من تسجيل الدخول من خلال رمز التحقق.
4. تسجيل الخروج من الجلسة الحالية (Log Out).

- .5. عرض المعلومات الأساسية للمستخدم (الاسم، الصورة، Bio، عدد المتابعين).
- .6. تعديل الملف الشخصي (Edit Profile) وتحديث البيانات.
- .7. عرض ملفات المستخدمين الآخرين (View Other Profiles)
- .8. البحث عن المستخدم من خلال الاسم او البايو او الموقع .
- .9. إنشاء منشورات جديدة (Create Post) نصية أو تحتوي على صور.
- .10. تعديل أو حذف المنشورات (Edit / Delete Post)
- .11. عرض المنشورات في الصفحة الرئيسية (Feed) بترتيب زمني.
- .12. التفاعل مع المنشورات عبر الإعجابات (Like)
- .13. التفاعل مع المنشورات عبر التعليقات (Comment)
- .14. مشاركة المنشورات (Share Post)
- .15. عرض حالة الطلب .
- .16. رفع قصة جديدة (Upload Story) بصيغة صورة أو فيديو قصير.
- .17. عرض القصص الخاصة بالمستخدمين الذين يتبعهم (View Stories)
- .18. عرض قائمة المشاهدين (Viewers List)
- .19. استخدام Clerk للتحقق، و HTTPS لتشفيير البيانات.
- .20. واجهة تفاعلية، تصميم متواكب (Responsive Design): Usability
- .21. دعم جميع المتصفحات والأجهزة (Cross-Platform): Compatibility

Use Case Diagram 2.5 مخطط الـ



Class Diagram مخطط الـ 6

2.6 توصيف حالات الأستخدام Use Case Specifications

2.7 مخططات النشاط Activity Diagrams

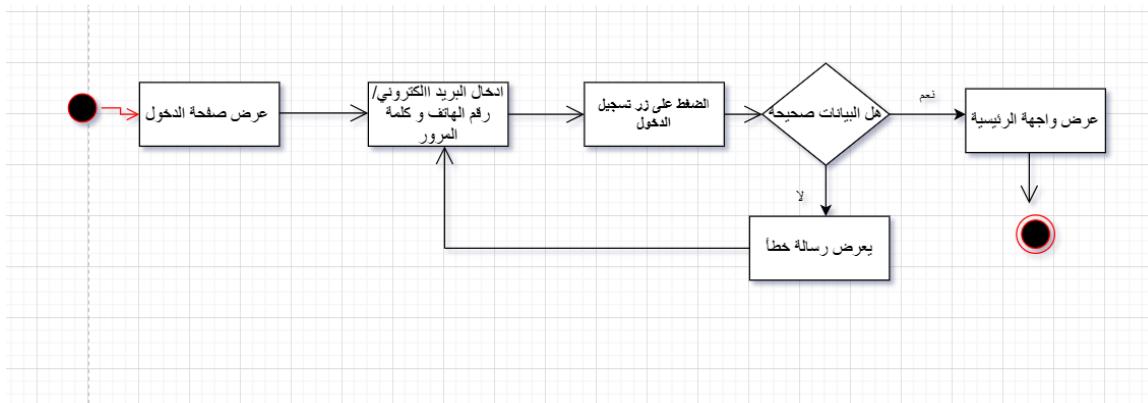
2.8 مخططات التسلسل Sequence Diagrams

UC-01	Use Case Name
المستخدم	Actors
تمكين المستخدم من انشاء حساب جديد على النظام باستخدام البريد الالكتروني او رقم الهاتف للتحقق من الهوية والوصول الى خدمات النظام.	Description
ليس لديه أي تسجيل دخول.	Precondition
1- يعرض النظام نموذج التسجيل . 2- يدخل المستخدم البيانات (الاسم و البريد الالكتروني وكلمة المرور او عن طريق GOOGLE) . 3- يضغط المستخدم على زر "REGISTER" . 4- يتحقق النظام من صحة البيانات ويرسل رمز تحقق لينشئ الحساب. 5- يدخل المستخدم رمز التتحقق الذي ارسل له عبر بريده الالكتروني. 6- يرسل رسالة تأكيد ويوجه المستخدم.	Main flow
A1: المستخدم يدخل البريد الالكتروني متكرر يرسل رسالة البريد الالكتروني : A2: المستخدم يدخل البريد الالكتروني متكرر يرسل رسالة البريد الالكتروني : A3: لا يرسل رمز للتحقق ويقوم المستخدم بإعادة ارسال رسالة	Alternative flow

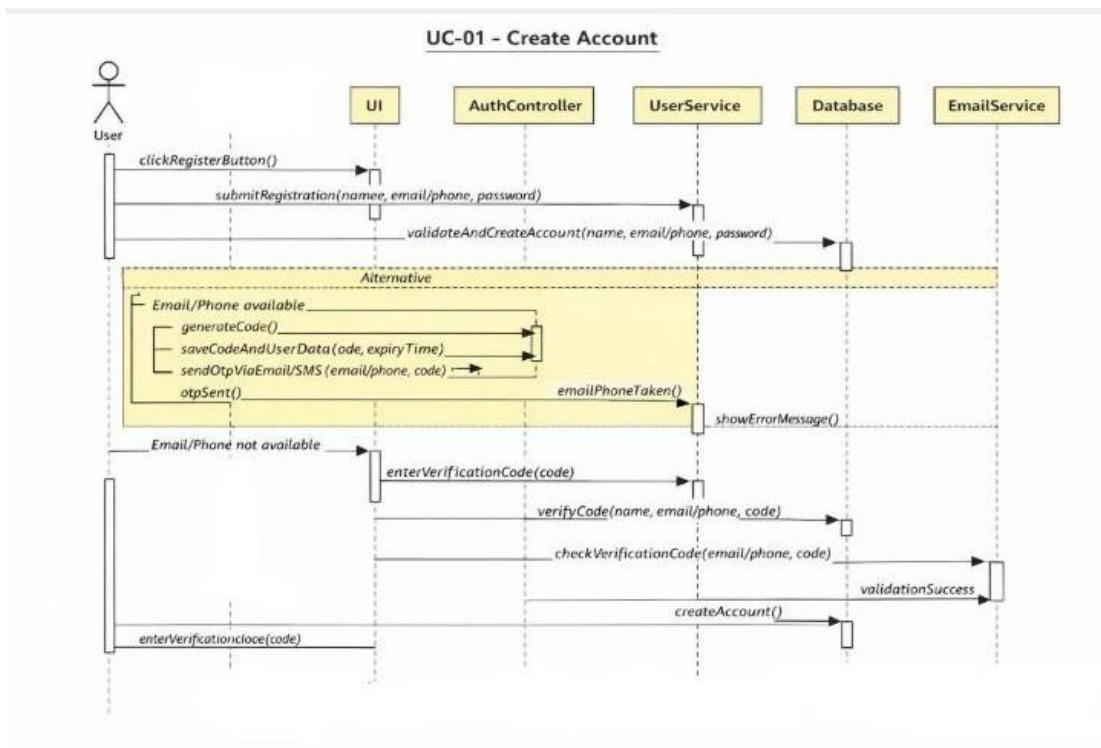
تم إنشاء الحساب وتسجيل الدخول تلقائي.

Postconditions

توضيف حالة تسجيل الحساب 2 جدول



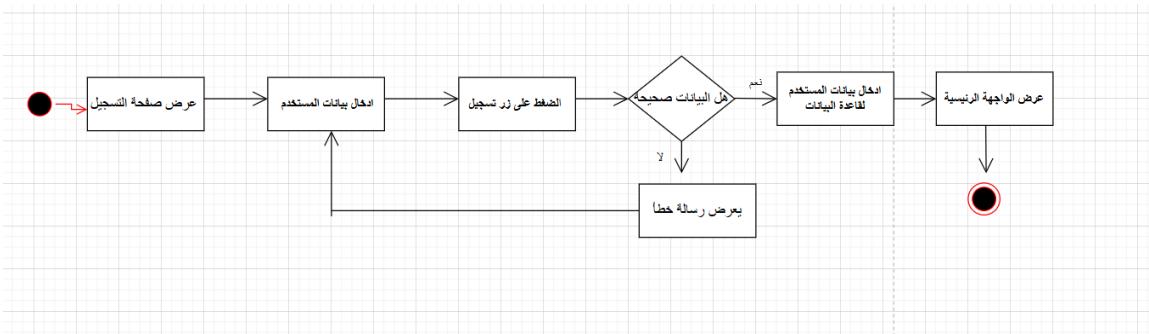
7 مخطط النشاط حالة تسجيل الحساب



8 مخطط التسلسل حالة تسجيل الحساب

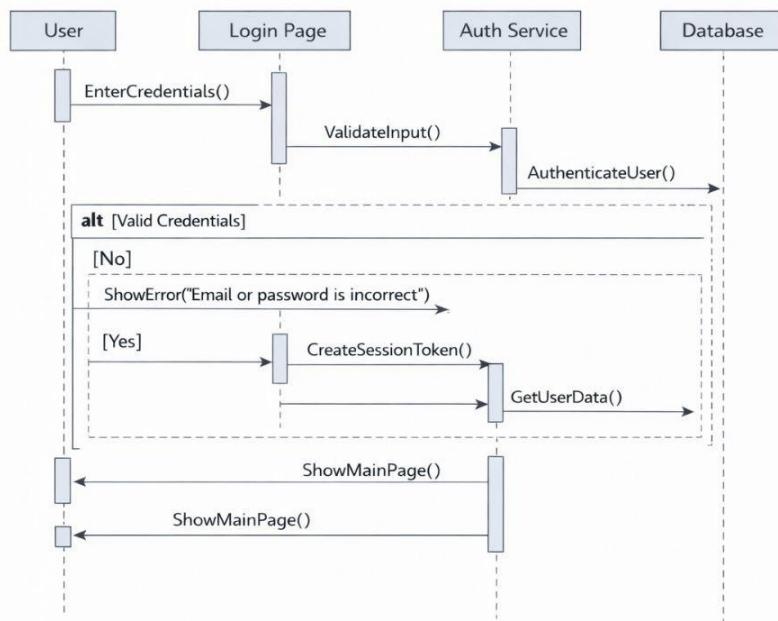
UC-02	Use Case Name
المستخدم	Actors
يتيح للمستخدم الوصول الى حسابه المسجل.	Description
ليس لديه أي تسجيل دخول.	Precondition
<p>1- يفتح المستخدم صفحة تسجيل الدخول .</p> <p>2- يدخل المستخدم البريد الإلكتروني أو رقم الهاتف وكلمة المرور.</p> <p>3- يتحقق النظام من صحة البيانات مقابل قاعدة البيانات أو خدمة Clerk Authentication.</p> <p>4- إذا كانت البيانات صحيحة النظام يصدر Token للجلسة ويعطي المستخدم وصولاً كاملاً للنظام.</p> <p>5- النظام يعرض واجهة المستخدم الرئيسية بعد تسجيل الدخول.</p>	Main flow
<p>A1: يتيح النظام تسجيل دخول عبر حساب facebook, google. A3: يعرض النظام رسالة خطأ "البريد الإلكتروني او كلمة المرور غير صحيحة".</p>	Alternative flow
يتم التحقق من هوية المستخدم ويسمح للمستخدم بالوصول الى حسابه وميزات النظام المختلفة.	Postconditions

جدول 3 توصيف حالة تسجيل الدخول



٩ مخطط النشاط حالة تسجيل الدخول

Login Sequence Diagram



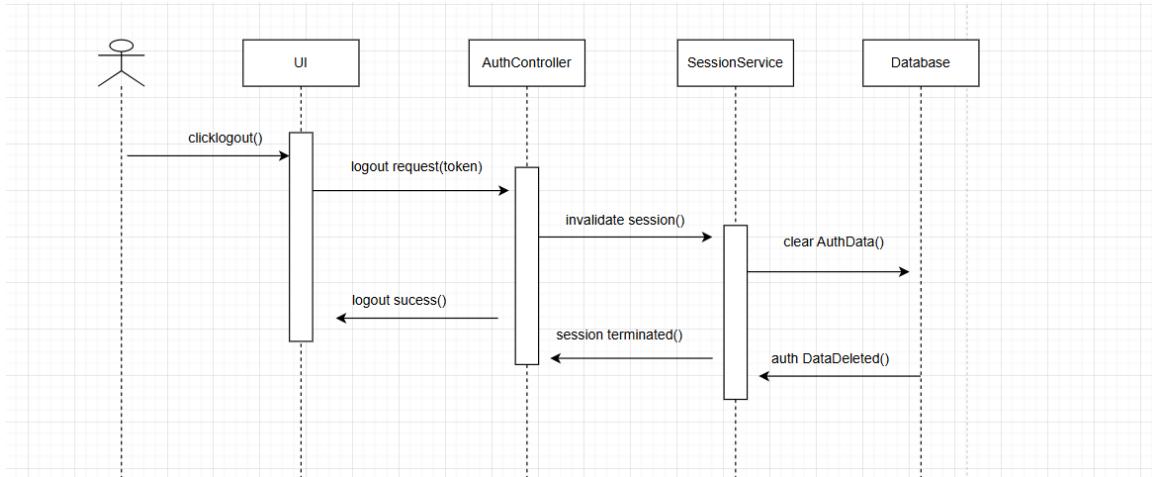
١٠ مخطط التسلسل حالة تسجيل الدخول

	UC-03	Use Case Name
	المستخدم	Actors
يتيح للمستخدم اكمال عملية تسجيل الدخول بعد ادخال اسم المستخدم وكلمة المرور بنجاح من خلال رمز التحقق المؤقت، يتم ارساله الى البريد الالكتروني لتعزيز الأمان.	Description	
المستخدم قام بإدخال اسم المستخدم وكلمة المرور بشكل صحيح.	Precondition	
<ul style="list-style-type: none"> - النظام يتحقق من صحة اسم المستخدم وكلمة المرور. - النظام ينشئ رمز تحقق مؤقت صالح لمدة زمنية محددة لبريد الالكتروني للمستخدم. - النظام يعرض شاشة ادخال الرمز ليقوم المستخدم بإدخاله. - النظام يتحقق من صحة الرمز المدخل ومن صلاحيته الزمنية. - النظام يؤكد نجاح العملية وينشئ الجلسة ويقوم بتوجيه المستخدم للصفحة الرئيسية. 	Main flow	
<p>المستخدم يدخل رمز تحقق خاطئ النظام يعرض رسالة خطأ توضح ان: A2: الرمز غير صحيح..</p> <p>A4: انتهاء صلاحية رمز التحقق، النظام يعرض رسالة بانتهاء صلاحية الرمز: حيث يوفر خيار إعادة رمز جديد.</p>	Alternative flow	
تم انشاء الجلسة بأمان.	Postconditions	

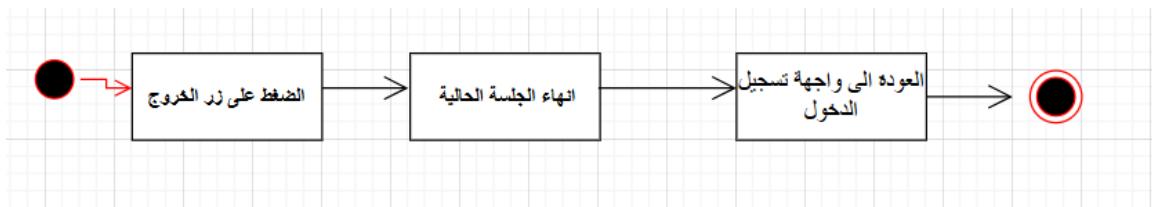
جدول 4 توصيف حالة الدخول الآمن

UC-04	Use Case Name
المستخدم	Actors
انهاء الجلسة الحالية والخروج بأمان من الحساب.	Description
النظام مسجل دخول بالفعل إلى النظام.	Precondition
1- المستخدم يضغط على زر تسجيل الخروج. 2- النظام يقوم بإلغاء الجلسة الحالية ويحذف بيانات المصادقة من المتصفح. 3- النظام يعيد توجيه المستخدم إلى صفحة تسجيل الدخول ويعرض رسالة "تم تسجيل الخروج بنجاح".	Main flow
A1: المستخدم يختار الغاء عملية الخروج ويبقى على الصفحة الحالية. A4: النظام يكتشف ان الجلسة منتهية ويتم تحويله مباشرة الى تسجيل الدخول.	Alternative flow
يتم تسجيل الخروج بنجاح ويتم تحويل المستخدم الى صفحة الدخول.	Postconditions

جدول 5 توصيف حالة تسجيل الخروج



11 مخطط التسلسل حالة تسجيل الخروج



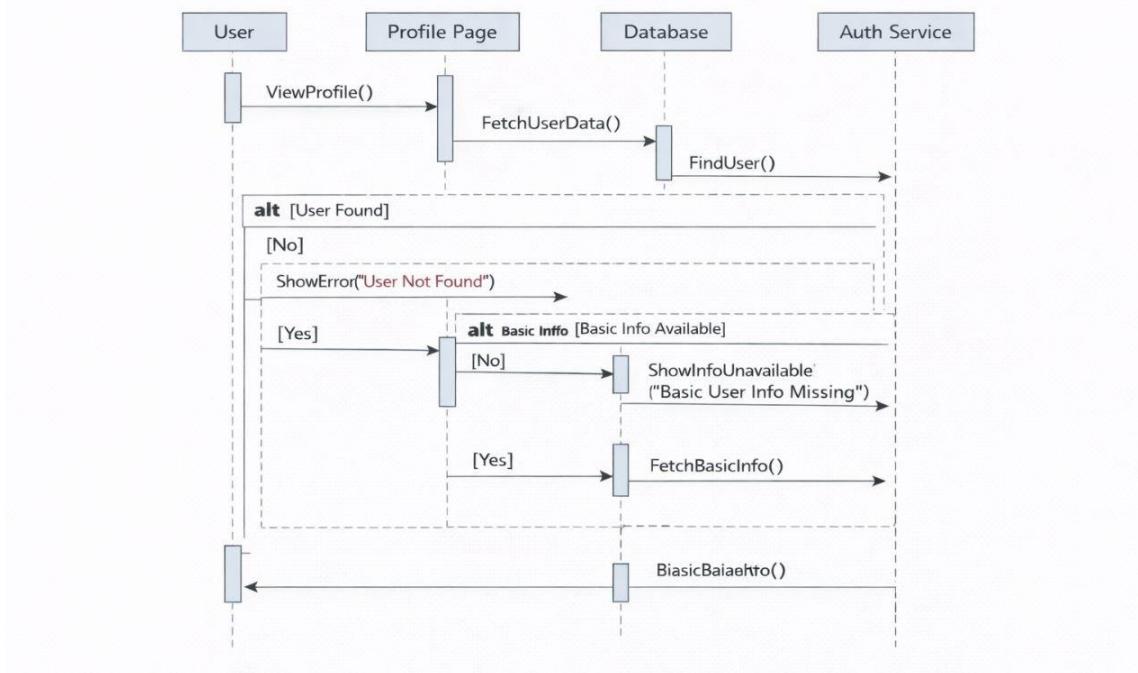
12 مخطط النشاط حالة تسجيل الخروج

UC-05	Use Case Name
المستخدم	Actors
تمكين المستخدم من رؤية المعلومات الأساسية لحساب مستخدم معين مثل الاسم والصورة الشخصية والنبذة وعدد المتابعين.	Description
1. تسجيل دخول المستخدم إلى حسابه مسبقا.	Precondition

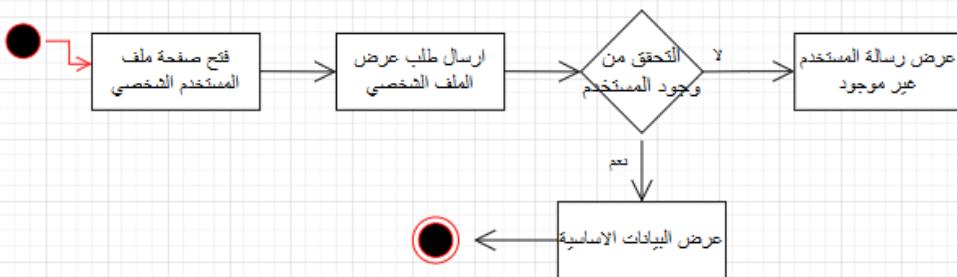
<p>1- يفتح صفحة ملف المستخدم الشخصي .</p> <p>2- يتحقق من وجود المستخدم في قاعدة البيانات .</p> <p>3- يجلب البيانات الأساسية (الاسم، الصورة، النبذة، عدد المتابعين) .</p> <p>4- يعرض المعلومات على واجهة المستخدم في صفحة ال .profile</p>	Main flow
<p>اذا لم يعثر على المستخدم في قاعدة البيانات يعرض النظام رسالة خطأ: A2: "المستخدم غير موجود".</p> <p>اذا لم تتوفر بعض المعلومات يعرض النظام عدم وجود المعلومات في الحقل A3: الاتي.</p>	Alternative flow
<p>تظهر المعلومات الأساسية للمستخدم بنجاح على الشاشة .</p>	Postconditions

جدول 6 توصيف حالة عرض الواجهة الأساسية

Show Basic Information Sequence Diagram



13 مخطط التسلسل حالة عرض الواجهة الأساسية

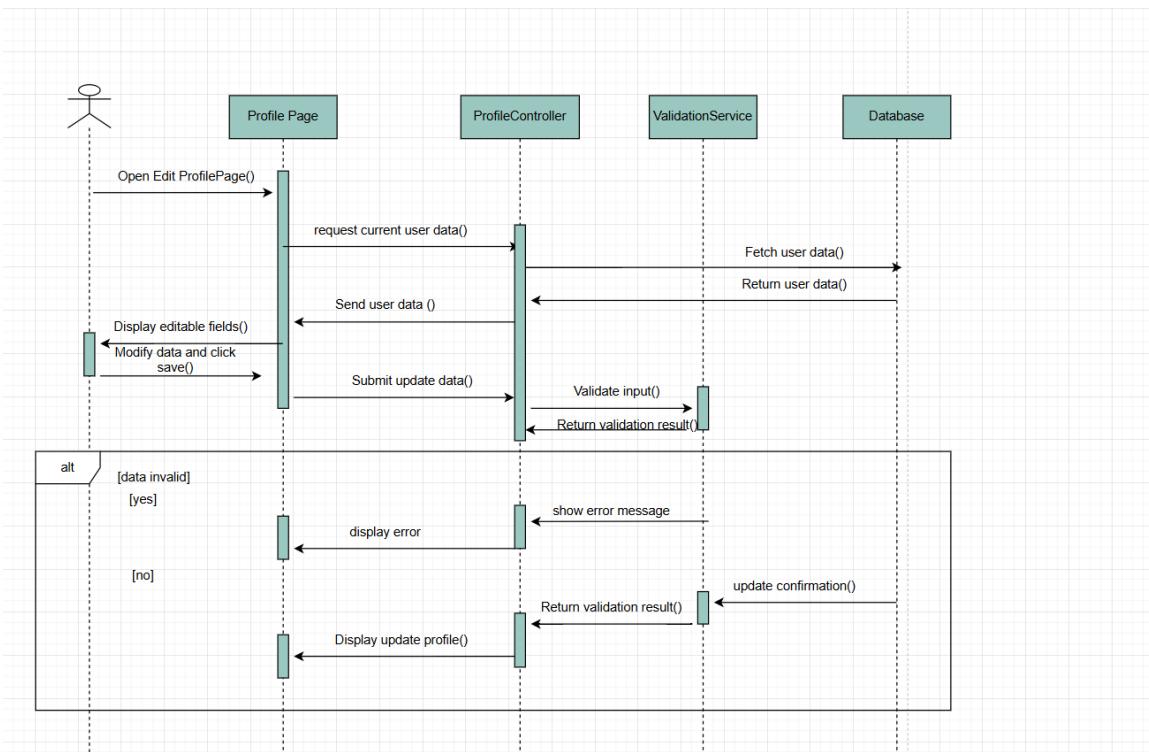


14 مخطط النشاط حالة عرض الواجهة الأساسية

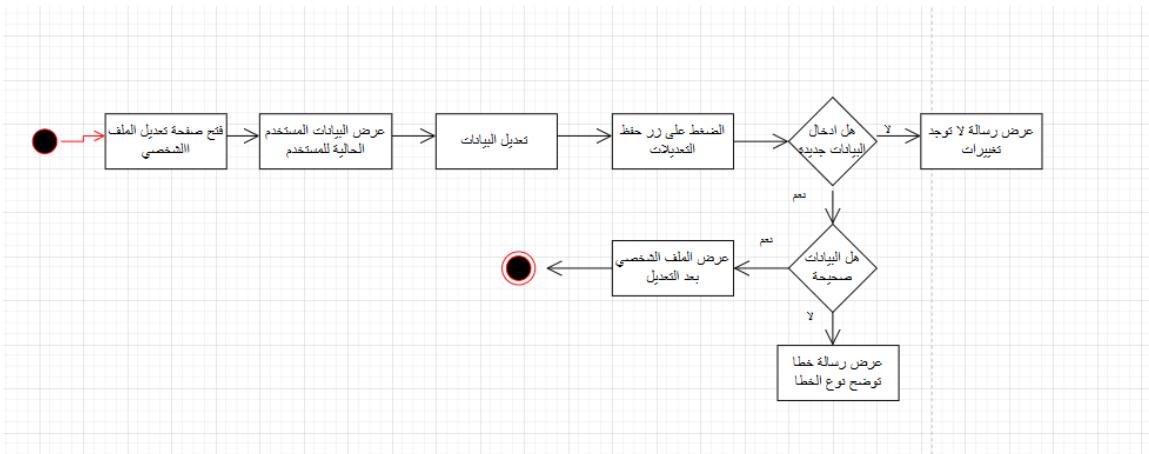
UC-06	Use Case Name
المستخدم	Actors
تمكين المستخدم من تعديل بياناتهم الشخصية المسجلة في النظام مثل الاسم و النبذة والصورة و البريد الإلكتروني و كلمة المرور .	Description
1. تسجيل دخول المستخدم إلى حسابه مسبقا.	Precondition

<p>1- يفتح صفحة تعديل الملف الشخصي .</p> <p>2- يعرض النظام البيانات الحالية المستخدم داخل حقول قابلة للتعديل .</p> <p>3- يقوم المستخدم بتعديل البيانات المطلوبة (الاسم والصورة والبريد ...) .</p> <p>4- يضغط المستخدم على زر "حفظ التعديلات" .</p> <p>5- يتحقق النظام من صحة المدخلات (تنسيق البريد الإلكتروني او حجم الصورة) .</p> <p>6- يقوم النظام بتحديث البيانات في قاعدة البيانات .</p> <p>7- يعرض النظام رسالة "تم تعديل الملف الشخصي بنجاح" .</p> <p>8- يعرض النظام الملف الشخصي بالبيانات الجديدة .</p>	Main flow
<p>A2: اذا كانت البيانات المدخلة غير صحيحة يعرض النظام رسالة خطأ توضح نوع الخطأ ويطلب من المستخدم التصحيح .</p> <p>A3: اذا لم يغير المستخدم اي بيانات وضغط "حفظ" يعرض النظام رسالة "لا: توجد تغييرات لتحديثها" .</p>	Alternative flow
<p>تظهر رسالة تأكيد بنجاح عملية التحديث وتعرض البيانات الجديدة في صفحة الملف الشخصي بعد التعديل .</p>	Postconditions

جدول 7 توصيف حالة التعديل على الحساب



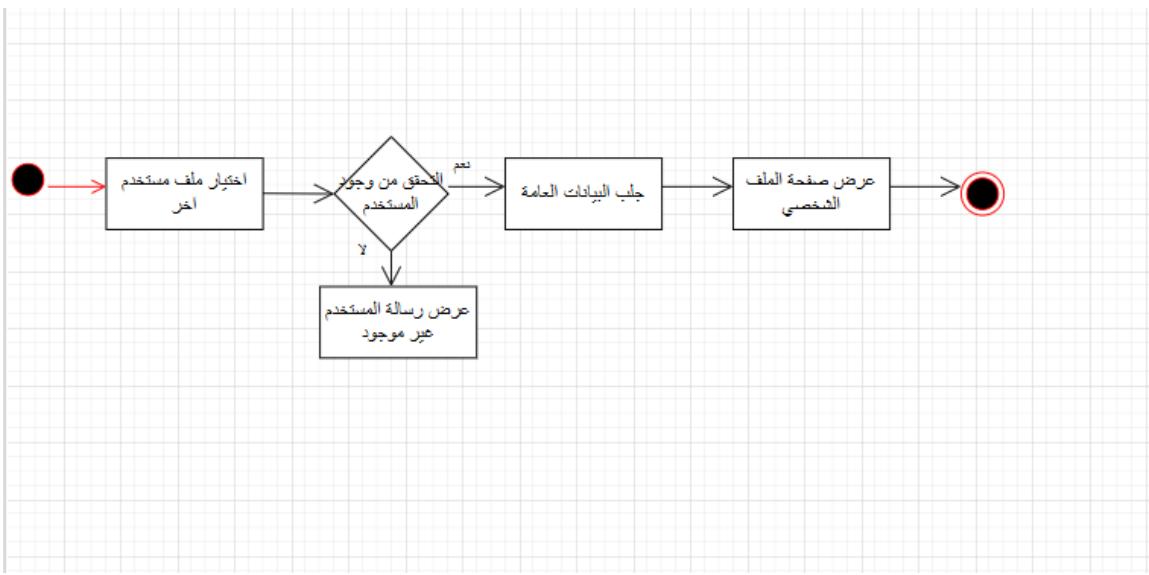
15 مخطط التسلسل حالة التعديل على الحساب



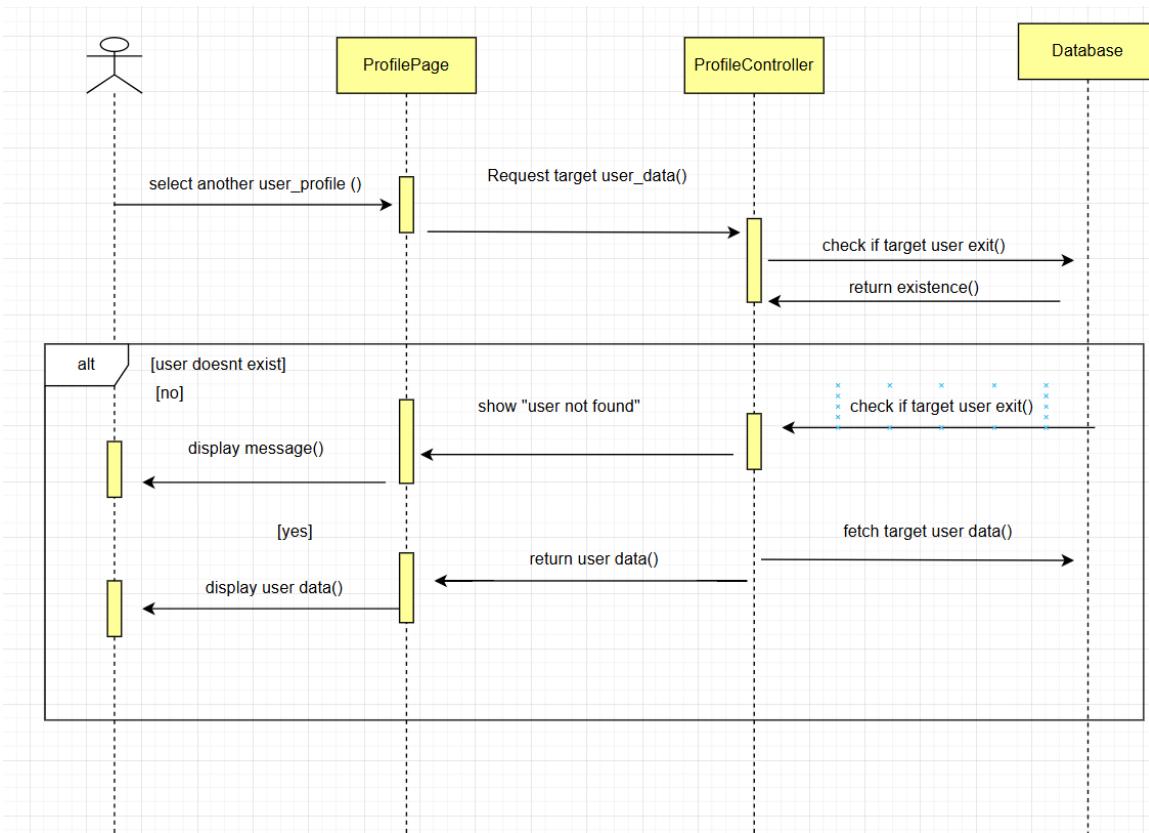
16 مخطط النشاط حالة التعديل على الحساب

UC-07	Use Case Name
المستخدم	Actors
تمكين المستخدم من زيارة صفحات المستخدمين الآخرين في النظام وعرض المعلومات عنه .	Description
1. تسجيل دخول المستخدم إلى حسابه مسبقا.	Precondition
1- يختار المستخدم ملف مستخدم آخر لعرضه اما من نتائج البحث او من قائمة المتابعين . 2- يتحقق النظام من وجود المستخدم الهدف في قاعدة البيانات. 3- يجلب النظام بيانات المستخدم الهدف وفق مستوى الصلاحية (الاسم والصورة والنبذة و عدد المتابعين و المنشورات العامة). 4- يعرض النظام صفحة الملف الشخصي للمستخدم الآخر بالمعلومات المسموح عرضها .	Main flow
A2: اذا لم يجد النظام المستخدم في قاعدة البيانات يعرض النظام رسالة "المستخدم غير موجود." A3: اذا كان الملف خاصا والمستخدم الحالى ليس من المتابعين المصرح لهم يعرض النظام رسالة "هذا الحساب خاص" ويظهر فقط المعلومات المحدودة (الصورة والاسم).	Alternative flow
تعرض صفحة المستخدم ملف المستخدم بالمعلومات العامة المسموح عرضها.	Postconditions

جدول 8 توصيف حالة عرض معلومات الحسابات



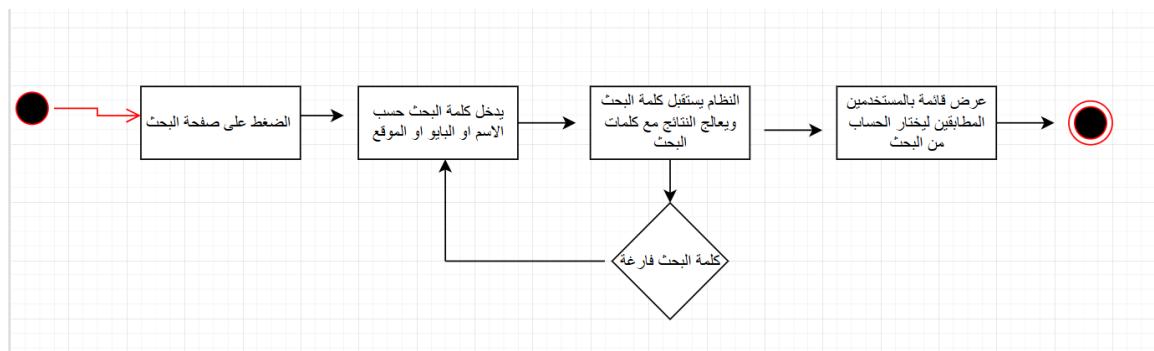
17 مخطط النشاط حالة عرض معلومات الحسابات



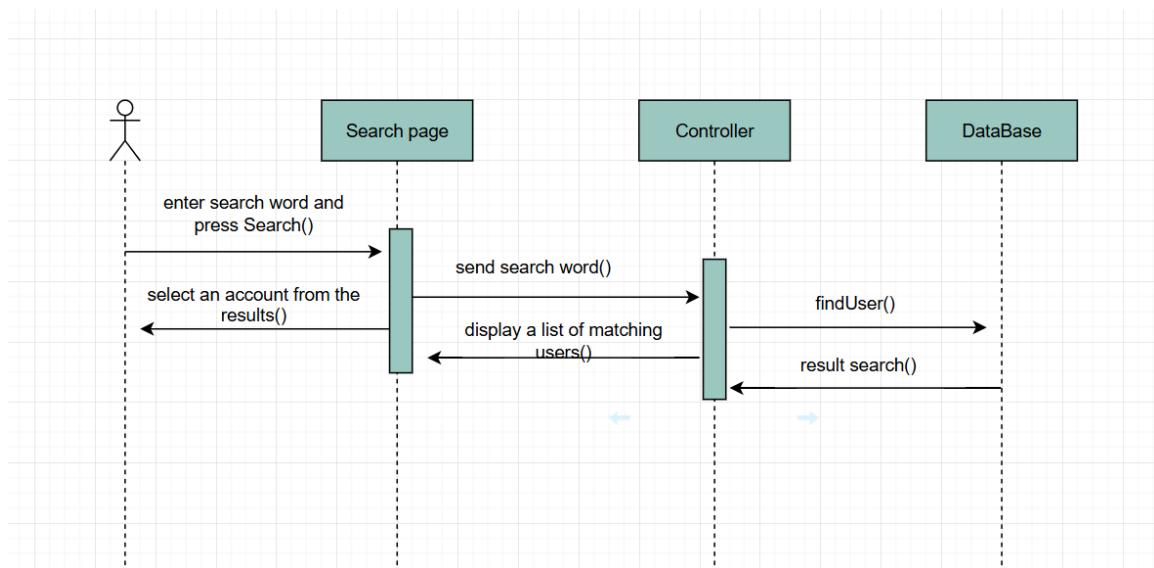
18 مخطط النشاط حالة عرض معلومات الحسابات

UC-08	Use Case Name
المستخدم	Actors
يتيح للمستخدم البحث عن مستخدمين آخرين داخل النظام باستخدام كلمات مفاتيحية .	Description
1. تسجيل دخول المستخدم إلى حسابه مسبقا.	Precondition
<p>1- يضغط المستخدم على صفحة البحث ويدخل كلمة البحث .</p> <p>2- المستخدم يضغط على زر البحث .</p> <p>3- النظام يستقبل كلمة البحث ويعالج النتائج ويطابقها مع كلمات البحث .</p> <p>4- النظام يعرض قائمة بالمستخدمين المطابقين ويختار الحساب من نتائج البحث.</p>	Main flow
<p>ادخال كلمة بحث فارغة النظام يعرض رسالة تطلب ادخال كلمة البحث.</p> <p>لا توجد نتائج النظام يعرض رسالة لا توجد نتائج مطابقة.</p>	Alternative flow
تم عرض نتائج البحث بنجاح.	Postconditions

جدول 9 توصيف حالة البحث عن الأصدقاء



19 مخطط النشاط حالة البحث عن الأصدقاء



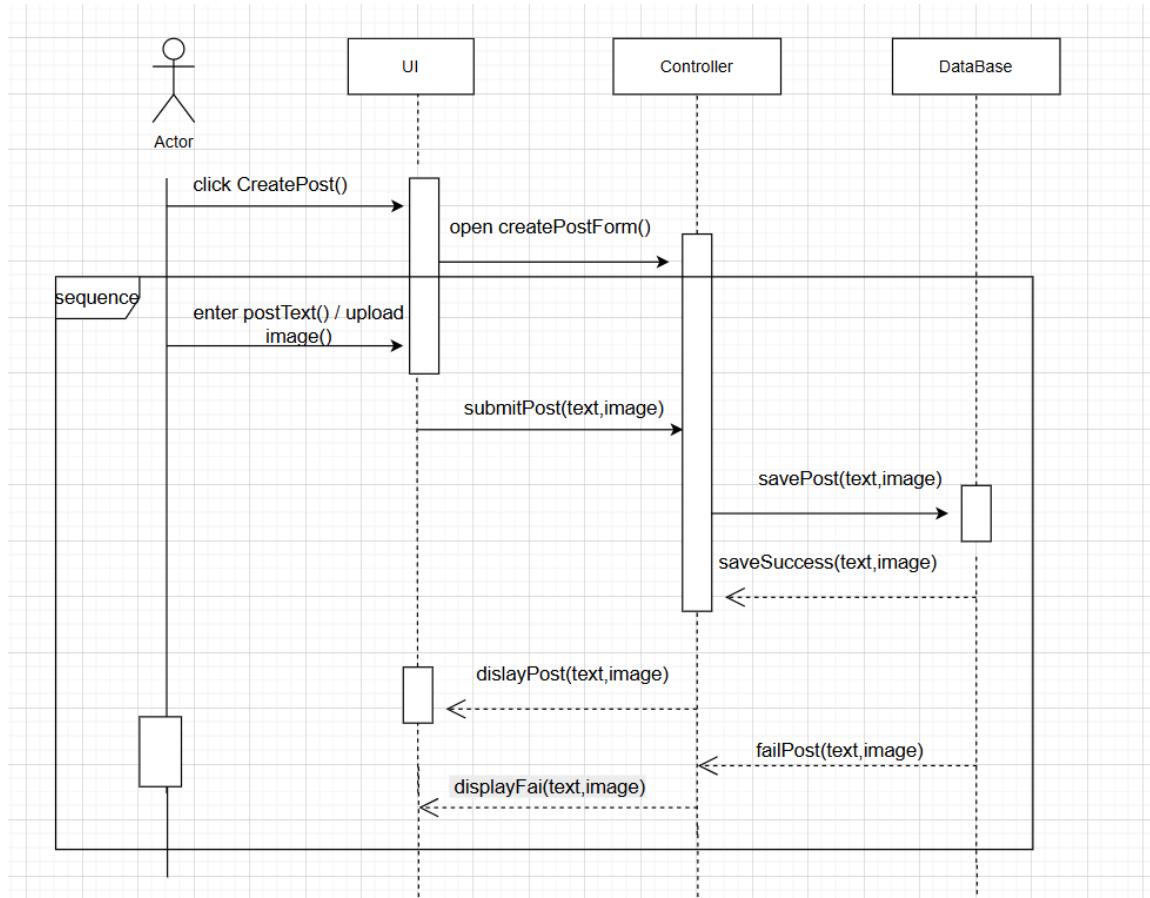
20 مخطط التسلسل حالة البحث عن الأصدقاء

	UC-09	Use Case Name
	المستخدم	Actors
تمكين المستخدم من نشر منشور جديد يحتوي على صورة او نص او كليهما .		Description
1. تسجيل دخول المستخدم إلى حسابه مسبقا.		Precondition
<p>5- يضغط المستخدم على زر "إنشاء منشور جديد" .</p> <p>6- تظهر نافذة لإدخال النص او تحميل صورة .</p> <p>7- يكتب المستخدم نص المنشور او يختار صورة .</p> <p>8- يضغط المستخدم على زر "تأكيد نشر" .</p> <p>9- يتحقق النظام من صحة البيانات (وجود صورة او نص) .</p> <p>10- يخزن النظام المنشور في قاعدة البيانات مع بيانات المستخدم والوقت .</p> <p>11- يعرض النظام المنشور الجديد في صفحة المستخدم .</p>	Main flow	
A4 - المستخدم لم يدخل أي محتوى (لا نص ولا صورة) يظهر تنبيه "المنشور فارغ" A5 - فشل رفع الصورة يظهر تنبيه "حدث خطأ أثناء رفع الصورة"	Alternative flow	

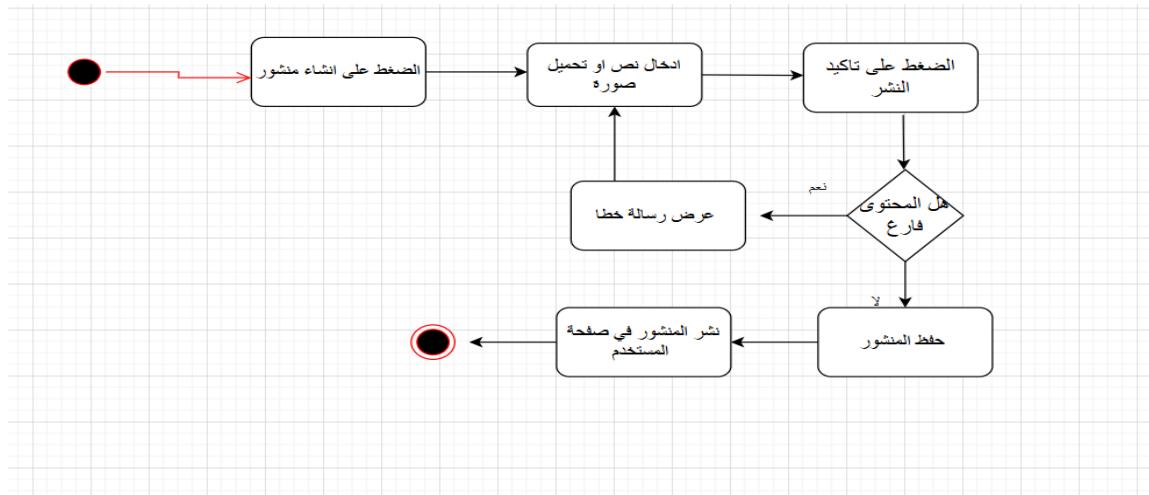
Postconditions

تم عرض المنشور في صفحة المستخدم وحفظه في قاعدة البيانات

جدول 10 توصيف حالة نشر منشور جديد



مخطط التسلسل حالة نشر منشور جديد 21

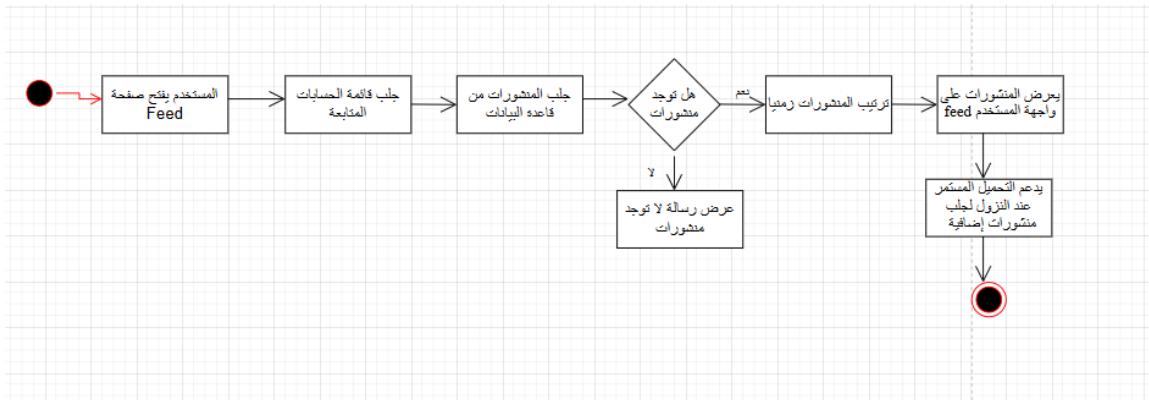


22 مخطط النشاط حالة نشر منشور جديد

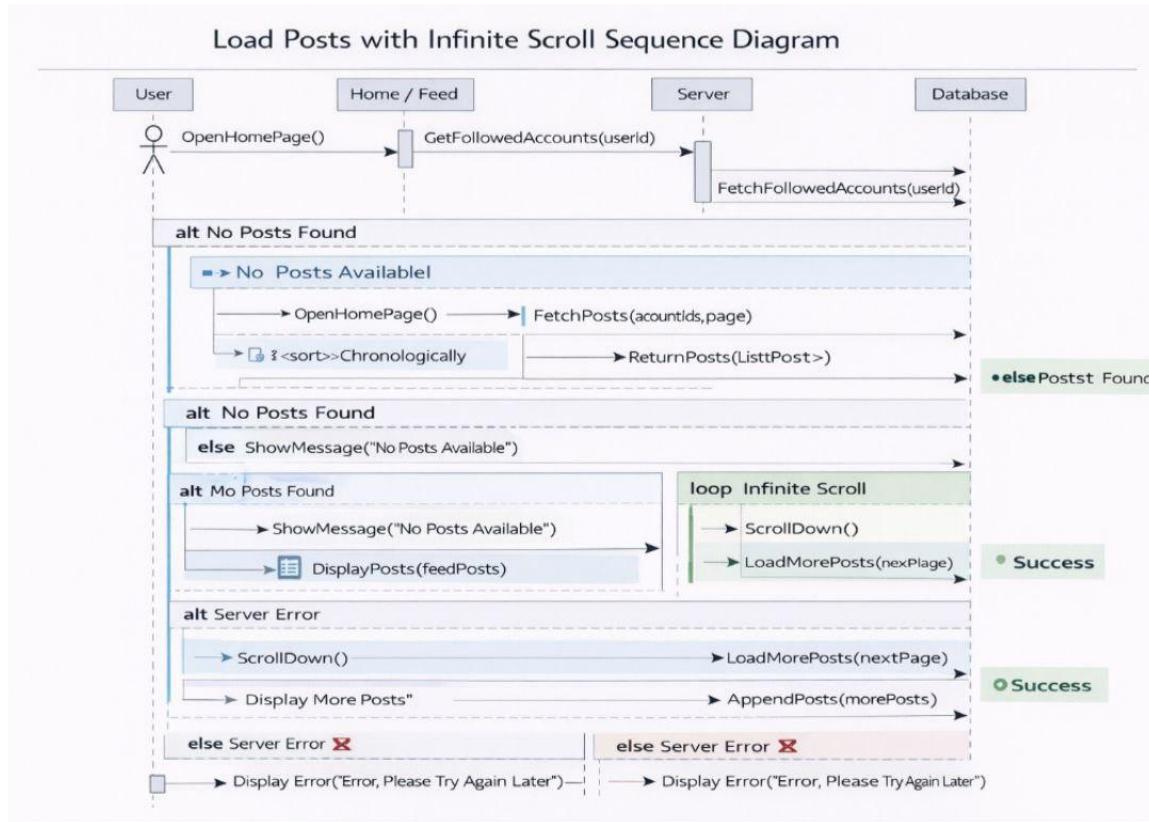
UC-10	Use Case Name
المستخدم	Actors
يجلب النظام المنشورات من قاعدة البيانات في الصفحة ويعرضها وفق ترتيب زمني مع دعم تحميل المستمر عند النزول.	Description

<p>1. تسجيل دخول المستخدم إلى حسابه مسبقاً.</p> <p>2. المستخدمون لديه متابعون او يتبع حسابات تنشر منشورات.</p>	Precondition
<p>1-يفتح المستخدم الصفحة الرئيسية (Home/Feed).</p> <p>2-يجلب النظام قائمة الحسابات التي يتبعها المستخدم .</p> <p>3-يجلب النظام المنشورات من هؤلاء الحسابات من قاعدة البيانات.</p> <p>4-يرتب المنشورات ترتيباً زمنياً من الاحدث الى الاقديم.</p> <p>5-يعرض المنشورات على واجهة المستخدم (Feed).</p> <p>6-يدعم التحميل المستمر عند النزول (infinite scroll) لجلب منشورات إضافية عند الحاجة.</p>	Main flow
<p>لم يجد النظام أي منشورات للحسابات التي يتبعها المستخدم يعرض رسالة "لا توجد منشورات حالياً".</p> <p>عند نشر منشورات جديدة يقوم النظام بتحديث الصفحة بشكل ديناميكي A5:</p>	Alternative flow
<p>1-تظهر جميع المنشورات للمستخدم بالترتيب الزمني الصحيح.</p> <p>2-يتتم تحديث الصفحة بشكل مستمر عند وجود منشورات جديدة.</p>	Postconditions

جدول 11 توصيف حالة عرض المنشورات



23 مخطط النشاط حالة عرض المنشورات

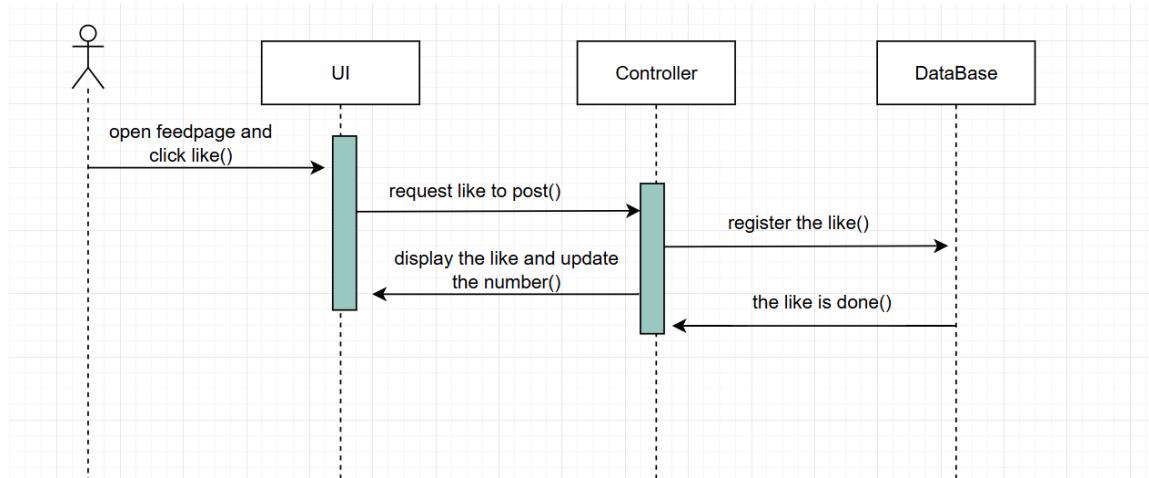


2 مخطط التسلسل حالة عرض المنشورات

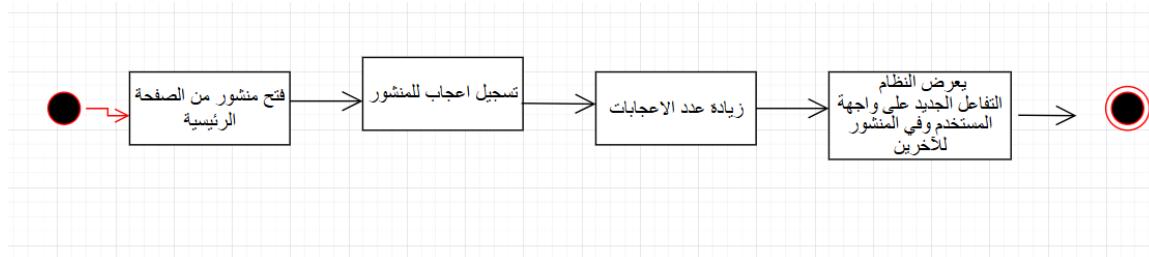
UC-11	Use Case Name
المستخدم	Actors
يسمح للمستخدم بإظهار رأيه تجاه المنشورات عبر إعطاء إعجاب	Description

<p>1. تسجيل دخول المستخدم إلى حسابه مسبقاً.</p> <p>2. المنشور موجود ويمكن الوصول إليه للسماح بالتفاعل مع المنشور.</p>	Precondition
<p>1-يفتح المستخدم المنشورة في الصفحة الرئيسية او الملف الشخصي.</p> <p>2-يضغط المستخدم زر اعجاب.</p> <p>3-يسجل النظام التفاعل ويزيد عدد الاعجابات.</p> <p>4-يعرض النظام التفاعل الجديد على واجهة المستخدم وفي المنشور لآخرين.</p>	Main flow
<p>A3: يعرض النظام رسالة: هذا المنشور غير متاح.</p>	Alternative flow
<p>يزداد عدد الاعجابات على المنشور ويظهر للمستخدمين الآخرين .</p>	Postconditions

جدول 12 توصيف حالة الإعجاب بالمنشورات



25 مخطط التسلسل حالة الإعجاب بالمنشورات



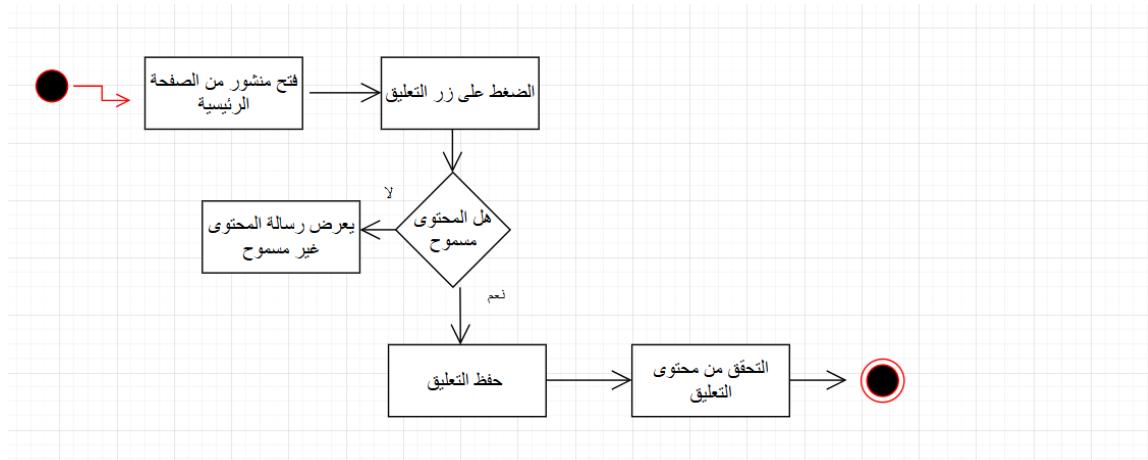
26 مخطط النشاط حالة الإعجاب بالمنشورات

UC-12	Use Case Name
المستخدم	Actors
Description	
1. تسجيل دخول المستخدم إلى حسابه مسبقا. 2. المنشور موجود ويمكن الوصول إليه للسماح بالتفاعل مع المنشور.	Precondition
1-يفتح المستخدم المنشورة في الصفحة الرئيسية او الملف الشخصي. 2-يضغط المستخدم زر تعليق . 3-يتتحقق النظام من وجود المنشور وصلاحية الوصول اليه. 4-يتتحقق النظام من محتوى التعليق ثم يحفظه في قاعدة البيانات. 5-يعرض النظام التفاعل الجديد على واجهة المستخدم وفي المنشور لآخرين.	Main flow
A3: يعرض النظام رسالة: هذا المنشور غير متاح. A5: النظام يرفض التعليقات ويعرض رسالة "التعليق يحتوي على محتوى غير مسموح به".	Alternative flow

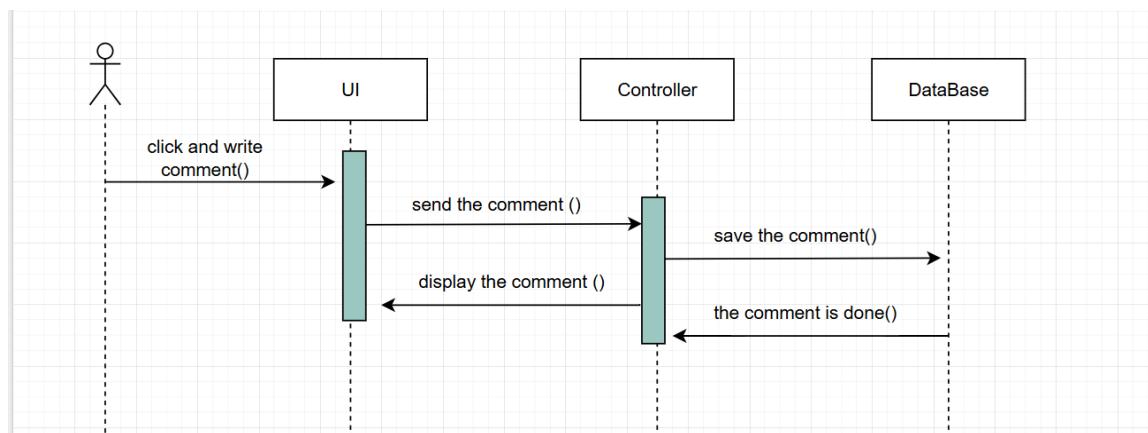
يتم حفظ التعليق في قاعدة البيانات ويظهر علو المنشور بالترتيب الزمني الصحيح.

Postconditions

جدول 13 توصيف حالة التعليق على المنشورات



27 مخطط النشاط حالة التعليق على المنشورات

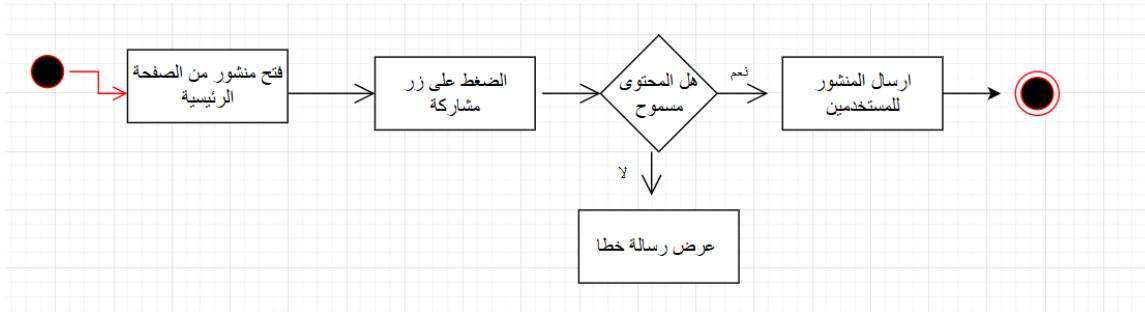


3 مخطط التسلسل حالة التعليق على المنشورات

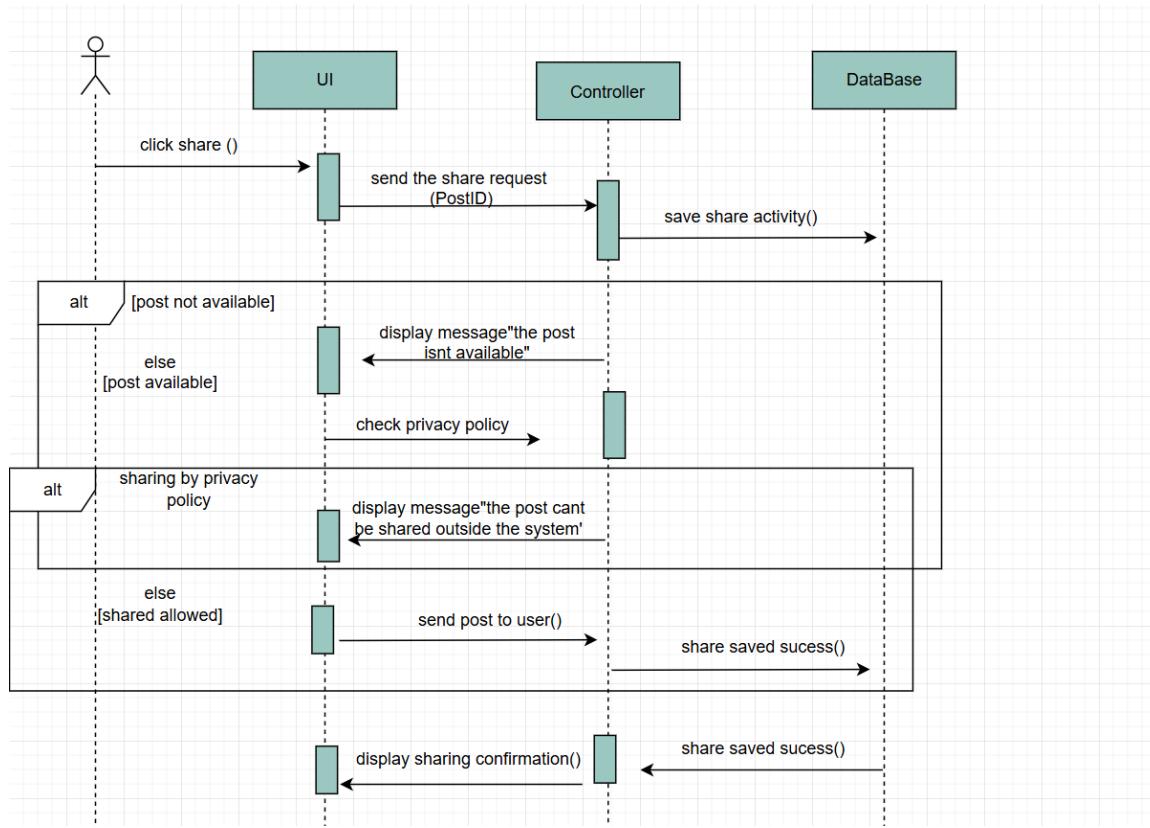
	UC-13	Use Case Name
--	-------	---------------

المستخدم	Actors
يتيح للمستخدم لتحكم بالمنشورات التي يود مشاركتها.	Description
1. تسجيل دخول المستخدم إلى حسابه مسبقاً.	Precondition
<p>1-يفتح المستخدم المنشورة في الصفحة الرئيسية او الملف الشخصي..</p> <p>2-يضغط المستخدم زر للمشاركة .</p> <p>3-يتتحقق النظام من وجود المنشور وصلاحية الوصول اليه.</p> <p>4-يقوم المستخدم بنشر المنشور للمستخدمين .</p>	Main flow
<p>A3: عرض النظام رسالة : هذا المنشور غير متاح: إذا كانت سياسة الخصوصية تمنع مشاركة المنشور خارجياً، يظهر النظام رسالة: "لا يمكن مشاركة هذا المنشور خارج النظام".</p>	Alternative flow
يتم انشاء رابط للمشاركة او ارسال المنشور للمستخدمين الاخرين وتسجل عملية المشاركة في النظام.	Postconditions

جدول 14 توصيف حالة مشاركة المنشورات



49 مخطط النشاط حالة مشاركة المنشورات

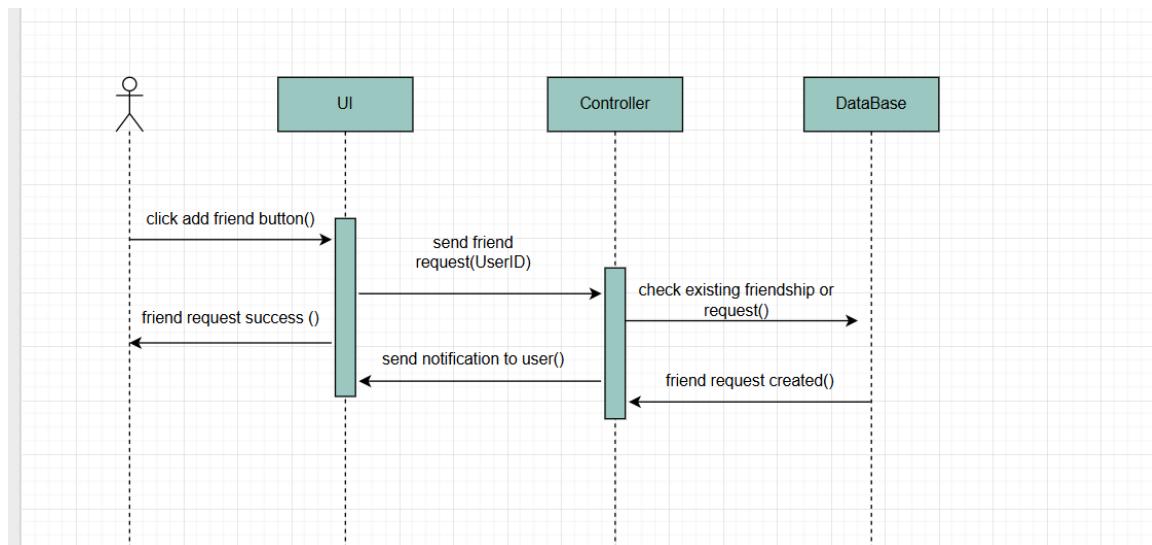


30 مخطط التسلسل حالة مشاركة المنشورات

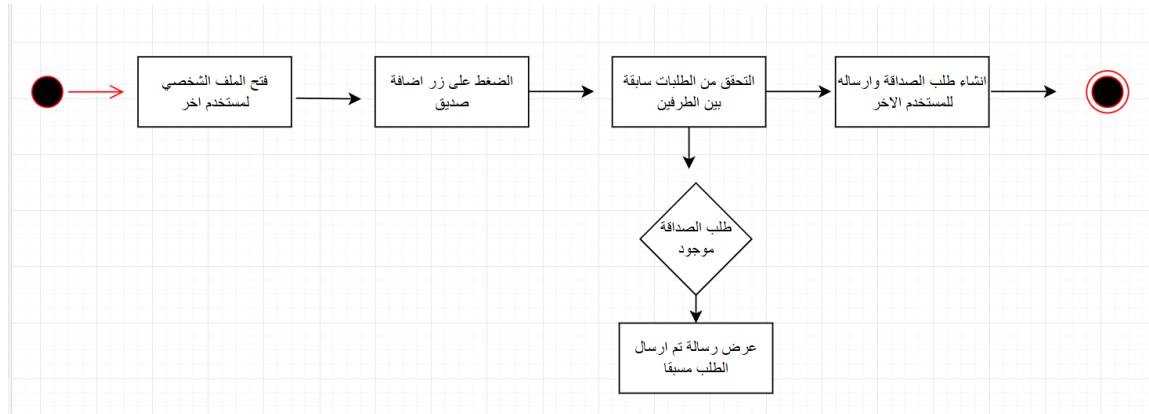
UC-14	Use Case Name
المستخدم	Actors
ارسال طلب صداقة الى المستخدم .	Description
1.تسجيل دخول المستخدم إلى حسابه مسبقا.	Precondition

<p>1-يفتح المستخدم الملف الشخصي لمستخدم اخر.</p> <p>2-يضغط المستخدم على زر إضافة صديق.</p> <p>3-النظام يتحقق من عدم وجود صداقه او طلب سابق بين الطرفين.</p> <p>4-النظام ينشئ طلب صداقه جديد.</p> <p>5-النظام يرسل اشعاراً للمستخدم الآخر بوجود طلب صداقه.</p> <p>6-النظام يعرض رسالة بان طلب الصداقه قد تم ارساله.</p>	Main flow
طلب الصداقه موجود مسبقاً النظام يعرض رسالة تم ارسال طلب مسبقاً: A3	Alternative flow
يتم انشاء طلب صداقه.	Postconditions

جدول 15 توصيف حالة إضافة صديق



5 مخطط التسلسل حالة إضافة صديق



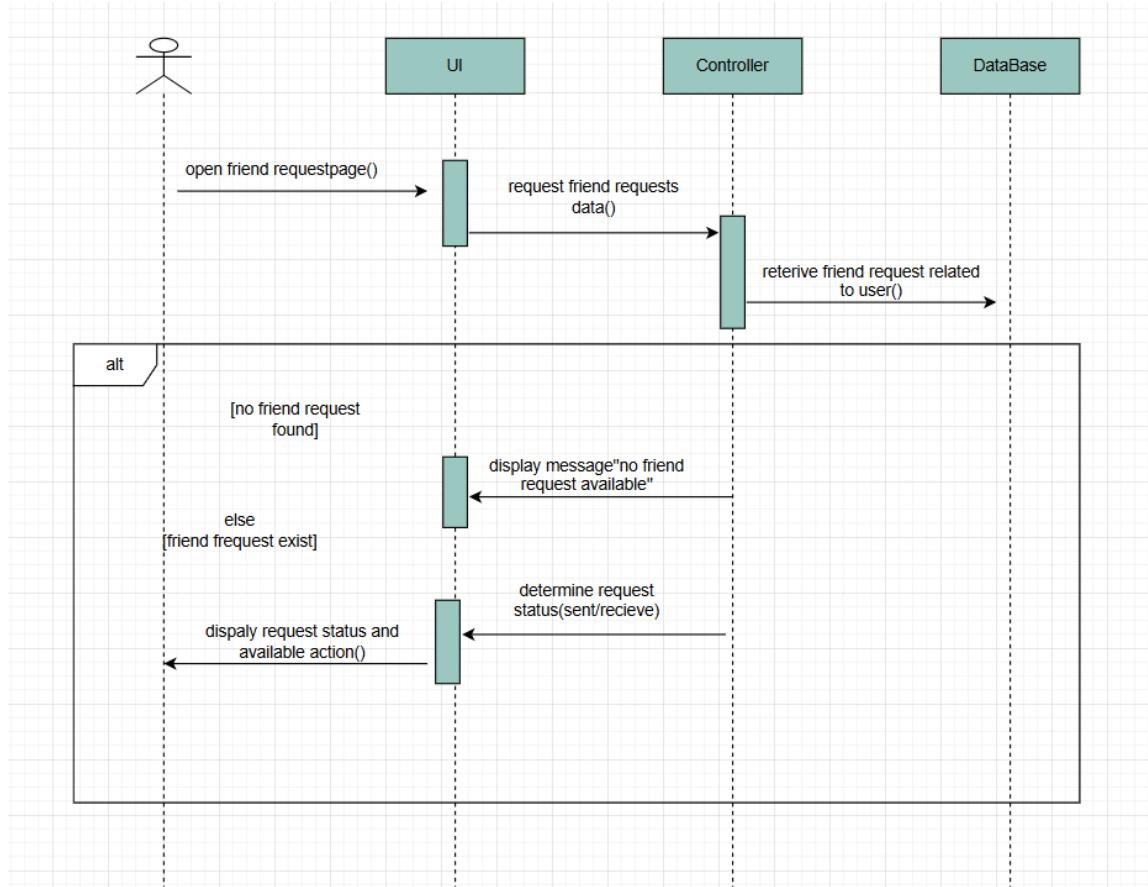
32 مخطط النشاط حالة إضافة صديق

UC-15	Use Case Name
المستخدم	Actors
يتيح للمستخدم معرفة حالة طلب الصداقة الذي قام بارساله او استقباله .	Description
1. تسجيل دخول المستخدم إلى حسابه مسبقا.	Precondition
1. المستخدم يفتح صفحة طلبات الصداقة . 2. النظام يسترجع بيانات طلب الصداقة المرتبط بين المستخدمين. 3. النظام يتحقق ويعرض حالة الطلب مع عرض الخيارات المتاحة(الغاء او قبول او رفض).	Main flow
لا يوجد أي طلبات صداقة: A2:	Alternative flow

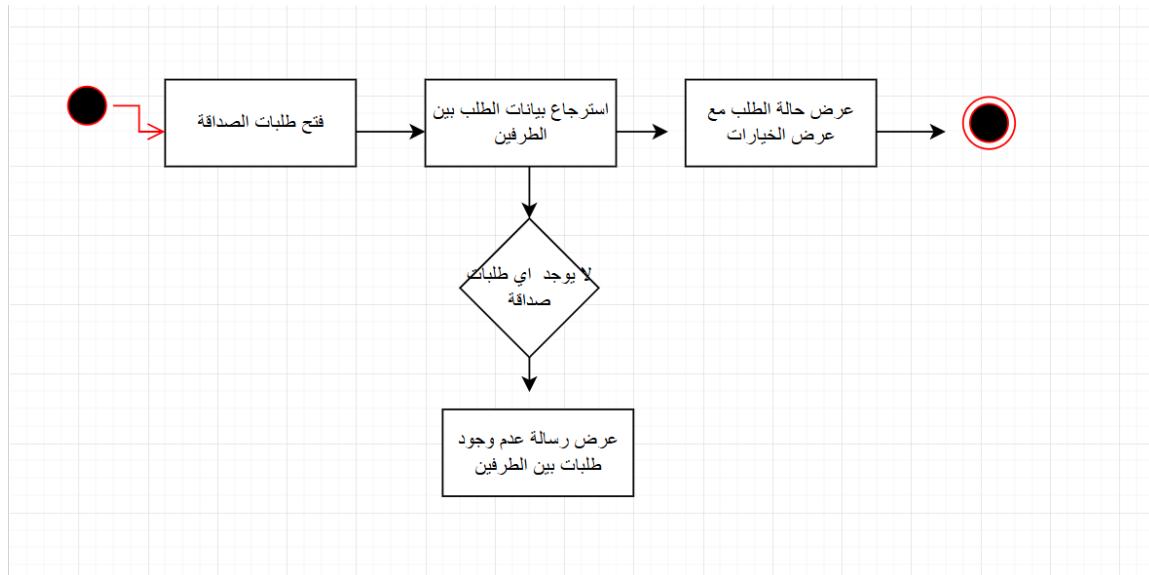
عرض حالة طلب المستخدم.

Postconditions

جدول 16 توصيف معرفة حالة طلب الصداقة



6 مخطط التسلسل معرفة حالة طلب الصداقة

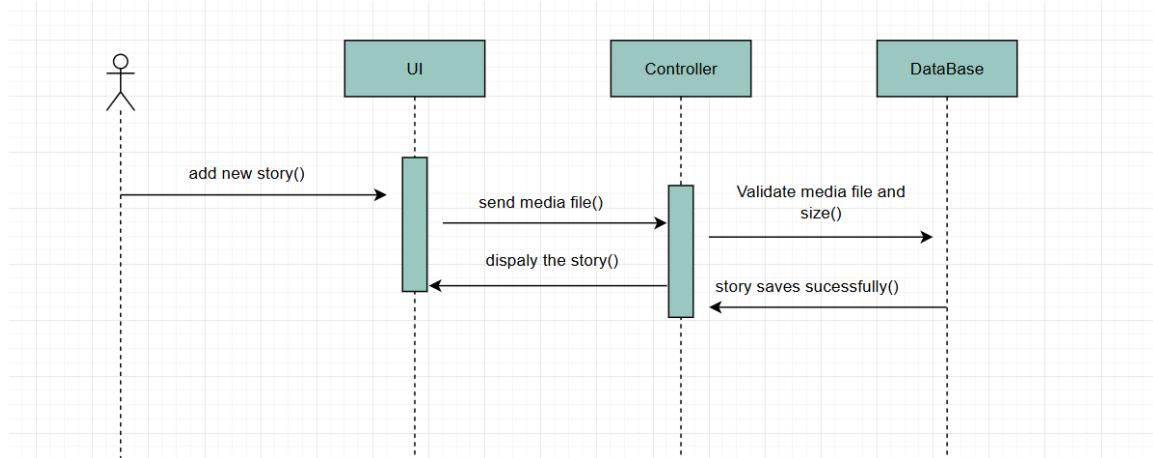


347 مخطط النشاط معرفة حالة طلب الصداقة

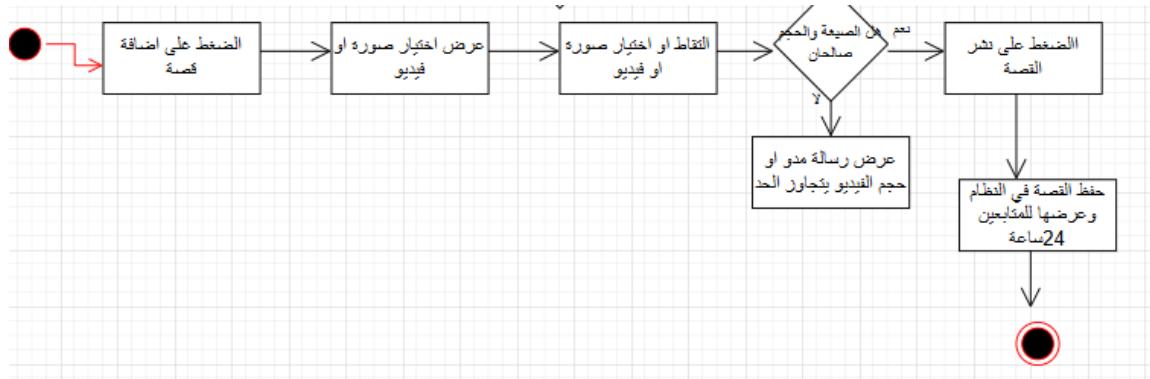
UC-16	Use Case Name
المستخدم	Actors
يتيح للمستخدم رفع قصة جديدة على حسابه بصيغة صورة أو فيديو قصير، بحيث تكون متاحة للمشاهدة لفترة زمنية محددة (24 ساعة)، مما يعزز التفاعل والمشاركة داخل المنصة.	Description
1. تسجيل دخول المستخدم إلى حسابه مسبقاً.	Precondition

<p>1. المستخدم يضغط على إضافة قصة جديدة.</p> <p>2. النظام يعرض خيار اختيار صورة أو فيديو قصير.</p> <p>3. المستخدم يلتقط صورة/فيديو أو يختار من المعرض.</p> <p>4. النظام يتحقق من صيغة وحجم الملف.</p> <p>5. المستخدم يضيف (نص، ملصقات، فلاتر – اختياري) ويضغط على نشر القصة.</p> <p>6. النظام ينشر القصة و يجعلها مرئية لمدة 24 ساعة.</p>	Main flow
<p>حجم الفيديو أكبر من المسموح يعرض رسالة: "مدة أو حجم الفيديو: A4: يتجاوز الحد المسموح"</p>	Alternative flow
<p>يتم نشر القصة بنجاح وتصبح مرئية للمتابعين أو الأصدقاء حسب إعدادات الخصوصية.</p>	Postconditions

جدول 17 توصيف حالة إضافة قصة جديدة



85 مخطط التسلسل حالة إضافة قصة جديدة

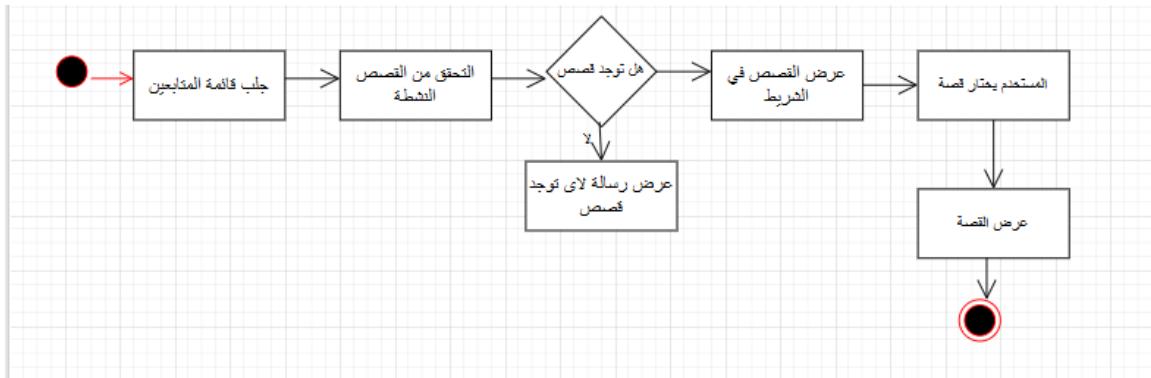


96 مخطط النشاط حالة إضافة قصة جديدة

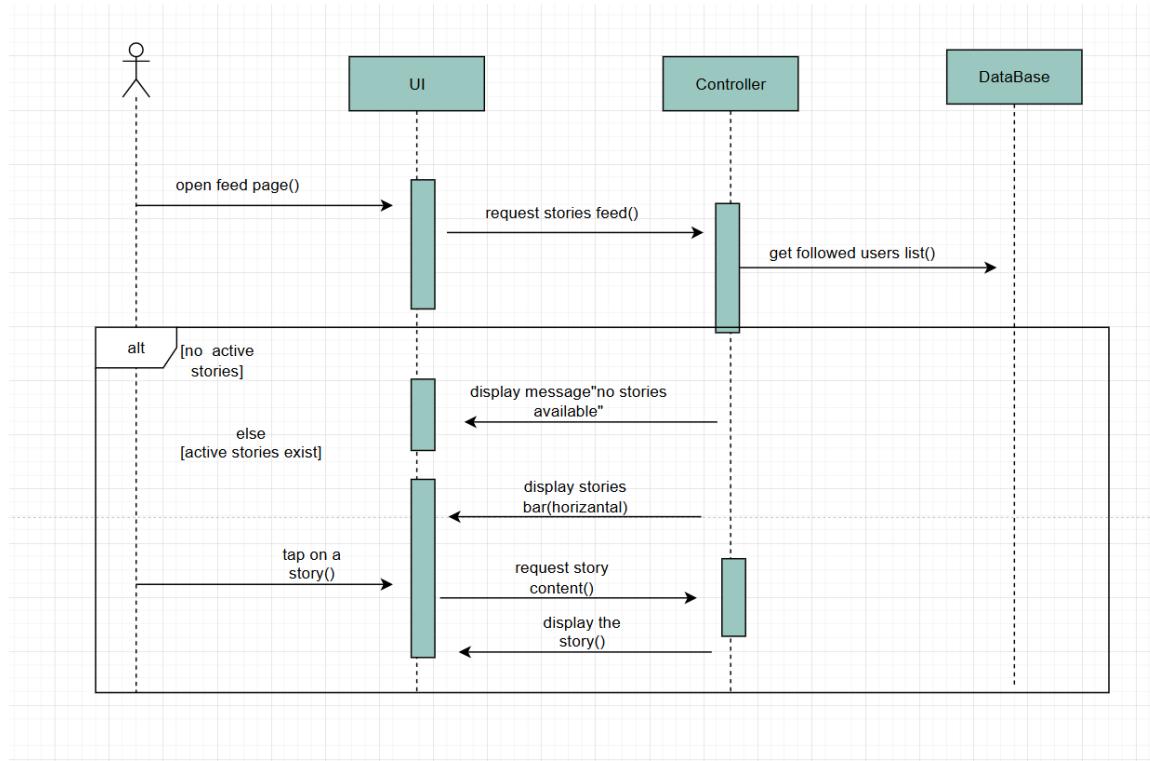
UC-17	Use Case Name
المستخدم	Actors
يتتيح للمستخدم عرض القصص المنشورة من قبل المستخدمين الذين يتبعهم فقط، وذلك وفقاً لإعدادات الخصوصية ومدة صلاحية القصة، مما يضمن تجربة مشاهدة مخصصة وآمنة.	Description
1. تسجيل دخول المستخدم إلى حسابه مسبقاً.	Precondition
1. المستخدم يدخل إلى الصفحة الرئيسية. 2. النظام يجلب قائمة المستخدمين الذين يتبعهم المستخدم ويتتحقق من وجود قصص نشطة. 3. النظام يعرض القصص في شريط علوي بشكل أفقي. 4. المستخدم يضغط على قصة أحد المتابعين. 5. النظام يعرض القصة (صورة أو فيديو). 6. بعد انتهاء القصة، النظام ينتقل تلقائياً إلى القصة التالية.	Main flow

A2: لا توجد قصص نشطة. A2: القصة ممحوبة بالخصوصية القصة غير مسموح بعرضها لهذا المستخدم. A6: انتهاء مدة القصة أثناء المشاهدة وتنتهي مدة صلاحية القصة.	Alternative flow
يتم عرض القصص المتاحة للمتابعين بترتيب زمني.	Postconditions

جدول 18 توصيف عرض القصص المتاحة للمتابعين



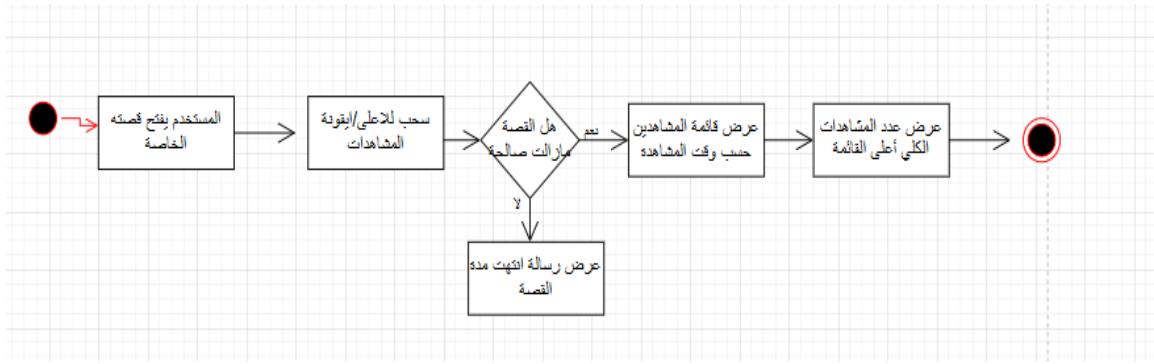
107 مخطط النشاط حالة عرض القصص المتاحة للمتابعين



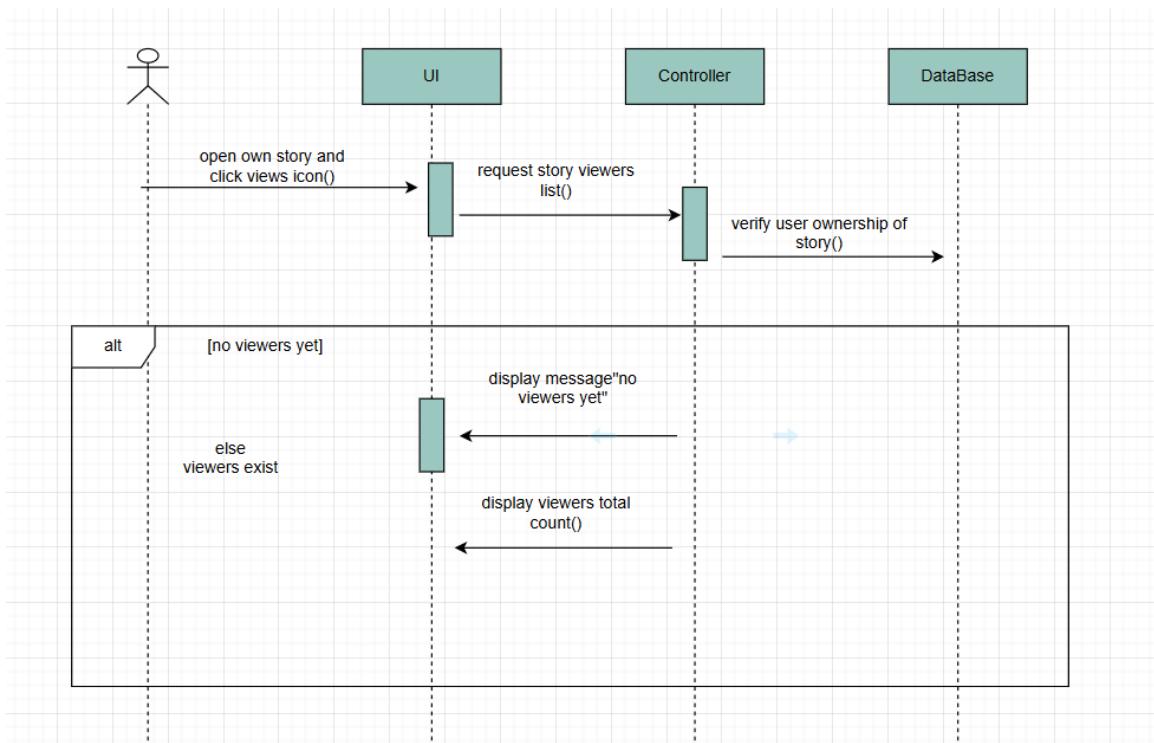
118 مخطط التسلسل حالة عرض القصص المتابعة للمتابعين

UC-18	Use Case Name
المستخدم	Actors
يتيح لصاحب القصة عرض قائمة المستخدمين الذين قاموا بمشاهدة قصته، مما يوفر شفافية حول التفاعل مع المحتوى المؤقت داخل المنصة.	Description
1. تسجيل دخول المستخدم إلى حسابه مسبقا.	Precondition
1. المستخدم يفتح قصته الخاصة. 2. المستخدم يسحب للأعلى أو يضغط على أيقونة المشاهدين. 3. النظام يتحقق من صلاحية المستخدم لعرض القائمة. 4. النظام يعرض قائمة المشاهدين مرتبة حسب وقت المشاهدة. 5. النظام يعرض عدد المشاهدات الكلية أعلى القائمة.	Main flow
A1:.. انتهاء مدة القصة (24 ساعة) A2:.. لا يوجد مشاهدون لا توجد أي مشاهدة بعد	Alternative flow
يتم عرض قائمة المشاهدين بشكل صحيح مع وقت المشاهدة	Postconditions

جدول 19 توصيف عرض قائمة المشاهدين



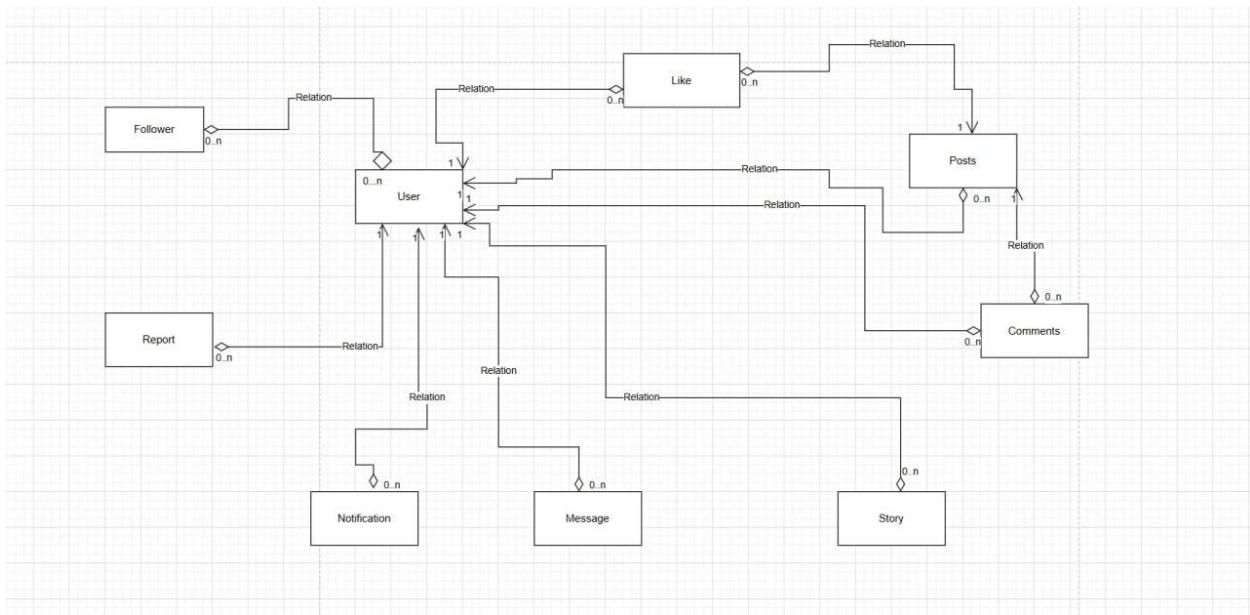
مخطط النشاط عرض قائمة المشاهدين 129



مخطط النشاط عرض قائمة المشاهدين 40

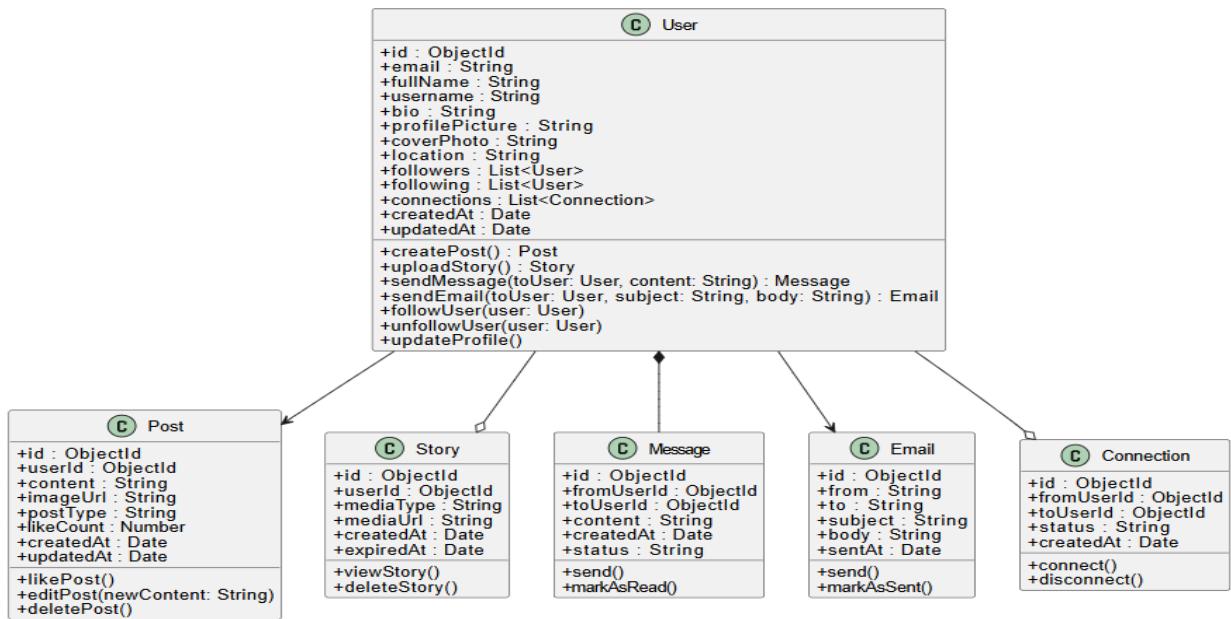
3 الفصل الثالث : الدراسة التصميمية

3.1 مخطط قاعدة البيانات ERD



ERD مخطط الـ 41

3.2 مخطط الصنوف Class Diagram



Class Diagram مخطط ال 42

3.3 جدول Test Cases

Test Case ID	Related Req ID	Test Case Description	Preconditions	Test Steps	Expected Result
TC-01	ID_01	User registration using email or phone	User not registered	Open app → Register → Enter email/phone → Submit	Account created successfully
TC-02	ID_02	Login using Google/GitHub	User has valid OAuth account	Open app → Login → Select Google/GitHub	User logged in successfully

TC-03	ID_03	Reset forgotten password	User registered	Click Forgot Password → Enter email	Password reset link sent
TC-04	ID_04	Logout from current session	User logged in	Click Logout	User session terminated
TC-05	ID_05	View user profile information	User logged in	Open Profile Page	Profile details displayed
TC-06	ID_06	Edit and update profile	User logged in	Edit profile → Save changes	Profile updated successfully
TC-07	ID_07	View other users' profiles	User logged in	Search user → Open profile	Other user profile displayed
TC-08	ID_08	Display online/offline status	Multiple users exist	View users list	Correct status shown
TC-09	ID_09	Create new post	User logged in	Create post → Publish	Post published
TC-10	ID_10	Edit or delete post	Post exists	Edit/Delete selected post	Post updated or deleted
TC-11	ID_11	View feed sorted chronologically	Multiple posts exist	Open feed	Posts ordered by time
TC-12	ID_12	Like a post	Post exists	Click Like	Like registered
TC-13	ID_13	Save or share post	Post exists	Click Save/Share	Post saved/shared
TC-14	ID_14	Report inappropriate post	Post violates policy	Click Report	Report submitted
TC-15	ID_15	Upload story (image/video)	User logged in	Upload story	Story uploaded
TC-16	ID_16	View stories from followed users	Stories available	Open stories section	Stories displayed

TC-17	ID_17	Auto delete story after 24 hours	Story older than 24h	System auto process	Story removed
TC-18	ID_18	View story viewers list	Story has viewers	Open viewers list	Viewers displayed

Test Cases جدول 20

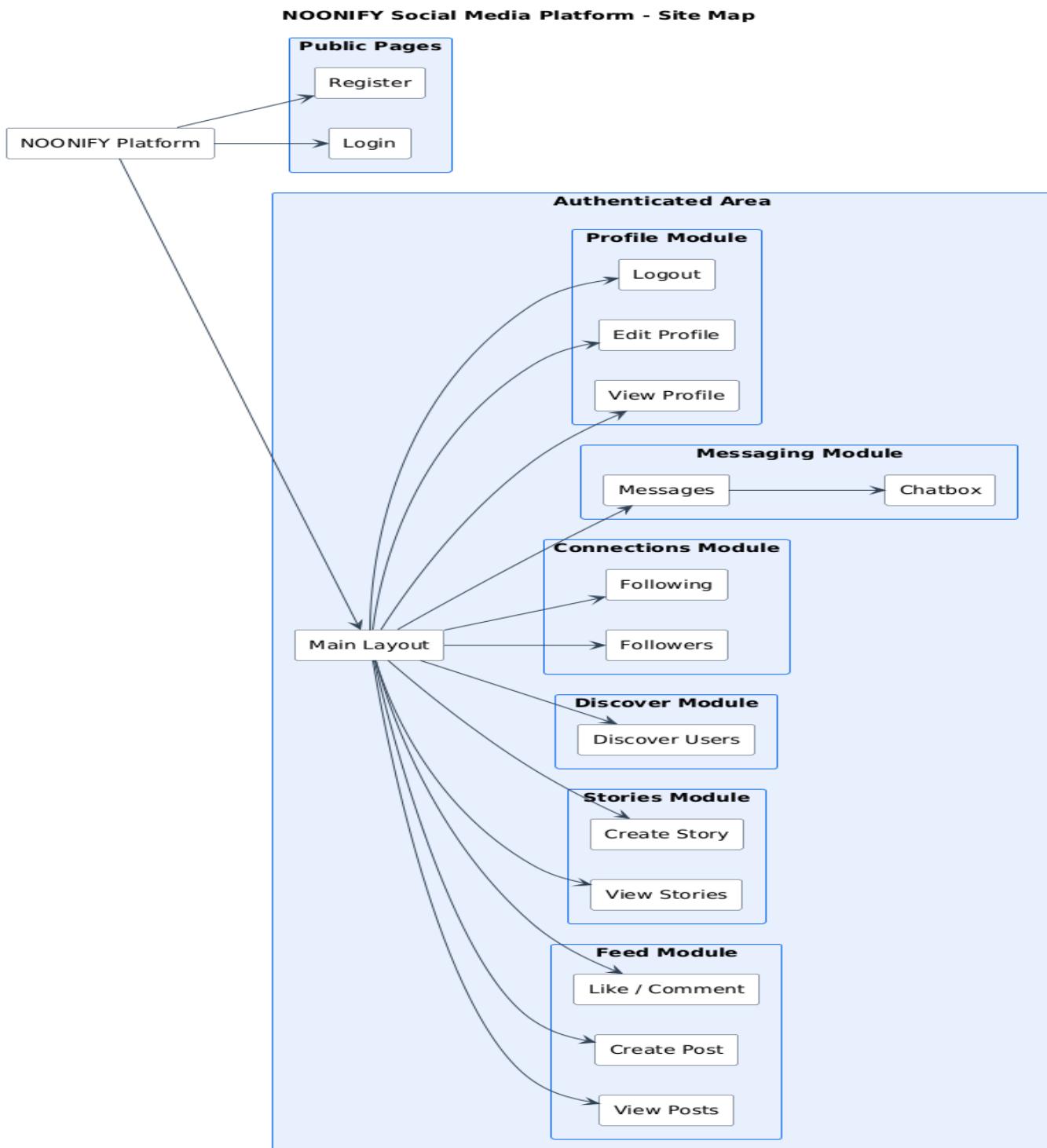
(Requirements Traceability Matrix) RTM جدول 3.4

Req ID	Title and Dependency	SRS Section	Coding Module	System Test Case
FR-01	Register New Account – depends on Auth module	3.1	auth_service, user_model	TC-01
FR-02	Login using Email/Google/GitHub – depends on Auth module	3.2	auth_service, oauth_service	TC-02
FR-03	Reset Password – depends on Auth module	3.3	password_service	TC-03
FR-04	Logout – depends on Session manager	3.4	session_manager	TC-04
FR-05	View User Profile – depends on Profile module	3.5	profile_service	TC-05
FR-06	Edit User Profile – depends on Profile module	3.6	profile_service	TC-06
FR-07	View Other Users Profiles – depends on Profile module	3.7	profile_service	TC-07
FR-08	Show Online Status – depends on Presence module	3.8	presence_service	TC-08

FR-09	Create Post – depends on Post module	3.9	post_service	TC-09
FR-10	Edit/Delete Post – depends on Post module	3.10	post_service	TC-10
FR-11	View Feed Chronologically – depends on Feed module	3.11	feed_service	TC-11
FR-12	Like Post – depends on Interaction module	3.12	interaction_service	TC-12
FR-13	Save/Share Post – depends on Interaction module	3.13	interaction_service	TC-13
FR-14	Report Post – depends on Moderation module	3.14	moderation_service	TC-14
FR-15	Upload Story – depends on Story module	3.15	story_service	TC-15
FR-16	View Stories – depends on Story module	3.16	story_service	TC-16
FR-17	Delete Story after 24h – depends on Story module	3.17	story_service, scheduler	TC-17
FR-18	View Story Viewers – depends on Story module	3.18	story_service	TC-18

جدول 21 | RTM

Site Map 3.5 مخطط الـ



Site Map الـ 43

4 الفصل الرابع : تنفيذ المشروع

4.1 الأدوات المستخدمة

يعرض هذا الفصل الأدوات والتقنيات التي تم الاعتماد عليها في تطوير منصة التواصل الاجتماعي **Noonify**، حيث تم تقسيم الأدوات إلى ثلاثة فئات رئيسية: أدوات الواجهة الأمامية (Front-End)، وأدوات قاعدة البيانات (Database)، وأدوات الواجهة الخلفية (Back-End)، وذلك بما يتوافق مع معمارية **MERN Stack**.

4.1.1 أدوات الواجهة الأمامية (Front-End Tools)

تم تطوير الواجهة الأمامية للتطبيق باستخدام مجموعة من الأدوات الحديثة التي تهدف إلى بناء واجهة مستخدم تفاعلية وسريعة الاستجابة، ومن أبرزها:

React.js •

مكتبة JavaScript تُستخدم لبناء واجهات المستخدم المعتمدة على المكونات (Component-Based Architecture)، مما يسهل إعادة استخدام الكود وتحسين قابلية الصيانة.

Vite •

أداة بناء (Build Tool) حديثة تُستخدم لتسريع عملية التطوير وتشغيل المشروع بكفاءة أعلى مقارنة بالآلات التقليدية.

JavaScript (ES6+) •

لغة البرمجة الأساسية المستخدمة في تطوير منطق الواجهة الأمامية والتعامل مع الأحداث والتفاعل مع المستخدم.

HTML5 •

تُستخدم لبناء الهيكل العام لصفحات الويب.

CSS3 •

تُستخدم لتنسيق واجهات المستخدم وتحسين التجربة البصرية.

Axios •

مكتبة تُستخدم لإرسال واستقبال طلبات HTTP بين الواجهة الأمامية والواجهة الخلفية.

ESLint •

أداة لتحليل الكود واكتشاف الأخطاء البرمجية وتحسين جودة الكود.

Vercel •

منصة تُستخدم لنشر تطبيق الواجهة الأمامية وتشغيله على بيئة إنتاجية.

4.1.2 أدوات الواجهة الخلفية(Back-End Tools)

تم بناء الواجهة الخلفية للتطبيق باستخدام بيئة تشغيل وتقنيات تضمن الأداء العالي وتنظيم منطق النظام، ومن أهم هذه الأدوات:

Node.js •

بيئة تشغيل تعتمد على JavaScript تُستخدم لتنفيذ كود الخادم (Server-Side) بكفاءة عالية.

Express.js •

إطار عمل (Framework) مبني على Node.js يُستخدم لإنشاء واجهات برمجية RESTful (Routes) وتنظيم المسارات. APIs

JavaScript •

تُستخدم لغة أساسية لتطوير منطق الخادم.

JWT (JSON Web Token) •

تُستخدم لتنفيذ نظام المصادقة (Authentication) والتحقق من هوية المستخدم.

Multer •

مكتبة تُستخدم لمعالجة ورفع الملفات (مثل الصور) من المستخدم إلى الخادم.

NodeMailer •

مكتبة تُستخدم لإرسال رسائل البريد الإلكتروني، مثل رسائل التحقق أو الإشعارات.

ImageKit •

خدمة تُستخدم لإدارة الصور ورفعها وتخزينها بشكل سحابي.

Middleware •

تُستخدم لمعالجة الطلبات الوسيطة مثل التحقق من الصلاحيات وتأمين المسارات.

4.1.3 أدوات قاعدة البيانات (Database Tools)

تم استخدام قاعدة بيانات مرنّة وقابلة للتّوسيع لتخزين بيانات المستخدمين والمحتوى، وتشمل الأدوات التالية:

MongoDB •

قاعدة بيانات NoSQL تعتمد على المستندات (Documents) وتتميز بالمرنة وسهولة التعامل مع البيانات غير المهيكلة.

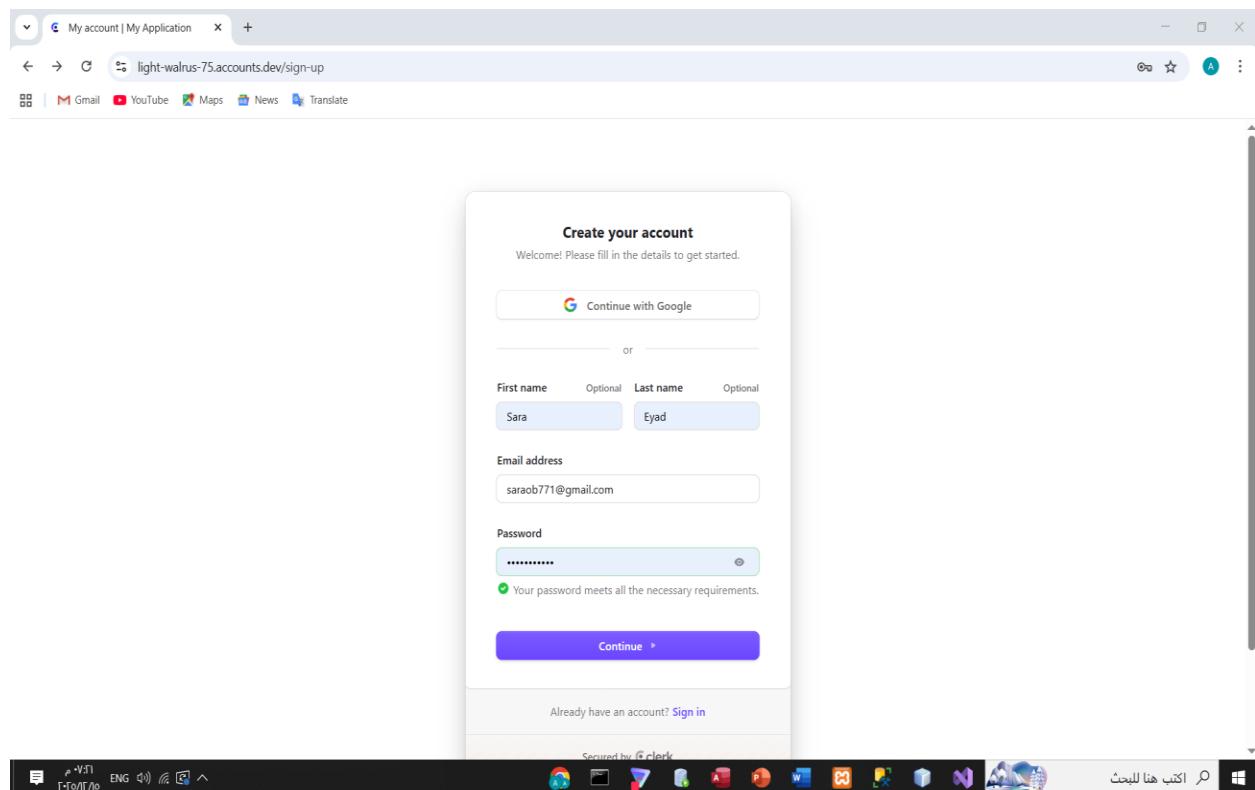
Mongoose •

مكتبة تُستخدم كنظام نمذجة (ODM) للتعامل مع MongoDB ، حيث تُسهل إنشاء النماذج وتنفيذ العمليات على البيانات (Schemas).

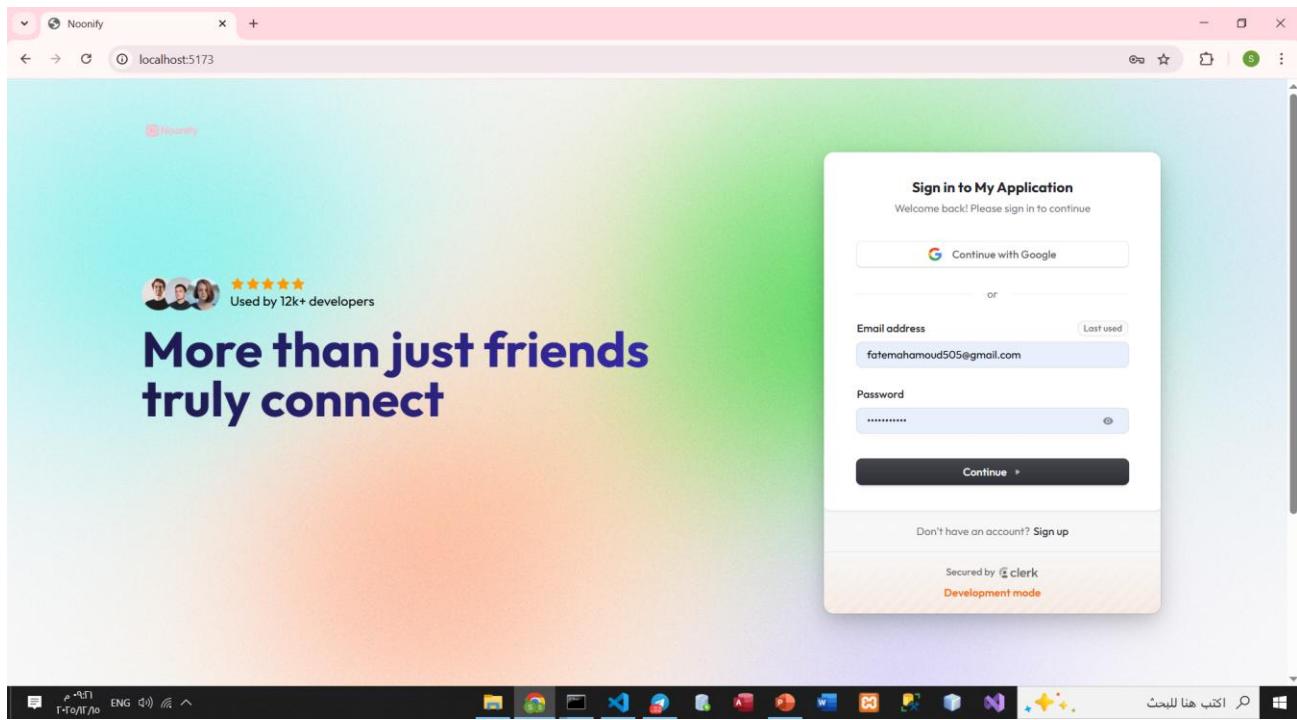
4.1.4 ملخص الأدوات المستخدمة

اعتمد المشروع على **MERN Stack** كمعمارية رئيسية، حيث تم تحقيق التكامل بين الواجهة الأمامية، الواجهة الخلفية، وقاعدة البيانات، مما ساهم في بناء منصة تواصل اجتماعي حديثة، منظمة، وقابلة للتطوير، وتلبي المتطلبات الأكاديمية والعملية للمشروع.

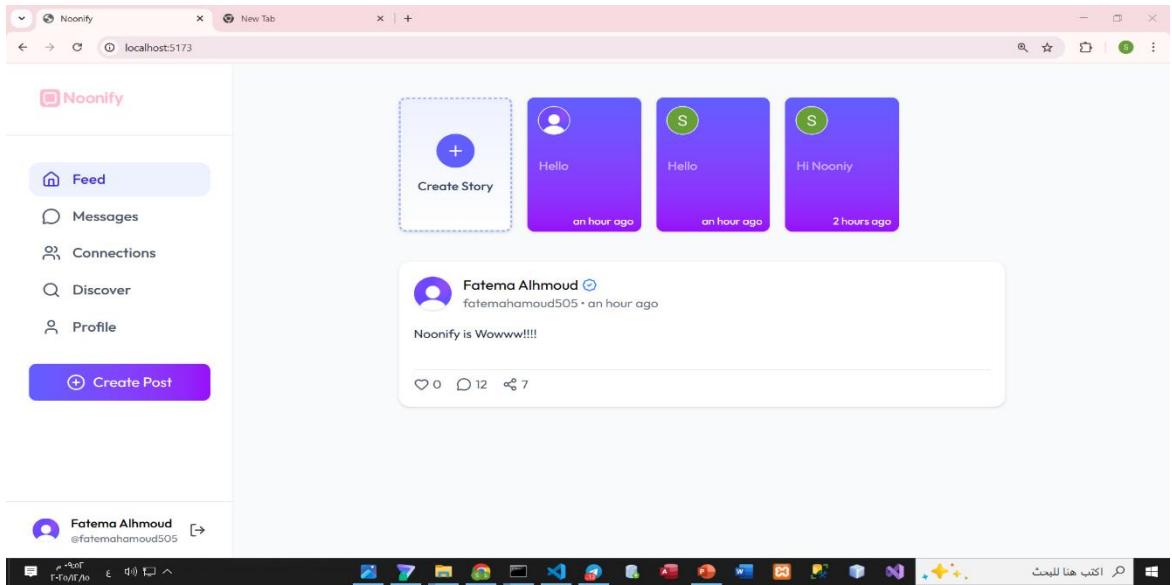
4.2 الواجهات



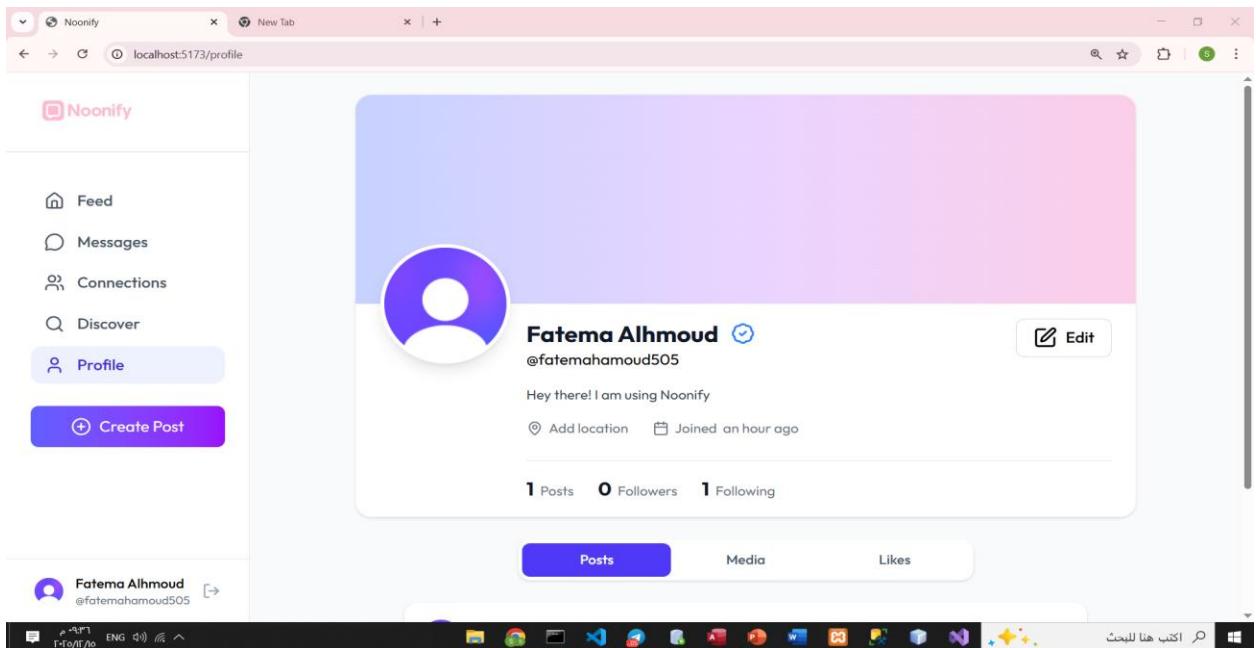
الشكل (1.1) واجهة تسجيل حساب جديد



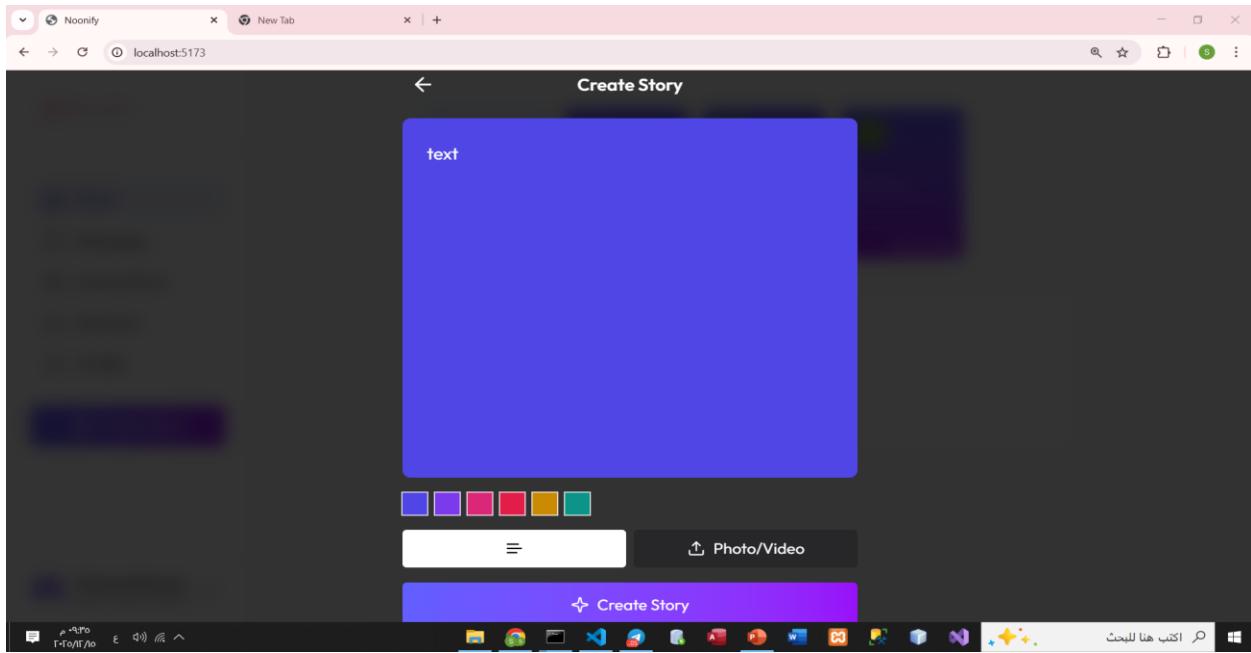
الشكل (1.2) واجهة تسجيل الدخول



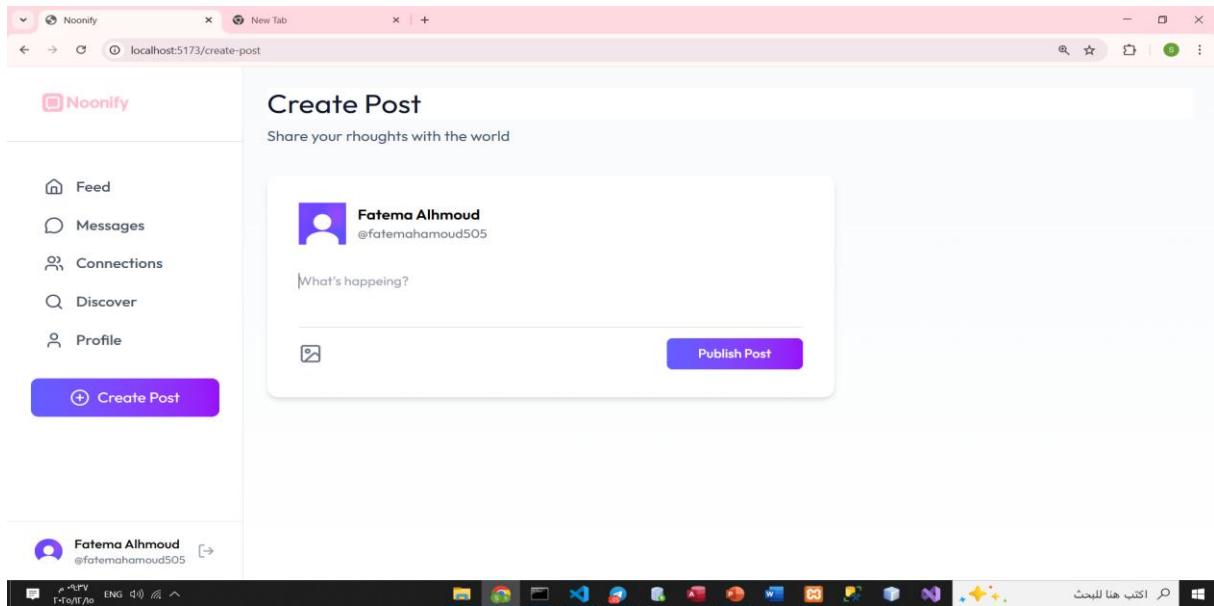
46 الشكل (1.3) زر تسجيل الخروج



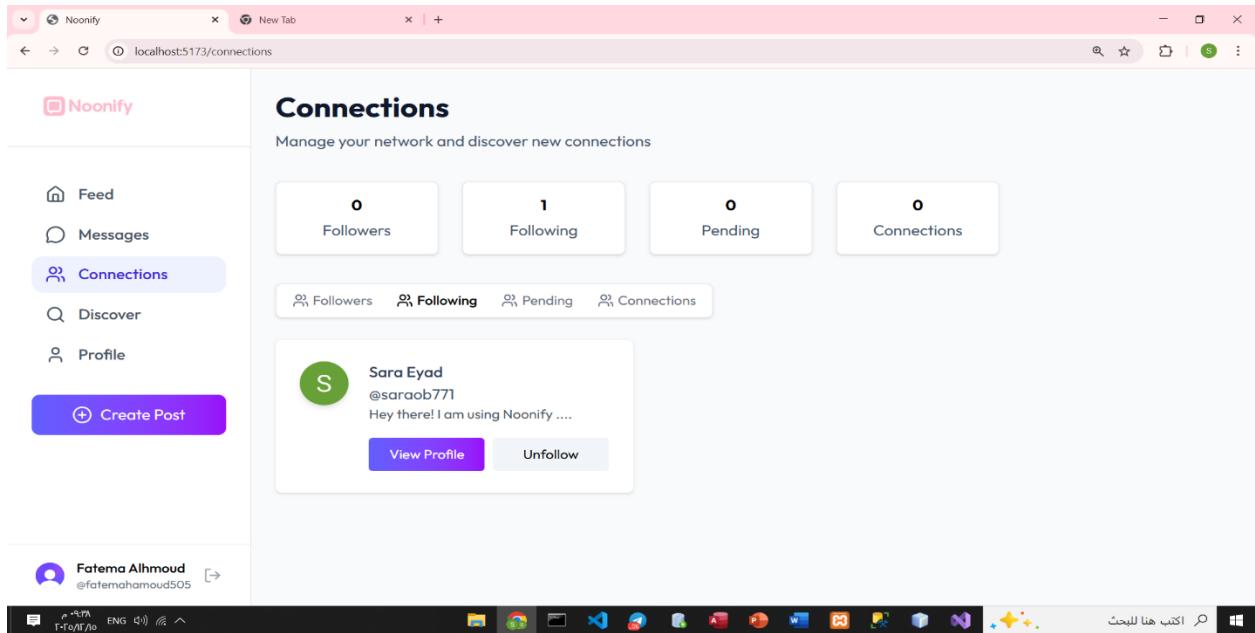
157 الشكل (1.4) عرض المعلومات الأساسية للمستخدم



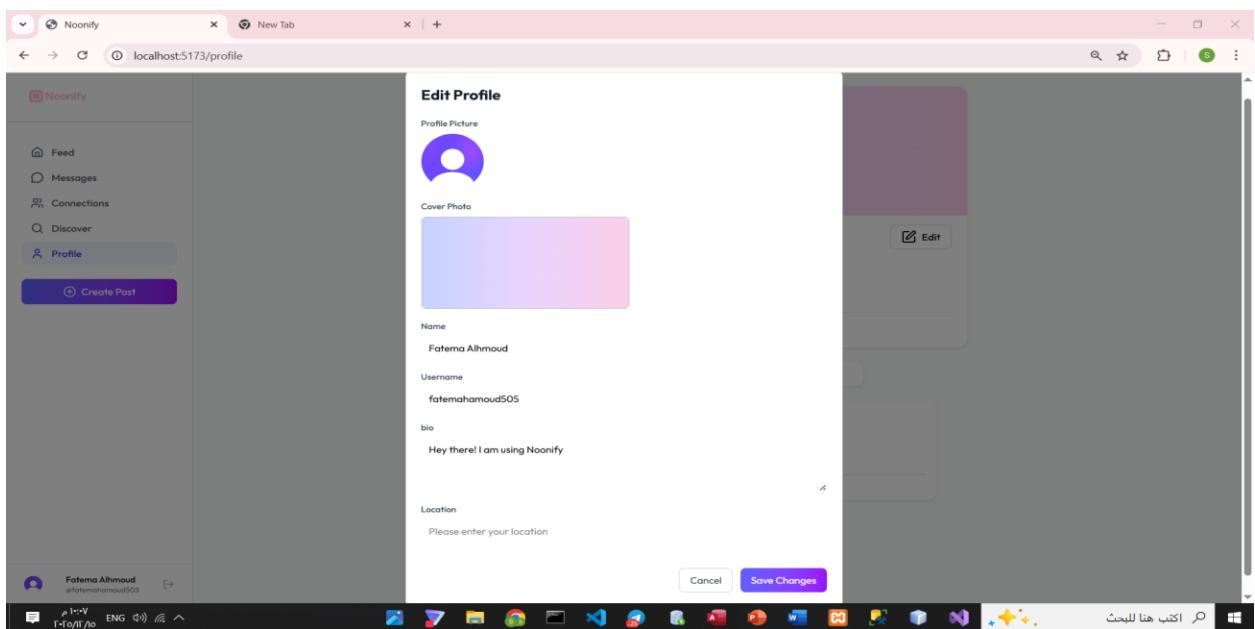
الشكل (1.5) رفع قصة جديدة



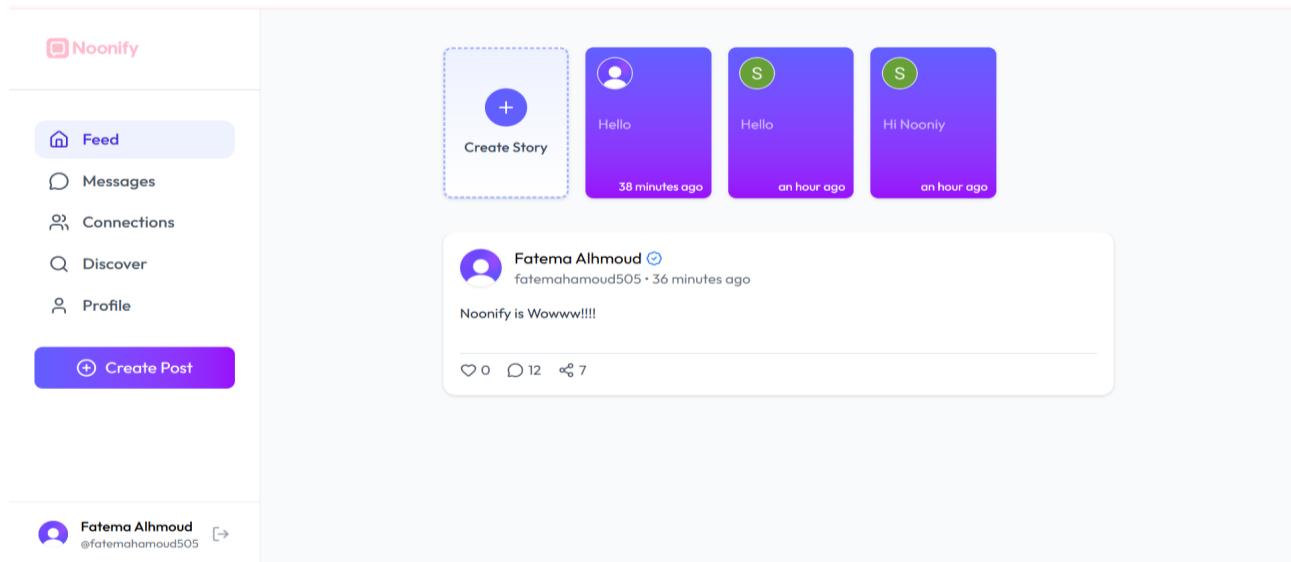
الشكل (1.6) إنشاء منشورات جديدة



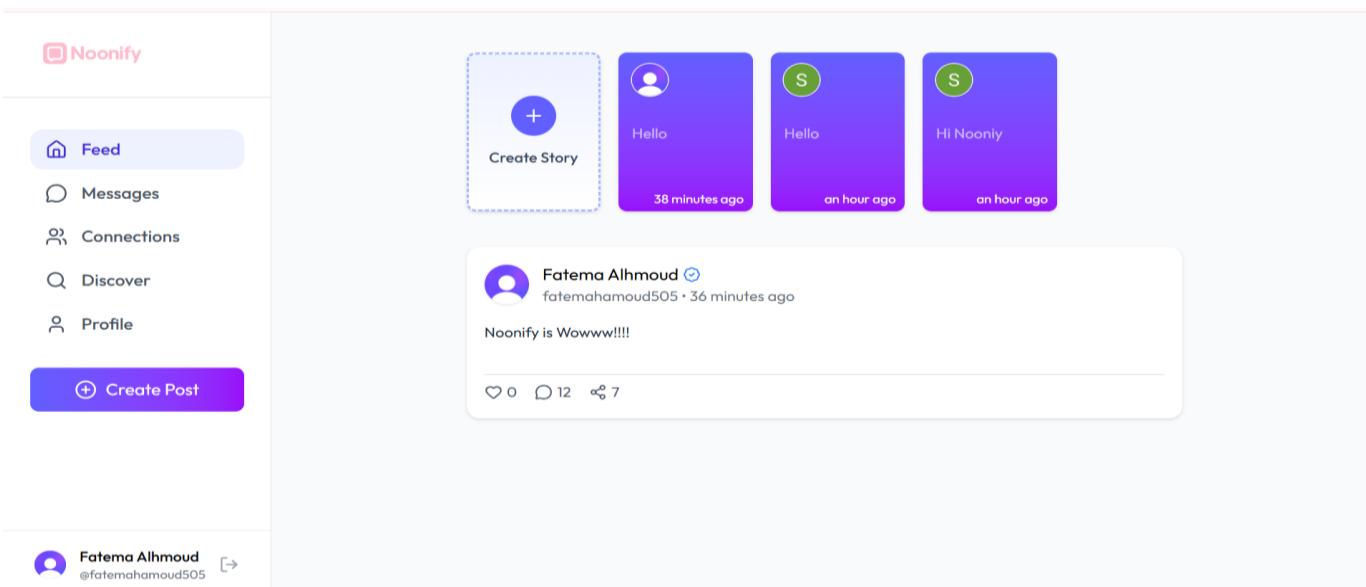
الشكل (1.7) عرض ملفات المستخدمين الآخرين 50



الشكل (1.8) تتعديل الملف الشخصي 51



الشكل (1.9) عرض القصص 5218



الشكل (1.10) عرض منشورات الآخرين 193

4.3 الخاتمة

في ختام هذا المشروع، تم تصميم وتطوير منصة تواصل اجتماعي متكاملة تحت اسم **NOONIFY**، تهدف إلى محاكاة الخصائص الأساسية لشبكات التواصل الاجتماعي الحديثة مع مراعاة الجوانب الأكاديمية والعملية في آنٍ واحد. اعتمد المشروع على معمارية **MERN Stack** التي تجمع بين الواجهة الأمامية التفاعلية باستخدام **React.js**، والواجهة الخلفية المبنية باستخدام **Express.js** و **Node.js**، إضافة إلى قاعدة البيانات **MongoDB**، مما ساهم في بناء نظام مرن، منظم، وقابل للتوسع.

نجح المشروع في تحقيق الأهداف المحددة له، حيث أتاح للمستخدمين إنشاء حساباتهم الشخصية، إدارة ملفاتهم الشخصية، مشاركة المحتوى النصي والمصري، التفاعل مع المنشورات، استخدام نظام القصص، إضافة إلى التواصل عبر الرسائل الخاصة. كما تم تطبيق مفاهيم معمارية مهمة مثل **Client-Server**، **MVC Architecture** و **Architecture** وتسهيل تطوير الميزات المستقبلية.

علاوةً على ذلك، ساهم هذا المشروع في تعزيز المهارات العملية في مجالات تطوير تطبيقات الويب الحديثة، العمل ضمن منهجية Scrum، وتحليل المتطلبات، مما جعله نموذجاً تطبيقياً يجمع بين الجانبين النظري والعملي في هندسة البرمجيات ونظم المعلومات.

4.4 الآفاق المستقبلية(Future Work)

على الرغم من تحقيق المشروع لأهدافه الأساسية، إلا أن هناك العديد من التحسينات والتوسعات التي يمكن تنفيذها مستقبلاً لرفع كفاءة المنصة وتوسيع نطاق استخدامها، ومن أبرز هذه الآفاق:

1. تعزيز الأمان

إضافة تقنيات أمان متقدمة مثل المصادقة الثنائية (Two-Factor Authentication) وتحسين سياسات التشفير لحماية بيانات المستخدمين.

2. تحسين الأداء وقابلية التوسع

دعم تقنيات التخزين المؤقت (Caching) واستخدام خدمات سحابية متقدمة لتمكين المنصة من التعامل مع عدد أكبر من المستخدمين.

3. إضافة الذكاء الاصطناعي

دمج خوارزميات التوصية لاقتراح الأصدقاء والمحتوى المناسب بناءً على اهتمامات المستخدم.

4. تطوير تطبيقات الهواتف الذكية

إنشاء تطبيقات مخصصة لأنظمة Android و iOS لتوفير تجربة استخدام أكثر سلاسة.

5. توسيع خصائص التفاعل الاجتماعي

دعم البث المباشر (Push Notifications)، الإشعارات الفورية (Live Streaming)، وتحسين نظام التعليقات والتفاعلات.

6. تحسين تجربة المستخدم (UI/UX)

إجراء دراسات استخدام حقيقية لتحسين تصميم الواجهات وجعلها أكثر سهولة وجاذبية.

References

1. Books and Articles

1. Pressman, R. S., & Maxim, B. R. (2020). *Software Engineering: A Practitioner's Approach* (9th ed.). McGraw-Hill Education.
2. Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson Education.

2. Online Resources

4. MongoDB Inc. (n.d.). *MongoDB Documentation*. Retrieved from <https://www.mongodb.com/docs/>
5. React. (n.d.). *React Official Documentation*. Retrieved from <https://react.dev/>
6. Node.js Foundation. (n.d.). *Node.js Documentation*. Retrieved from <https://nodejs.org/en/docs>
7. Express.js. (n.d.). *Express Framework Documentation*. Retrieved from <https://expressjs.com/>

3. Tools and Software Documentation

8. Vite. (n.d.). *Vite – Next Generation Frontend Tooling*. Retrieved from <https://vitejs.dev/>
9. Mongoose. (n.d.). *Mongoose ODM Documentation*. Retrieved from <https://mongoosejs.com/docs/>
10. NodeMailer. (n.d.). *NodeMailer Documentation*. Retrieved from <https://nodemailer.com/about/>

4. Additional References

11. Scrum.org. (n.d.). *Scrum Guide*. Retrieved from <https://www.scrumguides.org/>
12. Mozilla Developer Network (MDN). (n.d.). *Web Technologies Documentation*. Retrieved from <https://developer.mozilla.org/>
13. ImageKit. (n.d.). *ImageKit Media Optimization Documentation*. Retrieved from <https://imagekit.io/docs>