МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ МІСЬКОГО ГОСПОДАРСТВА імені О. М. БЕКЕТОВА
Навчально-науковий Інститут енергетичної, інформаційної та транспортної інфраструктури
Кафедра автоматизації та комп'ютерно-інтегрованих технологій

РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНА ЗАПИСКА ДО КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

на тему: «Розробка автоматизованої системи збору даних, обробки медіаконтенту та керування публікацією матеріалів у соціальних мережах»

Виконав: здобувач вищої освіти
4 курсу, групи Сінж 2022-1
напряму підготовки (спеціальності)
151 «Автоматизація та комп'ютерно-інтегровані технології»
Беззубов Ігор Олегович

Керівник: Арсеньєва О. П., д.т.н., проф.
Рецензент: Капустенко П.О., к.т.н., доц.

м. Харків – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МІСЬКОГО ГОСПОДАРСТВА імені О. М. БЕКЕТОВА

Навчально-науковий Інститут енергетичної, інформаційної та транспортної інфраструктури

Кафедра автоматизації та комп'ютерно-інтегрованих технологій

Освітньо-кваліфікаційний рівень – бакалавр

Галузі знань 17 «Електроніка, автоматизація та електронні комунікації»

Спеціальність 151 «Автоматизація та комп'ютерно-інтегровані технології»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри АКІТ
_____ БАРАНОВ О. О.
«19» __червня__2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Беззубов Ігор Олегович

(прізвище, ім'я, по батькові)

1. Тема роботи «Розробка автоматизованої системи збору даних, обробки медіаконтенту та керування публікацією матеріалів у соціальних мережах» затверджені наказом закладу вищої освіти від «22» травня 2026 р. № 440-03

керівник роботи Арсеньєва О. П., д.т.н., проф.

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

2. Термін подання студентом роботи 15.06.2026 р.

3. Вихідні дані до роботи: рекомендації щодо розробки додатку, індивідуальне завдання на розробку, дані енергоспоживання будинку та електромобіля

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз предметної області та наявних засобів автоматизації роботи з медіаконтентом у соціальних мережах. Формування вимог до системи збору даних, обробки медіаконтенту та керування публікаціями. Розробка структури системи, функціональної моделі, моделі даних і основних алгоритмів роботи. Обґрунтування вибору програмних засобів і технологій реалізації. Реалізація програмного рішення для збору даних, обробки контенту та керування публікаціями. Тестування системи, аналіз результатів і виконання розрахунків з охорони праці.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Структурна схема автоматизованої системи збору даних, обробки медіаконтенту та керування публікацією. Діаграма послідовності процесу публікації матеріалу. Логічна модель бази даних системи у нотації UML. Архітектура програмної реалізації системи. Екранні форми вебінтерфейсу з результатами тестування роботи автоматизованої системи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ I	О. П. Арсеньєва, професор		
Розділ II	О. П. Арсеньєва, професор		
Розділ III	О. П. Арсеньєва, професор		
Розділ IV	В. В. Малишева, доцент		

7. Дата видачі завдання 11.05.2026 р.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Предпроектні дослідження	12.05.2026	
2	Аналіз предметної області	14.05.2026	
3	Розробка алгоритму планування підзарядки батареї	16. 05.2026	
4	Розробка програмного забезпечення	24.05.2026	
5	Тестування і аналіз результатів роботи програмного продукту	06.05.2026	
6	Охорона праці в галузі	08. 06.2026	

Студент _____ (підпис) І. О. Беззубов (прізвище та ініціали)

Керівник роботи _____ (підпис) О. П. Арсеньєва (прізвище та ініціали)

РЕФЕРАТ

У роботі розглядається розробка автоматизованої системи збору даних, обробки медіаконтенту та керування публікацією матеріалів у соціальних мережах. Об’єктом дослідження є процес підготовки, обробки та публікації медіаконтенту в соціальних мережах. Предметом дослідження є методи автоматизованого збору статистичних даних, алгоритми обробки медіаконтенту, моделі керування публікаціями та програмні засоби для реалізації й контролю цих процесів.

Мета роботи – розробка автоматизованої системи, яка забезпечує збір та аналіз статистичних показників, підготовку медіаконтенту, планування і керування публікацією матеріалів у соціальних мережах, ведення журналу виконаних операцій та зіставлення опублікованих матеріалів із вихідними креативами.

Структура роботи представлена вступом, чотирма розділами, висновками, переліком посилань і додатками.

Структура та обсяг дипломної роботи бакалавра: випускна кваліфікаційна робота бакалавра складається зі вступу, чотирьох розділів, висновків, додатків та переліку посилань. Повний обсяг випускної кваліфікаційної роботи бакалавра становить приблизно 85 сторінок, робота містить 11 рисунків, 7 таблиць і перелік посилань із 30 найменувань.

Ключові слова: АВТОМАТИЗАЦІЯ, ЗБІР ДАНИХ, ОБРОБКА ВІДЕО, ПУБЛІКАЦІЯ КОНТЕНТУ, СОЦІАЛЬНІ МЕРЕЖІ, СКІНЧЕННИЙ АВТОМАТ, КОМП'ЮТЕРНИЙ ЗІР, ANDROID, FASTAPI

ABSTRACT

The bachelor's thesis deals with the development of an automated system for data collection, media content processing, and management of content publication on social media. The object of the study is the process of preparing, processing, and publishing media content on social media platforms. The subject of the study includes methods for automated statistical data collection, media content processing algorithms, publication management models, and software tools for implementing and monitoring these processes.

The aim of the thesis is to develop an automated system that enables the collection and analysis of statistical indicators, preparation of media content, scheduling and management of publications on social media, maintenance of an operation log, and matching of published materials with the original creative content.

The thesis consists of an introduction, four chapters, conclusions, a list of references, and appendices.

Structure and scope of the bachelor's thesis: the bachelor's qualification thesis consists of an introduction, four chapters, conclusions, appendices, and a list of references. The total length of the thesis is approximately 85 pages. It contains 11 figures, 7 tables, and a list of 30 references.

Keywords: AUTOMATION, DATA COLLECTION, VIDEO PROCESSING, CONTENT PUBLICATION, SOCIAL NETWORKS, FINITE-STATE MACHINE, COMPUTER VISION, ANDROID, FASTAPI

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

АС	– автоматизована система
БД	– база даних
ОС	– операційна система
ПЗ	– програмне забезпечення
СУБД	– система управління базами даних
ADB	– Android Debug Bridge – міст налагодження Android
API	– Application Programming Interface – програмний інтерфейс застосунку
ER	– Entity-Relationship – модель «сутність–зв’язок»
HTTP	– HyperText Transfer Protocol
JSON	– JavaScript Object Notation
OCR	– Optical Character Recognition – оптичне розпізнавання символів
REST	– Representational State Transfer
SMM	– Social Media Marketing – маркетинг у соціальних мережах
SPA	– Single Page Application – односторінковий вебзастосунок
TOTP	– Time-based One-Time Password – одноразовий пароль за часом
UI	– User Interface – інтерфейс користувача
UML	– Unified Modeling Language – уніфікована мова моделювання
VLM	– Vision-Language Model – мультимодальна модель «зір–мова»

ВСТУП

Розвиток цифрового маркетингу та зростання популярності коротких вертикальних відео сформували окрему індустрію керування присутністю брендів і авторів у соціальних мережах. За даними галузевих оглядів ринку, обсяги витрат на просування через короткий відеоконтент і кількість облікових записів, що ведуться централізовано, стабільно зростають із року в рік [25]. Аналогічні тенденції фіксують і дослідження стану маркетингу, які відзначають, що короткі відео залишаються форматом із найвищим показником залученості аудиторії, а регулярність і масштаб публікацій стають визначальними чинниками результативності кампаній [26]. Унаслідок цього перед операторами SMM постає задача одночасного супроводу десятків і сотень акаунтів, що в ручному режимі є трудомістким, повільним та схильним до помилок процесом.

Ручне ведення публікацій передбачає повторюване виконання однотипних операцій: підготовку відеоматеріалу, перенесення файлів на пристрій, послідовну навігацію інтерфейсом мобільного застосунку, введення опису та підтвердження публікації для кожного акаунта окремо. Додатковими ускладненнями є необхідність збору статистичних показників за опублікованими матеріалами, потреба в адаптації відеоконтенту під технічні вимоги платформ, а також особливості доступу до платформ при виконанні масових операцій. Усе це обґрунтовує актуальність створення автоматизованої системи, яка інтегрує збір даних, обробку медіаконтенту та керування публікацією в єдиний керований технологічний цикл.

Мета і завдання дослідження

Метою дипломної роботи є підвищення продуктивності та керованості процесів збору даних, обробки медіаконтенту і публікації матеріалів у соціальних мережах за рахунок розроблення автоматизованої системи, яка виконує ці функції через автоматизацію нативного Android-застосунку без участі оператора в рутинних операціях.

Для досягнення поставленої мети сформульовано такі завдання:

– проаналізувати предметну область, наявні підходи до автоматизації мобільних застосунків і збору статистики соціальних мереж, визначити їхні переваги та обмеження;

– сформувати функціональні та нефункціональні вимоги до системи, описати сценарії використання та логічну модель даних;

– спроєктувати багатошарову архітектуру системи, що поєднує серверну частину, веб-інтерфейс, шар автоматизації пристроїв та настільний застосунок керування;

– розробити підсистему збору даних, яка отримує метрики профілів і публікацій через публічні точки доступу з балансуванням навантаження та агрегує показники за креативами;

– розробити підсистему обробки медіаконтенту для адаптації відео під технічні вимоги платформ, транскодування у формат H.264 та формування ескізів засобами комп'ютерного зору;

– реалізувати підсистему керування публікацією матеріалів із підтримкою режимів заливу для одного акаунта, усіх акаунтів застосунку та кількох застосунків;

– формалізувати навігацію інтерфейсом у вигляді скінченного автомата з детекцією екранів та опрацюванням невідомих станів за допомогою моделі «зір-мова»;

– провести експериментальну перевірку працездатності системи та оцінити її ефективність порівняно з ручним виконанням операцій.

Об'єкт і предмет дослідження

Об'єктом дослідження є автоматизована система збору даних, обробки медіаконтенту та керування публікацією матеріалів у соціальних мережах.

Предметом дослідження є моделі, методи й технології мобільної автоматизації, обробки зображень та збору даних, що забезпечують виконання масових операцій публікації й аналітики в автоматичному режимі.

Методи дослідження

Методологічну основу роботи становлять системний аналіз предметної області та методи декомпозиції задач при формуванні вимог [3]. Для проєктування структури та поведінки системи використано об'єктно-орієнтований підхід і засоби візуального моделювання мовою UML [4]. Формалізацію навігації інтерфейсом виконано на базі теорії скінченних автоматів [5]. Для розпізнавання екранів і обробки медіаконтенту застосовано методи комп'ютерного зору, зокрема зіставлення шаблонів за нормованою крос-кореляцією [6, 7]. Програмну реалізацію виконано із застосуванням мови Python та сучасних бібліотек і фреймворків [13, 14, 16]. Перевірку результатів проведено методом експериментального тестування на наборі реальних пристроїв та облікових записів.

Практична значущість

Практична значущість роботи полягає в тому, що розроблена система дає змогу скоротити витрати часу та праці на масове ведення акаунтів за рахунок автоматизації повторюваних операцій публікації, забезпечує підготовку адаптованого відеоконтенту в єдиному технологічному циклі та надає інструменти централізованого збору й експорту аналітичних показників. Модульна багатошарова архітектура й документоорієнтоване зберігання даних з можливістю міграції на реляційну СУБД роблять систему придатною до розширення та адаптації під нові платформи й сценарії використання [24]. Запропоновані рішення можуть бути застосовані в агентствах цифрового маркетингу, а також використані як навчальний приклад інтеграції методів автоматизації, комп'ютерного зору та веброзробки.

Структура роботи

Робота складається зі вступу, основних розділів, висновків та списку використаних джерел. У вступі обґрунтовано актуальність теми, визначено мету, завдання, об'єкт і предмет дослідження. В аналітичному розділі

розглянуто предметну область, проаналізовано наявні підходи та сформульовано вимоги до системи. У розділі проєктування описано архітектуру, логічну модель даних і алгоритмічне забезпечення, зокрема скінченний автомат навігації. У розділі програмної реалізації висвітлено особливості розроблення підсистем збору даних, обробки медіаконтенту та керування публікацією, а також інтерфейсів користувача. В експериментальному розділі наведено результати випробувань і оцінку ефективності системи. У розділі охорони праці розглянуто питання безпеки життєдіяльності при роботі з обчислювальною технікою. Завершують роботу загальні висновки та перелік використаних джерел.

Обґрунтовано актуальність розроблення автоматизованої системи збору даних, обробки медіаконтенту та керування публікацією матеріалів у соціальних мережах, що зумовлена зростанням ринку короткого відеоконтенту та трудомісткістю ручного супроводу великої кількості акаунтів [25, 26]. Визначено мету роботи, що полягає в підвищенні продуктивності та керованості зазначених процесів, і сформульовано перелік завдань для її досягнення. Окреслено об'єкт і предмет дослідження, обрано методи, що поєднують системний аналіз, об'єктно-орієнтоване моделювання, теорію автоматів та методи комп'ютерного зору. Висвітлено практичну значущість запропонованих рішень і подано стислу структуру роботи, що визначає логіку подальшого викладу.

РОЗДІЛ 1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

Соціальні мережі перетворилися на основний канал просування товарів, послуг та особистих брендів, що зумовило стрімкий розвиток окремої галузі цифрового маркетингу, яку прийнято позначати абревіатурою SMM (Social Media Marketing). Згідно з аналітичними оглядами ринку, обсяги витрат на просування у соціальних медіа щороку зростають двозначними темпами, а сукупний світовий бюджет на рекламу в соціальних мережах уже перевищує позначку у кілька сотень мільярдів доларів на рік [25]. Формат коротких вертикальних відео став домінуючим способом донесення рекламного повідомлення до аудиторії: саме на нього припадає найбільша частка приросту рекламних інвестицій, оскільки він демонструє вищу швидкість поширення порівняно з традиційними банерними та текстовими форматами [25]. Дослідження стану маркетингу підтверджують, що відеоконтент демонструє найвищі показники залученості порівняно зі статичними публікаціями, а короткі ролики типу Reels та Stories забезпечують найбільше органічне охоплення за рахунок алгоритмів рекомендацій, що віддають перевагу динамічному контенту [26]. За оцінками тих самих досліджень, частка маркетологів, які регулярно використовують короткі відео у щоденних кампаніях, становить переважну більшість опитаних, а планований приріст бюджетів саме на цей формат випереджає всі інші типи контенту [26]. Така зміна споживчих звичок аудиторії формує стійкий запит на технологічні засоби, здатні підтримувати безперервний потік публікацій із прийнятними витратами праці.

Особливістю сучасного просування є те, що ефективність кампанії безпосередньо залежить від обсягу та регулярності публікацій. Алгоритми платформ заохочують облікові записи, що публікують матеріали часто й систематично, тому навіть якісний контент за відсутності регулярності втрачає охоплення. Маркетингові агенції та окремі фахівці змушені

одночасно обслуговувати десятки, а подекуди й сотні облікових записів, кожен з яких потребує власного плану публікацій, унікального оформлення матеріалів та постійного моніторингу результатів. Ручне виконання цих операцій є трудомістким, схильним до помилок та погано масштабованим: лінійне зростання кількості акаунтів зумовлює пропорційне зростання трудовитрат, що робить ручний підхід економічно невиправданим за певного порогу. Така ситуація суперечить інженерному принципу мінімізації повторюваної рутинної праці, згідно з яким операції, що піддаються формалізації, доцільно автоматизувати [1].

Предметна область, яку охоплює ця кваліфікаційна робота, об'єднує три функціонально пов'язані, але технологічно відмінні завдання. Перше завдання, збір даних, полягає в отриманні кількісних показників ефективності облікових записів та окремих публікацій: кількості переглядів, уподобань, підписників та охоплення. Без об'єктивної статистики неможливо оцінити результативність креативів і ухвалювати обґрунтовані рішення щодо подальшого наповнення акаунтів, оскільки порівняння варіантів оформлення можливе лише за наявності зіставних числових даних. Друге завдання, обробка медіаконтенту, передбачає підготовку відеоматеріалів до публікації, зокрема їх унікалізацію, що забезпечує технічну сумісність матеріалу з вимогами платформи та оригінальність кожного примірника ролика. Третє завдання, керування публікацією, охоплює власне розміщення підготовлених роликів на множині акаунтів із дотриманням заданого розкладу та режиму роботи.

Керування великою кількістю акаунтів породжує специфічний комплекс проблем. По-перше, платформи соціальних мереж застосовують механізми контролю якості та однорідності контенту, які реагують на ідентичні файли, повторювані шаблони дій та нестабільні мережеві адреси, тому матеріал потребує попередньої технічної підготовки. По-друге, офіційні програмні інтерфейси накладають обмеження на кількість запитів за одиницю часу та доступний набір операцій, через що частина потрібних дій,

зокрема публікація роликів Reels та Stories, є недоступною або суттєво обмеженою для масової автоматизації. По-третє, координація публікацій між акаунтами вимагає ведення централізованого журналу, без якого втрачається можливість зіставити опублікований матеріал із його джерелом та виміряними показниками. Сукупність цих проблем визначає потребу у спеціалізованій автоматизованій системі, яка поєднує підсистеми збору даних, обробки медіа та керування публікацією в єдиному технологічному циклі.

1.2 Опис технологічного процесу як об'єкта управління

Технологічний процес, що автоматизується, доцільно розглядати як об'єкт управління, на вхід якого надходять вихідні матеріали та керуючі впливи, а на виході формуються опубліковані матеріали та зібрані показники ефективності. Узагальнену структуру процесу наведено на рисунку (рис. 1.1), де виділено три послідовно та циклічно пов'язані стадії: збір даних, обробку медіаконтенту та керування публікацією. Поняття об'єкта управління тут застосовано у класичному сенсі теорії автоматичного керування: наявні вхідні та керуючі величини, перетворення всередині об'єкта та вихідні величини, частина яких повертається у контур ухвалення рішень як зворотний зв'язок.

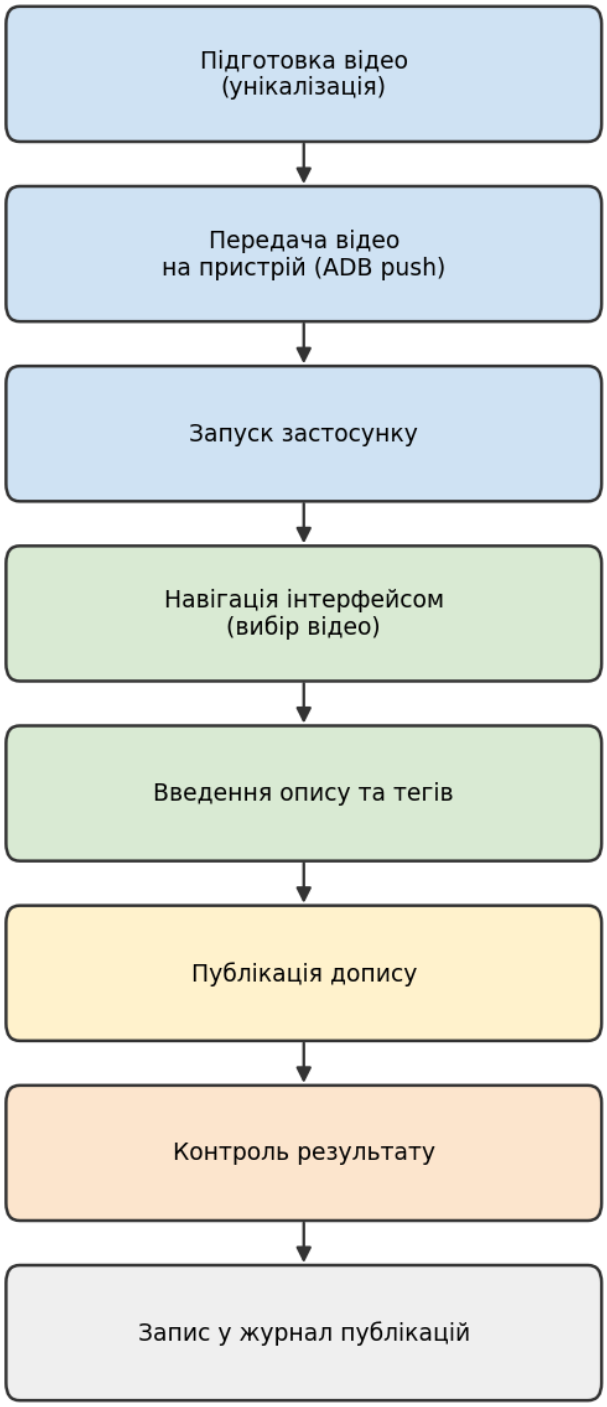

Рисунок 1.1 – Узагальнена схема технологічного процесу публікації медіаконтенту

Вхідними потоками процесу є вихідні відеоролики, списки облікових записів з реквізитами доступу, набори текстових описів, а також параметри

розкладу публікацій. Кожен із цих потоків має власну природу: відеоролики є двійковими медіафайлами значного обсягу, реквізити доступу та параметри розкладу є структурованими даними, а текстові описи становлять заготовки, з яких формується підпис до публікації. Керуючими впливами виступають команди оператора щодо запуску, призупинення та зупинення задач, налаштування режимів роботи та параметрів унікалізації; саме через ці впливи оператор спрямовує перебіг процесу, не втручаючись у кожну окрему операцію. Вихідними продуктами процесу є розміщені на платформі публікації, сформовані ескізи, а також агреговані статистичні показники, що повертаються у контур ухвалення рішень. Зворотний зв'язок реалізується через те, що зібрані показники ефективності стають підставою для коригування наступних керуючих впливів: за результатами вимірювання оператор змінює добір матеріалів, параметри оформлення або розклад, замикаючи цикл управління.

Стадія збору даних реалізується через звернення до програмного інтерфейсу платформи у режимі сесій без авторизації, що дозволяє отримувати загальнодоступні метрики профілів та роликів без прив'язки до конкретного акаунта. Для дотримання частотних обмежень інтерфейсу та рівномірного розподілу навантаження застосовується ротація проксі-серверів після обробки заданої кількості акаунтів. Отримані показники агрегуються за окремими креативами на підставі журналу публікацій, що дає змогу пов'язати кожен опублікований ролик із його джерельним відеофайлом та виміряними результатами. Результатом стадії є впорядкований набір числових показників, придатний для подальшого аналізу та експорту.

Стадія обробки медіаконтенту охоплює перетворення вихідних відеофайлів інструментами комп'ютерного зору та обробки зображень. Виконуються геометричні та колірні перетворення, зокрема поворот, дзеркалення й корекція кольору, транскодування у формат стиснення H.264, що є де-факто стандартом для відтворення відео на мобільних платформах та забезпечує сумісність із застосунками соціальних мереж, а також

формування ескізів. Ці операції водночас забезпечують технічну сумісність матеріалу з вимогами платформи та оригінальність кожного підготовленого примірника ролика [16]. На вхід стадії надходить вихідний відеофайл, на виході формуються готовий до публікації ролик та супровідний ескіз.

Стадія керування публікацією є найбільш складною з погляду автоматизації, оскільки взаємодія з платформою відбувається не через офіційний інтерфейс, а шляхом імітації дій користувача в нативному застосунку на пристрої під керуванням операційної системи Android. Передавання відеофайлу на пристрій, навігація екранами застосунку, введення опису та підтвердження публікації виконуються програмно з аналізом ієрархії елементів інтерфейсу [9]. Такий підхід дозволяє виконувати операції, недоступні через офіційний інтерфейс, проте потребує стійкості до можливих змін у компонуванні екранів застосунку. Кожна успішно завершена операція фіксується у журналі публікацій, який замикає технологічний цикл, повертаючи дані до стадії збору статистики.

1.3 Огляд наявних аналогів

На ринку засобів автоматизації роботи в соціальних мережах представлено кілька категорій рішень, кожна з яких має власну нішу застосування та притаманні обмеження. Для систематизації огляду доцільно розглянути кожну категорію окремо за такими ознаками: спосіб взаємодії з платформою, здатність працювати з множиною акаунтів, наявність засобів обробки медіа та глибина аналітики.

До першої категорії належать хмарні планувальники публікацій, серед яких найвідомішими є Hootsuite, Buffer, Later та SMMplanner. Hootsuite позиціонується як комплексна платформа керування присутністю бренду в кількох соціальних мережах одночасно, надає розвинені засоби планування, командної роботи та зведеної аналітики, проте працює виключно через офіційні інтерфейси, а отже успадковує їхні квоти та обмеження набору операцій. Buffer орієнтований на простоту планування й публікації за

розкладом і зручний для невеликих команд, але його аналітичні можливості та підтримка масових операцій із короткими відео обмежені. Later робить акцент на візуальному плануванні стрічки та роботі з графічним і відеоконтентом, проте так само спирається на офіційні канали та не передбачає масового заливу на десятки акаунтів. SMMplanner є поширеним на пострадянському ринку планувальником з підтримкою низки платформ та автопостингу, однак його можливості автоматизації обмежені рамками офіційно дозволених операцій. Спільною рисою всієї категорії є те, що вони орієнтовані переважно на легальну роботу через офіційні програмні інтерфейси платформ і тому не забезпечують повноцінного масового завантаження роликів типу Reels та Stories на велику кількість акаунтів.

До другої категорії належать настільні автоматизатори та інструменти на кшталт Postmonster, які імітують дії користувача у вебінтерфейсі або застосунку. Такі рішення дозволяють обійти частину обмежень офіційних інтерфейсів, оскільки відтворюють поведінку реального користувача, проте здебільшого зосереджені на одній платформі, не передбачають масштабованої роботи з фермою фізичних пристроїв і не містять вбудованих засобів підготовки медіаконтенту. Їхня аналітика, як правило, мінімальна або відсутня, що ускладнює оцінювання результативності окремих креативів.

Третю категорію утворюють антидетект-середовища та так звані ферми акаунтів, що забезпечують одночасне керування багатьма обліковими записами через ізоляцію цифрових профілів. Ці засоби ефективні у частині паралельної роботи з великою кількістю акаунтів, однак вони є інфраструктурним інструментом загального призначення: вони не містять вбудованих засобів обробки медіаконтенту, не виконують унікалізації відео та не надають аналітики ефективності за окремими креативами, тому потребують поєднання з іншими програмними засобами.

Окремо розглядається офіційний інтерфейс Graph API, який є технічно надійним та стабільним каналом взаємодії з платформою. Він забезпечує документований доступ до частини функцій, проте обмежений за

функціоналом, потребує верифікації застосунку та проходження процедур погодження, накладає квоти на кількість запитів і не дозволяє виконувати низку операцій, доступних лише в нативному застосунку, зокрема повноцінну публікацію окремих типів коротких відео.

Зіставлення розглянутих категорій рішень за ключовими критеріями наведено у таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика наявних аналогів

Критерій	**Hootsuite, Buffer, Later, SMMplanner**	**Postmonster**	**Антидетект, ферми**	**Graph API**	**Запропонована система**
Масовий залив Reels і Stories	обмежено	частково	так	ні	так
Робота з множиною акаунтів	частково	ні	так	частково	так
Унікалізація медіаконтенту	ні	ні	ні	ні	так
Аналітика за креативами	базова	ні	ні	базова	так
Автоматизація нативного застосунку	ні	частково	так	ні	так
Ротація проксі для збору даних	ні	ні	так	ні	так
Єдиний цикл збір, обробка, публікація	ні	ні	ні	ні	так

Кожна клітинка таблиці відображає узагальнену оцінку категорії за відповідним критерієм. Хмарні планувальники отримують позначку «обмежено» за масовим заливом коротких відео, оскільки офіційні інтерфейси не передбачають такої операції у потрібному обсязі, та «частково» за роботою з множиною акаунтів, бо керування акаунтами в них можливе, але обмежене квотами. Postmonster позначено «частково» за заливом та автоматизацією застосунку через імітацію дій користувача, проте «ні» за масштабованою роботою з багатьма акаунтами. Антидетект-середовища та ферми отримують «так» за масовими операціями та паралельною роботою з акаунтами, але «ні» за обробкою медіа й аналітикою, оскільки ці функції не входять до їхнього призначення. Graph API позначено «ні» за заливом коротких відео та автоматизацією застосунку, бо ці операції перебувають поза його дозволеним набором. Запропонована система отримує позначку «так» за всіма критеріями, оскільки за задумом поєднує функції, розподілені між різними категоріями аналогів.

Аналіз таблиці показує, що жодне з наявних рішень не поєднує в межах одного програмного продукту масовий залив контенту через нативний застосунок, вбудовану унікалізацію медіаматеріалів та аналітику ефективності за окремими креативами. Засоби автоматизації мобільних застосунків, такі як Appium та UI Automator, надають технічну базу для імітації дій користувача та аналізу ієрархії елементів інтерфейсу, проте самі по собі не є готовими SMM-рішеннями і потребують значного обсягу прикладної розробки [11]. Таким чином, виявлено незаповнену нішу: відсутній інтегрований інструмент, який охоплював би повний технологічний цикл від підготовки матеріалу до вимірювання його результативності. Саме на заповнення цієї ніші спрямована розробка, представлена в роботі.

1.4 Постановка задачі на розробку системи

На підставі аналізу предметного середовища, технологічного процесу та наявних аналогів сформульовано задачу розробки автоматизованої

системи збору даних, обробки медіаконтенту та керування публікацією матеріалів у соціальних мережах. Метою розробки є створення програмного комплексу, що автоматизує повний технологічний цикл просування короткого відеоконтенту на множині облікових записів та забезпечує об'єктивне вимірювання ефективності опублікованих матеріалів.

Для досягнення поставленої мети система має реалізовувати такі функціональні вимоги:

– підсистема збору даних повинна отримувати метрики профілів та роликів через сесії без авторизації зі зверненням до інтерфейсу платформи з ротацією проксі-серверів, агрегувати показники за креативами на основі журналу публікацій та забезпечувати експорт результатів у табличний формат для подальшого аналізу;

– підсистема обробки медіаконтенту повинна виконувати унікалізацію відео засобами повороту, дзеркалення та корекції кольору, транскодування у формат H.264 та формування ескізів інструментами комп'ютерного зору й обробки зображень, забезпечуючи технічну сумісність із вимогами платформи;

– підсистема керування публікацією повинна забезпечувати автоматизований залив роликів Reels та Stories у режимах одного акаунта, усіх акаунтів застосунку та кількох застосунків, передавання відео на пристрій, навігацію інтерфейсом, введення опису з пресетів та ведення журналу публікацій;

– система має надавати оператору вебінтерфейс із моніторингом виконання задач у реальному часі, відображенням стану кожної задачі та інструменти керування переліком пристроїв, застосунків і акаунтів.

До нефункціональних вимог віднесено такі групи. Масштабованість передбачає можливість одночасної роботи з кількома пристроями за принципом «один пристрій – одна задача», що дозволяє нарощувати продуктивність додаванням обладнання без зміни логіки системи. Надійність передбачає коректне оброблення помилок мережі та інтерфейсу застосунку, а

також можливість призупинення й зупинення задач без втрати цілісності журналу публікацій. Модульність архітектури спрощує супровід та подальший розвиток системи, оскільки підсистеми збору, обробки й публікації можуть удосконалюватися незалежно одна від одної [3]. Окремою вимогою є стійкість підсистеми навігації до змін інтерфейсу застосунку, що досягається застосуванням адаптивних механізмів розпізнавання екранів замість жорстко заданих координат. Додатково висуваються вимоги до зрозумілості інтерфейсу оператора та до продуктивності, достатньої для обслуговування цільової кількості акаунтів у прийнятні строки.

Вирішення сформульованої задачі потребує обґрунтованого вибору архітектурного підходу, методів автоматизації взаємодії з нативним застосунком, алгоритмів розпізнавання екранів та обробки медіа, а також логічної моделі даних. Ці питання послідовно розглядаються у наступних розділах роботи.

Висновки до першого розділу

У розділі виконано аналіз предметного середовища, у межах якого встановлено, що зростання ринку SMM та домінування формату коротких вертикальних відео формують стійкий запит на автоматизацію роботи з множиною облікових записів. Визначено три ключові функції предметної області: збір статистичних даних, обробку медіаконтенту та керування публікацією матеріалів, а також окреслено специфічні проблеми керування великою кількістю акаунтів, серед яких контроль однорідності контенту з боку платформ, обмеження офіційних інтерфейсів та потреба у централізованому журналі.

Технологічний процес розглянуто як об'єкт управління з визначенням вхідних потоків, керуючих впливів, вихідних продуктів та зворотного зв'язку, що дозволило виокремити три циклічно пов'язані стадії обробки. Проведений огляд наявних аналогів, узагальнений у порівняльній таблиці, показав, що наявні категорії рішень покривають лише окремі частини потрібного

функціоналу, чим підтверджено результат підрозділу 1.3 щодо наявності незаповненої ринкової ніші. На підставі здійсненого аналізу сформульовано задачу на розробку системи, визначено її мету, функціональні та нефункціональні вимоги, що становлять основу для подальшого проєктування, розглянутого у наступних розділах.

РОЗДІЛ 2 АНАЛІТИЧНИЙ РОЗДІЛ

Аналітичний розділ присвячено переходу від загального опису предметної галузі, наведеного у першому розділі, до строгої інженерної постановки задачі та обґрунтування проєктних рішень. У межах розділу формалізовано трифункціональне призначення системи, сформульовано вимоги та критерії якості, обґрунтовано вибір методів і технологій, описано структуру вхідних і вихідних даних, а також визначено методику проєктування та організацію перевірочного експерименту.

2.1 Формалізація задачі та вимоги до системи

Об'єктом автоматизації виступає сукупність рутинних операцій, що супроводжують ведення множини акаунтів у соціальних мережах коротких відео. Зважаючи на структуру теми, задачу доцільно розкладати на три взаємопов'язані функції: збір даних аналітики, обробку медіаконтенту та керування публікацією матеріалів. Кожна з функцій має власні входи, перетворення та результати, проте всі вони об'єднані спільним контуром керування й спільними сутностями даних.

Формально систему можна подати як кортеж із чотирьох складових $S = (D, A, C, P)$, де D – множина керованих пристроїв; A – множина акаунтів, розподілених за застосунками; C – множина одиниць медіаконтенту (вихідних відео та похідних від них креативів); P – множина задач публікації та збору даних, що виконуються над елементами попередніх множин. Перелічені складові пов'язані між собою спільним контуром керування: над пристроями розподіляються акаунти, до акаунтів прив'язується підготовлений медіаконтент, а задачі публікації та збору даних оперують усіма трьома попередніми множинами.

Перша функція, збір даних, визначається як зіставлення, що ставить у відповідність кожному акаунту чи опублікованому креативу набір агрегованих числових показників. Підсистема статистики звертається до публічних точок доступу анонімними сесіями, застосовує ротацію проксі

через кожні N оброблених акаунтів та накопичує метрики переглядів, уподобань, кількості підписників і охоплення. Зібрані показники зіставляються з журналом публікацій, що дає змогу агрегувати ефективність за окремими креативами та експортувати зведення у формат електронних таблиць.

Друга функція, обробка медіаконтенту, формалізується як композиція перетворень над кадрами вихідного відео. Послідовність операцій включає геометричні перетворення (поворот та дзеркалення), корекцію кольору, транскодування у кодек H.264 і формування ескізу. Метою перетворень є унікалізація матеріалу, тобто отримання похідного відео з відмінними низькорівневими характеристиками за збереження смислового змісту.

Третя функція, керування публікацією, описується як автоматизоване проходження сценарію заливу у нативному застосунку для трьох режимів роботи: один акаунт, усі акаунти одного застосунку, кілька застосунків. Сценарій охоплює передачу підготовленого відео на пристрій, навігацію інтерфейсом, введення опису з пресета та фіксацію результату у журналі заливу.

Функціональні вимоги до системи стисло формулюються так:

– забезпечення збору метрик профілів і креативів з ротацією мережевих адрес і експортом результатів;

– виконання повного циклу унікалізації та транскодування відео без участі оператора;

– автоматизований залив матеріалів формату Reels і Stories у трьох режимах із веденням журналу;

– обмеження паралельності за правилом «один пристрій – одна задача» та підтримка сигналів зупинки й паузи;

– надання веборієнтованого інтерфейсу керування з потоковим відображенням журналу подій у реальному часі.

Нефункціональні вимоги охоплюють кілька груп властивостей. Надійність передбачає коректне відновлення після збоїв окремих задач без

зупинки всієї черги та збереження цілісності журналів за аварійного завершення. Масштабованість визначає здатність нарощувати кількість одночасно задіяних пристроїв та обслуговуваних акаунтів без перепроєктування ядра системи. Супроводжуваність забезпечується модульністю архітектури та відповідністю принципам чистого коду [1], що спрощує внесення змін і локалізацію дефектів. Спостережуваність процесу виконання досягається наскрізним журналюванням подій та потоковою їх передачею оператору, завдяки чому стан кожної задачі залишається прозорим. Окремо враховано вимоги до безпеки зберігання облікових даних і секретів та до передбачуваності реакції системи на сигнали керування. Вимоги до програмного забезпечення формувалися з урахуванням рекомендованих практик специфікації [3].

Критеріями якості системи обрано: частку успішно завершених сценаріїв публікації, середній час обробки однієї одиниці контенту, пропускну здатність публікації за одиницю часу та точність детекції екранів під час навігації. Кожен з критеріїв має кількісне вираження та може бути виміряний під час перевірочного експерименту, що робить оцінювання якості об'єктивним. Пропускна здатність публікації як інтегральний показник продуктивності визначається співвідношенням:

$$W = \frac{N_d \cdot k}{t_p + t_n + t_u} \tag{2.1}$$

де W – кількість успішних публікацій за одиницю часу; N_d – кількість одночасно задіяних пристроїв; k – середня частка успішних сценаріїв $(0 < k \leq 1)$; t_p – час передавання відео на пристрій; t_n – час навігації інтерфейсом до екрана публікації; t_u – час очікування завершення вивантаження. Формула показує, що нарощування пропускної здатності досягається як збільшенням числа пристроїв, так і скороченням складових часу одного циклу.

2.2 Обґрунтування методів і технологій проєктування

Складність системи зумовлена поєднанням різнорідних задач, тому архітектурне проєктування виконано на засадах об'єктно-орієнтованого підходу з документуванням засобами уніфікованої мови моделювання [4]. Декомпозиція на класи-сутності (пристрій, застосунок, акаунт, секрет одноразових паролів, запис журналу публікації, конфігурація) забезпечує чіткий розподіл відповідальності та відповідає типовій логічній моделі предметної області. Застосування усталених шаблонів проєктування [2] спрощує реалізацію диспетчеризації задач та уніфікує взаємодію шарів.

Ключовим методом для підсистеми керування публікацією обрано апарат скінченних автоматів [5]. Навігація нативним інтерфейсом природно подається як детермінований скінченний автомат, де станами є екрани застосунку, а переходами – дії оператора. Функцію переходів описано у вигляді $\delta(s, e) = s'$, де s – поточний стан-екран, e – подія або виконана дія, s' – наступний стан. Шаблон навігації flow_template містить понад 37 відомих екранів, кожен з яких ідентифікується за характерними ознаками інтерфейсу. Такий формалізм дає змогу однозначно відстежувати позицію у сценарії, виявляти відхилення та коректно обробляти непередбачені переходи.

Детекція поточного екрана та локалізація елементів керування реалізуються методами комп'ютерного зору [6] і зіставлення шаблонів [7]. Для зіставлення застосовано нормовану крос-кореляцію, що інваріантна до рівномірної зміни яскравості та забезпечує стійке порівняння еталонного зображення елемента з областю скріншота. Рішення про наявність елемента ухвалюється за умовою перевищення порогу впевненості:

$$R(x, y) \geq \tau \tag{2.2}$$

де $R(x, y)$ – значення нормованої крос-кореляції в позиції з координатами (x, y); τ – задане порогове значення впевненості. Лише за виконання цієї умови відповідна координата вважається валідним елементом

і над нею виконується дія. Поряд із зіставленням шаблонів використовується розбір XML-дампа ієрархії інтерфейсу засобами UI Automator [9], що дозволяє знаходити елементи за ідентифікатором ресурсу, текстом або описом вмісту, а низькорівневі дії виконуються через ADB [8],[10].

Для ситуацій, коли поточний екран не зіставляється з жодним із відомих станів шаблону, передбачено резервний механізм на основі локальної мультимодальної моделі «зір–мова». Модель за скріншотом повертає рекомендовану дію разом з оцінкою впевненості; результат обов'язково валідовується, а у разі успіху новий екран додається до шаблону навігації, що поступово розширює покриття сценаріїв. Такий гібридний підхід поєднує детермінованість автоматної навігації з адаптивністю до оновлень інтерфейсу.

Серед альтернативних інструментів автоматизації розглядалися універсальні фреймворки мобільного й вебтестування [11],[12],[19]. Фреймворк Appium [11] передбачає розгортання проміжного сервера, який транслює команди до пристрою, що додає мережеву затримку та ускладнює одночасне обслуговування багатьох пристроїв. Інструмент Selenium [12] орієнтований передусім на вебсередовище і не покриває нативних жестів та передачі файлів на мобільний пристрій. Універсальні бібліотеки автоматизації введення [19] не надають прямого доступу до ієрархії елементів застосунку. З огляду на це інструменти відхилено на користь зв'язки ADB з UI Automator, яка забезпечує нижчі накладні витрати, відсутність потреби у проміжному сервері взаємодії та повний контроль над низькорівневими операціями введення й передачі файлів, що важливо за одночасного керування багатьма пристроями. Для серверної частини обрано асинхронний вебфреймворк FastAPI [14] із суворою валідацією моделей даних [15] та потоковою передачею журналів за протоколом WebSocket [22]. Засоби комп'ютерного зору й обробки зображень реалізовано бібліотеками OpenCV [16], NumPy [17] та Pillow [18]. Генерація одноразових паролів виконується за алгоритмом на основі часу [20],[23]. Настільний графічний

інтерфейс побудовано засобами Flet [21], а мова реалізації Python обрана з міркувань зрілості екосистеми та наявності потрібних бібліотек [13].

2.3 Вхідні та вихідні дані

Інформаційне забезпечення системи поділяється на три категорії: вхідні дані, які надходять ззовні та підлягають обробці; керувальні дані, що задають режими й параметри виконання; вихідні дані, які формуються як результат роботи. Розподіл даних за категоріями наведено у таблиці 2.1.

Таблиця 2.1 – Вхідні, керувальні та вихідні дані системи

Категорія	**Найменування даних**	**Джерело або призначення**	**Формат подання**
Вхідні	Вихідне відео для публікації	Завантаження оператором у відеоменеджер	Медіафайл (mp4)
Вхідні	Облікові дані акаунтів	Імпорт списком, прив'язка до застосунку	Текст, JSON-документ
Вхідні	Секрети одноразових паролів	Реєстрація акаунта	Рядок секрету, JSON
Вхідні	Метрики профілів і креативів	Публічні точки доступу через анонімні сесії	Числові показники
Керувальні	Режим заливу	Вибір оператором в інтерфейсі	Перелік режимів
Керувальні	Пресети опису та папок	Налаштування публікації	Текст, перевизначення
Керувальні	Параметри унікалізації	Конфігурація обробки відео	Числові параметри
Керувальні	Сигнали керування задачами	Диспетчер задач, інтерфейс	Команди стоп, пауза
Вихідні	Унікалізоване відео у кодеку H.264	Передача на пристрій для публікації	Медіафайл, ескіз
Вихідні	Запис журналу публікації	Фіксація результату заливу	JSON-документ ZalivLog
Вихідні	Зведена аналітика за креативами	Експорт оператору	Електронна таблиця Excel
Вихідні	Потік журналу подій	Відображення у вебінтерфейсі	Повідомлення WebSocket

Узагальнену картину руху даних між підсистемами подано на схемі інформаційних потоків (рис. 2.1). На схемі показано надходження вихідного відео й облікових даних, їх перетворення підсистемою обробки медіаконтенту, передачу підготовленого матеріалу до підсистеми керування публікацією, а також зворотний контур збору метрик, що замикається на формуванні зведеної аналітики та журналу публікацій.

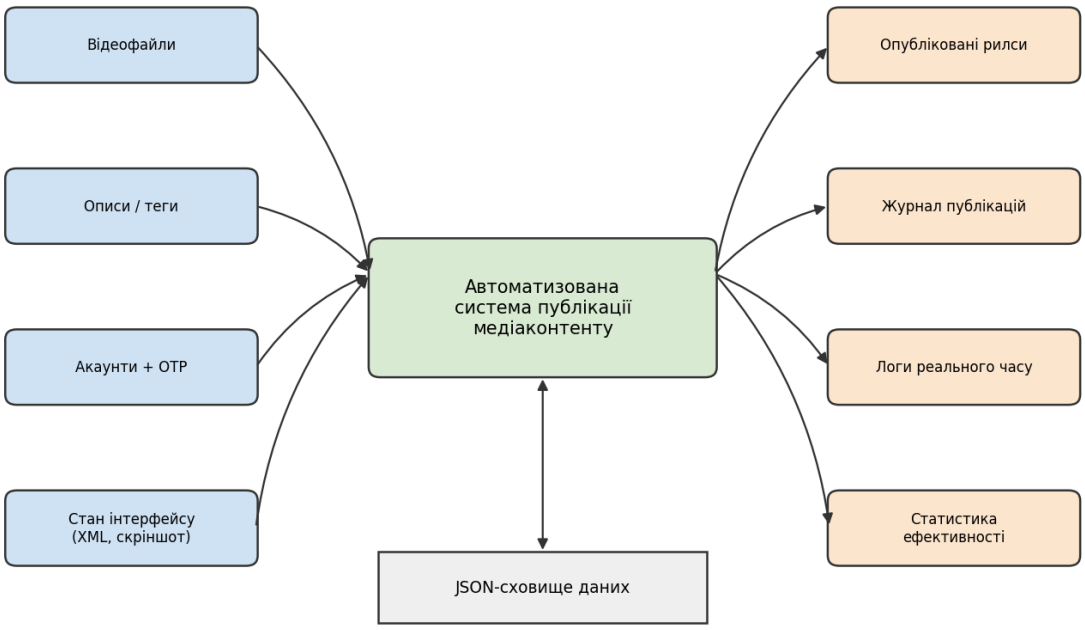

Рисунок 2.1 – Схема інформаційних потоків системи

Логічна модель даних реалізована у вигляді сукупності JSON-документів, що відповідають виокремленим сутностям. Такий спосіб зберігання обрано з огляду на гнучкість схеми та простоту інтеграції з валідаційними моделями серверної частини; за потреби передбачено можливість міграції структур на реляційну систему керування базами даних [24] без зміни прикладної логіки.

2.4 Методика проєктування і організація експерименту

Проєктування системи здійснювалося за ітеративною методикою, що поєднує висхідне уточнення вимог з низхідним архітектурним розбиттям. Послідовність основних етапів робіт наведено нижче:

- уточнення та формалізація вимог за трьома функціями, узгодження критеріїв якості;
- побудова логічної моделі даних і структурне моделювання класів-сутностей засобами UML;
- розроблення формальної моделі навігації у вигляді скінченного автомата та наповнення шаблону відомими екранами;
- реалізація підсистем збору даних, обробки медіаконтенту та керування публікацією з покомпонентним тестуванням;
- інтеграція шарів через серверні точки доступу та потокову передачу журналів;
- проведення перевірочного експерименту й аналіз отриманих показників якості.

Організація експерименту передбачає прогін типових сценаріїв на контрольованій множині пристроїв та акаунтів із фіксацією результатів кожної задачі у журналі. Перевірочний експеримент охоплює кілька груп сценаріїв: залив одного відео в один акаунт, груповий залив у всі акаунти одного застосунку та залив у кількох застосунках одночасно. Для кожної групи фіксуються вимірювані показники: час передачі відео на пристрій, час навігації до екрана публікації, час очікування завершення вивантаження та підсумковий стан задачі у журналі. Для оцінювання навігації окремо вимірюється точність детекції екранів як частка коректно розпізнаних станів від загальної кількості переходів, а також частота звернень до резервного механізму на основі мультимодальної моделі.

Основним критерієм успішності роботи системи прийнято частку успішно завершених сценаріїв публікації, яка обчислюється за співвідношенням:

$$k = \frac{N_{succ}}{N_{total}}, \tag{2.3}$$

де k – частка успішно виконаних сценаріїв; N_{succ} – кількість сценаріїв, завершених із позитивним результатом і відповідним записом у журналі; N_{total} – загальна кількість сценаріїв, запущених протягом експериментального періоду.

Поряд з основним критерієм аналізуються середній час циклу публікації, пропускна здатність за формулою (2.1) та стабільність роботи диспетчера задач за тривалого навантаження. Сукупність зазначених показників дає змогу об'єктивно оцінити ступінь досягнення поставлених у підрозділі 2.1 цілей автоматизації.

2.5 Проєктування системи

Проєктування системи виконано на основі вимог, сформованих у попередніх підрозділах, із застосуванням об’єктно-орієнтованого підходу та принципів модульної декомпозиції [1]. Головним проєктним рішенням є розподіл функціональності на чотири незалежні архітектурні шари, кожен з яких має чітко окреслену зону відповідальності та взаємодіє із суміжними шарами через формалізовані інтерфейси. Такий підхід забезпечує слабке зв’язування компонентів, спрощує тестування та дозволяє замінювати окремі реалізації без перепроєктування всієї системи.

Перший шар – серверний (Backend) – реалізовано засобами Python 3 та фреймворку FastAPI, який надає програмний інтерфейс у стилі REST і налічує понад 50 ендпоінтів [14]. Для трансляції журналів виконання задач у реальному часі застосовано протокол WebSocket згідно з RFC 6455 [22], що дає змогу клієнту отримувати потік повідомлень без періодичного опитування сервера. Паралельне виконання задач організовано через пул потоків ThreadPoolExecutor та керівний компонент TaskManager, який реалізує обмеження «один пристрій – одна задача» та обробляє сигнали

зупинення й призупинення. Взаємодія серверного шару з клієнтським відбувається двома каналами: синхронні запити обслуговуються через REST-ендпоінти, тоді як асинхронні повідомлення про стан задач надходять через постійне з’єднання WebSocket. Така роздільність каналів дозволяє розмежувати керівні команди та потокові дані, що спрощує обробку помилок і відновлення з’єднання.

Другий шар – клієнтський (Frontend) – побудовано як односторінковий застосунок (SPA) на чистому JavaScript стандарту ES6 із хеш-маршрутизацією. Інтерфейс охоплює одинадцять функціональних сторінок: дашборд, залив, сторіс, застосунки, акаунти, додавання акаунтів, аватарки, авторег, статистику, відеоменеджер та налаштування. Повторно використовувані елементи (вибір пристрою, вибір застосунків, перегляд логів, модальні вікна, сповіщення) винесено в окремі компоненти, що відповідає принципу єдиної відповідальності [1]. Кожна функціональна сторінка звертається до серверного шару виключно через визначений набір REST-викликів, а оновлення стану в реальному часі забезпечується підпискою на канал WebSocket, що зберігає незалежність клієнтської частини від внутрішньої реалізації сервера.

Третій шар – шар автоматизації – відповідає за безпосередню взаємодію з мобільним пристроєм. Його основу складають інструменти ADB та UI Automator: система формує XML-дамп ієрархії елементів інтерфейсу, виконує пошук вузлів за атрибутами resource-id, text та content-desc і генерує події дотику. Для розпізнавання графічних елементів, що не мають стабільних ідентифікаторів, застосовано зіставлення зразків (template matching) засобами OpenCV. У ситуаціях, коли екран не вдається ідентифікувати наявними правилами, керування передається локальній мультимодальній моделі «зір–мова». Інтерфейс цього шару щодо серверного зведено до набору команд автоматизації та зворотного потоку журналів виконання, завдяки чому конкретні засоби взаємодії з пристроєм можуть бути замінені без впливу на решту системи.

Четвертий шар – настільний графічний інтерфейс (Desktop GUI) – реалізовано засобами фреймворку Flet і призначено для локального керування системою без використання браузера. Цей шар використовує ті самі серверні інтерфейси, що й клієнтський SPA, що уникає дублювання логіки керування.

Функціональні можливості системи з погляду користувача систематизовано у вигляді діаграми варіантів використання (рис. 2.2). На діаграмі виокремлено основного актора (оператора SMM) та зовнішні системи, з якими взаємодіє система: мобільні пристрої, програмні інтерфейси соціальних мереж та проксі-сервери. До ключових варіантів використання віднесено збір статистики, обробку медіаконтенту, заливання Reels і Stories, керування акаунтами та налаштування системи.

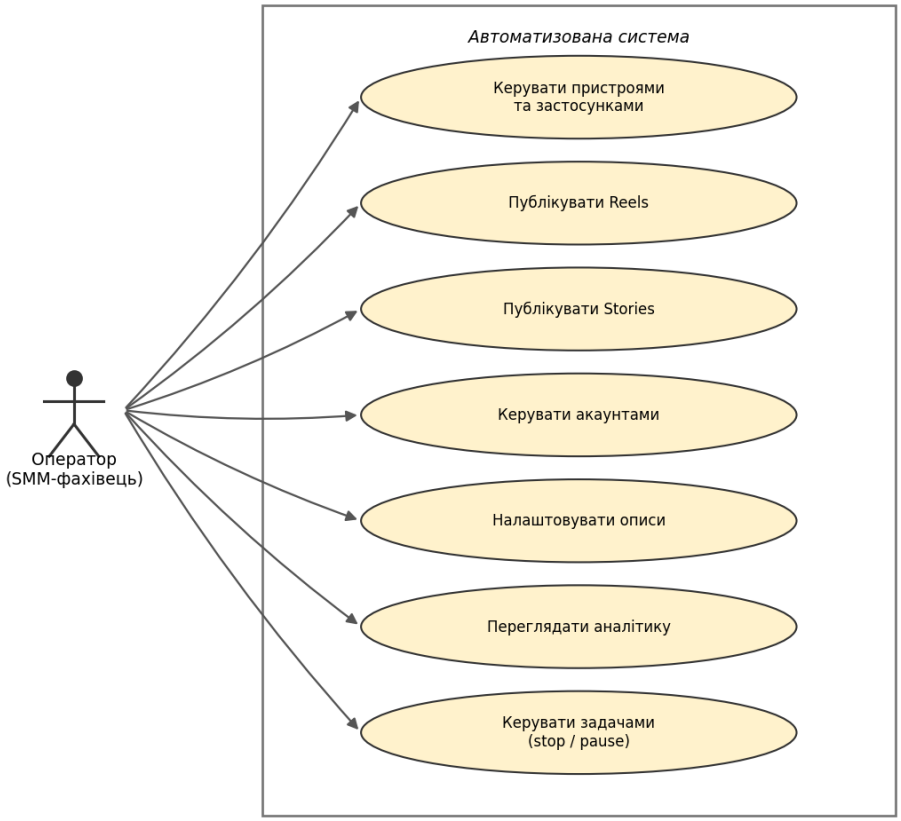

Рисунок 2.2 – Діаграма варіантів використання системи

Структурну організацію системи та зв’язки між її шарами відображає структурна схема (рис. 2.3). Схема демонструє висхідний потік запитів від клієнтських застосунків до серверного шару та низхідний потік команд автоматизації до мобільних пристроїв, а також зворотний канал передавання журналів через WebSocket.

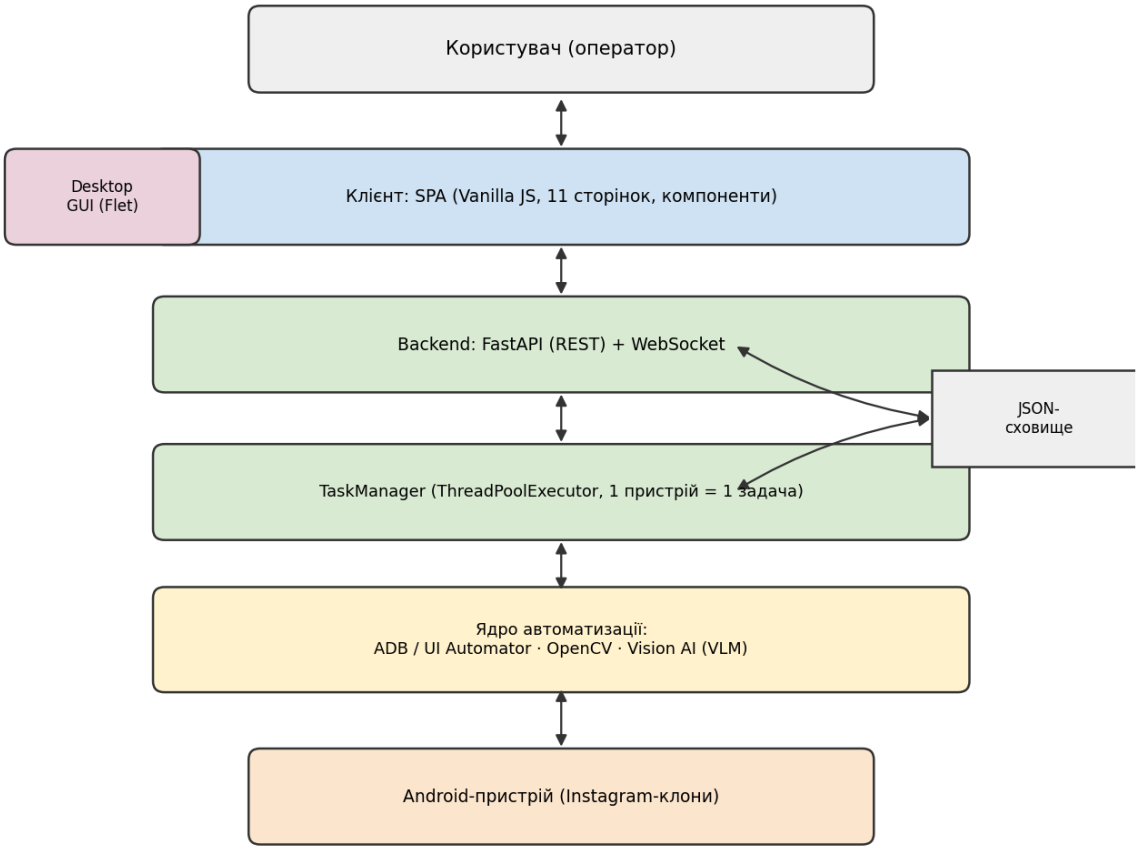

Рисунок 2.3 – Структурна схема автоматизованої системи

Для деталізації найбільш відповідального сценарію – публікації медіаматеріалу – розроблено діаграму послідовності (рис. 2.4). На ній відображено обмін повідомленнями між клієнтом, серверним шаром, компонентом TaskManager, шаром автоматизації та мобільним пристроєм. Сценарій охоплює надсилання запиту на залив, постановку задачі в чергу, передавання відеофайлу на пристрій командою ADB push до каталогу DCIM, послідовну навігацію інтерфейсом нативного застосунку, введення опису з пресета та фіксацію результату в журналі публікацій ZalivLog.

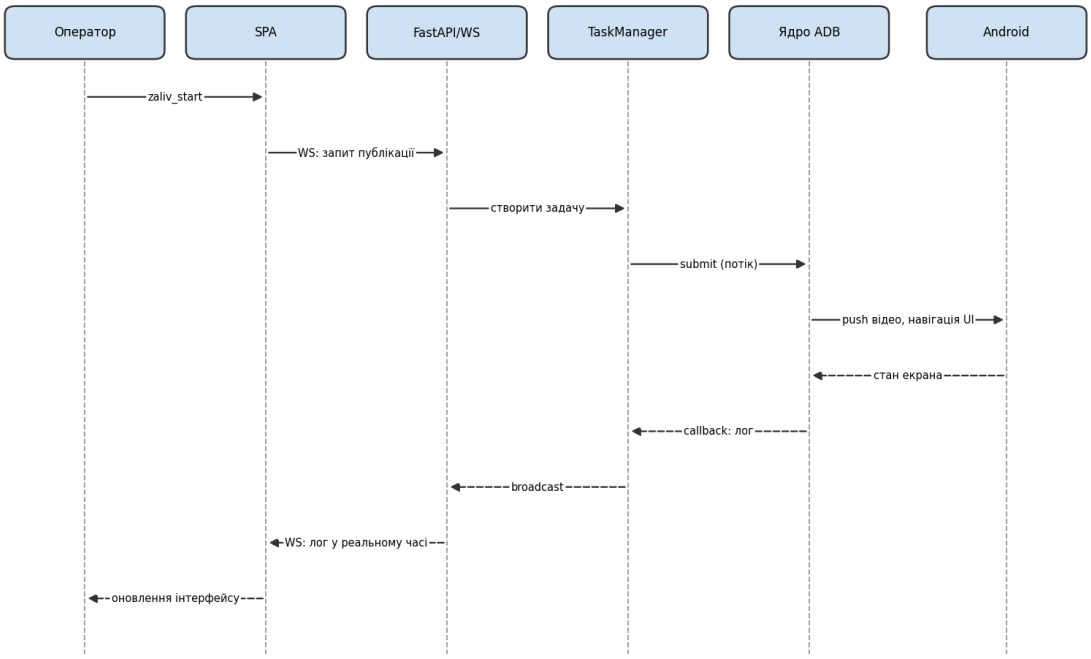

Рисунок 2.4 – Діаграма послідовності виконання публікації

При проєктуванні взаємодії компонентів застосовано низку перевірених шаблонів проєктування [2]. Зокрема, для уніфікації доступу до різних режимів заливання використано шаблон «Стратегія», для централізованого створення об’єктів задач – шаблон «Фабричний метод», а для оповіщення клієнтів про зміну стану задач – шаблон «Спостерігач», природно реалізований поверх каналу WebSocket. Уся проєктна документація виконана з використанням уніфікованої мови моделювання UML [4], що забезпечує однозначність трактування діаграм та узгодженість між етапами розроблення.

2.6 Проєктування бази даних

Проєктування сховища даних виконано з урахуванням специфіки інформаційних об’єктів системи, які мають переважно документну природу та змінний набір атрибутів. Логічну модель даних побудовано згідно з принципами реляційного проєктування [24], після чого її адаптовано до фізичної реалізації у вигляді документного JSON-сховища.

Логічну модель даних подано у нотації UML як діаграму класів із зазначенням типів даних кожного атрибута (рис. 2.5). Вибір нотації UML обумовлено потребою в єдиному формалізмі для всіх проєктних артефактів та можливістю однозначно відобразити як склад атрибутів сутностей, так і кратність зв’язків між ними [4].

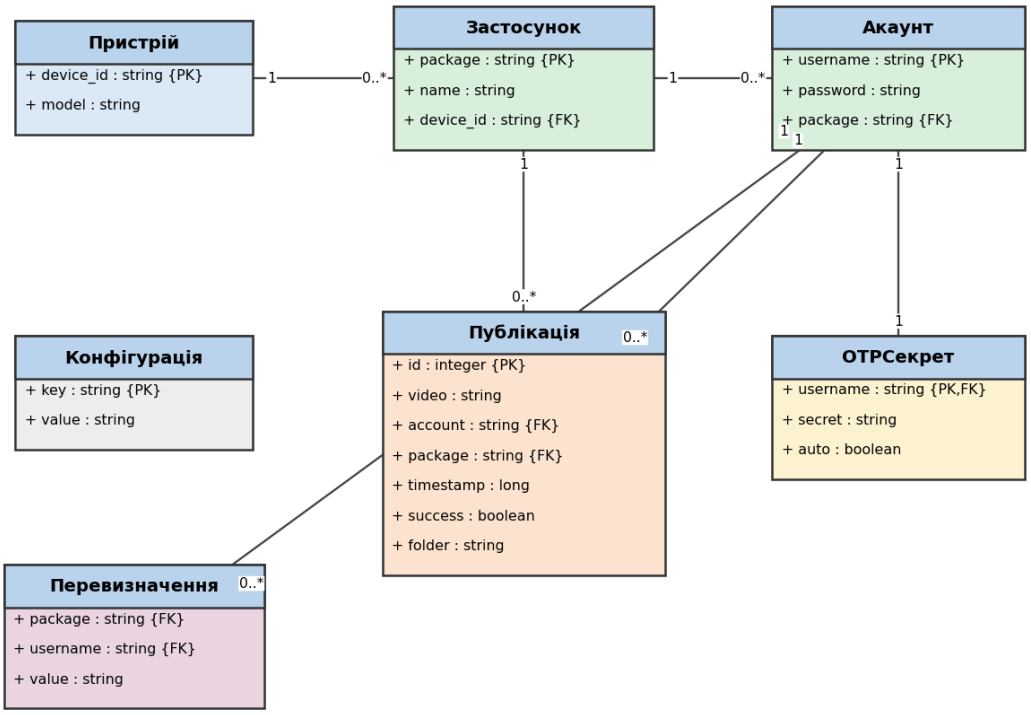

Рисунок 2.5 – Модель даних системи у нотації UML (діаграма класів)

Модель даних описано сімома основними сутностями. Сутність Пристрій представляє фізичний або емульований мобільний пристрій, ідентифікований унікальним серійним номером. Сутність Застосунок описує примірник встановленого на пристрої клієнта соціальної мережі. Сутність Акаунт містить облікові дані користувача в межах конкретного застосунку. Сутність ОТРСекрет зберігає секретний ключ для генерації одноразових паролів двофакторної автентифікації. Сутність Публікація (журнал ZalivLog) фіксує кожну спробу заливання матеріалу. Сутність Конфігурація реалізує сховище налаштувань у форматі «ключ–значення». Сутність Перевизначення

задає індивідуальні значення опису або робочої папки для пари «застосунок–акаунт». Склад сутностей та їх атрибутів наведено у таблиці 2.2.

Таблиця 2.2 – Сутності та атрибути моделі даних

Сутність	**Атрибут**	**Тип даних**	**Призначення**
1	2	3	4
Пристрій	device_id	string	Серійний номер пристрою (первинний ключ)
Пристрій	model	string	Модель пристрою
Застосунок	package	string	Ім’я пакета (первинний ключ)
Застосунок	name	string	Відображувана назва застосунку
Застосунок	device_id	string	Посилання на пристрій (зовнішній ключ)
Акаунт	username	string	Ім’я користувача (первинний ключ у межах пакета)
Акаунт	password	string	Пароль облікового запису
Акаунт	package	string	Посилання на застосунок (зовнішній ключ)
OTPСекрет	username	string	Посилання на акаунт (зовнішній ключ)
OTPСекрет	secret	string	Секретний ключ TOTP
OTPСекрет	auto	boolean	Ознака автоматичної генерації коду

Продовження таблиці 2.2

1	2	3	4
Публікація	id	integer	Ідентифікатор запису (первинний ключ)
Публікація	video	string	Шлях до вихідного відеофайлу
Публікація	account	string	Ім’я акаунта публікації
Публікація	package	string	Пакет застосунку
Публікація	timestamp	datetime	Момент публікації
Публікація	success	boolean	Ознака успішності операції
Публікація	folder	string	Робоча папка джерела матеріалу
Конфігурація	key	string	Ключ параметра (первинний ключ)
Конфігурація	value	string	Значення параметра
Перевизначення	package	string	Пакет застосунку (частина складеного ключа)
Перевизначення	username	string	Ім’я акаунта (частина складеного ключа)
Перевизначення	value	string	Значення перевизначеного параметра

Між сутностями встановлено такі зв’язки. Пристрій пов’язаний із Застосунком зв’язком «один до багатьох», оскільки на одному пристрої може бути встановлено кілька застосунків. Застосунок пов’язаний з Акаунтом зв’язком «один до багатьох». Акаунт пов’язаний з ОТРСекретом зв’язком «один до одного», бо кожному обліковому запису відповідає не більше

одного секрету двофакторної автентифікації. Акаунт пов’язаний з Публікацією зв’язком «один до багатьох», адже з одного акаунта може бути виконано багато заливань. Сутності Конфігурація та Перевизначення є слабко зв’язаними і використовуються як допоміжні словники параметрів. Описані кратності зв’язків формують обмеження цілісності: видалення Застосунку має узгоджено зачіпати залежні Акаунти, а наявність запису ОТРСекрет можлива лише за наявності відповідного Акаунта.

Логічну модель приведено до третьої нормальної форми [24]. Кожен неключовий атрибут функціонально залежить лише від первинного ключа своєї сутності, транзитивні залежності усунено винесенням характеристик пристрою та застосунку в окремі сутності, а повторюваних груп атрибутів модель не містить. Цілісність даних забезпечено на двох рівнях: сутнісна цілісність гарантується унікальністю первинних ключів, а посилальна цілісність – узгодженістю значень зовнішніх ключів (наприклад, поле package у сутності Акаунт має відповідати наявному запису сутності Застосунок).

Фізична реалізація сховища виконана у вигляді набору JSON-документів, що зберігаються у файловій системі сервера. Такий підхід обрано через невеликий обсяг даних, документну природу записів та потребу в простоті розгортання без встановлення окремої СУБД. Водночас логічна модель спроєктована так, щоб у разі зростання навантаження забезпечити безболісну міграцію на реляційну або документоорієнтовану систему керування базами даних: кожна сутність відповідає окремій таблиці чи колекції, а описані зв’язки реалізуються засобами зовнішніх ключів або вкладених документів [24].

2.7 Математичне та алгоритмічне забезпечення

Функціонування системи спирається на кілька математичних моделей та алгоритмів, які забезпечують розпізнавання елементів інтерфейсу,

формальне керування навігацією та генерацію кодів автентифікації. У цьому підрозділі наведено їх формальний опис.

Розпізнавання графічних елементів, що не мають стабільних програмних ідентифікаторів, виконується методом зіставлення зразків (template matching) із застосуванням нормованої крос-кореляції [6, 7]. Нехай I – зображення-сцена, тобто скріншот екрана, а T – зразок шуканого елемента розміром $w \times h$. Коефіцієнт нормованої крос-кореляції R у точці з координатами (x, y) обчислюється за формулою [16]:

$$R(x,y) = \frac{\sum_{x'=0}^{w-1} \sum_{y'=0}^{h-1} T(x', y') I(x+x', y+y')}{\sqrt{\left(\sum_{x'=0}^{w-1} \sum_{y'=0}^{h-1} T^2(x', y')\right)\left(\sum_{x'=0}^{w-1} \sum_{y'=0}^{h-1} I^2(x+x', y+y')\right)}}, \tag{2.4}$$

де підсумовування здійснюється за всіма пікселями зразка з локальними координатами (x', y'), що задовольняють умови $0 \le x' < w$ та $0 \le y' < h$. Значення коефіцієнта $R(x, y)$ належить інтервалу $[-1; 1]$. Елемент вважається знайденим, якщо максимальне значення $R(x, y)$ перевищує наперед заданий поріг впевненості τ. Нормування у знаменнику зменшує вплив рівномірних змін інтенсивності зображення, що є важливим під час роботи з різними темами оформлення застосунку.

Навігацію інтерфейсом нативного застосунку формалізовано у вигляді скінченного автомата [5]. Автомат подано кортежем

$$A = (S, E, \delta, s_0, F), \delta(s, e) = s', \tag{2.5}$$

де S – скінченна множина станів, що відповідають відомим екранам застосунку; у шаблоні flow_template описано понад 37 екранів; E – скінченна множина подій, які відповідають діям оператора або переходам інтерфейсу;

δ – функція переходів; $s_0 \in S$ – початковий стан; $F \subseteq S$ – множина завершальних станів, що відповідають успішній публікації матеріалу.

Функція переходів $\delta(s, e) = s'$ визначає стан, до якого переходить автомат у разі настання події e у поточному стані s, де s – поточний стан; e – подія; s' – наступний стан. Така формалізація дає змогу строго описати очікувану поведінку системи, виявляти неприпустимі переходи та забезпечувати відновлення після збоїв шляхом повернення до відомого стану.

Алгоритм детекції поточного екрана працює у два етапи. На першому етапі система формує XML-дамп ієрархії елементів та порівнює набір характерних ознак екрана (наявність вузлів із заданими resource-id, text або content-desc, а також результати зіставлення зразків) з описами станів шаблону. Якщо знайдено стан, ознаки якого виконуються, екран вважається ідентифікованим, і автомат застосовує відповідний перехід. На другому етапі, якщо жоден відомий стан не розпізнано, активується резервний механізм: скріншот екрана передається локальній мультимодальній моделі «зір–мова», яка повертає рекомендовану дію разом з оцінкою впевненості. Запропонована дія валідовується перед виконанням за двома критеріями: відповідність типу дії наявним елементам інтерфейсу та перевищення оцінкою впевненості заданого порога. Лише за умови проходження валідації дія виконується, а у разі успіху новий екран разом з його ознаками додається до шаблону flow_template, завдяки чому система поступово розширює власну базу відомих станів. Двоетапна побудова алгоритму поєднує детермінованість автоматної навігації з гнучкістю резервного розпізнавання у непередбачених ситуаціях.

Послідовність зміни станів автомата та умови переходів між ними відображено на діаграмі станів UML (рис. 2.6). Діаграма ілюструє типовий маршрут публікації, від початкового екрана застосунку до завершального стану, а також альтернативні гілки, що активуються резервним механізмом розпізнавання.

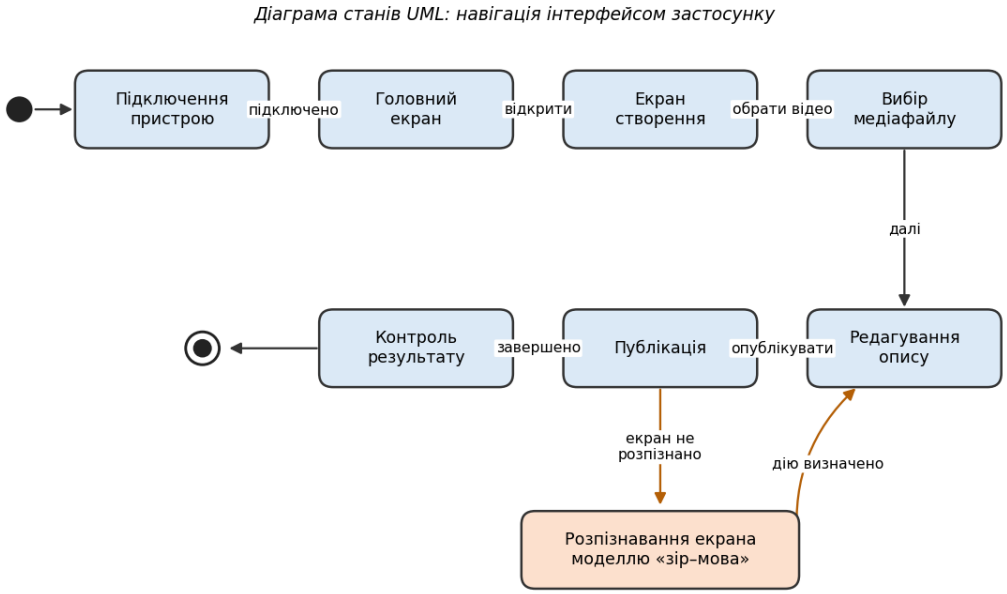

Рисунок 2.6 – Діаграма станів UML: навігація інтерфейсом застосунку

Для роботи з обліковими записами, що належать користувачу системи та захищені двофакторною автентифікацією, реалізовано генерацію одноразових паролів за стандартним алгоритмом TOTP згідно з RFC 6238 [23] засобами бібліотеки PyOTP [20]. Одноразовий пароль обчислюється на основі секретного ключа та поточного часу за виразом

$$TOTP(K,t)=HOTP\left(K,\lfloor\frac{t-T_0}{X}\rfloor\right), \tag{2.6}$$

де K – спільний секретний ключ; t – поточний час у секундах; T_0 – початковий момент відліку, який зазвичай дорівнює нулю; X – тривалість часового кроку, що, як правило, становить 30 секунд; $\lfloor\cdot\rfloor$ – операція округлення до найближчого цілого числа в менший бік; $HOTP$ – функція генерування одноразового пароля на основі геш-коду повідомлення та значення лічильника. Застосування часозалежних паролів дозволяє системі формувати дійсні одноразові коди для легітимних облікових записів власника без ручного введення на кожному кроці.

Висновки до другого розділу

У розділі виконано повне проєктування автоматизованої системи збору даних, обробки медіаконтенту та керування публікацією матеріалів. Обґрунтовано чотиришарову архітектуру, що охоплює серверний шар на FastAPI з підтримкою REST і WebSocket, клієнтський односторінковий застосунок, шар автоматизації на базі ADB та UI Automator і настільний графічний інтерфейс на Flet; функціональність системи відображено діаграмою варіантів використання, структурною схемою та діаграмою послідовності публікації. Спроєктовано логічну модель даних із семи сутностей у нотації UML, наведено її атрибутивний склад, зв’язки та обґрунтовано приведення до третьої нормальної форми з реалізацією у вигляді JSON-сховища та можливістю міграції на СУБД. Розроблено математичне та алгоритмічне забезпечення: формалізовано метод зіставлення зразків за нормованою крос-кореляцією, описано навігацію як скінченний автомат із функцією переходів, наведено двоетапний алгоритм детекції екрана з резервним розпізнаванням моделлю «зір–мова» та подано алгоритм генерації одноразових паролів TOTP. Отримані проєктні рішення утворюють завершену основу для програмної реалізації системи, розгляд якої винесено в наступний розділ.

РОЗДІЛ 3 ТЕХНІЧНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Цей розділ присвячено обґрунтуванню вибору інструментальних засобів для реалізації автоматизованої системи, а також формулюванню вимог до апаратного та програмного забезпечення, у межах яких система зберігає працездатність. Оскільки система об'єднує три функціонально відмінні підсистеми, а саме збір даних, обробку медіаконтенту та керування публікацією матеріалів, набір засобів розробки добирався з урахуванням сумісності цих підсистем у межах єдиної кодової бази, а також з урахуванням обмежень платформи Android, від якої залежить підсистема автоматизації. Виклад побудовано так: спершу обґрунтовано мову реалізації та бібліотеки кожної підсистеми, далі сформульовано вимоги до апаратного середовища, після чого наведено функціональні та нефункціональні вимоги до програмного забезпечення. Такий порядок дає змогу спершу зафіксувати інструментальну базу, а вже потім встановити умови, у яких ця база має функціонувати.

3.1 Засоби розробки

Базовою мовою реалізації обрано Python 3 [13]. Вибір зумовлено кількома чинниками. По-перше, мова має розвинену екосистему бібліотек для комп'ютерного зору, числових обчислень та роботи з мережею, що покриває всі три функції системи без переходу на сторонні середовища. По-друге, динамічна типізація у поєднанні з підказками типів дає змогу швидко прототипувати алгоритми навігації та обробки відео й водночас контролювати коректність структур даних на межах модулів. По-третє, інтерпретована природа мови спрощує налагодження сценаріїв взаємодії з пристроєм, де поведінка інтерфейсу застосунку є недетермінованою і потребує частих ітерацій. По-четверте, та сама мова застосовна як на серверній частині, так і в графічному інтерфейсі та у сценаріях автоматизації, що усуває потребу узгоджувати кілька технологічних стеків і знижує вартість супроводу. Дотримання принципів читабельності та поділу відповідальності,

викладених у [1], забезпечує супровідність коду за умови зростання кількості підтримуваних екранів та режимів заливу.

Серверну частину побудовано на вебфреймворку FastAPI [14]. Фреймворк обрано через асинхронну модель обробки запитів, автоматичну генерацію специфікації інтерфейсу та вбудовану валідацію вхідних даних. Доцільно окремо порівняти FastAPI з поширеними альтернативами, а саме Flask та Django REST Framework. Flask є мінімалістичним фреймворком, проте не містить вбудованої валідації запитів та асинхронної моделі обробки в базовій поставці, через що відповідні механізми довелося б додавати сторонніми розширеннями та узгоджувати між собою вручну. Django REST Framework, навпаки, є важким рішенням, орієнтованим на повний цикл вебзастосунку з власною об'єктно-реляційною моделлю та адміністративною панеллю, що є надлишковим для системи, у якій зберігання реалізовано у форматі JSON-документів, а основне навантаження припадає на асинхронну взаємодію з пристроями. FastAPI займає проміжну позицію: він поєднує невелику кількість обов'язкових залежностей з декларативною валідацією та нативною підтримкою асинхронності, що відповідає характеру навантаження системи, де паралельно обслуговується багато тривалих операцій вводу та виводу. Серверна частина надає понад п'ятдесят кінцевих точок REST, що покривають керування пристроями, акаунтами, застосунками, задачами заливу та збором статистики. Для передавання журналів виконання у реальному часі задіяно канал WebSocket згідно з [22], який транслює повідомлення про хід задачі від шару автоматизації до клієнтського інтерфейсу без періодичного опитування сервера. Така схема знижує затримку відображення подій та зменшує навантаження на серверну частину порівняно з регулярними HTTP-запитами на оновлення стану.

Валідацію та опис структур даних реалізовано засобами Pydantic [15]. Кожна сутність предметної області, зокрема пристрій, застосунок, акаунт, секрет одноразового пароля та запис журналу публікацій, описується як модель з явними типами полів. Роль Pydantic у системі полягає у тому, що він

виступає єдиним механізмом перевірки даних на межах модулів: дані, що надходять від клієнтського інтерфейсу, від шару автоматизації або зі сховища, перетворюються на типізовані об'єкти моделей перед тим, як потрапити до бізнес-логіки, а будь-яка невідповідність типів чи обмежень відхиляється з уніфікованим описом помилки. Це дає змогу автоматично перевіряти коректність тіл запитів, формувати узгоджені відповіді про помилки та підтримувати відповідність між шаром зберігання у форматі JSON-документів і логікою серверної частини. Оскільки FastAPI використовує ті самі моделі Pydantic для генерації специфікації інтерфейсу, опис даних та контракт інтерфейсу залишаються синхронізованими без дублювання. Узгодженість моделей даних із вимогами [3] спрощує супровід вимог до системи у міру їхньої зміни.

Підсистему обробки медіаконтенту побудовано на трьох бібліотеках, кожна з яких відповідає за свій етап конвеєра. OpenCV [16] застосовується для зчитування та запису відеопотоку, геометричних перетворень кадрів, корекції кольору, а також для зіставлення шаблонів під час розпізнавання екранів застосунку. Теоретичну основу застосованих операцій комп'ютерного зору викладено у [6], а практичні аспекти роботи з бібліотекою описано у [7]. NumPy [17] забезпечує подання кадрів як багатовимірних масивів та векторизовані обчислення, що прискорює покадрову обробку порівняно з поелементними циклами мовою Python. Оскільки OpenCV повертає кадри саме як масиви NumPy, обидві бібліотеки працюють над спільним поданням даних без додаткових перетворень, а корекція кольору, повороти та інші піксельні операції виконуються як векторні дії над масивами. Pillow [18] використовується для формування ескізів та операцій з растровими зображеннями, де потрібен зручний доступ до окремих форматів файлів і простих перетворень без розгортання повного конвеєра OpenCV. Конвеєр обробки медіаконтенту, таким чином, працює узгоджено: відеопотік декодується засобами OpenCV у масиви NumPy, числові перетворення пікселів виконуються над цими масивами, а підготовка зображень-ескізів для

попереднього перегляду покладається на Pillow. Розподіл завдань за спеціалізацією бібліотек дає змогу уникнути зайвих перетворень форматів і зберегти продуктивність покадрової обробки: декодування і трансформацію відео покладено на OpenCV, числові операції над пікселями – на NumPy, підготовку зображень-ескізів – на Pillow.

Генерацію одноразових паролів за стандартом TOTP реалізовано бібліотекою pyotp [20] відповідно до [23]. Це потрібно для автоматизованого проходження двофакторної автентифікації під час роботи з акаунтами, де секрет зберігається разом з обліковим записом, а код обчислюється на момент входу. Код TOTP обчислюється як значення геш-функції від спільного секрету та часового лічильника, утвореного діленням поточного часу на фіксований інтервал, що формально записується так:

$$TOTP(K,t)=HOTP(K,T),\quad T=\left\lfloor \frac{t-t_0}{X} \right\rfloor, \tag{3.1}$$

де K – спільний секретний ключ; t – поточний час; t_0 – початковий момент відліку; X – тривалість часового інтервалу дії коду; T – значення часового лічильника; $\lfloor \cdot \rfloor$ – операція округлення до найближчого цілого числа в менший бік; $HOTP$ – функція формування одноразового пароля на основі лічильника. Винесення цієї функції в окрему бібліотеку усуває потребу у самостійній реалізації криптографічного алгоритму та знижує ймовірність помилок у часовій синхронізації.

Настільний графічний інтерфейс побудовано засобами Flet [21]. Бібліотека дає змогу описати інтерфейс мовою Python без переходу на окремий стек технологій, що зменшує кількість мов у проєкті та спрощує повторне використання вже наявної серверної логіки. Для допоміжних сценаріїв автоматизації браузера, зокрема під час реєстрації та підготовки акаунтів, застосовано Playwright [19], який забезпечує керування браузером з очікуванням готовності елементів та перехопленням мережевих подій.

Ключовим для системи є шар автоматизації нативного Android-застосунку. Його реалізовано на основі засобів ADB [8], UI Automator [9] та SDK Platform Tools [10]. ADB використовується для встановлення з'єднання з пристроєм, передавання відеофайлів у каталог DCIM, запуску застосунку та надсилання подій вводу. UI Automator застосовується для отримання XML-дампу ієрархії елементів поточного екрана, пошуку вузлів за ідентифікатором ресурсу, текстом або описом вмісту, а також для визначення координат для тапів. SDK Platform Tools надають узгоджений набір системних утиліт, від яких залежать попередні два інструменти.

Окремо обґрунтуємо відмову від фреймворку Appium [11] на користь прямої взаємодії з ADB та UI Automator. Appium є зручним універсальним рішенням для кросплатформного тестування мобільних застосунків, проте у межах цієї системи його застосування є надлишковим. По-перше, Appium додає проміжний сервер та власний протокол, що збільшує кількість компонентів і потенційних точок відмови у конвеєрі керування пристроєм. По-друге, система працює виключно з платформою Android і не потребує абстракції над кількома платформами, заради якої вводиться додатковий рівень. По-третє, пряме звертання до ADB та UI Automator дає тонший контроль над послідовністю команд, що важливо для скінченного автомата навігації, де реакція на стан екрана має бути передбачуваною та швидкою. Порівняння з підходами вебавтоматизації, зокрема Selenium WebDriver [12], підтверджує загальний принцип: для вузькоспеціалізованого завдання на одній платформі доцільніше використовувати низькорівневі засоби, ніж універсальний драйвер, тоді як універсальні фреймворки виправдані за умови підтримки багатьох платформ і браузерів. Зведене порівняння двох підходів до автоматизації наведено у таблиці 3.1.

Таким чином, добір засобів розробки підпорядковано принципу мінімальної достатності: кожен інструмент покриває конкретну функцію системи, а спільна мова реалізації об'єднує підсистеми у єдину кодову базу зі зниженою кількістю технологічних залежностей.

Таблиця 3.1 – Порівняння підходів до автоматизації Android-застосунку

Критерій	**Пряма взаємодія (ADB та UI Automator)**	**Appium**
Кількість проміжних компонентів	мінімальна, прямі команди до пристрою	проміжний сервер та драйвер
Підтримка платформ	лише Android	кілька мобільних платформ
Контроль послідовності команд	високий, покрокове керування	опосередкований через протокол
Затримка реакції на стан екрана	низька	вища через проміжний шар
Складність розгортання	низька, лише Platform Tools	вища, потрібен сервер Appium
Доцільність для одноплатформного завдання	висока	надлишкова

3.2 Вимоги до технічного забезпечення

Технічне забезпечення системи поділяється на три групи: серверний вузол, на якому виконується серверна частина та шар автоматизації, набір Android-пристроїв, на які здійснюється залив, та робоче місце оператора. Серверний вузол виконує одночасну обробку кількох задач засобами пулу потоків, тому потребує достатньої кількості обчислювальних ядер, тоді як підсистема обробки медіаконтенту висуває вимоги до оперативної пам'яті та підсистеми зберігання через тимчасові файли відео. Кількість одночасно підключених пристроїв обмежується пропускною спроможністю USB-контролерів та правилом «один пристрій – одна задача», що визначається

менеджером задач серверної частини. Узагальнені вимоги до технічного забезпечення наведено у таблиці 3.2.

Таблиця 3.2 – Вимоги до технічного забезпечення

Складова	**Мінімальні вимоги**	**Рекомендовані вимоги**
Процесор серверного вузла	4 логічні ядра, 2.0 ГГц	8 і більше логічних ядер, 3.0 ГГц
Оперативна пам'ять	8 ГБ	16 ГБ і більше
Накопичувач	256 ГБ, SSD	512 ГБ і більше, NVMe SSD
Мережеве з'єднання	стабільний доступ до мережі Інтернет, 20 Мбіт/с	100 Мбіт/с і більше
Підключення пристроїв	USB 2.0, концентратор із зовнішнім живленням	USB 3.0, концентратор із зовнішнім живленням
Android-пристрої	Android 8.0, активований режим налагодження USB	Android 10 і новіше, активований режим налагодження USB
Робоче місце оператора	браузер з підтримкою ES6, роздільність 1366 на 768	браузер з підтримкою ES6, роздільність 1920 на 1080

Кожна з наведених у таблиці позицій має технічне обґрунтування. Вимога до процесора серверного вузла випливає з того, що менеджер задач паралельно обслуговує кілька пристроїв, а кожна задача заливу поєднує тривалі операції вводу та виводу з періодичними обчисленнями для розпізнавання екранів, тому чотири логічні ядра є нижньою межею, за якої система зберігає реактивність, а вісім і більше ядер дають запас для одночасної обробки медіаконтенту. Обсяг оперативної пам'яті у 8 ГБ

зумовлено тим, що декодований кадр відео у поданні NumPy займає кілька мегабайтів, а транскодування кількох відеофайлів водночас разом із серверною частиною та шаром автоматизації утримує в пам'яті проміжні буфери; 16 ГБ і більше прибирають ризик звернення до файлу підкачки під пікове навантаження. Накопичувач типу SSD рекомендовано через інтенсивні операції запису й читання тимчасових файлів під час транскодування відео у формат H.264, де механічний диск став би вузьким місцем конвеєра; інтерфейс NVMe додатково скорочує час доступу за паралельного транскодування. Вимога до мережевого з'єднання випливає з потреби завантажувати публічні метрики та відеоматеріали, а також обслуговувати клієнтський інтерфейс. Підключення пристроїв через концентратор із зовнішнім живленням є обов'язковим за умови одночасного підключення кількох пристроїв, оскільки під час передавання відеофайлів та виконання заливу пристрої споживають додаткову потужність, а нестача живлення спричиняє розрив з'єднання ADB. Вимога до версії Android та активованого режиму налагодження USB зумовлена тим, що шар автоматизації спирається на ADB та UI Automator, доступні починаючи з відповідних версій платформи. Параметри робочого місця оператора визначаються потребою відображати односторінковий клієнтський застосунок без горизонтального прокручування на типових роздільностях.

3.3 Вимоги до програмного забезпечення

Вимоги до програмного забезпечення сформульовано відповідно до підходу, викладеного у [3], з розмежуванням на функціональні та нефункціональні. Функціональні вимоги згруповано за трьома функціями системи, що відповідає її загальній структурі. Кожна група вимог відповідає одній з підсистем, описаних у попередніх розділах, що дає змогу простежити відповідність між функціями системи та інструментальними засобами їх реалізації.

Підсистема збору даних відповідає за отримання та узагальнення публічних показників ефективності опублікованих матеріалів, тому її вимоги охоплюють доступ до метрик, керування мережевими з'єднаннями, агрегацію та експорт. Функціональні вимоги до підсистеми збору даних:

– система повинна отримувати публічні метрики профілів та публікацій, зокрема перегляди, лайки, кількість підписників та охоплення, через відкритий програмний інтерфейс;
– система повинна підтримувати конфігурований пул мережевих з'єднань (проксі) для балансування навантаження запитів збору даних та розподілу їх між наборами облікових записів;
– система повинна агрегувати показники за окремими креативами на основі журналу публікацій, зіставляючи опубліковані матеріали з вихідними відеофайлами;
– система повинна вести журнал публікацій з фіксацією акаунта, застосунку, відеофайла, мітки часу та результату операції;
– система повинна забезпечувати експорт зведених показників у формат Excel.

Підсистема обробки медіаконтенту відповідає за підготовку відеофайлів до публікації, тому її вимоги охоплюють перетворення кадрів, транскодування, формування ескізів та автоматичність конвеєра. Функціональні вимоги до підсистеми обробки медіаконтенту:

– система повинна виконувати унікалізацію відео шляхом повороту, дзеркалення та корекції кольору кадрів;
– система повинна транскодувати відео у формат H.264 із збереженням придатної для публікації якості;
– система повинна формувати ескізи відеофайлів для попереднього перегляду;
– система повинна опрацьовувати відео без втручання оператора у проміжні етапи конвеєра обробки.

Підсистема керування публікацією матеріалів відповідає за безпосередню взаємодію з Android-пристроями та публікацію підготовлених матеріалів, тому її вимоги охоплюють режими заливу, передавання файлів, навігацію інтерфейсом та фіксацію результатів. Функціональні вимоги до підсистеми керування публікацією матеріалів:

– система повинна виконувати залив Reels та Stories у режимах одного акаунта, усіх акаунтів обраного застосунку та кількох застосунків одночасно;
– система повинна передавати відеофайли на пристрій у каталог DCIM засобами ADB перед публікацією;
– система повинна здійснювати навігацію інтерфейсом застосунку на основі скінченного автомата з розпізнаванням поточного екрана;
– система повинна викликати локальну мультимодальну модель «зір-мова» для невідомих екранів та валідувати запропоновану дію перед її виконанням;
– система повинна вводити опис публікації з набору наперед заданих пресетів;
– система повинна підтримувати планування задач та режими зупинки й призупинення виконання;
– система повинна фіксувати результати заливу в журналі публікацій.

Нефункціональні вимоги описують якісні властивості системи у цілому, незалежні від окремої підсистеми, і обґрунтовуються характером навантаження та умовами експлуатації. Нефункціональні вимоги до системи:

– надійність: система повинна дотримуватися правила «один пристрій – одна задача» та коректно опрацьовувати розрив з'єднання з пристроєм, не перериваючи виконання задач на інших пристроях; ця вимога зумовлена тим, що збій одного пристрою не повинен впливати на паралельні задачі;
– продуктивність: система повинна обробляти кілька задач паралельно засобами пулу потоків та транслювати журнали виконання у

реальному часі без помітної для оператора затримки; вимога випливає з потреби одночасно обслуговувати багато пристроїв та надавати оператору актуальний стан виконання;

– зручність використання: клієнтський інтерфейс повинен надавати єдиний доступ до всіх режимів роботи через односторінковий застосунок з відображенням стану задач та сповіщеннями, що скорочує кількість дій оператора для запуску й контролю задач;

– супровідність: програмний код повинен мати модульну структуру з розмежуванням серверної частини, шару автоматизації та інтерфейсу, що відповідає принципам читабельності та поділу відповідальності [1] і спрощує внесення змін у межах окремого модуля;

– розширюваність: архітектура повинна допускати додавання нових екранів до шаблону навігації та міграцію зі зберігання у форматі JSON-документів на повноцінну систему керування базами даних [24], що дає змогу нарощувати обсяг даних без переписування бізнес-логіки;

– безпека: секрети одноразових паролів та облікові дані повинні зберігатися у структурованому вигляді з контрольованим доступом, а генерація кодів автентифікації має відповідати стандарту [23];

– сумісність: клієнтська частина повинна працювати у браузерах із підтримкою стандарту ES6, а серверна частина – на операційних системах із наявним середовищем виконання Python 3 та інструментами ADB.

3.4 Опис реалізації

Реалізацію системи виконано відповідно до чотиришарової архітектури, обґрунтованої у попередніх підрозділах. Поділ на шари забезпечує слабку зв’язність компонентів, локалізацію змін та можливість незалежного тестування кожного модуля, що узгоджується з принципами чистого коду та проєктування на основі відповідальностей [1]. Кожен шар

має чітко визначену зону відповідальності та взаємодіє із суміжними шарами через формалізовані інтерфейси, що описано нижче. Систему призначено для роботи виключно з власними або належним чином авторизованими обліковими записами; усі описані сценарії передбачають дотримання правил відповідних платформ, ліцензійних умов та чинного законодавства.

Загальна структура за шарами

Перший шар, ядро автоматизації, інкапсулює всю низькорівневу взаємодію з фізичним або емуляторним Android-пристроєм. Він приховує деталі протоколу ADB та формат дампів UI Automator від решти системи, надаючи вищим шарам набір предметних операцій: запуск застосунку, пошук елемента, тап за координатами, введення тексту, передавання файлу. Другий шар, шар даних, відповідає за збереження та читання сутностей системи у форматі JSON-документів, а також за валідацію їхньої структури засобами Pydantic [15]. Третій шар, backend, реалізує прикладну логіку, керує виконанням задач та публікує REST- і WebSocket-інтерфейси засобами FastAPI [14]. Четвертий шар, клієнтський, представлений односторінковим вебзастосунком (SPA) на чистому JavaScript та альтернативним десктопним інтерфейсом на Flet [21].

Шар даних побудовано як сукупність репозиторіїв, кожен з яких відповідає за окрему колекцію документів та інкапсулює операції читання, запису, оновлення й видалення. Репозиторій приховує від прикладної логіки конкретний механізм збереження, надаючи їй типізований доступ до сутностей. Завдяки єдиному формату документів і моделям Pydantic як контракту даних кожен репозиторій є самодостатнім модулем, який можна тестувати ізольовано від решти системи [15],[24]. Така організація відокремлює правила доступу до даних від прикладної логіки та зменшує зв’язність шарів.

Координацію тривалих операцій покладено на диспетчер задач у складі backend. Диспетчер ставить запити користувача у чергу, призначає їх вільним виконавцям пулу потоків та відстежує життєвий цикл кожної задачі:

очікування, виконання, успішне завершення, помилка чи скасування. Помилки виконання локалізуються на рівні окремої задачі та не впливають на роботу решти задач: виняток перехоплюється, фіксується у журналі подій і відображається у стані задачі, після чого пристрій звільняється для подальшого використання. Для операцій, що можуть завершитися тимчасовою помилкою, передбачено повторні спроби з обмеженою кількістю повторень та паузою між ними, що підвищує стійкість системи до короткочасних збоїв зв’язку з пристроєм.

Ключові модулі

Ядро ADB-автоматизації побудовано як набір функцій-обгорток над утилітою командного рядка adb [8]. Базова операція полягає у формуванні команди оболонки, її виконанні через підпроцес та розборі стандартного виводу. Для аналізу інтерфейсу екранний стан отримується командою дампу ієрархії UI Automator, після чого XML-документ розбирається у дерево вузлів; пошук цільового елемента ведеться за атрибутами resource-id, text або content-desc [9]. У разі коли текстова ідентифікація неможлива, застосовується пошук за візуальним шаблоном засобами template matching OpenCV [7]. Координати знайденого елемента перераховуються у центр його обмежувального прямокутника та передаються команді тапу. Усі низькорівневі операції виконуються через єдиний інтерфейс, що дає змогу замінити фізичний пристрій емулятором без зміни вищих шарів.

Шар даних реалізовано у вигляді репозиторіїв, що серіалізують сутності у JSON та зчитують їх назад. Логічна модель даних, описана діаграмою класів у підрозділі 3.2, відображається на колекції документів: пристрої, застосунки, акаунти, секрети одноразових паролів, журнал публікацій ZalivLog, конфігураційні параметри та перевизначення опису й цільової папки. Кожна сутність супроводжується моделлю Pydantic, що гарантує типову коректність під час читання та запису [15]. Така організація спрощує подальшу міграцію на реляційну СУБД без зміни прикладної логіки [24].

Backend об'єднує три взаємопов’язані механізми: маршрутизатор REST, що публікує понад п’ятдесят ендпоінтів для керування пристроями, акаунтами, заливом та статистикою; канал WebSocket для трансляції журналів подій у реальному часі [22]; та диспетчер задач TaskManager, побудований на пулі потоків ThreadPoolExecutor. Диспетчер дотримується інваріанта «один пристрій – одна активна задача», що виключає конфлікти доступу до пристрою, та підтримує сигнали зупинки й призупинення виконання. Кожна довготривала операція реєструється як задача з унікальним ідентифікатором, що дозволяє відстежувати її стан і керувати ним з боку клієнта.

Клієнтський односторінковий застосунок реалізує одинадцять функціональних сторінок: дашборд, залив, сторіс, керування застосунками, акаунти, додавання акаунтів, аватарки, реєстрація облікових записів, статистика, відеоменеджер і налаштування. Маршрутизація між сторінками виконується за фрагментом адреси (hash-маршрутизація) без перезавантаження документа. Повторно використовувані компоненти, зокрема вибір пристрою та застосунків, панель перегляду логів, модальні вікна та сповіщення, винесено в окремі модулі, що відповідає принципу єдиної відповідальності [1].

Модуль збору даних instagram_stats отримує метрики профілів і публікацій, а саме перегляди, лайки, кількість підписників та охоплення, з використанням сесій до публічного API. Запити до API виконуються з дотриманням обмежень частоти (rate-limit) та з коректним розподілом навантаження; для цього передбачено зміну мережевого маршруту через задану кількість опрацьованих акаунтів. Зібрані показники агрегуються за окремими креативами шляхом зіставлення опублікованих рилсів із записами журналу публікацій, після чого результати експортуються у формат Excel засобами openpyxl.

Модуль обробки медіаконтенту виконує адаптацію та постобробку відеоматеріалів і підготовку допоміжних зображень. Операції повороту,

дзеркального відображення та корекції кольору реалізовано засобами OpenCV і NumPy шляхом покадрових перетворень матриці пікселів [6],[7],[17]; вихідний потік перекодовується у кодек H.264. Формування ескізів виконується вибором репрезентативного кадру та його масштабуванням засобами Pillow [18].

Опис ключових методів

Нижче наведено стислий опис ключових методів у форматі «призначення – вхідні дані – вихідні дані».

Метод публікації матеріалу призначений для заливу окремого відео у вибраний акаунт. Вхідними даними є ідентифікатор пристрою, ім'я пакета застосунку, обліковий запис, шлях до відеофайлу та опис із пресету. Метод передає відео на пристрій командою ADB push у каталог DCIM, відкриває застосунок, проходить екрани створення публікації під керуванням автомата навігації, вводить опис та підтверджує публікацію. Вихідним результатом є ознака успіху та запис у журналі ZalivLog.

Метод навігації реалізує перехід системи між екранами застосунку. Вхідними даними є поточний стан автомата та цільовий екран; вихідними – послідовність виконаних дій та новий стан. Логіку навігації формалізовано скінченним автоматом, функцію переходів якого описано у підрозділі 3.3 виразом $\delta(s, e) = s'$, де s – поточний екран, e – подія, s' – наступний екран [5].

Метод детекції екрана визначає, на якому з відомих екранів перебуває застосунок. Вхідними даними є дамп ієрархії інтерфейсу та знімок екрана; вихідними – ідентифікатор розпізнаного екрана або ознака невідомого стану. Спочатку перевіряються характерні ознаки кожного з понад тридцяти семи екранів шаблону навігації flow_template. Якщо жодну ознаку не виявлено, активується метод обробки невідомого екрана з локальною мультимодальною моделлю зору та мови. Ця модель приймає на вхід знімок екрана та повертає рекомендовану дію разом з оцінкою впевненості. Запропонована дія валідовується перед виконанням, а у разі успіху новий

екран із його ознаками додається до шаблону навігації, завдяки чому система поступово розширює власну базу відомих станів.

Метод зіставлення за шаблоном реалізує пошук візуального елемента на знімку екрана. Вхідними даними є зображення екрана та еталонний фрагмент; вихідними – координати найкращого збігу та значення метрики. Метрикою слугує нормована крос-кореляція, а збіг визнається дійсним за умови перевищення заданого порогу впевненості [7],[16]. Формула нормованої крос-кореляції наведена у підрозділі 3.3.

Метод генерації одноразового пароля призначений для проходження двофакторної автентифікації власного облікового запису. Вхідними даними є спільний секрет облікового запису; вихідним – шестизначний код, дійсний у поточному часовому вікні. Метод реалізовано засобами бібліотеки pyotp за алгоритмом TOTP згідно з RFC 6238 [20],[23].

Метод збору статистики отримує метрики для переліку акаунтів. Вхідними даними є перелік облікових записів та параметр частоти зміни мережевого маршруту; вихідними – структуровані показники профілів і публікацій. Метод відкриває сесію, послідовно запитує метрики з дотриманням обмежень частоти, змінює мережеву IP-адресу через задану кількість акаунтів та накопичує результати для подальшої агрегації й експорту.

Метод адаптації відео призначений для створення візуально відмінної копії вихідного матеріалу. Вхідними даними є шлях до відео та набір параметрів перетворення; вихідним – шлях до обробленого файлу у кодеку H.264. Метод послідовно застосовує геометричні та колірні перетворення до кадрів і перекодовує результат [6],[17].

Організація взаємодії шарів

Взаємодію між шарами організовано за патерном зворотного виклику (callback), що дозволяє нижнім шарам сповіщати верхні про події, не маючи прямих залежностей від них [2]. Ланцюг передавання повідомлення про подію виконання має такий вигляд:

– ядро автоматизації під час виконання дії формує текстове повідомлення журналу та викликає переданий йому зворотний виклик;

– цей виклик передає повідомлення диспетчеру задач TaskManager, який пов’язує його з відповідною задачею та пристроєм;

– диспетчер публікує повідомлення у канал WebSocket;

– backend надсилає його підключеним клієнтам;

– клієнтський застосунок відображає повідомлення на панелі логів у реальному часі [14],[22].

Завдяки такій організації інтерфейс отримує оновлення без періодичного опитування сервера, а ядро автоматизації залишається незалежним від способу подання даних користувачу. Узагальнену схему клієнт-серверної взаємодії наведено на рис. 3.1.

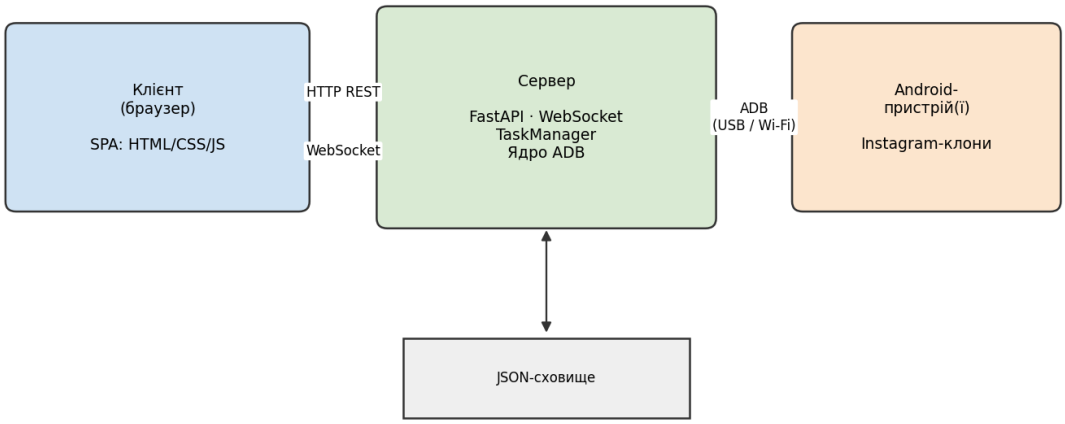

Рисунок 3.1 – Схема клієнт-серверної взаємодії компонентів

3.5 Методика застосування системи

Цей підрозділ описує порядок роботи з системою з позиції користувача та слугує стислим керівництвом з експлуатації. Усі наведені сценарії розраховано на використання з власними або авторизованими обліковими записами та за умови дотримання правил відповідних платформ.

Запуск та підключення пристрою

Перед початком роботи необхідно запустити серверну частину, після чого вебінтерфейс стає доступним у браузері за локальною адресою; альтернативно застосовується десктопний інтерфейс на Flet [21]. Android-пристрій підключається кабелем USB або через бездротове з’єднання за умови ввімкненого режиму налагодження USB. Коректність підключення перевіряється на сторінці керування пристроями: система виводить перелік доступних пристроїв з ідентифікатором і моделлю, отриманими засобами ADB [8],[10]. Після вибору пристрою користувач переходить до потрібного функціонального розділу.

Опис сторінок інтерфейсу

Головне вікно, або дашборд, є точкою входу до системи та подає зведену інформацію про стан підключених пристроїв, активні задачі й останні події (рис. 3.2). Звідси користувач переходить до решти розділів за допомогою навігаційного меню. Темне оформлення з елементами ефекту скла (glassmorphism) забезпечує комфортне сприйняття під час тривалої роботи.

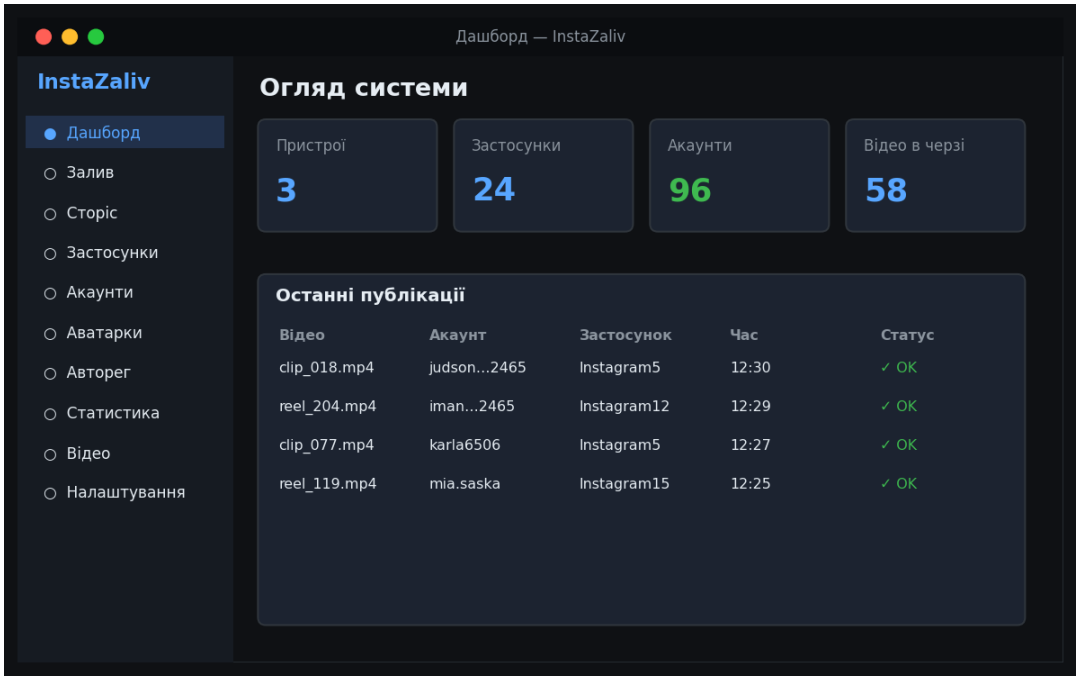

Рисунок 3.2 – Головне вікно (дашборд) вебінтерфейсу системи

Вікно публікації, або сторінка заливу, призначене для керування публікацією матеріалів (рис. 3.3). Користувач обирає пристрій, цільовий застосунок та режим роботи: публікація в один акаунт, в усі акаунти застосунку або одночасно в кілька застосунків. Далі вказуються відеоматеріали та опис із пресету, після чого задача ставиться у чергу диспетчера. Аналогічно організовано публікацію матеріалів у форматі Stories на окремій сторінці.

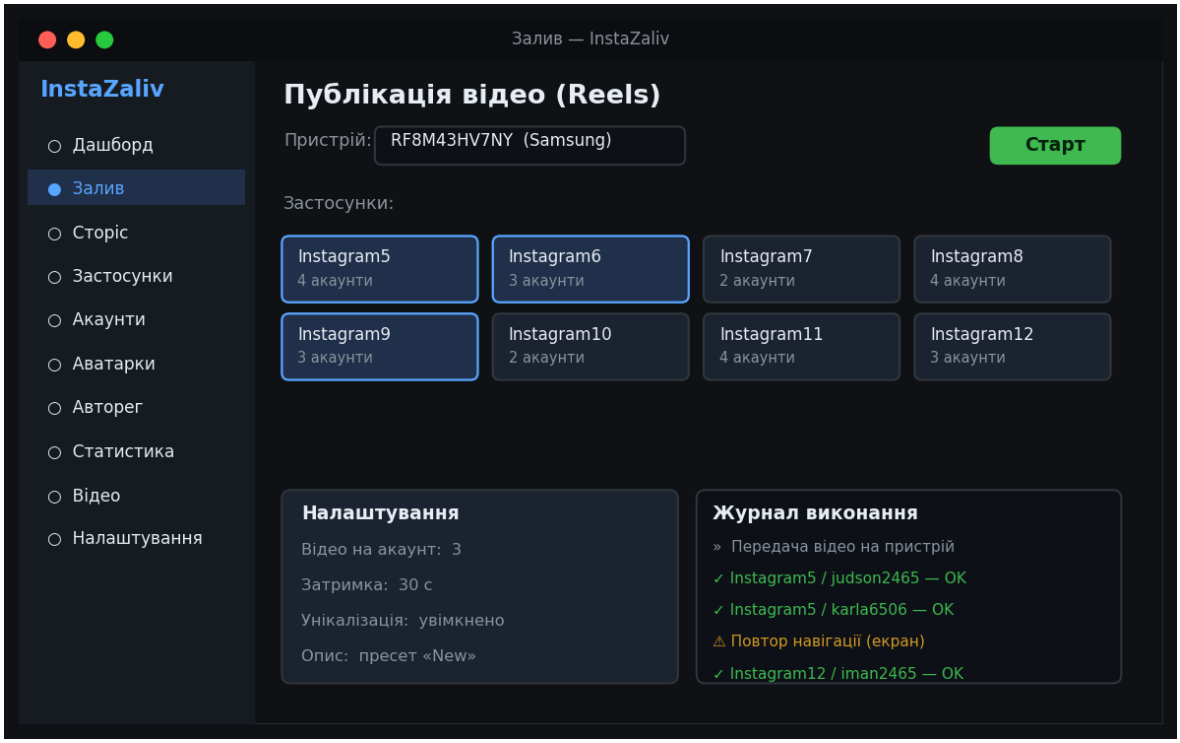

Рисунок 3.3 – Вікно публікації відео (модуль «Залив»)

Вікно аналітики, або сторінка статистики, забезпечує збір та перегляд показників акаунтів і публікацій (рис. 3.4). Користувач задає перелік акаунтів та параметри збору, зокрема частоту зміни мережевого маршруту, та запускає процес. Зібрані метрики, а саме перегляди, лайки, підписники й охоплення, подаються у таблиці з можливістю агрегації за креативами та експорту у файл Excel.

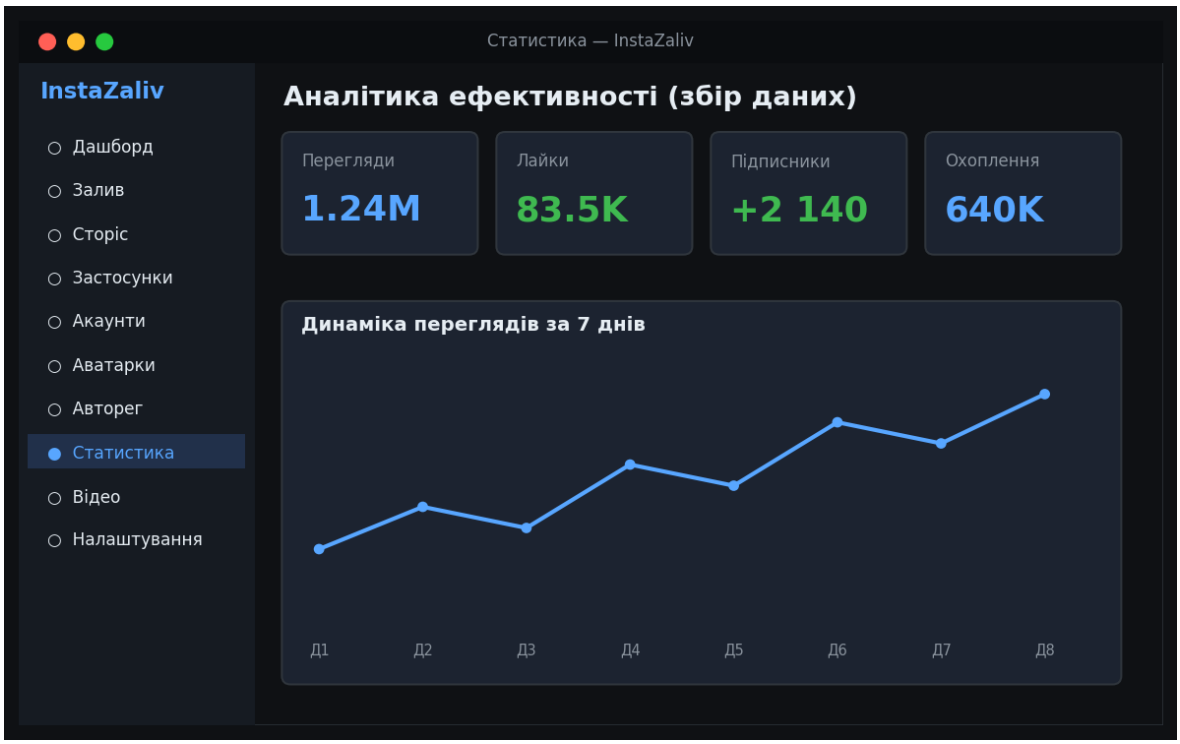

Рисунок 3.4 – Вікно аналітики ефективності (підсистема збору даних)

Типові сценарії та моніторинг

Типовий сценарій публікації передбачає такі кроки:

– підключення та перевірка пристрою на сторінці керування пристроями;
– за потреби, підготовка та адаптація відео у відеоменеджері;
– вибір застосунку, режиму роботи та матеріалів на сторінці заливу;
– запуск задачі та спостереження за її виконанням на панелі логів;
– перевірка результату за записами журналу публікацій.

Сценарій обробки медіа реалізується на сторінці відеоменеджера. Користувач завантажує вихідні матеріали, задає параметри перетворення, зокрема поворот, дзеркальне відображення та корекцію кольору, після чого запускає процес адаптації. Результатом є набір оброблених відеофайлів у кодеку H.264 із сформованими ескізами, готових до подальшого використання на сторінці заливу. Цей сценарій дозволяє підготувати

медіаматеріали окремо від процесу публікації та повторно застосувати їх за потреби.

Сценарій аналітики полягає у періодичному зборі статистики для відстеження ефективності креативів та подальшому експорті даних для звітності. Користувач задає перелік облікових записів і параметри збору, запускає процес, після чого переглядає метрики у таблиці та зберігає їх у файл Excel. Під час виконання будь-якої задачі користувач має змогу спостерігати за ходом її виконання у реальному часі: повідомлення журналу надходять від ядра автоматизації каналом WebSocket та відображаються на панелі логів без оновлення сторінки [22]. За потреби активну задачу можна призупинити або зупинити відповідними сигналами диспетчера, що особливо важливо у разі виявлення нештатної поведінки застосунку на пристрої.

Висновки до третього розділу

У розділі описано реалізацію автоматизованої системи збору даних, обробки медіаконтенту та керування публікацією матеріалів відповідно до чотиришарової архітектури. Розкрито призначення та внутрішню організацію ключових модулів: ядра ADB-автоматизації, шару даних на JSON-документах із валідацією засобами Pydantic, backend на FastAPI з диспетчером задач і каналом WebSocket, клієнтського односторінкового застосунку, а також спеціалізованих модулів збору статистики та обробки медіаконтенту. Наведено стислий опис ключових методів публікації, навігації, детекції екрана із залученням моделі зору та мови, зіставлення за шаблоном, генерації одноразових паролів, збору статистики та адаптації відео із зазначенням їхнього призначення, вхідних і вихідних даних. Описано організацію взаємодії шарів за патерном зворотного виклику та подано ланцюг передавання повідомлень від ядра автоматизації до клієнтського інтерфейсу через диспетчер задач і канал WebSocket. У підрозділі методики застосування наведено порядок запуску системи, підключення пристрою, призначення основних сторінок інтерфейсу та типові сценарії роботи з

акцентом на моніторинг виконання у реальному часі та дотримання правил платформ. Отримані результати підтверджують працездатність прийнятих архітектурних і технічних рішень та створюють підґрунтя для подальшого тестування системи, що розглядається у наступному розділі.

РОЗДІЛ 4 ОХОРОНА ПРАЦІ

4.1 Організаційно-правові основи забезпечення безпеки праці

Охорона праці є системою правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних та лікувально-профілактичних заходів і засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності. В умовах широкого впровадження інформаційних технологій роль охорони праці суттєво зростає, оскільки значна частина працівників виконує свої обов'язки за персональними комп'ютерами та з використанням різноманітних технічних і програмних засобів. Саме тому забезпечення безпеки та захисту здоров'я користувачів програмних і технічних засобів стає невід'ємною складовою організації будь-якого сучасного робочого процесу, зокрема діяльності, пов'язаної з підготовкою, обробкою та публікацією медіаконтенту [31].

Державна політика у сфері охорони праці ґрунтується на низці основних принципів, серед яких пріоритет життя і здоров'я працівників над результатами виробничої діяльності, повна відповідальність роботодавця за створення безпечних і нешкідливих умов праці, соціальний захист працівників, навчання населення з питань охорони праці, а також використання економічних методів управління охороною праці. Реалізація цих принципів забезпечується через систему організаційних заходів: створення служби охорони праці, проведення інструктажів і навчання, атестацію робочих місць, планування та фінансування заходів безпеки [29].

Законодавча база України у сфері охорони праці має кілька рівнів. Основоположним документом є Закон України «Про охорону праці», який визначає основні положення щодо реалізації конституційного права працівників на належні, безпечні та здорові умови праці [29]. Питання захисту населення і територій від надзвичайних ситуацій, у тому числі пожеж та наслідків воєнних дій, регулюються Кодексом цивільного захисту України

[30]. Конкретні вимоги до умов праці встановлюються підзаконними нормативно-правовими актами, державними будівельними нормами та державними санітарними нормами і правилами – зокрема щодо природного й штучного освітлення [33], електромагнітних полів [34] та мікроклімату виробничих приміщень [35]. Україна також орієнтується на міжнародні стандарти у сфері управління безпекою праці, серед яких стандарт ДСТУ ISO 45001:2019 [36].

Належне організаційне забезпечення охорони праці, своєчасне планування заходів безпеки та неухильне дотримання встановлених вимог під час експлуатації обладнання й програмних комплексів дають змогу мінімізувати рівень професійних ризиків та запобігти виникненню небезпечних ситуацій. З огляду на це аналіз умов праці та оцінювання ризиків на конкретному об'єкті проектування є обов'язковою складовою забезпечення безпеки.

4.2 Характеристика об'єкта та виявлення потенційних небезпек

Об'єктом проектування у цій роботі є автоматизована система роботи з медіаконтентом, безпосередня експлуатація якої здійснюється на робочому місці контент-менеджера (оператора). Розглянемо середовище, у якому виконується робота з контентом, оскільки саме воно визначає коло потенційних небезпек.

Робоче місце контент-менеджера розташоване в офісному приміщенні та обладнане персональним комп'ютером або ноутбуком з монітором (відеодисплейним терміналом), клавіатурою та маніпулятором «миша», робочим столом і кріслом. Додатково до робочого місця можуть під'єднуватися мобільні пристрої під керуванням ОС Android, через які виконується автоматизація публікації. Приміщення обладнане системами електроживлення, освітлення та, як правило, кондиціонування повітря.

Контент-менеджер виконує переважно розумову працю операторського типу: підготовку та обробку відеоматеріалів, формування описів,

налаштування й моніторинг процесів публікації, аналіз статистичних показників. Така діяльність характеризується тривалим перебуванням у вимушеній робочій позі (сидячи), безперервною роботою з відеодисплейним терміналом, високим зоровим та нервово-емоційним навантаженням, монотонністю окремих операцій. Усе це безпосередньо впливає на функціональний стан працівника, його працездатність та стан здоров'я у разі недотримання вимог охорони праці [31].

Виявлення небезпечних та шкідливих виробничих факторів на робочому місці виконано на основі їх класифікації за природою дії (фізичні, хімічні, біологічні та психофізіологічні фактори), наведеної у методичних вказівках [37]. Для досліджуваного об'єкта характерними є насамперед фізичні та психофізіологічні фактори, а також небезпеки загального характеру. Результати аналізу із зазначенням джерела кожної небезпеки та можливих наслідків її реалізації наведено у таблиці 4.1.

Таблиця 4.1 – Виявлення потенційних небезпек до об'єкта проектування

№	**Потенційна небезпека**	**Джерело небезпеки**	**Можливі наслідки**
1	2	3	4
1	Недостатня освітленість робочої зони, пряма і відбита блискість, підвищена пульсація світлового потоку	Нераціональне природне та штучне освітлення, відблиски на екрані монітора	Зорове втомлення, погіршення зору, зниження працездатності
2	Підвищений рівень електромагнітних випромінювань	Монітор, системний блок, мобільні пристрої, бездротові мережі	Головний біль, втома, погіршення самопочуття
3	Підвищене значення напруги в електричному колі (небезпека ураження струмом)	Електрообладнання ПК, електропроводка, зарядні пристрої	Електротравма, опіки, у тяжких випадках – загроза життю

Продовження таблиці 4.1

1	2	3	4
4	Незадовільні параметри мікроклімату (температура, вологість, рухливість повітря)	Недостатня вентиляція та кондиціонування приміщення	Дискомфорт, перегрівання чи переохолодження, зниження працездатності
5	Підвищений рівень статичної електрики	Робота ПК, синтетичні покриття та матеріали	Дискомфорт, накопичення пилу, ризик пошкодження техніки
6	Нервово-психічні перевантаження (розумове перенапруження, перенапруження зорових аналізаторів, монотонність, емоційні перевантаження)	Тривала однотипна робота з контентом, стислі терміни, інтенсивний інформаційний потік	Стрес, хронічна втома, зниження уваги та продуктивності
7	Фізичні статичні перевантаження	Тривале перебування у вимушеній робочій позі сидячи	Захворювання опорно-рухового апарату, порушення кровообігу
8	Небезпека виникнення пожежі	Несправність і перевантаження електрообладнання, коротке замикання	Загроза життю і здоров'ю, знищення обладнання та майна
9	Воєнна загроза (обстріли, повітряна тривога)	Воєнний стан на території держави	Загроза життю і здоров'ю працівників, руйнування приміщення та обладнання

Аналіз показує, що вплив більшості виявлених фізичних і психофізіологічних факторів (освітлення, мікроклімат, електромагнітні випромінювання, зорове та нервово-емоційне навантаження) можна суттєво зменшити шляхом раціональної організації робочого місця та дотримання санітарних норм. Водночас низку небезпек загального характеру (пожежа, ураження електричним струмом, воєнна загроза) повністю усунути неможливо, тому щодо них необхідно провести оцінювання ризику та розробити заходи із його зниження, що становить завдання наступного підрозділу.

4.3 Дослідження ризику реалізації потенційних небезпек на об'єкті проектування та розробка заходів щодо їх попередження

Оцінювання ризику – це процедура виявлення небезпек та визначення величини ризику їх реалізації з метою прийняття обґрунтованих рішень щодо забезпечення безпеки. Її основними задачами є ідентифікація небезпек, аналіз причин і можливих наслідків їх настання, кількісне або якісне визначення рівня ризику та обґрунтування пріоритетності захисних заходів. Оцінювання ризику дає змогу зосередити обмежені ресурси на найбільш значущих небезпеках і є основою сучасних систем управління охороною праці [36]. Для дослідження ризику в цій роботі застосовано два методи, наведені у методичних вказівках [37]: аналізування «дерева відмов» та матрицю оцінювання ризиків.

Метод аналізування «дерева відмов» полягає у тому, що спочатку визначають небажану (кінцеву) подію, а потім графічно, у формі логічної деревоподібної схеми, відображають усі способи, у які вона може відбутися. Для побудови дерева використовують логічні операції «Та» (подія настає лише за одночасної наявності всіх причин) та «Або» (подія настає за наявності хоча б однієї з причин). Як приклад на рисунку 4.1 побудовано «дерево відмов» для такої небезпечної події, як пожежа на робочому місці, що є однією з найпоширеніших небезпек у приміщеннях з обчислювальною

технікою [32]. Аналіз дерева показує, що пожежа є наслідком одночасної наявності горючого матеріалу, джерела запалювання та окисника, причому джерело запалювання, своєю чергою, найчастіше пов'язане з несправністю або перевантаженням електрообладнання. Це вказує на основні напрями профілактики – усунення джерел запалювання та обмеження кількості горючих матеріалів.

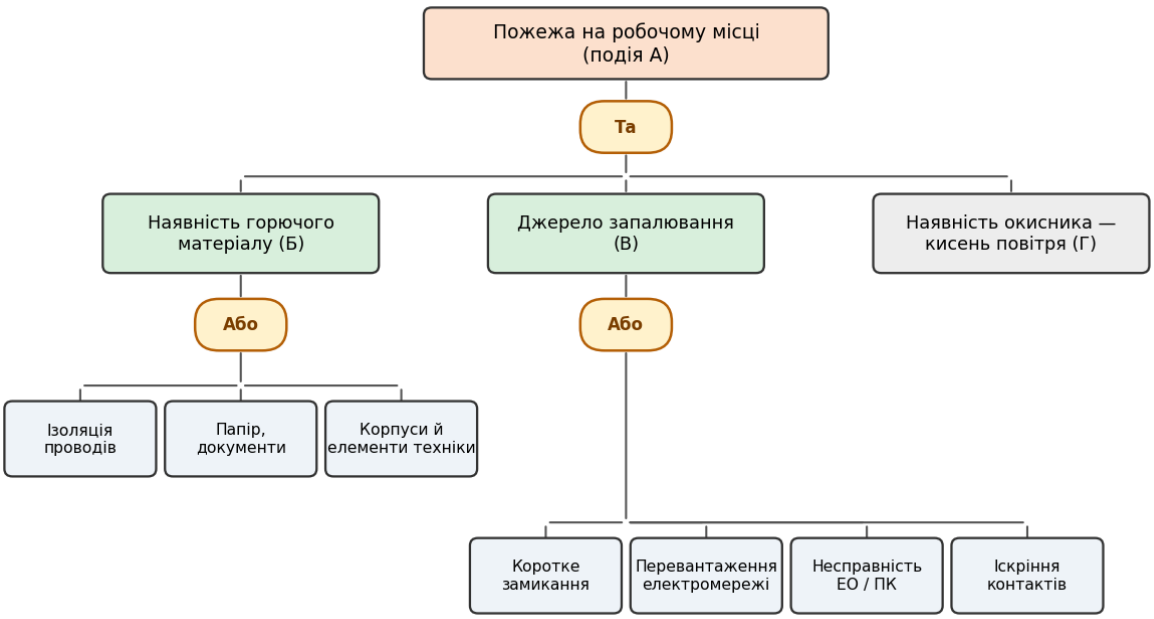

Рисунок 4.1 – «Дерево відмов» для події «Пожежа на робочому місці»

Другим застосованим методом є матриця оцінювання ризиків. Кожну небезпеку характеризують за категорією серйозності наслідків та рівнем ймовірності її настання, шкали яких (категорії серйозності, рівні ймовірності та власне матриця оцінки ризику) наведено у методичних вказівках [37]. За результатами зіставлення цих показників визначають індекс ризику та ступінь його припустимості, який може бути неприпустимим, небажаним (гранично допустимим), припустимим за умови контролю або припустимим. За цією методикою оцінено ризик п'яти найбільш значущих небезпек, виявлених у підрозділі 4.2. Результати оцінювання подано у таблиці 4.2.

Таблиця 4.2 – Оцінка ризику виявлених небезпек

№	Небезпека	Серйоз-ність	Ймовір-ність	Індекс	Ступінь припустимості
1	Пожежа на робочому місці	II	C	2C	небажаний (гранично допустимий)
2	Ураження електричним струмом	I	D	1D	небажаний (гранично допустимий)
3	Зорове перенапруження (освітлення, блискість)	III	B	3B	небажаний (гранично допустимий)
4	Нервово-психічне та емоційне перевантаження	III	C	3C	небажаний (гранично допустимий)
5	Воєнна загроза (обстріли, повітряна тривога)	I	C	1C	неприпустимий

Оцінювання показало, що більшість небезпек належать до категорії небажаного (гранично допустимого) ризику, а воєнна загроза – до неприпустимого рівня, що потребує першочергової уваги. Це означає, що експлуатація об'єкта можлива лише за умови впровадження організаційно-технічних заходів зниження ризику. Запропоновані рекомендації щодо підвищення рівня безпеки та охорони праці на досліджуваному об'єкті наведено у таблиці 4.3.

Таблиця 4.3 – Заходи щодо зниження ризиків

№	Небезпека	Запропонований захід	Очікуваний результат
1	Пожежа	Регулярна перевірка електропроводки, заборона перевантаження мереж, наявність вуглекислотних вогнегасників, дотримання правил пожежної безпеки та плану евакуації	Зниження ймовірності виникнення та наслідків пожежі
2	Ураження електричним струмом	Заземлення обладнання, справна ізоляція, застосування пристроїв захисного відключення, інструктаж з електробезпеки	Запобігання електротравмам
3	Зорове перенапруження	Забезпечення нормованої освітленості згідно з ДБН, усунення відблисків, регульована яскравість екрана, регламентовані перерви для зору	Зменшення зорового навантаження, збереження зору
4	Нервово-психічне перевантаження	Раціональний режим праці та відпочинку, чергування видів робіт, нормування навантаження, сприятливий психологічний клімат	Зниження стресу, підвищення працездатності
5	Воєнна загроза	Облаштування укриття, дотримання сигналів повітряної тривоги, інструктаж з дій у надзвичайних ситуаціях, резервне копіювання даних	Збереження життя і здоров'я працівників, захист майна та інформації

Додатково для зниження впливу психофізіологічних і фізичних факторів доцільно дотримуватися ергономічних вимог до організації

робочого місця оператора ВДТ, підтримувати нормовані параметри мікроклімату [35] та обмежувати рівень електромагнітних випромінювань відповідно до санітарних норм [34].

Висновки до четвертого розділу

У розділі «Охорона праці» розглянуто питання забезпечення безпеки та захисту здоров'я працівників на робочому місці контент-менеджера, що є середовищем експлуатації розробленої автоматизованої системи роботи з медіаконтентом. Метою розділу було виявлення потенційних небезпек на об'єкті проектування, оцінювання ризику їх реалізації та розроблення заходів щодо їх попередження.

Охарактеризовано організаційно-правові основи охорони праці та законодавчу базу України у цій сфері. Наведено характеристику об'єкта проектування та умов праці контент-менеджера, на основі класифікації небезпечних і шкідливих виробничих факторів виявлено дев'ять потенційних небезпек, для кожної з яких визначено джерело та можливі наслідки. Із застосуванням методів «дерева відмов» і матриці оцінювання ризиків оцінено ризик п'яти найбільш значущих небезпек: установлено, що більшість із них належать до небажаного (гранично допустимого) рівня, а воєнна загроза – до неприпустимого.

За результатами оцінювання запропоновано комплекс організаційно-технічних заходів зниження ризику (пожежної та електробезпеки, нормування освітлення й мікроклімату, раціональний режим праці та відпочинку, облаштування укриття), упровадження яких дає змогу знизити рівень професійних ризиків до прийнятного та забезпечити безпечні умови праці на досліджуваному об'єкті.

ВИСНОВКИ

У межах кваліфікаційної роботи поставлено та розв’язано науково-прикладну задачу створення автоматизованої системи збору даних, обробки медіаконтенту та керування публікацією матеріалів у соціальних мережах, побудованої на засадах автоматизації нативного Android-застосунку. Поставлену мету досягнуто: розроблено цілісне програмне рішення, яке поєднує три функціональні напрями, традиційно реалізовувані окремими інструментами, у єдиний керований комплекс з прозорим моніторингом стану в реальному часі. Актуальність роботи підтверджується стійким зростанням ринку маркетингу в соціальних мережах та домінуванням короткого відеоконтенту як основного формату комунікації [25, 26], що формує практичну потребу в інструментах масштабованого керування публікаціями.

Завдання роботи виконано послідовно та повно. Проведено аналіз предметної області, сформульовано функціональні та нефункціональні вимоги за методикою специфікації програмних вимог [3], обґрунтовано вибір технологічного стека та чотиришарову архітектуру системи. Спроєктовано логічну модель даних у нотації UML як діаграму класів з типами полів [4], що охоплює сутності пристрою, застосунку, акаунта, секрета одноразових паролів, публікації, конфігурації та перевизначень опису й папки. Реалізовано програмні компоненти всіх шарів, проведено їх інтеграцію та перевірку працездатності на реальних пристроях.

За функцією збору даних розроблено модуль статистики, що отримує метрики профілів і коротких відео, а саме перегляди, лайки, кількість підписників та охоплення, з використанням анонімних сесій до прикладного програмного інтерфейсу. Стабільність збирання забезпечено ротацією мережевих маршрутів (проксі) для рівномірного розподілу навантаження та підвищення стабільності збору даних. Зібрані показники агрегуються за окремими креативами завдяки журналу публікацій, який зіставляє

опубліковані матеріали з вихідними відеофайлами, а результати експортуються у формат електронних таблиць для подальшого аналізу. У такий спосіб розв’язано задачу формування об’єктивної аналітичної основи для оцінювання ефективності контенту.

За функцією обробки медіаконтенту реалізовано конвеєр підготовки та унікалізації відео, що поєднує геометричні перетворення, а саме поворот і дзеркалення, корекцію кольору, транскодування у формат H.264 та автоматичне формування ескізів. Реалізацію виконано засобами комп’ютерного зору та обробки зображень [6, 7], зокрема з використанням бібліотек OpenCV, NumPy та Pillow [16, 17, 18]. Такий конвеєр забезпечує варіативність вихідних файлів за умови збереження прийнятної якості, що є практично важливим для масової публікації однотипного матеріалу.

За функцією керування публікацією реалізовано автоматизований залив матеріалів формату Reels і Stories на множину акаунтів у трьох режимах роботи: для одного акаунта, для всіх акаунтів застосунку та для кількох застосунків. Передачу відеофайлів на пристрій організовано засобами ADB у відповідний каталог медіатеки, а взаємодію з інтерфейсом застосунку, зокрема введення опису з наперед заданих пресетів, реалізовано через інструментарій автоматизації інтерфейсу [8, 9, 10]. Ключовим результатом є формалізація навігації інтерфейсом у вигляді скінченного автомата [5]: шаблон переходів описує понад 37 відомих екранів, поточний екран визначається за характерними ознаками, а для невідомого екрана залучається локальна мультимодальна модель «зір-мова», яка за скріншотом повертає дію з оцінкою впевненості. Результат дії валідовується, а підтверджений новий екран додається до шаблону, що надає системі властивості адаптивності й самонавчання. Розпізнавання елементів інтерфейсу підкріплено зіставленням шаблонів за нормованою крос-кореляцією з порогом упевненості, а автентифікація акаунтів підтримана генерацією одноразових паролів за стандартом TOTP [20, 23].

За структурою роботи отримано такі результати за розділами. У першому розділі здійснено аналіз предметної області та сформовано вимоги до системи. У другому розділі обґрунтовано архітектуру та спроєктовано модель даних і алгоритми. У третьому розділі описано програмну реалізацію шарів серверної частини на основі FastAPI з підтримкою REST та WebSocket [14, 22], односторінкового інтерфейсу на чистому JavaScript та настільного графічного інтерфейсу на Flet [21], а також шару автоматизації. У завершальних розділах розглянуто питання тестування, охорони праці та безпеки життєдіяльності згідно з чинними нормативними документами [29, 30], а оформлення виконано відповідно до вимог стандартів [27, 28].

Власний внесок автора полягає в постановці задачі інтеграції трьох функціональних напрямів у єдину систему, у проєктуванні чотиришарової архітектури з обмеженням, за яким кожному пристрою відповідає одна задача, та сигналами керування виконанням, у формалізації навігації застосунком як скінченного автомата з резервним механізмом на основі мультимодальної моделі для невідомих екранів, а також у реалізації механізму зіставлення опублікованих матеріалів з вихідними відео для аналітики креативів. Запропоновані рішення спираються на усталені принципи якісного коду та проєктні шаблони [1, 2].

Практична значущість роботи полягає в тому, що створена система зменшує трудомісткість рутинних операцій публікації та збирання статистики, забезпечує масштабованість за рахунок паралельного керування пристроями та надає єдиний інструмент для повного циклу роботи з контентом, від підготовки відео до аналізу результатів публікації. Напрямами подальшого розвитку є міграція сховища з документів у форматі JSON на повноцінну систему керування базами даних [24], розширення переліку підтримуваних соціальних платформ, удосконалення мультимодальної моделі розпізнавання екранів для підвищення автономності, а також додавання модулів прогнозування ефективності креативів на основі накопичених аналітичних даних.

Отримані результати свідчать про наукову новизну та практичну цінність виконаної роботи. Наукова новизна полягає в поєднанні скінченно-автоматної навігації застосунком з адаптивним розпізнаванням невідомих екранів за допомогою локальної мультимодальної моделі, що надає системі властивості самонавчання. Практична цінність зумовлена об'єднанням трьох функцій, а саме збору даних, обробки медіаконтенту та керування публікацією матеріалів, у єдиний керований комплекс, придатний до масштабованого застосування та подальшого впровадження. Сукупність отриманих архітектурних, алгоритмічних і програмних результатів створює завершену основу для розвитку системи й нарощування її функціональних можливостей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мартін Р. Чистий код : створення і рефакторинг за допомогою Agile / Роберт Мартін ; пер. з англ. – Харків : Фабула, 2019. – 416 с.

2. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns : Elements of Reusable Object-Oriented Software. – Reading, MA : Addison-Wesley, 1994. – 395 p.

3. Wiegers K., Beatty J. Software Requirements. – 3rd ed. – Redmond, WA : Microsoft Press, 2013. – 672 p.

4. Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language User Guide. – 2nd ed. – Boston : Addison-Wesley, 2005. – 496 p.

5. Hopcroft J. E., Motwani R., Ullman J. D. Introduction to Automata Theory, Languages, and Computation. – 3rd ed. – Boston : Pearson, 2007. – 535 p.

6. Szeliski R. Computer Vision : Algorithms and Applications. – 2nd ed. – Cham : Springer, 2022. – 925 p.

7. Bradski G., Kaehler A. Learning OpenCV : Computer Vision with the OpenCV Library. – Sebastopol, CA : O'Reilly Media, 2008. – 575 p.

8. Android Debug Bridge (adb) [Електронний ресурс] / Android Developers. – Режим доступу: https://developer.android.com/tools/adb (дата звернення: 04.06.2026).

9. Write automated tests with UI Automator [Електронний ресурс] / Android Developers. – Режим доступу: https://developer.android.com/training/testing/other-components/ui-automator (дата звернення: 04.06.2026).

10. SDK Platform Tools release notes [Електронний ресурс] / Android Developers. – Режим доступу: https://developer.android.com/tools/releases/platform-tools (дата звернення: 04.06.2026).

11. Appium Documentation [Електронний ресурс]. – Режим доступу: https://appium.io/docs/en/latest/ (дата звернення: 04.06.2026).

12. Selenium WebDriver documentation [Електронний ресурс] / Selenium Project. – Режим доступу: https://www.selenium.dev/documentation/webdriver/ (дата звернення: 04.06.2026).
13. Python documentation [Електронний ресурс] / Python Software Foundation. – Режим доступу: https://docs.python.org/3/ (дата звернення: 04.06.2026).
14. FastAPI documentation [Електронний ресурс]. – Режим доступу: https://fastapi.tiangolo.com (дата звернення: 04.06.2026).
15. Pydantic documentation [Електронний ресурс]. – Режим доступу: https://docs.pydantic.dev/latest/ (дата звернення: 04.06.2026).
16. OpenCV documentation. Version 4.x [Електронний ресурс]. – Режим доступу: https://docs.opencv.org/4.x/ (дата звернення: 04.06.2026).
17. NumPy documentation [Електронний ресурс]. – Режим доступу: https://numpy.org/doc/stable/ (дата звернення: 04.06.2026).
18. Pillow (PIL Fork) documentation [Електронний ресурс]. – Режим доступу: https://pillow.readthedocs.io/ (дата звернення: 04.06.2026).
19. Playwright documentation [Електронний ресурс] / Microsoft. – Режим доступу: https://playwright.dev (дата звернення: 04.06.2026).
20. PyOTP : The Python One-Time Password Library [Електронний ресурс]. – Режим доступу: https://pyauth.github.io/pyotp/ (дата звернення: 04.06.2026).
21. Flet documentation [Електронний ресурс]. – Режим доступу: https://flet.dev/docs/ (дата звернення: 04.06.2026).
22. Fette I., Melnikov A. The WebSocket Protocol : RFC 6455 [Електронний ресурс]. – IETF, 2011. – Режим доступу: https://www.rfc-editor.org/rfc/rfc6455.html (дата звернення: 04.06.2026).
23. M'Raihi D., Machani S., Pei M., Rydell J. TOTP : Time-Based One-Time Password Algorithm : RFC 6238 [Електронний ресурс]. – IETF, 2011. – Режим доступу: https://www.rfc-editor.org/rfc/rfc6238.html (дата звернення: 04.06.2026).
24. Connolly T., Begg C. Database Systems : A Practical Approach to Design, Implementation, and Management. – 6th ed. – Boston : Pearson, 2015. – 1440 p.

25. Social Media Management Market : Size and Short-Form Video Trends [Електронний ресурс] / Statista. – Режим доступу: https://www.statista.com (дата звернення: 04.06.2026).
26. The State of Marketing Report : Short-Form Video and Social Media Trends [Електронний ресурс] / HubSpot. – Режим доступу: https://www.hubspot.com/state-of-marketing (дата звернення: 04.06.2026).
27. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. – Київ : ДП «УкрНДНЦ», 2016. – 26 с.
28. ДСТУ ГОСТ 7.1:2006. Система стандартів з інформації, бібліотечної та видавничої справи. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання. – Київ : Держспоживстандарт України, 2007. – 47 с.
29. Про охорону праці : Закон України від 14.10.1992 № 2694-XII [Електронний ресурс] / Верховна Рада України. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2694-12 (дата звернення: 19.06.2026).
30. Кодекс цивільного захисту України від 02.10.2012 № 5403-VI [Електронний ресурс] / Верховна Рада України. – Режим доступу: https://zakon.rada.gov.ua/laws/show/5403-17 (дата звернення: 19.06.2026).
31. Бедрій Я. І. Основи охорони праці користувачів персональних комп'ютерів : навч. посіб. / Я. І. Бедрій. – Тернопіль : Навчальна книга – Богдан, 2014. – 144 с.
32. Бедрій Я. І. Охорона праці та пожежна безпека : навч. посіб. / Я. І. Бедрій. – Тернопіль : Навчальна книга – Богдан, 2013. – 184 с.
33. Природне і штучне освітлення : ДБН В.2.5-28:2018. – Чинний від 2019-03-01. – Київ : ДП «НДІ БК», 2018. – 94 с.
34. Державні санітарні норми і правила при роботі з джерелами електромагнітних полів : ДСанПіН 3.3.6.096-2002. – Чинний від 2002-12-18. – Київ : МОЗ України, 2002. – 24 с.

35. Санітарні норми мікроклімату виробничих приміщень : ДСН 3.3.6.042-99. – Чинний від 1999-12-01. – Київ : МОЗ України, 1999. – 21 с.

36. Системи управління охороною здоров'я та безпекою праці. Вимоги та настанови щодо застосування : ДСТУ ISO 45001:2019. – Чинний від 2021-01-01. – Київ : ДП «УкрНДНЦ», 2019. – 23 с.

37. Малишева В. В. Методичні вказівки до виконання розділу «Охорона праці» в дипломних роботах бакалаврів / уклад. В. В. Малишева. – Харків : ХНУМГ ім. О. М. Бекетова, 2025. – 14 с.

ДОДАТОК А

Псевдокод ключового алгоритму публікації

```
# Псевдокод процесу публікації відео на всі акаунти застосунку
def run_zaliv_all_accounts(device, package, videos, config, callback, stop):
 accounts = load_app_accounts()[package] # акаунти застосунку
 for account in accounts: # 1 пристрій = 1 задача
 if stop(): break # перевірка сигналу зупинки
 switch_to_account(device, account) # перемикання профілю
 for video in pick_videos(account, config): # відбір відео під акаунт
 v = uniqualize(video, config) # унікалізація (поворот/колір)
 push_video_to_device(device, v) # ADB push у DCIM
 open_create_reel(device) # навігація: створення допису
 navigate(device, fsm_template) # скінченний автомат + Vision AI
 enter_description(device, desc(account))# введення опису
 publish(device) # публікація
 log_publication(video, account, package)# запис у журнал (ZalivLog)
```