

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____ Комп'ютерних наук _____
(повна назва)
Кафедра _____ Штучного інтелекту _____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____ перший (бакалаврський) _____

_____ Інтелектуальна система персоналізованих нагадувань _____
_____ для інклюзивної родинної комунікації _____
(тема)

Виконав:
здобувач _____ четвертого _____ року навчання,
групи _____ ІТШ-22-1 _____

_____ Тимофій ТРУЩЕНКО _____
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки _____

(код і повна назва спеціальності)

Тип програми освітньо-професійна _____

Освітня програма Штучний інтелект _____

(повна назва освітньої програми)

Керівник ас. Сергій ГРИНЬОВ _____
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ

_____ Лариса ЧАЛА _____
(підпис) (власне ім'я, прізвище)

2026 р.

Харківський національний університет радіоелектроніки

Факультет _____ Комп'ютерних наук _____

Кафедра _____ Штучного інтелекту _____

Рівень вищої освіти _____ перший (бакалаврський) _____

Спеціальність _____ 122 Комп'ютерні науки _____

(код і повна назва)

Тип програми _____ освітньо-професійна _____

Освітня програма _____ Штучний інтелект _____

(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

« _____ » _____ 20 ____ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві _____ Трущенко Тимофію Олександровичу _____

(прізвище, ім'я, по батькові)

1. Тема роботи _____ Інтелектуальна система персоналізованих нагадувань для інклюзивної родинної комунікації _____

затверджена наказом університету від 5 червня 2026 р. № 488Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 9 липня 2026 р.

3. Вихідні дані до роботи науково-технічні публікації з інклюзивної комунікації, рекомендації з доступності інтерфейсів, матеріали щодо систем нагадувань, документація з методів NLP та великих мовних моделей, вимоги до захисту персональних даних, сценарії родинної взаємодії.

4. Перелік питань, що потрібно опрацювати в роботі _____

1) Аналіз предметної галузі та теоретичних основ

2) Обґрунтування методів та технологій розробки

3) Розробка інтелектуальної системи персоналізованих нагадувань

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / термін виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	08.06.2026	виконано
2	Аналіз предметної галузі та теоретичних джерел	09.06.2026	виконано
3	Огляд існуючих систем нагадувань	10.06.2026	виконано
4	Аналіз методів NLP та LLM	11.06.2026	виконано
5	Аналіз вимог до доступності інтерфейсів	13.06.2026	виконано
6	Формування вимог до системи	15.06.2026	виконано
7	Оформлення пояснювальної записки	20.06.2026	виконано
8	Перевірка на академічний плагіат	23.06.2026	виконано
9	Нормоконтроль	24.06.2026	виконано
10	Підготовка презентації та доповіді	29.06.2026	виконано
11	Попередній захист	07.07.2026	виконано
12	Захист перед ЕК	09.07.2026	

Дата видачі завдання 8 червня 20 26 р.

Здобувач 
(підпис)

Керівник роботи 
(підпис)

ас. Сергій ГРИНЬОВ
(посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 78 с., 5 рис., 19 табл., 1 дод., 52 джерела.

ДОСТУПНІСТЬ, ІНКЛЮЗИВНА КОМУНІКАЦІЯ, КОГНІТИВНА ДОСТУПНІСТЬ, НАГАДУВАННЯ, ОБРОБКА ПРИРОДНОЇ МОВИ, ПЕРСОНАЛІЗАЦІЯ, РОДИННА КОМУНІКАЦІЯ, ШТУЧНИЙ ІНТЕЛЕКТ.

Об'єкт дослідження – процес організації інклюзивної родинної комунікації за допомогою цифрових нагадувань, адаптивного подання інформації та контролю виконання повсякденних доручень.

Предмет дослідження – методи персоналізації нагадувань, обробки природної мови, структурування голосових доручень і забезпечення доступності інтерфейсу для користувачів із різними потребами сприйняття інформації.

Мета роботи – проаналізувати предметну галузь і теоретичні основи створення інтелектуальної системи персоналізованих нагадувань для інклюзивної родинної комунікації та обґрунтувати вимоги до такого рішення.

Методи дослідження – аналіз літературних джерел, порівняння існуючих систем нагадувань, узагальнення вимог доступності, аналіз методів NLP та великих мовних моделей, формалізація користувацьких потреб.

У роботі розглянуто потреби дітей, літніх людей, користувачів із когнітивними труднощами та опікунів. Виконано порівняння стандартних органайзерів, голосових асистентів, медичних нагадувачів і спеціалізованих інклюзивних застосунків. Результатом є перелік вимог до майбутньої системи: родинні ролі, голосове і текстове створення нагадувань, адаптація повідомлень до профілю адресата, журнал виконання і захист персональних даних.

ABSTRACT

Bachelor's thesis contains: 78 pp., 5 fig., 19 tabl., 1 ann., 52 references.

ACCESSIBILITY, ARTIFICIAL INTELLIGENCE, COGNITIVE ACCESSIBILITY, FAMILY COMMUNICATION, INCLUSIVE COMMUNICATION, LARGE LANGUAGE MODELS, NATURAL LANGUAGE PROCESSING, PERSONALIZED REMINDERS.

The object of research is the process of organizing inclusive family communication by means of digital reminders, adaptive information presentation and control of everyday task completion.

The subject of research is methods for reminder personalization, natural language processing, structuring of voice instructions and interface accessibility for users with different information perception needs.

The purpose of the work is to analyze the subject area and theoretical foundations for creating an intelligent system of personalized reminders for inclusive family communication and to justify the requirements for such a solution.

The research methods include literature analysis, comparison of existing reminder systems, generalization of accessibility requirements, analysis of NLP methods and large language models, and formalization of user needs.

The work considers the needs of children, older adults, users with cognitive difficulties and caregivers. It compares standard organizers, voice assistants, medical reminder applications and specialized inclusive tools. The result is a set of requirements for the future system: family roles, voice and text reminder creation, message adaptation, execution log and personal data protection.

ЗМІСТ

Вступ.....	8
1 Аналіз предметної галузі та теоретичних основ	11
1.1 Інклюзивна комунікація: поняття, сучасні підходи та виклики для вразливих груп.....	11
1.2 Огляд існуючих систем нагадувань	17
1.3 Методи обробки природної мови та великі мовні моделі у задачах персоналізації контенту	23
1.4 Аналіз вимог до доступності інтерфейсів	29
2 Обґрунтування методів та технологій розробки.....	35
2.1 Загальний підхід до розробки системи	35
2.2 Функціональна модель та ролі користувачів	37
2.3 Обробка природномовних доручень	39
2.4 Персоналізація та адаптація повідомлень	41
2.5 Технології клієнтської частини	43
2.6 Серверна частина, база даних і сповіщення.....	45
2.7 Захист даних і контроль доступу.....	47
2.8 Оцінювання якості обраних рішень	49
3 Розробка інтелектуальної системи персоналізованих нагадувань.....	51
3.1 Загальна структура та склад реалізації	51
3.2 Серверна частина та API	53
3.3 Модель даних та збереження інформації	55
3.4 Обробка текстових і голосових нагадувань	57
3.5 Мобільний застосунок та адаптивні інтерфейси	59
3.6 Сповіщення, підтвердження виконання та журнал подій.....	62
3.7 Перевірка працездатності та обмеження реалізації.....	66
3.8 Результати розробки	68
Висновки	70
Перелік джерел посилання	73

Додаток А Відомість кваліфікаційної роботи	78
--	----

ВСТУП

Цифрові сервіси вже стали звичним способом родинної взаємодії. Через смартфони, месенджери, календарі й голосові асистенти передаються короткі доручення, нагадування про навчання, побутові справи, ліки або зустрічі. Проте сама наявність цифрового каналу ще не означає, що повідомлення буде зрозумілим для кожного члена родини. Дитина, літня людина або користувач із когнітивними труднощами може отримати правильно сформоване сповіщення, але не зрозуміти, що саме потрібно зробити, коли це зробити і як підтвердити результат.

Актуальність теми пов'язана з тим, що штучний інтелект поступово переходить від абстрактних технологічних демонстрацій до практичних інструментів підтримки людини у щоденних ситуаціях. Методи обробки природної мови дають змогу визначати намір у фразі, виділяти ключові сутності, спрощувати пояснення і підлаштовувати текст під конкретного адресата [1], [2]. У родинному середовищі це особливо цінно: опікун може сказати доручення природною мовою, а система перетворює його на зрозуміле нагадування для дитини, літньої людини або іншого члена сім'ї.

Проблема інклюзивної родинної комунікації має не тільки технічний характер. Вона стосується турботи, самостійності, довіри та щоденного розподілу відповідальності між близькими людьми. У багатьох сім'ях саме родичі або опікуни допомагають дітям, літнім людям чи людям із тимчасовими обмеженнями виконувати побутові, навчальні або медичні дії [3], [4]. Те, що для одного користувача є простою фразою, для іншого може виявитися надто довгою, абстрактною або незручною через дрібний шрифт, відсутність озвучення, складний інтерфейс чи велику кількість кроків [5], [6]. Звичайні календарі та нагадувачі зазвичай працюють із часом, датою і коротким текстом. Вони добре виконують роль сигналу, але не завжди допомагають зрозуміти зміст доручення [7], [8], [9]. Наприклад, фраза «підготуйся до школи» може бути достатньою для старшого школяра,

але молодшій дитині її доведеться розкласти на конкретні дії: зібрати зошити, перевірити пенал, покласти форму і натиснути кнопку підтвердження. Саме тому для аналізу важливо врахувати можливості календарів, списків справ, сімейних органайзерів, голосових асистентів і спеціалізованих нагадувачів [10], [11], [12].

У межах цієї роботи інтелектуальна система персоналізованих нагадувань розглядається як рішення для родини або малої групи опікунів. Вона поєднує голосове чи текстове введення, автоматичне структурування доручення, формування нагадування, адаптацію повідомлення і контроль виконання. Важливо, що система працює не з одним універсальним користувачем, а з різними ролями: опікуном, адресатом, адміністратором родинного простору та спостерігачем.

Об'єктом дослідження є процес організації інклюзивної родинної комунікації за допомогою цифрових нагадувань, адаптивного подання інформації та контролю виконання повсякденних доручень. У межах об'єкта увага зосереджується на взаємодії між опікуном і адресатом, коли повідомлення має бути не просто доставлене, а зрозуміле, своєчасне, безпечне і придатне для виконання конкретною людиною.

Предметом дослідження є методи персоналізації нагадувань, обробки природної мови, структурування голосових доручень і забезпечення доступності інтерфейсу для користувачів із різними потребами сприйняття інформації. Предмет охоплює мовні моделі, правила аналізу часових виразів, принципи когнітивної доступності, рольову модель доступу, механізми підтвердження виконання та критерії якості адаптованого повідомлення.

Метою роботи є аналіз предметної галузі та теоретичних основ створення інтелектуальної системи персоналізованих нагадувань для інклюзивної родинної комунікації, а також обґрунтування вимог до такого рішення. Досягнення цієї мети передбачає встановлення зв'язку між

інклюзивною комунікацією, системами нагадувань, методами NLP, великими мовними моделями та вимогами доступності інтерфейсів.

Для досягнення мети було визначено такі завдання: розглянути поняття інклюзивної комунікації та сучасні підходи до доступного подання інформації; проаналізувати потреби вразливих груп користувачів у родинному середовищі; виконати огляд існуючих систем нагадувань; оцінити можливості NLP і великих мовних моделей для персоналізації контенту; систематизувати вимоги WCAG і когнітивної доступності; сформулювати висновки для подальшого проектування системи. Наукова новизна роботи полягає у комплексному підході до побудови системи нагадувань, у якій персоналізація розглядається не лише як налаштування часу або каналу сповіщення, а як адаптація змісту, форми подання, рівня деталізації, способу підтвердження і сценарію повторення залежно від профілю адресата. Особливість підходу полягає у поєднанні родинної рольової моделі, обробки природномовних доручень і принципів когнітивної доступності. Практичне значення роботи полягає в тому, що її результати можна використати як основу для мобільного застосунку та серверної частини системи. Такий підхід може бути корисним для сімей, опікунів, соціальних працівників, освітніх асистентів і невеликих груп підтримки, де важливо не просто передати доручення, а зробити його зрозумілим, контрольованим і прийнятним для конкретної людини [4], [13].

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ТЕОРЕТИЧНИХ ОСНОВ

1.1 Інклюзивна комунікація: поняття, сучасні підходи та виклики для вразливих груп

Інклюзивна комунікація передбачає таке подання інформації, за якого враховуються різні можливості сприйняття, вік, досвід і цифрові навички користувачів. У цифровому середовищі це поняття охоплює не лише технічну доступність інтерфейсу, а й зрозумілість змісту. Для системи нагадувань це означає, що повідомлення має бути помітним, коротким, своєчасним і таким, яке можна виконати без додаткових пояснень [14], [15].

Сучасне розуміння інклюзії ґрунтується на відході від суто медичної моделі інвалідності до соціальної та правозахисної моделі. У такому підході проблема полягає не тільки у функціональних особливостях людини, а й у бар'єрах середовища. Цифровий сервіс може створювати бар'єр, якщо він використовує дрібний шрифт, складні формулювання, невидимі кнопки, непередбачувану навігацію або надмірну кількість кроків. Водночас правильно спроектована система може зменшувати залежність користувача від постійних пояснень з боку інших людей [3], [16].

У родинній комунікації інклюзивність має додатковий вимір, оскільки взаємодія відбувається не між абстрактним користувачем і сервісом, а між близькими людьми з різними ролями, очікуваннями та рівнем відповідальності. Опікун може формулювати доручення поспіхом, використовуючи неповні фрази, побутові скорочення або контекст, зрозумілий тільки сім'ї. Адресат, навпаки, потребує повідомлення, у якому явно зазначено, що потрібно зробити, коли саме, у якій послідовності й що означає успішне завершення. У контексті цієї роботи до вразливих груп належать не лише люди з офіційно встановленою інвалідністю. Практично в кожній родині можуть бути діти молодшого віку, літні люди, підлітки з труднощами самоорганізації, користувачі після хвороби або люди, які в

певний момент перебувають у стані втоми чи тривожності. Спільною рисою таких ситуацій є чутливість до інформаційного перевантаження [4], [17].

Особливістю родинних доручень є їхня практична спрямованість. Вони стосуються навчання, здоров'я, побуту, харчування, безпеки, дозвілля та спільного планування. Деякі нагадування мають низьку критичність, наприклад прохання прибрати іграшки або полити квіти. Інші можуть мати високий рівень важливості, наприклад приймання ліків, вихід до школи, заряджання медичного пристрою або перевірка дверей. Інклюзивна система повинна враховувати різні рівні критичності і не застосовувати однаковий сценарій для всіх повідомлень. Ключовим принципом інклюзивної комунікації є адаптація без приниження користувача. Повідомлення для дитини або літньої людини не повинно бути примітивним чи зверхнім. Воно має залишатися поважним, але простішим за структурою. Наприклад, замість складного речення «не забудь після завершення занять виконати домашнє завдання і повідомити мені про результат» система може сформувати короткі дії: «Зроби домашнє завдання. Перевір відповідь. Натисни кнопку “Готово”». Такий підхід зберігає зміст, але знижує когнітивне навантаження. Для інклюзивної комунікації важливе поняття когнітивного навантаження. Користувач повинен витратити мінімум зусиль на розуміння інструкції і максимум уваги спрямовувати на виконання дії. Надмірна кількість тексту, непослідовність, складні часові формулювання і непередбачувані елементи керування збільшують навантаження. Тому система нагадувань має перетворювати природне повідомлення на структуру, у якій окремо збережено дію, час, адресата, повторення, важливість і спосіб підтвердження [15], [17].

Сучасні підходи до інклюзивного дизайну часто спираються на принципи універсального дизайну, персоналізації та участі користувачів у проєктуванні. Універсальний дизайн передбачає створення рішень, придатних для максимально широкого кола людей без необхідності окремої адаптації. Однак у випадку родинних нагадувань повної універсальності

недостатньо, тому що різні користувачі потребують різної форми подання. Саме тому доцільно поєднувати універсальні вимоги до інтерфейсу з персональними профілями доступності [16], [18].

Профіль доступності може містити вік користувача, бажаний розмір шрифту, потребу в голосовому озвученні, допустиму кількість кроків, бажані піктограми, мову повідомлень, тип підтвердження і частоту повторів. Для дитини профіль може передбачати короткі фрази, візуальне підкріплення і позитивний тон. Для літньої людини – великий шрифт, контраст, кнопку «прослухати» і мінімум дрібних елементів. Для користувача зі стандартними потребами достатньо звичайного текстового нагадування. Інклюзивна комунікація також потребує надійного зворотного зв'язку. Якщо адресат не зрозумів повідомлення, пропустив його або не підтвердив виконання, система повинна допомогти опікуну виявити проблему. При цьому контроль не має перетворюватися на надмірний нагляд. Необхідний баланс між турботою, приватністю і самостійністю користувача. У родинному середовищі цей баланс особливо важливий, оскільки система працює з повсякденними діями, емоціями та довірою між членами родини. Уразливі користувачі часто стикаються з бар'єрами не через відсутність технологій, а через їхню фрагментарність. Календар може нагадати про час, месенджер може передати повідомлення, камера може зафіксувати результат, а голосовий асистент може створити подію. Однак ці інструменти рідко об'єднані у сценарій, де доручення створюється природною мовою, автоматично структурується, адаптується до адресата, повторюється за потреби і зберігається в журналі виконання.

Для дітей важливими викликами є концентрація уваги, мотивація, розуміння абстрактних формулювань і здатність планувати час. Дитина може знати, що потрібно зробити уроки, але не розуміти, з чого почати. Тому нагадування має не лише сигналізувати, а й направляти. Воно може включати покрокову інструкцію, піктограму, короткий таймер, дружній текст і просте підтвердження. При цьому система не повинна

перевантажувати дитину зайвими функціями або дозволяти випадково змінювати критичні налаштування.

Для літніх людей типовими бар'єрами є зниження зору, слуху, пам'яті, дрібної моторики та швидкості обробки інформації. Застосунок має уникати дрібних кнопок, багаторівневих меню, прихованих жестів і повідомлень, які швидко зникають. Нагадування повинно залишатися доступним після першого сигналу, повторюватися у визначених межах і за потреби передавати інформацію опікуну. Особливо важливими є прості дії: «прослухати», «виконано», «потрібна допомога».

Для опікунів основний виклик полягає в координації багатьох дрібних задач і зменшенні кількості повторних пояснень. Опікун повинен швидко створювати доручення, не витрачаючи час на складні форми. Водночас він має бути впевнений, що система правильно зрозуміла команду і не змінила зміст критичного повідомлення. Тому інтелектуальний модуль має працювати прозоро: показувати розпізнаний текст, структуру доручення, адресата, час і варіант адаптованого повідомлення.

Важливим аспектом є мова комунікації. У родині можуть використовуватися українська, суржик, побутові скорочення, імена, родинні ролі та неформальні позначення часу: «після школи», «перед сном», «коли повернешся», «через пів години». Система повинна вміти або обробляти такі формулювання, або ставити уточнювальні запитання. Інклюзивність у цьому випадку означає не вимагати від користувача ідеальної команди, а допомагати перетворити звичайне мовлення на коректне нагадування.

Отже, інклюзивна родинна комунікація не зводиться до звичайної доставки повідомлення. Вона поєднує соціальні, мовні, психологічні та технічні аспекти: зрозумілість, доречний тон, зручний канал подання, повторення, підтвердження і право користувача на приватність.

Саме ці чинники визначають потреби основних груп користувачів (таблиця 1.1).

Таблиця 1.1 – Бар'єри та потреби користувачів у родинній комунікації

Група користувачів	Типові бар'єри	Потреби у системі нагадувань
Дитина	Складність планування, коротка концентрація уваги, потреба у мотивації	Короткі інструкції, піктограми, покроковість, позитивне підкріплення
Літня людина	Зниження зору, слуху, пам'яті, складність роботи з дрібними елементами	Великий шрифт, озвучення, контраст, повтори, просте підтвердження
Користувач із когнітивними труднощами	Перевантаження текстом, складні часові вирази, труднощі з абстрактними діями	Простий текст, один крок на екрані, передбачувана навігація, уточнення
Опікун	Велика кількість доручень, потреба контролю, ризик помилки в критичних задачах	Голосове створення, журнал виконання, ролі, підтвердження структури команди

Дані таблиці показують, що одна функція нагадування потребує різної реалізації залежно від адресата. Для опікуна важливі швидкість створення і контроль виконання, для дитини – зрозумілі кроки, для літньої людини – помітний сигнал і просте підтвердження. Тому предметну галузь не можна описати лише як планування подій; вона пов'язана з людським фактором, доступністю і адаптацією змісту.

Систему персоналізованих нагадувань доцільно розглядати як посередника між мовою опікуна і потребами адресата. Такий посередник не повинен замінювати живу комунікацію, але може зменшити кількість непорозумінь. Наприклад, якщо опікун говорить «нагадай Денису після школи зробити математику і надіслати фото», система виділяє адресата, час, предмет, дію і тип підтвердження. Після цього вона створює для дитини коротке повідомлення з конкретними кроками. У сучасних підходах до доступності дедалі більше уваги приділяється когнітивним аспектам. Веб-стандарти традиційно докладно описують контраст, клавіатурну навігацію, альтернативний текст і структуру сторінок, проте для реального користування не менш важливими є зрозумілість мови, передбачуваність, підтримка пам'яті та зменшення кількості рішень, які користувач має прийняти одночасно. У системі нагадувань ці вимоги проявляються у вигляді коротких повідомлень, повторюваних сценаріїв і можливості повернутися до інструкції [14], [15], [17].

Окремим викликом є етична сторона використання штучного інтелекту. Якщо система автоматично спрощує або переформулює доручення, вона може ненавмисно втратити важливу умову, змінити тон або зробити неправильне припущення. Тому адаптація повинна бути контрольованою: критичні доручення потребують підтвердження опікуна, а система має зберігати вихідний текст. Це особливо важливо для нагадувань про здоров'я, безпеку, фінансові дії або вихід за межі дому [2], [19], [20].

Підсумовуючи, теоретична основа інклюзивної комунікації визначає першу групу вимог до системи: зрозумілість, персоналізацію, прозорість, повагу до користувача, підтримку різних каналів подання та контроль критичних дій. Далі потрібно перевірити, наскільки ці вимоги вже покривають існуючі нагадувачі, календарі, органайзери і спеціалізовані рішення.

1.2 Огляд існуючих систем нагадувань

Системи нагадувань є поширеним класом цифрових інструментів, які допомагають користувачам планувати події, керувати справами та отримувати сигнали у визначений час. До таких систем належать календарі, списки задач, будильники, нотатки, сімейні планувальники, медичні застосунки та голосові асистенти. На перший погляд, вони вирішують схожу задачу – не дати користувачу забути про дію. Проте для інклюзивної родинної комунікації цього недостатньо, оскільки важливо не тільки нагадати, а й подати повідомлення у зрозумілій для адресата формі. Стандартні органайзери орієнтовані переважно на індивідуального користувача. Вони дозволяють створювати події, списки справ, повтори, пріоритети, повідомлення і синхронізацію між пристроями. Такі системи ефективні для людей, які самостійно керують своїм часом і розуміють зміст власних задач. Однак вони менш придатні для сценарію, коли один член родини створює доручення для іншого, а система повинна враховувати профіль доступності адресата, право опікуна на контроль і необхідність підтвердження виконання. Поширені календарі, такі як Google Calendar, Microsoft Outlook Calendar або Apple Calendar, забезпечують добре налагоджену роботу з подіями, запрошеннями, повтореннями та синхронізацією. Їхня сильна сторона полягає у надійному плануванні часу. Водночас календарна модель зазвичай передбачає подію, а не адаптоване доручення. Подія має назву, опис, час і учасників, але не завжди містить механізм перетворення складної інструкції на покрокове повідомлення для конкретної вразливої групи [5].

Списки справ і нагадувачі, наприклад Apple Reminders, Google Tasks, Microsoft To Do або Todoist, краще підходять для коротких дій. Вони можуть підтримувати спільні списки, призначення задач, дедлайни, підзадачі й повтори. Проте навіть у таких системах адаптація змісту обмежена. Користувач сам вводить текст, сам визначає структуру і сам відповідає за

зрозумілість. Якщо доручення створює опікун для дитини або літньої людини, звичайний список справ не гарантує, що адресат правильно інтерпретує повідомлення [6], [7], [8].

Сімейні органайзери, такі як Cozi Family Organizer або подібні застосунки, додають спільний календар, списки покупок, сімейні задачі та розподіл відповідальності. Вони краще відповідають родинному контексту, але переважно залишаються універсальними інструментами планування. Їхня логіка не сфокусована на інклюзивній адаптації повідомлень. Для дитини або літньої людини такі системи можуть бути занадто складними, якщо інтерфейс не має спеціальних режимів, голосового дублювання або зниженого когнітивного навантаження [9].

Медичні нагадувачі вирішують вузьку, але важливу задачу – нагадування про приймання ліків, візити до лікаря, вимірювання показників або дотримання режиму лікування. Їхні переваги полягають у повторюваності, критичності сповіщень і можливості історії виконання. Проте вони часто не призначені для широкого спектра родинних доручень. Крім того, медичні застосунки повинні бути особливо обережними щодо рекомендацій: вони можуть нагадувати про вже визначену дію, але не повинні самостійно змінювати медичні призначення [12].

Голосові асистенти, такі як Siri, Google Assistant або Alexa, спростили створення нагадувань завдяки природному мовленню. Користувач може сказати фразу, і система створить подію або задачу. Такий спосіб введення є зручним для опікуна, який не хоче заповнювати форму вручну. Однак голосові асистенти здебільшого орієнтовані на персональний обліковий запис і не завжди забезпечують тонке керування родинними ролями, профілями доступності, підтвердженням виконання або збереженням адаптованого варіанта повідомлення [10], [11].

Спеціалізовані застосунки для людей з аутизмом, когнітивними труднощами або порушеннями пам'яті часто використовують візуальні розклади, піктограми, таймери, соціальні історії та покрокові інструкції.

Вони демонструють важливість адаптивного подання, однак можуть бути занадто вузькими або вимагати значного ручного налаштування. Інтелектуальна система родинних нагадувань має поєднати гнучкість стандартних органайзерів із доступністю спеціалізованих рішень, не змушуючи опікуна щоразу вручну створювати повну інструкцію [15], [17].

Порівняння стандартних і спеціалізованих систем показує, що проблема полягає не у відсутності окремих функцій, а у відсутності їхньої інтеграції. На ринку є голосове введення, спільні календарі, повтори, підтвердження, візуальні підказки й профілі доступності. Однак ці елементи рідко працюють як єдиний сценарій, у якому природномовне доручення автоматично перетворюється на структуроване нагадування і персоналізується для конкретного адресата. Для майбутньої системи важливо врахувати сильні сторони існуючих рішень. Від календарів доцільно взяти надійне планування часу, повтори і синхронізацію. Від списків справ – простоту створення та статуси виконання. Від сімейних органайзерів – спільний простір і ролі. Від медичних нагадувачів – підвищену увагу до критичних задач і журнал виконання. Від спеціалізованих інклюзивних застосунків – покроковість, піктограми, великий шрифт і зниження інформаційного шуму.

Обмеження стандартних нагадувачів особливо помітні у сценаріях з кількома адресатами. Якщо в родині є дитина, літня людина і опікун, один і той самий текст може бути непридатним для всіх. Опікун хоче бачити повний опис і статус, дитина – коротку інструкцію, а літня людина – великий текст і кнопку озвучення. Стандартні системи зазвичай не створюють різні представлення одного доручення, а просто показують однаковий текст усім учасникам або зберігають задачу в одному списку. Іншим обмеженням є недостатня робота з неповними командами. У реальному мовленні опікун може сказати: «Нагадай бабусі таблетки ввечері» або «Скажи малому після школи зробити математику». Для людини зміст зрозумілий з контексту, але система має уточнити час,

адресата, повторення, назву препарату або спосіб підтвердження. Більшість простих нагадувачів або створює неповну задачу, або вимагає від користувача самостійно редагувати її у формі. Для інклюзивного застосунку важливо не тільки створити нагадування, а й підтримати його життєвий цикл. Він складається зі створення доручення, перевірки структури, вибору адресата, адаптації тексту, доставки сповіщення, повторення за потреби, підтвердження виконання, збереження доказу та повідомлення опікуна про результат. Якщо хоча б один етап відсутній, система може втратити практичну цінність у родинному середовищі.

Окрему групу становлять рішення для догляду за літніми людьми, які можуть підтримувати моніторинг, екстрені виклики, нагадування про ліки та сповіщення родичів. Такі системи ближчі до потреб опікуна, однак часто мають медико-соціальну спеціалізацію і можуть бути надмірними для повсякденних родинних доручень. Майбутня система повинна залишатися легкою і придатною для щоденного використання, але водночас мати механізми підвищеної уваги до важливих дій. З погляду інтерфейсу багато стандартних застосунків мають значну кількість функцій, які корисні досвідченому користувачу, але створюють бар'єри для вразливих груп. Наявність вкладок, фільтрів, кольорових міток, підзадач, інтеграцій і налаштувань може ускладнювати взаємодію. Інклюзивний режим має приховувати зайве і показувати тільки найближчу дію. Для опікуна, навпаки, доцільно залишити розширені налаштування у вебкабінеті або адміністративному режимі. Важливим критерієм порівняння є підтримка української мови і природних фраз. Багато міжнародних застосунків мають локалізацію інтерфейсу, але не завжди однаково якісно обробляють українські часові вирази, відмінки і побутові формулювання. Для системи, орієнтованої на українські родини, потрібно враховувати мовні особливості: імена, кличний відмінок, слова «сьогодні», «завтра», «після обіду», «перед сном», «через пів години», а також можливі помилки розпізнавання мовлення [21].

Отже, наявні рішення створюють важливу технологічну базу, але не закривають задачу персоналізованих нагадувань для інклюзивної родинної комунікації повністю. Майбутня система має поєднувати можливості органайзера, сімейного простору, голосового асистента, адаптивного інтерфейсу і журналу виконання. Порівняння груп таких рішень подано нижче.

З порівняння видно, що жодна група рішень не охоплює всі вимоги одночасно. Найбільш перспективним є не створення ще одного календаря, а поєднання кількох підходів: простого екрана для адресата, гнучкого керування для опікуна і механізмів адаптації повідомлення. Саме такий поділ ролей дає змогу зменшити когнітивне навантаження і не втратити контроль над важливими дорученнями. Практичний аналіз також показує, що спеціалізовані функції мають сенс лише тоді, коли вони вбудовані в природний сценарій використання. Якщо для створення адаптованого нагадування опікуну потрібно заповнити багато полів, завантажити піктограму, обрати стиль повідомлення і вручну сформулювати кроки, система не зменшує, а збільшує навантаження. Тому ключовою вимогою є автоматичне або напівавтоматичне структурування доручення. Існуючі рішення також демонструють потребу в надійному механізмі підтвердження. У звичайному списку справ позначка «виконано» може бути достатньою, але для родинного догляду інколи потрібні коментар, фотографія, час виконання або повідомлення опікуна. З іншого боку, не кожне доручення потребує складного контролю. Тому система має підтримувати різні типи підтвердження: просте натискання, текстовий коментар, фото або відсутність підтвердження для неважливих задач.

Загалом огляд існуючих систем обґрунтовує потребу в інтелектуальному модулі, який не обмежується збереженням тексту.

Він має аналізувати природномовну команду, виявляти неповні дані, пропонувати уточнення, перетворювати повідомлення на структуроване доручення і формувати адаптований варіант для адресата (таблиця 1.2).

Таблиця 1.2 – Порівняння груп існуючих рішень для нагадувань

Група рішень	Сильні сторони	Обмеження для інклюзивної родинної комунікації
Календарі	Планування часу, повтори, синхронізація, запрошення	Слабка адаптація змісту, орієнтація на подію, а не на доступне доручення
Списки справ	Простота задач, статуси, спільні списки, пріоритети	Користувач сам відповідає за зрозумілість тексту і структуру інструкції
Голосові асистенти	Швидке створення нагадувань природною мовою	Обмежене керування родинними ролями, підтвердженням і профілями доступності
Медичні нагадувачі	Критичні повтори, історія виконання, акцент на безпеці	Вузька спеціалізація, неповна придатність для навчальних і побутових задач
Інклюзивні застосунки	Піктограми, покроковість, знижене когнітивне навантаження	Часто потребують ручного налаштування і не охоплюють усю родинну комунікацію

Саме це приводить до наступного питання – які методи обробки природної мови і великі мовні моделі можуть бути використані для персоналізації контенту.

1.3 Методи обробки природної мови та великі мовні моделі у задачах персоналізації контенту

Обробка природної мови є ключовою технологічною основою інтелектуальної системи персоналізованих нагадувань. Користувачі в родині не завжди хочуть або можуть заповнювати формальні поля. Їм природніше сказати або написати доручення звичайною мовою: «Нагадай доньці після тренування випити воду», «Скажи бабусі о восьмій прийняти ліки», «Щосуботи нагадуй прибрати кімнату». Завдання NLP полягає у перетворенні таких фраз на формальну структуру, придатну для збереження, планування і подальшої адаптації [1], [21], [22].

Типовий конвеєр обробки голосового доручення складається з кількох етапів. Спочатку аудіосигнал розпізнається модулем *speech-to-text*. Потім текст очищується від зайвих пауз, повторів, звертань і службових слів. Далі виконується визначення наміру, виділення сутностей, аналіз часових виразів, оцінка неповноти даних і формування структури нагадування. Після цього система може створити адаптований текст для адресата, вибрати формат подання і задати правила повторення [21], [23].

Розпізнавання мовлення є першим потенційним джерелом помилок. У побутових умовах можуть бути шум, переривання, нечітка дикція, дитячий голос, українсько-російські змішані фрази або власні імена, яких немає у словнику моделі. Тому система не повинна безумовно довіряти розпізаному тексту. Вона має показувати опікуну результат розпізнавання і дозволяти швидке редагування. Особливо це важливо для критичних нагадувань, пов'язаних зі здоров'ям або безпекою. Визначення наміру дає змогу зрозуміти, яку дію виконує користувач: створює нагадування, змінює його, скасовує, просить повторити, перевіряє журнал або запитує про статус. Для першого етапу системи достатньо зосередитися на намірі створення нагадування, але архітектура повинна залишати можливість розширення. У майбутньому користувач може сказати: «перенеси

нагадування на завтра», «покажи невиконані задачі» або «скасуй повторення для бабусі». Виділення сутностей у задачі нагадувань включає пошук адресата, дії, об'єкта дії, часу, дати, періодичності, важливості, способу підтвердження і додаткових умов. Наприклад, у фразі «Нагадай Сашку щодня о дев'ятій зробити зарядку і натиснути готово» адресатом є Сашко, дією – зробити зарядку, часом – дев'ята година, періодичністю – щодня, способом підтвердження – натискання кнопки. Така структура дає змогу системі не просто зберегти текст, а керувати процесом виконання. Аналіз часових виразів є окремою складною задачею. У реальному мовленні користувачі рідко формулюють час у строгому форматі. Вони кажуть «увечері», «після школи», «перед сном», «через десять хвилин», «коли повернеться», «кожного понеділка». Частина таких виразів можна обробити правилами, але для контекстних формулювань потрібні профілі користувача або уточнювальні запитання. Наприклад, «після школи» може означати різний час для різних дітей і різних днів тижня.

Методи, засновані на правилах, мають перевагу у передбачуваності. Регулярні вирази, словники і шаблони добре працюють із датами, годинами, повтореннями та типовими словами підтвердження. Їхній недолік очевидний: природна мова значно різноманітніша за набір шаблонів, тому кожен нову форму висловлення потрібно або передбачити, або обробити уточненням [21].

Методи машинного навчання дозволяють будувати моделі класифікації намірів і розпізнавання сутностей на основі прикладів. Для цього потрібен корпус фраз із розміченими адресатами, діями, часом та іншими полями. Перевагою є здатність узагальнювати на нові формулювання, недоліком – потреба в даних, перевірці якості та регулярному донавчанні. Для бакалаврської роботи доцільно комбінувати правила і мовні моделі, щоб отримати прийнятну якість без створення великого спеціального датасету [21], [22].

Великі мовні моделі відкривають додаткові можливості для персоналізації контенту. Вони можуть переформувувати повідомлення простішою мовою, розкласти доручення на кроки, змінювати тон, пояснювати незрозумілі слова і формувати альтернативні варіанти для різних профілів. Наприклад, одна команда опікуна може бути перетворена на коротке дитяче повідомлення, нейтральне повідомлення для стандартного користувача і збільшений текст із голосовим дублюванням для літньої людини [2].

Застосування LLM у системі нагадувань повинно бути обмеженим і контрольованим. Модель добре працює з мовою, але може додавати зайві деталі, пропускати важливу умову або створювати правдоподібний, але неправильний зміст. Тому доцільно розділити функції. Формальна структура нагадування має визначатися правилами, перевіркою і явними полями, а мовна модель може допомагати зі спрощенням, стилем і поясненням. Критичні параметри – час, адресат, дія і підтвердження повинні бути показані опікуну перед збереженням [2], [19].

Персоналізація контенту може виконуватися на кількох рівнях.

Перший рівень – лексичний. Заміна складних слів на прості, скорочення речень, уникнення абстрактних формулювань.

Другий рівень – структурний. Поділ дії на послідовні кроки.

Третій рівень – візуальний. Добір піктограми, виділення ключового часу, збільшений шрифт.

Четвертий рівень – поведінковий. Повтори, додаткове підтвердження, пропозиція звернутися по допомогу. Саме поєднання рівнів робить нагадування справді персоналізованим.

Для дитини модель персоналізації має зберігати доброзичливий тон, але уникати зайвої гри там, де потрібна точність. Повідомлення може бути коротким і мотивувальним: «Час зробити математику. Відкрий зошит. Розв’яжи завдання. Зроби фото». Для літньої людини тон має бути спокійним і поважним: «Зараз час прийняти ліки. Перевір назву на упаковці!

Натисни «виконано», коли завершиш». Обидва повідомлення походять від одного доручення, але мають різну форму. У задачі персоналізації важливо не ототожнювати спрощення з втратою сенсу. Якщо опікун додав умову «показати зошит» або «не виходити з дому до дзвінка», система не має прибирати її заради короткості. Тому спрощення повинно бути семантично контрольованим: адаптований текст має містити всі обов'язкові поля і не суперечити вихідному дорученню. Одним із практичних способів є формування повідомлення з уже виділених полів, а не довільне переписування всього тексту. Великі мовні моделі (LLM) також можуть допомагати у виявленні неповних або неоднозначних команд. Якщо у фразі немає часу, адресата або способу підтвердження, система може сформувати уточнювальне запитання: «Коли нагадати?», «Кому саме адресоване доручення?», «Чи потрібно фото-підтвердження?». Проте такі запитання мають бути короткими і не перетворювати процес створення нагадування на довгий діалог. Для типових випадків можна використовувати налаштування за замовчуванням. Окремою задачею є обробка української мови. Вона має відмінювання імен, зміну форм слів, вільний порядок речення і багато розмовних варіантів. Наприклад, «бабусі», «бабусю», «для бабусі» можуть позначати того самого адресата. Імена користувачів також можуть мати зменшувальні форми. Тому система повинна зберігати словник родинних ролей і імен, а також використовувати порівняння з профілями учасників родини. Це зменшує ризик неправильного призначення доручення [21].

Безпека та приватність є обов'язковими вимогами для мовних модулів. Родинні нагадування можуть містити персональні дані, інформацію про здоров'я, розклад, місцезнаходження або звички користувачів. Якщо застосовується зовнішня мовна модель, потрібно мінімізувати передавання чутливих даних, використовувати знеособлення там, де можливо, і отримувати згоду користувачів. Для демонстраційного прототипу доцільно передбачити тестовий режим або локальні правила,

щоб основна логіка могла працювати без передавання даних третім сторонам [19], [20].

Якість NLP-модуля можна оцінювати за кількома критеріями: точність визначення адресата, правильність часу, повнота виділення дії, кількість потрібних уточнень, частка помилкових нагадувань, які вчасно виявлено на етапі підтвердження, і суб'єктивна зрозумілість адаптованого тексту. Для інклюзивної системи важливо оцінювати не тільки технічну точність, а й користувацький результат: чи зрозумів адресат, що потрібно зробити, і чи зміг виконати дію без додаткових пояснень. Архітектурно доцільно розділити модулі розпізнавання мовлення, парсингу доручення, спрощення контенту і генерації повідомлення. Такий поділ полегшує тестування і заміну компонентів.

Наприклад, на ранньому етапі Speech-to-Text може бути демонстраційним, парсинг-правильним, а спрощення – шаблонним. Надалі окремі компоненти можна замінювати на більш складні моделі, не змінюючи загальної логіки системи.

Отже, методи NLP та LLM забезпечують перехід від звичайного нагадувача до інтелектуальної системи родинної комунікації. Вони допомагають працювати з природною мовою, структурувати доручення і персоналізувати повідомлення. Водночас ці методи мають ризики, тому повинні застосовуватися разом із правилами, валідацією, підтвердженням опікуна і вимогами доступності.

Таблиця показує, що кожен метод дає практичну користь, але водночас створює окрему зону ризику. Тому доцільно використовувати багаторівневу перевірку: спочатку система виділяє формальні поля, потім формує чернетку нагадування, показує її опікуну і лише після цього створює повідомлення для адресата.

Особливо важливо відокремити адаптацію форми від зміни змісту. Мовна модель може зробити текст простішим, дружнішим або коротшим,

але не повинна самостійно змінювати час приймання ліків, пріоритет безпеки або адресата (таблиця 1.3).

Таблиця 1.3 – Роль методів NLP у системі персоналізованих нагадувань

Метод або модуль	Призначення	Ризик або обмеження
Speech-to-text	Перетворення голосової команди на текст	Помилки через шум, нечітку вимову, власні імена
Визначення наміру	Розуміння типу дії користувача	Помилкова класифікація при коротких або неоднозначних фразах
Виділення сутностей	Пошук адресата, часу, дії, повторів і підтвердження	Неповні дані або неправильний адресат
Правиловий парсинг	Стабільна обробка типових фраз і часових виразів	Обмежена гнучкість і потреба підтримувати словники
LLM-спрощення	Адаптація тексту до профілю адресата	Можлива втрата умов або додавання зайвої інформації

Для цього формальні поля зберігаються окремо від текстового опису. Адаптоване повідомлення будується навколо цих полів, і кожен варіант має містити однаковий змістовий мінімум. Для подальшої реалізації прототипу важливо передбачити режим тестування, у якому можна порівнювати вихідну команду, розпізнаний текст, структуру нагадування і адаптовані повідомлення. Такий журнал допоможе виявляти типові помилки, покращувати правила української мови, створювати приклади для навчання

моделі та обґрунтовувати якість системи в експериментальній частині кваліфікаційної роботи.

Отже, NLP і LLM у цій темі не є самоціллю. Їхнє призначення – підтримати доступну родинну взаємодію, зменшити кількість ручних дій і підвищити зрозумілість доручень. Реальна ефективність залежить не тільки від точності моделі, а й від того, наскільки весь інтерфейс відповідає вимогам доступності.

1.4 Аналіз вимог до доступності інтерфейсів

Доступність інтерфейсу є обов’язковою умовою для системи, що позиціонується як інклюзивна. Якщо система вміє створювати персоналізовані нагадування, але сама є складною, непередбачуваною або незручною для вразливих користувачів, вона не виконує своєї основної функції. Тому вимоги доступності потрібно враховувати не після реалізації, а на етапі постановки задачі, опису сценаріїв і проектування архітектури. Міжнародною основою для веб-доступності є рекомендації WCAG, які групують вимоги навколо принципів сприйманості, керованості, зрозумілості й надійності. Для системи нагадувань ці принципи можна інтерпретувати практично. Сприйманість означає, що користувач має побачити або почути нагадування. Керованість – що він може натиснути потрібну кнопку, не виконуючи складних жестів. Зрозумілість – що текст, порядок дій і результат натискання очевидні. Надійність – що інтерфейс коректно працює з різними пристроями і допоміжними технологіями [14], [24].

Сприйманість у контексті персоналізованих нагадувань передбачає достатній контраст, читабельний шрифт, підтримку масштабування, голосове дублювання, чіткі піктограми і виділення ключової інформації. Час нагадування, основна дія і кнопка підтвердження повинні бути помітними одразу. Для літнього користувача або людини зі зниженим зором

важливо уникати дрібного тексту, світло-сірого шрифту, складних фонів і елементів, які втрачають зміст без кольору [14].

Керованість означає, що всі основні дії мають бути доступні без складних рухів. На мобільному пристрої це стосується розміру натискних елементів, відстані між кнопками, підтримки екранних читачів, зрозумілого фокусу і можливості повернутися назад. Для адресата достатньо кількох дій: прослухати, підтвердити, відкласти, попросити допомогу. Чим менше другорядних кнопок на екрані нагадування, тим нижчий ризик помилки. Зрозумілість є центральною вимогою для когнітивної доступності. Інтерфейс повинен використовувати прості слова, передбачувані назви кнопок, однакові шаблони екранів і явний зворотний зв'язок. Якщо користувач натиснув «виконано», система має показати, що дія зафіксована. Якщо потрібно фото, інтерфейс має пояснити, що саме сфотографувати. Якщо нагадування відкладено, користувач повинен знати, коли воно повториться. Невизначеність у таких сценаріях створює тривогу і знижує довіру. Надійність передбачає сумісність із різними пристроями, операційними системами, масштабуванням шрифту, темним режимом, екранними читачами і нестабільним інтернетом. Родинна система нагадувань не повинна залежати від ідеальних умов. Якщо мережа тимчасово недоступна, локальне сповіщення має все одно спрацювати або система повинна чесно показати стан синхронізації. Для критичних нагадувань важливо передбачити повтори і повідомлення опікуна про невиконання. Когнітивна доступність виходить за межі мінімальних технічних критеріїв. Вона передбачає підтримку пам'яті, уваги, розуміння і прийняття рішень. Для системи нагадувань це означає, що екран не повинен вимагати від користувача одночасно читати довгий текст, обирати між багатьма варіантами і пам'ятати попередні інструкції. Бажано показувати одну основну дію, одну ключову кнопку і короткий пояснювальний текст. Додаткова інформація має відкриватися лише за потреби [15], [17].

Персоналізація доступності повинна бути частиною профілю користувача. Опікун або сам користувач може встановити розмір шрифту, потребу в озвученні, режим контрасту, кількість кроків, формат підтвердження, частоту повторів і стиль повідомлень. Водночас профіль не має бути надто складним. Початково доцільно запропонувати кілька режимів: «дитина», «літня людина», «стандартний користувач», а детальні параметри залишити для розширеного налаштування [13], [18].

У дитячому режимі інтерфейс може використовувати піктограми, короткі речення, дружній тон, поступове відкриття кроків і позитивне підкріплення після виконання. Проте потрібно уникати надмірної гейміфікації для важливих задач. Якщо нагадування стосується безпеки або здоров'я, грайливий тон може бути недоречним. Тому система має враховувати не лише профіль користувача, а й тип доручення та його критичність.

У режимі для літньої людини основними вимогами є великий текст, високий контраст, озвучення, мінімум меню, зрозумілі кнопки і відсутність прихованих жестів. Повідомлення має залишатися на екрані достатньо довго. Якщо користувач випадково закрит нагадування, він повинен легко повернутися до нього. Для підтвердження бажано використовувати одну велику кнопку, а для складніших ситуацій – кнопку «потрібна допомога», яка сповіщає опікуна.

Для опікуна інтерфейс може бути складнішим, але також має бути доступним. Швидке створення доручення, перевірка розпізнаного тексту, редагування полів, перегляд журналу і налаштування профілів мають виконуватися без зайвих переходів. Особливо важливо показувати різницю між вихідним дорученням і адаптованим повідомленням. Опікун повинен бачити, що система не втратила ключову умову і правильно визначила адресата.

Доступність також включає зрозумілу обробку помилок. Якщо команда не розпізнана, не можна просто показати технічне повідомлення.

Потрібно пояснити, що саме не вдалося, і запропонувати дію: повторити запис, ввести текст вручну, вибрати адресата або уточнити час. Повідомлення про помилки повинні бути короткими і не звинувачувати користувача. Наприклад: «Не вдалося визначити час. Оберіть час нагадування» є кращим варіантом, ніж «помилка парсингу дати». Вимоги до доступності мають відображатися не тільки в інтерфейсі, а й у моделі даних. Якщо система зберігає профіль доступності, вона повинна враховувати його під час створення, доставки і відображення нагадування. Якщо адресат змінив параметри шрифту або озвучення, нові нагадування повинні автоматично використовувати ці налаштування. Це вимагає відокремлення змісту доручення від його презентаційної форми. Важливим питанням є безпека приватних даних. Профілі доступності можуть опосередковано вказувати на вік, стан здоров'я або інші чутливі характеристики. Журнал виконання може містити інформацію про ліки, поведінку, навчання або розклад. Тому доступ до цих даних має бути рольовим і мінімально необхідним. Спостерігач не повинен бачити більше, ніж потрібно, а адресат не повинен мати можливість змінити критичні налаштування без дозволу опікуна [19], [20].

Доступність повинна оцінюватися як технічно, так і користувацьки. Технічна перевірка може включати контраст, розмір елементів, структуру заголовків, підтримку екранного читача, підписи кнопок і коректність станів фокусу. Користувацька перевірка має з'ясувати, чи розуміє адресат нагадування, чи може виконати дію, чи не плутається в кнопках і чи не відчуває перевантаження. Для інклюзивної системи саме користувацький результат є визначальним [13], [14].

Окрему увагу потрібно приділити локалізації. Український інтерфейс має використовувати природні й короткі формулювання. Кнопки «Виконано», «Прослухати», «Відкласти», «Потрібна допомога» є зрозумілішими за технічні назви. Для дитячого режиму можуть використовуватися м'які формулювання, але основні дії мають залишатися

однозначними. Локалізація також стосується формату дат і часу, правильного відмінювання імен і родинних ролей. Для забезпечення доступності на практиці доцільно сформувати набір дизайн-правил.

Наприклад: не більше однієї головної дії на екрані нагадування; мінімум 14–16 пунктів для стандартного тексту і більший розмір для спеціальних режимів; високий контраст; кнопки з текстовими підписами; можливість озвучення; повторювана структура екранів; явне підтвердження результату; відсутність критичних дій без перевірки. Такі правила можуть бути використані під час розроблення прототипу.

Аналіз вимог до доступності показує, що інтелектуальність системи повинна проявлятися не лише у мовних моделях. Справді корисна система має адаптувати інтерфейс, поведінку і спосіб зворотного зв'язку. Якщо користувач потребує голосу, система озвучує повідомлення. Якщо користувач легко губиться в меню, система показує один крок. Якщо доручення критичне, система вимагає підтвердження і повідомляє опікуна. Такий підхід поєднує технічні стандарти і реальні потреби родини.

Отже, вимоги WCAG і когнітивної доступності формують четверту групу теоретичних положень для майбутньої системи. Вони визначають, як саме мають виглядати й поводитися нагадування, щоб ними могли користуватися різні люди. Зв'язок між принципами доступності та функціями системи подано нижче.

Таблиця підсумовує, що доступність не є декоративним налаштуванням наприкінці розробки. Вона впливає на структуру екранів, модель даних, сценарії повторення, типи підтвердження і правила адаптації повідомлень. Якщо ці вимоги не врахувати від початку, система може залишитися формально інтелектуальною, але фактично незручною для адресата.

Кожен сценарій має бути протестований з позиції різних профілів доступності (таблиця 1.4).

Таблиця 1.4 – Відповідність принципів доступності функціям системи

Принцип	Зміст принципу	Реалізація у системі нагадувань
Сприйманість	Інформація має бути доступною для зору, слуху або альтернативного сприйняття	Великий шрифт, контраст, озвучення, піктограми, виділення часу
Керованість	Користувач має мати змогу виконати дію без складних жестів	Великі кнопки, просте підтвердження, мінімум переходів, підтримка екранного читача
Зрозумілість	Інтерфейс і текст мають бути передбачуваними та ясними	Короткі фрази, покроковість, пояснення помилок, однакова структура екранів
Надійність	Система має працювати стабільно з різними пристроями і допоміжними технологіями	Синхронізація, локальні сповіщення, коректні стани, рольовий доступ
Когнітивна доступність	Потрібно зменшувати навантаження на пам'ять і увагу	Один крок на екрані, персональні профілі, повтори, кнопка допомоги

Такий підхід допомагає уникнути ситуації, коли інтерфейс виглядає зрозумілим для розробника, але не працює для реальної дитини, літньої людини або користувача з когнітивними труднощами.

2 ОБҐРУНТУВАННЯ МЕТОДІВ ТА ТЕХНОЛОГІЙ РОЗРОБКИ

Другий розділ присвячено вибору методів і технологій, на яких доцільно будувати інтелектуальну систему персоналізованих нагадувань. Після аналізу предметної галузі вже зрозуміло, що розроблюване рішення не може бути звичайним списком справ. Воно має враховувати ролі членів родини, доступність інтерфейсу, природномовні доручення, повтори, підтвердження виконання і приватність даних. Тому в цьому розділі технології розглядаються не окремо від задачі, а через практичні сценарії використання.

Якщо певний інструмент обрано, то він повинен пояснюватися конкретною потребою системи: швидким створенням нагадування, коректною доставкою сповіщення, адаптацією тексту або безпечним зберіганням родинних даних.

Окремо враховано вимоги доступності. Для системи, яка заявлена як інклюзивна, цього недостатньо залишити на фінальний етап. Вимоги WCAG і рекомендації щодо когнітивної доступності впливають на структуру екранів, довжину повідомлень, роботу кнопок, колірні контрасти, озвучення та спосіб пояснення помилок [25], [26].

У подальших підрозділах обґрунтовано функціональну модель, методи обробки природної мови, підхід до персоналізації, клієнтські та серверні технології, засоби захисту даних і критерії тестування. Таке подання дозволяє пов'язати програмну реалізацію з реальними потребами користувачів, а не просто перелічити популярні фреймворки.

2.1 Загальний підхід до розробки системи

Для розробки обрано поступове прототипування з орієнтацією на користувацькі сценарії. Цей підхід є більш придатним для теми роботи, ніж спроба одразу створити велику універсальну систему. У родинній

комунікації багато деталей стають зрозумілими лише після опису конкретних ситуацій: хто створює нагадування, кому воно адресоване, що саме потрібно підтвердити і що робити, якщо адресат не відреагував.

Першою вимогою є доступність за замовчуванням. Інтерфейс не повинен спочатку створювати перешкоди, а потім компенсувати їх окремим режимом. Наприклад, кнопка підтвердження має бути помітною відразу, а не тільки після додаткового налаштування. Те саме стосується тексту нагадування: він повинен бути коротким і зрозумілим, навіть якщо користувач не відкриває розширений опис.

Друга вимога стосується прозорості інтелектуального модуля. Система може допомагати опікуну перетворювати звичайну фразу на нагадування, але остаточні поля мають залишатися перевірюваними. Якщо користувач сказав: «Нагадай бабусі ввечері випити ліки», система не повинна непомітно вибирати конкретний час без підтвердження. Краще показати коротке уточнення, ніж створити формально правильне, але небезпечне нагадування.

Ще одна важлива ідея полягає у розділенні змісту доручення і форми його подання. Зміст зберігається як структура: дія, час, адресат, повторюваність, критичність і спосіб підтвердження. Форма подання може відрізнятися: для дитини це кроки й піктограма, для літньої людини - великий текст і озвучення, для опікуна - повний запис у журналі.

Такий підхід також спрощує розробку. Модулі авторизації, профілів, нагадувань, обробки природної мови та сповіщень можна реалізовувати окремо. Якщо на першому етапі NLP працює за правилами, у майбутньому його можна замінити або доповнити мовною моделлю без переписування всієї системи.

Зв'язок між користувацькими потребами і проєктними рішеннями наведено нижче (таблиця 2.1).

Таблиця 2.1 – Зв'язок потреб користувачів із проєктними рішеннями

Потреба або проблема	Проєктне рішення	Очікуваний результат
Користувач не розуміє довгу інструкцію	Спрощення тексту та поділ доручення на короткі кроки	Зменшення когнітивного навантаження
Опікун створює багато схожих доручень	Шаблони, голосове введення та повторювані сценарії	Скорочення часу створення нагадувань
Адресат може пропустити важливу дію	Повтори, статуси виконання і повідомлення опікуна	Кращий контроль критичних доручень
Користувачі мають різні можливості сприйняття	Профілі доступності і різні формати подання	Індивідуальне відображення одного доручення
Потрібно уникнути випадкових змін	Рольова модель доступу	Захист критичних налаштувань

Наведена таблиця показує, що в основі системи стоїть не набір екранів, а набір ситуацій. Це важливо для кваліфікаційної роботи, оскільки дозволяє пояснити технічні рішення через практичну користь. Наприклад, рольова модель потрібна не для ускладнення архітектури, а для того, щоб дитина випадково не змінила медичне нагадування, створене опікуном.

Отже, загальний підхід можна визначити як користувацько-орієнтований і модульний. Він залишає систему достатньо простою для реалізації прототипу, але не закриває можливості для подальшого розвитку.

2.2 Функціональна модель та ролі користувачів

Функціональна модель повинна відповідати тому, як реально організована взаємодія в родині. У звичайному нагадувачі людина часто

створює задачу для себе. У цій системі доручення може створювати один користувач, отримувати інший, а контролювати виконання третій. Через це просте поле «користувач» не описує всієї логіки.

У системі доцільно виділити чотири ролі: адміністратор родинного простору, опікун, адресат і спостерігач. Адміністратор керує складом групи та правами. Опікун створює і редагує доручення. Адресат отримує повідомлення та підтверджує виконання. Спостерігач може бачити окремі статуси без права втручатися в критичні налаштування. Такий поділ не є формальністю. У різних ролинах ролі можуть перетинатися. Наприклад, доросла дитина може бути адресатом побутових нагадувань, але водночас допомагати літній людині як опікун. Тому права доступу краще прив'язувати не тільки до ролі, а й до типу нагадування. Життєвий цикл нагадування починається зі створення команди. Після цього система визначає структуру, перевіряє повноту даних, формує адаптоване повідомлення, доставляє його адресату, очікує реакцію і записує результат у журнал. Якщо даних не вистачає, система має повернутися до етапу уточнення, а не створювати неповне доручення.

Послідовність основних станів доручення подано нижче, оскільки така схема зручна для подальшого проєктування API і тестування окремих переходів.

У цій моделі найважливішими є етапи перевірки та підтвердження. Саме вони відрізняють систему від звичайного календаря. Якщо нагадування створене для іншої людини, його потрібно не тільки запланувати, а й переконатися, що воно зрозуміле і може бути виконане. Функціональна модель також має підтримувати різні типи підтвердження. Для нескладної побутової задачі достатньо кнопки «Виконано». Для навчального завдання може знадобитися фото. Для нагадування, пов'язаного зі здоров'ям, варто додати повідомлення опікуна у разі пропуску (таблиця 2.2).

Таблиця 2.2 – Життєвий цикл персоналізованого нагадування

Етап	Зміст етапу	Відповідальний модуль
Створення	Опікун вводить текстову або голосову команду	Клієнтський застосунок
Структурування	Система визначає адресата, дію, час, повтори та підтвердження	NLP-модуль
Перевірка	Опікун бачить структуру і за потреби уточнює дані	Вебкабінет або мобільний інтерфейс
Адаптація	Формується повідомлення з урахуванням профілю адресата	Модуль персоналізації
Доставка	Нагадування надходить у потрібний час	Сервер і модуль сповіщень
Підтвердження	Адресат натискає кнопку, додає коментар або фото	Клієнтський застосунок
Журналювання	Результат зберігається для перегляда опікуном	Серверна частина

Отже, центральною сутністю системи є не подія календаря, а доручення. Воно має автора, адресата, зміст, час, правила повторення, адаптований текст, статус і журнал змін.

2.3 Обробка природномовних доручень

Обробка природної мови є однією з ключових частин системи, але її не слід перебільшувати. Користувачеві важливо не те, наскільки складна модель використовується всередині, а чи правильно система зрозуміла звичайну фразу. У родинному середовищі команда часто звучить неформально: «нагадай Марку після школи зробити математику» або «бабусі ввечері таблетки».

Для такого сценарію доцільно застосувати гібридний підхід. Частину завдань краще виконувати правилами: розпізнавання дат, часу, повторів, типових маркерів на зразок «щодня», «завтра», «через годину». Такі елементи повинні бути передбачуваними. Водночас живі формулювання, неповні фрази і переформулювання зручніше обробляти мовною моделлю.

На першому етапі команда перетворюється на текст. Якщо користувач говорить голосом, потрібен модуль розпізнавання мовлення. Далі текст очищується від пауз, повторів і зайвих звертань. Після цього система визначає сутності: адресата, дію, об'єкт дії, час, повторюваність, важливість і спосіб підтвердження.

Великі мовні моделі можуть бути корисними для роботи з неоднозначними фразами та для формування структурованої відповіді. Зокрема, сучасні API підтримують повернення результату у наперед заданій структурі, що зменшує ризик отримати довільний текст замість потрібних полів [35]. Але це не скасовує перевірку. Модель може помилитися, тому сервер має валідувати результат перед збереженням.

Основні поля структури нагадування наведено нижче. Така структура не прив'язана до конкретної бази даних, але показує, які дані потрібно отримати після обробки природної мови.

Для української мови потрібно окремо врахувати відмінювання імен та родинних ролей. «Бабусі», «бабусю» і «для бабусі» можуть означати одного адресата. Так само можуть використовуватися домашні скорочення і зменшувальні імена. Тому в системі потрібен словник учасників родини та їхніх варіантів звертання. Помилки слід обробляти як нормальну частину взаємодії. Якщо система не впевнена у часі, вона повинна спитати: «Коли нагадати?». Якщо не впевнена в адресаті - запропонувати вибір. Такий діалог має бути коротким, без технічних повідомлень на кшталт «помилка парсингу» (таблиця 2.3).

Таблиця 2.3 – Основні поля структури нагадування

Поле	Призначення	Приклад значення
author_id	Користувач, який створив доручення	Опікун
recipient_id	Адресат нагадування	Дитина або літня людина
action	Основна дія, яку потрібно виконати	зробити домашнє завдання
time_rule	Час або умова спрацювання	після школи, 18:00
repeat_rule	Правила повторення	щодня, щопонеділка
confirmation_type	Спосіб підтвердження	кнопка, фото, коментар
importance	Рівень критичності	звичайне, важливе, критичне
adapted_message	Текст для адресата	Відкрий зошит і виконай завдання

Отже, NLP-модуль у цій роботі виконує прикладну функцію: він допомагає перетворити живе мовлення на перевірювану структуру. Саме структура, а не згенерований текст, має бути основою для збереження нагадування.

2.4 Персоналізація та адаптація повідомлень

Персоналізація в межах цієї роботи розуміється ширше, ніж вибір теми оформлення або часу сповіщення. Вона стосується того, як людина сприймає доручення. Один і той самий зміст може бути поданий по-різному: короткими кроками, великим текстом, голосом, піктограмою або повним описом для опікуна.

Основою персоналізації є профіль доступності. У ньому можуть зберігатися бажаний розмір шрифту, потреба в озвученні, допустима кількість кроків, формат підтвердження, мова повідомлень і частота повторів. Водночас профіль не має перетворюватися на складну анкету. Для прототипу достатньо кількох початкових режимів, які потім можна налаштовувати детальніше. Адаптація повинна виконуватися після структурування доручення. Це важливо. Якщо спочатку спростити текст, можна випадково втратити умову або час. Краще зберігати формальні поля окремо, а повідомлення для адресата формувати вже на їхній основі.

Приклади правил адаптації подано нижче. Вони не є остаточним дизайном інтерфейсу, але допомагають зафіксувати логіку, яку потрібно реалізувати у програмній частині (таблиця 2.4).

Таблиця 2.4 – Правила адаптації повідомлень для різних профілів

Профіль	Особливості подання	Обмеження
Дитина	Короткі речення, кроки, піктограма, позитивний тон	Не використовувати гру для критичних задач
Літня людина	Великий шрифт, озвучення, контраст, прості кнопки	Не перевантажувати екран додатковими діями
Користувач із когнітивними труднощами	Один крок на екрані, повторювана структура, короткі фрази	Не змінювати зміст заради спрощення
Опікун	Повний текст, статуси, журнал виконання	Не показувати зайві персональні дані без потреби
Стандартний користувач	Звичайне текстове нагадування з можливістю деталей	Залишити доступ до налаштувань

Для дитини доречним може бути повідомлення: «Час зібрати рюкзак. Поклади зошит. Перевір пенал. Натисни “готово”». Для літньої людини той самий рівень простоти має звучати інакше: спокійно, без дитячого тону, з чіткою кнопкою підтвердження. У персоналізації є небезпека надмірного спрощення. Якщо опікун додав умову «надіслати фото зошита», система не має прибрати її тільки тому, що текст став довшим. Форма може змінюватися, але змістові поля мають залишатися незмінними.

Для контролю якості бажано зберігати три варіанти: вихідну команду, структуроване доручення і адаптоване повідомлення. Це дасть змогу перевірити, чи справді система лише змінила форму подання, а не змінила зміст доручення.

2.5 Технології клієнтської частини

Клієнтська частина системи має бути поділена на два основні інтерфейси: мобільний застосунок для адресата і вебкабінет для опікуна. Один універсальний інтерфейс у цьому випадку був би менш зручним. Адресату потрібна проста дія тут і зараз, а опікуну - створення, редагування, перегляд журналу та налаштування профілів. Для мобільного застосунку доцільно використати Flutter. Він дозволяє розробляти застосунки для різних платформ з однієї кодової бази, а також має окремі засоби для роботи з доступністю інтерфейсу [27]. Для кваліфікаційного прототипу це важливо: не потрібно підтримувати два різні мобільні проєкти, але можна показати логіку застосунку, близьку до реального використання. Мобільний екран адресата має бути дуже стриманим. Найближче нагадування, час, короткий текст, кнопка озвучення, кнопка підтвердження і, за потреби, кнопка «потрібна допомога». Усе інше краще винести в додаткові екрани або взагалі не показувати адресату. Для вебкабінету опікуна доцільно використати React у поєднанні з TypeScript. React зручний для компонентного підходу, а TypeScript допомагає контролювати структуру

даних під час розробки інтерфейсу [28]. Це корисно, коли в системі є повторювані елементи: картки нагадувань, статуси, профілі, фільтри, журнал подій. Порівняння двох клієнтських інтерфейсів наведено нижче. Воно показує, чому система потребує різних режимів взаємодії, а не одного перевантаженого екрана для всіх користувачів (таблиця 2.5).

Таблиця 2.5 – Призначення клієнтських інтерфейсів системи

Інтерфейс	Основний користувач	Ключові функції
Мобільний застосунок	Адресат нагадування	Отримання повідомлень, озвучення, підтвердження, відкладання
Мобільний застосунок	Опікун у швидкому сценарії	Створення простого доручення голосом або текстом
Вебкабінет	Опікун або адміністратор	Профілі, ролі, журнал виконання, налаштування доступності
Вебкабінет	Спостерігач	Перегляд дозволених статусів без зміни критичних даних

Таке розділення також полегшує тестування. Мобільний застосунок перевіряється через сценарії адресата: чи видно нагадування, чи зрозумілий текст, чи працює підтвердження. Вебкабінет перевіряється через сценарії керування: створення доручення, зміна профілю, перегляд невиконаних задач.

З погляду оформлення майбутнього прототипу важливо не копіювати складність звичайних органайзерів. Для цієї теми кращим є спокійний інтерфейс із невеликою кількістю дій. Менше елементів на екрані у цьому випадку означає не біднішу функціональність, а кращу доступність.

2.6 Серверна частина, база даних і сповіщення

Серверна частина має об'єднувати мобільний застосунок і вебкабінет у єдину систему. Вона відповідає за авторизацію, профілі користувачів, збереження нагадувань, обробку команд, планування сповіщень і журнал виконання. Це та частина, де зміст доручення зберігається незалежно від того, як саме його бачить адресат.

Для backend-частини доцільно використати FastAPI. Цей фреймворк працює з Python, підтримує типізацію і добре підходить для побудови REST API [29]. Вибір Python додатково зручний через те, що багато інструментів для NLP, обробки тексту і взаємодії з мовними моделями мають зрілу підтримку саме в цій екосистемі.

API системи варто будувати навколо кількох груп ресурсів: користувачі, родинні простори, профілі доступності, нагадування, підтвердження, журнал подій і NLP-обробка. Такий поділ робить код зрозумілішим і дозволяє обмежувати доступ до окремих операцій. Для збереження основних даних обрано PostgreSQL. Реляційна база добре підходить для системи з чіткими зв'язками: користувачі належать до родинних просторів, нагадування мають авторів і адресатів, підтвердження пов'язані з конкретними подіями. Для гнучких параметрів профілю можна використовувати JSON-поля, які підтримуються PostgreSQL [30]. Механізм нагадувань не повинен залежати від того, чи відкритий у цей момент клієнтський застосунок. Тому потрібні фонові задачі. Redis може використовуватися як черга або допоміжний механізм для обробки відкладених подій [31]. Для доставки push-сповіщень до мобільних і вебклієнтів доцільно використати Firebase Cloud Messaging [32].

Розподіл серверних технологій подано нижче. У таблиці показано не лише назви інструментів, а й причини їхнього вибору для конкретної системи (таблиця 2.6).

Таблиця 2.6 – Обґрунтування технологій серверної частини

Технологія	Роль у системі	Причина вибору
FastAPI	Побудова REST API та обробка запитів клієнтів	Python, типізація, зручність інтеграції з NLP-компонентами
PostgreSQL	Збереження користувачів, профілів, нагадувань і журналу	Надійність, реляційні зв'язки, підтримка JSON-полів
Redis	Черга фонових задач і допоміжне кешування	Підходить для відкладеної обробки нагадувань
Firebase Cloud Messaging	Доставка push-сповіщень	Підтримка мобільних і вебклієнтів
Docker	Підготовка демонстраційного середовища	Повторюваний запуск компонентів і залежностей

Docker доцільно використати для підготовки демонстраційного середовища. Контейнер містить необхідні залежності й дозволяє запускати застосунок однаково на різних комп'ютерах [36]. Для кваліфікаційної роботи це зручно, бо зменшує ризик ситуації, коли програма працює тільки в локальному середовищі розробника. Мінімальна серверна архітектура може складатися з API-сервера, бази даних, черги задач і модуля сповіщень. NLP-модуль на першому етапі можна реалізувати всередині backend-частини, а надалі винести окремо, якщо зросте навантаження або з'явиться потреба незалежного масштабування. Обраний набір технологій є достатнім для прототипу і не виглядає надмірним. Він покриває основні потреби системи: зберігання даних, роботу з API, фонову доставку нагадувань, push-сповіщення і стабільний запуск.

2.7 Захист даних і контроль доступу

Система персоналізованих нагадувань працює з чутливими побутовими даними. У дорученнях можуть бути згадки про здоров'я, навчання, розклад, місцезнаходження, звички або поведінку користувачів. Тому безпеку не можна додавати після розробки як окремий шар. Вона має впливати на модель ролей, формат сповіщень і роботу з журналом виконання.

Першим елементом є автентифікація та авторизація. Для реальної системи доцільно орієнтуватися на підходи OAuth 2.0, де доступ до ресурсів надається через обмежені токени [33]. У кваліфікаційному прототипі можна реалізувати спрощену модель входу, але вона повинна відображати той самий принцип: користувач бачить лише те, що дозволено його роллю.

Другий елемент – мінімізація даних. У push-сповіщенні не завжди потрібно передавати повний текст доручення. Іноді достатньо повідомлення «У вас є нове нагадування», а деталі відкриваються тільки після входу в застосунок. Такий підхід зменшує ризик випадкового розкриття приватної інформації на екрані телефону.

Для перевірки технічних рішень можна орієнтуватися на OWASP ASVS. Цей стандарт містить вимоги до автентифікації, керування сесіями, контролю доступу, валідації даних і журналювання [34]. У межах кваліфікаційної роботи він використовується як орієнтир для формування базових вимог безпеки. Основні вимоги до захисту даних у системі подано нижче. Вони пов'язані з конкретними механізмами, які можуть бути реалізовані у прототипі або закладені в архітектуру.

Окремо потрібно врахувати безпеку інтелектуального модуля. Відповідь мовної моделі не може автоматично вважатися правильною. Сервер має перевіряти обов'язкові поля, формат часу, наявність адресата і тип підтвердження. Для критичних нагадувань потрібен попередній перегляд для опікуна (таблиця 2.7).

Таблиця 2.7 – Вимоги до захисту даних у системі

Вимога	Реалізація в системі	Мета
Автентифікація	Вхід користувача, токени доступу, контроль активних сесій	Не допустити доступ сторонніх осіб
Авторизація	Рольова модель для опікуна, адресата, адміністратора і спостерігача	Обмежити дії відповідно до ролі
Мінімізація даних	Скорочений зміст push- сповіщень, приховані деталі без входу	Зменшити ризик розкриття приватної інформації
Журнал подій	Фіксація створення, зміни, доставки і підтвердження нагадувань	Підвищити прозорість без надмірного нагляду
Перевірка критичних дій	Додаткове підтвердження для важливих або медичних доручень	Запобігти небезпечним помилкам

Журнал подій також потребує обережності. Він потрібен для контролю і пояснення того, що сталося з дорученням, але не повинен перетворюватися на надмірний нагляд. Доступ до журналу має бути рольовим і обмеженим.

Таким чином, безпека в цій системі є частиною функціональної логіки. Вона впливає не тільки на вхід у систему, а й на те, які дані потрапляють у сповіщення, хто бачить статуси і коли потрібне додаткове підтвердження.

2.8 Оцінювання якості обраних рішень

Оцінювання якості системи має охоплювати не тільки технічні показники. Для звичайного застосунку достатньо перевірити стабільність API, швидкість відповіді і відсутність помилок. Для інклюзивної системи цього мало. Важливо зрозуміти, чи адресат справді побачив, почув і правильно зрозумів нагадування. Тестування доцільно поділити на кілька рівнів. На рівні модулів перевіряються окремі функції: парсинг часу, створення нагадування, зміна статусу, відправлення сповіщення. На рівні інтеграції перевіряється повний шлях від голосової або текстової команди до підтвердження виконання. Окремо треба перевірити сценарії з помилками, наприклад відсутній час або неоднозначний адресат. Доступність перевіряється не лише формально. Контраст, розмір кнопок і підтримка озвучення важливі, але головне питання простіше: чи може людина виконати дію без додаткового пояснення? Саме такий критерій краще відповідає темі роботи.

Критерії оцінювання якості наведено нижче. До них включено технічні, користувацькі та безпекові показники, оскільки система має працювати як єдине рішення, а не як набір окремих модулів.

Для прототипу достатньо підготувати набір типових сценаріїв: нагадування для дитини, нагадування для літньої людини, команда без точного часу, команда з фото-підтвердженням, критичне нагадування і пропущене виконання. Кожен сценарій має перевіряти не тільки створення запису в базі, а й поведінку інтерфейсу.

Окремо варто переглядати тексти адаптованих повідомлень. Якщо всі повідомлення звучать однаково, система виглядає формальною. Коротке побутове доручення, навчальна задача і медичне нагадування повинні мати різний тон, але однакову точність щодо ключових умов (таблиця 2.8).

Таблиця 2.8 – Критерії оцінювання якості розроблюваної системи

Група критеріїв	Що перевіряється	Приклад показника
Функціональність	Створення, редагування, доставка і підтвердження нагадувань	Частка успішно виконаних сценаріїв
NLP-обробка	Правильність визначення адресата, часу, дії та повторів	Кількість команд, що не потребували ручного виправлення
Доступність	Читабельність, простота навігації, озвучення, зрозумілі кнопки	Час виконання базової дії адресатом
Безпека	Контроль ролей, захист приватних даних, журнал подій	Відсутність доступу до заборонених дій
Стабільність	Робота API, фонових задач і сповіщень	Кількість збоїв під час тестових сценаріїв
Зрозумілість повідомлень	Наявність однієї головної дії і відсутність зайвого тексту	Суб'єктивна оцінка зрозумілості тексту

У підсумку обрані методи і технології мають оцінюватися разом. Стабільний backend не допоможе, якщо повідомлення незрозуміле. Гарний текст не має практичної цінності, якщо немає контролю доступу. Саме тому оцінювання поєднує технічні перевірки, доступність, безпеку і зрозумілість взаємодії.

3 РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ПЕРСОНАЛІЗОВАНИХ НАГАДУВАНЬ

3.1 Загальна структура та склад реалізації

Практична частина роботи виконувалася не як абстрактна схема, а як прототип, який можна запустити, перевірити і поступово розширювати. У наданому проєкті система розділена на дві основні частини: backend-сервіс та мобільний застосунок. Такий поділ добре відповідає самій задачі. Родинне нагадування має зберігатися на сервері, але отримувати його користувач буде переважно з телефона, у максимально простому інтерфейсі.

Серверна частина розміщена у каталозі backend. Вона побудована на FastAPI, а робота з даними організована через SQLAlchemy та PostgreSQL [37], [38], [39]. Для зміни структури бази використано Alembic [40]. Це важливо не тільки з технічної точки зору: у кваліфікаційному прототипі вже є кілька пов'язаних сутностей, і ручне оновлення таблиць швидко призвело б до помилок.

Мобільна частина розміщена у каталозі mobile. Вона реалізована на React Native з використанням Expo [41], [42]. Для навігації застосовано Expo Router, тому структура екранів у коді близька до структури реального застосунку: авторизація, головні вкладки, створення нагадування, голосовий режим, перегляд нагадування, журнал подій та профіль доступності [43]. Дані між екранами не передаються вручну ланцюжком пропсів; для цього використовуються API-клієнт, локальні сховища стану та запити через TanStack Query [44].

У такій архітектурі сервер не займається відображенням кнопок і кольорів, а мобільний застосунок не приймає критичних рішень щодо прав доступу. Це принципово. Якщо адресат бачить нагадування, він може його виконати або попросити допомогу, але не повинен випадково змінити час,

отримувача чи критичність доручення. Такі правила залишаються на backend-рівні.

Загальна структура реалізованої системи подана нижче. У таблиці наведено не тільки технічне призначення компонентів, а й те, яку роль вони виконують у сценарії інклюзивної родинної комунікації (таблиця 3.1).

Таблиця 3.1 – Склад реалізованої системи

Компонент	Реалізація у проєкті	Практичне призначення
Backend API	FastAPI, окремі маршрути auth, families, profiles, reminders, events, devices, uploads	Приймає запити, перевіряє права, зберігає нагадування та керує основною логікою
База даних	PostgreSQL, SQLAlchemy ORM, Alembic-міграції	Зберігає користувачів, родини, профілі доступності, нагадування, докази виконання і журнал подій
Мобільний клієнт	React Native, Expo, Expo Router, TypeScript	Дає опікуну і адресату різні екрани, не перевантажуючи вразливого користувача
AI-модулі	STT-сервіс, парсер нагадувань, спрощення тексту, українські евристики	Перетворюють голосове або текстове доручення на зрозуміле структуроване нагадування
Сповіщення	Expo push tokens, локальні сповіщення, планувальник на backend	Доставляють нагадування навіть тоді, коли користувач не перебуває на екрані застосунку
Тести	pytest-набір для авторизації, родин, нагадувань, AI-сервісів та українських евристик	Дозволяють перевірити не тільки запуск системи, а й поведінку основних сценаріїв

Реалізація не намагається замінити всі сімейні сервіси одразу. У прототипі залишено те, що справді потрібно для перевірки ідеї: створення

родинного простору, ролі користувачів, профілі доступності, створення нагадувань, голосове введення, спрощення повідомлення, підтвердження виконання та журнал подій.

Саме така межа функціональності виглядає найбільш виправданою для бакалаврської роботи. Якщо намагатися одночасно зробити календар, чат, медичний довідник і сімейну соціальну мережу, основна ідея загубиться. Тут же можна простежити повний шлях доручення: від фрази опікуна до адаптованого повідомлення і фіксації результату.

3.2 Серверна частина та API

Backend виконує роль центральної частини системи. Саме тут зосереджено перевірку користувача, права у родинній групі, створення нагадувань, роботу з профілями доступності та запис подій. У коді це зроблено через окремі маршрути, підключені у головному файлі FastAPI-застосунку. Така структура зручна тим, що кожна група дій має власну відповідальність і не змішується з іншими.

Авторизація реалізована через реєстрацію, вхід, оновлення токена і маршрут отримання поточного користувача. Для захисту запитів використовується токенна схема. У практичному застосунку це дозволяє не передавати пароль при кожній операції, а обмежувати доступ до сімейних даних через access token та refresh token [45]. Паролі зберігаються не у відкритому вигляді, а через хешування.

Окрема група маршрутів відповідає за родини. Користувач може створити сімейну групу, переглянути свої групи, додати учасника та отримати список членів. Це не декоративна частина. Без сімейного простору система перетворилася б на звичайний персональний нагадувач, а тема роботи вимагає саме родинної взаємодії.

Маршрути нагадувань є найбільш насиченими. Вони підтримують ручне створення, аналіз тексту, створення з голосової команди, перегляд

власних нагадувань, перегляд створених опікуном нагадувань, отримання детальної інформації, підтвердження виконання та завантаження фото-доказу. У цих маршрутах добре видно практичну логіку системи: нагадування живе не як статичний запис, а як процес.

Основні групи API подані нижче. У таблиці вони описані через користувацькі сценарії, бо для цієї роботи важливо не лише те, які endpoints існують, а й чому вони потрібні (таблиця 3.2).

Таблиця 3.2 – Основні групи серверних маршрутів

Група маршрутів	Приклади дій	Роль у системі
auth	реєстрація, вхід, refresh token, отримання поточного користувача	Забезпечує персональний доступ і захищає родинні дані
families	створення родини, перегляд груп, додавання учасників, список членів	Формує родинний простір і рольову взаємодію
profiles	перегляд і зміна профілю доступності користувача	Дозволяє підлаштовувати повідомлення під дитину, літню людину або стандартний режим
reminders	створення, парсинг тексту, голосове створення, виконання, фото-доказ	Описує повний життєвий цикл нагадування
events	перегляд журналу подій родини	Допомагає опікунові бачити не тільки результат, а й перебіг виконання
devices та uploads	реєстрація push-токена, завантаження аудіо і зображень	Підтримує сповіщення, голосові команди та фото-підтвердження

У серверній частині також використано репозиторний шар. Це невелика, але корисна архітектурна деталь. Маршрут не працює безпосередньо з усіма SQL-запитами, а звертається до репозиторію. Завдяки цьому код легше читати, а логіку доступу до даних простіше тестувати.

Помітна увага приділена перевірці прав. Наприклад, користувач не може просто за ідентифікатором відкрити нагадування іншої родини. Так само створення нагадування обмежується ролями опікуна або власника. Для інклюзивної системи це критично: вона працює з приватними побутовими діями, розкладом, іноді з темами здоров'я, тому відкритий доступ тут був би неприйнятним.

3.3 Модель даних та збереження інформації

Модель даних у реалізованому прототипі побудована навколо двох понять: родинний простір і нагадування. Родинний простір визначає, хто з ким взаємодіє, а нагадування описує конкретне доручення, його автора, адресата, час, текст, статус та докази виконання.

У базі даних створено таблиці `users`, `family_groups`, `family_memberships`, `accessibility_profiles`, `reminders`, `reminder_deliveries`, `completion_evidences`, `event_logs` та `device_tokens`. Такий склад не випадковий. Він відображає реальні потреби системи: знати учасників, розуміти їхню роль, зберігати індивідуальні налаштування доступності, доставляти повідомлення і фіксувати результат.

Профіль доступності має три основні типи: `CHILD`, `SENIOR` та `STANDARD`. Окрім типу, він містить масштаб шрифту, високий контраст, потребу в голосовому озвученні, спрощенні тексту, зображеннях і додаткові нотатки. Це досить практична модель: вона не намагається описати людину медичними категоріями, а зберігає саме ті параметри, які впливають на інтерфейс.

Нагадування зберігає `original_text` і `simplified_text` окремо. Це правильне рішення для такої теми. Вихідний текст потрібен для прозорості: опікун може побачити, що він сформулював. Спрощений текст потрібен для адресата. Якщо зберігати тільки фінальний варіант, буде складно зрозуміти, чи не втрачено зміст під час адаптації.

Основні сутності бази даних наведені нижче. Таблиця не повторює всю схему, але показує ті поля, які мають найбільше значення для логіки системи (таблиця 3.3).

Таблиця 3.3 – Основні сутності бази даних

Сутність	Ключові поля	Призначення
User	id, full_name, email, password_hash, is_active	Зберігає обліковий запис користувача та дані для входу
FamilyGroup	id, name, owner_id, created_at	Описує родинний простір, у межах якого створюються нагадування
FamilyMembership	family_id, user_id, role	Пов'язує користувача з родиною та визначає його роль
AccessibilityProfile	profile_type, font_scale, high_contrast, voice_enabled, simplified_text_enabled	Зберігає параметри подання повідомлень для конкретної людини
Reminder	author_id, recipient_id, original_text, simplified_text, scheduled_at, status, reminder_type	Є головною сутністю доручення і містить його стан
ReminderDelivery	reminder_id, delivered_at, delivery_type, was_opened	Фіксує факт доставки нагадування
CompletionEvidence	reminder_id, user_id, photo_url, comment, completed_at	Зберігає підтвердження виконання, зокрема фото або коментар

Продовження таблиці 3.3

Сутність	Ключові поля	Призначення
EventLog	family_id, user_id, reminder_id, event_type, description	Створює історію подій, яку може переглядати опікун
DeviceToken	user_id, expo_push_token, is_active	Потрібна для доставки push-сповіщень на пристрої користувача

3.4 Обробка текстових і голосових нагадувань

Однією з найважливіших частин прототипу є робота з природною мовою. У реальній родині опікун рідко формулює доручення як набір полів. Частіше це звичайна фраза: «Нагадай сину після школи зробити математику», «О восьмій бабусі ліки», «Щовечора полити квіти». Система повинна прийняти таку фразу і перетворити її на структуру, з якою вже можна працювати програмно.

У проєкті для цього виділено кілька сервісів: `speech_to_text`, `reminder_parser`, `uk_reminder_heuristics`, `content_simplification` та `recipient_message`. Такий поділ виглядає природно. Розпізнавання аудіо, пошук часу, визначення адресата і спрощення повідомлення – це різні задачі. Якщо змішати їх в одному великому модулі, його буде важко перевіряти і розширювати.

Speech-to-text підтримує кілька режимів: `mock`, `openai` та `local`. Для реального розпізнавання може використовуватися OpenAI Whisper API або локальний OpenAI-compatible endpoint [46]. Наявність `mock`-режиму корисна для кваліфікаційного прототипу, бо дозволяє демонструвати логіку системи без обов'язкової оплати хмарного сервісу і без залежності від мережі.

Після отримання тексту працює парсер нагадування. У коді є українські евристики, які допомагають розпізнавати типові фрази, часові

вирази й неформальні звертання. Це особливо доречно для теми роботи, бо українська побутова мова не завжди вкладається в формальні шаблони. Людина може сказати «до восьмої», «увечері», «малому», «бабусі», і система має хоча б частково підтримати такі варіанти.

Загальний шлях обробки доручення подано нижче. У таблиці показано, де система діє автоматично, а де залишає контроль за опікуном (таблиця 3.4).

Таблиця 3.4 – Обробка текстового або голосового доручення

Етап	Що відбувається	Що перевіряється
Введення	Опікун вводить текст або записує голосову команду у мобільному застосунку	Чи є команда непорожньою і чи доступний обраний режим введення
Розпізнавання мовлення	Аудіофайл передається до STT-сервісу або обробляється mosk-режимом	Чи повернувся зрозумілий текст, який можна показати користувачу
Парсинг	Виділяються адресат, дія, час, тип нагадування, повторення і фото-підтвердження	Чи не залишилися критичні поля порожніми
Українські евристики	Уточнюються часові вирази, родинні ролі та типові побутові формулювання	Чи не суперечать автоматичні підстановки вихідній фразі
Спрощення повідомлення	Текст адаптується до профілю CHILD, SENIOR або STANDARD	Чи не втрачено обов'язкову дію, час і адресата
Підтвердження	Опікун бачить результат і може виправити дані перед створенням нагадування	Чи готове нагадування до збереження

Для спрощення тексту передбачено окремий сервіс. У mosk-режимі він працює шаблонно. Для дитини формує коротший дружній текст, для

літньої людини – спокійне і просте повідомлення, для стандартного профілю залишає більш нейтральну форму. За потреби цей сервіс можна замінити на реальну мовну модель, зокрема через структуровані відповіді OpenAI API [47].

При цьому важливо, що мовна модель у цій системі не повинна бути єдиним джерелом істини. Вона може допомогти з формулюванням, але не має самостійно змінювати час, адресата або критичну дію. Саме тому structured data і текстове подання зберігаються окремо.

У такій реалізації є певна обережність, і вона виправдана. Краще поставити одне коротке уточнення, ніж створити красиве, але неправильне нагадування. Для інклюзивного сценарію помилка у часі або адресаті може бути значно серйознішою, ніж у звичайному списку справ.

3.5 Мобільний застосунок та адаптивні інтерфейси

Мобільний застосунок є тією частиною системи, з якою користувач взаємодіє щодня. Тому тут важлива не лише наявність функцій, а й те, як саме вони подані. Опікун може витримати складнішу форму, якщо вона економить час у майбутньому. Дитина або літня людина, навпаки, має бачити простий екран без зайвих налаштувань.

У проєкті використано Expo Router, тому екрани організовані файловою структурою. Є окремі екрани входу і реєстрації, основні вкладки home, reminders, family, profile, екран створення нагадування, голосовий режим, деталі нагадування, екран alarm для активного спрацювання, журнал подій і профіль доступності. Така структура добре відповідає логіці користування: більшість дій доступна з вкладок, а спеціальні сценарії відкриваються окремо.

Для роботи з формами застосовано React Hook Form і Zod. Це зручно для мобільного інтерфейсу: помилки можна показувати одразу біля поля, а не після невдалого запиту на сервер. Для зберігання токенів

використовується Expo SecureStore, а API-клієнт автоматично додає access token до запитів і намагається оновити його при відповіді 401. Користувач не бачить цієї логіки, але саме вона робить застосунок менш ламким у повсякденному використанні.

Основні екрани застосунку наведені нижче. У таблиці вони згруповані за роллю у користувацькому сценарії (таблиця 3.5).

Таблиця 3.5 – Основні екрани мобільного застосунку

Екран або група екранів	Для кого призначено	Призначення
login, register	усі користувачі	Вхід до системи та створення облікового запису
home	усі користувачі	Початковий екран із коротким доступом до актуальних дій
reminders	опікун та адресат	Перегляд отриманих і створених нагадувань
create-reminder	опікун	Ручне створення доручення з вибором адресата, часу і типу підтвердження
voice-reminder	опікун	Створення нагадування з голосової команди
reminder/[id]	опікун та адресат	Детальний перегляд нагадування, статусу і доступних дій
reminder-alarm/[id]	адресат	Окремий екран активного нагадування з мінімумом зайвих елементів
event-log	опікун	Перегляд історії подій у родині
accessibility-profile/[userId]	опікун або користувач	Налаштування способу подання повідомлень

Профіль доступності впливає на подання нагадувань. У коді він містить масштаб шрифту, контраст, голосове озвучення, спрощення тексту

і зображення (рисунок 3.1). Це не повний набір усіх можливих налаштувань, але для прототипу він охоплює найважливіше.

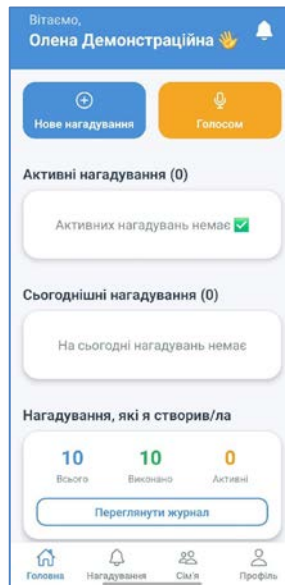


Рисунок 3.1 – Головна сторінка

Особливо важливим є голосове відтворення, реалізоване через Echo Speech, і голосове введення через Echo Audio (рисунок 3.2).

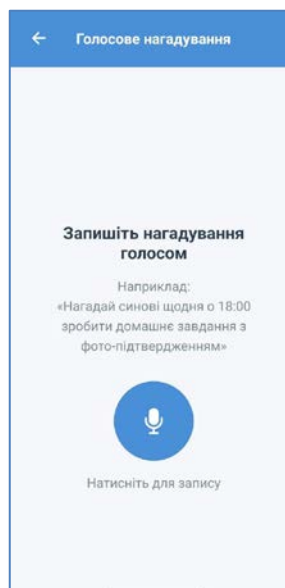


Рисунок 3.2 – Екран голосового вводу

Окремої уваги заслуговує екран активного нагадування. У звичайному застосунку можна було б просто показати toast або push-повідомлення. Для цієї теми цього недостатньо. Якщо адресат відкрив нагадування, він має бачити не список налаштувань, а одну головну дію: прочитати або прослухати, виконати, відкласти чи попросити допомогу.

У мобільному коді також передбачено локальні сповіщення. Це практичне рішення, бо в Expo Go remote push має обмеження, а демонстрація кваліфікаційного прототипу не повинна залежати тільки від production build. Застосунок синхронізує локальні нагадування зі списком отриманих задач і додатково перевіряє прострочені нагадування при активному стані програми.

Інтерфейс не повинен виглядати надто «розумним». Для користувача важливі не складні алгоритми, а відчуття, що система не плутає, не змушує зайвий раз думати і не губить важливі дії. Саме тому адаптивність у цьому прототипі пов'язана не лише з текстом, а й із поведінкою екрана.

3.6 Сповіщення, підтвердження виконання та журнал подій

Після створення нагадування система повинна довести його до адресата у потрібний момент. У реалізованому backend є планувальник, який кожні 30 секунд перевіряє прострочені активні нагадування (рисунок 3.3).

Якщо для користувача зареєстровано Expo push token, сервер надсилає push повідомлення через Expo Push API [48]. Для такого періодичного запуску використано APScheduler [49].

У мобільному застосунку використовується не лише remote push, а й локальні сповіщення. Це важливо для демонстраційного середовища. Якщо застосунок працює в Expo Go або push-токен не зареєстрований, система все одно може показати локальне нагадування на пристрої. Таким чином прототип не стає повністю залежним від зовнішнього сервісу доставки.

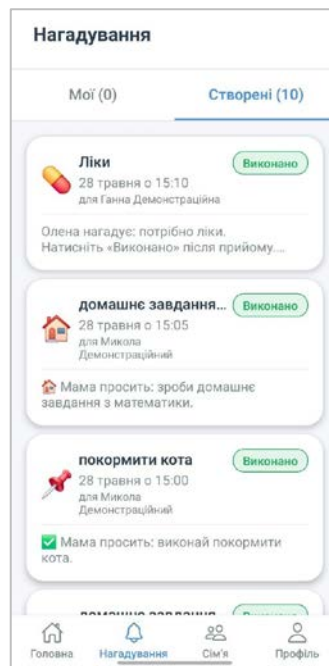


Рисунок 3.3 – Екран нагадування

Підтвердження виконання винесено в окремий сценарій. Адресат може завершити нагадування, а якщо потрібно – додати фото або коментар. У коді для цього є endpoint, complete та evidence (рисунок 3.4).

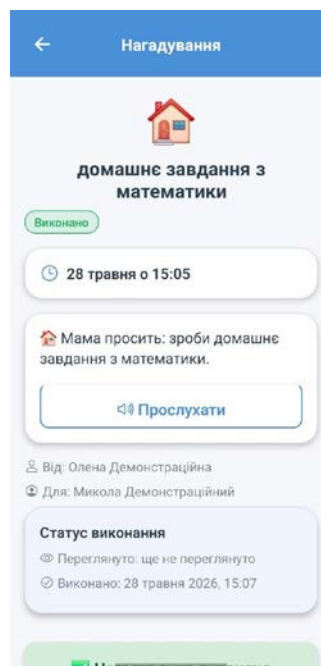


Рисунок 3.4 – Екран нагадування «Домашнє завдання з математики»

Фото завантажується через upload-механізм і зберігається як шлях до файлу, пов'язаний з CompletionEvidence. Для опікуна це означає, що результат не губиться в месенджері, а залишається в системі поруч із відповідним нагадуванням.

Журнал подій робить систему прозорішою. Він фіксує створення, доставку, виконання та інші важливі зміни. У сімейній взаємодії це знімає частину напруги: опікун не повинен щоразу перепитувати, чи було нагадування отримане, а адресат не отримує зайвих повідомлень без потреби.

Життєвий цикл нагадування подано нижче. Він показує, що нагадування в системі проходить кілька станів, а не просто зберігається як текст (таблиця 3.6).

Таблиця 3.6 – Життєвий цикл нагадування у прототипі

Стан або подія	Коли виникає	Що отримує користувач
PENDING	Нагадування створено і очікує запланованого часу	Опікун бачить активну задачу у списку створених нагадувань
DELIVERED	Сервер або мобільний застосунок доставив сповіщення адресату	Адресат бачить нагадування, а у журналі з'являється факт доставки
COMPLETED	Адресат підтвердив виконання	Опікун бачить виконаний статус і, за потреби, фото або коментар
MISSED	Підтвердження не отримано у визначеному сценарії	Система може повторити нагадування або привернути увагу опікуна
CANCELLED	Нагадування скасовано або втратило актуальність	Запис залишається в історії, але більше не очікує виконання
EventLog	Будь-яка значуща дія із нагадуванням	Опікун може відновити послідовність подій без ручного листування

При цьому важливо не перетворювати журнал на тотальний нагляд. У системі фіксуються ті події, які потрібні для роботи нагадувань: створення, доставка, завершення, доказ виконання. Для реального впровадження потрібно буде додатково обмежувати видимість журналу ролями та не зберігати зайві приватні відомості. Орієнтиром для таких рішень можуть бути вимоги до контролю доступу, валідації та журналювання з OWASP ASVS [52].

У поточному прототипі вже видно головну перевагу такого підходу. Опікун створює доручення один раз, система доставляє його у потрібний момент, адресат виконує дію у зрозумілому інтерфейсі, а результат повертається в журнал (рисунок 3.5).

← Нові нагадування

Назва нагадування

Наприклад: Домашнє завдання

Текст нагадування

Опишіть завдання детально...

ТИП НАГАДУВАННЯ

Навчання Ліки Побут Інше

ОТРИМУВАЧ

Олена Демонстраційна

Микола Демонстраційний

Ганна Демонстраційна

ДАТА І ЧАС

ДАТА

Сьогодні Завтра

Післязавтра

Рисунок 3.5 – Екран створення нагадування

Це простий сценарій, але саме він найчастіше потрібен у повсякденній родинній комунікації.

3.7 Перевірка працездатності та обмеження реалізації

Перевірка прототипу виконувалася не тільки через ручний запуск застосунку. У backend-частині є автоматичні тести на pytest [50]. Вони охоплюють авторизацію, створення родини, перевірку доступу, створення і виконання нагадування, AI-сервіси, формування повідомлень для адресата та українські евристики. Це корисно, бо саме в таких дрібних правилах найчастіше виникають непомітні помилки.

Наприклад, тестування українських евристик перевіряє розпізнавання адресата, часу, формату «до 20:30» і виправлення неправильного року. На перший погляд це технічні деталі. Насправді ж вони дуже близькі до реального користування: люди не диктують нагадування ідеальною мовою, а система має поводитися достатньо передбачувано.

Окремо перевіряються сценарії авторизації: успішна реєстрація, дублювання email, правильний і неправильний пароль, отримання поточного користувача та refresh token. Для прототипу це не менш важливо, ніж гарний інтерфейс. Якщо авторизація працює нестабільно, система не може бути основою для родинних приватних даних.

Склад перевірок подано нижче. У таблиці наведені не всі окремі тести, а групи, які показують, які частини системи вже мають автоматичну перевірку.

Для розгортання передбачено Docker Compose [51]. Це спрощує перевірку, бо backend і PostgreSQL можна запустити в узгодженому середовищі. У межах кваліфікаційної роботи це має практичну цінність: викладач або рецензент не повинен відтворювати випадкові локальні налаштування розробника вручну.

Разом із цим у реалізації залишаються обмеження. Частина AI-функцій може працювати у mock-режимі (таблиця 3.7).

Таблиця 3.7 – Перевірка працездатності прототипу

Група перевірок	Що перевіряється	Навіщо це потрібно
Авторизація	реєстрація, вхід, неправильний пароль, refresh token, отримання поточного користувача	Підтверджує, що доступ до системи не є випадковим
Родинні групи	створення родини, перегляд власних груп, додавання учасника, заборона для сторонніх	Перевіряє рольову основу системи
Нагадування	створення, доступ, виконання, перегляд отриманих нагадувань	Перевіряє головний сценарій роботи
AI-сервіси	парсинг навчального і медичного нагадування, спрощення для різних профілів	Показує, що текст адаптується не однаково для всіх
Українські евристики	часові вирази, адресати, корекція неточностей	Підтримує природні побутові формулювання
Повідомлення адресату	форматування дитячого і senior-повідомлення	Перевіряє практичну зрозумілість результату

Це не недолік для прототипу, але у промисловій версії потрібно буде чітко визначити, коли використовується локальна обробка, коли хмарний API, які дані передаються назовні та як отримується згода користувачів. Також варто посилити сценарії повторів, offline-роботу і користувацьке тестування з реальними представниками цільових груп.

Найголовніше, що прототип уже демонструє основну ідею: нагадування не просто записується як текст. Воно має автора, адресата, профіль доступності, адаптоване формулювання, статус, доставку і підтвердження. Саме ця зв'язка відрізняє систему від звичайного календаря.

Також перевірка показала, що система потребує не лише технічної стабільності, а й подальшого уточнення поведінки в неоднозначних ситуаціях. Наприклад, користувач може вказати неповний час, не назвати адресата прямо або сформулювати доручення занадто загально. У таких

випадках система має не просто «вгадувати», а або обережно уточнювати дані, або створювати нагадування у безпечному нейтральному форматі.

Тому наступним кроком після базового тестування має бути перевірка на реальних побутових прикладах. Саме вони дозволяють побачити, чи справді повідомлення зрозуміле адресату, чи не створює воно зайвого тиску, і чи достатньо опікуну тих підтверджень, які вже передбачені у прототипі.

3.8 Результати розробки

У результаті розроблено програмний прототип інтелектуальної системи персоналізованих нагадувань. Він складається з backend API, мобільного застосунку, бази даних, AI-сервісів, механізму сповіщень і набору автоматичних тестів. Система підтримує реєстрацію користувачів, родинні групи, ролі, профілі доступності, створення нагадувань текстом і голосом, адаптацію повідомлення, підтвердження виконання, фото-доказ і журнал подій.

Практичний результат полягає не лише у наборі екранів або таблиць бази даних. Важливіше, що у прототипі зібрано повний користувацький маршрут. Опікун може сформулювати доручення звичайною мовою, система допомагає виділити його зміст, адресат отримує простіше повідомлення, а виконання не губиться у зовнішньому листуванні.

На цьому етапі система ще не є готовим комерційним продуктом. Їй потрібні додаткові тести інтерфейсу, перевірка на різних пристроях, розширена політика приватності та більш глибока робота з повторюваними сценаріями. Проте як кваліфікаційний прототип вона показує, що тема може бути реалізована технічно і має практичний сенс для родинного середовища.

Подальше розширення доцільно виконувати поступово. Спочатку варто додати більше шаблонів для різних типів доручень, потім – кращу роботу з календарними контекстами, а вже після цього – користувацькі дослідження з дітьми, літніми людьми та опікунами. Такий порядок

дозволить не втратити головну мету: зробити нагадування зрозумілим, а не просто технологічно складним.

Окремо слід зазначити, що під час розробки вдалося перевірити життєздатність самої архітектурної ідеї. Backend, мобільний застосунок, база даних і модулі сповіщень не існують окремо один від одного, а працюють як пов'язана система. Це важливо, тому що для користувача цінність має не сам факт наявності AI-модуля чи API, а те, що нагадування проходить увесь шлях: створюється, зберігається, адаптується, доставляється і може бути підтверджене. Саме така послідовність робить прототип ближчим до реального використання, а не лише до демонстраційного прикладу.

Також результат розробки показав, що інклюзивність у подібній системі краще закладати не наприкінці, а ще на рівні логіки роботи застосунку. Якщо профіль доступності, роль користувача, спосіб подання повідомлення і підтвердження виконання враховуються одразу, система стає гнучкішою. У подальшому це дозволить без повного переписування проєкту додавати нові типи користувачів, інші способи сповіщень, додаткові формати підтвердження та більш точні правила адаптації тексту.

Водночас отриманий результат показує, що така система має сенс саме в родинному, а не лише в індивідуальному використанні. Звичайний застосунок нагадувань часто розрахований на одну людину, яка сама створює задачу і сама її виконує. У цьому прототипі ситуація інша: одна людина може сформулювати доручення, інша – отримати його у зручнішому вигляді, а потім підтвердити виконання. Через це система ближча до реальної побутової взаємодії, де важливі не тільки час і текст, а й довіра між учасниками, зрозумілість повідомлення та можливість спокійно перевірити результат.

ВИСНОВКИ

У кваліфікаційній роботі було розглянуто створення інтелектуальної системи персоналізованих нагадувань для інклюзивної родинної комунікації. Основна ідея роботи полягала в тому, що нагадування у родині не завжди є простим сигналом про час. Часто воно виконує роль короткого доручення, пояснення або підтримки для людини, якій складно сприймати довгі повідомлення, дрібний текст чи перевантажений інтерфейс. Тому в роботі нагадування розглянуто як частину живої взаємодії між опікуном і адресатом, а не лише як технічну функцію календаря.

У першому розділі було проаналізовано предметну галузь. Показано, що інклюзивна комунікація стосується не тільки людей з офіційно встановленою інвалідністю. У родинному середовищі вразливими можуть бути діти, літні люди, користувачі з порушеннями пам'яті, люди після хвороби або ті, хто просто погано орієнтується у цифрових сервісах. Для таких користувачів важливими є зрозумілість, передбачуваність, спокійний тон повідомлення і можливість виконати дію без зайвого напруження.

Огляд існуючих рішень показав, що календарі, списки справ, голосові асистенти, сімейні органайзери та медичні нагадувачі мають корисні окремі функції. Водночас вони рідко поєднують ці функції в один сценарій. Зазвичай користувач сам відповідає за формулювання задачі, її структуру і зрозумілість. Це обмежує такі рішення у випадках, коли один член родини створює доручення для іншого, а повідомлення потрібно не просто доставити, а подати у доступній формі.

У другому розділі було обґрунтовано вибір методів і технологій розробки. Для системи такого типу доцільно поєднати серверну частину, мобільний клієнт, базу даних, модулі обробки природної мови, механізм сповіщень і рольове керування доступом. Такий підхід є виправданим, оскільки кожен компонент відповідає за реальну потребу: збереження

даних, контроль родинних ролей, адаптацію повідомлень, доставку нагадувань, підтвердження виконання та збереження історії подій.

Окрему увагу приділено методам обробки природної мови. У побуті люди рідко формулюють доручення у вигляді точних полів. Вони говорять простіше: «нагадай бабусі ввечері прийняти ліки» або «скажи дитині після школи зробити математику». Для системи така фраза потребує розбору: потрібно визначити адресата, дію, час, важливість і можливе підтвердження. При цьому AI-модуль не повинен змінювати зміст доручення, а має лише допомогти швидше перетворити звичайну фразу на коректне нагадування.

У третьому розділі описано реалізацію прототипу. На основі наданого програмного проєкту розглянуто backend на FastAPI, мобільний застосунок на React Native та Expo, PostgreSQL-базу даних, профілі доступності, маршрути API, AI-сервіси, push- і локальні сповіщення, підтвердження виконання та журнал подій. Практична частина показує, що запропонована система може працювати як цілісний ланцюг: від голосової або текстової фрази опікуна до адаптованого повідомлення, яке отримує адресат.

Важливим результатом є те, що в системі не змішується вихідний текст доручення і його адаптований варіант. Це дає змогу зберегти прозорість. Опікун може бачити, що було сказано спочатку, а адресат отримує форму, зручнішу саме для нього. Такий підхід особливо важливий для нагадувань, пов'язаних із навчанням, здоров'ям або безпекою, де втрата навіть однієї умови може змінити зміст доручення.

Розроблений прототип підтримує родинні ролі, профілі доступності CHILD, SENIOR та STANDARD, створення нагадувань текстом і голосом, спрощення повідомлення, push- та локальні сповіщення, фото-підтвердження і журнал подій. Наявність автоматичних тестів для авторизації, родинних груп, нагадувань, AI-сервісів та українських евристик підтверджує, що практична частина не обмежується лише описом і макетами.

Практичне значення роботи полягає у можливості використати отримані результати як основу для подальшого розвитку мобільного застосунку для родин, опікунів або невеликих груп підтримки. Така система може зменшити кількість повторних пояснень, зробити доручення зрозумілішими і дати опікуну контроль без постійного ручного нагадування у месенджерах. Вона не замінює живе спілкування, але може зняти частину рутинного навантаження.

Наукова новизна роботи полягає у розгляді персоналізації нагадувань не тільки як налаштування часу або каналу доставки, а як адаптації змісту, форми подання, рівня деталізації, способу підтвердження та сценарію повторення відповідно до профілю користувача. У роботі поєднано інклюзивну комунікацію, рольову модель родинного простору, обробку природної мови та принципи когнітивної доступності.

Під час виконання роботи також було визначено практичні обмеження, які варто враховувати надалі. Якість адаптації залежить не лише від мовної моделі, а й від правильно заповненого профілю користувача, коректного визначення ролей та перевірки критичних доручень. Тому система має залишати за опікуном контроль у ситуаціях, пов'язаних зі здоров'ям або безпекою.

Мету роботи досягнуто. Було проаналізовано предметну галузь, обґрунтовано методи та технології розробки, а також описано реалізацію прототипу інтелектуальної системи персоналізованих нагадувань. Подальший розвиток системи доцільно спрямувати на розширення сценаріїв, локальне розпізнавання мовлення, кращу обробку українських побутових фраз, посилення приватності та користувацьке тестування з представниками цільових груп.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Attention is all you need / A. Vaswani et al. *arXiv*. 2017. URL: <https://doi.org/10.48550/arXiv.1706.03762> (date of access: 07.06.2026).
2. A survey of large language models / W. X. Zhao et al. *arXiv*. 2023. URL: <https://doi.org/10.48550/arXiv.2303.18223> (date of access: 07.06.2026).
3. Automatic sign language translation system using neural network technologies and 3D animation / Y. Shovkovyi et al. *Innovative technologies and scientific solutions for industries*. 2023. No. 4(26). P. 108–121. URL: <https://doi.org/10.30837/itssi.2023.26.108> (date of access: 24.06.2026).
4. Disability. *World Health Organization (WHO)*. URL: <https://www.who.int/news-room/fact-sheets/detail/disability-and-health> (date of access: 07.06.2026).
5. Change Google calendar notifications. *Google Help*. URL: <https://support.google.com/calendar/answer/37242> (date of access: 07.06.2026).
6. Reminders. *App Store*. URL: <https://apps.apple.com/us/app/reminders/id1108187841> (date of access: 07.06.2026).
7. Ростовцев В.В., Гриньов С.А. Гібридна система детекції синтезованого мовлення на основі SSL-ембеддінгів. *Радіoeлектроніка та молодь у XXI столітті* : матеріали 30-го Міжнар. молодіж. форуму, 22–24 квітня 2026 р. Харків : ХНУРЕ, 2026. Т. 6. С. 105–107.
8. Introduction to recurring dates. *Todoist*. URL: <https://www.todoist.com/help/articles/introduction-to-recurring-dates-YUYVJJAV> (date of access: 07.06.2026).
9. Cozi family organizer. *Cozi*. URL: <https://www.cozi.com/> (date of access: 07.06.2026).
10. Set reminders with Alexa. *Amazon*. URL: <https://www.amazon.com/gp/help/customer/display.html?nodeId=GDDXYWQUKGGWG79Y> (date of access: 07.06.2026).

11. Use reminders on your iPhone, iPad, or iPod touch. *Apple Support*. URL: <https://support.apple.com/en-us/102484> (date of access: 07.06.2026).
12. App features. *Medisafe*. URL: <https://medisafeapp.com/features/> (date of access: 07.06.2026).
13. Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems : ISO 9241-210:2019. URL: <https://www.iso.org/standard/77520.html> (date of access: 07.06.2026).
14. Web content accessibility guidelines (WCAG) 2.2 : W3C recommendation. URL: <https://www.w3.org/TR/WCAG22/> (date of access: 07.06.2026).
15. Making content usable for people with cognitive and learning disabilities. *W3C Working Group Note*. URL: <https://www.w3.org/TR/coga-usable/> (date of access: 07.06.2026).
16. The 7 principles of universal design. *Centre for Excellence in Universal Design*. URL: <https://universaldesign.ie/about-universal-design/the-7-principles> (date of access: 07.06.2026).
17. Cognitive accessibility at W3C. *W3C WAI*. URL: <https://www.w3.org/WAI/cognitive/> (date of access: 07.06.2026).
18. Inclusive design toolkit. *Microsoft*. URL: <https://inclusive.microsoft.design/> (date of access: 07.06.2026).
19. General data protection regulation. Art. 5. Principles relating to processing of personal data : regulation of 27.04.2016 no. 2016/679. URL: https://gdpr-info.eu/art-5-gdpr/?utm_source=chatgpt.com (date of access: 07.06.2026).
20. Про захист персональних даних : Закон України від 01.06.2010 № 2297-VI : станом на 14 черв. 2025 р. URL: <https://zakon.rada.gov.ua/laws/show/2297-17#Text> (дата звернення: 24.06.2026).
21. Jurafsky D., Martin J. H. Speech and language processing. 3rd ed. draft. URL: <https://web.stanford.edu/~jurafsky/slp3/> (date of access: 07.06.2026).

22. BERT: pre-training of deep bidirectional transformers for language understanding / J. Devlin et al. *arXiv*. 2018. URL: <https://doi.org/10.48550/arXiv.1810.04805> (date of access: 07.06.2026).
23. Robust speech recognition via large-scale weak supervision / A. Radford et al. *arXiv*. 2022. URL: <https://doi.org/10.48550/arXiv.2212.04356> (date of access: 07.06.2026).
24. Accessibility requirements for ICT products and services : ETSI EN 301 549 V3.2.1 (2021-03). URL: https://www.etsi.org/deliver/etsi_en/301500_301599/301549/03.02.01_60/en_301549v030201p.pdf (date of access: 07.06.2026).
25. Web content accessibility guidelines (WCAG) 2.2 : W3C recommendation. URL: <https://www.w3.org/TR/WCAG22/> (date of access: 14.06.2026).
26. Making content usable for people with cognitive and learning disabilities. *W3C Working Group Note*. URL: <https://www.w3.org/TR/coga-usable/> (date of access: 14.06.2026).
27. Accessibility. *Flutter Documentation*. URL: <https://docs.flutter.dev/ui/accessibility> (date of access: 14.06.2026).
28. Learn React. *React Documentation*. URL: <https://react.dev/> (date of access: 14.06.2026).
29. FastAPI documentation. *FastAPI*. URL: <https://fastapi.tiangolo.com/> (date of access: 14.06.2026).
30. JSON types. *PostgreSQL Documentation*. URL: <https://www.postgresql.org/docs/current/datatype-json.html> (date of access: 14.06.2026).
31. Redis job queue. *Redis Documentation*. URL: <https://redis.io/docs/latest/develop/use-cases/job-queue/> (date of access: 14.06.2026).
32. Firebase Cloud Messaging. *Firebase Documentation*. URL: <https://firebase.google.com/docs/cloud-messaging> (date of access: 14.06.2026).

33. Hardt D. The OAuth 2.0 authorization framework : RFC 6749. URL: <https://datatracker.ietf.org/doc/html/rfc6749> (date of access: 14.06.2026).
34. OWASP application security verification standard. OWASP. URL: <https://owasp.org/www-project-application-security-verification-standard/> (date of access: 14.06.2026).
35. Structured model outputs. *OpenAI API Documentation*. URL: <https://developers.openai.com/api/docs/guides/structured-outputs> (date of access: 14.06.2026).
36. Docker overview. *Docker Documentation*. URL: <https://docs.docker.com/get-started/docker-overview/> (date of access: 14.06.2026).
37. FastAPI documentation. *FastAPI*. URL: <https://fastapi.tiangolo.com/> (date of access: 14.06.2026).
38. SQLAlchemy 2.0 documentation. *SQLAlchemy*. URL: <https://docs.sqlalchemy.org/en/20/> (date of access: 14.06.2026).
39. PostgreSQL documentation. *PostgreSQL*. URL: <https://www.postgresql.org/docs/> (date of access: 14.06.2026).
40. Alembic documentation. *Alembic*. URL: <https://alembic.sqlalchemy.org/> (date of access: 14.06.2026).
41. React Native documentation. *React Native*. URL: <https://reactnative.dev/docs/getting-started> (date of access: 14.06.2026).
42. Expo documentation. *Expo*. URL: <https://docs.expo.dev/> (date of access: 14.06.2026).
43. Expo Router documentation. *Expo*. URL: <https://docs.expo.dev/router/introduction/> (date of access: 14.06.2026).
44. TanStack Query documentation. *TanStack Query*. URL: <https://tanstack.com/query/latest/docs/framework/react/overview> (date of access: 14.06.2026).
45. Jones M., Bradley J., Sakimura N. JSON web token (JWT) : RFC 7519. URL: <https://datatracker.ietf.org/doc/html/rfc7519> (date of access: 14.06.2026).

46. Audio transcriptions. *OpenAI API Documentation*. URL: <https://platform.openai.com/docs/guides/speech-to-text> (date of access: 14.06.2026).

47. Structured model outputs. *OpenAI API Documentation*. URL: <https://platform.openai.com/docs/guides/structured-outputs> (date of access: 14.06.2026).

48. Push notifications. *Expo Documentation*. URL: <https://docs.expo.dev/push-notifications/overview/> (date of access: 14.06.2026).

49. APScheduler documentation. *APScheduler*. URL: <https://apscheduler.readthedocs.io/> (date of access: 14.06.2026).

50. pytest documentation. *pytest*. URL: <https://docs.pytest.org/> (date of access: 14.06.2026).

51. Docker overview. *Docker Documentation*. URL: <https://docs.docker.com/get-started/docker-overview/> (date of access: 14.06.2026).

52. OWASP application security verification standard. *OWASP*. URL: <https://owasp.org/www-project-application-security-verification-standard/> (date of access: 14.06.2026).