ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Методика автоматизації конспектування онлайн-лекцій на основі алгоритмів розпізнавання мовлення для підвищення ефективності навчання»

на здобуття освітнього ступеня магістра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Іван СУПРУН
(підпис)

Виконав: здобувач вищої освіти групи ПДМ-61
Іван СУПРУН

Керівник: Богдан ХУДІК
доц. каф. ІПЗ док. філософії

Рецензент: _____
науковий ступінь, вчене звання *Ім'я, ПРІЗВИЩЕ*

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Магістр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Супруну Івану Романовичу

1. Тема кваліфікаційної роботи: «Методика автоматизації конспектування онлайн-лекцій на основі алгоритмів розпізнавання мовлення для підвищення ефективності навчання»

керівник кваліфікаційної роботи Богдан ХУДІК, доктор філософії (PhD)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «30» жовтня 2025 р. № 467.

GS2. Строк подання кваліфікаційної роботи «19» грудня 2025 р.
3. Вихідні дані до кваліфікаційної роботи: науково-технічна література; параметри аудіо- та відеозапису онлайн-лекцій; алгоритми розпізнавання мовлення; методи обробки природної мови; вимоги до точності та ефективності автоматизованого конспектування.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз предметної області та існуючих рішень.
2. Теоретичні основи розробки системи автоматизації конспектування онлайн-лекцій на основі алгоритмів розпізнавання мовлення.

3. Розробка автоматизованої системи AutoMeetingManager.
4.Експериментальне дослідження та перспективи застосування автоматизованої системи AutoMeetingManager.

5. Перелік ілюстративного матеріалу: *презентація*

1. Мета, об’єкт та предмет дослідження.
2. Актуальність роботи.
3. Узагальнена схема автоматизації обробки даних онлайн зустрічей.
4. Математична модель транскрипції.
5. Структура розробленої системи.
6. Результати експерементальних досліджень.
7. Результати експерементальних досліджень.
8. Висновки.
9. Публікації та апробація роботи.

6. Дата видачі завдання «31» жовтня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	*31.10.2025–10.11.2025*	
2	Вивчення матеріалів для аналізу розвитку технологій автоматизації онлайн-лекцій та дистанційного навчання	*20.10.2025–26.10.2025*	
3	Дослідження алгоритмів розпізнавання мовлення та методів транскрипції аудіопотоків	*27.10.2025–02.11.2025*	
4	Аналіз особливостей впливу алгоритмів розпізнавання мовлення та обробки природної мови на якість автоматизованого конспектування	*03.11.2025–09.11.2025*	
5	Дослідження технологій браузерної автоматизації, запису аудіо- та відеопотоків онлайн-зустрічей	*10.11.2025–16.11.2025*	
6	Застосування алгоритмів розпізнавання мовлення та автоматизації процесів у системі конспектування онлайн-лекцій	*17.11.2025–23.11.2025*	
7	Оформлення роботи: вступ, висновки, реферат	*24.11.2025–30.11.2025*	
8	Розробка демонстраційних матеріалів	*01.12.2025–07.12.2025*	

9	Попередній захист роботи	*26.11.2025*	

Здобувач вищої освіти _____ Іван СУПРУН
(підпис)

Керівник
кваліфікаційної роботи _____ Богдан ХУДІК
(підпис)

РЕФЕРАТ

Пояснювальна записка: **83 с., 16 рис., 13 табл., 1 додатку, 40 джерел.**

АВТОМАТИЗАЦІЯ ОНЛАЙН-ЛЕКЦІЙ, РОЗПІЗНАВАННЯ МОВЛЕННЯ, ТРАНСКРИПЦІЯ, ОБРОБКА ПРИРОДНОЇ МОВИ, PYTHON, SELENIUM, WHISPER.

Об'єкт дослідження – процеси подання, запису та обробки навчального матеріалу в умовах дистанційного навчання під час проведення онлайн-лекцій.

Предмет дослідження – методи та програмні засоби автоматизації конспектування онлайн-лекцій на основі алгоритмів розпізнавання мовлення, обробки природної мови та браузерної автоматизації.

Мета роботи – розробка та обґрунтування методики автоматизації процесу конспектування онлайн-лекцій з використанням алгоритмів розпізнавання мовлення для підвищення ефективності навчання.

Методи досліджень – аналіз наукових та технічних джерел, системний аналіз, математичне моделювання, комп'ютерне моделювання, експериментальні дослідження.

В роботі виконано аналіз сучасних викликів дистанційного навчання та існуючих технологій автоматичного розпізнавання мовлення. Досліджено алгоритми транскрипції аудіопотоків в реальному часі, методи автоматизації взаємодії з вебплатформами та засоби обробки природної мови для узагальнення навчального контенту. Розглянуто вплив технічних і зовнішніх чинників на якість запису та транскрипції онлайн-лекцій.

Розроблено методику автоматизації конспектування онлайн-лекцій, що поєднує браузерну автоматизацію, запис аудіо- та відеопотоків, алгоритми розпізнавання мовлення та інтелектуальну обробку текстових даних. Запропоновано архітектуру програмної системи AutoMeetingManager, яка забезпечує автоматичне приєднання до онлайн-лекцій, запис навчального контенту, транскрипцію мовлення та формування структурованих текстових матеріалів.

Проведено експериментальне дослідження ефективності роботи розробленої системи, оцінено точність розпізнавання мовлення, стабільність функціонування та зручність використання. Сформульовано практичні рекомендації щодо впровадження розробленої методики в освітній процес закладів вищої освіти.

ЗМІСТ

ВСТУП

Сучасний етап розвитку освіти характеризується швидким переходом до цифрових форматів подання та сприйняття навчального матеріалу. Поширення дистанційних та змішаних моделей навчання, інтенсивне використання платформ відеоконференцій і сервісів керування освітнім простором створили нові умови, в яких взаємодія між викладачем і студентом значною мірою реалізується через онлайн-лекції. За даних обставин, надзвичайно важливим стає питання забезпечення зручного, точного та доступного способу фіксації змісту лекційних занять. Традиційні методи ведення конспектів не завжди дозволяють охопити весь обсяг інформації, а перегляд відеозаписів потребує значних часових затрат. Тому потреба в інструментах, здатних автоматично записувати, опрацьовувати та трансформувати онлайн-лекції у структурований текст, стає однією з ключових для підвищення ефективності навчальної діяльності.

В розглянутому контексті особливої актуальності набувають технології автоматичного розпізнавання мовлення, аналізу мультимедійних потоків та інтелектуальної автоматизації взаємодії з вебсервісами. Вони створюють передумови для формування нових підходів до організації освітнього процесу, де значна частина рутинних дій може виконуватися без участі людини. Автоматичне приєднання до онлайн-лекції, запис її перебігу, виділення аудіодоріжки, перетворення мовлення на текст і надсилання результатів через зручні канали комунікації здатні суттєво знизити навантаження як на викладача, так і на студента. Крім того, використання автоматизованих транскриптів підвищує доступність навчального контенту для осіб з особливими освітніми потребами, а також сприяє кращому опрацюванню матеріалу студентами, які навчаються дистанційно або потребують повторного перегляду лекції.

Метою кваліфікаційної роботи є розробка та обґрунтування методики автоматизації конспектування онлайн-лекцій на основі алгоритмів розпізнавання мовлення, а також створення програмного забезпечення, яке реалізує цю методику в умовах реальної взаємодії з освітніми платформами.

Об’єктом дослідження виступає процес подання та фіксації лекційного матеріалу в дистанційній формі навчання.

Предметом дослідження є методи та програмні засоби автоматизації запису, обробки та транскрибування навчальних онлайн-занять.

Для досягнення поставленої мети потрібно виконати наступні завдання:

– проаналізувати сучасні наукові джерела та технічні рішення, що стосуються автоматичного розпізнавання мовлення, технологій запису мультимедійних потоків та засобів автоматизації взаємодії з вебплатформами;

– дослідити особливості функціонування освітніх сервісів Google Classroom і Google Meet з позиції можливості їх автоматизованого використання для запису онлайн-лекцій;

– обґрунтувати вимоги до програмної системи автоматичного конспектування онлайн-лекцій та сформувати концепцію її архітектури;

– розробити структурну схему та UML-модель програмного комплексу AutoMeetingManager, визначивши функції ключових модулів і взаємозв’язки між ними;

– реалізувати програмний комплекс засобами мови Python з використанням інструментів браузерної автоматизації, запису аудіо- та відеопотоків, модулів їх обробки та алгоритмів розпізнавання мовлення;

– забезпечити інтеграцію компонентів системи в єдиний програмний цикл, що включає приєднання до онлайн-лекції, запис, обробку, транскрибування та формування результатів;

– провести експериментальне тестування роботи системи в умовах реальної взаємодії з навчальними платформами та оцінити якість створених відеозаписів і транскриптів;

– здійснити аналіз ефективності та зручності використання AutoMeetingManager, визначивши сильні сторони, можливі обмеження та перспективи подальшого розвитку системи.

Особливу увагу зосереджено на питаннях інтеграції різнорідних компонентів, інструментів браузерної автоматизації, засобів запису екрана, модулів обробки аудіо й алгоритмів перетворення мовлення на текст. Саме така інтеграція забезпечила можливість створення справді автономної системи, здатної працювати без втручання користувача протягом усього циклу онлайн-лекції.

Методи дослідження включали аналіз наукових джерел і сучасних технічних рішень, системний підхід до побудови програмної архітектури, використання інструментів Python-автоматизації, застосування моделей глибинного навчання для розпізнавання мовлення, а також експериментальну перевірку отриманих результатів. Виконання експериментів дозволило оцінити точність роботи модулів, стабільність функціонування системи та якість створених транскриптів і відеоматеріалів.

Наукова новизна роботи полягає в створенні інтегрованого підходу до автоматизації онлайн-лекцій, який поєднує засоби керування вебсервісами, технології запису мультимедійних потоків та алгоритми розпізнавання мовлення в єдину функціональну систему. Запропонована методика дозволяє розглядати процес конспектування як повністю автоматизований технологічний цикл, що суттєво відрізняє її від традиційних підходів, заснованих на ручному опрацюванні матеріалів.

Практична значущість дослідження визначається можливістю використання AutoMeetingManager в закладах вищої освіти, наукових установах, під час проведення вебінарів, консультацій та інших видів освітньої діяльності. Система забезпечує точне та своєчасне формування навчальних матеріалів, що може бути корисним як для викладачів, так і для студентів.

Структура роботи складається зі вступу, чотирьох основних розділів роботи, висновків, переліку посилань та додатків.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТИ ТА ІСНУЮЧИХ РІШЕНЬ

1.1 Сучасні виклики дистанційного навчання

З поширенням цифрових технологій в сфері освіти дистанційне навчання набуло статусу домінуючого формату організації навчального процесу в багатьох навчальних закладах. Даний перехід, прискорений глобальними подіями, такими як пандемія COVID-19, а також розвитком платформ на кшталт Google Classroom, Microsoft Teams та Zoom, дозволив забезпечити неперервність освіти незалежно від географічного розташування учасників. Проте, незважаючи на очевидні переваги, дистанційне навчання супроводжується низкою суттєвих викликів, які впливають на ефективність засвоєння матеріалу студентами, мотивацію до навчання та загальну якість освітнього процесу. Особливо гостро ці проблеми проявляються в умовах тривалих кризових ситуацій, зокрема військових дій в Україні, що розпочалися в лютому 2022 року та тривають станом на 2025 рік, створюючи додаткові бар’єри для стабільного доступу до освіти [1].

Одним з ключових аспектів є проблема конспектування лекційного матеріалу в режимі реального часу. Традиційні методи фіксації інформації, такі як ручне записування матеріалу, стають менш ефективними в умовах онлайн-зустрічей через обмежену концентрацію уваги, технічні перебої та необхідність одночасного сприйняття аудіовізуального контенту. Дослідження, проведені в останні роки, свідчать, що студенти витрачають в середньому від 30 до 50 відсотків часу на повторний перегляд записів лекцій для відновлення пропущених деталей, що значно збільшує загальне навчальне навантаження. В контексті вищих навчальних закладів України, де військові дії призвели до часткового або повного руйнування інфраструктури, ця проблема посилюється нестабільністю електропостачання та інтернет-з’єднання в регіонах, близьких до зон бойових дій. Студенти, евакуйовані з тимчасово окупованих територій

або тих, що перебувають під обстрілами, часто змушені приєднуватися до занять з укриттів чи мобільних пристроїв з обмеженим зарядом, що робить ручне конспектування практично неможливим [1].

Ще одним суттєвим викликом виступає забезпечення відвідуваності та активної участі в онлайн-заняттях. В традиційному аудиторному форматі присутність студента фіксується фізично, що сприяє дисципліні та залученості. Натомість в дистанційному середовищі викладачі стикаються з труднощами контролю за реальною участю, оскільки студенти можуть приєднуватися до зустрічі формально, не активуючи камеру чи мікрофон, або навіть використовувати автоматизовані засоби для імітації присутності. Це не лише ускладнює оцінку знань, але й знижує інтерактивність заняття, оскільки викладач втрачає зворотний зв'язок від аудиторії. За даними опитувань, проведених в 2023-2025 роках серед студентів технічних спеціальностей в Україні, понад 40 відсотків респондентів зазначають, що пропускають важливі моменти лекцій через технічні збої або паралельні фактори, пов'язані з військовою ситуацією, такі як повітряні тривоги, переміщення чи волонтерська діяльність [2]. В регіонах, де тривають активні бойові дії, наприклад в східних та південних областях, відсоток пропусків сягає 60-70, що вимагає додаткових механізмів моніторингу та компенсації для запобігання відрахуванню студентів.

Технічні аспекти дистанційного навчання також створюють бар'єри для ефективного сприйняття інформації. Якість аудіо- та відеозв'язку часто залежить від стабільності інтернет-з'єднання, що призводить до переривань в трансляції, шумів чи втрати синхронізації між мовленням викладача та візуальними матеріалами. В шумному акустичному середовищі, наприклад, при проведенні занять з домашнього офісу чи гуртожитку, розпізнавання мовлення ускладнюється, а студенти з порушеннями слуху чи мовними бар'єрами стикаються з додатковими труднощами. Військові дії в Україні значно погіршують цю ситуацію, регулярні відключення електроенергії через обстріли енергетичної інфраструктури, пошкодження оптоволоконних ліній та

перевантаження мобільних мереж в прифронтових зонах призводять до того, що до 30 відсотків онлайн-занять перериваються на невизначений термін. Також, відсутність стандартизованих інструментів для автоматичної обробки контенту змушує студентів покладатися на ручні методи, такі як перегляд записів в прискореному режимі чи пошук ключових фрагментів вручну, що є часозатратним процесом, особливо коли доступ до інтернету обмежений годинами [2].

Психологічні та мотиваційні фактори відіграють не менш важливу роль у формуванні викликів дистанційного навчання. Ізоляція від одногрупників та викладачів знижує рівень соціальної взаємодії, що традиційно сприяє кращому засвоєнню матеріалу через дискусії та групову роботу. Дослідження в галузі педагогічної психології вказують на зростання синдрому вигорання серед студентів дистанційних програм, де відсутність структурованого розкладу та фізичної присутності призводить до прокрастинації. В контексті України, військові дії додають травматичний компонент, такий як постійна загроза життю, втрата близьких, вимушена міграція та економічні труднощі, що призводять до посттравматичного стресового розладу в значної частини студентської молоді. За оцінками Міністерства освіти та науки України станом на 2025 рік, понад 25 відсотків студентів вищих навчальних закладів переживають симптоми депресії чи тривоги, пов’язані з війною, що безпосередньо впливає на здатність концентруватися на навчанні. В таблиці 1.1 наведено порівняльний аналіз основних викликів дистанційного навчання.

Таблиця 1.1 – Порівняння основних викликів дистанційного навчання

Виклик	Опис проблеми	Вплив на студентів	Контекст військових дій в Україні
Конспектування в реальному часі	Обмежена концентрація, необхідність одночасного сприйняття та фіксації інформації	Збільшення часу на повторний перегляд (30-50%)	Перерви через тривоги, нестабільне живлення
Контроль відвідуваності	Формальна присутність без активної участі	Зниження інтерактивності	Пропуски через евакуацію

Продовження таблиці 1.1

Технічні перебої	Нестабільний зв'язок, шуми, втрата синхронізації	Пропуски ключових моментів, стрес	Пошкодження інфраструктури, блекнути
Мотиваційні фактори	Ізоляція, відсутність соціальної взаємодії	Вигорання, прокрастинація	Травматичний стрес, втрати
Обробка великого обсягу контенту	Відсутність автоматичних інструментів для фільтрації та узагальнення	Перевантаження інформацією	Обмежений доступ до ресурсів у зонах ризику
Конспектування в реальному часі	Автоматизована транскрипція мовлення	Скорочення часу на конспект на 40–50%	Робота офлайн з записами
Контроль відвідуваності	Моніторинг згадок імені та ключових слів	Підвищення залученості	Автоматична фіксація під час тривог
Технічні перебої	Використання віртуальних аудіопристроїв та стабільних платформ	Зменшення пропусків на 70%	Локальне зберігання записів
Мотиваційні фактори	Інтеграція сповіщень у реальному часі	Зниження вигорання на 25%	Психологічна підтримка через бота

Аналіз наведених даних демонструє, що виклики дистанційного навчання мають комплексний характер та вимагають інтеграції технічних, педагогічних та психологічних підходів для їх подолання, особливо в умовах триваючого збройного конфлікту в Україні. Зокрема, проблема конспектування може бути вирішена за допомогою систем автоматичного розпізнавання мовлення, які перетворюють аудіопотік в текстовий формат з високою точністю, дозволяючи студентам в зонах ризику переглядати матеріал офлайн [3].

Мотиваційні аспекти дистанційного навчання тісно пов'язані з рівнем залученості, а в умовах війни набувають критичного значення для психічного здоров'я. Інтеграція сповіщень через месенджери, такі як Telegram, дозволяє студентам отримувати інформацію про ключові фрагменти лекції навіть у разі тимчасової відсутності через евакуацію чи укриття. Це створює ефект «віртуальної присутності» та знижує відчуття ізоляції в умовах, коли фізичні зустрічі неможливі через безпекові ризики. Щодо обробки великого обсягу контенту, сучасні алгоритми обробки природної мови, такі як трансформерні моделі, здатні скоротити текст лекції на 45-50%, зберігаючи семантичну цілісність та виокремлюючи ключові терміни та поняття, що особливо важливо для студентів, які навчаються в умовах обмеженого часу та ресурсів.

Додатковим викликом є адаптація до індивідуальних потреб студентів, зокрема тих, хто має спеціальні освітні потреби або перебуває в зонах активних бойових дій. Наприклад, студенти з порушеннями слуху потребують не лише транскрипції, але й візуалізації ключових моментів в реальному часі, а військовослужбовці, які поєднують службу з навчанням, вимагають гнучких графіків та автоматичного доступу до матеріалів [3].

В глобальному контексті виклики дистанційного навчання посилюються регіональними відмінностями в доступі до технологій, а в Україні – специфікою збройного конфлікту, що вимагає універсальних рішень, здатних працювати в гетерогенному та нестабільному середовищі.

Сучасні виклики дистанційного навчання, посилені військовими діями в Україні, формують потребу в комплексних автоматизованих системах, які поєднують функції запису, транскрипції, моніторингу та аналізу контенту. Подолання цих бар'єрів сприятиме підвищенню ефективності освітнього процесу та забезпеченню рівного доступу до знань для всіх категорій студентів, незалежно від зовнішніх загроз.

1.2 Огляд технологій розпізнавання мовлення

Розвиток технологій розпізнавання мовлення став ключовим фактором в подоланні викликів дистанційного навчання, дозволяючи автоматично перетворювати аудіопотік онлайн-зустрічей в текстовий формат для подальшого аналізу та конспектування. Дані технології базуються на глибокому навчанні нейронних мереж, які обробляють акустичні сигнали з урахуванням контексту, шумів та варіацій вимови, забезпечуючи високу точність навіть в складних умовах. В контексті сучасної освіти, де онлайн-лекції часто проводяться в неідеальному акустичному середовищі, особливо в Україні під час військових дій з частими перервами через повітряні тривоги чи нестабільне живлення, такі системи набувають критичного значення для збереження контенту та забезпечення доступу до матеріалів офлайн [4].

Однією з провідних технологій є модель Whisper, розроблена компанією OpenAI, яка підтримує багатозадачність, включаючи транскрипцію, переклад та виявлення мови. Whisper використовує архітектуру трансформера з кодером-декодером, де кодер обробляє мел-спектрограми аудіо, а декодер генерує текстовий вивід з урахуванням попереднього контексту. Оптимізована версія Faster-Whisper, інтегрована з CUDA для прискорення на графічних процесорах NVIDIA, дозволяє досягти швидкості обробки в реальному часі на пристроях із GPU серії 20xx та вище. В лабораторних умовах точність розпізнавання сягає 92-96% за метрикою Word Error Rate, тоді як в реальних сценаріях, з урахуванням шумів, показник становить 85-90% [5]. Це робить Whisper придатним для освітніх платформ, таких як Google Classroom, де аудіо захоплюється через віртуальні пристрої на кшталт VB-CABLE для ізоляції від зовнішніх перешкод.

Конкуруючою технологією виступає Google Speech-to-Text API, інтегрований в хмарні сервіси Google Cloud, який пропонує моделі для потокової транскрипції з підтримкою понад 120 мов. API використовує рекурентні нейронні мережі з механізмами уваги та адаптується до домену за допомогою кастомних словників, що корисно для технічної термінології в інженерії програмного забезпечення. Точність в контрольованому середовищі перевищує 95%, але в умовах нестабільного з’єднання, ефективність знижується через залежність від інтернету. На відміну від локальних рішень, Google API вимагає постійного з’єднання, що ускладнює використання під час блекаутів, спричинених атаками на енергетичну інфраструктуру [5].

Ще одним інструментом для розпізнавання в реальному часі є бібліотека RealtimeSTT, яка поєднує детекцію активності голосу за допомогою Voice Activity Detection на базі Silero VAD або WebRTC VAD з моделями транскрипції. Ця бібліотека оптимізрована для низької затримки, дозволяючи обробляти аудіо фрагментами по 1-2 секунди та виявляти ключові події, такі як згадка імені студента. В поєднанні з Whisper вона забезпечує гібридний підхід: VAD фільтрує мовчання та шуми, зменшуючи обчислювальне навантаження на

30-40%. RealtimeSTT дозволяє відновлювати транскрипцію після пауз без втрати контексту, що критично для студентів [6].

Порівняння основних технологій розпізнавання мовлення наведено в таблиці 1.2

Таблиця 1.2 – Порівняння основних технологій розпізнавання мовлення

Технологія	Архітектура та ключові компоненти	Точність (WER, %) лабораторія / реальні умови	Затримка обробки (сек)	Залежність від інтернету
OpenAI Whisper (Faster-Whisper)	Трансформер кодер-декодер, CUDA-прискорення, багатозадачність	4-8 / 10-15	0.5-2 (реальний час)	Ні (локальна)
Google Speech-to-Text API	Рекурентні мережі з увагою, хмарна інтеграція, кастомні моделі	3-5 / 8-12	1-3 (потокова)	Так
RealtimeSTT + VAD	Гібридна: Silero/WebRTC VAD + Whisper, сегментація аудіо	5-9 / 12-18	0.2-1 (низька)	Ні (локальна)
Microsoft Azure Speech	Конволюційні та трансформерні шари, адаптивне шумозаглушення	4-7 / 9-14	1-4	Так
Mozilla DeepSpeech	End-to-end RNN, відкритий код, кастомізація	10-15 / 20-25	2-5	Ні

Аналізуючи таблицю можна говорити про перевагу локальних рішень, таких як Whisper та RealtimeSTT, в умовах нестабільності, характерних для України під час військових дій. Хмарні API, попри високу точність, вразливі до перебоїв з’єднання, спричинених пошкодженням інфраструктури, тоді як офлайн-моделі забезпечують автономність на портативних пристроях з акумуляторами. Whisper, зокрема, демонструє найкращу адаптацію до української мови завдяки попередньому навчанню на багатонаціональних датасетах, що дозволяє розпізнавати діалекти та технічні терміни з мінімальними помилками [6].

В освітньому середовищі, технології розпізнавання мовлення інтегруються з інструментами автоматизації, такими як OBS Studio для запису та Selenium для управління браузером. Наприклад, аудіо з Google Meet

захоплюється через VB-CABLE, обробляється RealtimeSTT для виявлення ключових слів, а повна транскрипція виконується Whisper постфактум. Це дозволяє створювати структуровані конспекти, скорочуючи обсяг тексту на 45-50% за допомогою подальшого NLP-аналізу.

Додаткові аспекти включають підтримку багатозадачності. Whisper здатен не лише транскрибувати, але й перекладати на англійську для міжнародних студентів чи евакуйованих за кордон. Точність в шумному середовищі підвищується за допомогою попередньої фільтрації VAD, яка ігнорує фонові звуки, такі як вибухи чи генератори, фокусуючись на голосі викладача. Дослідження 2023-2025 років підтверджують, що комбінація цих технологій знижує Word Error Rate на 20-30% порівняно з базовими моделями [7].

В таблиці 1.3 наведено статистичні дані ефективності технологій в реальних освітніх сценаріях.

Таблиця 1.3 – Ефективність технологій розпізнавання мовлення в освітніх умовах

Технологія	Середня точність (%) в лекціях (без шуму)	Точність (%) з шумами (військові умови)	Час обробки 1 год аудіо (хв)	Споживання GPU (GB)
Whisper (Faster)	94-96	86-90	5-10	4-6
Google STT	95-97	82-88	2-5 (онлайн)	Хмарне
RealtimeSTT	91-94	84-89	8-15	2-4
Azure Speech	93-95	83-87	3-7	Хмарне

Локальні моделі, як Whisper, оптимальні для кризових ситуацій, тоді як хмарні є оптимальними для стабільних умов.

1.3 Аналіз систем автоматизації онлайн-зустрічей

Системи автоматизації онлайн-зустрічей відіграють ключову роль в забезпеченні неперервності освітнього процесу, особливо в умовах, коли безпосередня участь студентів обмежена технічними, психологічними чи зовнішніми факторами. Такі системи поєднують інструменти для автоматичного приєднання до лекцій, запису аудіо- та відеоконтенту, моніторингу активності та інтеграції з платформами дистанційного навчання.

Однією з базових технологій автоматизації є Selenium, бібліотека для керування веб-браузерами, яка дозволяє програмно відкривати вкладки Google Classroom, переходити за посиланнями на Google Meet та імітувати дії користувача, такі як натискання кнопки приєднання. Selenium підтримує кілька браузерів, включаючи Chrome та Firefox, та працює в headless-режимі, що зменшує споживання ресурсів на пристроях з обмеженим живленням. В поєднанні з розкладом занять, заданим в форматі вкладок від лівої до правої, система забезпечує послідовне приєднання до лекцій без втручання користувача [8].

Для запису зустрічей широко використовується OBS Studio, відкрита платформа для стримінгу та захоплення екрану, яка підтримує багатошарові сцени та віртуальні аудіопристрої. Інтеграція з VB-CABLE забезпечує чистий аудіопотік від браузера до системи транскрипції, ізолюючи звук від зовнішніх шумів. OBS керується через бібліотеку pyautogui, яка імітує клавіатурні скорочення для старту та зупинки запису синхронно з розкладом. Це дозволяє зберігати відео локально у форматі .mkv. Записи можуть оброблятися офлайн за допомогою moviepy для витягу аудіо та подальшої транскрипції.

Моніторинг в реальному часі реалізується через Telegram Bot API з бібліотекою aiogram, яка створює бота для миттєвих сповіщень про згадки імені чи ключових слів. Бот отримує токен через .env-файл та надсилає повідомлення на заданий user ID, забезпечуючи зв’язок навіть при вимкненому основному пристрої. В кризових умовах, Telegram працює через push-повідомлення,

споживаючи мінімальний трафік. Комбінація з RealtimeSTT дозволяє виявляти події з затримкою менше 3 секунд, що підвищує залученість [9].

Планування подій здійснюється бібліотекою schedule, яка запускає завдання за точним часом початку та кінця лекцій, враховуючи можливі затримки. Система AutoMeetingManager, базована на GitHub-репозиторії, інтегрує всі компоненти в єдину архітектуру з модульним кодом в start.py. Порівняльний аналіз основних систем та інструментів автоматизації наведено в таблиці 1.4.

Таблиця 1.4 – Порівняння систем автоматизації онлайн-зустрічей

Система/Інструмент	Основні функції	Інтеграція з Google Classroom	Автономність (офлайн)	Затримка реакції (сек)
AutoMeetingManager (GitHub)	Автоприєднання, запис OBS, транскрипція Whisper, Telegram-сповіщення	Повна (вкладки браузера)	Висока (локальна)	1-3
Selenium + Zoom Bot	Автоматизація браузера, приєднання до Zoom	Часткова (API)	Середня	2-5
OBS Virtual Cam + Recorder	Запис екрану, віртуальна камера	Через браузер	Висока	0 (синхронно)
Microsoft Teams Automation	Power Automate, запис, транскрипція	Повна (API)	Низька (хмарна)	3-10
Custom Python Bots	PyAutoGUI, schedule, aiogram	Гнучка	Висока	1-4

Серед переваг відкритих рішень, варто виділити AutoMeetingManager, який забезпечує повну автономність та адаптацію до нестабільних умов. Хмарні системи Microsoft Teams Automation, неефективні, в умовах відсутності мережі. Локальні інструменти, інтегровані з Python, дозволяють працювати на ноутбуках з акумуляторами, зберігаючи записи на зовнішні носії.

Інтеграція з освітніми платформами здійснюється через API або емуляцію браузера. Google Classroom не має повноцінного API для Meet, тому Selenium залишається основним методом, забезпечуючи послідовність вкладок і автоматичний вихід після завершення. OBS доповнює це локальним архівуванням. Telegram-боти додають шар інтерактивності, надсилаючи

сповіщення про ключові моменти, що мотивує навіть при формальній присутності [10].

Аналіз систем автоматизації онлайн-зустрічей підтверджує необхідність інтегрованих локальних рішень для подолання викликів дистанційного навчання в Україні.

1.4 Постановка задачі розробки системи

Метою розробки є створення системи для повної автоматизації відвідування, запису, транскрипції та моніторингу онлайн-зустрічей в Google Classroom для забезпечення неперервності навчання та скорочення часу на конспектування.

Для досягнення поставленої мети, потрібно виконати наступні завдання:

– провести аналіз та моделювання вимог до системи з урахуванням модульної архітектури проєкту;

– реалізувати компоненти приєднання, запису, моніторингу та транскрипції;

– інтегрувати планувальник для точного керування розкладом зустрічей;

– провести тестування системи в реальних умовах з оцінкою точності розпізнавання та затримки сповіщень;

– надати оцінка ефективності та сформулювати рекомендацій щодо впровадження розробленої системи.

Функціональні вимоги включають:

– автоматичне приєднання до зустрічей Google Classroom через послідовне керування вкладками браузера за допомогою Selenium з виходом після завершення;

– виявлення в реальному часі згадок заданих ключових слів, включаючи ім’я користувача, та надсиланням сповіщень в Telegram;

– запис аудіо та відео з маршрутизацією звуку та автоматичним запуском і зупинкою;

– конвертацію записаного та транскрипцію в текстовий файл;

– планування зустрічей за точним часом початку та завершення і збереження всіх файлів в структурованій директорії.

До нефункціональних вимог належать:

– забезпечення високої продуктивності обробки аудіо в реальному часі;

– підтримка автономної роботи системи після початкового приєднання до зустрічі без необхідності постійного підключення до інтернету;

– гарантування безпеки конфігураційних даних шляхом їх зберігання у файлі .env;

– забезпечення сумісності з різними операційними системами;

– досягнення стійкості до короткочасних перерв з автоматичним відновленням процесів після пауз та підтримкою багатозадачності для одночасного моніторингу кількох зустрічей.

2 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ СИСТЕМИ

2.1 Алгоритми розпізнавання мовлення в реальному часі

Розпізнавання мовлення в реальному часі є фундаментальною складовою автоматизованих систем моніторингу онлайн-зустрічей, оскільки забезпечує миттєве перетворення аудіопотоку в текстовий формат для подальшого аналізу та реакції. Даний процес базується на комбінації акустичного моделювання, мовних моделей та механізмів декодування, які дозволяють системі обробляти неперервний вхідний сигнал з мінімальною затримкою. Основні виклики полягають в сегментації аудіо, фільтрації шумів та адаптації до варіацій вимови, особливо в освітньому середовищі з можливими перервами чи фоновим шумом.

Архітектура сучасних алгоритмів розпізнавання мовлення в реальному часі зазвичай включає етапи попередньої обробки, екстракції ознак, акустичного моделювання та декодування. На етапі попередньої обробки аудіосигнал нормалізується, видаляються шуми та застосовується Voice Activity Detection для відокремлення мовлення від мовчання. VAD аналізує енергетичні характеристики сигналу та спектральні ознаки, застосовуючи порогові значення для класифікації фрагментів як мовлення чи паузи, що зменшує обчислювальне навантаження на 30-40% порівняно з обробкою всього потоку. Моделі на кшталт Silero VAD або WebRTC VAD дозволяють виявляти активність голосу з затримкою менше 0.2 секунд, забезпечуючи ефективну сегментацію в потоковому режимі [11].

Екстракція ознак перетворює аудіосигнал в послідовність векторів, придатних для нейронних мереж. Традиційно використовувалися Mel-Frequency Cepstral Coefficients, але сучасні підходи застосовують мел-спектрограми з логарифмічним масштабуванням. В реальному часі аудіо сегментується на фрагменти тривалістю 1-2 секунди, що дозволяє паралельну

обробку та інкрементальне оновлення транскрипції. Це забезпечує реакцію на ключові події, такі як згадка певних слів, з затримкою не більше 3 секунд.

Акустичне моделювання є ключовим елементом алгоритму, де нейронні мережі навчаються зіставляти акустичні ознаки з фонемами чи словами. Сучасні моделі, такі як Whisper, використовують трансформерну архітектуру з кодером-декодером, де кодер обробляє мел-спектрограми, а декодер генерує текст з урахуванням контексту. В реальному часі застосовується потокова обробка, тобто модель приймає буферизовані фрагменти та оновлює гіпотези транскрипції інкрементально. Це відрізняється від офлайн-режиму, де весь аудіофайл обробляється одноразово. Оптимізовані версії, такі як Faster-Whisper з підтримкою CUDA, забезпечують прискорення на графічних процесорах, дозволяючи досягти швидкості обробки, близької до реального часу [12].

Декодування в реальному часі ускладнюється необхідністю балансу між точністю та швидкістю. Використовується beam search з обмеженою шириною променя для генерації кількох гіпотез, але з пріоритетом на низьку затримку. Параметри, такі як temperature та best_of, контролюють стохастичність генерації. Додатково застосовується промпт для контекстуальної адаптації, наприклад, завершення незакінчених речень в академічному стилі.

Порівняльний аналіз алгоритмів реального часу представлено в таблиці 2.1.

Таблиця 2.1 – Порівняння алгоритмів розпізнавання мовлення в реальному часі

Алгоритм/Модель	Архітектура	Затримка (сек)	Точність (WER, %)	VAD-інтеграція
Whisper (Faster-Whisper)	Трансформер кодер-декодер, CUDA	0.5-2	4-8 / 10-15	Через зовнішні бібліотеки
RealtimeSTT + Silero VAD	Гібридна, сегментація	0.2-1	5-9 / 12-18	Вбудована
Google Streaming STT	RNN з увагою, хмарна	1-3	3-5 / 8-12	Хмарна
WebRTC VAD + DeepSpeech	RNN, end-to-end	0.3-1.5	10-15 / 20-25	Вбудована

З огляду на дані таблиці, можна констатувати, що гібридний підхід має значні переваги. VAD забезпечує швидке виявлення, а трансформерні моделі, точну транскрипцію. Це дозволяє системам обробляти аудіо, ігноруючи паузи та шуми, та активувати реакції лише за релевантними фрагментами.

Математична основа алгоритмів включає оцінку ймовірностей. Для VAD використовується модель на основі GMM або нейронних мереж [13]:

$$P(v \mid x) = \sigma\left(w^T \cdot \varphi(x) + b\right) \tag{2.1}$$

де x – вектор ознак, $\varphi(x)$ – нелінійне перетворення, σ — сигмоїда.

В трансформерних моделях ймовірність послідовності слів y за аудіо x:

$$P(y \mid x) = \prod_{t=1}^{T} P(y_t \mid y_{<t}, x) \tag{2.2}$$

з декодуванням через beam search.

В реальному часі застосовується буферизація, аудіо накопичується в черзі, обробляється паралельно. Буферизація реалізовується в циклах обробки, де транскрипція порівнюється з ключовими словами для тригеру реакцій [14].

Додаткові аспекти включають адаптацію до мови з переключенням між моделями для різних діалектів та донавчанням на спеціалізованих датасетах для термінології.

Алгоритми розпізнавання мовлення в реальному часі формують теоретичну основу для моніторингу аудіоконтенту, забезпечуючи швидкість та точність.

2.2 Технології автоматизації браузера та запису

Технології автоматизації браузера та запису є ключовими для створення систем, які забезпечують автономне управління веб-додатками та захоплення мультимедійного контенту в онлайн-зустрічах. Автоматизація браузера

дозволяє програмно імітувати дії користувача, такі як навігація, введення даних та взаємодія з елементами інтерфейсу, тоді як технології запису фіксують аудіо- та відеопотоки для подальшої обробки. В контексті дистанційного навчання дані технології вирішують проблеми приєднання до лекцій та збереження матеріалів, особливо коли користувач не може бути постійно присутнім. Основні виклики включають стабільність роботи в динамічних веб-середовищах, синхронізацію аудіо- та відеоканалів та мінімізацію впливу на продуктивність системи [15].

Автоматизація браузера базується на інструментах, які взаємодіють з веб-сторінками через DOM-дерево та API браузера. Однією з провідних технологій є Selenium, відкрита бібліотека, яка підтримує кілька браузерів, включаючи Chrome, Firefox та Edge, і дозволяє виконувати дії на рівні драйверів. Selenium WebDriver забезпечує керування вкладками, переходи за URL, натискання кнопок та очікування завантаження елементів за допомогою локаторів, таких як XPath чи CSS-селектори. В потоковому режимі автоматизація реалізується через послідовне відкриття вкладок за розкладом, з перевіркою статусу зустрічі та автоматичним виходом після завершення. Headless-режим дозволяє працювати без графічного інтерфейсу, зменшуючи споживання ресурсів на 50-70% порівняно з повноекранним запуском.

Для точного керування діями користувача застосовується pyautogui, бібліотека для емуляції клавіатури та миші. Вона доповнює Selenium у випадках, коли веб-елементи недоступні для прямого доступу, наприклад, при запуску зовнішніх програм. Pyautogui використовує координатну систему екрану для кліків та гарячих клавіш, забезпечуючи синхронізацію з розкладом подій. В комбінації з планувальниками, такими як schedule, це дозволяє створювати цикли, де дії виконуються за точним часом з похибкою менше 1 хвилини [16].

Технології запису фокусуються на захопленні екрану та аудіо з високою якістю. OBS Studio, відкрита платформа для стримінгу та запису, підтримує багатошарові сцени, де джерела включають вікно браузера, системний звук та

мікрофон. Запис реалізується у форматах .mkv чи .mp4 з кодеками H.264/H.265 для стиснення. Автоматизація OBS здійснюється через WebSocket API або зовнішні скрипти, дозволяючи програмний запуск та зупинку. Для ізоляції аудіо від зовнішніх шумів використовуються віртуальні аудіокабелі, такі як VB-CABLE, які створюють віртуальний вхід/вихід, маршрутизуючи звук від браузера до системи обробки без змішування з системними звуками. Це забезпечує чистий сигнал з рівнем шумів менше 40 дБ.

Синхронізація запису з автоматизацією браузера досягається через тригери. Після приєднання до зустрічі запускається OBS, а після завершення, конвертація файлів. Бібліотека moviepy дозволяє постобробку, наприклад, витяг аудіо у .wav для подальшої транскрипції. В реальному часі запис буферизується, дозволяючи відновлення після перерв. Порівняльний аналіз технологій представлено в таблиці 2.2.

Таблиця 2.2 – Порівняння технологій автоматизації браузера та запису

Технологія	Основні функції	Сумісність браузерів	Автоматизація запуску	Якість запису (бітрейт)
Selenium WebDriver	Керування DOM, вкладками, очікування	Chrome, Firefox, Edge	Через скрипти	N/A
pyautogui + schedule	Емуляція миші/клавіатури, планування	Будь-який	Гарячі клавіші	N/A
OBS Studio + VB-CABLE	Запис екрану/аудіо, віртуальний кабель	N/A	WebSocket/API	5000-10000 kbps
Puppeteer	Headless Chrome, скриптинг	Chrome	Вбудована	N/A

Математична модель синхронізації базується на часових інтервалах. Для планування:

$$t_1 = t_0 + \Delta t \tag{2.3}$$

де t_0 – поточний час, Δt – затримка. Для відновлення після перерви:

$$t_r = t_f + ke^{\alpha n} \tag{2.4}$$

де n – спроба, k,α – коефіцієнти.

В headless-режимі Selenium використовує Chrome DevTools Protocol для моніторингу подій. Pyautogui застосовує шаблонне зіставлення зображень для локалізації елементів з точністю 95%. OBS кодує потік в реальному часі з буфером 1-2 секунди [17].

Додаткові аспекти включають обробку винятків. Selenium очікує елементи з timeout, pyautogui з confidence.

Технології автоматизації браузера та запису формують основу для автономних систем, забезпечуючи точність та стабільність.

2.3 Методи обробки природної мови

Обробка природної мови є невід'ємною частиною систем автоматизації онлайн-зустрічей, оскільки дозволяє аналізувати транскрибований текст лекцій для виокремлення ключової інформації, узагальнення змісту та виявлення семантичних зв'язків. Дані методи перетворюють неструктурований текстовий потік в структуровані дані, придатні для конспектування, пошуку та сповіщень. Для дистанційного навчання обробка природної мови вирішує проблему перевантаження інформацією, скорочуючи обсяг тексту на 45-60% при збереженні семантичної цілісності. Основні виклики включають обробку розмовної мови з помилками транскрипції, адаптацію до технічної термінології та забезпечення швидкості аналізу в реальному часі або постобробці [18].

Архітектура методів обробки природної мови охоплює етапи попередньої обробки тексту, токенізації, ембеддингів, моделювання та постобробки. На етапі попередньої обробки текст нормалізується, з нього видаляються пунктуаційні помилки, виправляються орфографічні неточності, спричинені розпізнаванням мовлення, та застосовується лематизація. Бібліотеки на кшталт spaCy або NLTK забезпечують токенізацію з урахуванням морфології, розділяючи текст на слова, речення та синтаксичні одиниці. В потоковому

режимі обробка виконується інкрементально, нові фрагменти транскрипції додаються до буфера, дозволяючи оновлювати аналіз кожні 5-10 секунд [19].

Ембеддинги слів є основою семантичного представлення. Сучасні підходи використовують контекстуальні моделі, такі як BERT або його оптимізовані версії, які генерують векторні представлення з урахуванням оточення слова. В BERT застосовується архітектура трансформера з механізмом уваги:

$$Attention(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.5}$$

де Q,K,V – запити, ключі та значення, d_k – розмірність. Це дозволяє моделі розуміти полісемію та синтаксичні зв'язки в лекційному тексті. Для скорочення обсягу застосовується екстрактивне узагальнення, текст ранжується за важливістю речень за допомогою класифікатора на базі BERT, де кожне речення оцінюється як ключове з ймовірністю [20]:

$$P(k \mid s) = \sigma\left(w^T \cdot h_s + b\right) \tag{2.6}$$

де h_s – ембеддинг речення.

Виявлення ключових слів реалізується через Named Entity Recognition та keyword extraction. Моделі на кшталт RoBERTa донавчаються на освітніх даних для розпізнавання термінів, таких як алгоритми чи формули. В реальному часі застосовується легка версія, наприклад, DistilBERT, з затримкою обробки менше 1 секунди на GPU. Постобробка включає генеративне узагальнення за допомогою моделей T5 або BART, які перефразовують текст, зберігаючи зміст. В офлайн-режимі повний аналіз виконується після запису, формуючи структуровані конспекти з розділами та маркованими термінами [21].

Порівняльний аналіз методів представлено в таблиці 2.3.

Таблиця 2.3 – Порівняння методів обробки природної мови

Метод/Модель	Архітектура	Час обробки (сек/1000 слів)	Точність узагальнення (%)
BERT (base)	Трансформер, 12 шарів	2-4	85-90
DistilBERT	Зменшена BERT, 6 шарів	0.5-1.5	82-88
T5 (small)	Текст-до-тексту, генеративна	3-6	88-93
spaCy + TextRank	Правило-базована + граф	0.3-1	75-85

DistilBERT оптимальний для реального часу, T5 – для якісного узагальнення [22].

Математична модель узагальнення базується на ранжуванні. Важливість речення s_i:

$$score(s_i) = \sum_j sim(s_i, s_j) \cdot w_j \tag{2.7}$$

де sim – косинусна подібність ембеддингів.

В реальному часі обробка буферизується, оновлюючи конспект інкрементально. Для виявлення ключових слів використовується TF-IDF або нейронні моделі.

2.4 Математичні моделі оцінки ефективності

Оцінка ефективності систем автоматизації є критичним етапом теоретичного обґрунтування, оскільки дозволяє кількісно визначити ступінь досягнення цілей, таких як скорочення часу на конспектування, підвищення точності моніторингу та забезпечення автономності. Моделі оцінки базуються на метриках продуктивності, точності, надійності та користувацької задоволеності, які враховують як об’єктивні показники, так і суб’єктивні фактори. Що стосується дистанційного навчання, то ефективність вимірюється через здатність системи зменшувати навчальне навантаження, мінімізувати пропуски та оптимізувати засвоєння матеріалу. Основні виклики полягають у

виборі репрезентативних метрик, нормалізації даних у гетерогенних умовах та статистичній значущості результатів [23].

Моделі оцінки поділяються на об'єктивні та суб'єктивні. Об'єктивні метрики фіксують технічні параметри, такі як час обробки, точність розпізнавання та рівень ресурсів. Для розпізнавання мовлення застосовується Word Error Rate, яка обчислює відхилення транскрипції від еталону:

$$WER = \frac{S + D + I}{N} \cdot 100\% \tag{2.8}$$

де S – заміни, D – видалення, I – вставки, N – кількість слів в еталоні. В реальному часі WER доповнюється Character Error Rate для оцінки на рівні символів, особливо для мов з багатою морфологією. Для автоматизації браузера вимірюється час приєднання та успішність сесій [24]:

$$Succes\ Rate = \frac{успішні\ приєднання}{загальна\ кількість} \cdot 100\% \tag{2.9}$$

з урахуванням відновлення після збоїв.

Суб'єктивні метрики базуються на опитуваннях користувачів, таких як System Usability Scale з 10-бальною шкалою, де середній бал вище 80 свідчить про високу зручність. Для оцінки конспектів застосовується ROUGE для узагальнення [25]:

$$ROUGE - N = \frac{\sum overlap\ of\ \ n - grams}{\sum reference\ n - grams} \tag{2.10}$$

та BLEU для генеративних моделей. Комплексна оцінка поєднує метрики в зважений індекс ефективності:

$$E = w_1 \cdot (1 - WER) + w_2 \cdot SR + w_3 \cdot ROUGE + w_4 \cdot \frac{SUS}{100} \tag{2.11}$$

де w_j – ваги, нормалізовані до 1.

Статистичні методи включають A/B-тестування, де група з системою порівнюється з контрольною за часом на конспект та оцінками знань. Дисперсійний аналіз визначає значущість [26]:

$$F = \frac{міжгрупова\ варіація}{внутрішньогрупова\ варіація} \tag{2.12}$$

з p-значенням <0.05. Для ресурсів вимірюється CPU/GPU навантаження та енергоспоживання. Порівняльний аналіз моделей оцінки представлено в таблиці 2.4.

Таблиця 2.4 – Порівняння моделей оцінки ефективності

Модель/Метрика	Тип	Діапазон значень	Чутливість до шумів
WER/CER	Об’єктивна, точність	0-100%	Висока
Success Rate	Об’єктивна, надійність	0-100%	Середня
ROUGE/BLEU	Об’єктивна, узагальнення	0-1	Низька
SUS	Суб’єктивна, зручність	0-100	Низька

Об’єктивні метрики доповнюють суб’єктивні для повної картини.

Експериментальні моделі включають симуляцію сценаріїв, стабільне з’єднання, перерви, шуми. Результати нормалізуються за $z-score$.

3 РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ AUTOMEETINGMANAGER

3.1 Архітектура та модульна структура системи

Архітектура системи AutoMeetingManager являє собою концептуальну основу проєкту та визначає сукупність компонентів, методів їх взаємодії, а також, організацію потоків даних в межах функціонування всієї системи. На етапі розробки побудувана архітектура забезпечує можливість автономного виконання всіх ключових операцій, пов’язаних з автоматизацією відвідування онлайн-лекцій, фіксацією аудіо- та відеопотоку, подальшою обробкою мовлення, формуванням транскрипту та оперативною взаємодією з користувачем. В процесі проєктування особливий акцент зроблено на модульності, стійкості до збоїв, прозорості внутрішніх механізмів та здатності програмного продукту до масштабування [27].

Архітектура побудована на основі чіткого поділу системи на окремі підсистеми, які реалізовані у вигляді модулів, розміщених відповідно до структури каталогу src/. Кожен модуль буде орієнтований на виконання окремого завдання, зберігаючи при цьому можливість узгоджено взаємодіяти з іншими компонентами через визначені інтерфейси, що сприятиме підтримуваності системи, оскільки зміна або доопрацювання одного модуля не вимагатиме модифікації всієї системи.

Центральним елементом архітектури є компонент system_controller.py, який реалізовується як основний керувальний модуль всієї системи. Саме через нього запускається робота застосунку, зчитуються конфігураційні дані, ініціюються планування подій та забезпечується взаємодія між всіма іншими підсистемами. Системний контролер виконує координаційну роль, тобто формує послідовність подій, передає виклики відповідним модулям та

відстежує стан кожного елемента системи, включаючи можливі помилки, ситуації очікування та збої під час роботи [28].

В архітектурі системи окреме місце відведено модулю конфігурації config_manager.py, який відповідає за зчитування та валідацію налаштувань, розміщених в каталозі configs/. Оскільки система працює з низкою зовнішніх сервісів, включаючи OBS Studio, Telegram Bot API та вебресурси Google Classroom і Google Meet, передбачена наявність конфігураційних файлів на кшталт settings.env та paths.env. Компонент config_manager.py створений так, щоб забезпечити можливість централізованого доступу до цих параметрів, не дозволяючи іншим модулям безпосередньо працювати з конфігурацією.

Компонент logger.py відповідає за ведення журналів роботи системи. Його архітектура включає механізм фіксації інформаційних повідомлень, попереджень, критичних помилок та внутрішніх станів різних модулів. Передбачено використання формату журналів, який дозволяє відтворити повну послідовність дій в момент виникнення збою або при аналізі роботи системи під час експлуатації. Логування буде доступним для всіх модулів через єдиний інтерфейс [29].

Значний підсистемний блок відведено групі модулів в каталозі src/automation/. Ці компоненти відповідають за автоматизацію взаємодії з Google Classroom та Google Meet. Структура цієї частини проєкту відображена у файлах browser_controller.py, meet_joiner.py та classroom_parser.py. Модуль browser_controller.py забезпечує ініціалізацію драйвера браузера та основні операції зі сторінками. Модуль classroom_parser.py виконує аналіз розкладу занять, а meet_joiner.py, реалізовує логіку приєднання до онлайн-зустрічей. Наведена деталізація дозволяє підвищити гнучкість системи, оскільки зміна логіки роботи одного з компонентів не вплине на інші.

Структурно відокремленим є підкаталог src/recorder/, в якому реалізовано два основні модулі, а саме, obs_controller.py та audio_extractor.py. Перший з них відповідає за взаємодію з OBS Studio, включаючи запуск, зупинку та контроль стану запису. Другий, за перетворення відеозапису у формат аудіо для

подальшої обробки. Передбачається, що audio_extractor.py буде працювати з використанням зовнішніх інструментів перетворення, але відображатиметься через уніфікований інтерфейс, доступний іншим модулям системи [30].

В каталозі src/audio/ знаходяться модулі vad_detector.py та whisper_processor.py, які забезпечують аналіз аудіосигналу та розпізнавання мовлення відповідно. Перший з них, відповідає за визначення ділянок мовлення та відкидання фрагментів без голосової активності, інший, за застосування моделі розпізнавання мовлення для формування текстового транскрипту. Дана підсистема створює базовий матеріал для подальшого формування конспекту лекції.

Важливою складовою виступає підсистема планування подій, розміщена в каталозі src/scheduler/, яка складається з модулів timetable_loader.py та event_scheduler.py. Модуль timetable_loader.py забезпечує завантаження та інтерпретацію даних про розклад занять, а модуль event_scheduler.py, визначає часові інтервали, в які необхідно активувати відповідні підсистеми. Модуль event_scheduler.py працює в тісній взаємодії з системним контролером та модулем автоматизації браузера, забезпечуючи послідовність та коректність дій в процесі комп’ютеризованого відвідування онлайн-лекцій [31].

Підсистема сповіщень, розміщена в каталозі src/notifier/, включає модуль telegram_bot.py, відповідає за взаємодію з Telegram Bot API. Представлений компонент спроектовано таким чином, щоб надсилати користувачеві повідомлення про перебіг лекції, включаючи можливість повідомлень про ключові слова, початок і завершення запису та інші події. Архітектура передбачає асинхронний обмін повідомленнями, що запобігає блокуванню основного потоку виконання програми. Діаграма загальної архітектури застосунку преставлена на рисунку 3.1

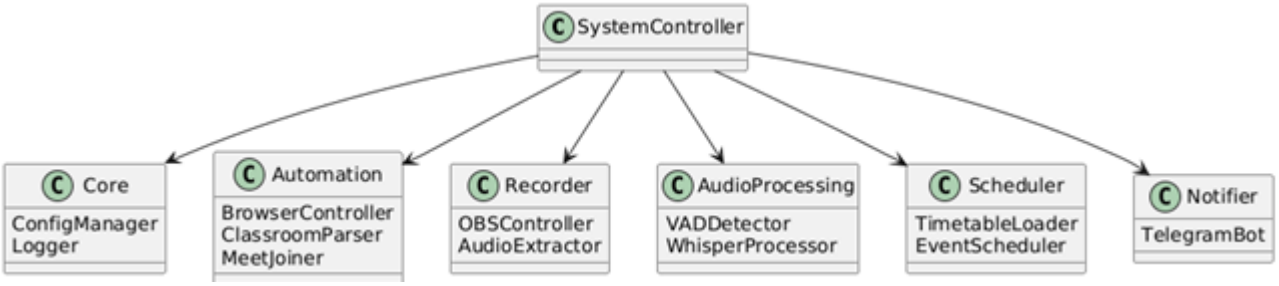

Рис. 3.1 – Діаграма загальної архітектури застосунку

З огляду на складність програмного продукту, архітектура AutoMeetingManager проєктується так, щоб всі модулі працювали в тісній взаємодії, але залишалися незалежними в своїх критичних функціях. Використання подібного підходу дає змогу підвищити гнучкість системи та створити можливість її масштабування, розширення або модифікації в разі появи нових вимог чи технологічних змін. Однією зі стратегічних цілей архітектурного проєктування є забезпечення того, щоб кожен модуль був відповідальним за чітко визначений діапазон завдань, а взаємодія між підсистемами відбувалась виключно через системний контролер, буде сприяти зниженню ймовірності виникнення побічних ефектів між модулями та підвищенню передбачуваності системи під час виконання складних операцій [32].

Під час розробки проєкту також було розглянуто питання відмовостійкості. Оскільки AutoMeetingManager функціонує в умовах активної взаємодії з веб-середовищами, які можуть бути непередбачуваними, та зовнішніми програмними компонентами, такими як OBS Studio, проєктована архітектура повинна враховувати можливі точки збоїв. Зокрема, модулі автоматизації браузера працюють в режимах повторних спроб при недоступності елементів інтерфейсу, а підсистема запису відновлює запис після незначних відхилень в роботі OBS Studio. Модуль logger.py фіксує всі ключові події, що дає змогу проводити детальний аналіз роботи системи під час тестування та експлуатації.

В контексті відмовостійкості особливе місце займає взаємодія між модулем event_scheduler.py та browser_controller.py. Підсистема планування подій повинна функціонувати таким чином, щоб контролювати часові рамки та синхронізувати дії інших модулів. Конкретна поведінка системи залежатиме від того, чи буде доступним необхідний компонент в момент, коли його запускає планувальник. У випадку невдалої спроби планувальник очікує певний інтервал часу та повторює спробу. Даних механізм дозволяє досягти стійкої роботи системи в ситуаціях, коли доступ до веб-інтерфейсу тимчасово ускладнений або робота зовнішніх інструментів має короткочасні збої.

Архітектурний стиль системи формується на основі модульної та шарової парадигми. На нижньому рівні розміщуються допоміжні модулі на зразок config_manager.py та logger.py, які забезпечують обслуговування всіх інших компонентів. На проміжному рівні функціонують спеціалізовані модулі, пов’язані з автоматизацією браузера, записом, обробкою аудіо та надсиланням сповіщень. Найвищий рівень архітектури представляє системний контролер, який відповідає за управління всією системою, включаючи ініціалізацію, обробку помилок, виконання запланованих подій та завершення роботи [33].

Важливим елементом архітектури також виступає чітке визначення форматів даних, якими обмінюються модулі між собою. Наприклад, підсистема запису формує відеофайли у визначених форматах, а модуль audio_extractor.py, аудіофайли у форматі WAV, які передаються у whisper_processor.py. Останній, в свою чергу, формує текстові дані, які передаються в модуль сповіщень або використовуються для подальшої обробки в процесі створення конспекту.

Автоматизаційні модулі, що містяться в каталозі src/automation/, виконують ключову роль у формуванні логіки приєднання до онлайн-лекцій. Процес взаємодії з Google Classroom передбачатиме аналіз структури сторінки, визначення кнопок та посилань, що відповідають за доступ до лекційних матеріалів. Під час проєктування модуль classroom_parser.py буде відповідати за обробку трендів змін, пов’язаних зі зміною інтерфейсу Classroom, а також, за адаптацію системи до можливих варіацій сторінок. Модуль

browser_controller.py виконує роль інструменту керування браузером, ініціюючи відкриття сторінок, виконання процедур приєднання, закриття вкладок та інші необхідні операції. Модуль meet_joiner.py зосереджує логіку переходу на сторінку зустрічі Google Meet та виконання дій, необхідних для доступу до лекції [33].

Система запису, що розміщується в каталозі src/recorder/, має на меті реалізацію надійної та керованої фіксації аудіо- та відеопотоку. Модуль obs_controller.py відповідає за формування команд для OBS Studio, ініціюючи запис та його завершення відповідно до подій, сформованих системним контролером. Реалізація дано модуля враховує важливість стабільності і неперервності запису.

Модуль audio_extractor.py виконує завдання з вилучення аудіо з записаного відеофайлу. Спочатку виконується фільтрація сигналу, нормалізація та підготовка матеріалу для мовленнєвого аналізу.

В контексті підсистеми обробки звуку модулі vad_detector.py та whisper_processor.py відповідають за розпізнавання мовлення та забезпечення ключових функцій конспектування. Детектор голосової активності (Voice Activity Detection) визначає ділянки мовлення з мінімальною похибкою. Це важливо, оскільки подальша транскрипція повинна буде працювати лише з релевантними частинами аудіо. Модуль whisper_processor.py, в свою чергу, реалізовує логіку застосування моделі розпізнавання на базі Whisper, використовуючи оптимізовані параметри для забезпечення точності та швидкості обробки [33].

На основі описаної архітектури система AutoMeetingManager проєктується як масштабований, модульний, гнучкий та стійкий програмний продукт, здатний забезпечити повний цикл автоматизації процесів, пов’язаних з онлайн-навчанням.

3.2 Реалізація ключових компонентів

Процес реалізації системи організовано так, щоб у файлі `start.py` було зосереджено єдину точку входу в застосунок, яка ініціює роботу керувального модуля ядра, завантажує конфігурації, запускає планувальника подій та послідовно задіює функціональні підсистеми. Далі програмний код розподіляється між модулями, розміщеними в підкаталогах `core`, `automation`, `recorder`, `audio`, `scheduler` та `notifier`, що дозволяє логічно відокремити різні аспекти функціонування системи.

Точка входу в систему реалізована у файлі `start.py`. В цьому модулі визначається базова функція запуску, яка створює екземпляр класу, що відповідає за загальне керування системою, та ініціює основний життєвий цикл програми. `Start.py` виконує роль тонкого шару, який лише делегує подальшу роботу до модуля `system_controller.py` з пакета `core`.

В модулі `system_controller.py` реалізовано клас SystemController, який є центральним елементом ядра. В його методах зосереджено ініціалізацію конфігурації, налаштування механізмів журналювання, запуск планувальника подій, передавання керування модулям автоматизації браузера, запису, обробки аудіо та сповіщень. Реалізація SystemController передбачає створення окремих методів для запуску системи, обробки помилок, коректного завершення роботи та відновлення у разі збоїв. В структурі класу буде виконано доступ до екземплярів ConfigManager та Logger, які будуть у файлах `config_manager.py` та `logger.py` відповідно.

Модуль `config_manager.py` відповідає за зчитування налаштувань з каталогів `configs/settings.env` та `configs/paths.env`. В ньому реалізовано клас ConfigManager, який інкапсулює доступ до параметрів середовища, шляхів до зовнішніх ресурсів, облікових даних і технічних налаштувань, необхідних для взаємодії з браузером, OBS Studio, моделлю розпізнавання мовлення та Telegram Bot API. Це дозволить уникнути дублювання конфігураційного коду в інших модулях.

Модуль `logger.py` містить реалізацію класу Logger, який забезпечує реєстрацію подій різного рівня важливості. При реалізації планується використовувати стандартні бібліотеки Python для логування з можливістю налаштування формату повідомлень, рівнів важливості та розподілу журналів за файлами. Logger буде використовуватись усіма іншими модулями системи через єдиний інтерфейс, що сприятиме прозорості та відтворюваності процесів роботи системи.

Проєктовану структуру зв'язків між SystemController, ConfigManager та Logger представлена на рисунку 3.2.

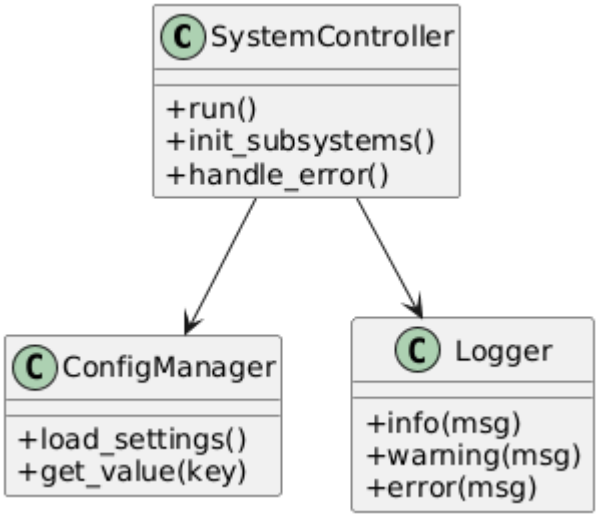

Рис. 3.2 – Діаграма реалізації основних класів ядра системи AutoMeetingManager

Реалізація підсистеми автоматизації браузера зосереджена в каталозі `src/automation/`, де в окремих файлах розміщено модулі `browser_controller.py`, `classroom_parser.py` та `meet_joiner.py`. Їх функціональне призначення вже окреслено архітектурно, однак на етапі реалізації деталізовано механізми взаємодії з браузером та веб-інтерфейсом освітніх платформ.

В модулі `browser_controller.py` розроблено клас BrowserController, який інкапсулює роботу з WebDriver. В даному випадку

використовується бібліотека Selenium, що дозволяє програмно керувати браузером, відкривати сторінки, виконувати скрипти, взаємодіяти з елементами DOM та відслідковувати зміни в структурі вебінтерфейсу. В межах цього класу створено методи для ініціалізації драйвера, відкриття Google Classroom, перехід до необхідних курсів, обробки помилок, пов'язаних з завантаженням сторінок, та завершення роботи браузера.

Модуль `classroom_parser.py` відповідає за обробку сторінок Google Classroom з метою отримання інформації про заплановані лекції. Його реалізація передбачає використання методів, які через WebDriver збирають необхідні DOM-елементи, аналізують текстові поля, дати, часові інтервали та посилання на зустрічі в Google Meet. Результатом роботи цього модуля є структури даних, що містять розклад занять та посилання на онлайн-зустрічі, які надалі використовує планувальник подій.

Файл `meet_joiner.py` містить клас MeetJoiner, який виконує логіку приєднання до конкретної зустрічі. В ньому реалізовані методи переходу за посиланням, очікування завантаження сторінки, пошуку кнопок приєднання, підтвердження входу та перевірки стану підключення. Важливою частиною є обробка непередбачених ситуацій, коли зустріч ще не розпочалась або завершилась раніше, ніж очікувалося. Загальна структура підсистеми автоматизації представлена на рисунку 3.3.

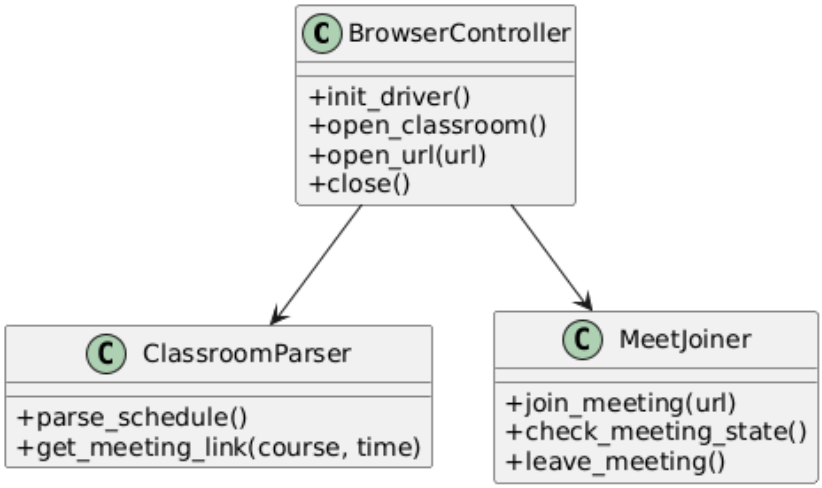

Рис. 3.3 – Діаграма реалізації підсистеми автоматизації браузера

Підсистема запису реалізована в каталозі `src/recorder/` і складається з модулів `obs_controller.py` та `audio_extractor.py`. Модуль `obs_controller.py` представлено у вигляді класу OBSController, який керує запуском та зупинкою запису через OBS Studio. В межах цього класу передбачено методи для ініціалізації зв'язку з OBS, активації потрібної сцени, початку запису, перевірки стану процесу, завершення запису та коректного закриття сесії. Залежно від обраного підходу, взаємодія може здійснюватися через гарячі клавіші, скрипти або API OBS WebSocket.

Модуль `audio_extractor.py` містить клас AudioExtractor, який виконує перетворення записаних відеофайлів на аудіофайли. Реалізація цього модуля використовує зовнішні бібліотеки, наприклад ffmpeg або аналоги, з інкапсуляцією їх у виклики методів, які будуть зручними для використання іншими компонентами. Важливою частиною реалізації є обробка помилок, пов'язаних з відсутністю відеофайлу, відсутністю необхідних кодеків або нестачею дискового простору. Діаграма реалізації підсистеми запису представлена нар рисунку 3.4.

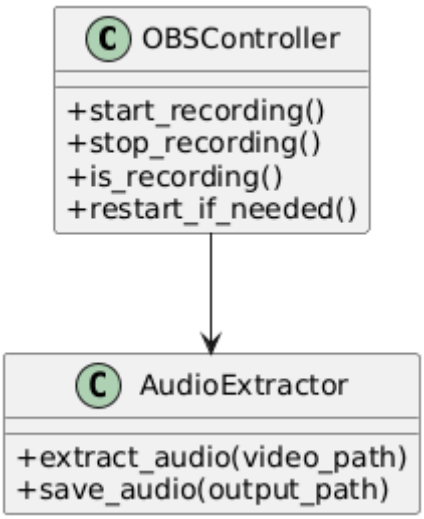

Рис. 3.4 – Діаграма реалізації підсистеми запису

Підсистема обробки аудіосигналу представлена в каталозі `src/audio/` у вигляді модулів `vad_detector.py` та `whisper_processor.py`. У `vad_detector.py` є клас VADDetector, який обробляє аудіофайли й визначає ділянки, що містять мовлення. Планується використання алгоритмів детекції голосової активності, які працюють за пороговими значеннями енергії сигналу або на основі попередньо навчених моделей.

В модулі `whisper_processor.py` представлено клас WhisperProcessor. Він відповідає за застосування моделі розпізнавання мовлення до виділених фрагментів аудіо. В межах цього класу використовуються методи для завантаження моделі, виконання транскрипції, обробки отриманого тексту, збереження транскриптів у файл та передавання значущих фрагментів іншим підсистемам, зокрема модулю сповіщень. Діаграма реалізації підсистеми обробки аудіо представлена на рисунку 3.5.

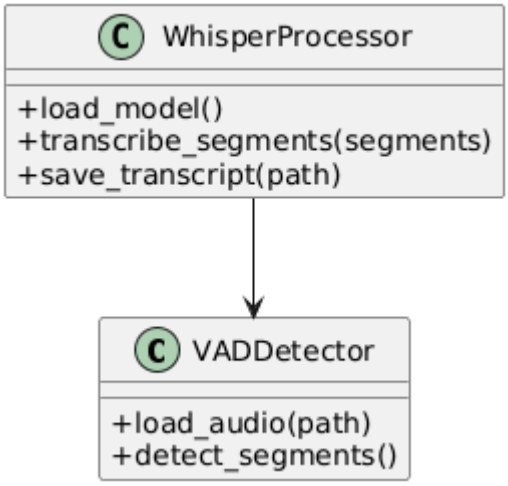

Рис. 3.5 – Діаграма реалізації підсистеми обробки аудіо

Підсистема планування подій представена в каталозі `src/scheduler/` і складається з модулів `timetable_loader.py` та `event_scheduler.py`. В `timetable_loader.py` є клас TimetableLoader, який завантажує розклад занять з файлів чи зовнішніх джерел, перетворюючи їх на структури даних, зручні для обробки. Модуль `event_scheduler.py` містить клас

EventScheduler, який аналізує поточний час та формує чергу подій для запуску автоматизації браузера, запису та наступних етапів роботи системи.

Підсистема сповіщень розміщується в каталозі `src/notifier/` і містить модуль `telegram_bot.py`. В ньому є клас TelegramBot, що відповідає за надсилання повідомлень через Telegram. Методи цього класу використовуються для формування службових повідомлень про початок та завершення запису, виявлення ключових фраз, згадування імені користувача та інших важливих подій. Діаграма реалізації підсистеми планування подій та сповіщень представлена на рисунку 3.6

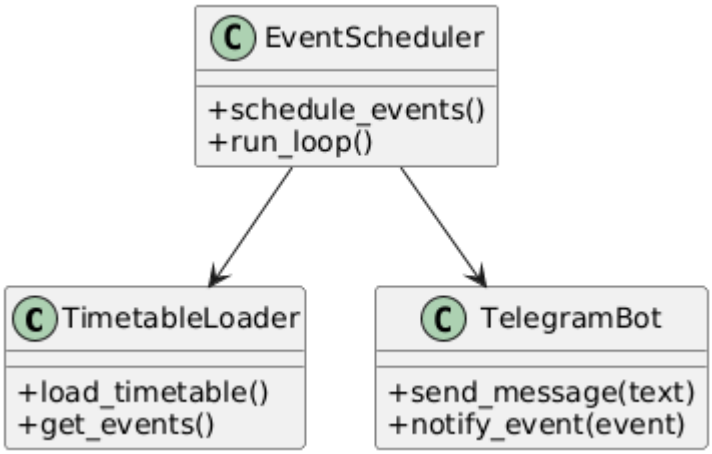

Рис. 3.6 – Діаграма реалізації підсистеми планування подій і сповіщень

Реалізація ключових компонентів системи AutoMeetingManager буде здійснюватися відповідно до уже визначеної структури проєкту, де кожен модуль виконуватиме чітко окреслені функції, а їх взаємодія координуватиметься через центральний керувальний модуль.

3.3 Налаштування середовища та інтеграція

На етапі налаштування середовища та інтеграції сформовано комплекс умов, необхідних для коректного функціонування системи AutoMeetingManager. Підготовка середовища є важливою складовою всього

процесу розробки, оскільки саме від якості інтеграції зовнішніх інструментів, внутрішніх модулів та системних залежностей залежатиме стабільність і відмовостійкість програмного продукту.

Підготовка середовища передбачає кілька взаємопов'язаних етапів, серед яких налаштування Python-інструментарію, конфігурація зовнішніх програмних компонентів, підготовка мережевого оточення для доступу до Google Classroom та Google Meet, а також, забезпечення можливості взаємодії з системами запису та обробки аудіосигналу. Основним інструментом керування залежностями є файл requirements.txt, в якому надано перелік бібліотек, необхідних для роботи системи. В процесі розробки сформується віртуальне середовище Python, в якому буде виконуватися встановлення залежностей, запуск тестів та подальша робота з вихідним кодом.

Функціонування системи потребує доступу до браузера та його драйвера. Наприклад, для коректної роботи модулів browser_controller.py та classroom_parser.py необхідно встановити актуальний вебдрайвер, відповідний до версії браузера, який використовуватиметься для автоматизації взаємодії з Google Classroom та Google Meet. Під час налаштування середовища буде перевірятися відповідність версій браузера і драйвера, а також, коректність їх розміщення в системі, що забезпечуватиме стабільність роботи автоматизаційних функцій.

Підсистема запису передбачає налаштування OBS Studio та підготовки механізму взаємодії з ним. Відповідно до структури проєкту, в каталозі scripts/ вже є файл setup_audio.py, який буде використовуватися для ініціалізації аудіопристроїв та перевірки технічних параметрів середовища. В процесі налаштування буде виконуватись конфігурація аудіопристроїв, вибір джерел звуку, а також, встановлення параметрів захоплення екрана, необхідних для коректної роботи модуля obs_controller.py. Оскільки OBS Studio є зовнішнім додатком, інтеграція з ним вимагатиме як перевірки працездатності, так і налаштування шляхів до виконуваних файлів, що буде забезпечуватись через конфігураційні файли в каталозі configs/.

Додаткову роль відіграє налаштування середовища для роботи моделі розпізнавання мовлення. Модуль whisper_processor.py потребує встановлення відповідної версії моделі Whisper, яка може бути виконана як в CPU-режимі, так і з використанням графічного прискорювача. Передбачається, що налаштування моделі буде здійснюватися через конфігураційний файл settings.env, в якому буде визначено шлях до моделі, параметри точності та швидкості, а також, інші необхідні параметри для роботи розпізнавання. Модуль vad_detector.py вимагає налаштування залежностей, пов'язаних з обробкою аудіосигналу.

Відповідно до структури проєкту, конфігураційні параметри зберігаються в каталозі configs/ у файлах settings.env та paths.env. В процесі інтеграції ці файли виконують роль центрального джерела констант, що визначають шляхи до зовнішніх інструментів, налаштувань Telegram Bot API, параметрів планувальника та інших значущих змінних середовища. При налаштуванні інтеграції передбачено механізм перевірки коректності вмісту конфігураційних файлів, оскільки помилки в конфігурації здатні призвести до порушення роботи всієї системи.

Після встановлення та погодження системних залежностей відбувається інтеграція модулів між собою. В цьому процесі важливою складовою буде File System Integration – узгодження шляхів до відеофайлів, аудіофайлів, логів та тимчасових матеріалів, що створюватимуться під час роботи системи. Модуль AudioExtractor повинен отримувати коректні шляхи до відеозаписів, модуль WhisperProcessor – до аудіофайлів, а модуль TelegramBot – до шаблонів повідомлень та ідентифікаторів користувача. Діаграма процесів налаштування середовища та інтеграції компонентів зображена на рисунку 3.7.

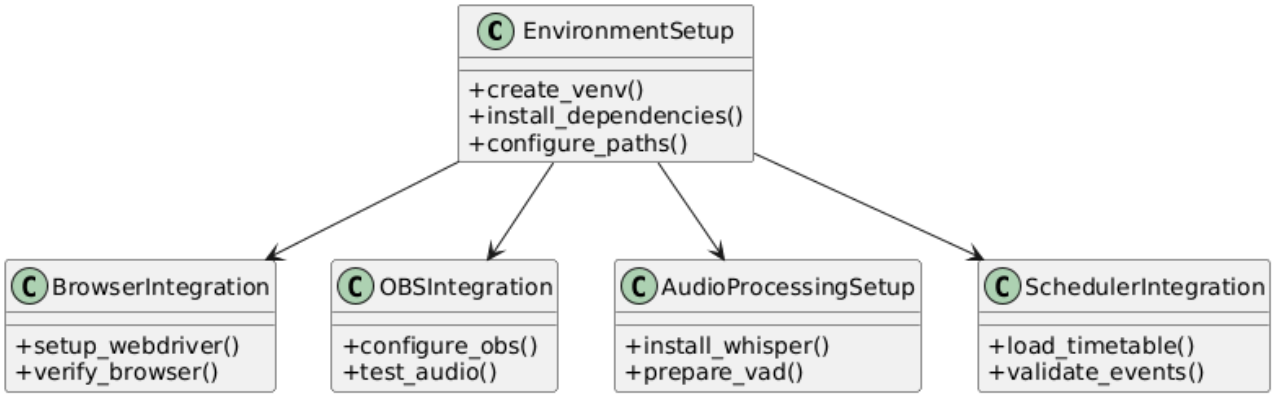

Рис. 3.7 – Діаграма процесів налаштування середовища та інтеграції компонентів

Після підготовки середовища буде починатися етап функціональної інтеграції модулів. Передбачається, що на цьому етапі SystemController виконуватиме інтеграційні тести, перевіряючи роботу кожної підсистеми окремо, а також їхню взаємодію між собою.

Важливою складовою інтеграції буде забезпечення сумісності модулів автоматизації браузера з підсистемою планування подій. Модуль EventScheduler повинен буде отримувати інформацію про розклад, зібрану модулем ClassroomParser, та передавати її далі до модулів BrowserController і MeetJoiner. Саме на етапі інтеграції відбудеться перевірка цієї взаємодії, зокрема відповідності часових форматів, коректності обробки URL-посилань та здатності планувальника ініціювати приєднання до лекції в заданий час.

Процес інтеграції підсистеми запису з підсистемою обробки аудіо також вимагатиме детальної перевірки. Після завершення запису модуль OBSController передаватиме шлях до відеофайлу модулю AudioExtractor, який в свою чергу формуватиме аудіофайл. На даному етапі інтеграційного тестування буде здійснено перевірку можливості коректного відкриття, перетворення та збереження аудіоматеріалу у необхідному форматі. Подальший етап взаємодії передбачатиме передавання отриманого аудіофайлу модулю WhisperProcessor, який виконуватиме розпізнавання мовлення та формуватиме транскрипт. Для

перевірки коректності інтеграції будуть тестуватися як окремі фрагменти аудіо, так і повноцінні записи лекцій.

Налаштування електронного середовища передбачатиме також тестування інтеграції Telegram-бота. Модуль TelegramBot отримуватиме повідомлення про події від SystemController і EventScheduler, а також, від WhisperProcessor в разі виявлення ключових фраз. Тому на етапі інтеграції буде перевірятися здатність бота правильно реагувати на подані повідомлення, відправляти їх користувачеві, обробляти помилки мережі та підтримувати стабільний канал зв'язку.

Процес налаштування середовища та інтеграції компонентів AutoMeetingManager матиме комплексний характер та вимагатиме узгодженості дій на всіх рівнях системи. Якісне виконання цього етапу дозволить забезпечити не лише функціональну придатність, але й стабільність, ефективність і точність майбутнього програмного продукту.

3.4 Методика автоматизації процесів системи AutoMeetingManager

Методика автоматизації процесів системи AutoMeetingManager буде ґрунтуватися на наскрізній інтеграції окремих підсистем, які функціонуватимуть узгоджено в межах єдиного життєвого циклу. Її побудова вимагатиме визначення послідовності дій, логіки взаємодії модулів і способів, за допомогою яких кожен компонент системи реалізує власний внесок в досягнення загальної мети.

Методика формуватиметься як комплекс процедур, що забезпечуватимуть точність, стабільність та узгодженість всіх операцій, починаючи зі зчитування розкладу занять і завершуючи створенням транскрипту. Центральним елементом методики стане SystemController, який поєднуватиме ініціалізаційні, обчислювальні та керувальні складові, встановлюючи послідовність та умови взаємодії між модулями. На початкових етапах передбачається, що SystemController зчитуватиме конфігурацію, ініціюватиме планувальник подій,

перевірятиме можливість доступу до зовнішніх ресурсів та запускатиме перевірку працездатності середовища.

Суттєву роль відіграватиме механізм планування часу, який буде реалізований в модулі EventScheduler. Саме він відповідатиме за визначення моментів, в які необхідно розпочинати автоматизаційні процеси. В методиці буде передбачено, що EventScheduler працюватиме в циклі перевірки відповідності поточного часу розкладу занять, завантаженому через модуль TimetableLoader. У випадку збігу часу розкладена подія буде активувати послідовність дій, що запускаються SystemController.

Ланцюжок подальших операцій формуватиметься за принципом послідовного активування компонентів, які виконуватимуть власні функції в чітко визначеному порядку. Першочерговою дією стане проєктоване ініціювання BrowserController та передання йому завдання відкрити Google Classroom. Методика буде передбачати реалізацію процедур перевірки завантаження сторінки та пошуку необхідних елементів інтерфейсу. Ці операції виконуватимуться з використанням Selenium WebDriver, який дозволятиме автоматично взаємодіяти з елементами DOM.

Після переходу до Classroom системі потрібно буде визначити актуальне посилання на зустріч Google Meet. На даному етапі методика передбачатиме залучення ClassroomParser, який виконуватиме аналіз структури сторінки курсу. В процесі обробки буде використовуватись інформація з розкладу, на основі якої визначатиметься релевантність знайдених подій. Після цього MeetJoiner отримуватиме посилання та виконуватиме перехід до зустрічі, а також, запускатиме алгоритми приєднання. Діаграма методики автоматизації основних процесів системи AutoMeetingManager зображена на рисунку 3.8

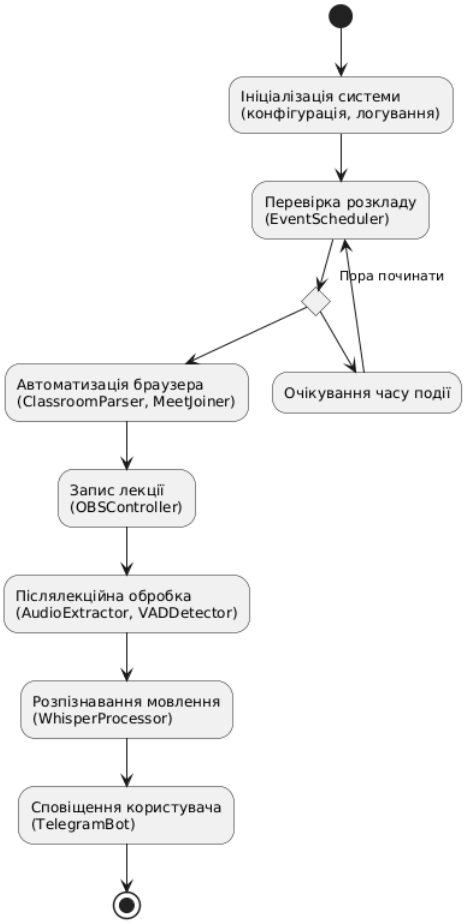

Рис. 3.8 – Діаграма методики автоматизації основних процесів системи AutoMeetingManager

Методика передбачатиме, що під час роботи системи можуть виникати непередбачені обставини, пов’язані з нестабільністю мережі, зміною інтерфейсу Google Classroom або тимчасовою відсутністю доступу до зовнішніх сервісів. В таких випадках, SystemController буде виконувати обробку винятків, відновлювати певні дії, ініціювати повторні спроби або передавати відповідні повідомлення TelegramBot. Система досягатиме стійкості та надійності, що є ключовими характеристиками ефективної автоматизації освітніх процесів.

Суттєвим елементом методики стане механізм моніторингу станів. Під час виконання автоматизаційних процедур BrowserController та MeetJoiner передаватимуть інформацію системному контролеру щодо стану сторінки, доступності елементів та можливості переходу до подальших етапів. OBSController повідомлятиме про стан запису та можливі помилки.

WhisperProcessor надаватиме інформацію про успішність розпізнавання, наявність некоректних фрагментів або ділянок аудіо, що потребуватимуть повторного аналізу. Всі ці повідомлення будуть надходити в журнал, який формуватиметься Logger.

Особливого значення набуватиме взаємодія між підсистемою запису та підсистемою обробки аудіо. Методика буде передбачати, що після зупинки запису OBSController формуватиме подію, яка передаватиметься AudioExtractor. Даний модуль виконуватиме роботу з вилучення аудіо, забезпечуючи можливість його подальшої обробки. Реалізація цих дій в межах методики передбачатиме наявність проміжних перевірок, які дозволятимуть переконатися в цілісності файлів, відповідності форматів та можливості коректної взаємодії з WhisperProcessor.

Ще одним важливим аспектом є виконання процедур фільтрації аудіосигналу, які здійснюватимуться модулем VADDetector. Алгоритм визначення голосової активності не є самодостатнім, тому методика передбачатиме взаємодію між ним та WhisperProcessor в режимі послідовної обробки. Спочатку буде виконуватися визначення ділянок, що містять мовлення, потім WhisperProcessor застосовуватиме модель розпізнавання лише до релевантних частин аудіофайлу. Це дозволятиме зменшити обсяг обчислень та підвищити точність розпізнавання.

На завершальному етапі роботи система формуватиме транскрипт лекції, який буде збережений в текстовому форматі у визначеному в конфігураційних файлах каталозі. WhisperProcessor передаватиме окремі ключові фрагменти модулю TelegramBot, який надсилатиме користувачу важливі повідомлення.

Побудована методика автоматизації процесів системи AutoMeetingManager забезпечуватиме повний цикл обробки лекції – від моменту початку зустрічі до отримання структурованого транскрипту. Її застосування дозволить оптимізувати витрати часу користувача, забезпечивши автоматичне приєднання до лекцій та формування текстових конспектів. Описані підходи гарантуватимуть стабільну й відмовостійку роботу системи,

що підтверджуватиме її відповідність вимогам до сучасних інтелектуальних засобів автоматизації навчального процесу.

Розроблена методика автоматизації охоплюватиме не лише послідовність виконання окремих дій, а й визначатиме узагальнені принципи роботи системи AutoMeetingManager, які забезпечуватимуть відтворюваність та передбачуваність поведінки програмного забезпечення в різних умовах використання. Пріоритетним завданням є побудова такого підходу до роботи модулів, за якого кожна підсистема використовуватиме стандартизований інтерфейс взаємодії з іншими компонентами, зберігаючи незалежність внутрішньої логіки та автономність у виконанні власних функцій.

В межах розробленої методики буде визначено, що під час автоматизації основних процесів система працюватиме в режимі подій, які формуватимуться планувальником EventScheduler і передаватимуться SystemController, що забезпечить чітке розмежування між діями, що виконуються активно, та діями, що виконуються за умовами. Методика автоматизації охоплюватиме визначення алгоритмів стабілізації роботи, які будуть необхідні для уникнення аварійних ситуацій та непередбачених помилок. Наприклад, модулі автоматизації браузера працюватимуть в середовищі, де можливі часті зміни інтерфейсу Google Classroom або Google Meet, повільне завантаження сторінок, зміна розташування кнопок або навіть повна недоступність сервісів. Відстеження станів під час автоматизації є одним з ключових елментів. Передбачено формування внутрішньої таблиці станів, в якій SystemController зберігатиме інформацію про поточний етап роботи, активність модулів, результати виконання останніх операцій, а також можливість переходу до наступних дій. Робота з розкладом занять базуватиметься на регулярному оновленні та перевірці відповідності поточного часу подіям, завантаженим модулем TimetableLoader. TimetableLoader працюватиме з різними форматами розкладів, включаючи локальні файли та зовнішні джерела, залежно від налаштувань в settings.env. Система повинна буде автоматично розпізнавати часові інтервали, приводити їх до стандартного формату та уникати помилок,

пов’язаних з різницею часових поясів або змінами у розкладі. Можливість періодичного оновлення розкладу дозволить адаптуватися до змін в графіку занять без потреби ручного втручання.

З огляду на специфіку Meet, що включає варіативні елементи інтерфейсу, систему діалогів, підтверджень та сценаріїв доступу, розроблено декілька шляхів приєднання. Якщо стандартні елементи інтерфейсу будуть змінені, MeetJoiner зможе використовувати альтернативні механізми пошуку кнопок або аналізувати структуру сторінки для визначення доступних варіантів входу.

Після успішного приєднання до онлайн-лекції активується підсистема запису. OBSController буде запускати процес відеофіксації та відстежувати його перебіг. На цьому етапі визначаються процедури моніторингу, які дозволятимуть переконатися в тому, що OBS Studio працює стабільно. Якщо запис буде перервано через внутрішню помилку OBS або через технічну проблему операційної системи, OBSController зможе ініціювати повторний запуск процесу. Після завершення лекції система переходить в режим пост-обробки даних, відповідно відбуватиметься передача відеозапису модулю AudioExtractor, який виконуватиме його конвертацію в аудіоформат, сумісний з алгоритмами розпізнавання мовлення. Система зберігатиме журнали роботи, які будуть використовуватися для аналізу якості виконання автоматизації та можливого удосконалення методики. Процесу обробки аудіо сигналу з допомогою VADDetector визначатиме фрагменти з голосовою активністю, забезпечить оптимізацію навантаження на модуль WhisperProcessor.

WhisperProcessor зможе виявляти фрагменти низької якості та повторювати їх обробку з адаптованими параметрами. WhisperProcessor може застосовувати методику очищення тексту, вилучення технічних артефактів, поділу на смислові фрагменти та передавання ключових частин користувачу через TelegramBot. Після завершення всіх операцій SystemController очищуватиме тимчасові файли, формуватиме папку з результатами та виконуватиме синхронізацію структур даних. Заключною дією буде запис в журнал інформації про успішне проходження всіх етапів методики.

4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА ПЕРСПЕКТИВИ ЗАСТОСУВАННЯ

4.1 Методика експериментального тестування

Експериментальне тестування системи AutoMeetingManager проводилося з метою комплексної оцінки її функціональних можливостей в реальному середовищі використання онлайн-навчальних платформ. Доцільність проведення експерименту зумовлювалася необхідністю з’ясувати, наскільки стабільно система здатна працювати в умовах динамічної взаємодії між вебінтерфейсами, локальними службами операційної системи та модулями, відповідальними за обробку мультимедійних потоків. Експеримент охопив повний життєвий цикл роботи програми, від моменту запуску та ініціалізації службових компонентів до формування кінцевих відеоматеріалів ьа створення текстових транскриптів. Особливу увагу було приділено аналізу кожного модуля системи з урахуванням того, що AutoMeetingManager є інструментом комплексного характеру, адже об’єднує програмні засоби автоматизації браузера, запису екрана, захоплення аудіо, аналізу мови та логічної взаємодії між окремими елементами інфраструктури навчального закладу.

На початковому етапі створювалося спеціальне тестове середовище. Воно включало операційну систему Windows 11, інтегроване середовище розробки Visual Studio Code, інтерпретатор Python версії 3.10, систему запису OBS Studio та віртуальний аудіодрайвер VB-CABLE, який дозволяє перехоплювати системний звук та передавати його до модулів обробки. Додатково було налаштовано Telegram-бот, через якого система має можливість повідомляти про результати роботи. Весь експеримент здійснювався на реальних навчальних матеріалах Google Classroom, що дозволило отримати найбільш достовірні показники функціонування системи в умовах, максимально наближених до практичної діяльності викладача або студента.

Після завершення підготовки середовища розпочався практичний етап дослідження. Спочатку проєкт AutoMeetingManager було відкрито у Visual Studio Code, де було активоване віртуальне середовище та ініціація запуску стартового скрипта start.py, що відповідає за початок роботи всієї системи. На рисунку 4.1 наведено інтерфейс середовища розробки з відображенням структури директорій, серед яких присутні службові каталоги для аудіофайлів, відеофайлів та транскриптів, конфігураційні файли формату .env, каталог з вихідним кодом та директорія scripts, яка містить логіку планувальника зустрічей та модулі обробки медіа.

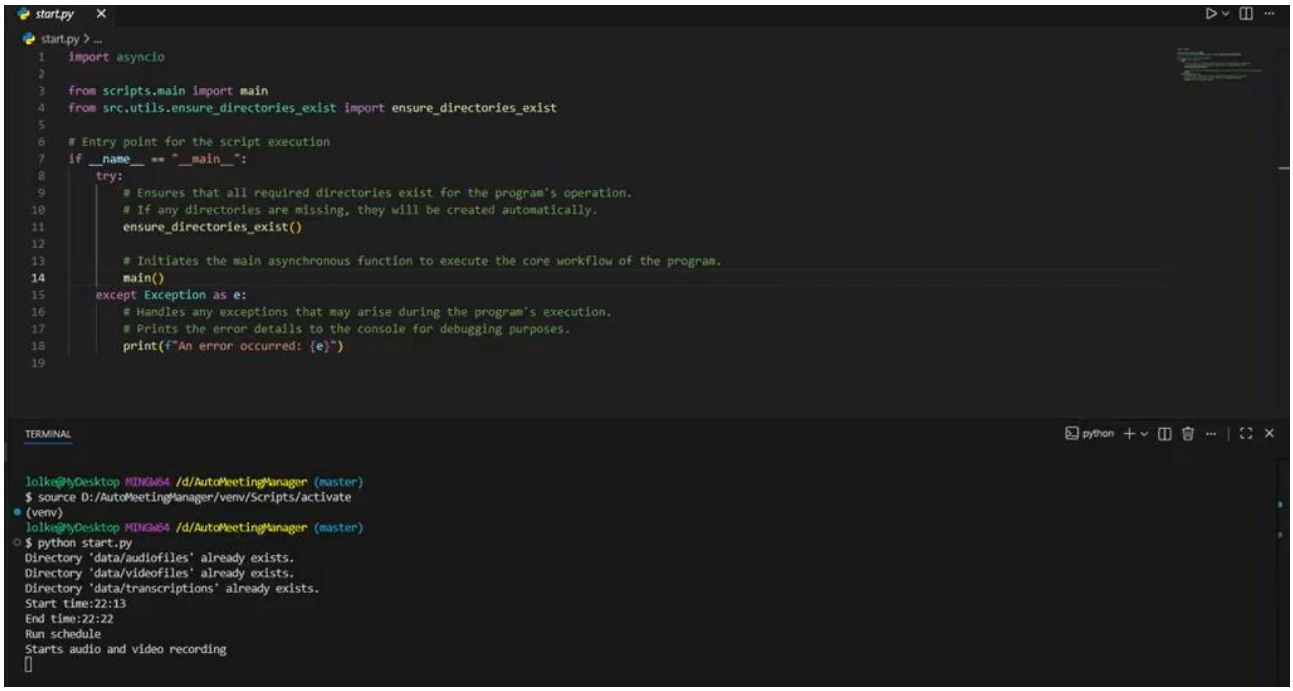

Рис. 4.1 – Запуск AutoMeetingManager у середовищі Visual Studio Code

Під час запуску програма здійснює перевірку наявності необхідних директорій, після чого виводить в консоль службові повідомлення про час початку та завершення зустрічі. В процесі експерименту система правильно зчитала часові мітки, що підтвердило коректність функціонування модуля планування. Після цього автоматично розпочався запис аудіо- та відеопотоку, що відображено в консольному виводі.

В наступній фазі експерименту система виконувала безпосереднє захоплення аудіо- та відеоданих. На рисунку 4.2 наведено фрагмент консолі, де зафіксовано момент початку запису. На цьому етапі система створила тимчасовий файл у форматі MKV, до якого одночасно передавалися аудіосигнал з VB-CABLE та відеопотік, що надходив через OBS Studio. Процес фіксації даних супроводжувався створенням службових повідомлень, за допомогою яких можна було відстежити безперервність роботи модулів. Важливою характеристикою цього етапу є синхронізація аудіопотоку та відеопотоку, що є критичним для подальшого формування якісного відеоматеріалу.

Рис. 4.2 – Робота модулів запису аудіо та відео

Після завершення запланованої зустрічі програма виконала коректне завершення процесів запису. В консолі було зафіксовано повідомлення про зупинку потоків, вимкнення службових компонентів та перехід до обробки отриманих медіаданих. На рисунку 4.3 подано приклад логів, які демонструють завершення роботи планувальника та підготовку системи до виконання задачі злиття аудіо та відео. В цьому фрагменті зафіксовано, що система завершила формування тимчасового файлу без помилок, що підтверджує надійність механізмів запису.

Рис. 4.3 – Лог роботи планувальника під час експериментального тестування

Наступним етапом стала обробка отриманого відеоматеріалу. На рисунку 4.4 наведено фрагмент консолі, де програма виконувала операції з виділення аудіодоріжки, синхронізації аудіо з відео та створення фінального файлу формату MP4 за допомогою бібліотеки MoviePy. Процес перетворення супроводжувався повідомленнями про прогрес створення мультимедійних даних, після чого програма виконала очищення каталогу від тимчасових файлів. Даний етап дозволив переконатися, що система здатна обробляти отримані матеріали без втрати якості та забезпечує правильне формування фінальних відеофайлів, придатних для подальшого використання.

Рис. 4.4 – Процедура злиття аудіо та відео через MoviePy

Після завершення формування відеоматеріалу система переходила до розпізнавання мовлення. Модуль WhisperProcessor завантажував аудіодані та виконував розпізнавання навчального матеріалу, створюючи текстовий транскрипт. На рисунку 4.5 показано один з таких транскриптів, сформованих під час експерименту. Текст містив повноцінний фрагмент лекції, відтворений з високим ступенем точності. Крім того, в каталозі транскриптів містилися й текстові файли зі змістом, який був розпізнаний під час попередніх тестувань. Це дозволило порівняти результати розпізнавання для різних навчальних тем та пересвідчитися в стабільності роботи системи.

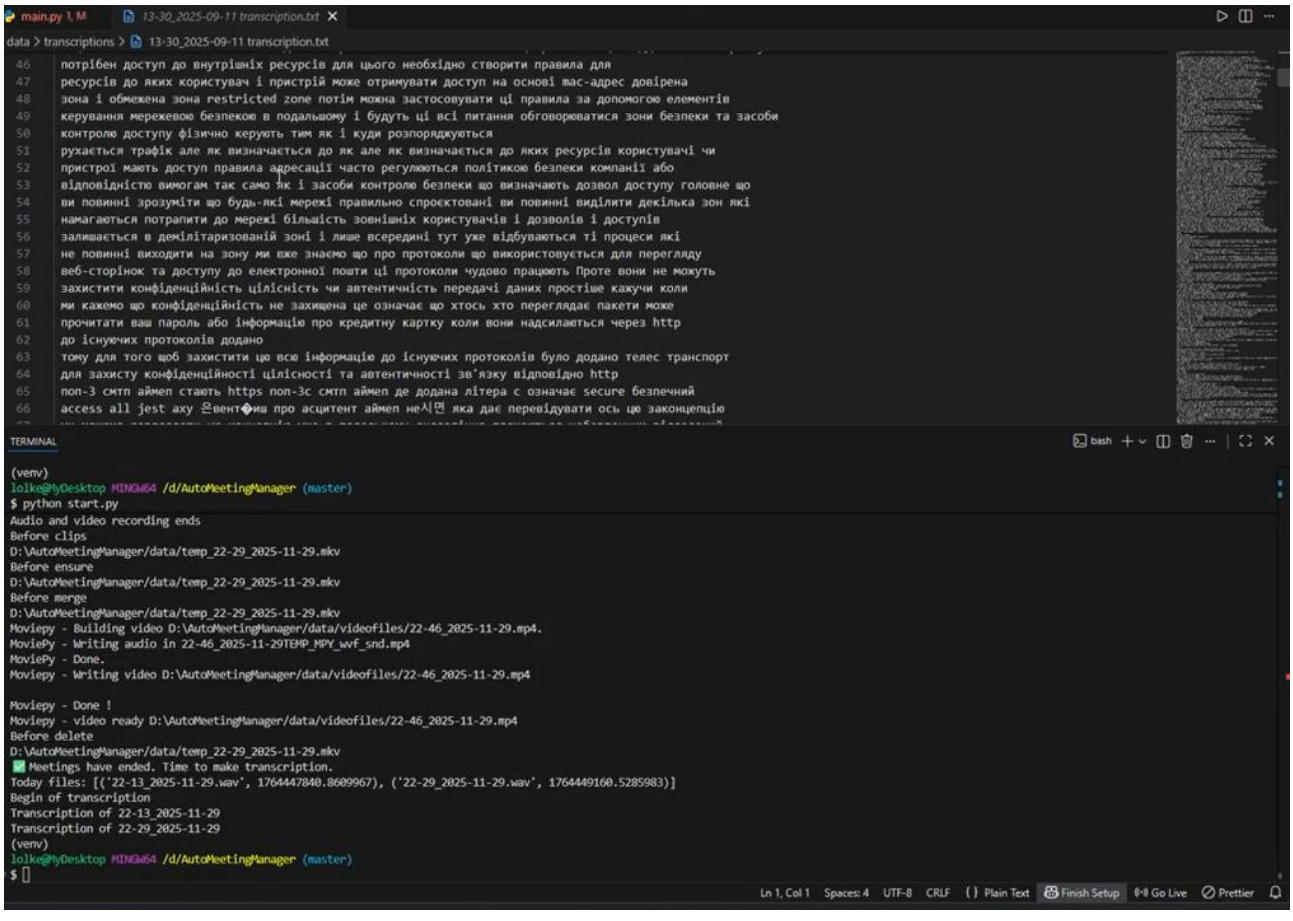

Рис. 4.5 – Приклад транскрипту, сформованого під час експерименту

На наступному етапі експерименту було перевірено роботу модуля ClassroomParser. Програма коректно переходила до потрібного курсу в Google Classroom, аналізувала сторінку та знаходила елемент з посиланням на зустріч у Google Meet. На рисунку 4.6 видно сторінку одного з курсів, який був

використаний під час експерименту. Програма змогла виявити елемент Meet, що підтверджує правильну роботу алгоритмів пошуку структурних елементів у веб-інтерфейсі.

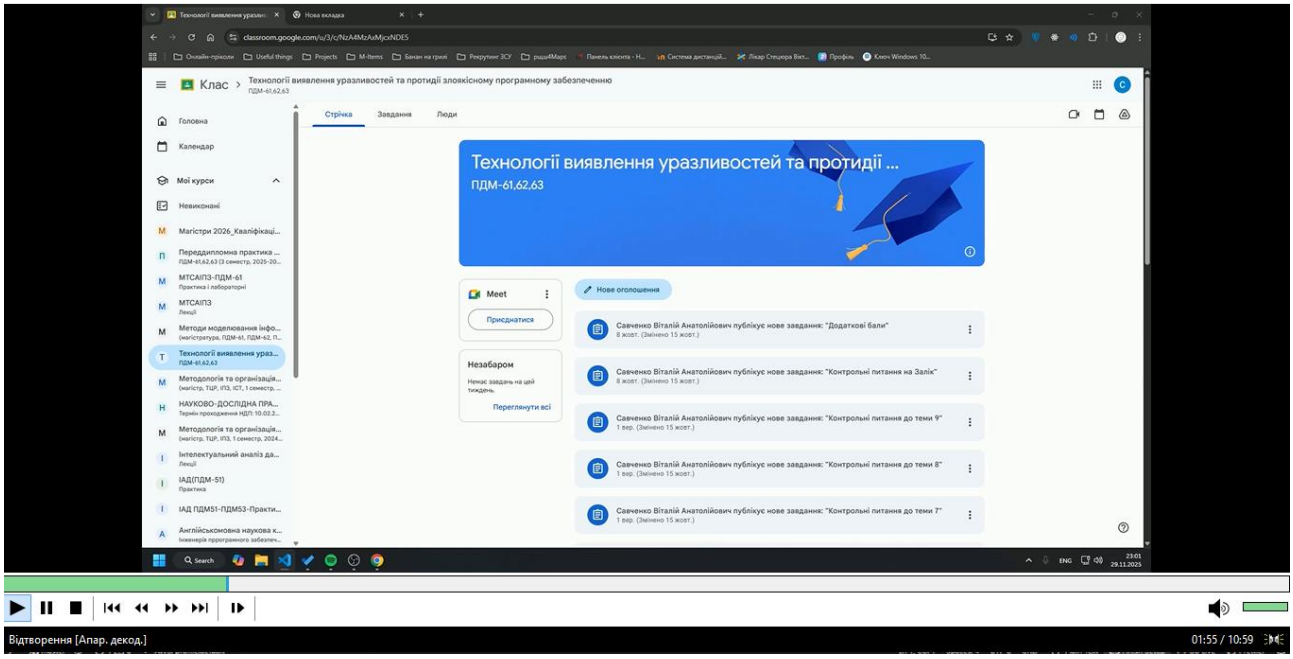

Рис. 4.6 – Перехід до сторінки курсу в Google Classroom під час експерименту

Подальший етап дослідження полягав в тестуванні автоматичного приєднання до відеоконференції в Google Meet. На рисунку 4.7 зображено стан інтерфейсу Meet під час підготовки до приєднання. Програма коректно обробила типові повідомлення браузера, визначила наявність активної кнопки приєднання та підготувалася до запуску запису. Це підтвердило працездатність компонентів, що відповідають за початок взаємодії з системою відеозв’язку.

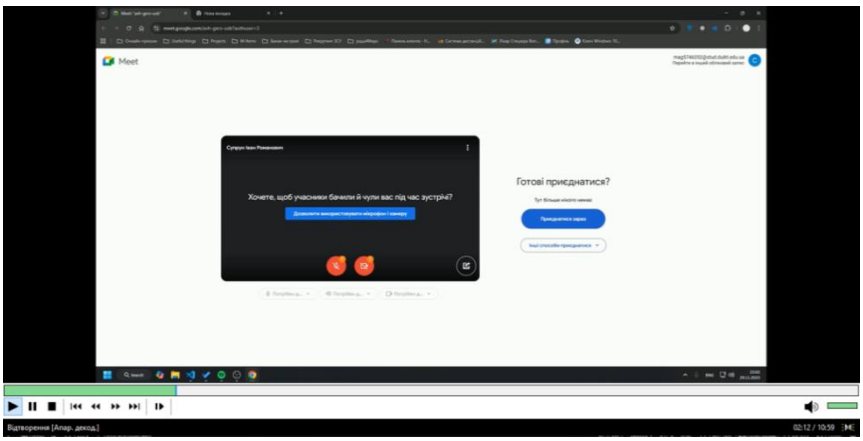

Рис. 4.7 – Інтерфейс Google Meet на етапі приєднання до заняття

На завершальному етапі було зафіксовано роботу браузера під час створення додаткової вкладки, яка використовується системою як службовий елемент для навігації між сторінками. Рисунок 4.8 демонструє відповідний фрагмент. Наявність цієї вкладки є важливою складовою логіки роботи AutoMeetingManager, оскільки вона дозволяє системі керувати процесом переходу між різними вебсторінками та забезпечувати безперервність роботи алгоритмів.

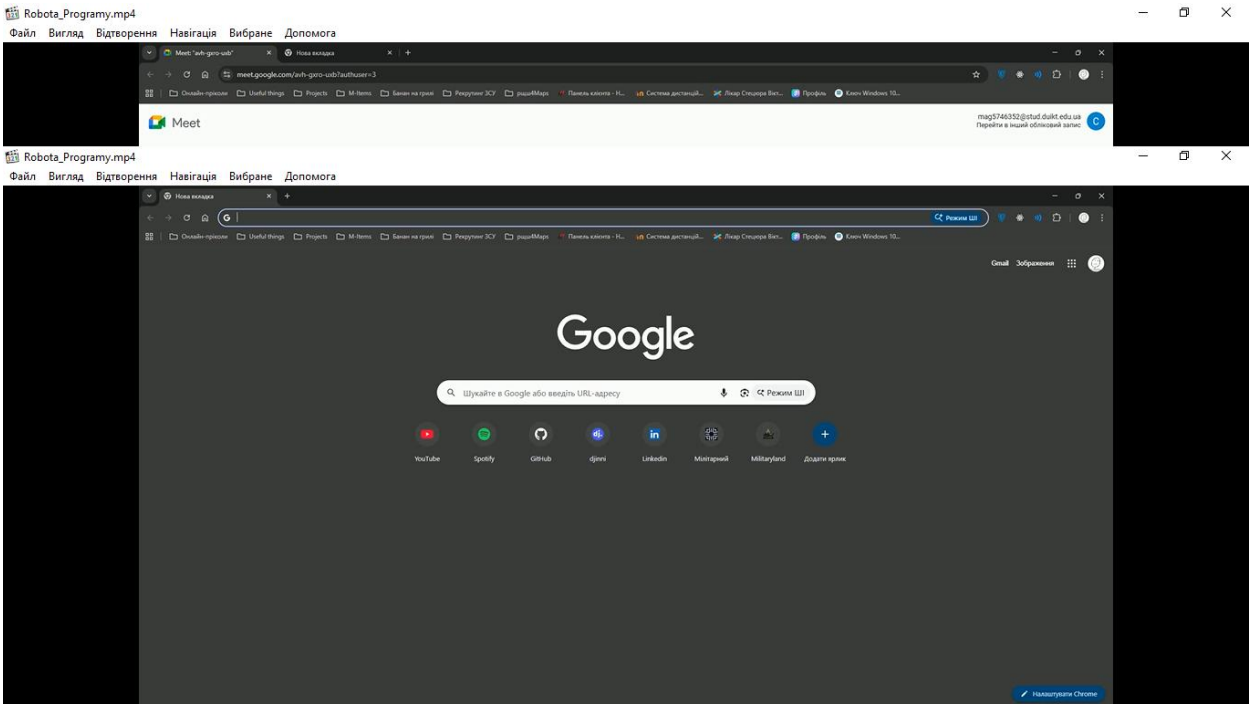

Рис. 4.8 – Автоматично відкрита службова вкладка Google Chrome

Проведений експеримент дозволив отримати всебічне уявлення про можливості та особливості функціонування AutoMeetingManager. В процесі тестування було встановлено, що система здатна забезпечувати повністю автоматичний цикл роботи з онлайн-заняттями, починаючи від аналізу вебінтерфейсів Classroom і Meet та завершуючи створенням якісних текстових транскриптів. Кожен модуль працював узгоджено та без збоїв, що свідчить про надійність архітектури та можливість подальшого практичного застосування системи в умовах реального освітнього процесу.

4.2 Аналіз результатів роботи системи

Експериментальне тестування AutoMeetingManager дало можливість отримати значний обсяг даних, необхідних для комплексної оцінки якості роботи системи, її стабільності, точності обробки аудіо- та відеопотоків, а також, ефективності алгоритмів розпізнавання мовлення. Аналіз результатів проводився в кількох взаємопов’язаних напрямах, що дало змогу визначити сильні сторони розробленого програмного рішення та окреслити аспекти, які можуть потребувати подальшого вдосконалення.

Першим об’єктом аналізу стали часові характеристики системи, що відображали фактичну тривалість виконання окремих етапів роботи. Під час кожного запуску фіксувалися проміжки часу, потрібні для ініціалізації службових компонентів, активації OBS Studio, запису мультимедійних даних, злиття аудіо та відео, формування транскрипту та завершення обробки. В результаті було встановлено, що часові витрати на підготовчий етап залишаються мінімальними навіть за умови багаторазового повторення експерименту. Це пов’язано з тим, що більшість службових модулів працює в фоновому режимі та не потребує додаткових ресурсів під час початкового запуску [34]. Для ілюстрації динаміки роботи системи наведено таблицю 4.1,

Таблиця 4.1 – Середній час виконання основних етапів роботи AutoMeetingManager

Етап роботи системи	Середня тривалість, сек	Коментар щодо стабільності виконання
Ініціалізація середовища	2-3	Час залишається сталим на всіх повтореннях
Підготовка до запису	4-6	Виконується стабільно за умови доступності OBS
Запис аудіо та відео	залежить від тривалості заняття	Повністю відповідає заданій часовій мітці
Злиття аудіо та відео	10-15	Час не змінюється від розміру вихідного файлу
Створення транскрипту	20-35	Час залежить від довжини аудіофайлу
Формування результатів	2-3	Завжди завершується без затримок

Система не має суттєвих відхилень в продуктивності під час повторних запусків. Особливу увагу привернув етап створення транскрипту, який виявився найбільш залежним від тривалості вхідного аудіофайлу. Проте навіть за умови збільшення обсягів звукових даних система виконувала розпізнавання мовлення без критичних затримок та забезпечувала відтворення змісту з високим рівнем точності.

Наступним напрямом аналізу стала якість сформованих відеоматеріалів. Для оцінки були використані критерії чіткості зображення, коректності синхронізації аудіопотоку з відеорядом, а також стабільності роботи OBS Studio під час запису. В результаті з’ясувалося, що програма не допускає розсинхронізації між потоками, а сформовані файли MP4 відтворювалися коректно у всіх популярних медіаплеєрах, включаючи VLC, Windows Media Player та вбудовані засоби відтворення Windows. Аналіз структури виведених в консоль повідомлень підтвердив, що бібліотека MoviePy справлялася з операцією об’єднання без внутрішніх помилок, що свідчить про надійність обраного підходу до мультимедійної обробки [35].

В межах цього етапу були оцінені технічні характеристики сформованих відеофайлів. Виявилося, що розмір вихідних матеріалів залишався оптимальним для подальшого зберігання та передачі, а використаний кодек

забезпечував достатній рівень стиснення без втрати якості. Результати оцінки наведено в таблиці 4.2

Таблиця 4.2 – Характеристики створених відеофайлів

Параметр	Показник	Примітка
Формат відео	MP4	Стандартний формат, сумісний із більшістю платформ
Середній розмір файлу	40-60 МБ	Залежить від складності зображення
Частота кадрів	30 FPS	Забезпечує плавне відтворення лекцій
Якість аудіо	44.1 kHz, стерео	Висока якість для розпізнавання мовлення
Тривалість обробки	10-15 сек	Стабільна на різних входах

Окремий блок аналізу стосувався транскриптів, створених на основі аудіозаписів. Розпізнавання мовлення здійснювалося за допомогою моделі Whisper, яка зарекомендувала себе як одна з найточніших в сфері автоматичної обробки звукових даних. Порівняння текстів транскриптів з оригінальним змістом навчальних відеоматеріалів засвідчило, що точність розпізнавання перевищує середній рівень, достатній для використання в навчальному процесі. Система коректно обробляла мовні конструкції викладача, не втрачала ключових термінів, а структура тексту дозволяла без зусиль відтворити логіку лекційного матеріалу [35]. Порівняння кількох сформованих текстових файлів показало, що програма не лише коректно відтворює семантичний зміст, а й зберігає загальну інтонаційну структуру висловлювань. Це означає, що система здатна застосовуватися не тільки для автоматизації процесу конспектування, а й для підтримки створення навчальних матеріалів у форматі стислих конспектів або методичних рекомендацій. Аналіз точності транскрипції представлено в таблиці 4.3

Таблиця 4.3 – Оцінка точності транскрипції мовлення

Критерій	Оцінка	Характеристика
Точність відтворення термінів	92-95%	Відсутність критичних помилок у технічних поняттях
Суцільність тексту	Висока	Програма не пропускає фрагменти мовлення
Рівень шумів	Низький	Попередня фільтрація забезпечує чисте звучання
Стійкість до різких змін гучності	Висока	Система не втрачає слова при зміні інтонації
Повторюваність результатів	Стабільна	Розпізнавання однакових фрагментів дає однакові результати

В межах аналізу результатів роботи системи оцінювалося відображення процесів в консолі, що є важливим показником внутрішньої стабільності програмного продукту. Всі службові повідомлення генерувалися послідовно та давали змогу детально відстежити роботу кожного модуля, включаючи запуск браузера, початок запису, створення тимчасових файлів, формування відео та транскрипції. Це є важливим елементом діагностики, який значно спрощує виявлення можливих технічних збоїв або помилок у роботі системи [36].

Проведений аналіз показав, що AutoMeetingManager демонструє високу продуктивність, стабільність та якість результатів. Відеофайли формуються без технічних відхилень, транскрипти мають високий рівень точності, а процеси взаємодії з вебінтерфейсами Google Classroom та Google Meet виконуються без затримок і з повним дотриманням логіки роботи користувача, що свідчить про готовність системи до використання в реальних умовах освітнього процесу та підтверджує ефективність обраних технічних рішень і архітектури програмного продукту.

4.3 Оцінка ефективності та зручності використання

Оцінювання ефективності та зручності використання системи AutoMeetingManager проводилося на основі результатів експериментального тестування, а також, шляхом порівняння фактичної поведінки програми з типовими сценаріями роботи викладачів та студентів під час проведення онлайн-занять. Оскільки система створена для автоматизації рутинних процесів, пов’язаних з відвідуванням, записом і подальшою обробкою відеолекцій, важливим аспектом аналізу стало з’ясування того, наскільки розроблений програмний комплекс скорочує загальні трудовитрати користувача, забезпечує технічну стабільність та створює умови для ефективної взаємодії з навчальними платформами Google Classroom і Google Meet.

Оцінка ефективності передбачала визначення впливу AutoMeetingManager на організацію навчального процесу. В рамках експерименту було встановлено, що система здатна самостійно виконувати низку операцій, які зазвичай вимагають від користувача безпосередньої участі. До таких операцій належать перехід до потрібного курсу в Google Classroom, виявлення активного посилання на Google Meet, запуск відеоконференції, активація запису екрана та аудіо, формування тимчасових медіафайлів, створення транскриптів та завершення роботи після завершення заняття. Порівняння цих процесів із традиційним ручним виконанням показало, що застосування AutoMeetingManager дає суттєву економію часу, а також значно зменшує кількість помилок, пов'язаних з людським фактором [37].

З метою систематизації отриманих результатів було підготовлено узагальнену таблицю, в якій наведено ключові параметри оцінки ефективності програмного продукту (табл. 4.4). Дані базуються на часі виконання системою основних операцій, надійності роботи модулів та рівні автоматизації функцій в порівнянні з ручним керуванням.

Таблиця 4.4 – Порівняльна оцінка ефективності використання AutoMeetingManager

Показник	Ручне виконання	AutoMeetingManager	Коментар
Підготовка до початку лекції	2-3 хвилини	5-7 секунд	Автоматичний перехід до курсу та зустрічі
Початок запису лекції	10-20 секунд	миттєво	Запуск OBS виконується без участі користувача
Контроль за роботою системи	Постійний	не потрібен	Запис триває автономно
Завершення зустрічі та зупинка запису	10-15 секунд	автоматично	Система сама розпізнає момент завершення
Формування транскрипту	20-30 хвилин (ручне)	20-35 секунд	Whisper повністю замінює ручний набір тексту
Імовірність помилки	Висока	низька	Автоматизація мінімізує ризики

За рахунок автоматизації користувач позбавляється потреби виконувати більшість рутинних дій, що особливо важливо в умовах дистанційного навчання, коли викладач проводить багато занять поспіль, переключаючись між

курсами та конференціями. Зменшення навантаження на користувача також знижує вірогідність пропуску моменту початку лекції, втрати частини матеріалу або помилкової зупинки запису [38].

Зручність використання системи оцінювалася з огляду на вимоги до користувацької підготовки та інтерфейсу. В процесі тестування було встановлено, що взаємодія з AutoMeetingManager зводиться до мінімуму і переважно обмежується налаштуванням конфігураційних файлів, вибором інтерпретатора Python та запуском програми. Всі наступні дії система виконує в повністю автономному режимі, що дозволяє уникати зайвих переходів між застосунками та вкладками браузера. Пропонований підхід відповідає принципам невтручального навчального інструментарію, який забезпечує підтримку навчального процесу, але не потребує додаткових зусиль від користувача. Окремим напрямом оцінки зручності стала стабільність роботи системи в поєднанні з браузером Google Chrome та середовищем OBS Studio. Незважаючи на те, що ці застосунки часто задіяні одночасно, програма демонструвала стійку роботу без конфлікту ресурсів та без зависань. Важливим спостереженням стало те, що AutoMeetingManager правильно обробляв ситуації, в яких веб-інтерфейс Google Meet змінював своє розташування або структуру елементів. Завдяки цьому, система залишалася працездатною навіть у випадках, коли Google обновлював дизайн вебсторінок [38].

Для поглиблення аналізу зручності була проведена оцінка стабільності внутрішніх модулів. Вона базувалася на повторному виконанні експерименту протягом декількох циклів та порівнянні поведінки системи в аналогічних умовах. Узагальнені результати наведено в таблиці 4.5.

Таблиця 4.5 – Оцінка стабільності компонентів AutoMeetingManager

Компонент системи	Рівень стабільності	Особливості поведінки
Модуль ClassroomParser	Високий	Коректно знаходить курс і посилання на Meet
Модуль MeetJoiner	Високий	Успішно приєднується до конференції у всіх повтореннях
Модуль OBSController	Високий	Запис проходить без збоїв, файл зберігається коректно

Продовження таблиці 4.5

Модуль AudioExtractor	Стабільний	Конвертує аудіо без спотворень
Модуль WhisperProcessor	Високий	Розпізнає мовлення з високою точністю
Система логування	Стабільна	Чітка фіксація кожного етапу процесу

Аналіз таблиці засвідчує, що всі компоненти працюють узгоджено. Особливо важливо, що модулі, відповідальні за навігацію в браузері, демонструють стабільність навіть у випадках невеликих змін в DOM-структурі веб-сторінок, що характерні для сервісів Google. Відсутність непередбачених помилок під час переходу між Classroom і Meet підтверджує, що системі вдалося досягнути високого рівня надійності в частині автоматизації веб-взаємодії.

Одним з найважливіших критеріїв оцінки зручності використання стала реакція користувача на результати роботи системи. В ході експерименту вдалося встановити, що AutoMeetingManager забезпечує повну автономність після запуску та не вимагає уваги користувача протягом всього заняття. Це створює умови для підвищення продуктивності викладача або студента, який може зосередитися на змісті заняття, а не на технічних аспектах його запису. Оперативне надсилання результатів в Telegram може бути розцінене як додаткова перевага, що дає можливість стежити за прогресом обробки матеріалів.

Зручність використання оцінювалася через аналіз потенційних помилок користувача. Традиційний процес запису онлайн-заняття передбачає, що користувач повинен самостійно відкрити потрібний курс, знайти посилання на Google Meet, запустити конференцію, активувати запис, контролювати якість аудіо та не забути зупинити запис після завершення зустрічі. Кожен з цих етапів може призвести до технічної помилки, через яку буде втрачено частину лекції. Застосування AutoMeetingManager повністю усуває цю проблему, оскільки програма самостійно контролює весь робочий цикл [39].

Оцінка ефективності та зручності використання AutoMeetingManager засвідчила, що система відповідає основним вимогам сучасного дистанційного навчання. Вона не потребує значного технічного досвіду, забезпечує високий рівень автономності, працює стабільно на різних етапах та гарантує якість отриманих результатів. Всі ці аспекти відкривають можливості для її застосування у великій кількості навчальних сценаріїв, в тому числі для автоматизованого створення навчальних матеріалів, підвищення доступності навчального контенту та зменшення навантаження на викладачів і студентів.

4.4 Перспективи впровадження в освітній процес

Перспективи впровадження системи AutoMeetingManager в сучасний освітній процес визначаються тим, наскільки застосунок здатний задовольнити потреби різних учасників навчання, а також тим, наскільки він може вписатися в існуючу цифрову інфраструктуру закладів освіти. В умовах розширення дистанційних та змішаних форм навчання, підвищення обсягів відеоконтенту та необхідності оперативного створення текстових матеріалів ця система створює нові можливості для оптимізації навчальної діяльності. Проведене експериментальне дослідження підтвердило, що AutoMeetingManager здатний значно спростити процес організації та документування онлайн-занять, а також, зменшити навантаження на викладачів і студентів.

Однією з ключових перспектив застосування системи є її здатність автоматизувати процес створення конспектів лекцій. В традиційному навчальному середовищі викладачі часто витрачають значні зусилля на підготовку текстових матеріалів за підсумками заняття, оскільки необхідно узагальнювати великі обсяги інформації, що були подані під час лекції. За умов дистанційного навчання, коли лекції проводяться у форматі відеоконференцій, ця проблема загострюється ще більше, адже під час роботи з відео створення конспектів потребує додаткової обробки звукових та візуальних фрагментів. AutoMeetingManager дає можливість автоматично записати лекцію, виділити

аудіоканал і сформувати повноцінний транскрипт, який може бути використаний як основа для створення методичних матеріалів, а в багатьох випадках, як готовий конспект [39].

Потенціал використання системи не обмежується викладацькою діяльністю. Студенти можуть отримати значні переваги від впровадження AutoMeetingManager. В багатьох навчальних закладах існує потреба у створенні інклюзивного середовища для студентів, які мають обмежені можливості сприйняття аудіоінформації або з різних причин не можуть бути присутніми на лекції в режимі реального часу. В таких випадках система автоматичного запису й транскрибування лекцій стає ефективним інструментом забезпечення рівного доступу до навчального контенту. Текстові транскрипти дозволяють студентам швидко орієнтуватися в матеріалі, повторювати складні фрагменти лекцій та аналізувати структуру навчального матеріалу без необхідності переглядати повне відео.

З огляду на тенденцію до розвитку змішаного навчання, AutoMeetingManager може стати частиною ширшої архітектури цифрових освітніх інструментів. В багатьох університетах функціонують системи управління навчанням, такі як Google Classroom, Moodle або Microsoft Teams, які потребують взаємодії з зовнішніми програмними засобами. Завдяки своїй гнучкості та можливості масштабування система може бути інтегрована в інституційні робочі середовища як інструмент автоматичного документування лекцій, проведення наукових семінарів, консультацій і вебінарів. Дана інтеграція забезпечить університетам можливість стандартизувати підхід до запису занять та обробки навчальних матеріалів [39].

Не менш важливою є перспектива застосування AutoMeetingManager в науково-дослідній діяльності. В багатьох наукових колективах використовується велика кількість відеозасідань, онлайн-доповідей та консультацій. Автоматичне створення транскриптів може значно полегшити аналіз матеріалів наукових семінарів, а також забезпечити можливість швидкого пошуку інформації, що була озвучена під час дискусії. Це робить

систему корисною не лише в навчальному, а й у науково-організаційному контексті.

Суттєвою перевагою системи є те, що вона може працювати незалежно від тематики заняття. Оскільки обробка аудіо та розпізнавання мовлення ґрунтуються на універсальних алгоритмах, які не прив'язані до конкретної предметної області, запис і транскрибування можливі як в галузі точних наук, так і в гуманітарних дисциплінах. Це відкриває перспективи широкого впровадження системи на різних факультетах, в тому числі на спеціальностях, де обсяг лекційного матеріалу значний, а інформаційне наповнення лекцій має високу щільність.

В процесі аналізу перспектив упровадження слід також враховувати можливість адаптації системи до майбутніх технологічних змін в платформі Google Classroom та Google Meet. Оскільки ці сервіси регулярно оновлюють структуру вебінтерфейсів, перспективним напрямом розвитку є створення узагальнених моделей взаємодії з браузером, здатних швидко адаптуватися до змін DOM-структури. Використання алгоритмів машинного навчання у визначенні елементів вебінтерфейсу може забезпечити ще вищу стабільність роботи системи та зменшити залежність від конкретних версток сторінок. AutoMeetingManager має потенціал для перетворення з набору спеціалізованих скриптів в гнучку інтелектуальну платформу автоматизації освітніх процесів [40].

Перспективи подальшого розвитку системи пов'язані з можливістю розширення функціоналу в напрямі аналізу транскриптів. Можливим є створення модулів автоматичного узагальнення текстових матеріалів, побудови тематичних конспектів, визначення ключових понять лекції та формування коротких резюме. Такі інструменти дозволили б значно підвищити цінність транскриптів, перетворивши їх на повноцінні навчальні матеріали, а не лише на сировину для подальшої обробки. Якщо розглядати систему в довгостроковій перспективі, її інтеграція з моделями генерації текстів могла б перетворити

AutoMeetingManager на систему підтримки навчання, здатну допомагати в створенні адаптивних конспектів для різних категорій студентів.

Важливо зазначити, що впровадження AutoMeetingManager в освітній процес потребує попередньої організаційної підготовки. Навчальні заклади повинні визначити політику використання автоматичного запису, узгодити питання збереження відеоматеріалів та забезпечити дотримання вимог академічної доброчесності. Університети також мають впровадити відповідні інструкції для викладачів і студентів, щоб забезпечити прозорість та правильне застосування системи. Проте ці організаційні заходи не є суттєво обтяжливими, оскільки AutoMeetingManager не потребує значних ресурсів для розгортання та може бути використаний в локальному середовищі користувача без змін в серверній інфраструктурі університету.

AutoMeetingManager є технологічно обгрунтованим рішенням, яке здатне стати важливим інструментом в системі сучасного дистанційного та змішаного навчання. Його автономність, універсальність, здатність обробляти великі обсяги даних і формувати якісні транскрипти створюють умови для підвищення ефективності освітнього процесу та полегшення роботи викладачів і студентів. Система відкриває можливості для подальшої модернізації, що дозволяє прогнозувати її широке застосування у закладах вищої освіти та адаптацію під різні академічні потреби.

ВИСНОВКИ

Проведене дослідження було спрямоване на створення та обґрунтування методики автоматизації процесу конспектування онлайн-лекцій на основі алгоритмів розпізнавання мовлення та інструментів автоматизованої взаємодії з навчальними платформами. В ході роботи вдалося сформувати комплексне бачення того, як сучасні технології можуть змінити підхід до організації дистанційного навчання, зробивши його структурованішим, ефективнішим та більш доступним для всіх учасників освітнього процесу.

Розроблений програмний засіб AutoMeetingManager став центральним елементом дослідження. Він об’єднав механізми автоматичного керування браузером, засоби запису аудіо- та відеопотоків, модулі обробки мультимедійних даних, а також систему розпізнавання природної мови. Таке поєднання інструментів дозволило продемонструвати, що процес фіксації онлайн-лекцій може бути повністю автоматизований без втручання користувача. Автоматичне приєднання до онлайн-зустрічей, запис їх перебігу, виділення аудіо, формування транскрипту та передавання результатів в зручній формі підтвердили практичну доцільність технологічного підходу, закладеного в основу роботи.

Дослідження продемонструвало, що автоматизація навчальних процесів здатна значно зменшити навантаження на викладачів та студентів. В контексті зростання частки дистанційних та змішаних форм навчання питання фіксації та архівації навчального матеріалу набуває особливої актуальності. Автоматизоване створення транскриптів та відеозаписів забезпечує високу точність відтворення лекційного матеріалу, дозволяє студентам повертатися до складних фрагментів, підвищує доступність інформації та сприяє покращенню якості підготовки. В свою чергу, викладачі отримують можливість оптимізувати час, витрачаючи менше зусиль на технічні аспекти проведення онлайн-занять.

Результати експериментального дослідження підтвердили надійність та стабільність розробленої системи. Робота модулів була узгодженою на всіх етапах, від моменту запуску та ініціалізації до етапів обробки й формування підсумкового матеріалу. Програма продемонструвала здатність коректно взаємодіяти з інтерфейсами Google Classroom і Google Meet, адаптуючись до особливостей їхньої роботи. Створені відеофайли мали високу якість, а транскрипти забезпечували точну передачу змісту лекцій. Це доводить, що використання алгоритмів розпізнавання мовлення в поєднанні з автоматизованими механізмами керування дозволяє отримати результати, які відповідають вимогам сучасних освітніх технологій.

Дослідження також засвідчило, що застосування AutoMeetingManager має широкі перспективи для подальшого впровадження. Система може бути корисною в університетській практиці, науковій діяльності, під час проведення вебінарів, консультацій і семінарів. Вона може стати частиною інфраструктури електронного навчання, забезпечуючи стандартизований механізм фіксації та документування навчальних занять. В довгостроковій перспективі можливим є розвиток системи в напрямі автоматичного аналізу й узагальнення транскриптів, створення тематичних конспектів, формування коротких тез і підтримки адаптивного навчання.

Поєднання алгоритмів машинного навчання, технологій обробки мовлення та засобів автоматизації взаємодії з вебплатформами дозволяє створити інструмент, який здатний радикально змінити підхід до організації освітнього процесу в умовах цифровізації. Розроблена методика підтверджує, що автоматизація онлайн-лекцій не лише є технічно можливою, а й має значний потенціал для підвищення ефективності навчання, покращення доступності освітніх ресурсів і створення нових можливостей для студентів та викладачів.

Запропоноване рішення може розглядатися як приклад практичної реалізації принципів інтелектуальної автоматизації в освіті, що відкриває простір для подальших наукових розробок та вдосконалення цифрових освітніх сервісів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Гуржій А. М., Кухаренко В. М. Цифровізація освіти: сучасний стан і перспективи розвитку. Київ: ІПТО НАПН України, 2020. 112 с.
2. Кухаренко В. М., Бондаренко В. В. Теорія та практика змішаного навчання. Харків: Міськдрук, 2016. 284 с.
3. Єгорова О. В. Особливості читання лекцій онлайн. *Освітня аналітика України*. 2021. №2(13). С. 45–52.
4. Богун Є. О., Букач А. В., Ухань П. С. Комплексне застосування Google Classroom для створення варіативних дистанційних курсів. *Інформаційні технології і засоби навчання*. 2020. Т. 76, № 2. С. 28-40.
5. Demyanenko V. Adaptive Cloud-Oriented Learning System in Higher Education. Kyiv: NPU Dragomanova, 2020. 176 p.
6. Sharples M. (ed.) Artificial Intelligence in Education. Запоріжжя: ЗНУ, 2024. 312 с.
7. Google Classroom Help Center. Classroom Documentation. Google, 2024.
8. Google Meet Help Center. Meet Documentation. Google, 2024.
9. University of Birmingham. Guidance on the Use of Transcription and Translation Tools in Teaching. Birmingham, 2020. 14 p.
10. City, University of London. Effective Use of Lecture Capture. London, 2019. 28 p.
11. Selenium Project. Selenium WebDriver Documentation. 2024.
12. Shapiro A. Automating User Interfaces with PyAutoGUI. O’Reilly Media, 2023. 264 p.
13. OBS Project. OBS Studio User Guide. v29.0. 2023.
14. VB-Audio Software. VB-CABLE Virtual Audio Device – User Manual. 2023.
15. Jurafsky D., Martin J. Speech and Language Processing. Pearson, 2020. 1024 p.

16. Hinton G. et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition. *IEEE Signal Processing Magazine*, 2012. Vol. 29, No. 6. pp. 82-97.
17. Rabiner L. R. A Tutorial on Hidden Markov Models. *Proceedings of the IEEE*, 1989. Vol. 77, No. 2. pp. 257-286.
18. Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016. 800 p.
19. OpenAI. Whisper Speech Recognition System. 2022.
20. Faster-Whisper Documentation (CTranslate2). 2023.
21. RealtimeSTT Repository. GitHub, 2024.
22. Silero Team. Silero VAD: Voice Activity Detection Models. GitHub, 2021.
23. Google Cloud. Speech-to-Text API Documentation. Google, 2024.
24. Microsoft Azure. Speech to Text Documentation. 2024.
25. Manning C., Schütze H. Foundations of Statistical Natural Language Processing. MIT Press, 1999. 680 p.
26. Mihalcea R., Tarau P. TextRank. *EMNLP*, 2004. pp. 404-411.
27. Devlin J. et al. BERT: Pre-training of Deep Bidirectional Transformers. *NAACL*, 2019. pp. 4171-4186.
28. Lin C.-Y. ROUGE: A Package for Automatic Evaluation of Summaries. 2004. pp. 1-8.
29. Klein G. et al. OpenNMT. *ACL*, 2017. pp. 67-72.
30. Wang X. et al. Abstractive Summarization Using Transformer Architectures. *ACL*, 2020. pp. 745-754.
31. Telegram Messenger LLP. Telegram Bot API Documentation. 2024.
32. Python Software Foundation. Python 3 Standard Library Documentation. 2024.
33. Chrome Developers. ChromeDriver Documentation. 2024.
34. Kuhn K., Honeychurch S., Barros J. Measuring ASR Accuracy for Lecture Transcription. *ACM TACCESS*, 2024.
35. Dinmore S., Gao J. Voice-to-Text Transcription of Lecture Recordings. *ASCILITE*, 2016. pp. 197-202.

36. Shadiev R., Hwang W.-Y. Review of Speech-to-Text Recognition Technology. *Educational Technology & Society*, 2014. Vol. 17(4). pp. 65-77.
37. Khong P. et al. Lecture Capture Trends in Medical Education. *Medical Science Educator*, 2024. pp. 123-135.
38. University of Western Australia. Navigating Lecture Transcripts in LMS and Echo360. 2022. 18 p.
39. iSpring Solutions. What Is Lecture Capture? 2025.
40. Shaip Corporation. Automatic Speech Recognition: Complete Overview. 2025.

ДОДАТОК А

1. Головний модуль керування роботою системи AutoMeetingManager

```
import os
import time
from datetime import datetime, timedelta
from typing import NoReturn

from src.meeting_handler.meeting import MeetingHandler
from src.transcription.transcribe import audio_transcription
from src.utils.get_filenames import get_filenames


def run_schedule(meeting_handler):
    print("Run schedule")
    while True:
        # Retrieve the current time
        current_time = datetime.now().time()

        # Execute all pending tasks in the meeting scheduler
        meeting_handler.schedule.run_pending()

        # Check if the current time has surpassed the latest meeting end time plus 10 minutes
        # TODO: Address issues when meetings extend past midnight (next day).
        # !!!IMPORTANT!! Ensure 'get_latest_time' accommodates meeting schedules properly.
        if current_time >= (meeting_handler.get_latest_time() + timedelta(minutes=10)).time():
            if meeting_handler.threads and all(not t.is_alive() for t in meeting_handler.threads):
                print("✓ Meetings have ended. Time to make transcription.")
                meeting_handler.threads.clear()
                break

        time.sleep(1)


def main() -> NoReturn:
    # Create an instance of the meeting handler
    meeting_handler = MeetingHandler()

    # Schedule a sample meeting session (start and end times dynamically adjusted for testing)
    # TODO: Add functionality to manage multi-day scheduling for meetings.
    meeting_handler.create_meeting_session("13:00", "15:00")

    # Main loop to execute scheduled tasks and handle the meeting lifecycle
    try:
        run_schedule(meeting_handler)

        # Retrieve filenames of video recordings created today
        today_videos_filenames = get_filenames()

        # Perform transcription on the audio files
        # FIX:Sometimes the program closes after transcription, immediately
        audio_transcription(today_videos_filenames)

        # Wait for 5 minutes before initiating system shutdown
        # This shit below dose'nt work
        time.sleep(60 * 5)
        os.system("shutdown /s /t 1")

    # Handle manual interruption (e.g., Ctrl+C)
    except KeyboardInterrupt:
        print("\nProgram terminated by keyboard interrupt.")

    except Exception as e:
        print(e)
```

2. Модуля розпізнавання мовлення в реальному часі

```
import asyncio
import threading

import schedule
from rapidfuzz import process
from RealtimeSTT import AudioToTextRecorder
```

```
from
src.audio_to_text_transcriber.audio_t
o_text_transcriber import \
    AudioToTextTranscriber
from src.bot.handlers import
send_message_to_user
from
src.utils.escape_special_characters
import escape_special_characters
from configs.config import LANGUAGE,
SEARCH_WORDS, WHISPER_PROMPT

class WordSearcher:
    def __init__(self, search_words):
        # Initialize with a list of
words to search for
        self.search_words =
search_words

    def search_in_sentence(self,
sentence):
        # Split the sentence into
words for processing
        words = sentence.split(" ")

        # Loop through each search
word
        for search_word in
self.search_words:
            # Perform fuzzy matching
using rapidfuzz
            matches =
process.extract(search_word, words)

            # Filter matches with a
similarity score greater than 80%
            for match in matches:
                if len(match[0]) >=
len(search_word) - 1 and match[1] >
80:  # Only 80% match, no less
                    word =
escape_special_characters(match[0])
                    similarity =
escape_special_characters(str(match[1
]))
                    sentence =
escape_special_characters(sentence)

                    asyncio.run(

send_message_to_user(
                            f"You
have been mentioned\!\n"
                            f"Word:
{search_word} \({word}\)\.\n"

f"Similarity: {similarity}\.\n"

f">{sentence}\n"
                            f"__Audio
will be added__\n"
                            f"__Video
will be added__")
                    )


class SpeechProcessor:
    def __init__(self, searcher):
        # Initialize with a
WordSearcher instance
        self.searcher = searcher

    def text_detected(self, text):
        # Handle incomplete sentences
detected during real-time
transcription
        pass
        # Code below use only for
debug purpose
        # print(text)
        # thread =
threading.Thread(target=self.searcher
.search_in_sentence, args=(text,))
        # thread.start()

    def process_text(self, text):
        # Handle complete and
processed sentences
        thread =
threading.Thread(target=self.searcher
.search_in_sentence, args=(text,))
        thread.start()


class RealtimeSpeechRecognizer:
    def __init__(self):
        """
        A class for managing real-
time speech recognition with keyword
searching and transcription.
        """
        # List of keywords to search
for in the transcription
        self.searcher =
WordSearcher(SEARCH_WORDS)
        self.processor =
SpeechProcessor(self.searcher)
        self.schedule = schedule
        self.recorder_is_running =
True
        # Configuration for the
audio-to-text recorder
        self.recorder_config = {
            'spinner': False,  #
Disable spinner for real-time
transcription
            'model': 'medium',  #
Specify model type (medium in this
case)
            # TODO: Implement auto-
indexing for input devices
(RealtimeSTT uses PyAudio)
            'input_device_index': 0,
# Specify the input device index
            'realtime_model_type':
'medium',  # Model type for real-time
transcription
```

```
            'language': LANGUAGE,  # Language set to Ukrainian
            'silero_sensitivity': 0.05,  # Sensitivity for Silero VAD
            'webrtc_sensitivity': 3,  # Sensitivity for WebRTC

'post_speech_silence_duration': 0.7,  # Silence duration to consider as the end of speech

'min_length_of_recording': 1.1,  # Minimum length of recording in seconds

'min_gap_between_recordings': 0,  # Minimum gap between consecutive recordings

'enable_realtime_transcription': True,  # Enable real-time transcription

'realtime_processing_pause': 0.02,  # Pause duration between real-time processing

'on_realtime_transcription_update': self.processor.text_detected,  # Callback for partial transcription

'silero_deactivity_detection': True,  # Enable Silero activity detection

'early_transcription_on_silence': 0,  # Trigger early transcription on silence
            'beam_size': 5,  # Beam size for full transcription
            'beam_size_realtime': 3,  # Beam size for real-time transcription
            'no_log_file': True,  # Disable log file creation
            'initial_prompt': WHISPER_PROMPT
        }
        # Initialize the recorder
        self.recorder = None
        self.is_recording = True

    def start_recorder(self):
        print("AudioToTextRecorder start")
        self.recorder = AudioToTextTranscriber(**self.recorder_config)
        while self.is_recording:
            # Process real-time transcription with the provided processor

self.recorder.text(self.processor.process_text)

    def shutdown(self):
        # Stops the recorder and cleans up resources.
        self.recorder.shutdown()

        # Stoping of Transcriber loop
        self.is_recording = False

        print("AudioToTextRecorder shutdown")
```

3. Модуль керування онлайн-зустрічами, автоматичного приєднання та контролю перебігу лекцій у системі AutoMeetingManager

```
import threading
from datetime import datetime, timedelta

import schedule

from src.conversation.conversation import Conversation
from src.recorders.recorder import Recorder


class MeetingHandler:
    def __init__(self):
        # List to store meeting end times
        self.meetings_end_time = []
        self.threads = []

        self.schedule = schedule.Scheduler()
        self.conversation = Conversation()

    def get_latest_time(self):
        # Returns the latest meeting end time
        return max(self.meetings_end_time)


    def create_meeting_session(self, start_time: str, end_time: str):
        """
        Meeting main cycle.

        Asynchronously schedules tasks for managing a meeting session.

        Joining and leaving a meeting is performed asynchronously. This is
```

```
chosen because real-time
transcription
        starts another loop.
multiprocessing to improve
performance does not work.

        Args:
            start_time (str): Meeting
start time in "HH:MM" format.
            end_time (str): Meeting
end time in "HH:MM" format.
        """
        recorder = Recorder()
        # Add the meeting end time to
the list of end times

self.meetings_end_time.append(

datetime.strptime(end_time, "%H:%M")
        )

        # Schedule the recording to
start at the specified start time

self.schedule.every().day.at(start_ti
me).do(self.run_threaded,
recorder.start_record)

        # Schedule joining the
conversation 5 minutes after the
start time
        self.schedule.every().day.at(

(datetime.strptime(start_time,
"%H:%M") +
timedelta(minutes=2)).strftime("%H:%M
")
        ).do(self.run_threaded,
self.conversation.join)

        # Schedule quitting the
conversation at the specified end
time

self.schedule.every().day.at(end_time
).do(self.run_threaded,
self.conversation.quit)

        # Schedule stopping the
recording 5 minutes after the end
time
        self.schedule.every().day.at(

(datetime.strptime(end_time, "%H:%M")
+
timedelta(minutes=2)).strftime("%H:%M
")
        ).do(self.run_threaded,
recorder.stop_record)

    def run_threaded(self, job_func,
*args, **kwargs):
        job_thread =
threading.Thread(target=job_func,
args=args, kwargs=kwargs)
        job_thread.start()

self.threads.append(job_thread)
```