Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Навчально-науковий інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня бакалавра

здобувача Гаркуші Ярослава Володимировича
(ПІБ)

академічної групи 123-22ск-1
(шифр)

спеціальності 123 Комп'ютерна інженерія
(код і назва спеціальності)

за освітньо-професійною програмою 123 Комп'ютерна інженерія
(офіційна назва)

на тему «Голосовий модуль мобільного застосунку з використанням AI-сервісів хмарного розпізнавання»
(назва за наказом ректора)

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
кваліфікаційної роботи	доц. Дмитрієва І.С.			
Рецензент				
Нормоконтролер	проф. Цвіркун Л.І.			

Дніпро
2025

ЗАТВЕРДЖЕНО:
завідувач кафедри
інформаційних технологій
та комп'ютерної інженерії
(повна назва)
Гнатушенко В.В.

"__" 2025 року

ЗАВДАННЯ
на кваліфікаційну роботу ступеня бакалавр

здобувача Гаркуші Ярослава Володимировича
академічної групи 123-22ск-1
спеціальності 123 Комп'ютерна інженерія
за освітньо-професійною програмою Комп'ютерна інженерія
(офіційна назва)
на тему «Голосовий модуль мобільного застосунку з використанням AI-сервісів хмарного розпізнавання»

затверджену наказом ректора НТУ «Дніпровська політехніка»
від 05.05.2025 №336-с

Розділ	Зміст	Термін виконання
Загальна частина	На основі матеріалів виробничих практик, інших науково-технічних джерел показати актуальність завдання, сформулювати мету та задачі виконання кваліфікаційної роботи	10.03.2025
Спеціальна частина	розробити мовний інтерфейс систем інтелектуальної телефонії.	09.06.2025

Завдання видано _____ доц. Дмитрієва І.С.
(підпис керівника) (прізвище, ініціали)

Дата видачі 10.03.2025

Дата подання до екзаменаційної комісії 10.06.2025 р.

Прийнято до виконання _____ Гаркуша Я.В.

РЕФЕРАТ

Пояснювальна записка: 67 стор., 34 рисунки, 1 таблиці, 12 джерел.

МОВНИЙ ІНТЕРФЕЙС, ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ ТЕЛЕФОНІЇ, РОЗПІЗНАВАННЯ МОВИ, ANDROID, GOOGLE.

Об'єкт дослідження — мовний інтерфейс систем інтелектуальної телефонії.

Предмет дослідження — існуючі мережеві сервіси розпізнання мови.

Мета роботи — розробка мовного інтерфейсу систем інтелектуальної телефонії.

Методи дослідження — теорія розпізнавання мови, інструментальні засоби для розроблення програмних додатків.

У даній роботі розглядається процес розробки програмного додатку для мобільних пристроїв, який підтримує такі функції: набір номеру телефону, надсилання SMS, голосове редагування контактних даних, голосове введення тексту та номеру для SMS, а також номеру для здійснення дзвінків. Крім того, передбачено можливість перемикання між функціями додатку за допомогою голосових команд. На основі аналізу наявних рішень зроблено висновок про актуальність створення мовного інтерфейсу з використанням сучасних мережевих сервісів розпізнавання мовлення. Обґрунтовано вибір інструментів для розробки додатків під операційну систему Android. За результатами тестування встановлено, що середня точність розпізнавання голосових команд становить 92%, що відповідає результатам тестування системи від Google. Розроблений мовний інтерфейс може слугувати основою для створення перспективної інтелектуальної мовної системи.

ЗМІСТ

ВСТУП

Однією з актуальних задач розробки сучасних інформаційних систем є розробка інтерфейсів, які забезпечують можливість спілкування з користувачем природними мовними засобами. Практична цінність мовних інтерфейсів головним чином пов'язана з можливістю видаленого керування інформаційною системою з використанням тільки голосового каналу. У цьому випадку виключаються потреби в додатковому каналі керування і апаратному забезпеченні цього каналу. При використанні мовного управління також з'являється можливість звільнення кінцівок оператора, що вже використовується в системах hand-free. Огляд існуючих систем показує наявність недоліків:

- не може розпізнати в режимі off-line,
- обмежена функціональність – можна використовувати тільки як записну книжку;
- немає можливості вбудованого перекладу тексту .

Всі ці недоліки головним чином визначаються тим, що кожен розробляє свою систему розпізнавання мови. У цей же час новітні технології дають можливість використовувати мережеві сервіси, які надають можливість використання програмних продуктів, розроблених другими фірмами. Це дозволяє зменшити термін розробки нових додатків і підняти якість розпізнавання за рахунок використання добре налагоджених і протестованих систем.

Метою даної роботи є розробка мовного інтерфейсу інформаційної системи для потреб телефонії. Розроблений в роботі інтерфейс у вигляді програмного додатку працює під керуванням операційної системи Android. Використання цієї системи пов'язано з швидким зростанням пристроїв зв'язку під керуванням цієї системи та можливістю використання існуючих мережних сервісів розпізнавання мови.

1 ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ МОВИ В ІНТЕЛЕКТУАЛЬНИХ СИСТЕМАХ ТЕЛЕФОНІЇ

1.1 Поняття інтелектуальної інформаційної системи

Штучний інтелект – це один з напрямків інформатики, метою якого є розробка апаратно-програмних засобів, що дозволяють користувачеві-непрограмісту ставити і вирішувати свої, що традиційно вважаються інтелектуальними завдання, спілкуючись з ЕОМ на обмеженій підмножині природної мови. На сьогодні вважається, що інтелектуальні системи – це самонавчальний інструмент, який посилює діяльність людини по генерації та прийняття рішень. За останнім трактуванням ШІ є експериментальною науковою дисципліною, в якій роль експерименту полягає у перевірці та уточненні інтелектуальних систем, що представляють собою апаратно-програмні інформаційні комплекси. Як і в будь-якій дисципліні йому властиві характерні терміни та визначення.

Характерною особливістю знань є те, що вони не містяться у вихідній системі, знання виникають в результаті зіставлення інформаційних одиниць (людина - годинник - спізнюється), знаходження та вирішення суперечностей між ними. Тобто знання активні, їх поява або недостача призводить до реалізації деяких дій. Стан - другий найважливіший термін. Кожна інформаційна одиниця як і вся система в цілому, може перебувати в одному з станів. Наприклад, включений - вимкнений. У загальному випадку простір станів може бути різним. Його зручно представляти у вигляді графіка або таблиці переходів. Перехід зі стану в стан може мати властивість симетричності. У цьому випадку система здатна повернутися в початковий стан (а комп'ютерна програма зациклитися).

1.2 Системи з інтелектуальним інтерфейсом

Інтелектуальні бази даних дозволяють на відміну від традиційних БД забезпечувати вибірку необхідної інформації, що не присутня ні в явному

вигляді, виведеної із сукупності збережених даних природно - мовний інтерфейс. Застосовується для доступу до інтелектуальних баз даних, контекстного пошуку документальної текстової інформації, голосового введення команд в системах управління, машинного перекладу з іноземних мов. Для реалізації ПР-інтерфейсу необхідно вирішити проблеми морфологічного, синтаксичного і семантичного аналізу, а також завдання синтезу висловлювань на ПР. Морфологічний аналіз передбачає розпізнавання і перевірку правильності написання слів за словниками, синтаксичний контроль - розкладання вхідних повідомлень на окремі компоненти (визначення структури) з перевіркою відповідності граматичними правилами внутрішнього подання знань і виявлення відсутніх частин і, нарешті, семантичний аналіз - встановлення смислової правильності синтаксичних конструкцій. Синтез висловлювань вирішує зворотню задачу перетворення внутрішнього подання інформації в природно-мовну. Розглянемо класифікацію систем.

Гіпертекстові системи. Використовуються для реалізації пошуку за ключовими словами в базах даних з текстовою інформацією. Для більш повного відображення різних смислових відносин термінів потрібна складна семантична організація ключових слів. Вирішення цих завдань здійснюється за допомогою інтелектуальних гіпертекстових систем, в яких механізм пошуку працює спочатку з базою знань ключових слів, а потім - з текстом.

Системи контекстної допомоги відносяться до класів систем поширення знань. Такі системи, як правило, є додатками до документації (видавництво знань). Системи контекстної допомоги - окремий випадок гіпертекстових і ПР-систем.

Системи когнітивної графіки орієнтовані на спілкування з користувачем за допомогою графічних образів, які генеруються відповідно до зміни параметрів модельованих або спостерігаються. Когнітивна графіка дозволяє в наочному і виразному вигляді уявити безліч параметрів, що характеризують досліджуване явище.

1.3 Огляд існуючих технологій та систем розпізнавання мови

Завдання створення надійної системи розпізнавання мови, стійкої до шумів, з низькою частотою появи помилок, є однією з актуальних на сьогоднішній день. Технології розпізнавання мови з'явилися вельми давно. Добре відомі історичні роботи Девіса, Біддульфа і Балашека (1952), Нагата, Като і Чіби (1962), Зайцева і Тимофєєва (1965), Кінга і Тьюніса (1966), Голда (1966), Величко та Загоруйко (1969). Особливо швидко розвиток технології розпізнавання мови отримали після появи пристроїв цифрової обробки, виконаних у вигляді мікросхем і що дозволили створити відносно дешеві розпізнаваючі системи, що працювали в режимі реального часу. У світі зростання обчислювальної потужності спочатку спеціалізованих акустичних, а потім і цифрових сигнальних процесорів ускладнювалися і удосконалювалися алгоритмами, що використовувалися в системах розпізнавання мови. Однак точність систем розпізнавання мови досягла свого піку в 1999 році і з тих пір застигла на місці. Різні тести показують, що сучасні системи загального профілю так і не подолали рівень розпізнавання в 80%, тоді як у людини цей показник становить 96-98%. Тому вкрай необхідно продовжувати дослідження в цій області.

1.3.1 Огляд функцій додатку Siri

Siri – це персональний помічник для iPhone, який далеко не обмежується лише реагуванням на прості команди. Siri розуміє контекст і дозволяє вам говорити природною мовою, щоб давати телефону різні завдання: прочитай мої SMS, відправ лист такому-то адресату, нагадай мені зробити що-небудь, перевір погоду і т.д. Після того як всі необхідні налаштування внесені, ви вже можете користуватися своїм електронним асистентом.

Існує кілька способів того, як можна задіяти Siri. Якщо ви тримаєте телефон в руках, ви можете просто утримувати кнопку Home протягом 2 секунд: з'явиться піктограма мікрофона, пролунає звуковий сигнал, і це означає, що ви вже можете говорити. Крім того, можна також піднести

телефон до вуха, після чого ви почуєте такий же сигнал. Якщо ви користуєтеся гарнітурою Bluetooth, ви можете активувати Siri, натиснувши на ній кнопку. Після того як ви почули звуковий сигнал, ви можете питати Siri про те, що вам потрібно.

1.3.2 Огляд функцій додатку ListNote

Додаток ListNote отримало інформацію користувача оцінку 4,2 і займе на вашому пристрої з Android всього 1,8 МБ пам'яті. Демонструє простоту, надійність і хорошу сумісність з ОС Android починаючи з версії 2.3.3 і далі. Розпізнавання сказаних слів відбувається практично без помилок. Розуміє широкий запас слів, але не безмежний, про що докладніше викладено в перевагах та недоліках. В цілому, роботою цього додатка користувачі залишаються задоволені, тому в ньому немає нічого зайвого, з поставленим завданням воно добре справляється. Має інтуїтивно зрозуміле меню і хорошу довідку.

Недоліки

- обмежена функціональність - можна використовувати тільки як записну книжку;
- не може розпізнати слова які рідко зустрічаються, нецензурну лексику;
- немає можливості вбудованого перекладу тексту;
- не працює розпізнавання мови без підключення до інтернету.

1.3.3 Огляд функцій додатку Speech to Text Translator TTS

А це вже додаток для Android з більш широким набором функцій, нехай і отримав користувальницьку оцінку 4,0. Але тому є очевидні причини - дуже нав'язлива яскрава реклама псує весь інтерфейс. Крім того реклама додатково їсть мобільний трафік, а без підключення до інтернету додаток працювати не буде. Тим не менш, цей додаток для розпізнавання мови показав себе більш ніж добре: відмінно розпізнає голос, отриманий текст можна відправити як СМС, поштою або іншим способом.

Недоліки

- не може розпізнати в режимі оффлайн;
- довго здійснює переказ;
- платна версія обійдеться в 59,9 доларів;
- можливі проблеми з установкою - потрібно повторити спробу.

1.3.4 Додаток Voice to text Messenger

Менш популярне, але дуже зручний додаток для Android, примітною особливістю даної програми є зручна можливість редагування і подальшої відправки отриманого тексту. Меню не напружує око своїм розфарбуванням. У головному меню є всього 4 кнопки, з яких вам знадобляться 2: «нотатки» і «додати». У першому випадку можна подивитися раніше створені нотатки. У другому випадку ви зможете створити новий запис. Відправити отриманий текст можна тим способом, яким вам зручно: СМС, пошта mail.ru, Bluetooth, Skype, Gmaili та інші.

Недоліки

- немає можливості розпізнавати мову без підключення до інтернету;
- немає перекладача тексту.

Слід зазначити, що якість розпізнавання мови багато в чому залежить не тільки від коректної роботи програми, апаратної частини мобільного пристрою і самої ОС Android. Істотну роль надає чистота вимови слів користувача. Так при тестуванні англомовних додатків, часто спостерігається некоректне розпізнавання слів вимовлених користувачем з акцентом. Крім того, наявність сторонніх шумів у свою чергу також негативно виявляється на їх роботі. Всі ці фактори створюють певні обмеження в застосуванні даних програм. Наприклад, записати щось «на бігу» буде простіше звичайним диктофоном. В інших випадках, відправити смс якщо руки зайняті можна і по засобом власного голосу.

1.4 Методи розпізнавання мови

1.4.1 Методи і моделі

Для успішного розпізнавання мови слід вирішити наступні завдання:

- обробку словника (фонемний склад),

- обробку синтаксису,

- скорочення мовлення (включаючи можливе використання жорстких сценаріїв),

- вибір диктора (включаючи вік, стать, рідну мову і діалект),

- тренування дикторів,

- вибір особливого виду мікрофона (беручи до уваги спрямованість і місце розташування мікрофона),

- умови роботи системи та отримання результату із зазначенням помилок.

Проводиться процес, першим кроком якого є первісне трансформування вводиться для скорочення оброблюваного обсягу так, щоб її можна було б піддати комп'ютерному аналізу. Прикладом є «техніка зіставлення відрізків», що дозволяє скоротити ввод інформації з 50'000 до 800 бітів в секунду. Наступним етапом є спектральне подання мови, що вийшло шляхом перетворення Фур'є. Результат перетворення Фур'є дозволяє не тільки стиснути інформацію, але й дає можливість сконцентруватися на важливих аспектах мови, які інтенсивно вивчалися в сфері експериментальної фонетики. Спектральне подання досягнуто шляхом використання широко-частотного аналізу запису.

Хоча спектральне подання мови дуже корисно, необхідно пам'ятати, що досліджуваний сигнал досить різноманітний. Різноманітність виникає з багатьох причин, включаючи:

- відмінності людських голосів;

- рівень мови мовця;

- варіації у вимові;

- нормальне варіювання руху артикуляторів (язика, губ, щелепи, піднебіння).

Для усунення негативного ефекту впливу варіювання голосового тракту на процес розпізнавання мови було використано безліч методів. Насамперед розглядалася характеристика простору траєкторії артікуляторних органів, включаючи голосні, використовувані мовцем. Найбільш вдалі форми трансформації, використаної для скорочення відмінностей, були вперше представлені Сако & Чібо і називалися динамічними спотвореннями (dynamic time warping). Техніка динамічного перекручування використовується для тимчасового витягування і скорочення відстані між спотвореним спектральним поданням і шаблоном для мовця. Використання даної техніки дало поліпшенні точного розпізнавання (~ 20-30%). Метод динамічного перекручування використовують практично всі комерційно доступні системи розпізнавання, що показують високу точність повідомлення при використанні. Спочатку сигнал перетворюється в спектральне подання, де визначається нечисленний, але високоінформативний набір параметрів.

Потім визначаються кінцеві вихідні параметри для варіювання голосу (слід зазначити, що дана задача не є тривіальною) і виробляється нормалізація для складання шкали параметрів, а також для визначення ситуаційного рівня мови. Вищеописані змінені параметри використовуються потім для створення шаблону. Шаблон включається в словник, який характеризує проголошення звуків при передачі інформації мовцем, що використовує цю систему. Далі в процесі розпізнавання нових мовних зразків (вже зазнали нормалізації і отримали свої параметри), ці зразки порівнюються з шаблонами, вже наявними в словнику, використовуючи динамічне перекручування і схожі метричні виміри. В цей час цей метод вивчається і доповнюється.

Очевидно, що спектральне подання мови дозволяє характеризувати особливості голосового тракту людини і спосіб використання його мовцем. Найпростіший спосіб моделювання специфічних ефектів "модель-джерело" -

використання фільтрів. Мовний апарат моделюється з використанням джерел, що викликають резонанс, провідний до пікових точках інтенсивності звуку в сусідстві з окремими частотами, званими формантами. При проголошенні звуків вібрація голосових зв'язок є джерелом порушення, і ці короткі імпульси викликають резонанс між голосовими зв'язками і губами. Так як мова, щелепа, губи, зуби і альвеолярний апарат рухаються, розмір і місце цих резонансів змінюються, даючи можливість відтворення особливих параметрів звуків.

Можливо побудувати дуже точну модель, також прямо змоделювати рух артикуляторів фізіологічно реальним шляхом. Використання цих моделей привели до розуміння шляху, в якому відбувається мовний сигнал. Але так як спостереження над артикуляторами утруднене, залишаються недоліки. Хоча природа вокального тракту дуже сильно впливає на вихідний сигнал мови, це не єдине обмеження, яке необхідно брати до уваги, так як контроль над м'язами звукового тракту обумовлений сигналами моторного кортекса мозку. Можливо всі аспекти впливу акустичної структури контролюють сигнали і форму звукового виходу мови (хоча це не може бути доведено за систематичною точки зору).

Аспекти впливу акустичної структури включає в себе:

- природу сегментів індивідуального звуку (голосні / приголосні),
- структуру складу,
- структуру морфем (приставки, коріння, суфікси),
- лексикон,
- рівень синтаксису фраз і пропозицій,
- довгострокові обмеження мови (long-term discourse constraints).

Нижче розглядається вплив обмежень і спосіб їх впливу виробництво сигналу мови. Необхідно також взяти до уваги той факт, що людський апарат сприйняття також повинен бути змодельований, він сам по собі накладає на процес сприйняття додаткові обмеження. Нещодавно процес сприйняття був вивчений за допомогою методу сигнального придушення барабанних

перетинок через порушення нервових клітин, які утворюють приблизно 30 тисяч нервових закінчень слухового нерва. Але вивчення нервових закінчень здатне тільки прояснити формування простих синтетичних голосних. Перед дослідниками встало нове головний напрямок в області вивчення відтворення мови, пов'язане з інтеграцією всієї фізіології сприйняття людини. На даний момент з'являються деякі моделі явищ, що відбуваються у вусі, і не без підстав можна чекати подальшого поліпшення розуміння процесу розпізнавання мови через більш повного розуміння характеристик цього впливу.

Що стосується рівня артикуляторного контролю, першим рівнем є індивідуальний фонетичний сегмент, інакше кажучи, - фонема. У багатьох природних мовах їх приблизно 40. Але їх набір істотно різниться. Тому, наприклад, англійські голосні можуть бути носовими, навіть ненавмисно, в той час як у французькому носалізація голосних є фонетичним контрастом, і тому впливають на значення сказаного. У французькій мові носова коартикуляція домінує в голосних і суттєво впливає на сприйняття фонем і отже на головний зміст значення. Хоча всі говорять мають однаковий голосовий апарат, використання його різне. Так наприклад, використання кінчика язика або клацання, як у деяких африканських мовах. Ясно, що природа артикуляційних рухів має сильний вплив на метод відтворення мови. Ці обмеження завжди активно використовуються в практичних системах.

На наступному рівні лінгвістичної структури фонетичні сегменти згруповані в приголосні / голосні, а отже і у склади. Далі, в залежності від ролі фонетичного сегмента усередині цих складів їхня реалізація може бути сильно змінена. Так наприклад, початковий приголосний в складі може бути реалізований як абсолютно відмінний від кінцевої позиції. Згодні дуже міцно зв'язуються між собою, що знову ж впливає на наступні обмеження. Наприклад, в англійський якщо початкова група згодних складається з трьох фонем, перша фонема повинна бути / s /, наступної фонемой повинен бути невимовний приголосний, третьою або / r / або / l /, як наприклад, у слові /

scrape / або / split / . Ті, що говорять рідною мовою уникають цих обмежень або можуть активно їх використовувати під час процесу сприйняття. З вище наведених прикладів очевидно, що хоча й існують сильні обмеження, що впливають на слухача, але їх сила не є вирішальною під час проголошення промови. Тобто будь-яке моделювання процесу сприйняття може бути активним і може надати велику допомогу в розумінні головного сенсу.

Інший приклад, що показує необхідність застосування сфокусованого пошуку, може бути представлений у сприйнятті кінцевого приголосного. Серед багатьох ключових слів для розпізнавання кінцевого приголосного існує спектральна природа шуму, відтвореного при звільненні кінцевої перемички і переходу резонансу другий форманти в голосний, наступний за цією перемичкою. Багато дослідників вивчали ці впливи, і результати їх досліджень показали, що обмежуючий вплив обох вищеописаних характеристик на сприйняття варіюється природою наступного голосного, і отже, потужна стратегія розпізнавання повинна мати деякі знання про тверду позицію голосного перед кінцевим згодним перед тим, як буде зроблено саме розпізнавання кінцевого приголосного. Кінцеві приголосні дають яскравий приклад вельми цікавого комплексу фонетики, використовуваного для лінгвістичного забарвлення. Наприклад, при розгляді слів rapid і rabid виявляється 16 фонетичних відмінностей.

Крім сегментного та складового рівнів існують обмежені впливу через структури морфем, які є мінімальними синтаксичними одиницями мови, що включають в себе приставки, коріння, суфікси. Можна собі уявити, що це синтаксис на слоговом і на морфемном рівнях, також як і нормально розпізнаний синтаксис, що характеризується способом, в якому англійські слова об'єднуються у фрази і пропозиції. Можливо представити дані обмеження як наслідки розгляду граматики поза контекстом. У цьому виді обмежень багато "гучних" варіацій сегментів мови, які так само ставляться і до ієрархічним синтаксичним обмеженням.

Додаткові обмеження на природі входу нової лексики в мову можуть бути рівнем слова. Багато досліджень виявили, що характеристика слів при введенні розбивки на 5 жорстких класів фонетичних сегментів може бути скорочена до мінімуму, часто маючи єдине у своєму роді розпізнавання. Далі занадто підсилюється ефект порядку двох букв і фонетичних сегментів з тих пір як у вивченні англійських і французьких словників було виявлено, що понад 90% слів мали єдине значення і лише 0,5% мали 2 і більше альтернатив. На фонемному рівні було виявлено, що всі слова в англійському словнику з 20 тисяч слів мали одне значення через безладних фонемних пар. Цей приклад допомагає показати, що все ще існує обмежуючий вплив на лексичному рівні, яке ще не визначено в сучасних системах розпізнавання мови. Природно, що дослідження в цій області продовжуються.

Крім рівня слів синтаксис має додаткове обмежувальний вплив. Його вплив на послідовний порядок слів часто характеризується в системах фактором, який у свою чергу характеризує кількість можливих слів, які можуть слідувати за попереднім словом в процесі проголошення. Синтаксис також має обмежувальний вплив на просодичні елементи, такі як наголос, наприклад у випадку, коли наголос слів в incline і survey варіюється залежно від частини мови. Можливо для того, щоб охарактеризувати наголос у слові, потрібно взяти до уваги не тільки індивідуальне слово, але вищенаведені додаткові обмеження синтаксису. Далі, крім синтаксичного рівня обмеження домінують над семантикою, прагматикою і мовою, що погано усвідомлюється людьми, однак має дуже важливе значення для процесу розпізнавання.

Незважаючи на складність опису характеристики джерел різних обмежень, важливу роль відіграють сучасні системи впливу, які представлені всіма можливими варіантами проголошення звуків. Наприклад, система HARPI університету Carnegie-Mellon University є системою, в якій звуковідтворення описується як шлях через комплексну мережу. У цьому способі обмеження структури складу, слова і синтаксису пов'язані однією

структурою. Структура контролю, використовувана для пошуку, є адаптацією динамічної програмної техніки. Більш сильний підхід був запропонований моделями використання ланцюгів Маркова. Ці моделі використовувалися як єдина структура, де можливості можуть бути точно вивчені експериментальним шляхом. Закодовані уявлення спектральної трансформації відтворення мови використовуються для знаходження самого правильного шляху через мережу, і недавно були отримані дуже гарні результати. Дуже важливо підкреслити використання такого формально-структурного підходу, який сприяє автоматичності визначенню класів символів через структурування і параметризацію.

При іншому підході бази даних і пов'язані з ними процеси обробки використовуються структурою контролю. Цей підхід був вивчений системою HEARSAJ 2, яка була розроблена в інституті Carnegie-Mellon University, і системою HWIM (hear what I mean). У цих системах комплексна структура даних, яка містить всю інформацію про відтворення звуків, вивчається з точки зору конкретних обмежень. Але як вище зазначено, кожне з цих обмежень має особливу внутрішню модель, і повний аналіз не може бути проведений. Для проведення аналізу в цілому структура даних повинна мати взаємодію між різними процесами, а також засоби для інтеграції. Незважаючи на те, що структура включає в себе кілька дуже різних джерел знань і її внесок у розуміння мови дуже загальний, вона також має велику кількість ступенів свободи, які можуть бути використані для ретельного системного відтворення. На відміну від цього, техніка, заснована на ланцюгах Маркова, має математичну підтримку. Щоб мати можливість сфокусованого дослідження обмежень взаємодії та інтеграції в контексті, необхідно застосовувати обидві системи. Ті системи, які описують обмеження взаємодії, сфокусовані в чому на відтворенні знань, і вони відносно слабко контрольовані, а системам з математичною підтримкою, які в свою чергу мають чудову техніку для встановлення параметрів і оптимізації вивчення, не дістає використання комплексної структури даних,

необхідних для характеристики обмежень високого рівня, таких як синтаксис. Обидва напрямки зараз знаходяться в процесі розвитку.

На закінчення слід зробити акцент на вплив виробничої технології на ці системи. Технологія інтеграції не є великою проблемою для систем розпізнавання мови, навпаки, це є архітектурою цих систем, включаючи спосіб подання обмежень. Необхідно провести грандіозні експерименти і знайти нові способи, які необхідні для обмежувального впливу взаємодії.

У багатьох способах розпізнавання мов має типовий приклад стрімко розвивається класу високо інтегрованих комплексних систем, які повинні використати кращу комп'ютерну техніку і самі останні досягнення сучасного математичного забезпечення.

1.4.2 Огляд процесу розпізнавання мови

Процес розпізнавання мови може бути розділений на дві основні фази: оцифровка і декодування. На першій фазі вхідний аудіосигнал записується і розбивається на фрагменти. На фазі декодування отримана інформація аналізується на основі використання різних моделей і алгоритмів.

Алгоритми декодування можуть спиратися на зразки як цілих слів, так і окремих частин слів. Самою малою частиною слова є фонема, і будь-якій мові зазвичай доста¬точно 40-60 фонем, щоб описати вимови всіх слів.

Найбільш точними з точки зору розпізнавання є моделі, засновані на розпізнаванні слів. Однак вони можуть використовуватися лише в системах зі словарями невеличкого обсягу. Моделі, засновані на фонемній структурі, є набагато більш універсальними і значною мірою вирішують проблему обсягу словника.

В основу запропонованого підходу, і це є його головною відмінною рисою, покладено складне (ієрархічне і багатоярусне) подання простору акустико-фонетичних ознак і фонетичних одиниць, задіяних у процесі розпізнавання. Нижче в найзагальнішому вигляді описуються основні етапи процедури

формування такого подання і спосіб його використання безпосередньо в процесі розпізнавання.

Спочатку для мовного сигналу, який буде використовуватися в процесі навчання розпізнання, складається детальна сегментна транскрипція. Сегменти - Алофон фонем - описуються за допомогою двох основних класів стандартних фонетичних ознак - автономних і ієрархічних. Автономні ознаки (такі як назалізація, напруженість, лабиализация та ін.) Визначають «багатоярусний» характер уявлення акустико-фонетичного простору; вони володіють відносною незалежністю, оскільки їх наявність або відсутність ніяк не зумовлюється і не обмежується реалізацією інших ознак, і можуть використовуватися для опису фонетичних одиниць будь-якого рівня ієрархії. Ієрархічна ознака, навпаки, характеризується обов'язковою віднесеннестю з іншими класифікаційними ознаками. Так, наприклад, тільки приголосний звук може бути вибуховим, і тільки вибуховий, в свою чергу, може бути реалізований з носовим вибухом. В цілому для докладного фонетичного опису використовується приблизно 40 фонетичних ознак, автономних та ієрархічних. Процес сегментації та транскрибування мовного сигналу може виконуватися як вручну (експертом-фонетиста), так і в (напів-) автоматичному режимі (особливо у разі використання великих обсягів мовного матеріалу), з подальшою експертною корекцією.

Складається словник системи розпізнавання мови, при цьому кожне слово отримує транскрипційні уявлення. За основу приймається стандартне проголошення, яке визначається як вихідна транскрипція слова (ІТС). Надалі, в процесі розпізнавання, кожне слово буде співвідноситися з наявними в словнику ІТС.

Ієрархічна багатоярусна мережа (ІБМ), в яку організовані всі одиниці і мета-одиниці, являє собою однозв'язний багатоярусне дерево. На заданому рівні дерева кожна пара одиниць або мета-одиниць може бути або незалежною (автономною), або ієрархічно пов'язаною з вищим рівнем (ієрархічної). Таке структуроване подання дозволяє встановити міру

близькості для будь-якої заданої пари звуків. На кожному вузлі ІБМ є ієрархічна вагова функція (ІВФ), що описує відносну значимість додавання заперечення даної фонетичної ознаки для розпізнавання конкретної фонеми в даному слові. Вагові функції спочатку відображають статистичну інформацію про вплив чисто фонетичного рівня реалізації звуків (виведену на основі фонетичних модифікаційних правил) і мають поправки за рахунок загальнолінгвістичних факторів впливу - рівня (фонетичного) слова (позиція, контекст та ін.), Рівня лексикону (частотність слова, омонімія і т.п.), рівня виголошення (темп, стиль). Така структура признакового опису одиниць і мета-одиниць дозволяє досить просто і стандартизовано визначати міру подібності між собою різних варіантів аллофон реалізації слова з урахуванням багатьох лінгвістичних та екстралінгвістичних факторів.

У процесі навчання системи для кожної одиниці і мета-одиниці, включеної в ІБМ, створюється шаблон. Для простих одиниць такі шаблони створюється стандартним способом (наприклад, за допомогою СММ). Шаблони для мета-одиниць мають ієрархічну структуру і складаються з шаблонів простих одиниць, що входять до складу даної мета-одиниці. Також існує можливість створення додаткових шаблонів безпосередньо для мета-одиниць.

У процесі розпізнавання відбувається порівняння вхідних даних і наявних ІТС. При цьому з урахуванням значень ієрархічної вагової функції встановлюється міра подібності між знайденої поточної реалізацією розпізнаваного слова і ІБМ, побудованої за вихідної транскрипції порівнюваного слова. Чим вище значення міри близькості порівнюваних транскрипцій з урахуванням ІВФ, тим більш імовірним є розпізнаний варіант слова.

1.4.3 Аналіз основних проблем розпізнавання мови

На перший погляд все дуже просто: якщо друкований текст розпізнається, то і мову теж можно розпізнати, адже комп'ютеру все одно, що

обробляти - звук або малюнок. Здавалося б, треба тільки розділити отримане зображення або звуковий потік на повторювані стандартні образи, зіставити їх з використовуваними нами знаками і дати їм числові значення, за якими їх буде дізнаватися машина. Все б так і було, якби друкований текст і мова були дійсно аналогічними методами передачі інформації, але насправді вони дуже несхожі, і справа тут зовсім не в типі носія інформації. Людську мову скоріше можно порівняти з рукописним текстом, який, як і людська мова, дуже залежить від індивідуальних характеристик кожної людини. Почерк і тембр голосу унікальні і практично неповторні, і ці непередбачувані в кожному випадку параметри серйозно ускладнюють вичленення і систематизацію знакових образів.

Незважаючи на перераховані труднощі, системи розпізнавання мови удосконалюють досить швидко і поступово починають конкурувати з клавіатурним введенням. При цьому необхідно підкреслити, що поки комп’ютер ще дуже далекий від людини, що уловлює інтонації і настрій співрозмовника.

Зазвичай людина, вперше почувши про технології розпізнавання мови, покладається, що для надиктовування тексту системі, яка розпізнає мову, не потрібно особливих навичок, проте це не так. На відміну від клавіатурного, мовне введення крім основної інформації несе і дані про пол розмовляючого, про його вік, стан здоров'я, настрої, ставлення до переданої інформації, а також багато інших додаткових відомостей. Для розпізнавання мови абсолютну більшість цих даних - не допомога, а перешкода, тобто як для розмови по телефону, так і для надиктовування тексту системі розпізнавання від людини потрібно так чи інакше пристосовувати мову до цих пристроїв.

Сьогодні нам здається, що для того, щоб ефективно користуватися телефоном, не потрібні ніякі навички. Це пов'язано з тим, що навчання відбувається поволі: з раннього віку діти спостерігають, як дорослі розмовляють по телефону, і непомітно для себе набувають певні вміння.

Отже, мовне введення інформації пред'являє наступні вимоги:

• говорити слід не надто громко і не надто тихо. Краще за все - звичайним спокійним голосом.

Підвищені інтонації несуть багато побічних даних, внаслідок чого відсоток розпізнавання падає;

• вимовляти слова потрібно монотонно, але чітко. Не повинні проковтуватися закінчення, так як на відміну від людини комп'ютер поки не може стежити за контекстом закінчення;

• чим менше сторонніх шумів, тим краще;

• треба намагатися підтримувати постійну відстань до мікрофону;

• в мікрофон не повинен потрапляти подих, тому мікрофон потрібно тримати не прямо навпроти рота, а приблизно на сантиметр вправо і на сантиметр нижче.

Погане апаратне забезпечення теж є джерелом проблем для розпізнавання мови, тому якісний мікрофон і хороша звукова плата з вбудованим фільтром шумів можуть значно поліпшити роботу системи розпізнавання мови. Але коли всі труднощі вирішені, перед користувачем програми розпізнавання звукової мови відкриються абсолютно нові можливості. По-перше, швидкість введення будь-якого тексту збільшується в кілька разів порівняно з введенням з клавіатури; при цьому витрати необхідних зусиль зменшуються, а навчання взагалі не потрібно, так як говорити ми всі вміємо. По-друге, така програма дозволяє управляти іншими додатками і операційною системою в цілому за допомогою голосових команд, що дуже полегшує і прискорює роботу за комп'ютером.

Наша країна відноситься до розробників систем розпізнавання української мови. Ще один сюрприз - діалекти і говори: необхідно також враховувати відмінності у вимові в різних регіонах України. Як правило, подібні проблеми вирішуються за допомогою попереднього налаштування. А технології, розроблені спеціалістами фірми VoiceLock, дозволяють налаштовувати програму всього за декілька хвилин. Головна проблема, що виникає при розробці САРР (системи автоматичного розпізнавання мови),

полягає в варіативній вимові одного і того ж слова як різними людьми, так і однією і тією же людиною в різних ситуаціях. Людину це не збентежить, а ось комп'ютер - може. Крім того, на вхідний сигнал впливає велика кількість факторів, такі як навколишній шум, віддзеркалення, відлуння і перешкоди в каналі. Ускладнюється це й тим, що шум і спотворення заздалегідь невідомі, тобто система не може бути підстроєна під них до початку роботи.

Однак більш ніж піввікова робота над різними САРР дала свої плоди. Практично будь-яка сучасна система може працювати в декількох режимах. По-перше, вона може бути залежною або незалежною від диктора. Залежна від диктора система вимагає спеціального навчання під конкретного користувача, щоб точно розпізнавати те, що він говорить. Для навчання системи користувачеві треба вимовити кілька певних слів або фраз, які система проаналізує і запам'ятає результати. Цей режим звичайно використовується в системах диктування, коли з системі працює один користувач.

Дикторонезалежна система може бути використана будь-яким користувачем без навчальної процедури. Цей режим звичайно застосовується там, де процедура навчання неможлива, наприклад у телефонних додатках. Точність розпізнавання дікторозалежної системи вище, ніж у дикторонезалежної. Однак незалежна від диктора система зручніша у використанні, наприклад вона може працювати з необмеженим колом користувачів і не вимагає навчання.

По-друге, системи діляться на працюючі тільки з ізольованими командами і на що здатні розпізнавати зв'язну мову. Розпізнавання мови є значно складнішим завданням, ніж розпізнавання окремо вимовлених слів. Наприклад, при переході від розпізнавання ізольованих слів до розпізнавання мови при словнику в 1000 слів відсоток помилок збільшується з 3,1 до 8,7, крім того, для обробки мови потрібно в три рази більше часу.

Режим використання вимовлених команд найбільш простий і найменш ресурсномісткий. При роботі в цьому режимі після кожного слова користувач робить паузу, тобто чітко відзначає начало та кінець слів. Системі не потрібно

самій шукати початок і кінець слова у фразі. Потім система порівнює розпізнане слово із зразками в словнику, і найбільш ймовірна модель приймається системою. Цей тип розпізнавання широко використовується в телефонії замість звичайних DTMF-методов1.

Режим злитого виголошення більш натуральний і близький користувачеві. При цьому передбачається, що система сама розрізнить кордон слів у фразі. Однак цей режим вимагає більше системних ресурсів і пам'яті, а точність розпізнавання нижче, ніж у попередньому режимі. Чому це так? Причин кілька. По-перше, при злитій мові вимовляння слів менш акуратне, ніж в «режимі PIN-коду», тобто коли кожне слово вимовляється окремо. По-друге, швидкість мовлення навіть у одній людини різне. Він може задуматися, засумніватися, забути слово. У разговорній мові часто зустрічаються слова-паразити: «ну», «а», «ось». Крім того, межі слів часто змазуються, вимовляються нечітко, що ускладнює роботу системи.

Додаткові варіації в мові виникають також через довільних інтонацій, наголосів, нестрогої структури фраз, пауз, повторів і т.д.

На стику злитого і роздільного проголошення слів виник режим пошуку ключових слів. У цьому режимі САРР знаходиться заздалегідь певне слово або група слів в загальному потоці мови. Де це може бути використане? Наприклад, в підслуховуючих пристроях, які вмикаються і починають запис при появі в мові певних слів, або в електронних довідниках. Отримавши запит у довільній формі, система виділяє смислові слова і, розпізнає їх, видає необхідну інформацію.

Розмір використовуваного словника - важлива складова САРР. Очевидно, що чим більше словник, тим вища ймовірність того, що система помилиться. У багатьох сучасних системах є можливість або доповнювати словарі в світі необхідності новими словами, або довантажувати нові словарі.

1.5 Постановка задачі

В результаті аналізу існуючих систем мовного управління та функцій необхідних для систем телефонії було встановлено, що мовний інтерфейс інтелектуальної системи телефонії повинен виконувати наступні функції:

- розпізнавання голосу
- голосове редагування контактних даних
- голосове введення тексту та номеру при відправці SMS
- голосове введення номера для дзвінків
- перехід між вікнами додатку завдяки голосовим командам.

У даній роботі вирішується завдання створення програмного додатку з функціями набор номеру, відправки SMS, голосового редагування контактних даних додатку, голосового введенню тексту та номеру SMS, голосове введення номеру для дзвінків, перехід між функціями додатку завдяки голосовим командам. У зв'язку з поширенням мобільних пристроїв телефонії під керуванням операційної системи Android, розроблюваний додаток буде реалізований на системі Android.

2 МЕТОДИ ТА ЗАСОБИ РОЗРОБКИ ПРОГРАМНИХ ДОДАТКІВ ДЛЯ ОПЕРАЦІЙНОЇ СИСТЕМИ ANDROID

2.1 Ключові та загальні особливості Android

Протягом останніх років стався важливий перерозподіл складу апаратних платформ для кінцевих користувачів. Доля ринку мобільних пристроїв та планшетних комп'ютерів уперше перевищила кількість персональних комп'ютерів. Наразі три операційні системи, iOS, Android та Windowsmobile конкурують на ринку. Кількість мобільних додатків у кожній еко-системі для мобільних та планшетних комп'ютерів нелінійно зростає кожного року, перевищив мільйон програм в Андроід [2], 617 тисяч програм для iOS, та 156 тисяч для WindowsMobile [3].

Android це унікальна операційна система. Розробник додатків повинен знати її особливості та нюанси для отримання хорошого результату. Існують деякі труднощі, які потрібно враховувати при розробці. Перерахую їх коротко:

Додаток вимагає для установки в два рази (або навіть в чотири) більше місця, ніж оригінальний розмір програми;

Швидкість роботи з файлами на вбудованої флеш-карті падає в десятки разів при зменшенні вільного місця;

Кожен процес може використовувати до 16 Мб (іноді 24 Мб) оперативної пам'яті, заснований на Linux. Між додатком і ядром лежить шар API і шар бібліотек на нативному коді. Додаток виконується на віртуальній машині Java (Dalvik Virtual Machine).

В Android можна запускати багато додатків. Але одне з них є головним і займає екран. Від поточного додатка можна перейти до попереднього або запустити нове. Це схоже на браузер з історією переглядів.

Кожен екран користувача інтерфейсу представлений класом Activity в коді. Різні Activity містяться в процесах. Activity може навіть жити довше процесу. Activity може бути призупинена і запущена знову зі збереженням

всієї потрібної інформації використовує спеціальний механізм опису дій заснований на Intent. Коли потрібно виконати дію (зробити дзвінок, надіслати листа, показати вікно), викликається Intent.

Також Android містить сервіси подібні демонам в Linux для виконання потрібних дій у фоновому режимі (наприклад, програвання музики). Для обміну даними між додатками використовуються Content providers (провайдери вмісту).

Для даної роботи були використані провайдер даних про місцезнаходження і положення в просторі користувальницького пристрою.

Загальна схема роботи програми Android. Програми для Android в своїй роботі використовує вікна (аналогічно Windows), проте в даній системі вищевказані вікна носять іншу назву - Activity. Як і в Windows, кожне вікно має свій життєвий цикл і свої особливості. При створенні нового вікна викликається метод onCreate (), при розробці даний метод перевизначається і в ньому відбувається ініціалізація додатка і його компонентів. Далі викликаються методи onStart () і onResume (). Обидва методи викликаються перед відображенням вікна при його створенні, або відновленні (при перемиканні з іншої програми, при розгортанні згорнутого додатки і тп). При згортанні викликаються методи onPause () і onStop (). При закритті програми і вікна викликається onDestory (), у цьому методі можна зберегти дані і параметри. Повний опис та послідовність виклику методів можна знайти на офіційному сайті. Загальна схема життєвого циклу програми для Android (рис. 2.1)

Методика розробки додатків за допомогою модульних тестів полягає, в першу чергу, у створенні тесту, а потім реалізацію відповідного коду програми для виконання функцій, які вимагає створений тест. TestDrivenDesign, методологія створення програмного забезпечення за допомогою тестів досить широко підтримується для розробки мобільних додатків у Android. Стандартні середовища розробки Android додатків має у

своєму складі потужну систему створення юніт (модульних) тестів та інфраструктуру для їх запуску і аналізу результатів виконання.

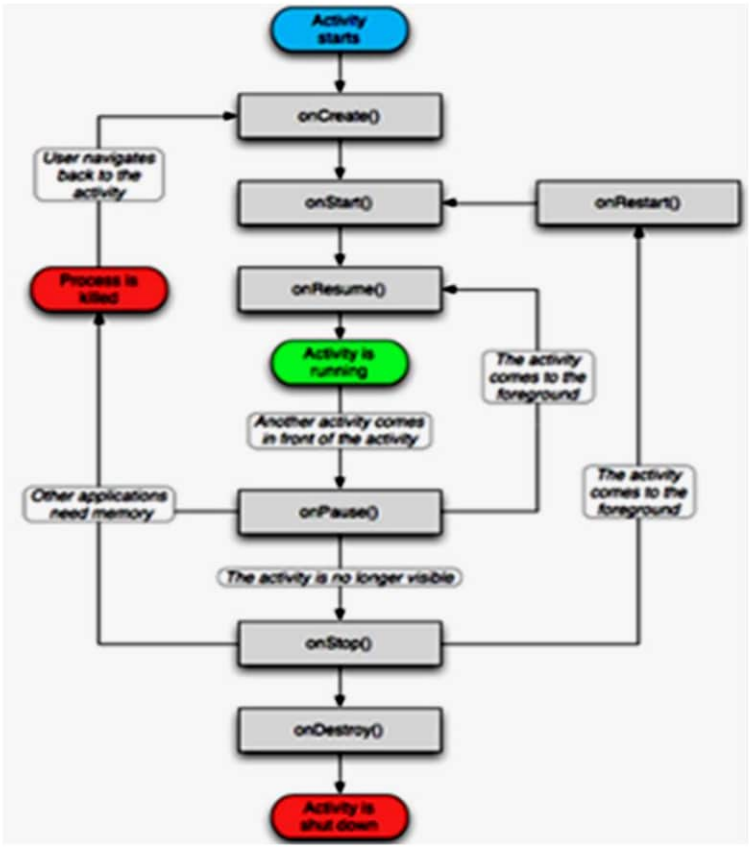

Рисунок 2.1 – Життєвий цикл програми для системи під управлінням Android

Діаграма станів для методології розробки програмного забезпечення за допомогою юніт-тестів зображено на (рис. 2.2). Наприклад, середовище розробки Eclipse надає користувачам можливість виконувати юніт тести на мобільних пристроях та на емуляторі. Для виконання тестів створюється окремий проект, що зручно дозволяє залишати основний проект без суттєвих змін. Для більш ефективної функціональної перевірки роботи мобільного додатку можливо використовувати скриптиавтоматичного тестування сценаріїв користувача, що дозволяє проводити серію тестів через вплив на мобільний додаток з боку графічного інтерфейсу користувача. Такий підхід надає можливість більш гнучко будувати набір регресійних тестів для верифікації функціональної частини мобільних додатків і обирати найбільш ефективний метод створення тестів: через юніт-тестування, та

автоматизоване тестування сценаріїв користувача (рис. 2.3).

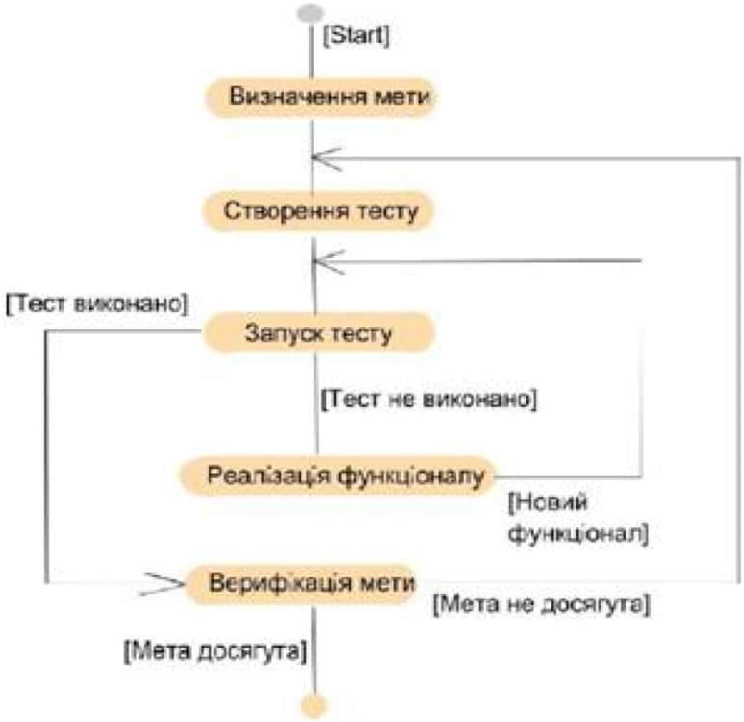

Рисунок 2.2 – Діаграма станів розробки програмного забезпечення за допомогою юніт-тестів

Рисунок 2.3 – Одночасне використання автоматизованого тестування за сценаріями користувача та модульного тестування

Для тестування сценаріїв користувача зручно використовувати системи, подібні до Robotium [4]. Основними перевагами розробки за допомогою тестів є:

- стрункa архітектура проекту
- гарний відсоток покриття тестами основного функціоналу мобільного додатку
- інкрементальний процес створення регресійних тестів
- якісний аналіз сценаріїв користувача та їх покриття автоматизованими тестами.

2.1.1 Системи розповсюдження та моніторингу роботи мобільного додатку.

Оскільки фрагментація (різниця у версіях операційної системи) Андроід має великий вплив на стабільну роботу мобільного додатку, перевірка повного функціоналу на цільових операційних системах вимагає великої кількості ресурсів та часу. В такому випадку використовується система моніторингу роботи мобільного додатку з інфраструктури Google. Якщо виникає необхідність використання мобільного додатку до публікації в системі GooglePlay, збір інформації про можливі помилки в роботі мобільного додатку на етапі тестування можливо використовувати системи розповсюдження та моніторингу мобільних додатків, таких як TestFlight [5], BetaFamily [6], систему Бета-тестування та поетапного впровадження від Google [7] з можливістю вибору тестувальної групи користувачів (рисунок 2.4). Серед систем автоматичного звіту про помилки в роботі програми на етапі тестування можна виділити TestFlightSDK, Acra [8] – вони дозволяють отримувати стек від помилок під час збою програми до розміщення мобільного додатку у GooglePlay.

Рисунок 2.4 – Технологічний маршрут етапів тестування мобільних додатків

2.1.2 Використання хмарних систем з мобільними програмними додатками в Android

За сучасним станом, хмарні системи отримали широке вживання у якості серверної частини для збереження даних у мобільних технологіях. Найчастіше використовується архітектура клієнт-сервер, де мобільна частина виконує роль терміналу доступу та обробки даних. Протокол обміну найчастіше обирають серед HTTPREST [9] з використанням JSON стандарту RFC4627 [10] представлення даних. Вимога зменшення обсягу даних, що передаються за протоколами HTTP обумовлена можливими повільними з'єднаннями мобільного Інтернету. Загальна схема взаємодії мобільного додатку з хмарним серверним середовищем зображена на (рисунок 2.5). Архітектура мобільного додатку найчастіше відповідає стандартному шаблону проектування Model-Vew-Controller. Серед поширених хмарних сервісів для організації серверної частини можливо використовувати GoogleApplicationEngine [11].

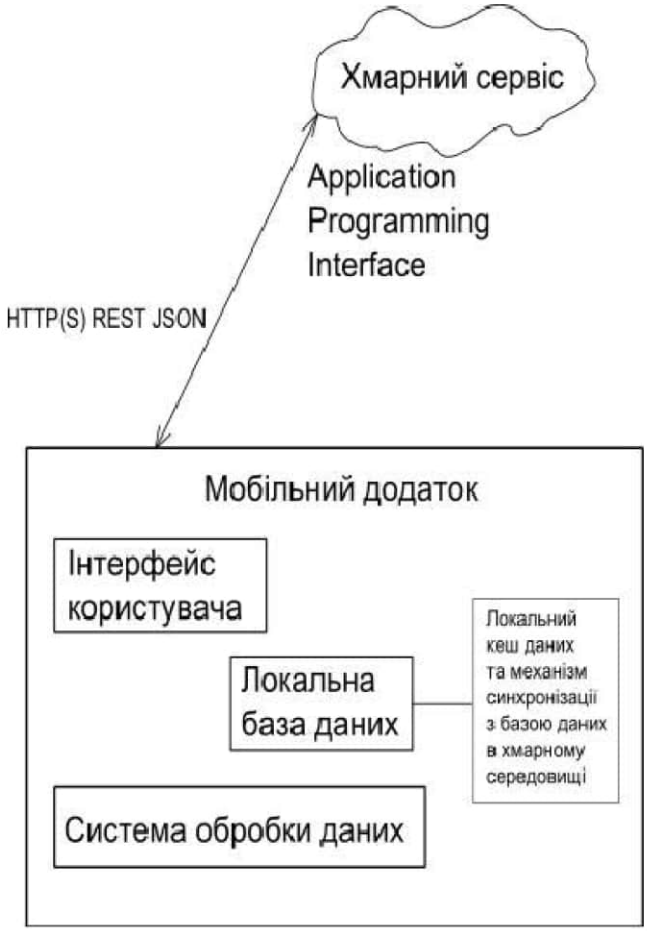

Рисунок 2.5 – Схема взаємодії мобільного додатку з хмарним середовищем

Даний хмарний сервіс відноситься до класу PAAS, що дозволяє використовувати гнучкий фреймворк для серверних частини, та створювати код у хмарному середовищі за допомогою мов програмування Java, Python, PHP, Go. Хмарне середовище Google Application Engine надає бібліотеку розробки для розробки серверної частини, підтримує сучасний інструментарій IDEE clipse, хмарний сервіс збереження даних, швидку реалізацію REST протоколів, безкоштовну роботу серверної частини для невеликої кількості запитів до бази даних.

2.2 Огляд систем розробки додатків для OS Android

Особливості розробки програмних додатків для мобільних пристроїв. Процес розробки програмного забезпечення для мобільних пристроїв вимагає

підтримувати певні обмеження що до специфіки роботи додатків у мобільних операційних системах. Особливості роботи мобільних додатків стосуються перш за все:

- на витрати живлення акумуляторної батареї мобільного пристрою
- обмеження на кількість даних, що передаються через мережу Інтернет
- зменшену швидкість передачі пакетів в мобільному Інтернет з'єднанні
- можливість втрати пакетів при передачі в мобільному Інтернет з'єднанні
- кількість обмінів даними з зовнішніми пристроями на Bluetooth
- безпеку даних користувача
- значну фрагментацію версій операційних систем та фреймворків
- велику різноманітність розмірів екранів мобільних пристроїв
- обмеження запитів до системи визначення місцезнаходження абонента
- обмеження на розмір файлу додатку порівняно невеликий ліміт оперативної пам'яті мобільного пристрою
- порівняно обмежену швидкість передачі даних в мобільному Інтернет з'єднанні.

2.2.1 Візуальне середовище розробки додатків AppInventor 2

Візуальне середовище розробки мобільних додатків AppInventor (рисунок 2.6) розроблений в Google Labs, однак після закриття лабораторії було передано в MIT (Массачусетський технологічний інститут). MIT запустив першу публічну бета-версію проекту в березні 2011 року.

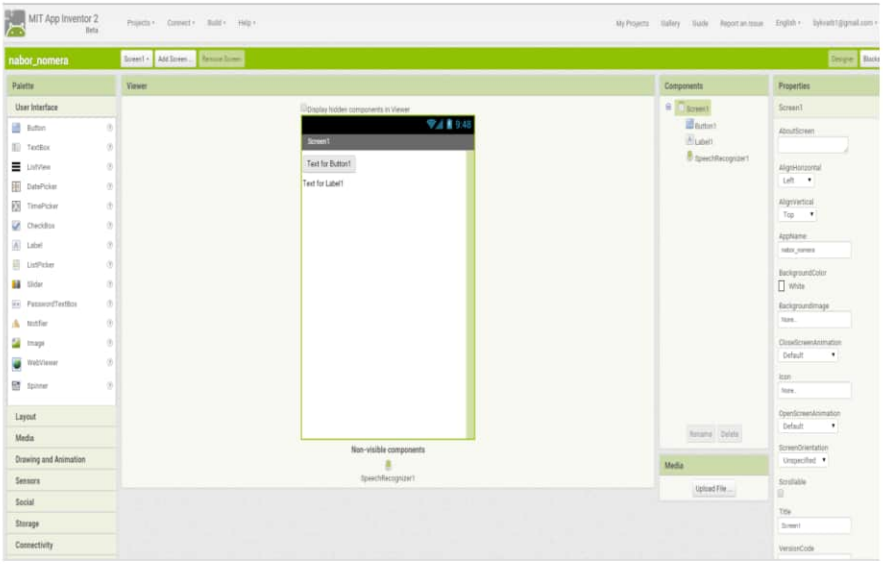

Рисунок 2.6 – Візуальне середовище розробки мобільних додатків AppInventor

Даний програмний продукт дозволяє створювати додатки для мобільної платформи Android. Перед початком розробки необхідно налаштувати комп'ютер і телефон для роботи в App Inventor.

Для створення додатків в App Inventor використовується візуальна мова програмування, додатки будуються об'єднанням стандартних елементів, наприклад Label (мітка), який відображає текст на екрані, Button (Кнопка), Canvas, в якому можно розташовувати зображення чи анімацію; accelerometer (motion) sensor, який визначає, коли телефон трясеться або перевертається. Також є елементи за допомогою яких можна відправити повідомлення, програти відео та багато інших. Ось це і є та сама фішка від Google! Натискаємо на кнопочку "Open the Blocks Editor" і потрапляємо в обробник, який запуститься у Вас на комп'ютері. Тут, використовуючи все той же Drag & Drop, створюється код! Є об'єкти, класи, властивості, if - и. Але всі ці елементи являють собою блоки, які як в конструкторі LEGO можна з'єднувати між собою, отримуючи цілісний код, але, як і в дитячому конструкторі, деякі деталі один до одного не підходять. Ось така от інтерпретація об'єктно-орієнтованого програмування.

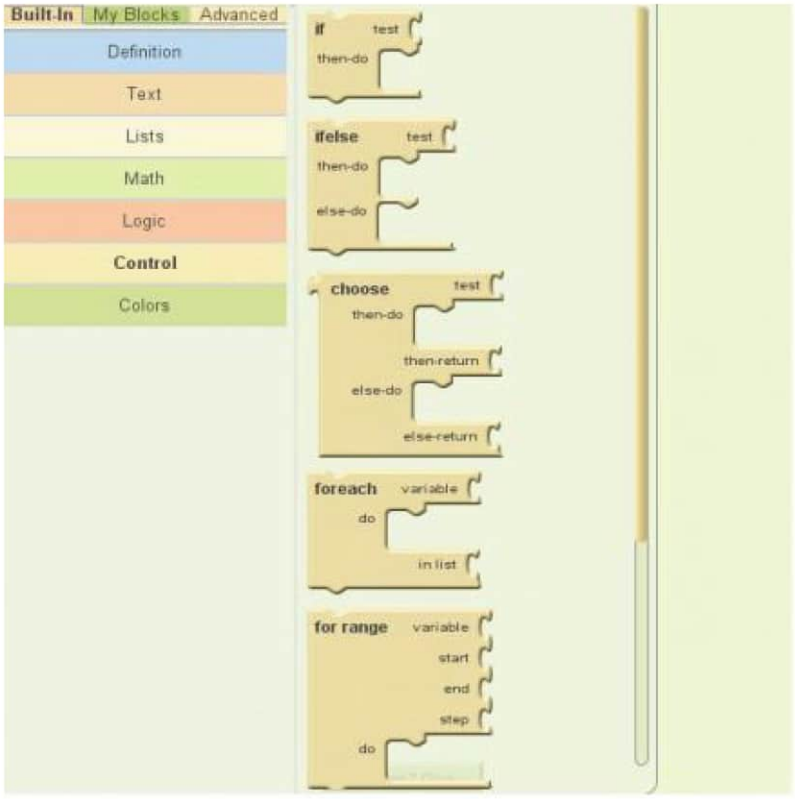

Рисунок 2.7 – Редактор блоків у конструкторі додатків AppInventor

Після опису подій підключаємо телефон або запускаємо акомулятор. Потім додаток можно завантажити на комп'ютер, натиснувши відповідну кнопку в Chrome і завантажити до телефона.

Великими плюсами цього середовища є безкоштовність розробки і не обов'язково знання Java, але маючи досвід у програмуванні на інших мовах, стає можливим написати добре функціонуючий додаток для OS Android та бачення відразу результатів роботи в App Inventor 2 завдяки тестуванню розробленої прогрми в кожну мить написання додатка на телефоні підключеному через Wi-Fi.

2.2.2 Середовище розробки Android Studio

Основні особливості - реалізована можливість верстки в реальному часі, доступно безліч варіантів розмірів і дозволів екранів. Присутній розділ довідки, вбудовані інструменти поліпшення якості додатків і монетизації - є

інструменти для відстеження ефективності рекламних оголошень. Додано засіб взаємодії з бета-тестерами. І багато іншого.

Процес установки простий - завантажуємо програму і починаємо процес установки. Після закінчення установки, якщо стоїть відповідний прапорець, чекаємо запуску студії. Йдемо в папку, в яку встановлювали програму і пробуємо запустити файл studio.exe.

Потрібно прописати змінну середовища оточення. У провіднику клацаємо правою кнопкою миші на значку Computer і вибираємо Properties, щоб відкрити компонент панелі управління System. В даному вікні вибираємо пункт Advanced system settings - відкриється діалогове вікно, в якому потрібно клацнути на кнопці Environment Variables .... Відкриється ще одне діалогове вікно.

Розглянемо по кроках.

Рисунок 2.8 – Вікно вибору робіт з Android Studio

Вибираємо новий проект і заповнюємо необхідні поля. Треба вказати додатки в першому полі. У другому слід вказати домен компанії. За замовчуванням студія може підставити ім'я користувача на комп'ютері. Краще відразу змінити на зручне назву, яка студія запам'ятає і буде підставляти в нових проектах. Поле Company Domain використовується для формування пакета (Package name) у перевернутому вигляді, як це прийнято

в Java. Якщо ви не змінили ім'я домену, але ім'я пакету вас не влаштовує, то можете натиснути на кнопку Edit і відредагувати ім'я пакета.

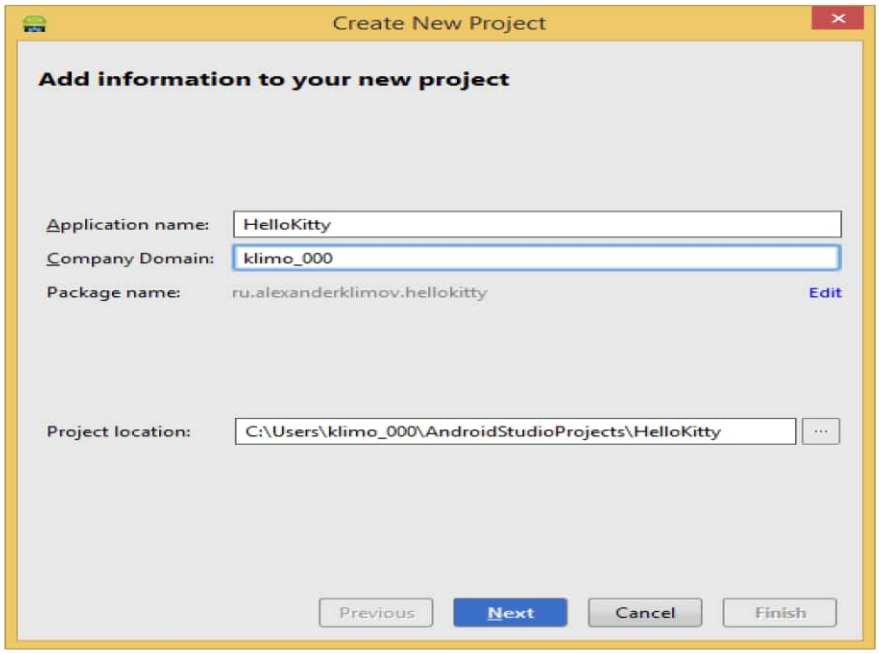

Рисунок 2.9 - Вікно введення назви нового проекту в Android Studio

Далі вибирається мінімальний рівень для програми.

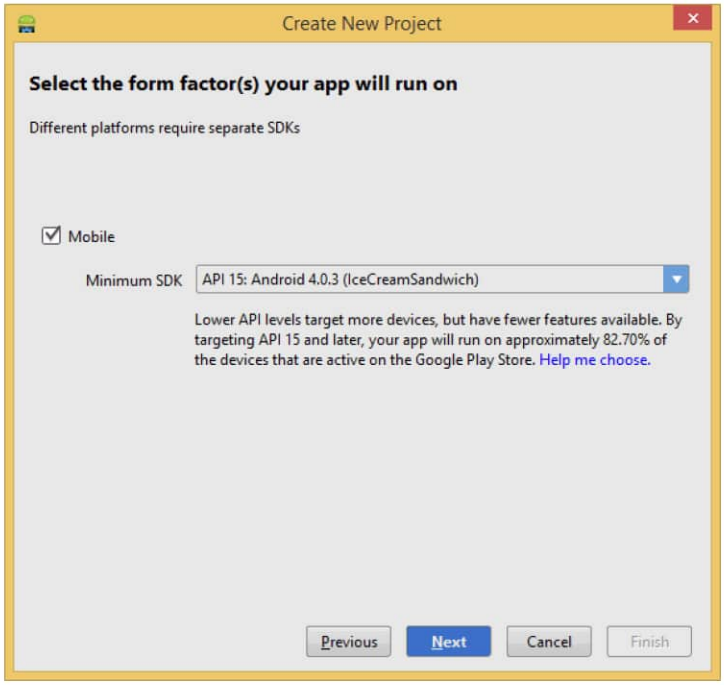

Рисунок 2.10 – Вікно вибору мінімальної версії Android

Далі вибирається тип програми. Для новачків слід вибирати варіант Blank Activity.

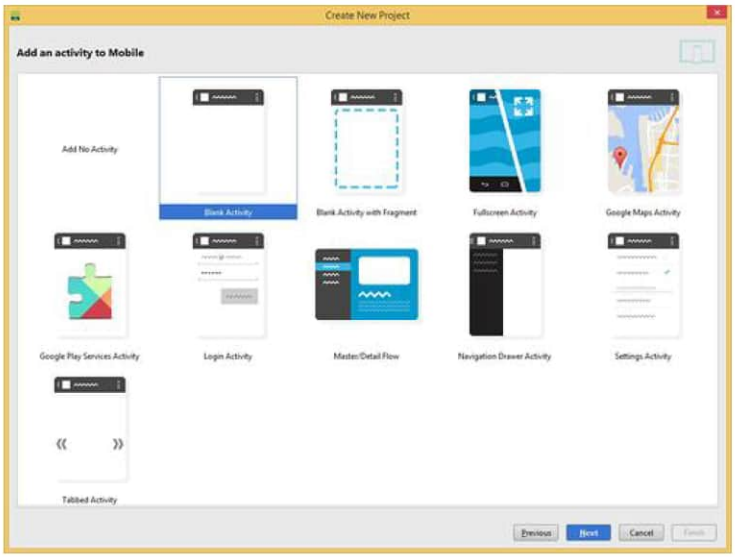

Рисунок 2.11 – Вікно вибору типу програми

Останній крок - вибираємо імена для активності, розмітки і заголовка. Тут теж відбулися зміни. Раніше використовувався варіант зі словом "main" - MainActivity, activity_main.

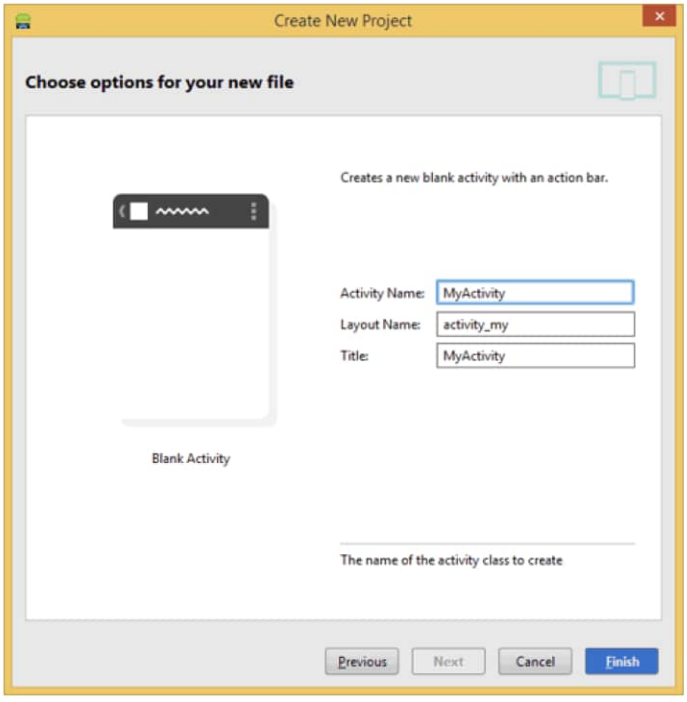

Рисунок 2.12 – Вікно вибору для активності, розмітки та заголовка

Далі на екрані виник індикатор прогресу, який щось завантажував з використанням Gradle.

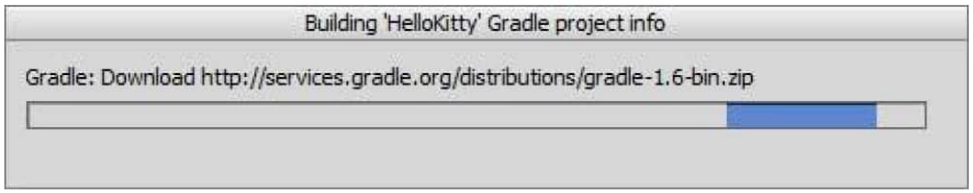

Рисунок 2.13 – Індикатор прогресу

Середовище Android Studio є безкоштовним середовищем для створення додатків але щоб у ньому працювати теба мати добрі знання прогрмування HTM та Java.

2.2.3 Сервіс розробки додатків BuildAnApp

Сервіс BuildAnApp дозволяє генерувати додатки для платформ Google Android, BlackBerry, Apple iPhone, Windows Mobile без написання коду і досвіду програмування. Програми, створені за допомогою сервісу BuildAnApp, можуть виконуватися з використанням мобільного веб-браузера, проте представляють собою native-код обраної платформи.

Користувачам цього сервісу пропонуються набір шаблонів, малюнків для фону, піктограм для кнопок і вкладок, інструменти для завантаження власних зображень, засоби інтеграції зі списком контактів смартфона та інструменти автоматичного оновлення програми (рисунки 2.14 – 2.15).

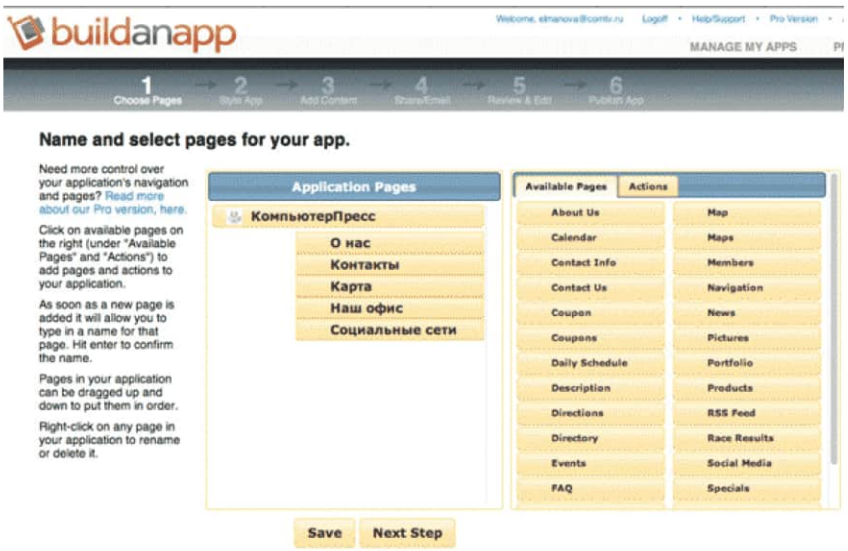

Рисунок 2.14 – Формування набору екранів додатки в сервісі BuildAnApp

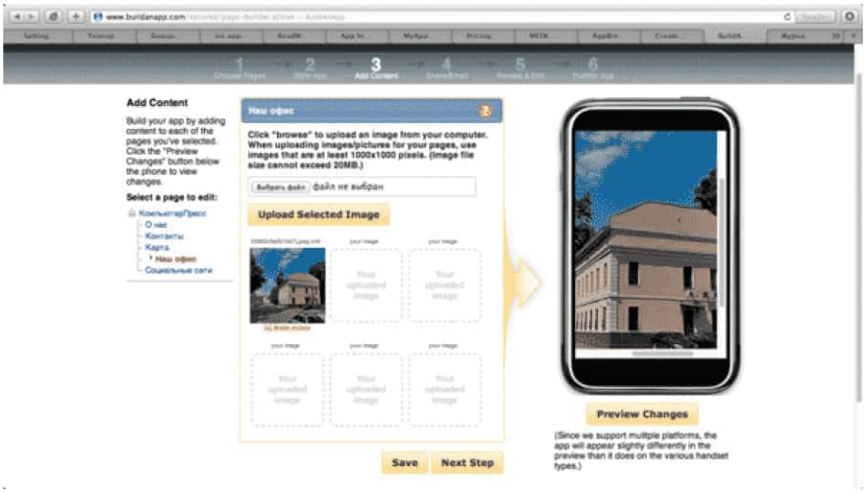

Рисунок 2.15 – Формування екрану для відображення графіки в сервісі BuildAnApp

В якості цікавих особливостей можна відзначити підтримку списків розсилки потенційних користувачів програми.

Створення і модифікація додатків за допомогою сервісу BuildAnApp - платна послуга, за окрему плату додаток публікується в онлайнових магазинах додатків, таких як Apple App Store. Публікація в онлайнових магазинах здійснюється від імені користувача сервісу, що у разі App Store передбачає придбання користувачам відповідної підписки Apple Developer Account та надання відомостей про неї власникам сервісу BuildAnApp.

2.3 Розпізнавання голосу від Google

Інструменти, що перетворюють мову у текст «speech-to-text» протягом довгого часу є досить популярними. Звичайно, на зорі свого існування, такі програми не відрізнялися якістю, багато в чому через тонкощі мови, але сучасні програми для розпізнавання мови значно перевершують своїх родоначальників.

Робота з пошуковим інтерфейсом, керованим голосом тільки на перший погляд може здатися простою. По-перше, більшість пошукових запитів короткі, близько 2-3 слів, максимум 6. По-друге, важливо, щоб пошуковий запит відповідав обмеженому словниковому запасу системи. Франц і Мильх у своїй праці відзначають, вокабуляр в 100 000 слів може покрити лише 80 % запитів.

Але ці дві проблеми відступають на задній план, якщо згадати, що голосовий пошуковий інтерфейс повинен працювати в реальному часі. Користувачі звикли отримувати результати протягом часток секунди, навряд чи їх влаштує тривале очікування. Ще однією проблемою може стати інтерпретація запиту користувача, викликана незвичайним акцентом або присутністю сторонніх шумів. З текстом працювати значно легше, навіть неправильне написання передбачає обмежене число можливих варіантів, з якими користувач взаємодіє.

Втілення нових голосових технологій у мобільні системи, зокрема у смартфони було цілком передбачувано. Адже який не був хороший мультитач, в багатьох випадках голосове управління смартфоном краще і зручніше, ніж дотики до сенсорного екрану.

Обробивши запис голосового повідомлення, смартфон відправляє ці звукові дані на сервери Google. Саме тут, в «хмарах», а не на смартфоні, відбудеться обробка і розпізнавання звукового фрагмента, після чого розпізнаний запит у вигляді тексту повертається в браузер, і на екрані з'являється звичайний вивід пошукової системи.

Для точного розпізнавання голосу на серверах Google зібрані і постійно збираються звукові фрагменти, які, за заявою розробників сервісу, дозволяють створювати моделі мови, що забезпечують коректну роботу сервісу і точне розпізнавання голосових фрагментів. Для кожної мови на серверах Google створюються словники, що містять вже біля 230 мільярдів розпізнаних слів. Природно, ці багатомовні словники постійно поповнюються.

Після установки на смартфон системного додатка Google Voice Search значок мікрофона з'явиться в пошуку Google Maps. Також цей значок може з'явитися на клавіатурі, що дозволяє наговорити SMS або який-небудь інший текст. На основі голосового пошуку з'явилися вже й додаткові програми, які дозволяють здійснювати пошук і по інших пошукових систем, відмінним від Google.

Google Voice Search - це хмарний веб-сервіс перетворення звуку в текст. Це не просто окремий модуль для голосового пошуку в Інтернеті. Google Voice Search є голосовим API для інших прикладних програм, які потребують перетворення мови в текст, наприклад:

- Голосовий блокнот Speak2Send Advance Voice NotePad. Програма дозволяє надиктувати послідовно, речення за реченням, звичайний текст.

- Додаток Voice Command. Ця програма встановлюється у вигляді віджета на робочий стіл і приймає голосові команди від користувача. Voice Command має кілька режимів роботи з голосовим пошуком. Перший режим - це копіювання в буфер обміну тексту, отриманого після розпізнання в Voice Search. Після цього текст можно копіпастом вставити в будь-яку форму введення тексту. Другий режим - це пошук в контактах. Третій - пошук в Інтернеті. І четвертий режим - це пошук на географічних картах Google.

3 РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ

3.1 Характеристики компонентів, які використовувались у розробці додатку

Speech Recognizer (розпізнавання мови) компонент використовується для прослуховування користувача і конвертування мови звуків в текст, використовуючи функції розпізнавання мови Android.

Компонент Button (кнопка) з можливістю виявлення клацань. Багато аспектів його зовнішнього вигляду може бути змінено, а також він є інтерактивним (Enabled), може бути змінений в конструкторі або в редакторі блоків.

TextBox (текстове вікно) початкове або введене користувачем значення тексту. Якщо текст порожній, можна використовувати підказки щоб надати користувачеві пропозицію щось надрукувати. Властивість Miltiline визначає можливість введення тексту більш ніж на один рядок. Для одного текстового рядка поля, клавіатура автоматично зникне, коли користувач натискає клавішу Готово. Щоб закрити клавіатуру для багаторядкових текстових полів, програма має використовувати метод HideKeyboard.

Невидимий компонент TiniDB (посилання на документ) виконує посилання на документи, які зберігають данні для програми. Програми створені з App Inventor 2 виконуються кожен раз, коли вони виконуються. Це означає, що якщо додаток встановлює значення змінної, а потім користувач виходить з програми, значення цієї змінної не буде запам'ятовано. На відміну від посилань на документи стійких сховищ даних для додатків. Дані, що зберігаються в TinyDB будуть доступні кожен раз коли додаток запускається. Прикладом може бути гра, яка зберігає високий бал і зберігає його що разу.

Компонент ContactPicker (кнопка відкриття контактів телефона) при натисканні на яку, відображається список контактів, щоб вибрати серед них.

Після того як користувач зробив вибір, наступні властивості будуть встановлені до інформації вибраного контакту:

- ContactName: ім'я контакту
- EmailAddress: основна адреса електронної пошти контакту
- EmailAddressList: список про адреси контактів електронної пошти
- PhoneNumberList: список телефонних номерів контактів (на більш пізніх версіях Android)

Компонент PhoneCall який робить телефонний дзвінок на номер, вказаний в PhoneNumber, що може бути встановлене в конструкторі блоків Editor. Компонент має MakePhoneCall метод, що дозволяє програмі запускати телефонний дзвінок. Часто, цей компонент використовується з ContactPicker, який дозволяє користувачеві вибрати контакт з тих, що зберігаються на телефоні, і встановлюються у PhoneNumber.

3.2 Розроблення конструкцій у редакторі блоків App Invventor 2

Створюємо кнопки та вказуємо їх дії при натисканні (рисунок 3.1).

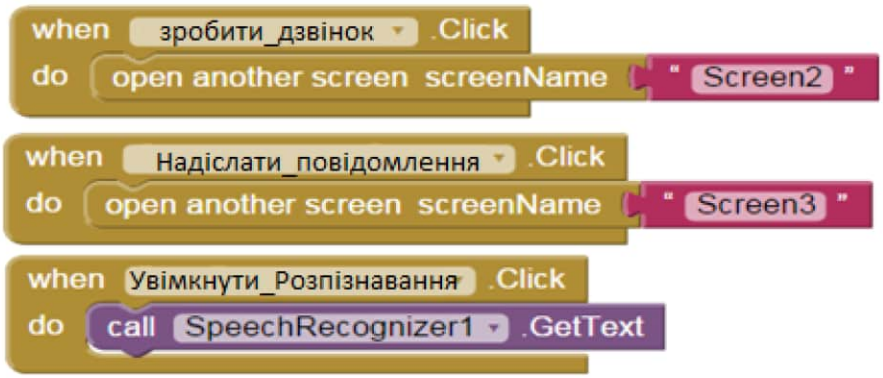

Рисунок 3.1 – Функції кнопок

Для розпізнавання мови додаємо SpeechRecognizer1. AfterGettingText resultat та вказуємо куди зберігати результат розпізнавання, та вказуємо які функції виконувати, якщо результат дорівнює одним із вказаних варіантів (рисунок 3.2).

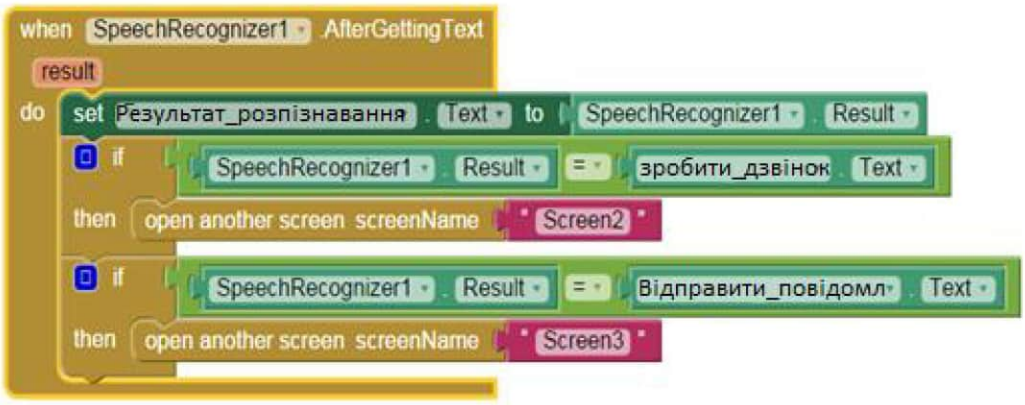

Рисунок 3.2 – Блок SpeechRecognizer1

Ініціалізуємо розпізнавання мови при запуску програми (рис. 3.3).

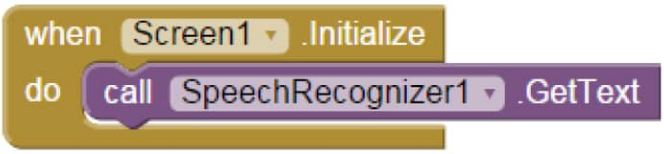

Рисунок 3.3 – Ініціалізація розпізнавання

В Screen2 створюємо глобальну змінну (рис. 3.4).

Рисунок 3.4 – Вказуємо глобальну змінну name

Описуємо дії які виконуються при відкритті вікна дзвінків (рис 3.5).

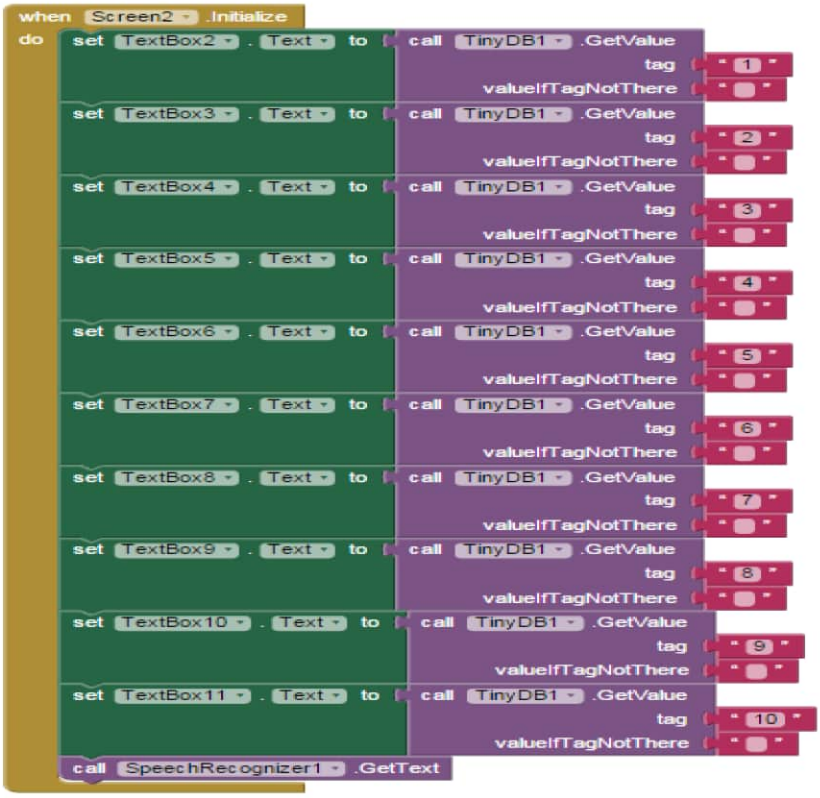

Рисунок 3.5 – Опис ініціалізації при відкриття вікна дзвінків

Організуємо запис контактів в пам'ять телефону (рис 3.6).

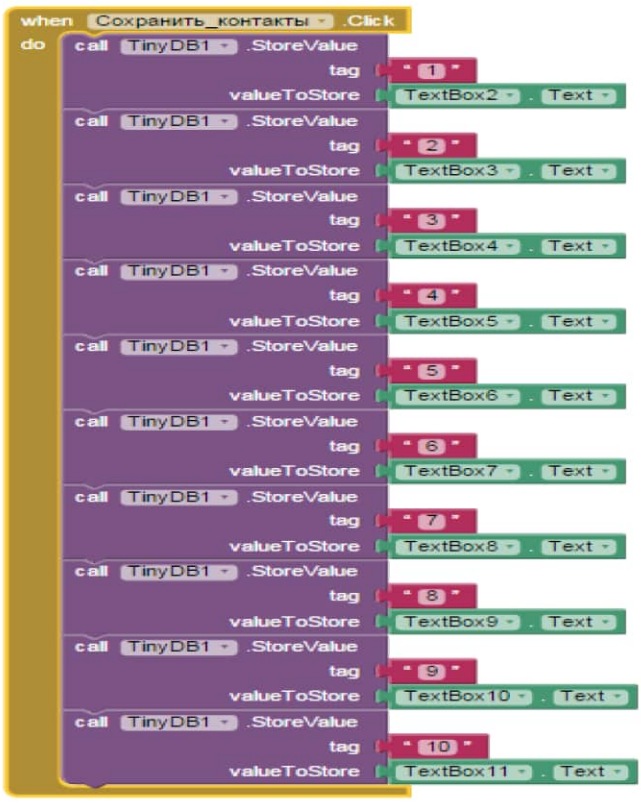

Рисунок 3.6 – Запис контактів у телефон

Вказуємо куди буде зберігатись результат розпізнавання мови (рис 3.7).

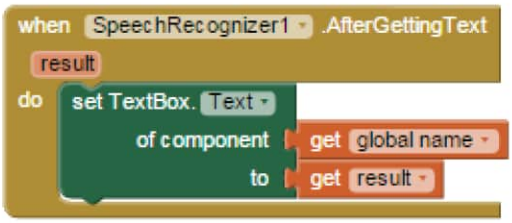

Рисунок 3.7 – Адреса зберігання результату розпізнавання

Описуємо дії кнопок для набору номера контакту(рис 3.8).

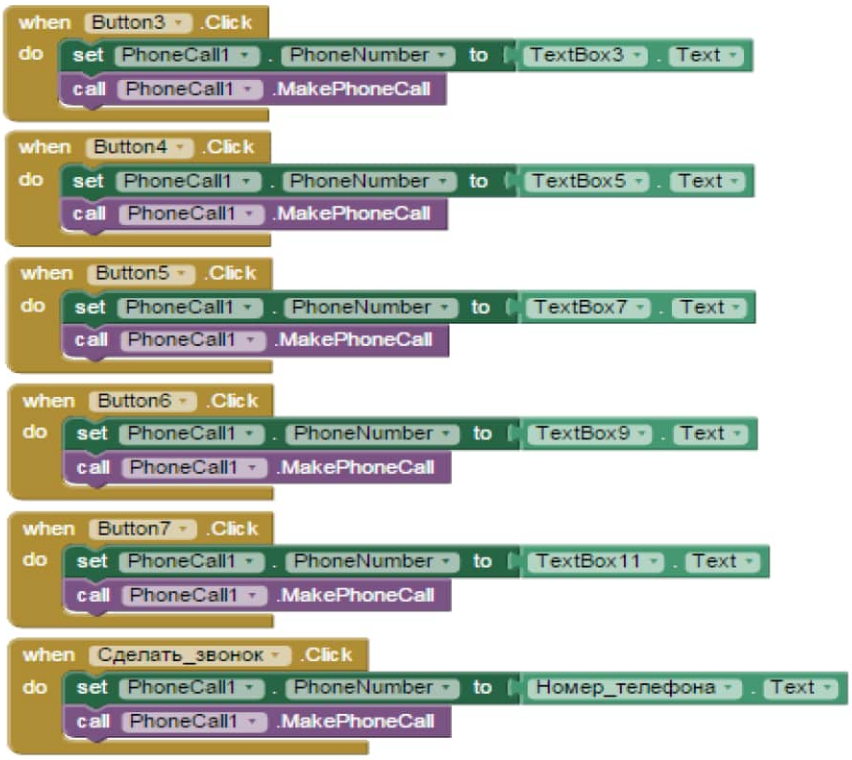

Рисунок 3.8 – Дії кнопок набору контактів

Описуємо функції голосового введення номеру та кнопки повернення в стартове вікно (рис 3.9).

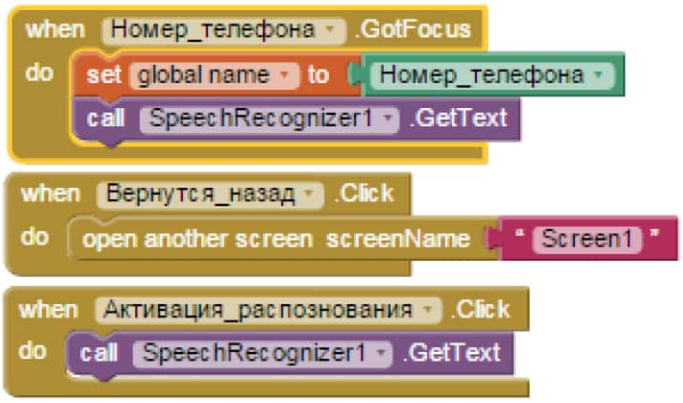

Рисунок 3.9 – Кнопки введення номеру та повертання в стартове вікно

При розробці вікна повідомлень була розроблена ініціалізація при його відкритті (рис. 3.10).

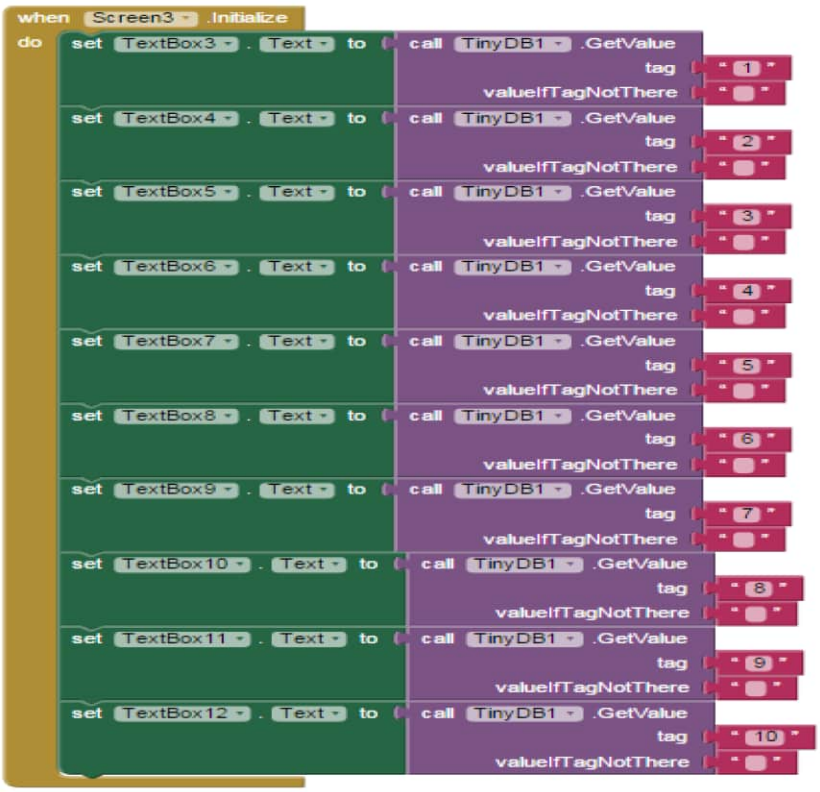

Рисунок 3.10 – Ініціалізація при відкритті відправки повідомлень

Для розробки запису контактів була розроблена система (рис. 3.11).

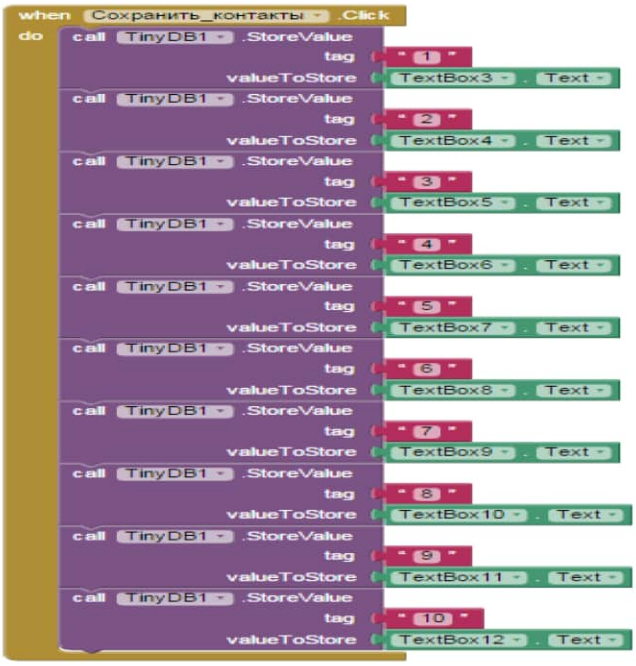

Рисунок 3.11 – Запис контактів у додаток

Описуємо відправку повідомлень контактові чи на номер телефону де була розроблена система відправки смс (рис. 3.12).

Рисунок 3.12 – Опис відправки повідомлень

Організувавши голосовий ввід в базу контактів додатку була добавлена глобальна перемінна (рис. 3.13).

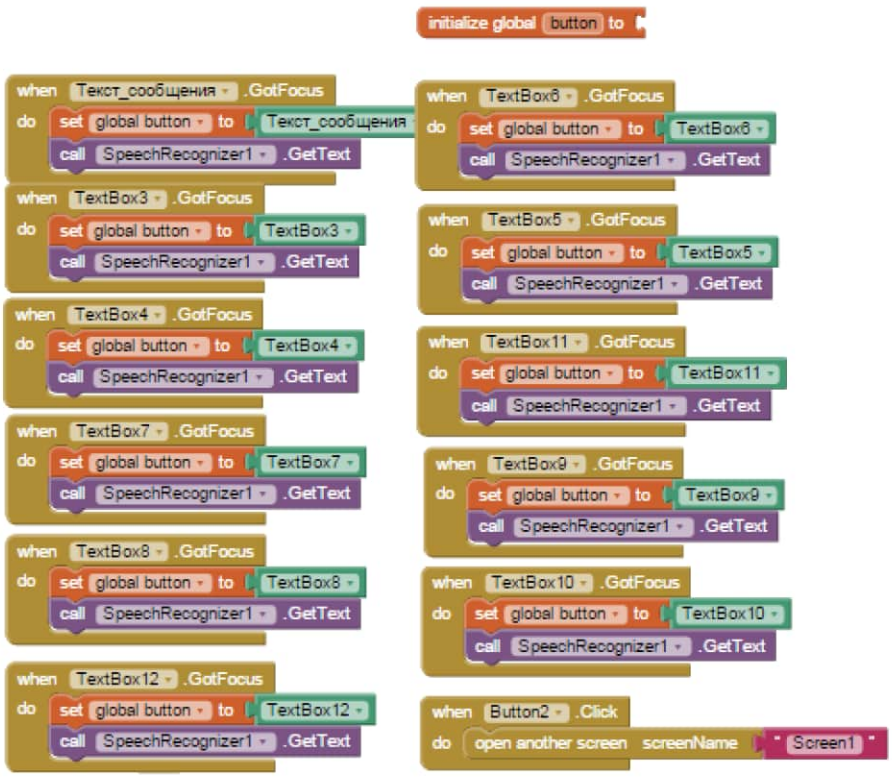

Рисунок 3.13 – Ініціалізація глобальної змінної та голосового введення в базу контактів додатку

Для активації розпізнавання мови була створена система з записом у глобальну змінну (рис 3.14).

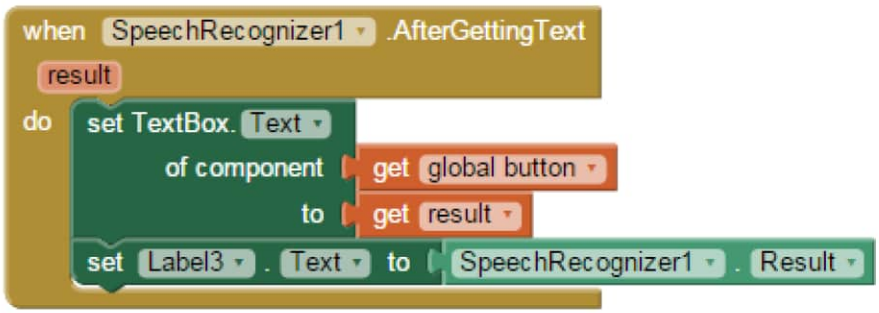

Рисунок 3. 14 – Організація результатів розпізнавання мови

3.3 Огляд розроблення графічного інтерфейсу

Було розроблено стартове вікно яке відображає програму з автоматичним включенням розпізнавання голосу при запуску (рис. 3.15).

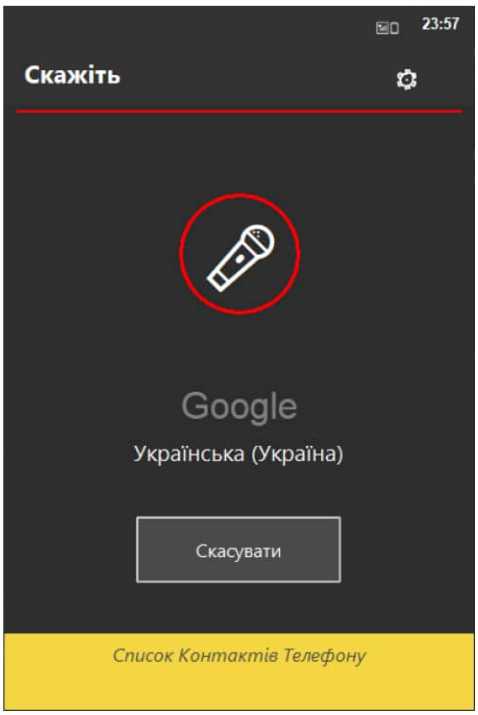

Рисунок 3.15 – Стартове вікно запуску програми

Також відображаються кнопки вибору переходів в розділи і включення розпізнавання мови (рис 3.16):

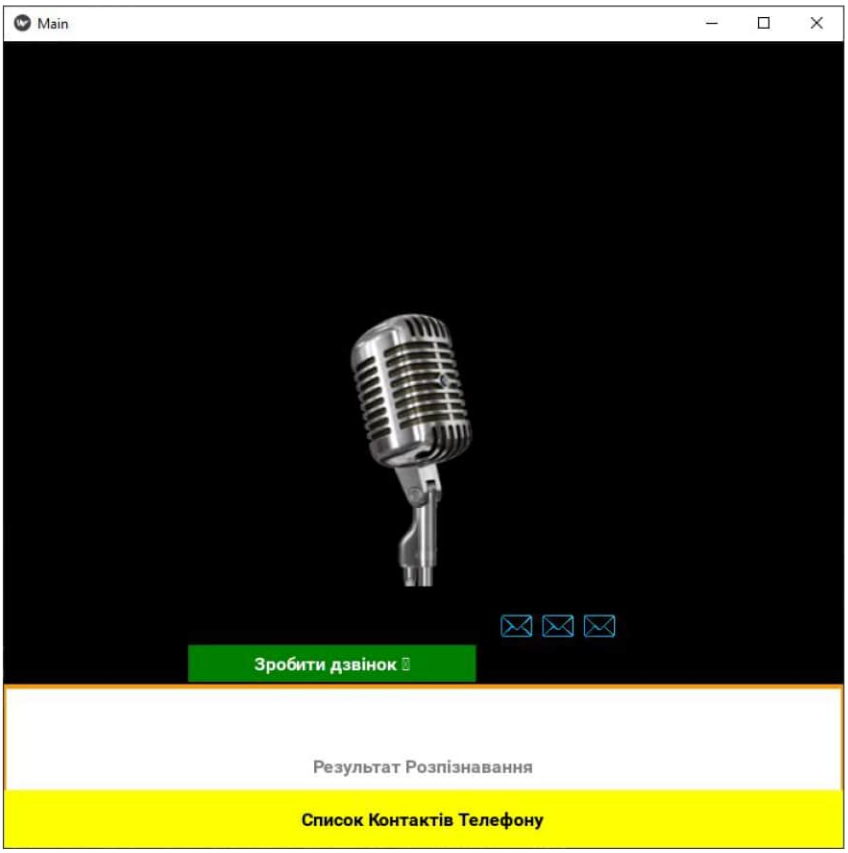

Рисунок 3.16 – Вікно вибору роботи з програмою

Перехід між функціями здійснюється шляхом розпізнавання голосу або натисканням на кнопки вибору дій. Якщо результат розпізнавання мови не є командою то він залишиться в результат розпізнавання і його можна буде скопіювати. У вікно здійснення дзвінків було розроблено систему набору номера відповідно до поставленого завдання. Список контактів на п’ять імен та телефонних номерів, які пов’язані з контактами які одночасно редагуються у вікні відправки повідомлень та здійснення дзвінків, та повернення до стартового вікна програми (рис. 3.17).

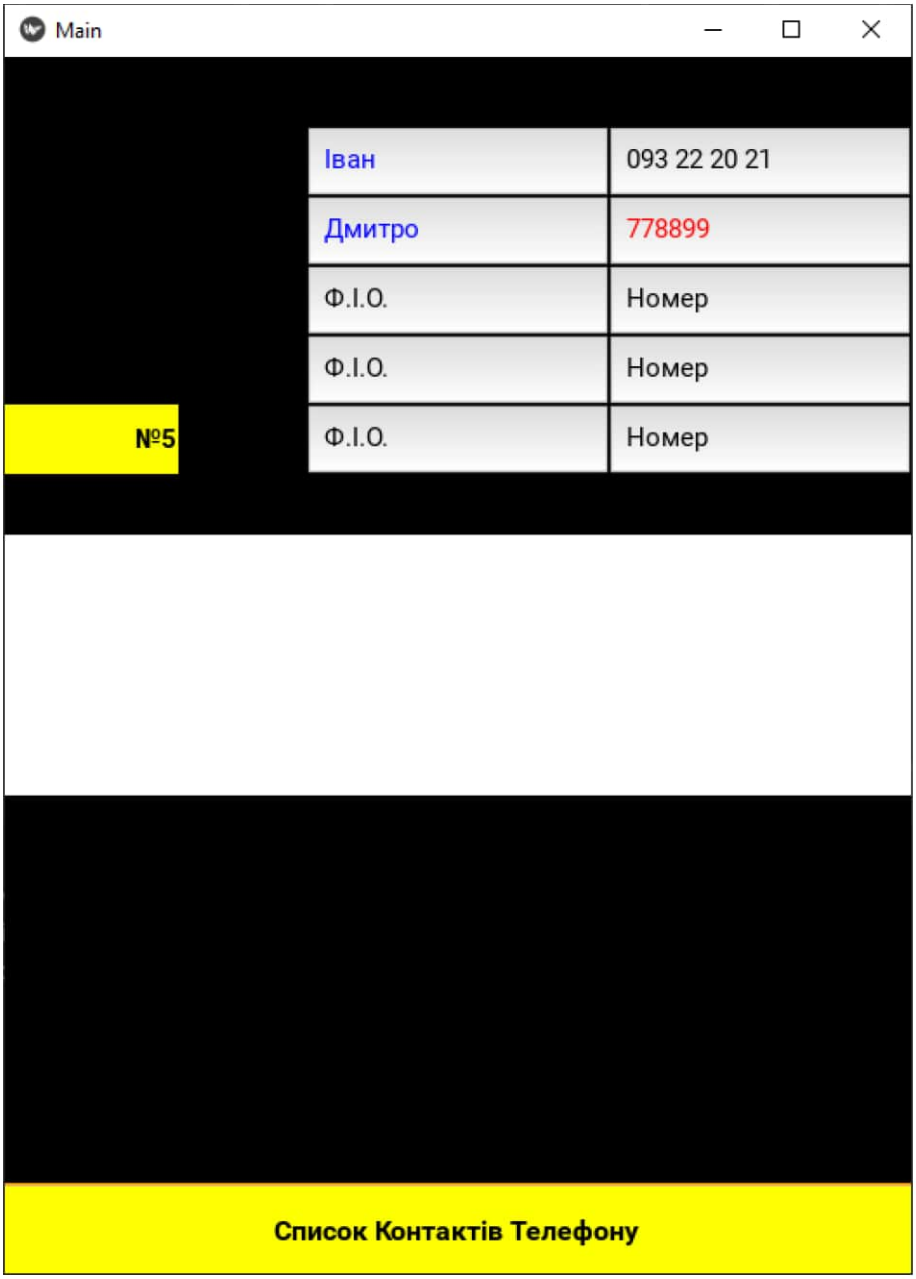

Рисунок 3.17 – Вікно здійснення дзвінків

У вікні написання повідомлень була розроблена система відправки повідомлень на введений номер або контакту в розробленій програмі, також список контактів на п’ять імен та телефонних номерів, які одночасно редагуються у вікні відправки повідомлень та створення дзвінків, повернення до стартового вікна програми (рис. 3.18).

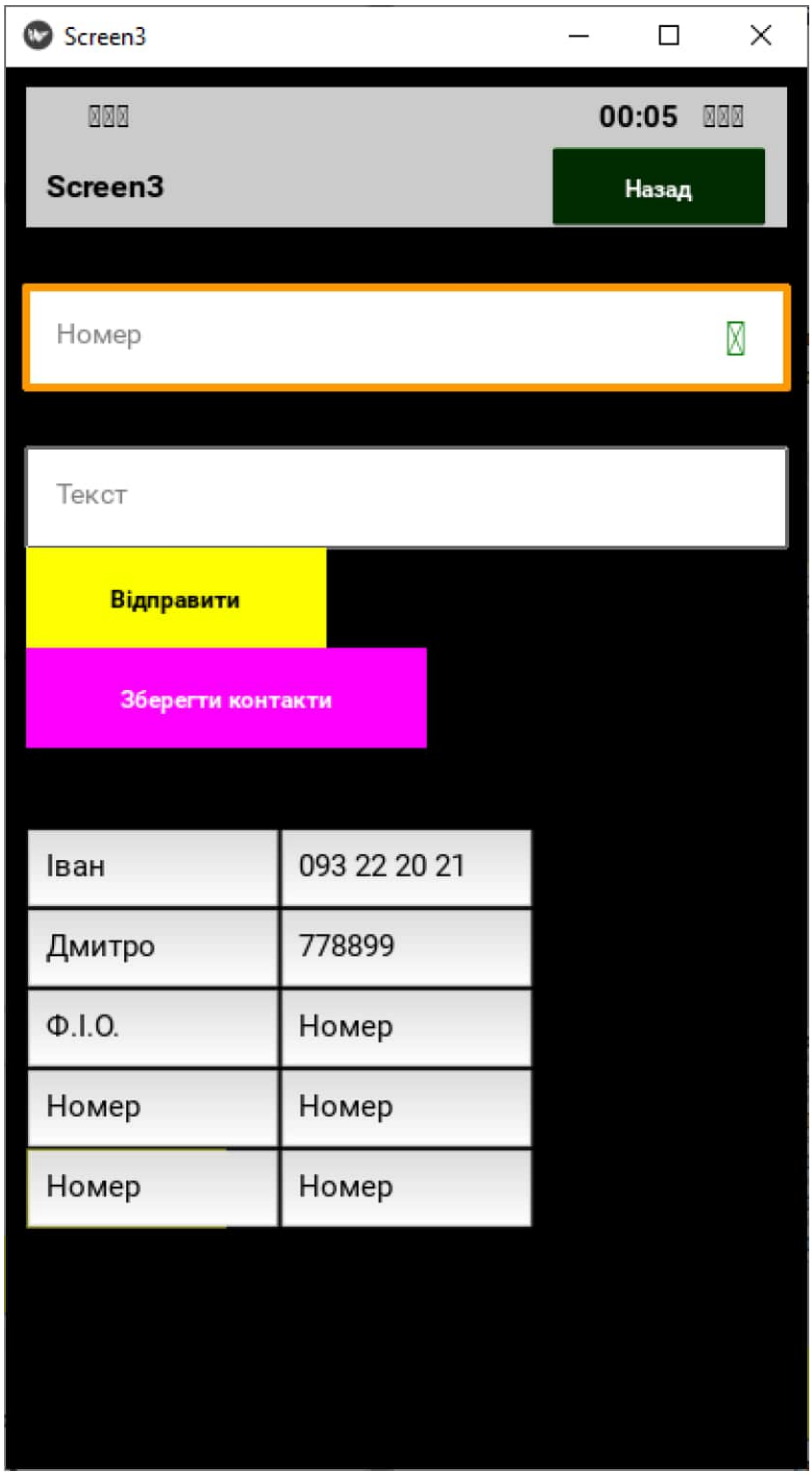

Рисунок 3.18 – Вікно відправки повідомлень

3.4 Тестування системи

Тестування розробленого додатка проводилося на різних версіях операційної системи, короткий список:

• Android-Google API 2.4.1;

• Android-Google API 3.6;

• Android-Google API 4.0 (Ice Cream)

• Android-Google API 4.0.4 (Ice Cream)

Тестування відбувалося на пристрої Android Google Nexus One Розробник Телефон 2.0, Amazon Kindel, а також при використанні емулятора Android-пристроїв поставляється в складі Android SDK. Інтерфейс емулятора представлений на рисунку нижче:

Рисунок 3.19 – Інтерфейс емулятора Android

Тестування проводилося для різних дозволів екранів:

• 400x240;

• 800x420;

• 800x480;

Також було проведено тестування розпізнавання голосу додатка, в ньому приймало участь дві жінки та два чоловіка. Кожен з них перевірив по 100 слів, результат представлений в таблиці 3.1.

Таблиця 3.1 – Тестування розпізнавання голосу

Тестувач	Кількість слів	Кількість розпізнаних слів	Кількість не-розпізнаних слів	відсоток розпізнавання
Перший чоловік	100	95	5	5
Другий чоловік	100	90	10	10
Перша жінка	100	93	7	7
Друга жінка	100	90	10	10

Загальний відсоток розпізнавання слів склав 92%, що відповідає нормативній документації від Google.

ВИСНОВКИ

У роботі виконано аналіз існуючих систем та технологій розпізнавання мови, також були розроблені вимоги до мовного інтерфейсу системи інтелектуальної телефонії. На основі зробленого аналізу було зроблено висновок, що доцільним є використання існуючого мережного сервісу Google, який вирішує задачі розпізнавання мови.

У зв’язку з поширенням великої кількості засобів під управлінням OS Android було визнано доцільним розробити мовний інтерфейс у вигляді програмного додатку працюючого під управлінням цієї операційної системи.

Були проаналізовані інструментальні системи створення програмних додатків під управління OS Android: Android Studio, App Iventor 2, BuildAnApp. За результатами аналізу для створення мовного інтерфейсу було вибрано App Inventor 2, оскільки в ньому є безкоштовне середовище для розробки Android додатків від Google Labs. Для створення додатків в App Inventor використовується візуальна мова програмування, додатки будуються об'єднанням стандартних елементів. Також є елементи, за допомогою яких можна відправити повідомлення, програти відео та багато інших. Було реалізовано програмні функції додатку, такі як:

- набір номеру,
- голосове редагування контактних даних контактів у додатку,
- голосове введення номеру та тексту SMS,
- голосове введення номеру для дзвінків,
- перехід функціями додатку завдяки голосовим командам.

По результату тестування було встановлено, що середня надійність розпізнавання команд становить 92% та встановлено, що надійність співпадає з результатами тестування від Google.

ПЕРЕЛІК ПОСИЛАНЬ

1. Jurafsky D., Martin J. H. Speech and Language Processing. 3rd ed. (draft) [Електронний ресурс]. – Stanford University, 2023. – Режим доступу: https://web.stanford.edu/~jurafsky/slp3/ – Дата звернення: 11.05.2025.
2. Young S., Gašić M., Thomson B., Williams J. D. POMDP-based statistical spoken dialog systems: A review // Proceedings of the IEEE. – 2013. – Vol. 101, No. 5. – P. 1160–1179. – DOI: 10.1109/JPROC.2012.2225812.
3. Google Cloud. Speech-to-Text Documentation [Електронний ресурс]. – Режим доступу: https://cloud.google.com/speech-to-text/docs – Дата звернення: 11.05.2025.
4. Android Developers. SpeechRecognizer API Reference [Електронний ресурс]. – Режим доступу: https://developer.android.com/reference/android/speech/SpeechRecognizer – Дата звернення: 10.05.2025.
5. Wang W., Acero A. Spoken Language Processing: A Guide to Theory, Algorithm and System Development. – Upper Saddle River: Prentice Hall, 2011. – 520 p.
6. Tang R., Lin J. Recent Advances in Deep Learning for Speech Recognition [Електронний ресурс] // arXiv preprint. – 2021. – arXiv:2101.07435. – Режим доступу: https://arxiv.org/abs/2101.07435 – Дата звернення: 01.05.2025.
7. Hinton G., Deng L., Yu D., Dahl G. E. et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition // IEEE Signal Processing Magazine. – 2012. – Vol. 29, No. 6. – P. 82–97. – DOI: 10.1109/MSP.2012.2205597.
8. Piczak K. J. Environmental Sound Classification with Convolutional Neural Networks // 2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP). – 2015. – P. 1–6. – DOI: 10.1109/MLSP.2015.7324337.

9. Vinyals O., Fortunato M., Jaitly N. Pointer Networks [Електронний ресурс] // arXiv preprint. – 2015. – arXiv:1506.03134. – Режим доступу: https://arxiv.org/abs/1506.03134 – Дата звернення: 11.05.2025.
10. Këpuska V., Bohouta G. Comparing Speech Recognition Systems (Microsoft API, Google API and CMU Sphinx) // International Journal of Engineering Research and Applications. – 2017. – Vol. 7, No. 3. – P. 20–24. – DOI: 10.9790/9622-0703022024.
11. Mitchell T. Machine Learning. – New York: McGraw-Hill, 1997. – 414 p.
12. Purcell D. Mastering App Inventor: Create no-code Android apps with ease [Електронний ресурс]. – Leanpub, 2020. – Режим доступу: https://leanpub.com/masteringappinventor – Дата звернення: 04.06.2025.

ДОДАТОК А. ФРАГМЕНТИ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Фрагмент лістингу maim.py

```
from kivy.app import App
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.textinput import TextInput
from kivy.uix.button import Button
from kivy.uix.label import Label
from kivy.uix.gridlayout import GridLayout
from kivy.uix.scrollview import ScrollView
from kivy.properties import ObjectProperty, StringProperty
from kivy.core.window import Window
from kivy.metrics import dp
from kivy.graphics import Color, Rectangle
from kivy.lang import Builder
from kivy.uix.screenmanager import ScreenManager, Screen
from kivy.clock import Clock

print("--- Завантаження KV-файлів ---")
try:
    Builder.load_file('screen1layout.kv')
    print("screen1layout.kv завантажено успішно.")
    Builder.load_file('screen2layout.kv')
    print("screen2layout.kv завантажено успішно.")
    Builder.load_file('screen3layout.kv')
    print("screen3layout.kv завантажено успішно.")
except Exception as e:
    print(f"Помилка при завантаженні KV-файлу: {e}")
    print(
        "Переконайтеся, що файли KV знаходяться в тій же директорії, що й main.py,
та не містять синтаксичних помилок.")
    exit()

print("--- Ініціалізація віджетів ---")


class WindowManager(ScreenManager):
    pass


class Screen1Layout(Screen):
    recognition_result_label = ObjectProperty(None)

    def go_to_screen2(self):
        print("Screen1: Перехід до Screen2 (Зробити дзвінок).")
        self.manager.current = 'screen2'

    def go_to_screen3(self):
        print("Screen1: Перехід до Screen3 (Відправити повідомлення).")
        self.manager.current = 'screen3'
```

```
    def start_speech_recognition(self):
        print("Screen1: Початок розпізнавання мови...")
        if self.recognition_result_label:
            self.recognition_result_label.text = "Результат: Привіт, Kivy!"
            print(f"Screen1: Оновлено 'Результат Розпізнавання' на:
{self.recognition_result_label.text}")
        else:
            print("Screen1: Помилка: recognition_result_label не знайдено (можливо, не
прив'язано в KV).")

    def open_phone_contacts(self):
        print("Screen1: Відкриття списку контактів телефону (Перехід до Screen3).")
        self.manager.current = 'screen3'


class Screen2Layout(Screen):
    def go_back(self):
        print("Screen2: Повернення до попереднього екрану (Screen1).")
        self.manager.current = 'screen1'


class Screen3Layout(Screen):
    phone_input = ObjectProperty(None)
    message_input = ObjectProperty(None)
    contacts_grid = ObjectProperty(None)

    microphone_image_source = StringProperty('images/microphone.png')

    def __init__(self, **kwargs):
        super(Screen3Layout, self).__init__(**kwargs)
        print(f"Screen3: __init__ викликано. contacts_grid на цьому етапі:
{self.contacts_grid}")

    def on_kv_post(self, base_widget):
        print(f"Screen3: on_kv_post викликано.")
        print(f"Screen3: Стан self.ids: {self.ids}")

        if 'contacts_grid_id' in self.ids:
            self.contacts_grid = self.ids.contacts_grid_id
            print(f"Screen3: 'contacts_grid_id' знайдено в self.ids. Присвоєно:
{self.contacts_grid}")
            self.populate_contacts_table()
        else:
            print("Критична помилка Screen3: ID 'contacts_grid_id' не знайдено в
self.ids!")
            print("Це вказує на проблему у screen3layout.kv: неправильний ID або
синтаксична помилка.")

    def create_background_label_wrapper(self, text, bg_color, text_color=(0, 0, 0, 1),
bold=False, size_hint_y=None,
                                        height=None, halign='center', valign='middle'):
        box = BoxLayout(size_hint_y=size_hint_y, height=height)
```

```
        with box.canvas.before:
            Color(*bg_color)
            Rectangle(pos=box.pos, size=box.size)

        label = Label(text=text, color=text_color, bold=bold, halign=halign, valign=valign)
        label.bind(size=label.setter('text_size'))
        box.add_widget(label)
        return box

    def populate_contacts_table(self):
        print(f"Screen3: populate_contacts_table викликано. contacts_grid:
{self.contacts_grid}")
        if self.contacts_grid:
            self.contacts_grid.clear_widgets()
            print("Screen3: contacts_grid очищено.")

            # Заголовки таблиці
            self.contacts_grid.add_widget(
                self.create_background_label_wrapper("№", (1, 1, 0, 1), bold=True,
size_hint_y=None, height=dp(30)))
            self.contacts_grid.add_widget(
                self.create_background_label_wrapper("Ф.І.О.", (1, 1, 1, 1), bold=True,
size_hint_y=None,
                                                     height=dp(30)))
            self.contacts_grid.add_widget(
                self.create_background_label_wrapper("Номер", (1, 1, 1, 1), bold=True,
size_hint_y=None, height=dp(30)))
            print("Screen3: Додано заголовки таблиці.")

            # Дані контактів (як на вашому скріншоті)
            initial_contacts = [
                {"name": "Іван", "phone": "093 22 20 21"},
                {"name": "Дмитро", "phone": "778899"},
                {"name": "Ф.І.О.", "phone": "Номер"},
                {"name": "Ф.І.О.", "phone": "Номер"},
                {"name": "Ф.І.О.", "phone": "Номер"},
            ]

            for i, contact in enumerate(initial_contacts):
                num_label_box = self.create_background_label_wrapper(
                    f"№{i + 1}",
                    (1, 1, 0, 1),
                    bold=True,
                    size_hint_y=None,  # Виправлено: тепер це іменований аргумент
                    height=dp(40)
                )
                self.contacts_grid.add_widget(num_label_box)

                name_input = TextInput(text=contact["name"], multiline=False,
size_hint_y=None, height=dp(40),
                                       foreground_color=(0, 0, 0, 1), background_color=(1, 1, 1, 1),
```

```
padding=dp(10))
            if contact["name"] == "Іван" or contact["name"] == "Дмитро":
                name_input.foreground_color = (0, 0, 1, 1)  # Синій
            self.contacts_grid.add_widget(name_input)

            phone_input = TextInput(text=contact["phone"], multiline=False,
size_hint_y=None, height=dp(40),
                                    foreground_color=(0, 0, 0, 1), background_color=(1, 1, 1, 1),
padding=dp(10))
            if contact["phone"] == "778899":
                phone_input.foreground_color = (1, 0, 0, 1)  # Червоний
            self.contacts_grid.add_widget(phone_input)
        print("Screen3: Додано початкові контакти до таблиці.")
    else:
        print(
            "Критична помилка Screen3: contacts_grid все ще None у
populate_contacts_table. Таблиця не буде заповнена.")

def go_back(self):
    print("Screen3: Натиснуто 'Назад'. Повернення до Screen1.")
    if self.manager:
        self.manager.current = 'screen1'

def send_message(self):
    if not self.phone_input or not self.message_input:
        print("Screen3: Віджети phone_input або message_input не ініціалізовано.")
        return

    phone_num = self.phone_input.text
    message_txt = self.message_input.text

    if phone_num == "Номер телефону" or not phone_num.strip():
        print("Screen3: Будь ласка, введіть номер телефону.")
        return
    if message_txt == "Текст" or not message_txt.strip():
        print("Screen3: Будь ласка, введіть текст повідомлення.")
        return

    print(f"Screen3: Повідомлення надіслано на номер
{phone_num}:\n'{message_txt}'")

def save_contacts(self):
    print("Screen3: Функціонал збереження контактів буде реалізовано тут.")
    print("Screen3: Контакти збережено (логіка збереження не реалізована).")

def open_phone_contacts(self):
    print("Screen3: Натиснуто 'Список Контактів Телефону'.")
    print("Screen3: Відкриття інтерфейсу для додавання/перегляду нових
контактів (поки імітація).")


class MainApp(App):
```

```
    def build(self):
        print("MainApp: build() викликано.")
        Window.size = (dp(400), dp(700))
        Window.title = "Мобільний Асистент"

        sm = WindowManager()
        sm.add_widget(Screen1Layout(name='screen1'))
        sm.add_widget(Screen2Layout(name='screen2'))
        sm.add_widget(Screen3Layout(name='screen3'))

        sm.current = 'screen1'  # Стартовий екран зазвичай Screen1
        print(f"MainApp: Поточний екран встановлено на '{sm.current}'.")
        return sm


if __name__ == "__main__":
    print("--- Запуск програми Kivy ---")
    MainApp().run()
    print("--- Програма Kivy завершила роботу ---")
```

screen1layout.kv

```
#:kivy 2.1.0

<Screen1Layout>:
    name: 'screen1'
    # Змінюємо кореневий макет на RelativeLayout для кращого контролю позиціонування
    # RelativeLayout дозволяє розміщувати віджети відносно батьківського за допомогою pos_hint
    RelativeLayout:
        size_hint: (1, 1) # Займає весь екран
        pos_hint: {'x': 0, 'y': 0}
        background_color: (1,1,1,1) # White background для цього віджету
        canvas.before:
            Color:
                rgba: self.background_color
            Rectangle:
                pos: self.pos
                size: self.size

        recognition_result_label: recognition_result_label_id

        # --- 1. Верхня панель (імітація рядка стану телефону) ---
        BoxLayout:
            size_hint: (1, None)
            height: dp(30)
            pos_hint: {'top': 1} # Прив'язуємо до верху батьківського RelativeLayout
            padding: dp(10), 0
            canvas.before:
                Color:
                    rgba: (0.8, 0.8, 0.8, 1) # lightgray
```

```
            Rectangle:
                pos: self.pos
                size: self.size
        Label:
            text: "🔍✉🖼"
            color: (0,0,0,1)
            size_hint_x: 0.3
        Widget: # Розпірка
        Label:
            text: "00:05"
            color: (0,0,0,1)
            bold: True
            size_hint_x: 0.2
        Label:
            text: "□🔔🔋"
            color: (0,0,0,1)
            size_hint_x: 0.2

    # --- 2. Заголовок "Screen1" ---
    BoxLayout:
        size_hint: (1, None)
        height: dp(40)
        pos_hint: {'top': 1 - dp(30)/root.height if root.height > 0 else 1} # Під верхньою
панеллю
        padding: dp(10), 0
        canvas.before:
            Color:
                rgba: (0.8, 0.8, 0.8, 1) # lightgray
            Rectangle:
                pos: self.pos
                size: self.size
        Label:
            text: "Screen1"
            font_size: '16sp'
            bold: True
            color: (0,0,0,1)
            halign: 'left'
            valign: 'center'
            text_size: self.size

    # --- 3. Кнопки "Зробити дзвінок" та "Надіслати повідомлення" ---
    BoxLayout:
        size_hint: (1, None)
        height: dp(100)
        # Розміщуємо нижче заголовка Screen1.
        # 1 - (висота верхньої панелі + висота заголовка) / загальна висота
        pos_hint: {'top': 1 - (dp(30) + dp(40))/root.height if root.height > 0 else 1}
        padding: dp(5)
        spacing: dp(20)

        BoxLayout: # Кнопка "Зробити дзвінок"
            orientation: 'vertical'
```

```
            size_hint_x: 0.5
            Button:
                text: "Зробити дзвінок 📞"
                background_normal: ''
                background_color: (0, 0.5, 0, 1) # Green
                color: (1,1,1,1)
                font_size: '16sp'
                bold: True
                on_release: root.go_to_screen2()
                canvas.before:
                    Color:
                        rgba: (0, 0.5, 0, 1) # Green oval
                    Ellipse:
                        pos: self.pos
                        size: self.size

        BoxLayout: # Секція "Надіслати повідомлення"
            orientation: 'vertical'
            size_hint_x: 0.5
            Label:
                text: "Надіслати повідомлення"
                font_size: '14sp'
                color: (0,0,0,1)
                halign: 'center'
                valign: 'middle'
                text_size: self.size
                size_hint_y: None
                height: dp(40)
            BoxLayout: # Для іконок конвертів
                size_hint_y: None
                height: dp(60)
                spacing: dp(5)
                # Ці іконки викликають помилку "Not found", перевірте
images/envelope_icon.png
                Image:
                    source: 'images/envelope_icon.png'
                    allow_stretch: True
                    keep_ratio: True
                Image:
                    source: 'images/envelope_icon.png'
                    allow_stretch: True
                    keep_ratio: True
                Image:
                    source: 'images/envelope_icon.png'
                    allow_stretch: True
                    keep_ratio: True
                Button: # Невидима кнопка, яка переводить на screen3
                    text: ""
                    background_normal: ''
                    background_color: (0,0,0,0)
                    on_release: root.go_to_screen3()
                    size_hint: (1,1)
```

```
                opacity: 0.0


    # --- 4. Результат розпізнавання (над мікрофоном, якщо він буде знизу) ---
    BoxLayout:
        size_hint: (1, None)
        height: dp(60)
        # Розміщуємо над кнопкою "Натисніть для розпізнавання мови"
        # 1 - (висота нижнього блоку + висота кнопки розпізнавання + висота цього
блоку) / загальна висота
        # Це буде динамично обчислюватися, але поки що розташуємо його над
кнопкою
        # Або, якщо мікрофон буде знизу, то цей блок буде вище за нього.
        # Для RELATIVELAYOUT, розташуємо його відносно низу екрана.
        pos_hint: {'y': (dp(50) + dp(50)) / root.height if root.height > 0 else 0} # Над
нижньою кнопкою і кнопкою розпізнавання
        padding: dp(5)
        canvas.before:
            Color:
                rgba: (1, 0.6, 0, 1) # Orange border
            Line:
                width: dp(2)
                rectangle: self.x, self.y, self.width, self.height
            Color:
                rgba: (1,1,1,1) # White background
            Rectangle:
                pos: self.x + dp(2), self.y + dp(2)
                size: self.width - dp(4), self.height - dp(4)
        Label:
            id: recognition_result_label_id
            text: "Результат Розпізнавання"
            font_size: '16sp'
            color: (0.5,0.5,0.5,1) # Grey
            bold: True
            halign: 'center'
            valign: 'middle'
            text_size: self.size

    # --- 5. Кнопка "Натисніть для розпізнавання" ---
    Button:
        text: "Натисніть для розпізнавання мови"
        size_hint: (1, None)
        height: dp(50)
        # Розміщуємо над "Списком Контактів Телефону"
        pos_hint: {'y': dp(50)/root.height if root.height > 0 else 0} # Над нижньою
кнопкою
        background_normal: ''
        background_color: (0, 0.5, 0, 1) # Green
        color: (1,1,1,1)
        font_size: '14sp'
        bold: True
        on_release: root.start_speech_recognition()
```

```
        # --- 6. Нижній "Список Контактів Телефону" ---
        BoxLayout:
            size_hint: (1, None)
            height: dp(50)
            pos_hint: {'y': 0} # Прив'язуємо до низу батьківського RelativeLayout
            canvas.before:
                Color:
                    rgba: (1,1,0,1) # Yellow
                Rectangle:
                    pos: self.pos
                    size: self.size
            Button:
                text: "Список Контактів Телефону"
                color: (0,0,0,1)
                font_size: '16sp'
                bold: True
                background_normal: ''
                background_color: (0,0,0,0)
                on_release: root.open_phone_contacts()

        # --- 7. Іконка мікрофона (окремо в центрі, не перекриваючи) ---
        Image:
            source: 'images/microphone.png'
            size_hint: (0.4, 0.4) # Дуже малий розмір, щоб перевірити
            # pos_hint: {'center_x': 0.5, 'center_y': 0.5} # Центруємо відносно
RelativeLayout

            # Розрахуємо позицію мікрофона, щоб він був між елементами
            # Загальна висота: root.height
            # Висота верхніх елементів (разом з відступами): dp(30) + dp(40) + dp(100) =
dp(170)
            # Висота нижніх елементів (разом з відступами): dp(60) + dp(50) + dp(50) =
dp(160)
            # Залишається вільного місця: root.height - dp(170) - dp(160) = root.height -
dp(330)
            # Мікрофон буде по центру цього вільного місця.
            # Його нижня межа буде: (висота нижніх елементів) + (вільний простір / 2) -
(висота мікрофона / 2)
            # Або простіше: 'center_y': 0.5
            pos_hint: {'center_x': 0.5, 'center_y': 0.5}
```